Project Quebec

Requirements Specification

Project Brief

"Remote DIY software plumbing
Systems like readyourmeter.org (from Cambridge) were developed to help remote monitoring of your house from the web. The next step is remote control of your house. Create a web-based programming environment, that will allow homeowners to do DIY software plumbing (scripting and configuration) of home media, energy and control systems."

Project Goal

A consumable web-based system to control house automation devices and display information from sensors.

Project Objectives

Through an aesthetically pleasing web interface allow users to:

- Easily specify the devices and sensors they have connected
- View the status of these devices and sensors
- Display the past state and events
- Remotely interact with the home in a secure fashion
- Schedule actions to be preformed by devices (or groups of devices)
- Create logic that controls devices (or groups of devices) based on certain events

Additionally the system should:

- Communicate with the devices via the interface to cause them to perform the scheduled actions
- Communicate with the devices via the interface to obtain their current state
- Comply with applicable standards and reuse code where possible
- Ensure all error messages are informative.

Background

The X10 protocol has been in use since the 1970s but never took off in the UK. This is partly due to the complexity of setting up such a system. For a long time it has been predicted that the home of the future will be fully automated and this project is a step along the road to that goal. Currently existing products tend towards custom hardware packages which plug into the house and can be controlled by some limited user interface on the device or via scripting from a computer (for example using http://www.jabberwocky.com/software/xtend/). Unfortunately the average user can't write shell scripts and so something simpler and easier to use is required.

Use Cases

For this project to be useful, there must be many scenarios in which it would be more convenient to use the system than to interact with the devices in the normal way. A subset of these scenarios is listed below in order to focus the project on catering for real people.

After leaving their house, putting the alarm on and locking it, a user may notice a bedroom light on. They should be able to easily turn a single light off using a mobile device.

When leaving the house, or at night a user may with to turn all lights off without having to walk around the whole house.

On waking up, a quick button press to open all the curtains would be convenient.

When it gets dark the user might want the curtains to close as this helps keep the house warm and reduces loss of energy through the windows. This should be possible through the use of a timed event on the system or sensors detecting dusk.

A user may want to turn heating off during the night, they should be able to set the system to automatically turn the heating off at 12AM and on at 6AM.

A user may want the peripherals connected to a computer or television to automatically turn off at the same time as the main device. If an X10 device sends a signal when the main device turns off, it should be possible to use this to trigger the other devices to turn off.

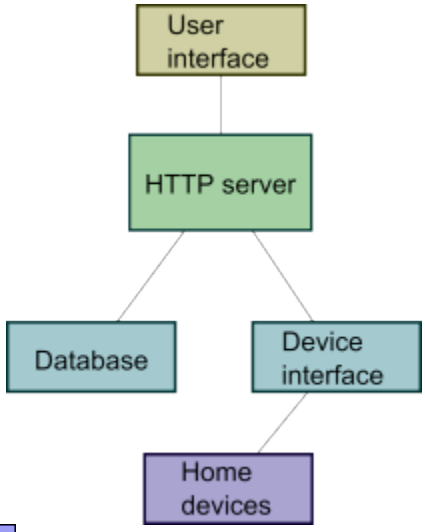
If a user has a remote control they may want to program the buttons for commonly used functions. They should be able to assign each button to a control a different device or set of devices.

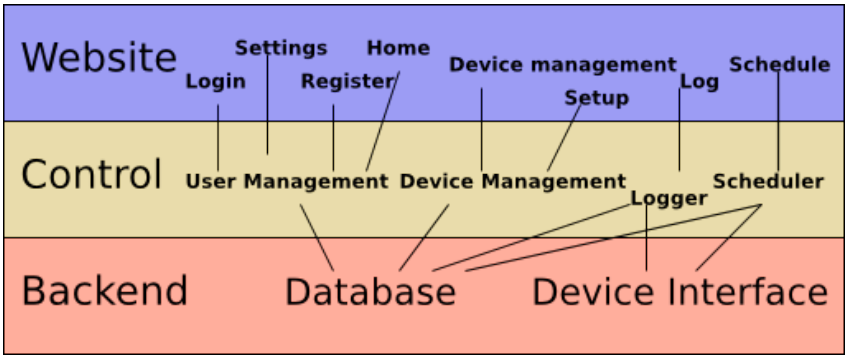
The user should be able to choose a temperature, as a threshold for turning the heating on/ off. The temperature would be taken from a water tank sensor (the middle sensor).

