

Project Final Report

Intelligent Plant Monitoring System

April 25, 2007

Ya-Shian Li, yashian@gmail.com

Project Abstract

A networked house plant monitoring system was developed to notify a user over the Internet when the current environmental condition is outside of the plant's optimal tolerances. When the conditions for the plant type fall outside of the specified tolerances, the system will alert plant caretakers to water their plants, adjust the temperature/humidity, and light intensity according to specification entered by the user. The monitoring system is based on Systronix's TStik, an embedded Java system, running a web server with a browser interface to allow the user to enter the precise moisture, temperature and humidity range preferred by a specific type of plant. A soil moisture, temperature, humidity, and solar intensity sensors are utilized to detect the plant's environmental conditions. A one-wire hub is utilized to connect multiple sensors to the TStik in a star topology and to allow for future ease of extensibility. During data collection, a Java servlet polls each hub interface in a round-robin at a frequency specified by the user, which can range from once every 30 seconds to once a day. Sensor data are periodically refreshed on the user's browser interface. When the sensor detects a parameter which is close to sub-optimal or outside of the user's desired tolerance window for a specific plant, the device will generate the proper alert message.

Status

The basic concepts of the plant monitoring system are in place. The TStik was set up as a web server using Tynamo. The web server is used to provide the GUI for user plant configuration and to deploy the plant monitoring servlet which collects and displays the sensor data, and using simple logic, to provide summary notifications when non-optimal conditions are reached. Additionally, when the system detects conditions that are *almost* sub-optimal, a warning is issued to the user.

Due to some challenges in learning the TINI API and the tools of embedded Java, I did not have a chance to implement all the features included in the initial proposal. It took some tweaking to understand Ant and TiniAnt, but afterwards the Ant file for building the *webserver.tini* file and for deploying the file from the development laptop to the TStik, was elegantly simple. The temperature, humidity, soil moisture, and solar sensors are all working. I was able to readily calibrate the soil moisture sensor, but did not get a chance to calibrate the remaining sensors.

The most challenging part of the project was trying to obtain database connectivity for the TINI. A MySQL server was set up on a laptop, with a simple table to store the plant type and their respective tolerance levels. However, I had overlooked the fact that the TINI API does not encompass all of the capabilities of Java, something which made sense later on, since JDBC connectivity is not especially lightweight. Therefore I looked into other methods for connecting to a database. An API from Hangar 5, called remote JDBC offered a solution, but was not simple. Eventually after properly specifying all the dependencies in Ant, I was able to get the project to compile, only to find build errors in the subsequent build command. *StringBuilder* was not found, which is most likely due to Java compatibility issues. At the very end, to have a simple way to demonstrate what I would have done with the database, I ended up hardcoding the tolerances and plant types in JavaScript. A more dynamic option would have been to store the information in text files and to allow the user to specify a plant type and store the respective tolerance information inside.

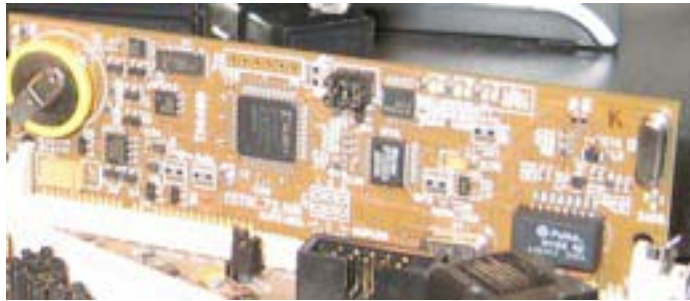
Specification

Hardware

The Systronix TStik Development Kit was used as the web server and data collection system. The parts of the development kit used in this project included the following:

