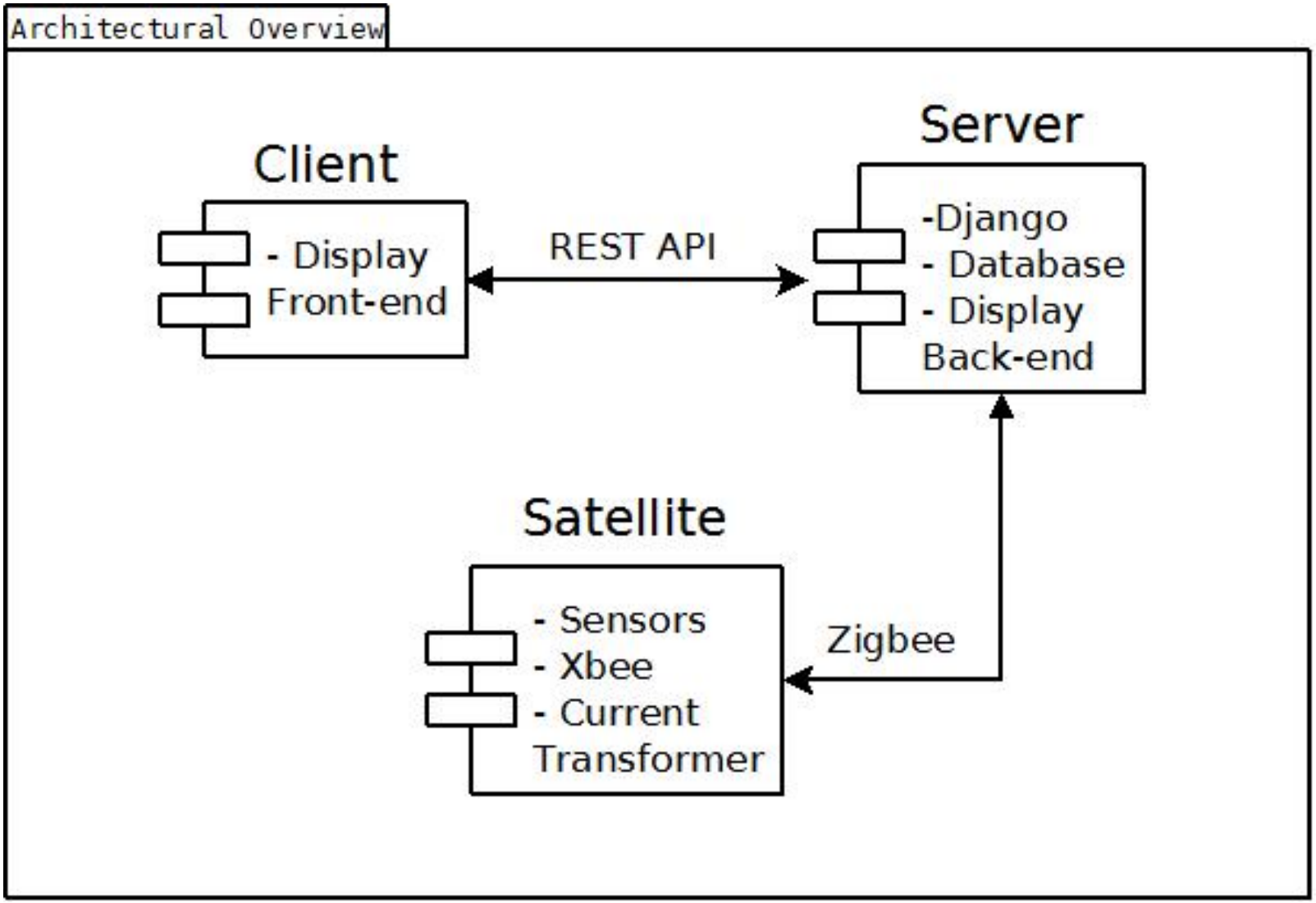
POW-R

Power Outlet Wireless Reporter

Grace De Geus
Charles Hathaway
Forest Immel
Nate Pickett
Niloc Quimby

April 24^{th}, 2013

Overview


Architectural Overview
Client
- Display
Front-end
REST API
Server
-Django
- Database
- Display
Back-end
Satellite
- Sensors
- Xbee
- Current
Transformer
Zigbee

Why XBee Radios?