If the temperature gets above a certain value in the hallway during the day (on a summers day), the curtains will be closed to keep the house cool.

Solution Overview

The main body of the project will be written in Java, which will reside on a web server. The Java will interface with a PostgreSQL database and X10 devices. It will also generate pages composed of Hyper Text Markup Language, Cascading Style Sheets and possibly Java Script to be served to the user on demand.



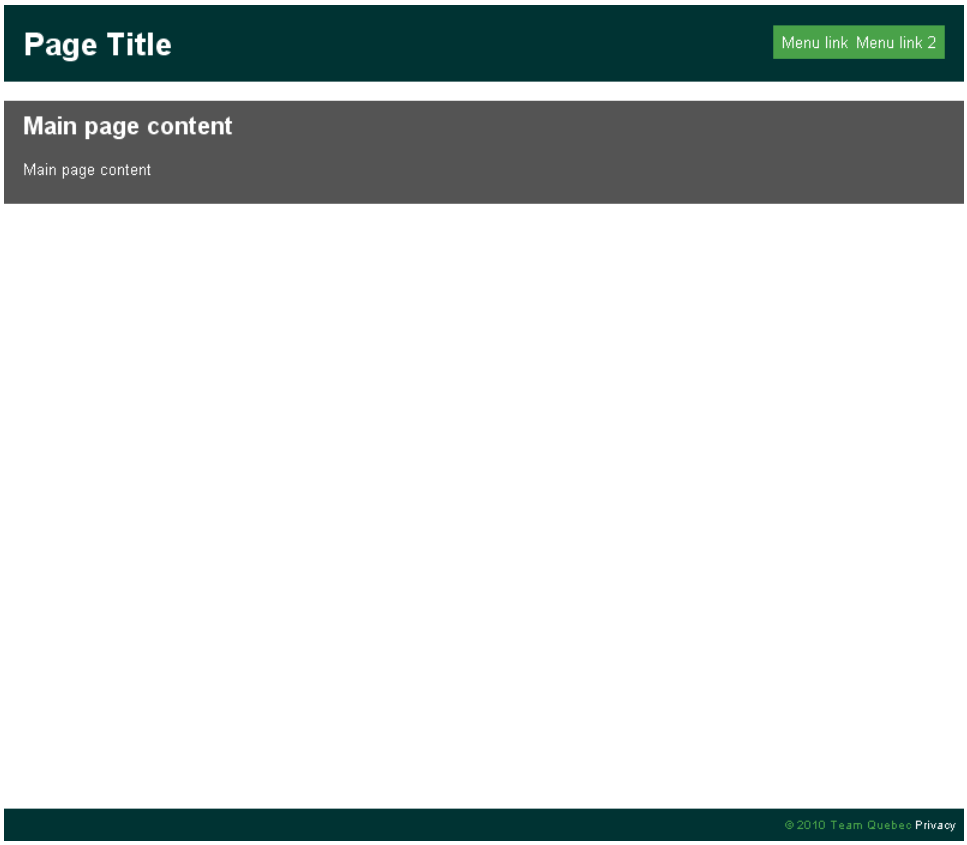


User Interface Requirements

This section details the minimum functionality required to be present in the user interface.

[In this section, website means the web interface, and web pages are screens of the interface. Sensors are a subset of devices.]

Below is a general template for the website showing proposed locations of features common to every page.



It is intended that the box containing the main page content will grow as needed to hold all its content. Additionally, multiple such boxes could be present on a page if its content can be sensibly separated into two or more sections.

The user interface must provide a consistent, unified user experience.

- The colour scheme, typeface and text size of ordinary text must be the same on every page of the website when viewed through the same browser.
- The locations of the website header and footer must be the same (with the exception of the footer moving further down to accommodate the main page content if necessary) on every page of the website when viewed through the same browser.

The website must be usable on as wide a variety of computers as possible.

