# Dynamote

Software Engineering 319
**Project Report**


## Group Members:

John Ryan - jpryan@iastate.edu
Michael Carter - carter00@iastate.edu
Stephen Brossart - brossart@iastate.edu
Sylvain Heiniger - sylvainh@iastate.edu
Ryan Sanders - rks@iastate.edu


### Project Proposal:

The main idea for our project is to create an IR universal remote control that can be used by anyone with a smart phone or tablet device with a web browser and

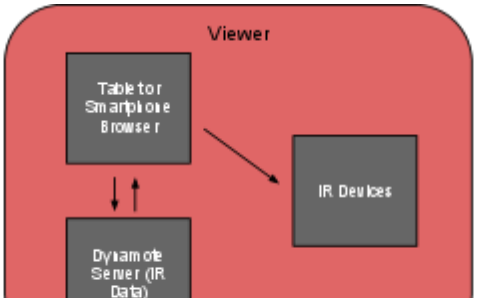
a computer. This software will have two main parts in the eyes of the user: A remote builder interface, and a remote viewer interface. The remote builder interface is meant to be used on any regular computer with a browser and is used to configure the remote(s). The viewer is meant to be viewed on a smart phone or tablet and is to be used as the remote control.

The other main part of this project is to create an interface to manage the remotes and user accounts system. This interface is contained in a site to be hosted on a computer that is on the same network as the smart phone or tablet. The interface allows the user/admin to add, remove and change permissions on accounts, manage controlled devices and change languages on the site itself. All this information is to be stored in a database that accompanies the site.

The builder interface allows the user to create a remote or multiple remotes to control IR devices in the AV system they have. These remotes are managed in a tab interface. Each tab shows a panel with the approximate size and aspect of the viewer device. It also has a another panel which contains buttons that can be placed on the viewer screen. The builder also allows the user to name the remotes.

The viewer interface takes what the user has created in the builder interface and wraps it into a interface for easy browsing from a smart phone or tablet. The viewer, instead of allowing the user to edit buttons on the screen, fixes them in place and sends their corresponding commands to a gateway on the network that can convert these commands to actual IR to be received by the controlled devices.

## High Level Interaction:



There are two basic modes of interaction with the software. These are Remote Viewer (on a tablet or smart phone) and Remote Builder and Management (on a computer). In either mode, all interaction is done through a web browser. Building can then be divided up into actual remote building and account / device configuration. The Remote Builder is a typical sectioned website designed for use with a mouse and keyboard. It has has different forms for adding, editing and removing devices and accounts. All of this configuration data is saved on the local Dynamote Server. In addition to these management features a builder is included. The user interacts with this module by dragging buttons that contain IR actions onto a grid with that represents the remote viewer screen. The Remote Viewer is a site designed for touch screen interaction. This can be used much like a conventional remote control, where the user presses buttons that are associated with device actions. The builder talks to the local Dynamote Server to get this IR data which is then transmitted over the network to the IR devices.

## Usability Features:

### Implemented Features:

**User Accounts -** individual accounts for each member of a household, business or

other group so that each one can make his/her own remote layout.

**User Restrictions -** allows for admin and non-admin users, this way, a non-admin user can't delete devices in a system and so that non-admin users don't have to add devices themselves.

**User Remote Layouts - (user remote layout profiles) -** each user can create his/her own unique remote layout with the devices available.

**Remote Builder (editor) -** An interface that allows the user to arrange buttons on the remote and add and create multiple remotes.

**IR Devices - Pick from multiple IR devices.**

**Multiple Languages -** All interfaces can be automatically viewed in English, French, and Spanish, as well as "Pirate"!

**Device GUI Optimization -** create correctly sized layouts for tablets or phones.

**Drag and Drop / Grid System -** A feature of the builder that allows the user to drag buttons onto the remote from a panel and delete them by moving them to the trash.

**Unimplemented Features:**

**Macros -** Assign multiple IR commands to each button in a specified sequence.

**Button Builder -** An interface for making new graphical buttons with text and pictures.

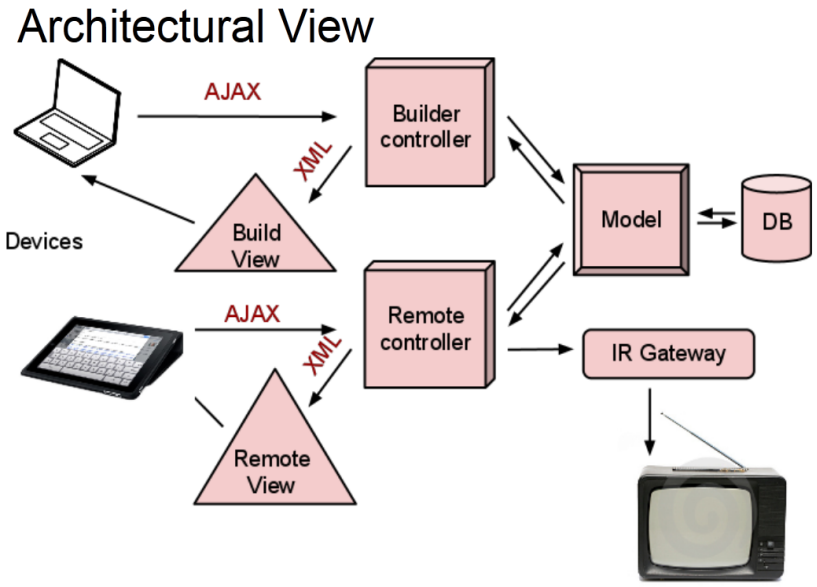
**IR Learner (IR learn mode) -** A feature that allows the user to add IR codes from existing remotes that are mote already in the Dynamote IR Database.