- Small form factor (just larger than U.S. quarter)
- Low power consumption (˜.1 W)
- Talk over ZigBee 802.15.4 standard

ZigBee Specification

- High level communications protocol
- Designed for low power digital radios
- Mesh network topology
- Network can expand on the fly
- 2.4GHz operating spectrum

ZigBee Mesh and POW-R

- One Coordinator per mesh
  - Maintains mesh
  - Receives transmissions from all router XBees
  - Attached to POW-R server via Arduino
- All Satellites have router XBees
- Router XBees "bounce" transmissions to Coordinator

Coordinator Arduino

- Hosts Coordinator XBee
- Powers LCD to display IP address of Server
- Sends Server data readings over serial

Server

- Raspberry Pi
- Small form factor (˜8.5 x 5.6 cm)
- Low power consumption (˜3.5 W)
- Acts as data center and web server for Display

Software Architecture

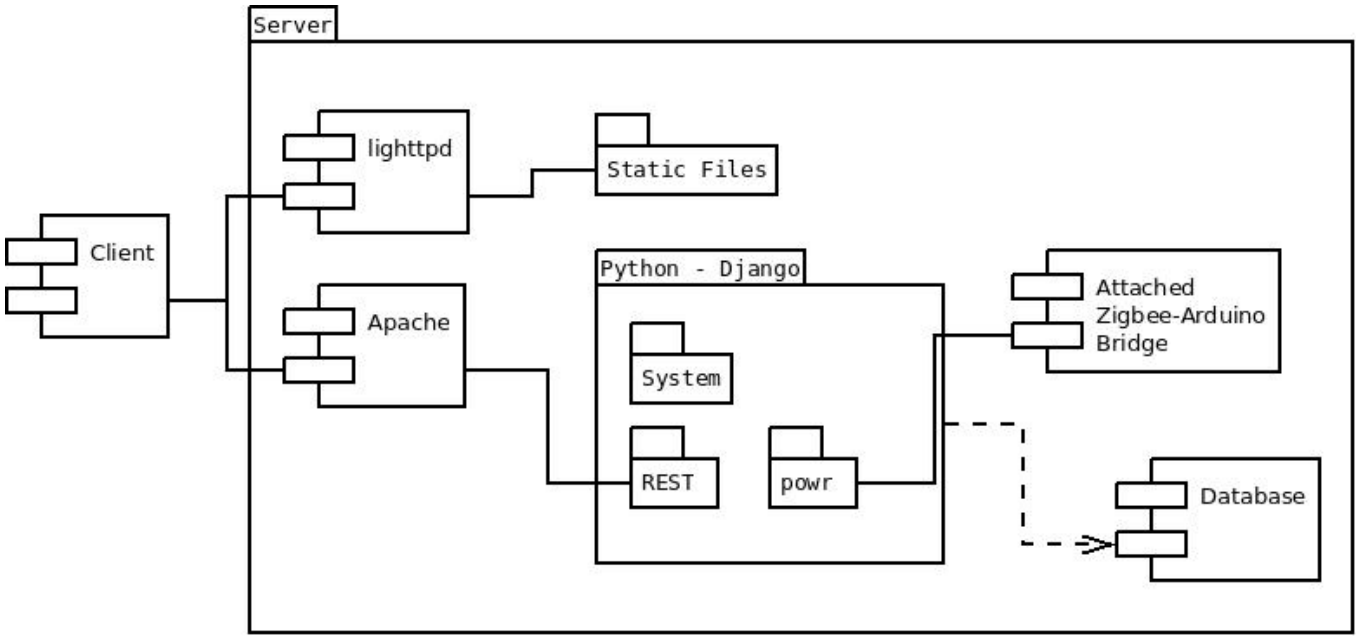
Software Overview

- Python on the backend
  A. Django for the REST, HTTP stuff [**?**]
  B. Custom python to interact with Arduino interface
- Heavy Javascript on the frontend
  A. jqplot for creating graphs (jQuery included) [**?**]
  B. django-compress to reduce the Javascript files to a manageable size.
  C. AngularJS for a MVC architecture that consumes the REST backend

Backend Overview

- Django will be used to handle database
- Tastypie will be used to prototype the REST API
- Each functional area of the project is a Django module
  A. System
  B. POW-R
  C. REST API

