Mote User Manual

Table of Contents

Disclaimer

THIS SOFTWARE IF PROVIDED BY WORRIED CAT LLC "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL WORRIED CAT LLC OR ITS MEMBERS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; BUSINESS INTERRUPTION; DAMAGE TO PROPERTY; AND, TO THE EXTENT PROVIDED BY LAW, PERSONAL INJURY) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

THIS PRODUCT IS NOT INTENDED FOR USE IN THE OPERATION OF ANY EQUIPMENT IN WHICH THE FAILURE OF THE PRODUCT COULD LEAD TO DEATH, PERSONAL INJURY OR SEVERE PHYSICAL OR ENVIRONMENTAL DAMAGE.

1. Introduction

Mote turns your iPhone, iPad, or iPod touch into a fully programmable Wi-Fi remote. Use your iPhone/iPad/iPod touch to control any ethernet-enabled device which communicates via TCP, UDP, HTTP, or HTTPS. ***Mote*** is programmable directly on your iPhone/iPad/iPod touch, allowing you to layout buttons the way you want, with full macro functionality.

Mote is designed primarily for the home automation do-it-yourself hobbyist. iPhones, iPads, and iPod touches do not have IR transmitters. Additional hardware is required for most home theater applications, though some home theater devices are Wi-Fi enabled. ***Mote*** has applicability beyond home automation. Any device that can be controlled via TCP, UDP, HTTP, or HTTPS should be controllable using ***Mote***.

Mote does provide specific support for the SQ Blaster, a hardware IR blaster with a built-in IR code database for most home theater devices, as well as Global Cache GC100 and iTach devices. Throughout the manual specific instructions are provided when using an SQ Blaster or Global Cache device.

This manual is for the latest version of ***Mote***. Some features are not available in earlier versions. All figures are for an iPhone; the iPad interface will look somewhat different.

NOTE: ***Mote*** sends commands via wi-fi. It should not be used for applications where loss of wi-fi (or a bug in the code) could cause irreparable harm to life or equipment. Please refer to the disclaimer.

2. Basic Concepts

The following basic concepts are central to understanding and programming ***Mote***:

- **Controller** — A controller is a piece of hardware which responds to commands sent via one of the supported protocols (TCP, UDP, HTTP, HTTPS). A controller can be a computer running an appropriate server, or a standalone piece of hardware with a wired or wireless ethernet connection. ***Mote*** talks directly to controllers over the iPhone/iPad/iPod touch's wireless ethernet connection. Controllers then relay commands sent from ***Mote*** to those devices which they control.

- **Device** — A device is a piece of hardware which you wish to control, such as a television, DVD player, a lamp, etc. Commands sent by ***Mote*** to a controller are relayed by that controller to one or more devices. In some cases, a device may be the same as it's controller. For example, a TiVo DVR can receive Wi-Fi commands directly, and thus be it's own controller.

- **Device Group** — For display purposes, devices may be logically grouped. For example, you may wish to group related devices by their function (i.e., "Home Theater") or location (i.e., "Living Room").

- **Activity** — An activity is any activity you may perform which involves one or more devices, but which logically involves one set of controls. An example of an activity is watching a movie. This may involve your television, DVD player, receiver, and room lights. You'd like to control all devices involved in that activity from a single set of controls. An "activity" allows you to do so.

- **Activity Group** — For display purposes, activities may be logically grouped. For example, you may wish to group related activities by their function (i.e., "Home Theater") or location (i.e., "Living Room").

- **Room** — You may further group controllers, device groups, and activity groups by the room they are located in. If you choose, rather than grouping by room you can group by whatever makes sense to you.

- **Command** — A command is a single command sent to a controller, which is then typically relayed by the controller to a device. For example, a command might be to turn a television on.

- **Control** — A control is any interface element used to send a command or sequence of commands. ***Mote*** supports four types of controls: buttons, sliders, timers, and switches. A button is like any button on a typical remote control, except that a button in ***Mote*** can send either a single command or a sequence of commands. For example, to start up your home theater to watch a movie, a button might send a sequence of commands to turn on your receiver, set its input to the BluRay player, turn on the television, turn on the BluRay player, and dim the room lights. A slider allows a user to select a value from a range of values, and send commands with that value embedded in the command. A timer is similar to a slider, but sends time values. Switches allow you to send two different sequences of commands, depending on whether the switch is in the on or off position. Most users will only require buttons. Any control can be programmed to automatically send a sequence of commands at regular intervals (referred to as "polling"). Further, if a command returns a response, any control can parse and display that response, either as a label on a button, or as the position of the slider, timer, or switch.

- **Panel** — A panel is a single set of controls on the display of your iPhone/iPad/iPod touch. Each activity and device can have multiple panels of controls, in both portrait and landscape orientation.

- **Gesture** — Besides sending command sequences using controls, you may also send command sequences using gestures. This is particularly useful when looking at the display for the appropriate control may be inconvenient. The supported set of gestures includes swipe right, swipe left, swipe up, swipe down, and shake. Each device and activity has its own sent of command sequences assigned to each gesture.

- **Template** — A template is a pre-defined panel of controls which can be copied to a blank panel for an activity or device. Templates ease programming by provided standard button configurations. Templates are fully programmable.

- **Template Group** — For display purposes, templates may be logically grouped. For example, you may wish to group templates for all devices from the same manufacturer.

The combination of device/activity groups and rooms provides a 2-level organizational hierarchy. For example, in Figure 1 we have four rooms listed along the tab bar on the bottom of the screen. Just click on the room whose activities/devices you wish to view. In the figure, we have rooms "Den", "Master BR", "Guest BR", and "Kitchen". Selecting "More" will bring up a list of your additional rooms. Currently we have selected the room "Den", and are displaying its activities using two activity groups, "Home Theater" (containing activities "Listen To Music", "Watch TV", and "Watch Blu Ray") and "Environment" (containing activities "Lights", "Curtains", and "Heat").

Most users probably don't need two levels of organization, in which case you may use either groups or rooms, depending on which interface you prefer. By default you start with a single room (named "House"). If you have only one room then the room tab bar at the bottom of the screen will not be displayed. To organize using just groups, keep that single room. To organize using rooms, just use a single unnamed activity group and device group in each room. "Rooms" need not be actual rooms. They may represent whatever organizational structure you prefer.

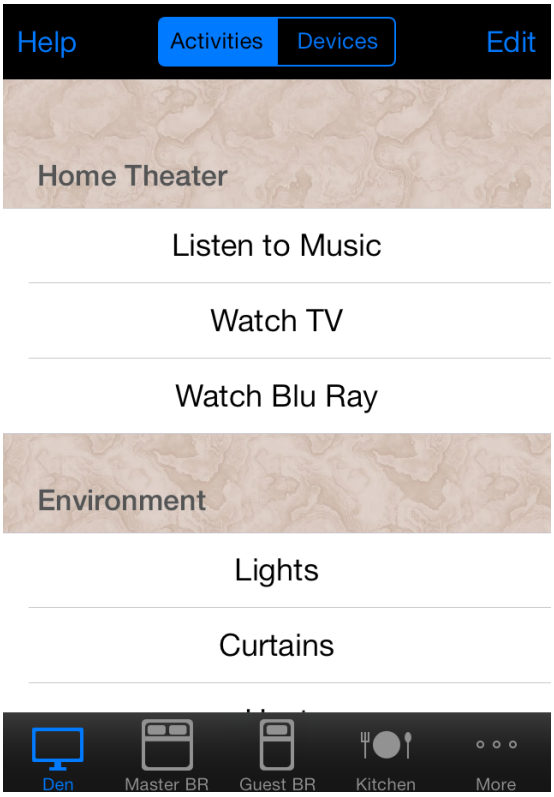

Figure 1.

3. Programming Mote

You must, of course, program ***Mote*** before it can do anything useful. This chapter describes how to program ***Mote***. The typical order to follow when programming is:

1. Select your theme. A theme simply defines the default appearance of your controls (buttons, sliders, timers, and switches) and the background control panels. You may change your theme at any time. However, since your theme specifies the default color used when creating new controls, you'll want to choose the default color before creating all of your controls.

2. Set up your rooms. You need do this only if you wish to use rooms to organize your controllers, devices, and activities. Initially you will be set up with a single room. If you don't need rooms, just stick with the single default room.

3. Define your controllers. You can't define a device without first defining its controller, and you can't define commands without defining the devices you'll send them to, so the first step is to define your controllers.

4. Define your device groups. You must define at least one device group per room to put your devices in.

5. Define and program your devices.

6. Define your activity groups. You must define at least one activity group per room to put your activities in.

7. Define and program your activities.

To program ***Mote***, press the "Edit" button on the home screen (Figure 2). (Note: if the "Edit" button is not present, then turn on the "Programming Mode" option in the ***Mote*** settings menu in the system "Settings" application.) The home page for programming mode will appear (Figure 3). If you've specified a programmer password, you will be prompted for the programmer password before entering programming mode.

Most options are self-explanatory. Just press the appropriate button to edit your rooms, controllers, device groups, devices, activity groups, activities, and theme. The "More" button presents a second page of programming options (Figure 4). The second page allows you to program your templates and template groups, save and restore your programming (discussed later), and set passwords.

Figure 2.

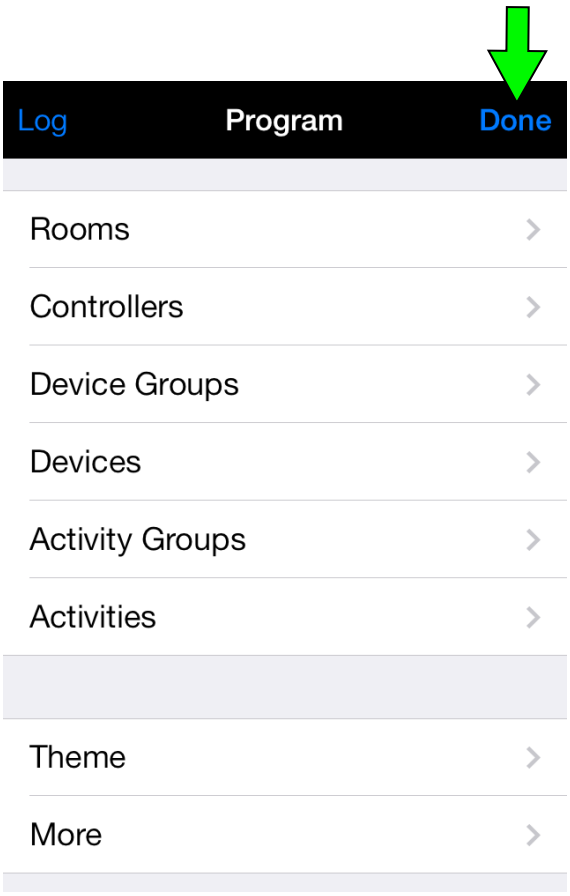

Figure 3.