**Wizard -** Set up the remote using a wizard style interface, that asks the user questions to determine what type of layout they need.

## Architectural Diagram:

Dynamote uses a MVC architecture with slight modifications. The application

is hosted by a Ruby on Rails server with a MySQL back end. The HTML view is generated by .erb files, and uses javascript and jQuery UI for client side features. When using the Remote Builder side of the application, the browser talks to the Builder Controller through AJAX requests, which queries the database through the model to retrieve or store any necessary information. Information from the model is returned to the client by the controller as XML data, which is parsed on the client side into the builder layout. Looking from the Remote View side, from a smartphone or tablet, the device sends a request to the server for the remote view, which is assembled from the database into XML and returned to the client. When the user presses a button on the remote, an AJAX request is sent to the server. The controller gets the IR code corresponding to the button action from the database, and sends the code over ethernet to an IR gateway, which sends it to the device being controlled.

**Design** 

**&**

## Implementation Challenges:

We encountered several challenges with this project. One of the largest

challenges we faced was our choice of languages and frameworks to use. We debated for a while between using Google Web Toolkit with Java, or Ruby on Rails with jQuery, etc. We ended up deciding to use Ruby on Rails with MySQL, and jQuery UI. While these technologies were well suited to the needs of our project, none of us were familiar with any of the languages we used, except for a little JavaScript and SQL. This meant that we had to spend some time at the beginning going through language tutorials before we were able to write any code. We also had some troubles setting up the language runtimes and development environments on our computers, since we had Windows, Mac, and Linux users in our group.

When designing the project we had several important and difficult decisions to make. We had to decide how to design our account permissions to give the users enough power, but not to give too much power to change the system set up to just basic users.

We had to decide when implementing the project how to take the remote data from the client's browser and store it in the database, and then what format to send the buttons back to the mobile interface to be displayed on a smartphone or tablet. We decided to use XML to send the data and parse it on the client side into the HTML layout. We also had some problems implementing the actual hardware communication to send IR codes to the user's devices. We use a device called and iTach, which communicates over ethernet and translates commands it receives into an IR signal. To communicate with the iTach, we open a TCP socket and send a hex code to the device, but we had a problem getting the iTach to send a response to our command.

## Changes to Project:

There were several features listed in our proposal that we didn't have time to implement, or went beyond the scope of the project. We originally planned to include a Button Builder as part of the Remote Builder, which would have allowed users to design and save their own custom remote buttons from within the application, in addition to using the built in buttons. This feature we simply didn't have enough time to implement.

We also planned to have a Setup Wizard to assist the user the first time they used the application. The Wizard would guide the user through setting up accounts for other users and adding the devices in their system to their list of devices. This also didn't get implemented due to lack of time.

IR Learn Mode would allow the user to add devices that we don't support by default by teaching the software the IR codes. This is done by pointing the remote for the device at a sensor connected to the server, which reads the IR signal from the remote and translates it into a hexadecimal code, which it could then store in the database for future use. This feature went a little beyond the scope of the project because of the additional hardware involved (an IR sensor on the server), and the extra time required to implement it.

Finally, we were also planning to implement macros, which allow the user to map several IR commands to the same button. For instance, the user could create a macro to power on a TV, BluRay player, and sound system, and set them to the correct inputs all with one button. This feature we started to implement, but we ran into implementation difficulties that would have caused it to take too long to complete. Our proposed interface was to drag a Macro Button onto the remote builder panel and then drag the buttons for the commands in the macro on top of that button, but dragging

buttons onto other buttons proved too complicated to implement in JavaScript and jQuery given the amount of time we had left.

## Design & Implementation Responsibilities:

**Sylvain Heiniger:**

- Wrote client side JavaScript and jQuery for the builder interface.
- Wrote XML parsing/remote display code.
- French translations

**Ryan Sanders:**

- Wrote user accounts code
- Wrote login code
- Spanish and Pirate translations
- Themes

**Stephen Brossart:**

- Database design
- Communication between the model and controller
- Image loading code

**Mike Carter**

- Database Design
- Database Setup
- Hardware communication

**John Ryan**

- Builder and accounts integration
- Site Layout
- Inter-module Code