- The website must be usable on desktop computers on modern browsers (IE7+, Firefox, Opera, Safari, Chrome)
- The website must be usable on handheld devices such as Android and Windows Mobile phones.
- Users who either have disabled or do not have support for advanced features such as AJAX and JavaScript must be able to use the website without these features.
- W3C standards (<http://www.w3.org/>) for markup language and style sheets must be followed with the exception of providing additional visual enhancements that do not impair the usability of the website regardless of their presence.
- Accessibility guidelines (<http://www.w3.org/TR/WCAG/>, <http://www.opsi.gov.uk/acts/acts1995/Ukpga_19950050_en_1.htm> in particular Part III, sections 19-21) must be followed or an accessible version of the website made available.

The website must provide a good user experience.

- The website must be aesthetically pleasing.
- The website must not contain distracting elements such as audio or excessive animation (the latter is acceptable when updating the page with response from an AJAX request, however.)
- The website must have a link to either the Home page (if the user is not logged in) or the page containing the user's log and schedule (if the user is logged in) on every page.
- The website must not contain loud colours.
- The website must be easy to navigate - e.g. links should be clearly different to normal text.
- The sign-in/registration form must be present on every page of the website if the user is not logged in to an account.
- The website must not be overly complicated.
- Plain English should be used in preference to technical jargon wherever possible.
- Help should be provided in context, e.g. through the use of tool-tips to explain what buttons do however the website should be designed such that this is not necessary.

Users must be able to specify the devices in their home.

- A form must exist by use of which users can specify a device by some identifier for later control from the website.
- A form must exist by use of which users can remove details of devices stored on the website.
- A form must exist to allow users to specify the details of a connection from the website to the devices in their home in order that the website can interact with them.

Users must be able to view the status of these devices.

- A listing of all the devices the user has specified to the website (and not subsequently removed) and the current status of each device must exist on the website.

Users must be able to see events as they occur.

- A frequently updated list of all recent events must exist on the website. Where support is available, this list must update itself without user intervention.

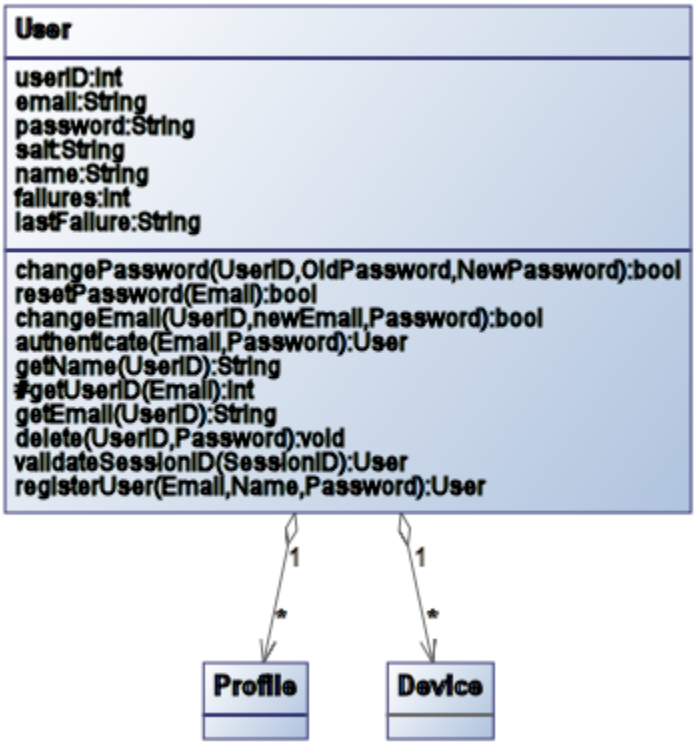
Users must be able to create events triggered by specific conditions

- One or more forms must exist that allow the user to create a set of preconditions to trigger a command to be sent to a device.
- Users must be able to use time or sensors as preconditions for triggering an event.
- An example of a precondition would be "After 6AM".

The website must interact with the home in a secure fashion.

- The website must not allow unauthenticated users to access device information.
- The website must not allow connections made by non-users to perform any actions except registering.
- The website must have a readily accessible page detailing its privacy policy, which must explain how users' information will be used.
- The website must comply with the Data Protection Act 1998 (<http://www.opsi.gov.uk/Acts/Acts1998/ukpga_19980029_en_1>).

User Management



Information about a user will be stored to facilitate login and use of the system.