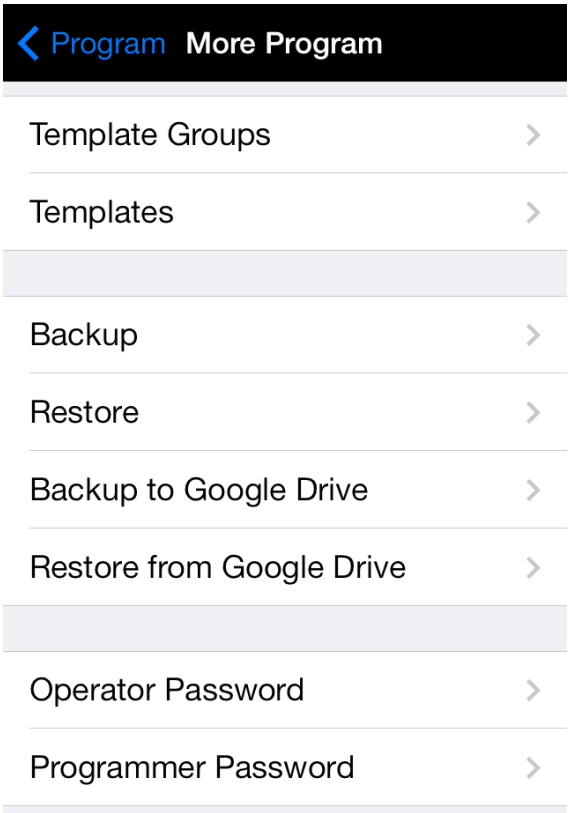

Figure 4.

Most users won't need passwords, however the option is available. Two passwords are available:

- **Operator Password** — If this password is set, then it must be entered to operate ***Mote***. You will be prompted for the password whenever ***Mote*** starts up, or resumes operation in the foreground.

- **Programmer Password** — If this password is set, then it must be entered to program ***Mote***. You will be prompted for the password whenever you press the "Edit" button on ***Mote's*** home page (unless you previously entered the password, and ***Mote*** has not quit or gone into the background since then). If you are programming ***Mote*** and it is forced to the background or your iPhone/iPod is put to

sleep, then you will automatically be exited from programming mode if this password is set.

To unset a password, simply enter a blank password.

WARNING: Use passwords with caution. If you forget a password, there is no way to recover it, and you will not be able to operate and/or program ***Mote***.

The “Log” button (upper left hand corner on the main “Program” page) will display the 20 most recently transmitted commands, along with their responses and any errors. This is provided primarily to help in debugging programming, so that one can see the fully parsed command and its response. Any terminating newlines or carriage returns are explicitly indicated using “\n” and “\r”, respectively. If either a command or its response contains binary characters, then the entire string will be printed out as a hexadecimal string, enclosed within angle brackets (<...>) to indicate the string is binary encoded. Each byte is printed as a 2-digit hexadecimal string (00 - ff), with four-byte groups separated by a space to ease readability. For example, if a command received a binary response consisting of the six bytes 0x00, 0x14, 0xa6, 0xc3, 0x01, 0x00, it would be displayed as “<0014a6c3 0100>”.

When finished programming, touch the “Done” button (Figure 3), which returns you to ***Mote’s*** home page.

The “Help” button on the home page displays help information, including a “User Manual” button that displays this manual.

3a. Themes

Press the “Theme” button on the programming home page to edit your theme. The theme editor will appear (Figure 5). You can define your theme for your control panels and the home page.

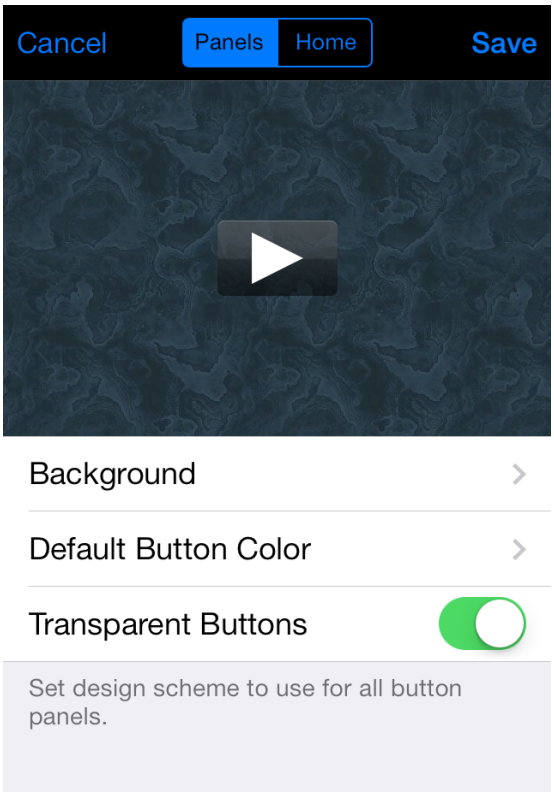

Figure 5.

Control Panel Theme

The control panel theme (Figure 5) is defined by:

- **Background** — The background for button panels. Pressing the “Background” button will offer you a selection of various backgrounds, including your own custom backgrounds if they were previously loaded.
- **Default Button Color** — Newly created buttons will be this color, however you can change an individual button’s color at any time. Black or gray buttons work best. The more colorful buttons are really intended for the occasional button you may want to be obvious. You can also choose clear buttons (the last color choice, indicated as a dim white square), in which case only the button icons and labels will appear on a panel.
- **Transparent Buttons** — If off, then buttons are completely opaque; if on, then they are partially transparent, lending them a glass-like appearance.

The top of the window offers a preview of your selected theme, so you may try out various themes and see how they appear.

The default theme consists of opaque black buttons on a gray leather background. You can

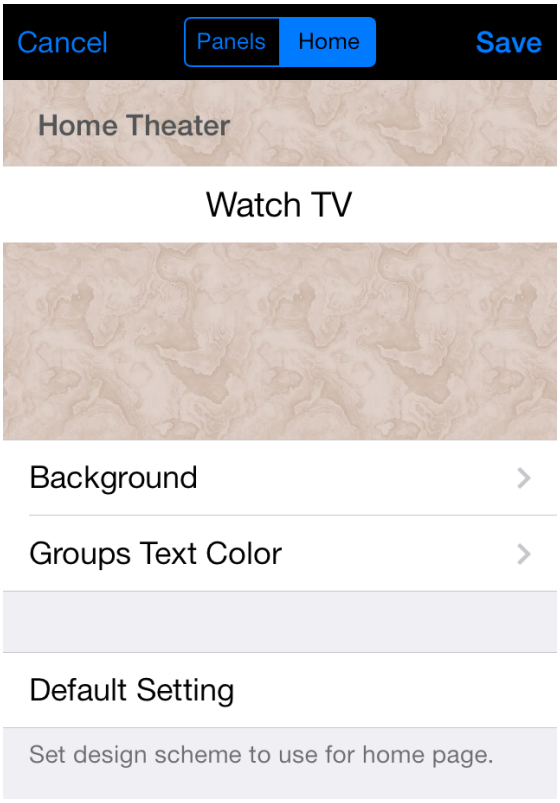

Figure 6.

change the theme at any time. The theme applies to all panels for all devices and activities.

Home Page Theme

The home page theme (Figure 6) is defined by:

- **Background** — The background for the home page. Pressing the "Background" button will offer you a selection of various backgrounds, including your own custom backgrounds if they were previously loaded.

- **Groups Text Color** — The color for the groups label (e.g., "Home Theater" in Figure 6). Different colors stand out better on different backgrounds.

The "Default Setting" button will return you to the default theme, which is gray labels on a brown marble background.

Custom Backgrounds

You can install your own custom background images, which will then be available for both the control panel and home page background. The following rules apply:

- Your images must be png files.

- The file name must begin with "p_" (for portrait orientation) or "l_" (for landscape orientation). For example, if you had a background image of grass, you would name the portrait orientation image "p_grass.png" and the landscape orientation image "l_grass.png" ("p_grass@2x.png" and "l_grass@2x.png" for a retina display device — see below). It is not necessary to provide both portrait and landscape orientation images; if only one is provided, then it will be used in both orientations, rotated appropriately.

- The maximum image size in either dimension is 600 pixels for a non-retina display iPhone (1200 pixels for a retina display iPhone), and 1024 pixels for a non-retina display iPad (2048 pixels for a retina display iPad). Images for retina displays need to have twice the resolution as images for non-retina displays. There is no minimum size. If the image is smaller than the display size in one or both dimensions, then it will be tiled to fill the display. For reference, the size of the display for the non-retina display iPhone 4 is 320 x 480 pixels (640 x 960 pixels for the retina display iPhone 4), and for non-retina display iPads is 768 x 1024 pixels.

- Images must be named using the ".png" extension. For retina display devices, they must be named using "@2x" before the extension. That is an Apple convention for identifying high-

resolution images. Thus, for a retina-display device, you would name the previous example images as “p_grass@2x.png” and “l_grass@2x.png”.

- Images intended for a non-retina display will show on a retina display, however since they are at a lower resolution than the display they will appear blocky.
- The file name must not use the name of one the provided backgrounds (after stripping off the “p_” and “l_” prefix and the “.png” and “@2x.png” suffix). Thus, the following names can not be used: leather, stucco, green marble, brown marble, aluminum, tile, asphalt, concrete, wood, old wood, parquet, fabric, water, clouds, dark gray, gray, light gray.

To upload a custom background image into ***Mote***, use iTunes File Sharing. Start up iTunes, and connect your device. Click on the device in iTunes to display its properties. Click on the “Apps” tab. Under “File Sharing”, select ***Mote*** from the list of apps. Below the panel entitled “***Mote*** Documents”, click the “Add...” button to add one of your custom images.

3b. Rooms

Press the “Rooms” button on the programming home page to edit your rooms. You will be presented with a list of your rooms with the familiar iPhone editing controls to add, delete, edit, or re-order your rooms (you may delete only those rooms which don’t contain any controllers, devices, or activities). The order in this list determines the order of rooms along the tab bar in the main window. Editing or adding a room will present the room editor window (Figure 7).

Each room is characterized by the following attributes:

- **name** — The room name.
- **image** — An image used to represent the room.

Each room must have at least a name or image, and can have both.

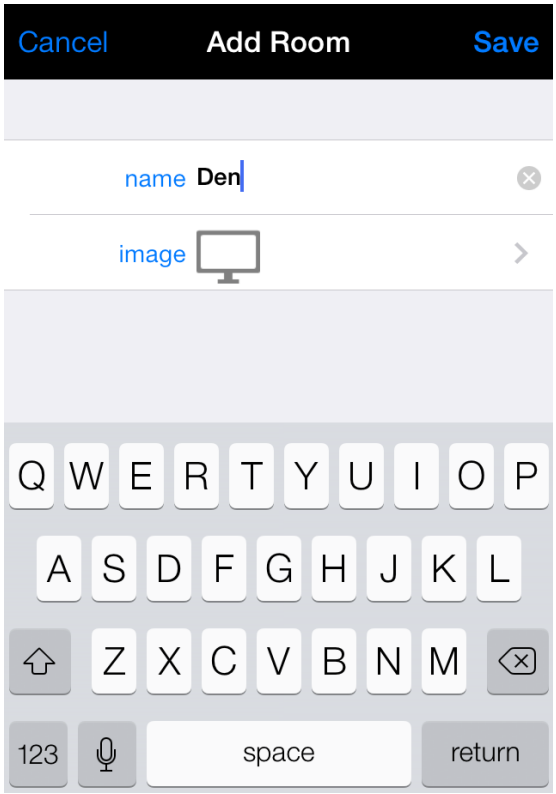

Figure 7.