- **TINI TStik Version 2**

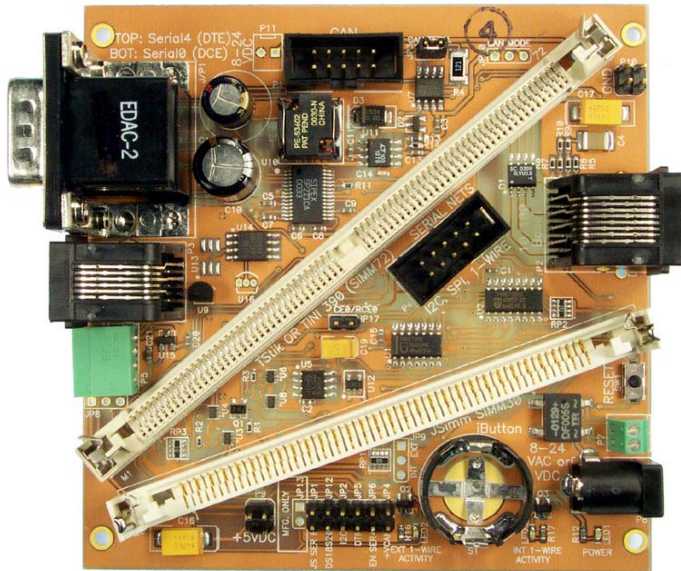
The TStik is a simple system with the Dallas TINI microcontroller (DS80C400) at 30 (15 x 2) MHz clock, 1 MB of SRAM, and 2 MB of flash. It also supports 10/100 BaseT Ethernet. The TStik includes a real-time clock, backed up by battery, which can be set by the user and used for time stamping after each data collection of the environmental conditions.



TStik Version 2 on a TILT board

- **TILT400.pro Socket Board**

The socket board provides the hardware interface between the TINI microcontroller and the various sockets including an RJ45 (for Ethernet-based connections) and RJ12 (Serial 1) jacks (which was used for one-wire, but can be used for UART serial interfaces with the latest version of TStik). TILT also includes the RS232 (DB9) ports which was instrumental for initializing the TINI microcontroller and installing the TINI OS and *Slush* interface application using the Microcontroller ToolKit (MTK).



TILT400.Pro Board

The sensors were acquired from HobbyBoards as pre-built modules. A 1-wire Temperature/Humidity/Solar (THS) board, moisture meter board, and gypsum soil sensor were used.

Sensors:

Temperature/Humidity/Solar



Humidity (HIH3610):

A 1-wire RH sensor that measures 0-100% RH with an accuracy of +/- 2%.



Temperature (DS18S20):



It is a 1-wire digital thermometer that guarantees a 0.5C accuracy from -10C to +85C, and can measure between -55C to 125C with 9-bit data resolution.

Solar (CLD140 Photodiode):

It is a Clairex CLD140 Photodiode with peak detection at 860 nm and an acceptance angle of 140 degrees.



Temperature, Time and Voltage Reader (DS2438)



The DS2438 is designed to be a battery monitor, but has the features necessary to convert the values from the humidity sensor and the photodiode (solar sensor) into a 1-wire format.

Soil Moisture Sensor

Gypsum Soil Sensor



The gypsum soil sensor works by detecting the change of resistance in the block as the gypsum absorbs the surrounding moisture. The more moisture surrounding the sensor, the less resistance there is in the gypsum block which generates more current and therefore a higher reading.

Moisture Meter

The moisture meter functions by measuring the resistance of the attached soil sensor and sends the measurement data through the 1-wire interface. The sensor's resistance reduces as the moisture level increases.



•*High Precision Li+ Battery Monitor (DS2760)*: This component measures temperature, voltage and current. Through current measurement, it can sense the resistance from the sensor connected to the sensor input of the board, and therefore acts as a data acquisition device.

One-Wire Hub



A 1-wire hub, also from HobbyBoards, provided the mechanism to connect multiple 1-wire devices in a star topology for simpler configuration and addition of new sensors. The 1-wire hub also served as the power source for the THS and moisture meter sensor boards.

Connectors were also acquired to connect the sensors to the system. An RJ12 - RJ45 connector and RJ45-RJ45 1-wire connector were also purchased to connect between the

1-wire hub and the TILT board. Lastly, a spare Ethernet cable was split such that the bare wires on one end allowed the moisture sensor to be integrated into the 1-wire network of devices. For networking at home, two options were configured. One method was to use a LAN hub to connect the TStik, and for demonstration purposes, a crossover cable was used to connect the TStik system to the laptop which servers as the client.

Software

To interface with the hardware a software module was written for each of the major sensors, namely humidity, temperature, light, and moisture meter. Additionally a 1-wire device API was written to allow easy access to 1-wire data such as the serial numbers needed for the TStik to locate the 1-wire device to retrieve data from. Since the sensors were connected via the 1-wire hub, an One-Wire Hub API was also necessary to activate and deactivate the respective channels necessary to read from a specific sensor board.

- Temperature Sensor API:
 - Read temperature from the sensor.
 - Initialize the sensor.
 - Convert the temperature from Celsius to Fahrenheit.