- Unique ID (number, generated on registration, internal)
- Hashed password
- Salt for the hash (system must be reasonably secure as nobody wants a hacker in control of their house)
- Email address (for logins and password resets)
- User's name
- Number of failed login attempts since last successful login
- Timestamp of last failed login attempt

User objects must support a number of methods so that they can be used appropriately and the data they contain is useful to other components of the system.

- Change password
- Reset password (generates random password, changes password and sends email)
- Change email address
- Authenticate (takes an email address and password as an input - checks against user's hashed password)
- Get name
- Get unique ID
- Get email
- Delete (deletes user and all associated data - with proper use of Foreign Keys in the database this should be automatic when the user is deleted)
- Validate session ID
- Register User (saves a new user to the database)

Users must have associated device and schedule data.

- Devices must be associated with a user ID.
- Custom profiles - complete, named sets of device settings, that can be applied all at once. There are also default profiles - e.g. winter, summer, etc. - the main difference between custom and default profiles is that default profiles will have events which trigger them to be activated such as the date changing from summer to winter.These also need an associated user ID.

The login system must provide adequate security.

- 3 failed log in attempts with the same email address will lead to the account being locked for 15 minutes, this count will be incremented until the next successful login occurs at which point any non zero count will be displayed to the user. This should prevent dictionary attacks but also not confuse the user. A CAPTCHA such as reCAPATCHA may be used to require a human presence at all attempts after the first 3.

Physical Interface for Devices

X10 Devices

- All incoming information from X10 devices will be received by the system on a port which generates character strings.
- The strings begin with the address of the device: a combination of a house code (a letter from A to P) and a unit code (a number from 1 to 16).
- This will be followed by a simple textual representation of the command, e.g. an entire string may be "C12 ON".
- Commands can also be sent out to X10 devices using the same port. The commands will be transmitted in the same format.
- Incoming commands will be decoded and then:
    - A query will be sent to the device management system to retrieve the device ID associated with the house code/unit code received.
    - If the lookup is successful, the command will be passed to the scheduler, which will decide whether an event should be triggered.
    - In either case, details of the command received will be written to the log.
- Outgoing commands will be received by the scheduler, encoded, and written to the port.

1-Wire temperature sensor devices

- Similarly, all incoming information from 1-Wire devices will be received by the system on a (different) port which generates character strings.
- The strings begin with the 16 hexadecimal character address of the device, which is factory set and unique.
- This will be followed by a floating point temperature value.
- Incoming data will be decoded an then:
    - A query will be sent to the device management system to retrieve the device ID associated with the device address received.
    - If the lookup is successful, the a new entry will be written to the log.

Log

In general:

- The log should be append only.
- All log entries should have a unique immutable identifier.

The log should record input from X10 devices.

- The log should be updated whenever a command is received from an X10 device.
- Users should be able to view log entries, but the log entry ID and device IDs should be hidden.
- The log entries should contain:
    - A unique immutable log entry ID.
    - A timestamp.
    - The house code and unit code of the command received.
    - The device ID and name of the X10 device, if it is registered with the system.
    - The user associated with the device.
    - The command received.

The log should record input from 1-Wire sensor devices.

- The log should be periodically updated with the current output values of sensor devices.
- It should be possible to query the most recent output values of the sensor devices via the sensor device's name or device ID. This information should also be visible to users.
- The log entries should contain:
    - A unique immutable log entry ID.
    - A timestamp.
    - The factory set ID of the 1-Wire sensor device.
    - The device ID and name of the sensor device, if it is registered with the system.
    - The user associated with the device.
    - The reading from the sensor.

The log should record attempted actions.

- If a particular time is reached or event occurs which is scheduled to perform an action, details of the action should be written to the log.
- It should be possible to query the most recent attempted actions, and also to limit this to only actions which have succeeded/failed.
- Users should be able to view log entries, but the log entry ID and device and group IDs should be hidden.
- The log entries should contain:
    - A unique immutable log entry ID.
    - A timestamp.
    - The user and profile associated with the scheduled event.
    - The event that caused the action to be performed. If the event was triggered by a command sent from an X10 device or sensory input from a 1-Wire device, the ID and name of the device should be recorded.
    - The ID and name of the device or device group that the action was to be performed on.
    - Whether or not the action was successful. In the event of an action failing the reason why it failed should also be recorded.