3c. Controllers

Press the “Controllers” button on the programming home page to edit your controllers. You will be presented with a list of your controllers with the familiar iPhone editing controls to add, delete, edit, or re-order your controllers. Editing or adding a controller will present the controller editor window (Figure 8).

Each controller is characterized by the following attributes:

- **name** — The name of the controller will be used when assigning devices to controllers.
- **protocol** — The protocol used to communicate with the controller. The supported protocols are “TCP”, “TCP no response”, “UDP”, “HTTP”, “HTTPS”, “SQ Blaster”, and “Global Cache”. The “TCP” protocol waits for a response after each command (which the user has the option of ignoring or displaying separately for each command). The “TCP no response” never waits for a response.
- **IP address** — The IP address (i.e., 10.137.248.2) or hostname (i.e., “www.mycontroller.net”) of the controller.
- **port** — The port number that commands will be sent to on the controller. For example, the default port number for HTTP is 80.

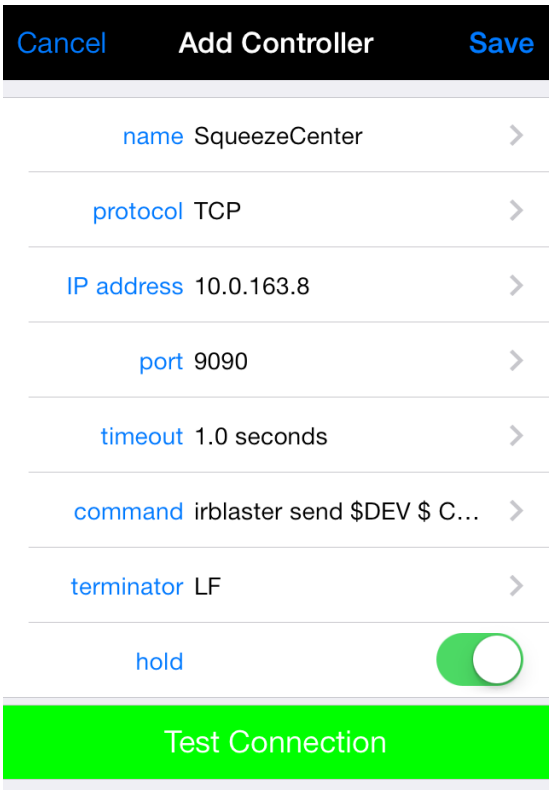

Figure 8.

- **timeout** — The timeout period when sending a command to the controller. Any command which has not completed within the timeout period will be considered to have failed, and the command sequence will be aborted. The timeout period is ignored for devices using the UDP protocol.
- **command** — The default command string sent to the controller. This is described in more detail below (this will not appear for the “SQ Blaster” protocol).

The following two attributes further apply to the “TCP”, “TCP no response”, and “UDP” protocols:

- **terminator** — Terminate all command strings with these characters. The available options are “LF” (linefeed), “CR” (carriage return), “LF + CR” (linefeed followed by a carriage return), “CR + LF” (carriage return followed by a line feed), “HomeVision”, and “none”. The “HomeVision” option is specifically for HomeVision controllers, which use different terminators for sending commands and receiving responses. If you require a terminating character other then one of the supported options, then set this to “none” and add the terminating character to the controller’s default command string (see below).

- **hold** — If ON, hold the socket connection open continuously. If OFF, open the socket connection before each command is sent, and close it immediately after each command is sent.

SQ Blasters

The “SQ Blaster” protocol supports SQ Blaster devices from Square Connect. There is no default command string for SQ Blasters, as all commands are either chosen from the built-in IR code database or learned using the IR code learning capabilities of the SQ Blaster. Nor is there a timeout parameter. SQ Blasters advertise themselves using bonjour. Thus, when you select the “SQ Blaster” protocol, a “Find SQ Blasters” button will appear. Press it and you will be presented with a list of SQ Blasters on your network. Select the one you want, and the “IP Address” and “port” fields will be filled out appropriately. This is the recommended procedure when using SQ Blasters, to assure that you get the correct IP address of the device.

Global Cache

Mote supports Global Cache’s GC100 and iTach devices. Global Cache devices differ from most other devices, in that they can be communicated with over multiple ports. Most commands, including commands to send IR commands and control and poll relays, led lighting, and digital sensors use port 4998. However, serial commands are sent to ports 4999 and higher (depending on how many serial ports are on the Global Cache device). Thus you may have to define more than one controller when using a Global Cache device. For communicating with port 4998 (to control IR devices, relays, led lighting, and digital sensors), define a controller using the “Global Cache” protocol and port 4998. Using the “Global Cache” protocol is required in this case, as IR commands for Global Cache devices are treated

specially. For communicating with serial devices (port 4999 and higher), you'll need to define an additional controller for each serial port you communicate with. Use the "TCP" or "TCP no response" protocol (depending on the device you're controlling), and port 4999 or higher (refer to the documentation that came with your Global Cache device). Global Cache devices advertise themselves. When using the "Global Cache" protocol, you can press the "Find Global Cache Devices" buttons to find all Global Cache devices on your network. Allow up to 60 seconds for all devices to be found.

Default Command String

The default command string for the controller greatly simplifies how one assigns commands to controls and gestures. One typically sends a formatted command string to the controller which includes the identifier for the device you wish to control, as well as the command to send to that device. For example, if your controller is a Squeezebox Server, you might send the following command to the controller to turn on your receiver: "irblaster send receiver on". In this example command string, the controller is sending the command "on" to the device named "receiver". We can assign that full command string to a button, and it would work. However, that will involve a fair amount of typing when multiplied by the number of buttons that you'll want to define for that device. Rather than having to assign a full command string to each command, you can define a default command string for each controller with place holders for the device identifier and the specific command you wish to send to that device. The strings "$DEV" and "$CMD" serve as the place holders for the device identifier and command, respectively. For the current example, you would set the default command as "irblaster send $DEV $CMD". Then, since each command is assigned to a specific device (which will be described later), you need only assign the command "on" to the button to send the full command.

The default command string is ignored when sending IR commands to a Global Cache controller.

Testing the Controller

The green "Test Connection" button at the bottom of the window allows you to send a test command to your controller to verify that you've entered the correct information for your controller, and that all is working properly. It opens a new window (Figure 9) that allows you to enter a test command. The controller's default command is ignored in this case (since we haven't defined any devices yet), so you must enter the full command string to send to the controller. If the command returns a response, you can display the response as either plain

text or HTML via the “display” button; if the command does not return a response, set “display” to “none”. Click the “Send” button to send the test command.

When connecting to SQ Blasters, you are not prompted for a command string. Pressing the “Test Connection” button simply tests that a connection can be made to the SQ Blaster, without sending an actual command.

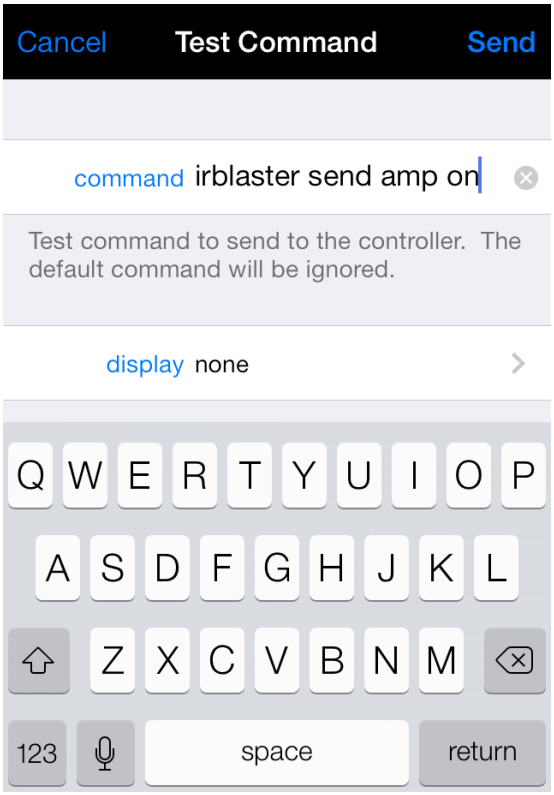

Figure 9.

3d. Device Groups

Press the “Device Groups” button on the programming home page to edit your device groups. You will be presented with a list of your device groups (grouped by room) with the familiar iPhone editing controls to add, delete, edit, or re-order your groups (Figure 10). Only groups that contain no devices may be deleted. Editing or adding a device group simply prompts you to edit the name of the group. A device group need not have a name.

If you’d rather not use device groups, just define a single unnamed device group per room to which you assign all the devices for that room.

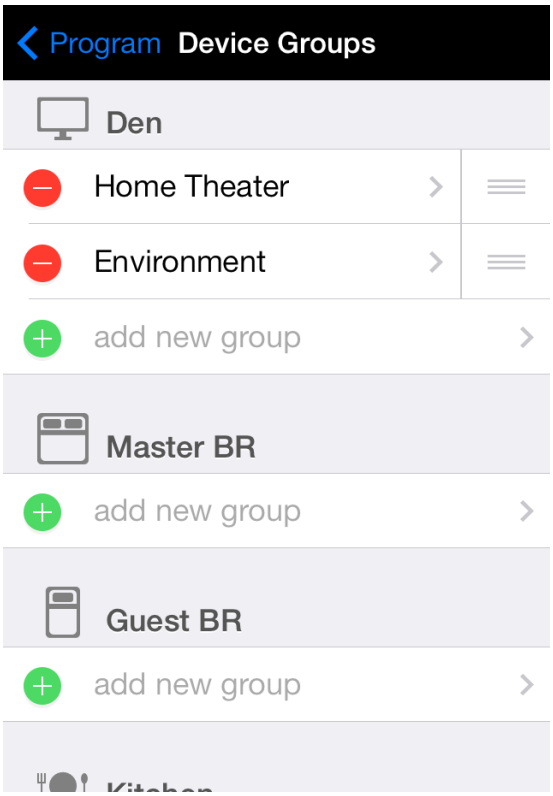

Figure 10.

3e. Devices

Press the “Devices” button on the programming home page to edit your devices. You will be presented with a list of your devices with the familiar iPhone editing controls to add, delete, edit, or re-order your devices (Figure 11). Use the tabs on the bottom of the screen to select the room whose devices you wish to edit (the room tab bar will appear only if you have more than one room defined). To program buttons for the device, click on its row in the list (this will be discussed later). To edit the attributes of an existing device, such as its name, controller, and the commands assigned to gestures for that device, touch the info (i) symbol at the end of the row for that device.

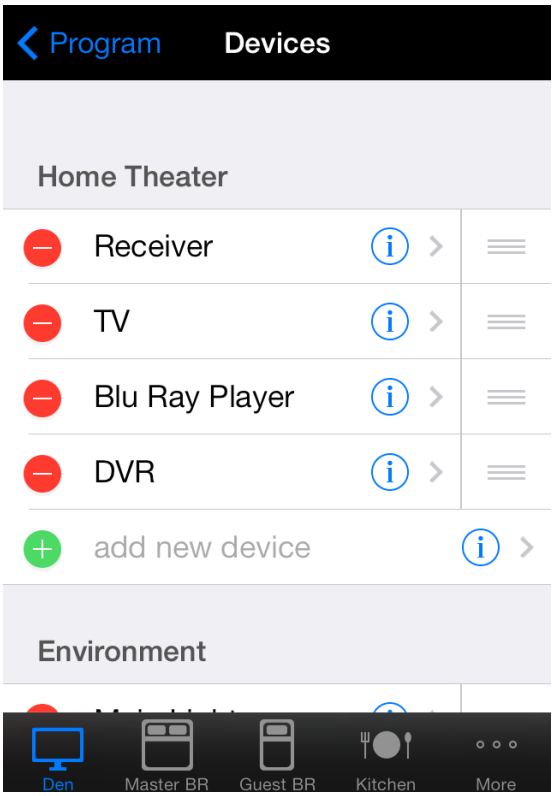

Figure 11.

