Nortel SALSA

Administrative Web Interface Application

As part of this project's initial objectives, an online interface for parking lot administrators was created. This web application is a tool for parking lot administrators to check the status of parking spots and the sensors which monitor them. The design of the application was based on a paper prototyping phase as well as a series of user interviews with parking lot administrators. The interface also leverages several popular internet technologies including a web framework, JavaScript, asynchronous requests, and cascading style sheets (CSS).

Administrative Web Application

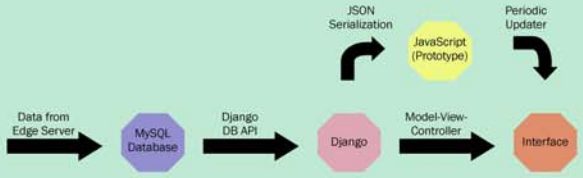

Figure 9. Flow diagram of the administrative web application

Functionality

Through a modified version of a Sample Driver, courtesy of Nortel, running on the Edge Server, a MySQL database connection is made every time data is sent from the Edge Server. This provides the real-time data which needs to be presented to the administrator.

Using the Django Database API, the relevant data is serialized using the JavaScript Object Notation (JSON) format, and posted as a website. The Prototype JavaScript framework then executes a request to retrieve the serialized data. This data is then presented in the user interface, which is governed by the Django Controller and View components.

Web Application Components

Web Frameworks

Web frameworks have been developed as a means for non-web programmers to rapidly develop and deploy internet-based applications. The web framework makes application development easier by providing tools for database interaction, URL and site structure management, and webpage serving.

Django, the web framework used in the parking management application, uses an open-source object-oriented scripting language called Python. In our application we utilized the built-in Django administrative interface (Figure 10) to modify and monitor sensor data without needing to log into the MySQL database on our sever. Additionally, the Model-View-Controller (MVC) architecture of the framework made development of functions, such as a threshold check for sensor data, easy to implement and deploy. Django also has the capability to build a database structured on the Model, which allowed for quick database creation but also makes it difficult to change the structure of an existing, populated database.

Architecture

Another difference between custom web applications and framework-based applications is the overall architecture of the application. In Django's case, the web framework operates under the Model-View-Controller (MVC) structure. In this structure, the model determines the structure of the data, the view governs how the model is presented, and the controller performs operations on the model.

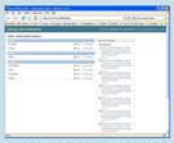

Figure 10: Django administrative interface

Figure 11. An example of a rollover

JavaScript

JavaScript provides the basic functionality required for interactive web pages as well as asynchronous requests. While coding JavaScript functions is acceptable for UI development, JavaScript frameworks exist to assist with more complicated functions, such as asynchronous requests.

Each JavaScript framework is tailored for a specific implementation, such as user interface development or asynchronous request management. Because the asynchronous request functionality of the application was the most important element of the interface (due to the time-sensitive nature of the data), the Prototype framework was used.

The periodic updater we used is one of the primary attractions of using Prototype. In the application, we used its ability to insert JSON data into a div (division) element to retrieve pre-generated HTML from the application Controller.

Asynchronous Requests

In order to provide some semblance of real-time data in the interface, the JavaScript Object Notation (JSON) update method was used. Django has built-in capability for serving JSON objects, and Prototype provided the functionality for periodic JSON requests. These two technologies combined formed the heart of the data management aspect of our application.

CSS

Cascading Style Sheets allow for more flexibility in element layout than exclusively using HTML will allow. CSS provides functionality for changing transparency, depth of elements, generalizing color schemes and placement, as well as a variety of other options that made it an attractive choice for our page design.

In our application, the rollovers, image positioning, and dynamic arrangement of parking spots for each lot display are all governed by CSS and JavaScript (Figure 11). Additional planned functionality including pop-up displays and toggling hidden page divisions also requires a combination of JavaScript and CSS.

Further Refinements

Hardware Optimization

The system could potentially use one mote for up to four spots, decreasing hardware costs.

At this point, we have one Zigbee module per sensor node. There are other more optimal solutions that could decrease the hardware costs of the system . For example, because the microcontroller has four different timers, it is possible to have at least three more sensors connected to a microcontroller. Furthermore, it is possible to connect several sensors using a bus to share a single Zigbee module and one master microcontroller.

Power Optimization

A deployed system eliminates debugging components that increase the battery life from 3.7 days to 6.6 days.

With batteries rated at 10,000mAh, the expected run time of the system is 3.7 days. The system cycles the PIC and sonar sensor on and off for eight seconds with the PIC's sleep capability.

A mass manufactured system would not include debugging parts, such as the 3.3V voltage regulator and two status LEDs. These simple improvements increase the run time of a node to 6.6 days.

More Power Optimization

Further improvements can be made in the future as the hardware's firmware becomes more developed.

The component that uses the most power is the Zigbee module: each node is transmitting every 16 seconds and the power consumption is relatively constant even if the node is idling. Because, parking space occupancy changes relatively slowly, we can get additional power savings if we only transmit when there is a change in status. Once the Zigbee firmware supports a sleep mode, power consumption could be reduced further.

Business Analysis

Bill of Materials for Mote

Part Name	Individual Price	Wholesale Price in Quantities of 100
Printed Circuit Board	$17	$4.22
Zigbee Module	$19	n/a
Ultrasound Sensor	$24.95	$19.96
Batteries (x3)	$7.95	$6.35
Battery Holder	$6.67	$5.69
Polycarbonate Box	$14.22	$10.29

The parking industry is currently deploying technology in the area of revenue control systems. Such systems provide users with electronic forms of payment and improve security & operating efficiency for lot administrators. Wireless sensor based inventory would be a valuable add-on to these systems and can be used to enhance their capabilities.

Interesting facts relevant to Pain Free Parking:

Size of the market:
No of parking spaces in the US: 105 Million
Forecasted parking industry revenues in 2008 : $20.86 Billion
Yearly technology spending by the parking industry : $1 Billion

Average spaces per location:
Airports - 2875
University/College Campuses - 1,675
Private Sector - 870
Public Sector - 795

Franklin W. Olin College of Engineering

BABSON MBA

Team Members:
Daniel Bufford
Dean Dieker
Christie Lee
Girish Wadhwani
Michael Wu

Faculty Adviser:
Dr. Bradley Minch

Nortel Networks Liaison:
Inder Monga

SCOPE
Senior Consulting Program for Engineering

NORTEL NETWORKS