- Humidity Sensor API
 - Initialize sensor.
 - Read temperature from the sensor.
 - Read 1-wire sensor RH data from the DS2483 connected to the humidity sensor.
 - Convert sensor RH to true RH based on temperature and sensor RH.

- Light Sensor API
 - Initialize sensor.
 - Read 1-wire light intensity data from the DS2483 connected to the photodiode.

- Soil Moisture Sensor API
 - Initialize sensor.
 - Read 1-wire soil moisture level data from the DS2760 connected to the gypsum soil moisture sensor.

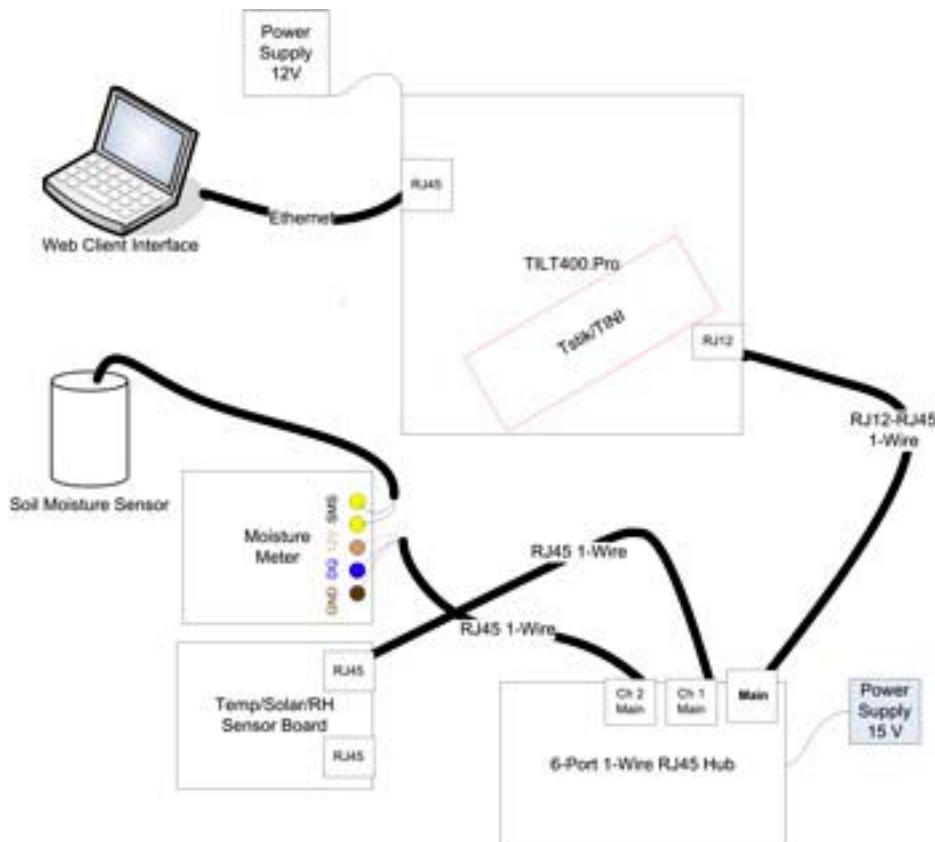
For the main application, the Tynamo Web Server was used with properties modified to incorporate the Plant Monitor servlet and the configuration menu file. The Plant Monitor servlet served as the main application to process the user's plant specifications and to collect data from the respective sensors via the 1-wire hub. An HTML file was written to

create a GUI that dynamically fills in the tolerance windows depending on the type of plant selected.

- Plant Monitor Servlet
 - Initialize all sensors and one wire hub.
 - Processes the form data.
 - Collects data at user-specified intervals, time stamp each data collection group, and continuously refreshes the data to be displayed to the user.
 - “Intelligence” ... Built-in logic to notify user when plant conditions are not optimal or near not optimal.
 - A color-coded warning is issued when sensor data is over or below optimal levels.
 - A warning is issued when sensor data nears sub-optimal levels ... e.g. within 10%.
- Plant Monitor HTML
 - GUI for configuring the plant type, data collection frequency and tolerance windows.
 - JavaScript enables tolerance values to be automatically filled based on plant type selected.
 - Provides form data to Plant Monitor servlet.