Editing the attributes of an existing device, or adding a new device, will present the device editor window (Figure 12). Each device is characterized by the following attributes:

- **name** — The name of the device. This name will be used when assigning commands to devices.
- **controller** — The controller used to control this device, selected from the list of currently defined controllers.
- **identifier** — The identifying string by which the controller knows this device. This may or may not be the same as the device name. For some controllers this may be a cryptic identifier, which is why a device has both a name and an identifier. This string is substituted for the string “$DEV” in a controller’s default command string for any commands which target this device (and use the controller’s default command string — more on that later).

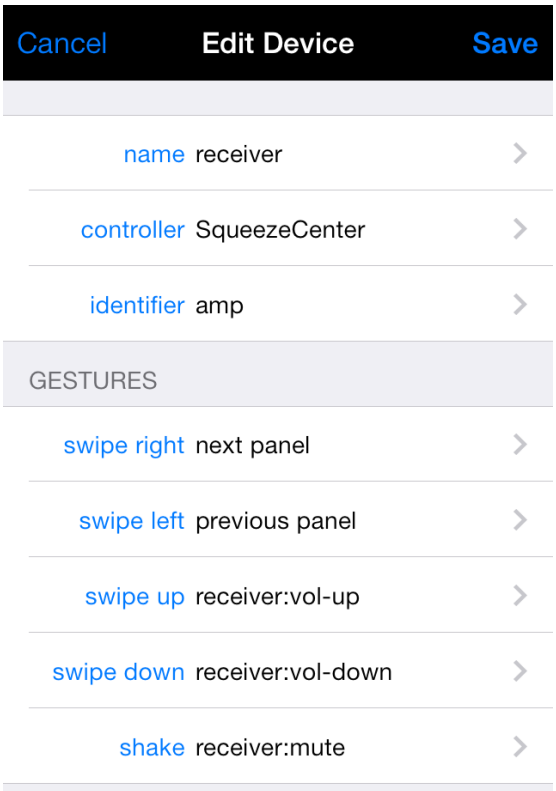

Figure 12.

Gestures

Each device can assign a separate sequence of commands to each of five gestures: swipe right, swipe left, swipe up, swipe down, and shake. Select a gesture to edit its command sequence (editing command sequences will be discussed later). If no command sequence is assigned to the swipe right gesture, then swiping right changes to the next panel of buttons for that device (panels are discussed later). Similarly, if no command sequence is assigned to the swipe left gesture, then swiping left changes to the previous panel of buttons for that device. For the remaining gestures, if no command sequences are assigned to them, then they do nothing.

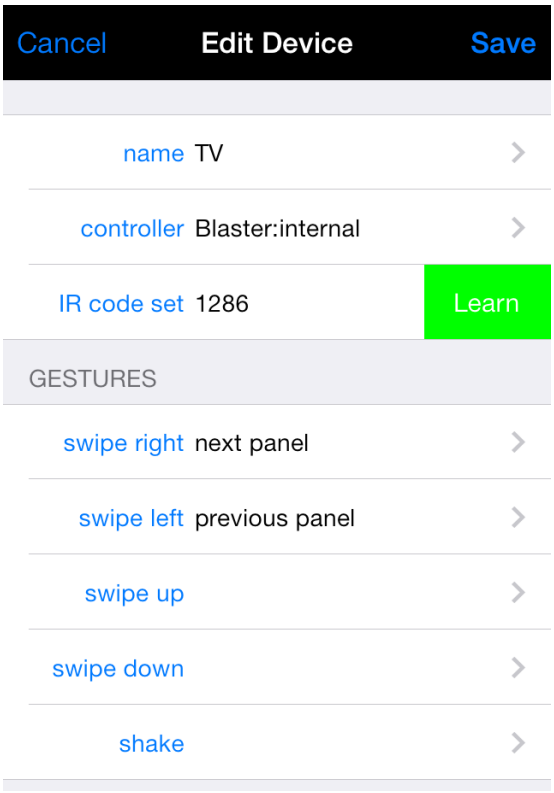

Figure 13.

SQ Blasters

Devices controlled by SQ Blasters are handled differently, as we access their IR commands directly from the built-in IR code database. When the device is controlled by an SQ Blaster, instead of Figure 12 you'll see Figure 13. The "identifier" field in Figure 12 is replaced in Figure 13 with a field labeled "IR code set". There is no "identifier" for a device controlled by an SQ Blaster. Rather, you must select an IR code set for your device. Press the "IR code set" button (the white part of the button, not the green "Learn" button on it, which won't be present anyway if you haven't yet chosen a code set) and you will be presented with a list of brands (Figure 14). Choose the brand for your device. If there is no code set for your device, choose "unknown", in which case you'll have to learn all of the commands. After choosing a brand, you'll be presented with a list of device

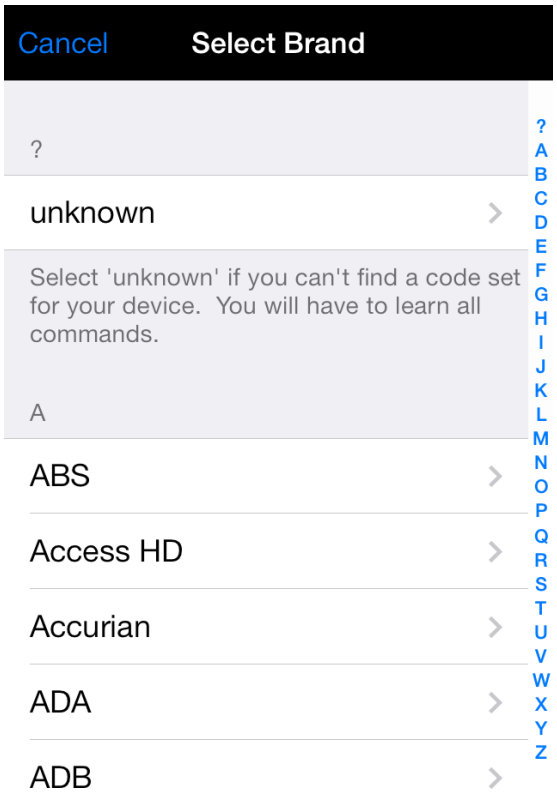

Figure 14.

types (TV, amplifier, etc) for that particular brand (Figure 15). Choose the appropriate device type. You'll then be presented with the first code set for that combination of brand and device type (Figure 16). Different manufacturers use multiple code sets for their devices. You can use the arrow buttons at the bottom of the screen to switch between different code sets. Press the green "Test" button for a command to see if it works; the SQ Blaster will blast that command. When you find a code set that works, press the "Save" button at the top of the screen.

Besides using the built-in IR codes, you can also learn IR codes using the SQ Blaster. Press the green "Learn" button next to the IR code set identifier (Figure 13). You'll be presented with a list of previous learned commands for that device, as well as a button to learn a new command. Press that button and you'll be presented with the learning screen (Figure 17). Enter a command name, which you'll use when assigning that command to buttons and gestures. You can then either enter the IR code directly or, more usefully, use the SQ Blaster to learn the command. Simply hold your remote with the emitter about an inch from the learning port on your SQ Blaster. Press the "Learn" button in ***Mote***, then press and hold the button on your remote that you wish to learn. Keep the button pressed until the code appears on the screen in ***Mote***. See the manual that came with your SQ Blaster for further instructions on learning commands. All commands that you learn are available for assignment to buttons and gestures. To learn a toggle code (the button on the remote toggles between two different codes), use the "Lean" button to learn the command twice (this is an experimental feature —most users won't need this). Use the "Test" button to send the learned command.

The original SQ Blaster had only a single emitter.

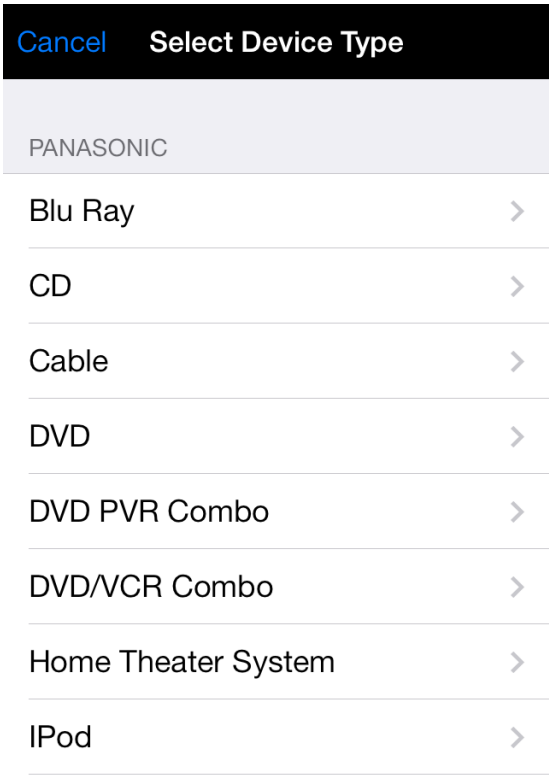

Figure 15.

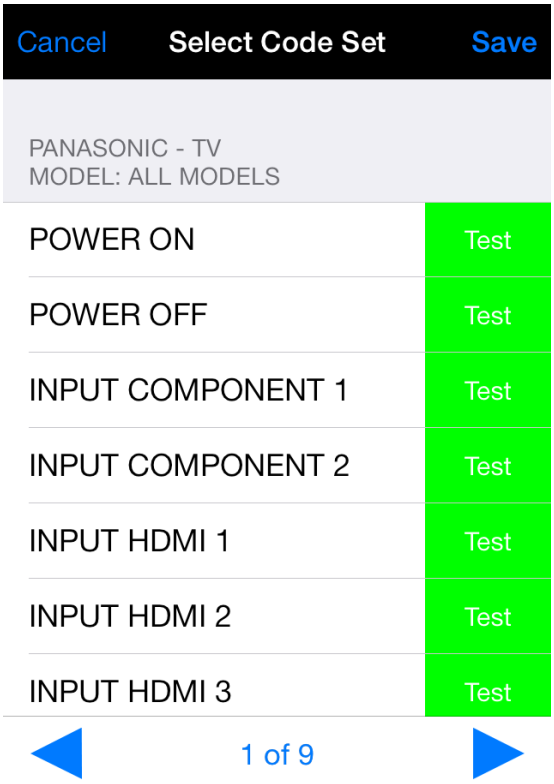

Figure 16.

The SQ Blaster Plus has both an internal emitter, as well as four IR ports to which external emitters may be attached. Thus, when you select an SQ Blaster as the controller, you must also select the emitter to use (either “internal” for the original SQ Blaster or for the internal emitter on the SQ Blaster Plus, or one of port 1 - 4 to use the external emitters on the SQ Blaster Plus).