Schedule

Actions are scheduled commands. They are acted upon when their associated event is true, and they are be associated with a DeviceGroup or a Device. They have an associated command, which is sent to the relevant device(s) if their precondition not false. Examples of events: Profile start (for example profile now DAY or NIGHT), at 18:00, 2010-03-13 22:00

Preconditions are logical statements that are either true or false. Some will be implemented in the project, but an interface for plugins that deal with their own settings independently could be provided so that more preconditions can be supported. Plugins would return true or false when queried - for example, a Google Latitude plugin may return true when a user is within a mile of their house. Examples: is Saturday, is Weekend, is Weekday, > 18 (temperature source), < 16 (temperature source), <Command> (device).

The user should be able to schedule actions.

- It should be possible for actions to be scheduled to happen at a particular time (once or periodically) and it should be possible for actions to be scheduled to happen when some input is received (such as from a temperature sensor).
- It should also be possible for an action to be scheduled to happen at a particular time (once or periodically) only if a precondition holds (such a temperature sensor being below a certain level).
- Users should only be able to schedule actions for a device if they are supported by that device.

The user should be able to edit, disable and delete scheduled actions.

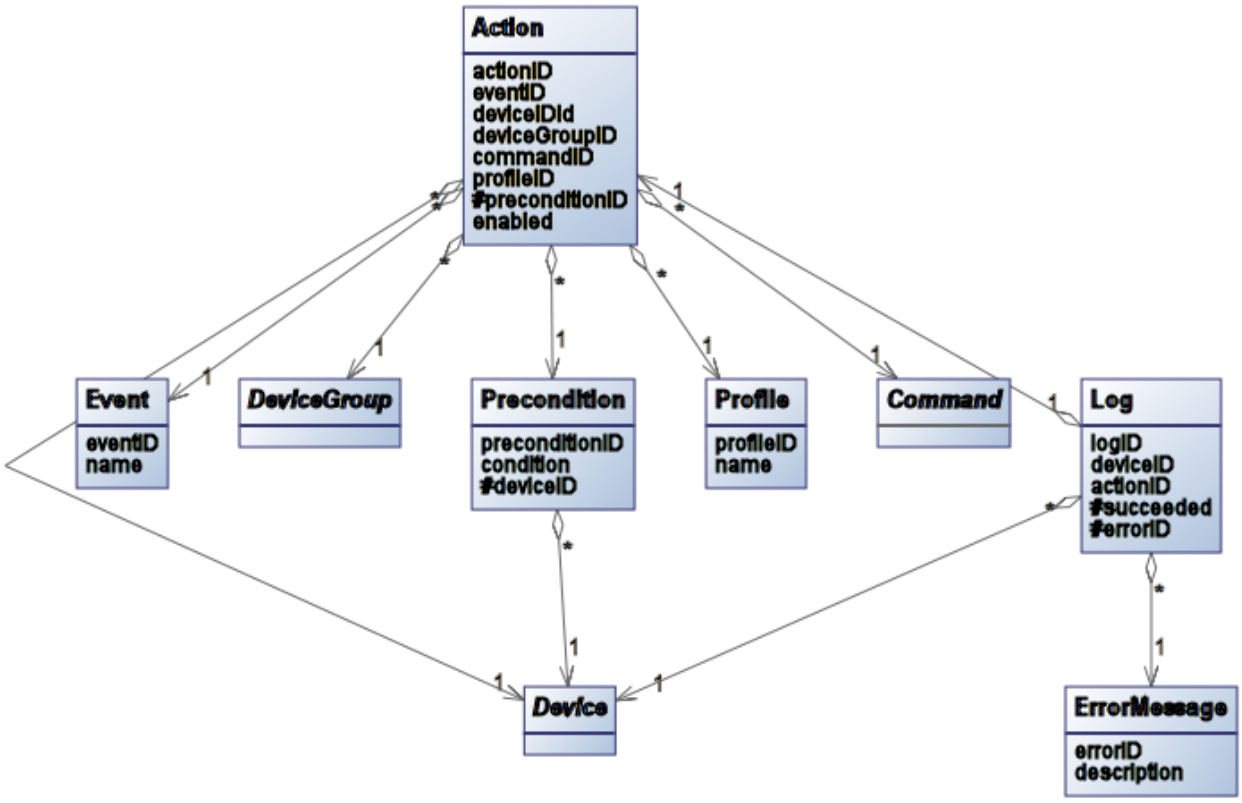
- Any aspect of an action should be editable.
- Users should be able to easily reactivate a disabled action (for example, during a cold snap, they might want to temporarily disable actions that turn the heating off).