Implementation and Construction

Hardware



Hardware Diagram



Initial Hardware Integration

The hardware components were readily integrated. The TStik fits into the TILT socket, the Ethernet cable is plugged into the the 10/100 Ethernet Interface, and the power adapter was plugged in. The RJ12-RJ45 is connected to the RJ12 (serial 1) port on the TILT board, and the RJ45 end is connected to the one-wire 6-port hub. The THS sensor board is connected using RJ45 one-wire connectors between the hub channel 1 and the THS board.

The soil moisture sensor device required connection to the one-wire hub via screw terminals. Therefore a CAT5 Ethernet cable was split revealing the wires. The pertinent wires are the brown, light brown and blue. The wires were stripped, and the brown was connected to the ground terminal. Similarly, the light brown was connected to the 12V terminal and the blue was connected to the data terminal.

To prepare the gypsum soil moisture sensor, the gypsum device was soaked in water overnight. The next day, the copper wire was stripped and attached to the moisture meter's sensor terminals. Typically the gypsum device would need to be buried far enough down to be at the same level as the roots of the plant. However for this experiment, I only barely covered



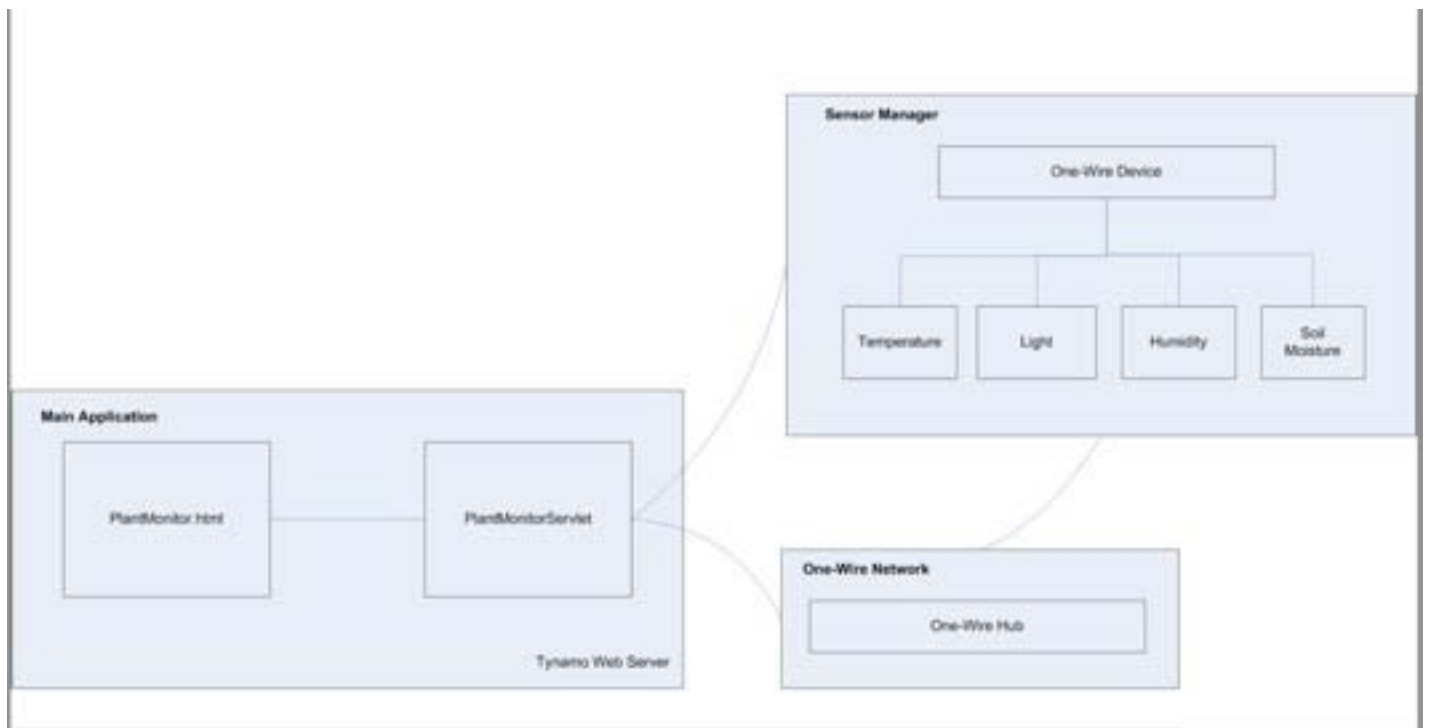


the device for demonstration purposes.

The soil moisture sensor also needed to be calibrated. The moisture meter offset was determined by removing the copper wires from the sensor and reading the moisture meter. This simulated a completely dry environment.