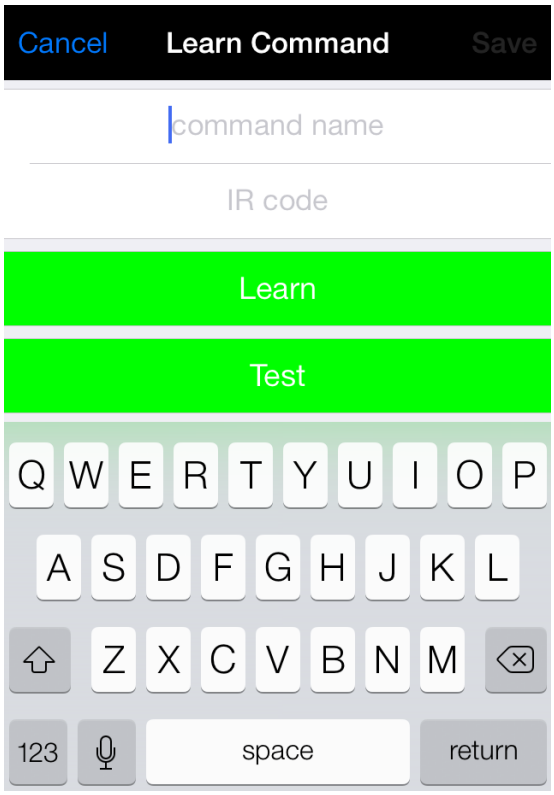

Figure 17.

Global Cache

Devices controlled by a Global Cache controller are also handled differently. When you touch the “identifier” button (Figure 12), you’ll be presented with a screen prompting for the module and connector you’re addressing on your Global Cache product (refer to the documentation that came with your Global Cache product for how its various device ports are addressed). You’ll also be asked to select whether the device is an IR device or not. If the device is an IR device, then a green “Learn” button will appear next to the identifier in the main device editing page (similar to Figure 13, though “IR code set” will instead be “identifier”). Touching the “Learn” button will present a list of previously learned commands, as well as a button to learn a new command. Press that button and you’ll be presented with the learning screen (Figure 17). Enter a command name, which you’ll use when assigning that command to buttons and gestures. You can then either enter the IR code directly or, more usefully, use an iTach IR controller to learn the command. Simply hold your remote with the emitter pointed at the learning port on your iTach. Press the “Learn” button in ***Mote***, then press and release the button on your remote that you wish to learn. Unlike SQ Blasters, pressing and releasing the button works best, rather than continuously holding it. After a second or so, the code should appear on the screen in ***Mote***. See the manual that came with your iTach for further instructions on learning commands. All commands that you learn are available for assignment to buttons and gestures. The button in the upper right hand corner on the screen listing all learned commands allows you to export the commands you have learned to a file, or import a set of previously learned commands. These learned command files can be transferred to and from your iPhone using iTunes File Sharing. Thus, for example, you could learn commands using an iTach IR device, and then transfer them to devices controlled by a GC100 controller (which doesn’t have built-in IR learning capabilities).

3f. Activity Groups

Press the “Activity Groups” button on the programming home page to edit your activity groups. You will be presented with a list of your activity groups (grouped by room) with the familiar iPhone editing controls to add, delete, edit, or re-order your groups (Figure 18). Only groups that contain no activities may be deleted. Similarly to device groups, editing or adding an activity group simply prompts you to edit the name of the group. An activity group need not have a name.

If you’d rather not use activity groups, just define a single unnamed activity group per room to which you assign all of the activities for that room.

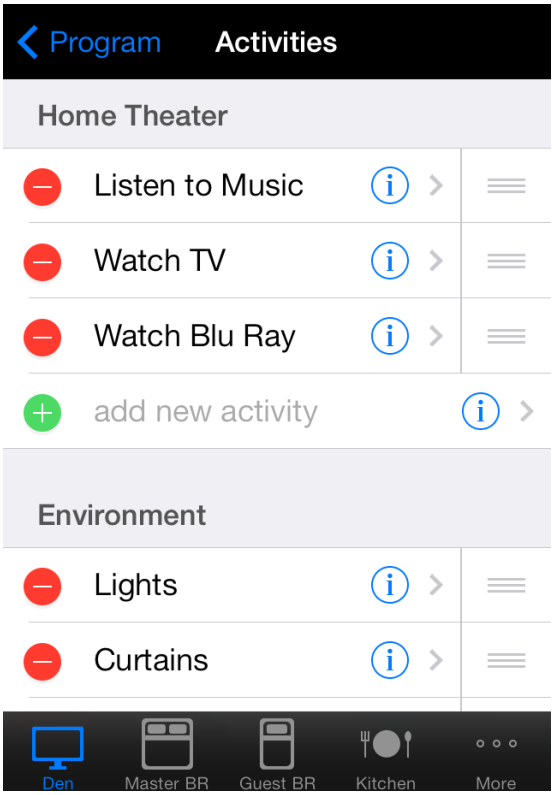

Figure 18.

3g. Activities

Press the “Activities” button on the programming home page to edit your activities. You will be presented with a list of your activities with the familiar iPhone editing controls to add, delete, edit, or re-order your activities (Figure 19). Use the tabs on the bottom of the screen to select the room whose devices you wish to edit (the room tab bar will appear only if you have more than one room defined). To program buttons for the activity, click on its row in the list (this will be discussed later). To edit the attributes of an existing activity, such as its name and the commands assigned to gestures for that activity, touch the info symbol (i) at the end of the row for that activity.

Editing the attributes of an existing activity, or adding a new activity, will present the activity editor window (Figure 20). The only attribute of an activity is its name.

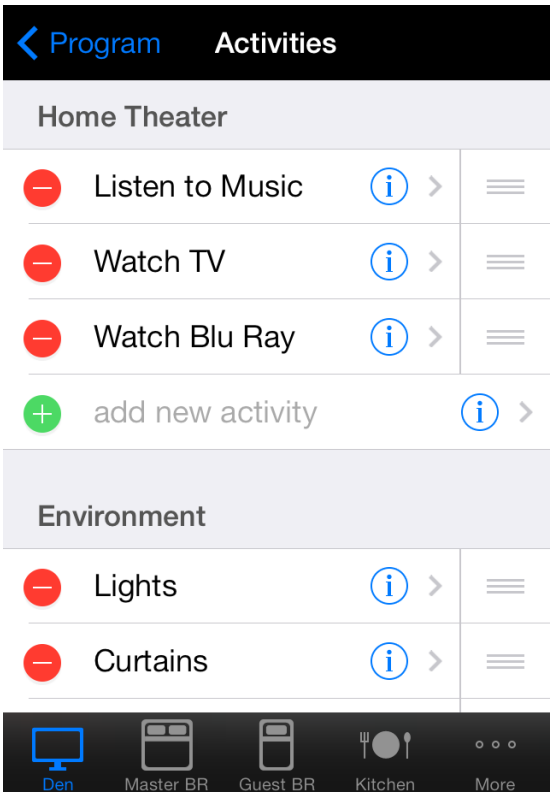

Figure 19.

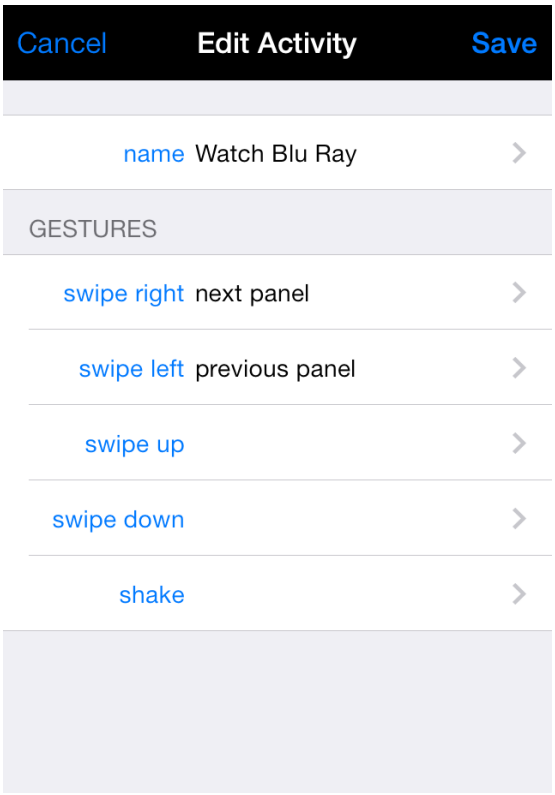

Figure 20.

Gestures

Each activity may have a separate command sequence assigned to each of five gestures, exactly the same as with devices. Similarly as with devices, if no commands are assigned to either the swipe right or swipe left gestures, then those gestures change to the next or previous panel of buttons, respectively, for that activity.

3h. Template Groups

Press the “Template Groups” button on the “More Program” programming page to edit your template groups. You will be presented with a list of your template groups with the familiar iPhone editing controls to add, delete, edit, or re-order your groups (Figure 21). Similarly to device and activity groups, editing or adding a template group simply prompts you to edit the name of the group. A template group need not have a name.

If you’d rather not use template groups, just define a single unnamed template group to which you assign all of your templates.

Mote comes packaged with an initial template group named “Pre-Supplied”. This group contains a single template named “Home Theater”, which offers a number of pre-defined panels of buttons (both portrait and landscape) appropriate for home theater applications.

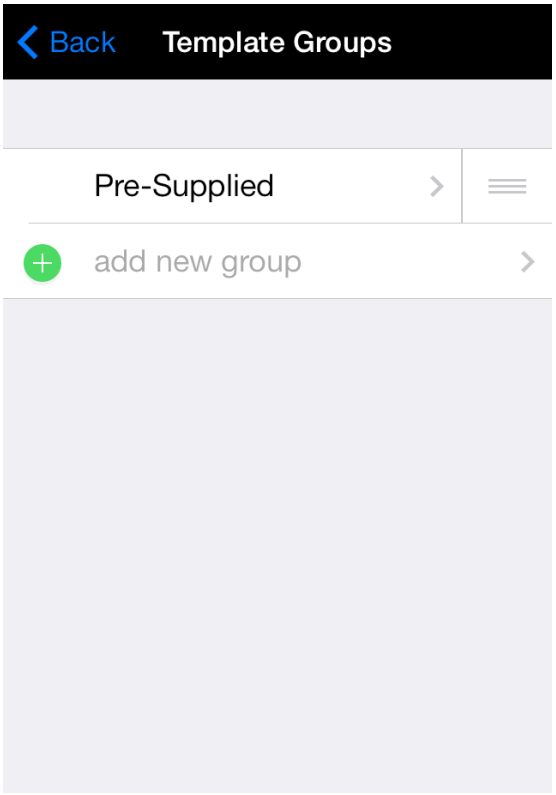

Figure 21.

3i. Templates

Press the “Templates” button on the “More Program” programming page to edit your templates. You will be presented with a list of your templates with the familiar iPhone editing controls to add, delete, edit, or re-order your templates (Figure 22). To program buttons for the templates, click on its row in the list (this will be discussed later). To edit the name an existing template touch the info (i) symbol at the end of the row for that template.

Unlike devices and activities, templates do not have associated gestures.