It should be possible to see actions that are due to take place.

- Users should be able to see a list of all the actions scheduled to occur before a certain date, or up to some specified maximum number.
- The list of actions scheduled to occur should be listed with those actions which are scheduled to occur when some input is received (such as from a temperature sensor) first followed by those scheduled against points in time in order from the closest to the most distant future. The difference between the two types of actions should be made clear.

Commands should be sent to devices at the correct time and/or when preconditions are met.

- This needs to be very stable. The main task of the system cannot be fulfilled if commands are not sent when they should be, or if commands are sent when they shouldn't be - users would rapidly abandon the system.



The # symbol means the column may be NULL all other columns are NOT NULL.

Direct device control requirements

The user must be able to locate a representation of a specific device which has already been named and potentially grouped. A hierarchical representation should be used.

- There should be some way to distinguish between a group and a device (different coloured background or text or an icon).
- Any devices not currently grouped will be in the top level of the hierarchy.
- Any groups without a parent group will be in the top level.
- Devices should be shown twice if they belong to multiple groups.
- If screen size is not sufficient, it is preferable to fit all of a single level of the hierarchy in the view rather than a single vertical line in the hierarchy.

Once the desired device has been found, the user should be able to view information such as its current status (on or off).

- Each device element should show the device status and show at least some of the supported commands on the element (at minimum toggling on/off).
- The supported commands may be displayed as icons for brevity, but their usage should be explained somewhere.
- If some actions are not included (for example for specialist devices) there should be something to indicate that the user should click the device to see more.

It should be possible to send a command to the device selected.

- Clicking a supported command (icon) should execute the relevant command.
- The user should see some immediate feedback when they click a command.

- All X10 devices should be assumed to support On/Off commands.
- Users must be able to send at least the following commands to devices:
- The All On command.
- The All Off command.
- Dim and brighten commands.
- On command (for a relay) - can either be continuous as with the normal on command, or a single pulse on.

The user may require a more detailed view of the item or wish to display a subsection of the hierarchy in order to more locate a device when browsing using a small screen.

- Clicking on an element E (but not on any of the commands) should display a device specific page.
- The device specific page should display information about the status and available commands on E in a more verbose form.
- It should also display a hierarchy where E is at the top level and the only nodes displayed other than E are descendants of E (if E is a group).
- Information on E should show if the device provides inputs, and provide a link to create an event the input should trigger.
- There should be a customised link to the event set-up page which automatically selects the current device as the output device.

In order to program events a user might want to see what input signals their device is sending.

- They will need a log showing the different events received from the input device.
- The log should display the most recently received signals from the selected device.
- A "wizard" should guide the user through the event creation based on a signal previously sent (selected from the log).

A user should be able to send a command to every descendant device and group simultaneously (i.e. a single click sends the message to all devices)

- A device group will support the intersection of set of functions of each descendant device and should be displayed in the same manner as actions on a single device.
- If the user has stated that a device is of a type not supporting command C, there should be no way to execute command C on the device provided in the interface.

Device Management requirements

The device management database should support a several types of device.

- Initially, it needs to support:
- X10 receiver devices
- X10 transmitter devices
- X10 2-way devices
- 1-Wire devices (receiver only)
- A further possibility is to include more types of device via plugins.
- Plugins could support "virtual devices" which could send commands based on information from other sources, e.g. Google Latitude to turn the lights on when nearing home.

For X10 transmitter devices (or 1-way devices), the database should store:

- House code/unit code pair
- Transmitter device type
    - This determines what commands can be sent from a transmitter device. Example types: on/off switch, dimmer switch etc.

For X10 receiver devices (or 2-way devices), the database should store:

- House code/unit code pair
- Receiver device type
    - This determines what commands can be sent to a receiver device. Example types: simple lamp, dimmer lamp etc.
    - For both X10 receiver and X10 transmitter devices, having device types instead of a simple list of supported commands eases setup and removes the need to e.g. add support for on and off commands to every controllable device. However, it should be possible to add new device types.