The RJ45 end of the moisture sensor was connected to Channel 2 of the one-wire hub. The hub itself is connected to 15V of power in order to supply the moisture meter and THS board with the needed power supply.

Software



Software Diagram

The software was developed in a modular process. To test one-wire access and fetch to ROM ID of each sensor, a one-wire lister test program was developed. This provided the ROM IDs for implementing the system to allow the system to call a specific sensor.

The next step consisted of implementing the one-wire sensors on the THS board. A command-line program was developed to test the functionality. The one-wire sensors each required dependencies on its specific family's API. For example the temperature sensor's family ID was 10, and therefore it was necessary to use the OneWireContainer10. A one-wire hub interface was added to be able to connect to multiple one-wire sensor boards in a star topology.

Next, the Tynamo web server was set up to implement servlets. The Plant Monitor Servlet was developed to test the ability to fetch sensor information and to display it on the web interface. Once this was found to be successful, the GUI interface was developed to specify the tolerance level. The servlet fetches the information through the POST command and is able to determine if the sensor readings are out of the optimal tolerance ranges specified by the user for a specific type of plant. The frequency of data collection is also specified by the user. It also determines whether the plant is within a percentage before being outside of the optimal tolerance and can therefore warn the user.

The database was not able to be configured in time for the demonstration. Therefore, JavaScript was added to "store" the tolerance levels and to dynamically fill in the fields based on the type of plant selected.



The screenshot shows a web interface titled "Plant Monitor Configuration Menu". It contains several input fields for configuring monitoring parameters. The fields are arranged in two columns. The first column includes "Monitoring Frequency" (set to "Every 30 seconds"), "Plant Type" (with a "Show View" button), "Minimum Soil Moisture Level" (set to 10), "Minimum Temperature" (set to 20), "Minimum Humidity" (set to 10), and "Minimum Light" (set to 0). The second column includes "Maximum Soil Moisture Level" (set to 20), "Maximum Temperature" (set to 25), "Maximum Humidity" (set to 40), and "Maximum Light" (set to 200). A "Monitor Plant Conditions" button is located at the bottom center of the form.



Simple configuration GUI.



Specify plant type

Specify data collection frequency



Plant environment data with warnings based on tolerance levels specified by the user.

Milestones	Tasks	Status
<p>Acquire all hardware and software materials and become familiar with component datasheets and communication protocols. (February 28)</p>	<ul style="list-style-type: none"> • Researched availability of appropriate sensors which can interface with TStik using 1-wire, I2C, or CAN • Decided to go with 1-wire sensors for its simplicity and sensor availability. • Ordered sensors THS, moisture meter, and soil moisture from HobbyBoards. • Installed JavaKit. Loaded the TINI firmware TINI 1.17 and Slush_400.tbin. • Tested the TStik and network function by configuring the device to a hub as a member of the LAN. • Used both Telnet and FTP to the TStik which readily worked. • Wrote a sample HelloWorld Program to understand how to compile and build executables for the TINI using TINICConverter. 	<p>COMPLETED</p>
<p>Deploy a web server application on TINI with access to real-time clock capabilities (March 12)</p>	<ul style="list-style-type: none"> • Installed Tynamo Web Server onto the TINI. • Set up a MySQL RDBMS on the laptop. • Installed TiniAnt 1.2.0, and after some configuration issues were resolved was able to build a custom version of the webserver.tini. • Deployed the web server. Was not able to get Ant to create servlets right away. • Test connectivity and web server to the LAN using laptop. • Set the real-time clock 	<p>COMPLETED</p>