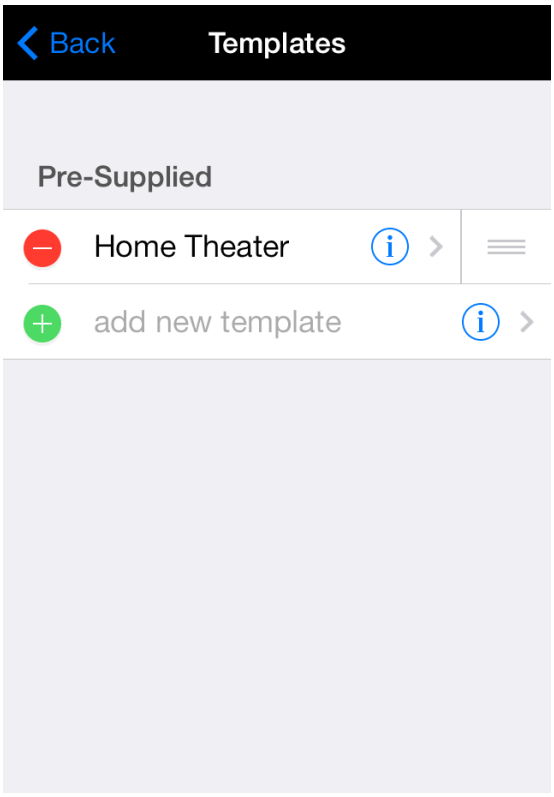

Figure 22.

3j. Controls and Panels

Each activity, device, and template can have multiple panels of controls, in both portrait and landscape orientation (each must have at least one panel in each orientation, even if it has no controls). Selecting an activity/device/template in the activity/device/template list will display its first portrait panel of controls (Figure 23).

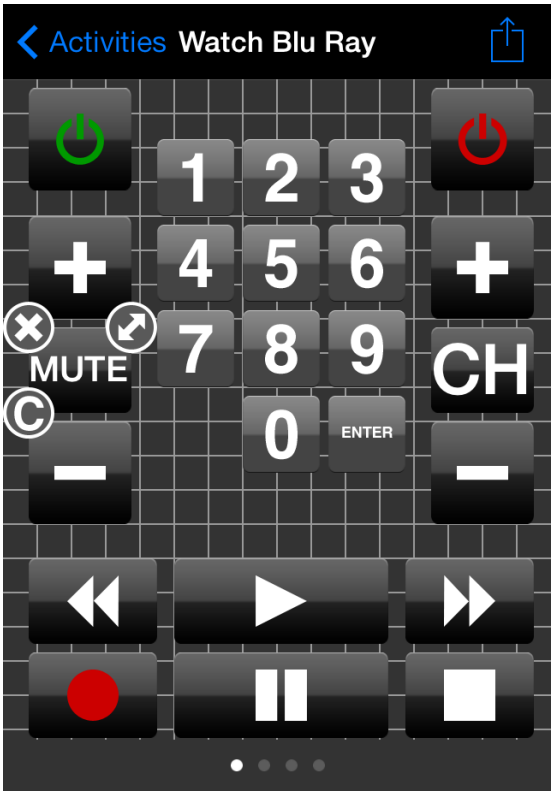

Figure 23.

Controls

To add a control to a panel, just touch the panel where you'd like the control. The button will appear, highlighted with a delete button (X) in the upper left hand corner of the button, a resize button (double-headed arrow) in the upper right hand corner of the button, and a copy button (C) in the lower left hand corner of the button (e.g., see the "MUTE" button in Figure 23). To edit an existing control, just touch it and it will be similarly highlighted with delete, resize, and copy buttons. When a control is highlighted you may:

- Touch the delete button to delete the control.
- Touch and drag the resize button to resize it.
- Touch the copy button to make a copy of the control in the paste buffer.
- Touch and drag the control itself to move it.
- Touch and release the control to edit its properties, including its appearance and the command sequence assigned to it.

To copy and paste a control, highlight it and then touch the copy button to copy it to the paste buffer. Then double tap the screen where you'd

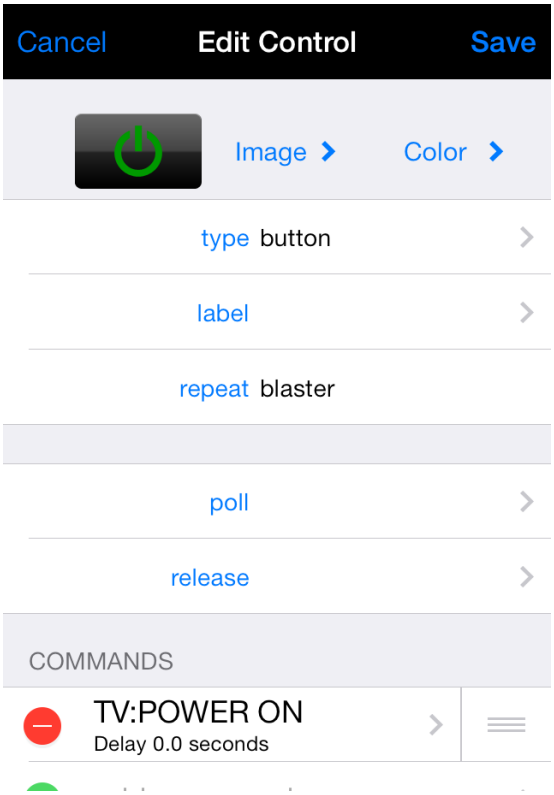

Figure 24.

like a copy of the control, and a copy will be placed there. The copy includes the commands assigned to the original control. If you have not copied a control, then double tapping will not do anything.

When you resize or move a control, its location and size is forced to align with a grid to ease the design of your control layout. Use of the grid can be disabled by setting the ***Mote*** preference "Button Grid" to "OFF" in the system "Settings" application. By default, grid lines are displayed to facilitate button layout; these may be hidden by setting the ***Mote*** preference "Grid Lines" to "OFF".

When you touch a highlighted control to edit it, the control editor window will be presented. The exact appearance will depend on the type of control. ***Mote*** supports four types of controls (most users will only require the "button" type):

- **button** — A button sends a sequence of commands when you press it. The command sequence can be sent once, or at regular intervals while the button is being held pressed. The control editor for a button is shown in Figure 24. The button's appearance is shown in the upper left hand corner, and will be updated as you change its properties. The color of the button can be changed by selecting the "Color" button. A button can have an image, a label, or both assigned to it. Selecting "Image" presents you with a large selection of common images used on remote controls, as well as any custom images you may have already loaded. There are more than one page of such images; swipe right or left, or touch on either side of the page selection indicator on the bottom of the screen to view the different images. When you see the image you want, just touch it to select it. Labels can be static, or can be updated with responses

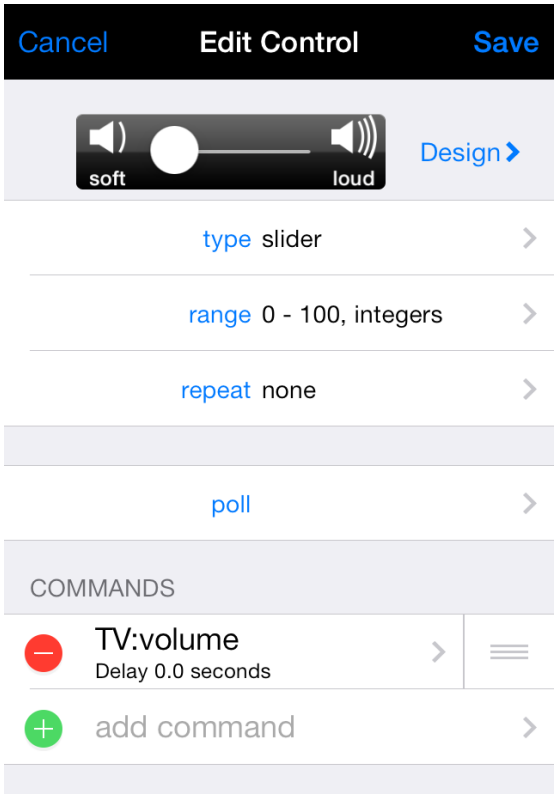

Figure 25.

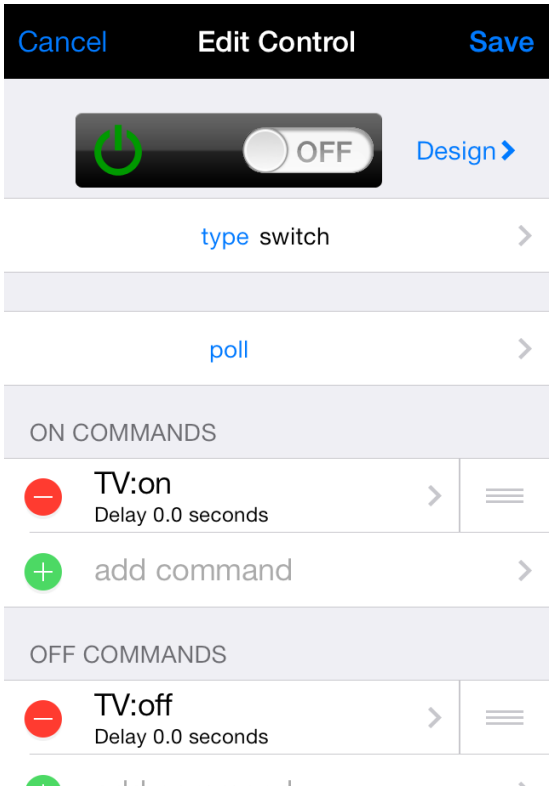

Figure 26.

to queries (discussed later). The “repeat” attribute determines the delay between finishing one command sequence and starting the next. If you do not wish the button to repeat (which is probably true for most buttons), set “repeat” to “none” (the default value).

- **slider** — A slider allows you to select a value from a range of values, and send commands embedding that value (using the $VAL token in command strings, described later). The control editor for a slider is shown in Figure 25. Press the “Design” button to design the appearance of the slider, including its color, the image and label to assign to the minimum and maximum values, whether it is vertical or horizontal, and whether a heads-up-display (HUD) shows the current value as you operate the slider. The “range” button is used to specify the minimum and maximum values of the slider, and whether it uses floating point or integer values. For most uses, you’ll want to set “repeat” to “none”, meaning the command sequence associated with the slider is sent only when the slider is released. You can set it to an interval, in which case the command sequence is sent at regular intervals as long as you’re touching the slider, using the slider value at that time. When repeating, there is a one second delay when you first touch the slider to give you time to initially position the slider before the first command is sent.

- **timer** — A timer is a slider for selecting time intervals. It functions exactly like a slider, where the range of values is in hours. You also specify a time interval, in minutes. The slider can only output values that are integral values of that interval. For example, if you specify a range of 0 to 2 hours and an interval of 15 minutes, then the slider will only output the values 0, 0.25, 0.5, 0.75, 1.0 1.25, 1.5, 1.75 and 2.0. If you use the HUD, the values are displayed in HH:MM format. However the $VAL token is set to a floating pointing representation (i.e., 1.5 rather than 1:30). This is to give the user greater flexibility formatting the output timer interval, as described later in the manual (in the “Commands” section). Most users will not need to use timers.

- **switch** — A switch allows you to send one set of commands when the switch is in the “on” (left) position, and another when it is in the “off” (right) position. The control editor for a switch is show in Figure 26. Press the “Design” button to design the appearance of the switch, including its color, image, label, and the labels for the “on” (left) and “off” (right) position, which replace the usual “ON” and “OFF” labels. Commands are sent only when the switch position actually changes. Switch commands can’t be set to repeat at regular intervals.