For 1-Wire devices, the database should store:

- 1-Wire ID
- 1-Wire device type
- Initially, the only 1-Wire device type will be temperature sensor, but this could conceivably be extended.

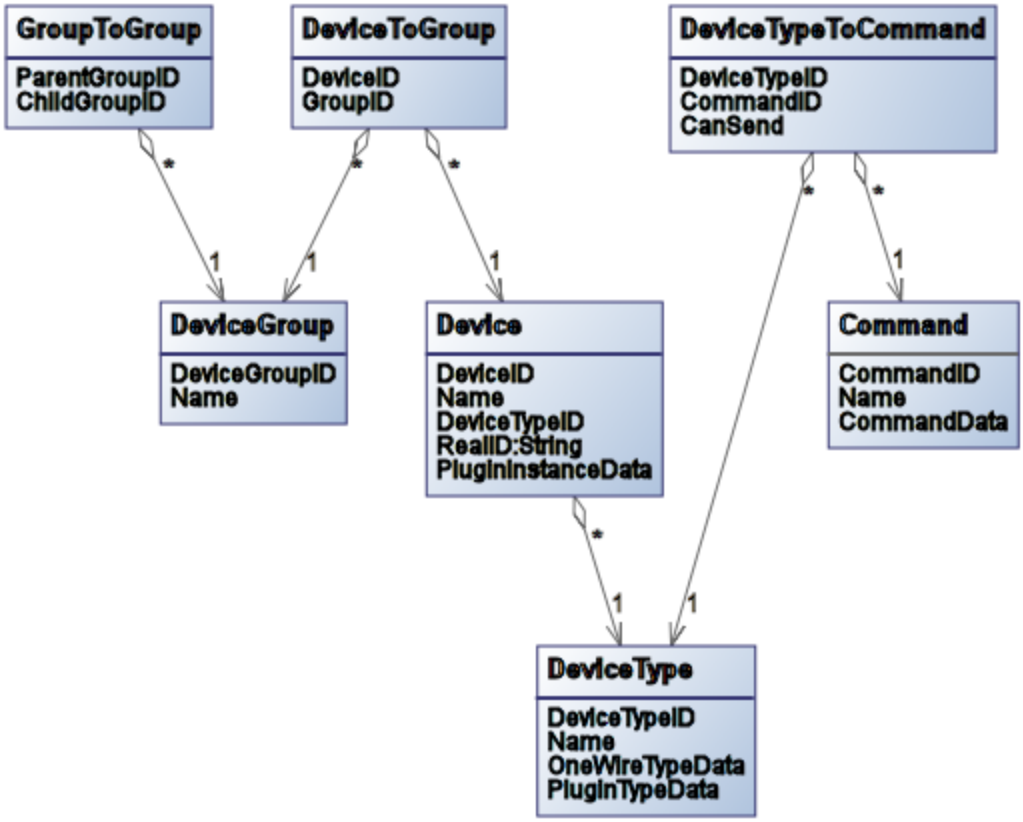
Devices should be able to be grouped together, e.g. downstairs lighting, all curtains etc.

- Commands could be sent to all devices in a group.

The database should support these functions:

- Add/remove device
- On creation, the name of the device, whether it is a X10 transmitter, X10 receiver, 1-Wire etc., and additional details based on which type was selected should be given.
- It will not permit adding an X10 device which transmits or receives on the same house code/unit code combination as an existing X10 device.
- Create group
- A set of devices should be given at creation; others can be added later.
- Delete group
- Add device to group
- Add receiver device type
- Need to specify what commands are supported - on/off will always be supported.
- Add command
- Get house code/unit code combinations currently in use (to help setting up new devices).
- Get group of device
- List devices in a group
- List commands supported by a specific device
- List commands supported by a device type

Device database schema:



Descriptions of tables:

- DeviceGroup
  - This stores the name of a group.
- Device
  - This stores details about a single devices. Each device has a single device type. If the device is a 1-Wire device, its unique factory set 1-Wire ID number is stored in the RealID column. In the case of an X10 device the House Code and Unit code is stored in the RealID column e.g. A1. As a possible extension of the core functionality, the website may offer the ability to define new classes of device via plugins; information about an instance of such a device would be stored in the PluginInstanceData column.
- GroupToGroup
  - Each row is a mapping between a child group and one of its parents.
- DeviceToGroup
  - Each row is a mapping between a device and one of its parents.
- DeviceType
  - The type of a device characterises what data it can send and receive. If the device type describes a 1-Wire device, data relevant to that type is stored in the OneWireTypeData column. As a possible extension of the core functionality, the website may offer the ability to define new classes of device via plugins; information about the type of such a device would be stored in the PluginTypeData column.
- Command
  - This table stores commands that can be sent over the X10 protocol. Each X10 transmitter type and X10 receiver type is able to transmit and receive

commands respectively, so there is a many-to-many link between Command and DeviceType via DeviceTypeToCommand.
- DeviceTypeToCommand
    - A row will store a single mapping from a device type to a command it is able to send or receive.
    - The CanSend field is a boolean. If it is true, the command is one hich the device can send, otherwise it's one that the device can receive.

Security

Threat Model

- Bots looking for known exploits.
- Crackers trying to access someone else's account to control their house.
- Criminals trying to find out if people are home and try to disable their security system.
- Users trying to access each others accounts.

Attacks

- SQL injection.
- Brute force password attacks.
- Man in the middle attacks against users trying to login to the site and the site trying to connect to the user's devices.
- Denial of Service: attempts to break the site so that it is not able to control devices.
- Automatic account creation by bots until the database runs out of space or to try and create spam.
- DDoS by a botnet making many requests to try and prevent the site being able to do anything.

Countermeasures

- Use Prepared Statements in SQL and sanitise input.
- Defensive programming: check all arguments before using them.
- Unit testing: test throwing wrong and malicious arguments at functions as well as correct ones.
- Hostile attack: ask some people to try and break the site as part of user testing.
- Use SSL for all connections.
- CAPTCHA on account creation to discourage bots from registering.

Acceptance Criteria

The following points summarise the criteria that we must complete before the system is delivered to the end-use.
The user can:

- Log in to the system
- View the current state from each device
- Be able to switch each device on and off
- Schedule events affecting devices

- Turn collections of events (profiles) on/off

Project Plan

Project Life Cycle

01/02/10		Finish external interface definition of assigned module
	13:30 – 16:30	Group Coding Session – Intel Lab WGB
02/02/10	13:30 – 15:00	Team Meeting -WGB
05/02/10	13:30 – 14:30	Team meeting - WGB
09/02/10		Finish module implementation stage
		Finish summary of testing carried out and pass to project leader
10/02/10		Project Leader will submit progress report
12/02/10	14:00 – 16:00	Second Group Project Meeting - FW09 WGB
18/02/10		Get integrated modules to a usable state
22/02/10		Draft project reports
23/02/10		Finish project reports
26/02/10	14:00 – 16:00	Third Group Project Meeting - FW09 WGB
03/03/10	14:00 – 16:00	Group Project Demonstrations - Intel Laboratory
	16:15 – 17:15	Group Project Presentations - LT1 WGB

Responsibilities

The project has been partitioned into several initial development sections. The work load may be redistributed based on difficulties experienced.

Graham (Project leader):
Database - Provide a database using PostgreSQL and a Java wrapper for accessing it such that no other module contains any SQL.

Device Control - Provide a user interface to select an device or group from its hierarchy and execute commands on it.

Chris:
Device Interface - A wrapper in Java for the communication with the devices. Provide a method to send a command on the interface. Listen for commands and notify the log and then the scheduler. Provide a mechanism for testing this interface and the rest of the system by emulating the interface (just a simple wrapper that sends random commands and accepts them).

Log - Provide a method to accept a command which has been received and pass it to the database to be recorded, methods to provide a log of recent events on the entire system or a specific device. Also method that can be accessed publicly to log an event in a textual form.

Mark:
Scheduler - Provide a method to accept a command which has been received and test whether it should trigger any event. This module must have functions for event creation, editing and execution based on the appropriate conditions.

Robbie:
User Interface - Implement helper functions to create forms etc. Code the page sections common to every page and create a template for other files to adhere to.

Device Management - Provide a user interface to add, edit and remove devices and device groups.

Daniel:
SVN - Setup and maintain and document its use for the rest of the team. Ensure compliance with coding standards.

TomCat - Setup a testing environment for the project to run in.

User Management - Login, registration, authentication.
Security.