Milestones	Tasks	Status
Integrate the humidity sensor. (April 10)	<ul style="list-style-type: none"> • Implemented a Humidity Sensor API • Tested the sensor: <ul style="list-style-type: none"> • Method to display data to the command line interface. • Data appeared reasonable. 	COMPLETED Basics
Integrate a light exposure intensity/duration sensor. (April 11)	<ul style="list-style-type: none"> • Determined how to interface photodiode to the TStik. The THS board has the DS2483 which can read voltages generated by the photodiode and converts it to a 1-wire format. • Implement a Light Sensor API <ul style="list-style-type: none"> • Initialize sensor. • Read light intensity from the sensor. • Tested the sensor using a desk lamp. Achieved near 95% and 0 when covered. 	COMPLETED Basics
Created a servlet to display sensor data. Integrated the one-wire hub to the TStik to allow servlet the read data through the hub. (April 12)	<ul style="list-style-type: none"> • Learned how to configure Tynamo to manage and deploy custom made servlets. • <i>Build.props</i> and <i>Servlet.props</i> were modified to build the servlets and to map the servlet URL to the servlet class. • Added <i>OneWireDevice API</i> and <i>OneWireHub API</i>. • Implemented Plant Monitor Servlet and updated with capabilities to communicate with the 1-Wire hub which polls the sensor data on the THS board connected to Channel 1. 	COMPLETED Basics
Integrated a soil moisture sensor. (April 14)	<ul style="list-style-type: none"> • Split a CAT5 cable and stripped brown, light brown, and blue wires to be attached to the screw terminals of the moisture meter. The RJ45 end is connected to • Stripped the soil moisture sensor cable to reveal the copper wires which is attached to sensor terminals of the moisture meter. • Implemented the soil moisture sensor API. • Updated the Plant Monitor Servlet to read from the Moisture Meter at Channel 2 to reveal the soil moisture data. 	COMPLETED Basics

Milestones	Tasks	Status
Database connectivity (April 20)	<ul style="list-style-type: none"> • Implemented a database schema to store plant configuration data. • Researched methods to connect TINI to the MySQL database. • Found Remote JDBC by Hangar5. Updated Plant Monitor code to retrieve data from the database. Configured the build dependencies. Compiled but ended up with StringBuilder errors in the BuildDependency process, which were not resolved in time. • Built the HTML-based Plant Monitor GUI. 	ONGOING
Test, debug and final evaluation of product. (April 22)	<ul style="list-style-type: none"> • Integrated the Plant Monitor system to the laptop via a crossover connection. • Had trouble with JavaKit, but downloaded MTK, which worked like a charm ... to ensure changing the IP configurations would not break the system. • Had to go for Plan 2 for storing tolerance data. Tolerance data was hard coded into the HTML using JavaScript to enable dynamic filling of tolerance windows. • Tested the final product by using the interface to retrieve the data. 	Completed Basics

Retrospective

The key design decisions were the which embedded platform to use and which sensor interfaces. The combination of the TINI embedded system and the one-wire sensor interfaces helped to ensure a rapid prototyping process. The selection of one-wire sensor had the most variety to choose from, making it the key benefit of selecting one-wire. Additionally the TStik combined with the TINI API provided ready integration of the one-wire sensors. TINI API library offered a container for each family of one-wire sensors. These containers provide methods to easily interface and fetch data from the particular family.

This project offered the opportunity to learn about the TINI embedded Java environment and the ease of use. While it took some time to understand the tools and getting the development environment established, once it was in place, the system provided an efficient mechanism to develop and deploy the programs. Ant and TiniAnt provided an efficient method to quickly build and deploy the web server. I particularly thought the capability to run TINICConverter and BuildDependency to be very useful, as they were typically long and multiple commands when issued from the command line. Additionally they provided the capability to directly FTP the files to the TStik, another nice feature. Another key feature of TINI is the libraries available for one-wire sensors and other interfaces such as SPI, CAN, etc. These libraries make it easy to rapidly develop and test the various sensors. However, one of the key things I would do differently would be to set up the TStik environment earlier in order to give more time to develop the features of the project. I also learned that using MTK is much better than using JavaKit.

However, I also learned that TINI has limitations. It was not like typical Java and some of the JDK features available today were not readily found. The most significant one to this project was the lack of database support. The only API readily available was based on RPC, but it was not easy to configure and had unresolvable build errors, most likely due to compatibility issues. This was rather surprising, as I figured that any data collection mechanism, which the TINI can provide would have methods to access databases. However, after more research, it appeared most data logging is done as text files, and the text files can be parsed to serve as a database. Knowing the difficulty of connecting to a database without much experience in RPC, I would have used the text-based data logging technique.

The TStik was also limited, and sometimes appeared rather slow in loading servlet pages. One of the things I would improve upon is optimizing the way I write my servlets to ensure optimal performance on the TStik. The luxury of not caring about performance in coding is not as available as on a typical system.

This project also offered the opportunity to understand different types of sensors and how they work and how to calibrate them. I learned about clever techniques of using battery monitors to read from the photodiode, humidity and soil moisture sensors. The idea of using gypsum to create a current through the copper wire was interesting and appeared to work very well. HobbyBoards provided easy to interface sensors and connectors and other modules to rapidly create a one-wire network to monitor the plant environment.