Note that the control in the control editor is active. Command sequences will be sent when you interact with the control, and poll commands (discussed later) will also operate. Thus, you can test out the control directly in the editor.

Each control can have a sequence of commands assigned to it, which is executed when you interact with the control. You can add, delete, and reorder commands with the usual iPhone

editing controls. Edit an existing command by selecting it in the list of commands (editing commands is discussed later). Delays may be added between commands. Switches have both an “on” sequence of commands and an “off” sequence of commands. Buttons can also execute a separate sequence of commands when you release the button (programmed using the “release” option — see Figure 24). Care should be taken with release commands, as if you rely on the release command and you loose your wi-fi connection, the release command may not be received. Most users won’t need release commands.

All controls can be used to “poll” devices. Pressing the “poll” button in any control editor will allow you to specify a sequence of commands which are sent automatically by the control at regular intervals. This is typically used to regularly query a device and display the results on the control itself (this will be discussed later). The command sequence is always sent whenever a control’s panel appears. If a repeat interval is specified, then the command sequence will continue to repeat at that interval until the panel is exited. Care should be taken when polling. Controls don’t respond to touches whenever a command sequence is being sent out. Thus, if your polling sends out a sequence of commands with built in delays, they can freeze out your remaining controls. You should typically use polling to send out just a single command which, as long at it responds quickly, should not affect the responsiveness of the rest of your controls. Most users won’t need the polling capability.

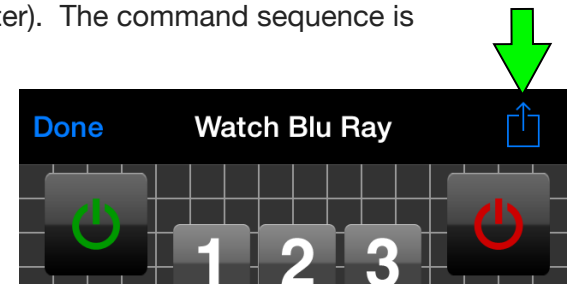

Figure 27.

Panels

Each newly created activity, device, or template has a single portrait panel. You can add, delete, and move panels by selecting the panel button in the upper right hand corner of a button panel (see Figure 27). Doing so displays the list in Figure 28. You have the following mostly self-evident options:

- **Delete This Panel** — Delete the current panel. You must always have at least one portrait panel.

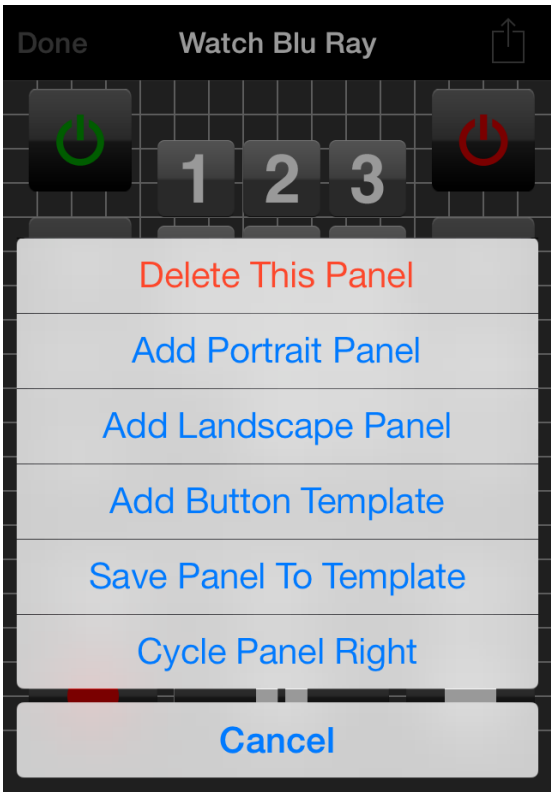

Figure 28.

- **Add Portrait Panel** — Add a new portrait panel.

- **Add Landscape Pane**l - Add a new landscape panel.

- **Add Button Template** — This adds a pre-defined template of buttons to the current panel. This only works if the panel is blank (has no buttons). You will be presented with a list of the templates. Select the desired template, and the first panel for that template will be displayed. If there is more than one panel for that template you can swipe or touch on either side of the page indicator to view the available panels. Press the “Save” button (upper right hand corner) when you identify the panel you want to use. It can be faster to start with a template and change it than starting from scratch. Beware that templates may already have commands assigned to buttons.

- **Save Panel To Template** — Save the current panel as a template panel. You will be prompted with the list of templates. Select the template to which you wish to add this panel to. Note that the commands assigned to the buttons will also be copied to the template.

- **Cycle Panel Right** — Move the current panel one slot right in the list of panels for this device/activity/template. If its already the right-most panel, then it will cycle around to become the first panel.

To switch from the current portrait panel to the first landscape panel, or from the current landscape panel to the first portrait panel, just rotate your iPhone/iPod to the appropriate orientation.

3k. Commands

When you select a command in a list of commands, or select the “add command” button while editing a button or gesture, the command editor window appears (Figure 29). Each command has the following attributes:

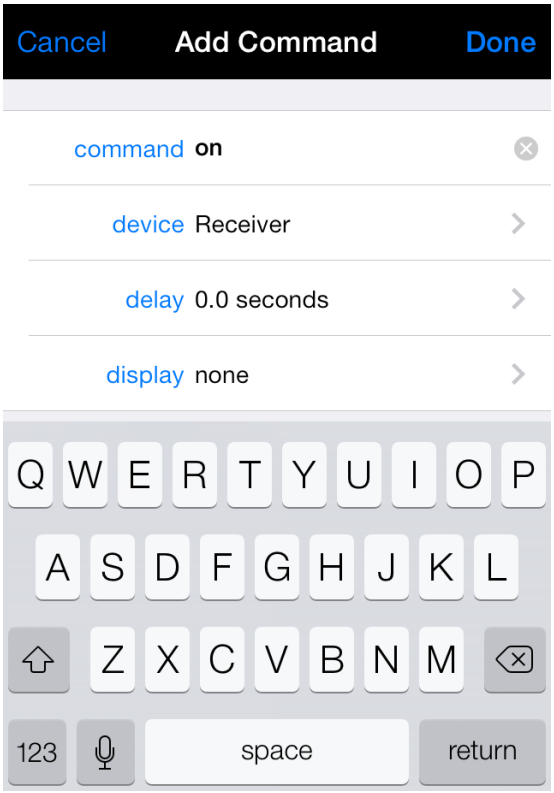

Figure 29.

- **command** —The command string to send. The format of this string is discussed below.
- **device** — The device to send the command string to, selected from the list of currently defined devices. The command string is actually sent to that device’s controller, which in most cases will then relay the command to the device. You can also choose to launch a different application, rather than sending a command to a device, or send a Wake-on-LAN signal; see below for details.
- **delay** — Wait the specified delay time (in seconds) before sending the command. Delays are useful when sending a sequence of commands. For example, say you have a sequence of commands which first turns on your receiver and then sets its input to the DVD player. You may wish to delay the command which selects the input until the receiver has a chance to turn on so it is ready to respond to subsequent commands. If you do not require a delay, then select “0.0 seconds”. Most single commands will not require a delay.
- **display** — (NOTE: for IR devices controlled by an SQ Blaster or Global Cache controller, this field will be labeled “IR repeat”, which is described below). Most commands are one-way commands. That is, you send a command without expecting a response. However, if a command returns a response then you can display it immediately via a pop-up window, or by updating a button label or the position of a slider or switch. If displayed using a pop-up window and the command is part of a sequence, the sequence will be suspended until you dismiss the pop-up window, at which time the sequence will resume. Binary responses are displayed in the same format as displayed in the “Log” window. The following options are available:
    - **none** — Don’t display any response. Most commands will use this option.

- **text** — Display the response as simple text in a pop-up window.
- **html** — Display the response as HTML in a pop-up window.
- **$RSP** — Update the button label or slider/switch position using the response.

The “$RSP” string can be edited to parse the command appropriately. Any occurrences of the string “$RSP” in “display” will be replaced with the command response. You can display just a portion of the response using the string “$RSP:X:Y:”, where Y is a single character used to split tokens in the response, and X is a number between 0 and 9 specifying which token to display (0-indexed). This is best explained using an example. Say you query a temperature controller which returns four temperatures, separated by commas. For our example, say the response is “32.6,38.3,40.2,29.7”. If you want to display the entire response as is, set “display” to “$RSP”. If you want to display just the second temperature (38.3), set “display” to “$RSP:1:,:”. The “1” says to display the second token (we’re 0-indexed), and the “,” is the separator between tokens. Leading and trailing spaces around a token will be stripped before displaying it. If the tokens were separated by spaces, then you would substitute the comma in “display” with a space. This example would also work with a slider; the slider position would be set to 38.3. Alternatively, you can parse the response by selecting a substring based on character position, using the nomenclature “$RSP[X:Y]”, where X and Y are the first and last characters to display, respectively (0-indexed). Thus, for the previous example, “$RSP[10:11]” would return “40”. You can combine the token and character based parsing. For example, “$RSP:1:,:[0:1]” would return “38”. Finally, you can embed the response within a larger string. For example, if you wanted the button to display the string “Temp: 38.3 deg F”, you would set “display” to “Temp: $RSP:1:,: deg F”. For switches, you need to parse the string to the point where it is equal to one of the values associated with the “on” (left) or “off” (right) position of the switch, as set using the buttons “left value” and “right value” when editing the polling for a switch.

$RSP can be replaced with $HEX to convert a hex response to a decimal integer. Say your device returns a value in hex in the format “VAL,98F” (where “98F” is a sample return hex value). To display that value in an integer slider, you would set the “display” variable for the command that queries the device to “$HEX:1:,:”.

If a command which is supposed to display its response fails, then its failure will be indicated by setting the label on a button to blank, or by setting the color of the “thumb” of a slider or switch to gray (for sliders, this only works on devices running iOS 5 or later).

Most commands will likely use it’s associated controller’s (via the target device) default command string. For example, in Figure 29 we’re sending the command “on” to the device “receiver”. Let’s assume that the device “receiver” has an identifier of “amp”, and that our controller is a Logitech Squeezebox Server, using the TCP protocol. The appropriate default command string for a Squeezebox Server is

irblaster send $DEV $CMD

Then for this example, “amp” is substituted for “$DEV” and “on” is substituted for “$CMD” in the controller’s default command string, and what is actually sent to the controller is:

irblaster send amp on

You can override the controller’s default command string by preceding your command with “$”. For example, if you set the command string to “$version ?”, then the full string sent to the controller would be:

version ?

If you wish to send a command whose first character is “$”, then just set the beginning of the string to “$$”.