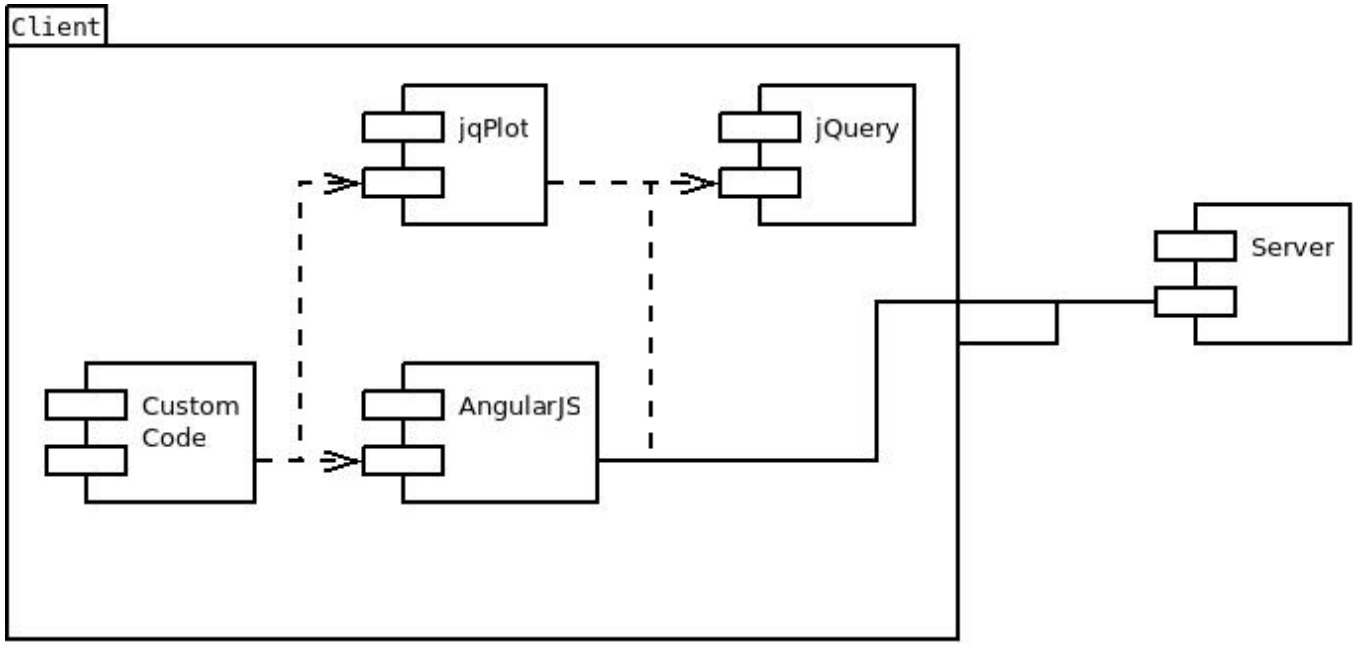
Backend Architecture



Frontend Architecture

- django-compress
  A. Minifies the JavaScript code so clients load faster
  B. Easy to use, can be tested
- AngularJS - MVC Architecture on the client-side
- jqplot - Renders charts and graphs
- jQuery - Deals with all the behind-the-scenes AJAX stuff

Frontend Architecture



Software Implementation

Software Implementation

- Went more-or-less according to plan
  1. We switched away from Backbone.js and iCanHaz because AngularJS covered both domains
  2. We didn't use Asynchronous Module Definition (AMD) because very few libraries supported it
  3. There were some small modifications to the REST API to make it more compatible with the world
- Biggest software problem was the complexity of the setup
  1. Because of the number of libraries and frameworks, it was difficult to do development in Windows
  2. We developed two solutions; a completely isolated Python development environment, and a VirtualBox for people with Windows

Software Implementation

- Some things didn't make it to the final product for a variety of reasons
  - Adding satellites was ditched because it was too confusing for an end user
  - The power-bill-guestimater was ditched because it would be impossible to accurately track all power consumption, thus leading to incorrect guestimates
  - The user-management stuff was simplified because we don't need a complex permission system
- Some things were added
  - We used Intro.JS to create a "help" feature in our website
  - We added a JS compressor to keep the codebase small when delivered to the client

Demonstration time!

Check it out!

Questions and Closing

Questions?

Presentation made using LaTeX

Our website: `http://powr.logrit.com/`