For sliders, you can embed the current value of the slider in the command string using the string “$VAL”. Thus, a command to set a temperature might look something like “SetTemp $VAL”. You can control the format in which $VAL is sent by appending a C-style formatting string, enclosed by a pair of colons. Thus, if your current temperature was 28.523 deg, “SetTemp $VAL:%05.1f:” would send the command string “SetTemp 028.5”. In a C-style format string, the inital “%” simply indicates this is a format string, and the closing “f” indicates this is a floating point number. The number to the left of the period specifies the width in characters of the formatted string (including any period itself), and the number to the right of the period specifies the number of digits after the period. If the number to the left of the period starts with a “0”, then the formatted string is padded with “0”s on the left to fill the full width. To format the output as an integer, specify a 0 to the right of the period. Thus, the string “SetTemp $VAL:%.0f” would send the string “SetTemp 29”. You can use most C-string formatting features, for example to output an integer in hexadecimal or octal format. We’ve also extended C-formatting to support sexagesimal output. This is most useful when using “timer” controls, which output floating point values. Thus, “%s” will output a value as “HH:MM” and “%S” will output a value as “HH:MM:SS”. For example, if your timer outputs a $VAL of 1.5, the format string “$VAL:%02s:” would output “01:30”.

Command strings sent using the “TCP”, “TCP no response”, or “UDP” protocols may also embed binary data (sequences of bytes, where each byte has an integer value between 0 and 255). Hexadecimal encoded binary data may be embedded by enclosing it with ‘h ... ‘, and decimal encoded binary data embedded by enclosing it with ‘d ...’. Thus, for example, you may send the string “hello mary” in any of the three following ways:

hello mary

‘d104 101 108 108 111’ mary

'h68 65 6c 6c 6f' mary

Each encoded byte is separated by one or more spaces. The spaces are not included in the output string (encode them if you want their equivalent within a binary string). Binary substitution is done after $CMD and $DEV are substituted. Thus, if sending data via a controller which takes only binary data, set its default command string to:

'h$CMD'

Then you may assign each button its binary command without the enclosing 'h ...'.

Commands sent using either the HTTP or HTTPS protocols are simply appended to the controller IP address or hostname to form the URL used to issue the command. Continuing the example we used above, assume our Squeezebox Server controller has an IP address of "10.192.0.8" and we're communicating to it using the HTTP protocol. The appropriate default command string in this case is:

status.txt?p0=irblaster&p1=send&p2=$DEV&p3=$CMD

After replacing "amp" for "$DEV" and "on" for "$CMD", the full URL used to issue the command is then:

http://10.192.0.8/status.txt?p0=irblaster&p1=send&p2=amp&p3=on

The utility of the controller's default command string is obvious. Rather than having to enter a lengthy command string for each button, you simply select the device you're sending the command to, and enter the short command you wish to send (such as "on").

IR Devices

For IR devices controlled by an SQ Blaster or Global Cache controller, you don't directly enter the command. Rather you choose from the list of commands from the IR code database (for SQ Blasters), as well as any commands you have learned for that device. Press the "command" field in Figure 29 (which will look slightly different, indicating it will display another screen). A listing of all the learned and supplied commands will be presented (Figure 30). Simply choose the command you wish to use.

For IR devices, the "display" field in Figure 29 is replaced with "IR repeat" (IR devices don't return responses that can be displayed). If a button sends a single command to an SQ Blaster, this field will be ignored; the SQ Blaster continuously sends the IR code as long as you hold the button. If, on the other hand, the command is part of a sequence of commands assigned to a button, or is assigned to a poll button, or is talking to a Global Cache device, then this field

controls how many times the IR code is blasted. For most applications, setting this to 0 should work (that is, the IR code will be sent once, with no repeats). If you find that the command is failing, then try increasing the repeat count. Some devices need a command to be repeated before it will respond (see the manual that came with your SQ Blaster or Global Cache device for more information).

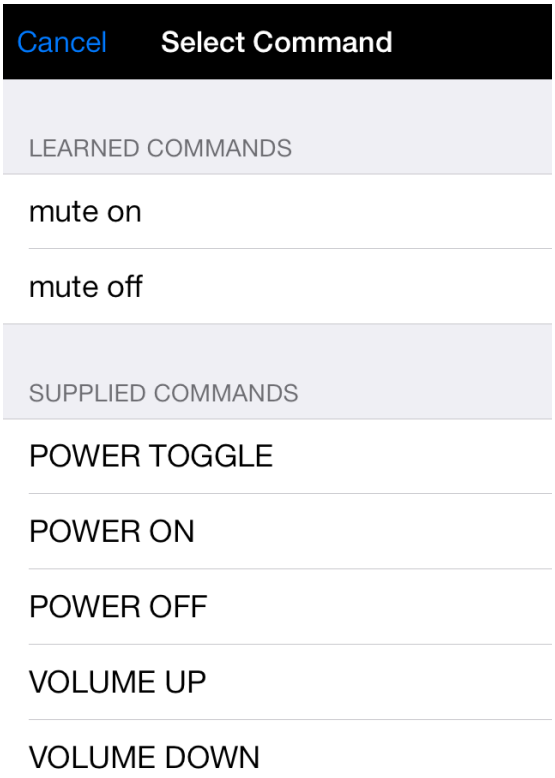

Figure 30.

Launching Applications

In addition to sending commands to devices, you can also launch a different application on your iPhone/iPad, such as a music server. For example, you can have a single button send a series of commands to turn on your stereo, set the input to your Apple TV, and then launch the "Remote" application so you're ready to select the music you want to listen to. When you launch a different application, ***Mote*** of course terminates. Quitting the different application does not return you automatically to ***Mote***.

To launch a different application, just select an application from the list of applications at the end of the device list. A number of standard music servers are already listed, including "Remote", "Spotify", "Pandora", "IPeng", and "last.fm". The device field will be set to "application", and the "command" field will be filled in with a URL that launches that application. You can also choose "custom", which will leave the "command" field blank, allowing you to enter a custom URL appropriate for the application you'd like to launch. Of course, the application must be installed on your iPhone/iPad. Applications must register a URL to enable this launching feature. To see a list of applications and their registered URLs, see the website "handleopenurl.com". The URLs for some applications (including the default ones listed in ***Mote***) can be customized to enter the application at a specific point, such as listings of genres, artists, etc. These details are given at "handleopenurl.com".

Wake-on-LAN

Mote can send Wake-on-LAN signals to servers. Choose the "Wake-on-LAN" option from the device list. For the "command", enter the broadcast IP address and the MAC address of the server, separated by a single space character (for example, "10.0.1.225 00:11:22:33:44:55").

4. Backups

You can save your current program on your device and later restore it. This gives you the ability to switch between programs for multiple locations, as well as providing the ability to save different configurations while programming. You may also export and import your programming as XML-formatted property lists, allowing you to:

- Backup your programming remotely from your device.
- Easily transfer programming between your various devices.
- Edit your configuration with external software, such as Apple's property list editors.
- Share your programming with other users.
- For professional installers, easily program your clients' devices by just installing a configuration file.

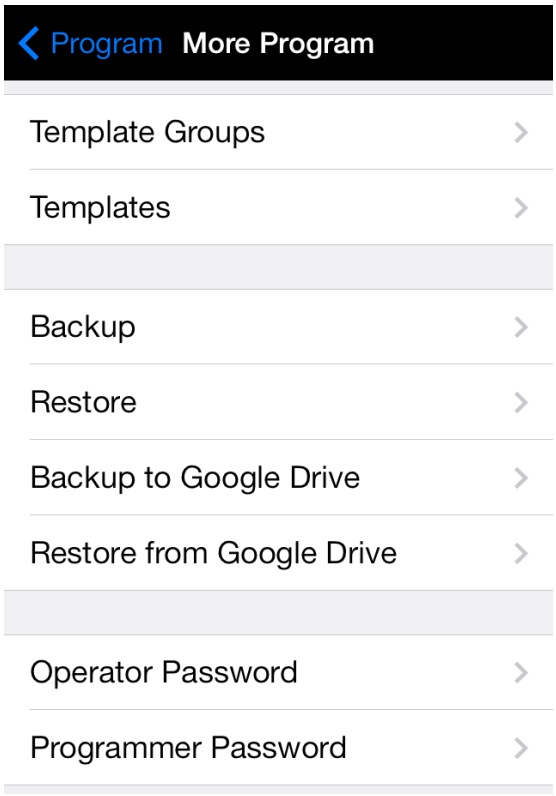

Figure 31.

There are two ways to export/import programmed configurations:

- All program backups are available as XML-formatted property list documents via iTunes File Sharing (your device must be running iOS 4 or higher). In iTunes, click on your device, then click on the "Apps" tab. Under "File Sharing", click on "Mote", and you will be presented with a list of all of the backups you've made on your device, as well as the file "config.plist". You should avoid "config.plist", as that is your current configuration, is stored in a binary format, and you could wreck havoc if you mess with it. Better to save your current configuration as a backup file. You can simply drag backup files from this list to your desktop to copy it to your computer, and vice versa to make an external file available for restoring from. Do NOT delete or replace the "config.plist" file.
- You may also save and restore configurations to and from your Google Drive. Files are saved as plain text files (in XML format). When restoring from Google Drive, you will be presented with a list of just the plain text files stored in your Google Drive. Thus, when loading an external file into Google Drive for download to ***Mote***, be sure to load it as a plain text file.

Google Drive likes to convert plain text files to one of their formats. To prevent that, name the file with the extension ".txt" and turn conversion off in your Google Drive upload settings.

To make or restore a backup, press the "More" button on the programming mode home page. The "More Program" window will appear (Figure 31) with the following backup/restore options:

- **Backup** — Make a backup of the current programmed configuration locally on your device. You will be prompted for the name of the backup.

- **Restore** — Restore a backup stored locally on your device. You will be presented with the list of saved programs from which you select the one to restore. Care should be taken when restoring from a file which was edited outside of ***Mote***. While the file is checked for consistency, not all errors can be detected. Touching the "Edit" button allows you to delete saved configurations.

- **Backup to Google Drive** — Save the current program to Google Drive. You will be prompted for your Google user name and password, and then for the name of the backup file. The program is saved as a property list in XML format.

- **Restore from Google Drive** — Restore a program saved in a file stored in your Google Drive. You will be presented with a list of your files (the plain text files only) stored in your Google Drive. Again, care should be taken when restoring from a file that was edited outside of ***Mote***.

When you restore a configuration, either from a local or remote backup, you will be prompted whether to replace or append the current programming. If you choose to replace, then the backup will simply replace the current programming. If you choose to append, then the backup will be appended to the current programming. This is useful, for example, to add a new set of templates from another user, or add a new device. When appending, if your backup contains a room with the same name as a current room, then the room from the backup will be added as a new room, with a number appended to the new room name to differentiate it from the old room.

When restoring a configuration, any panel of buttons that doesn't fit your device's screen size will be compressed to fit the screen. This is useful when transferring your programming from a device with a large screen, such as an iPad, to a device with a smaller screen, such as an iPhone.

On the "Restore" page, below the list of your backup files, is a button entitled "Reset". "Reset" erases all programming, deleting all of your rooms, controllers, devices, activities, templates, device groups, activity groups, and template groups; use with caution.

5. Using Mote

When you're done programming ***Mote***, you may want to go to the system "Settings" application and turn off the "Programming Mode" preference for ***Mote***. This removes the "Edit" and "Help" buttons from the home page for ***Mote***, thus preventing other users from easily changing the programming. If you later need to change ***Mote's*** programming, just go back to the "Settings" application and set "Programming Mode" to on.

An example home page is shown in Figure 32. Touch the room tab at the bottom of the screen for the room whose activities and devices you wish to view (the room tab bar will appear only if you have more than one room defined). Touch the appropriate button at the top of the screen to list that room's activities or devices. If you wish to list either just your activities or just your devices on the home page, set the "Display" preference in the "Settings" application.

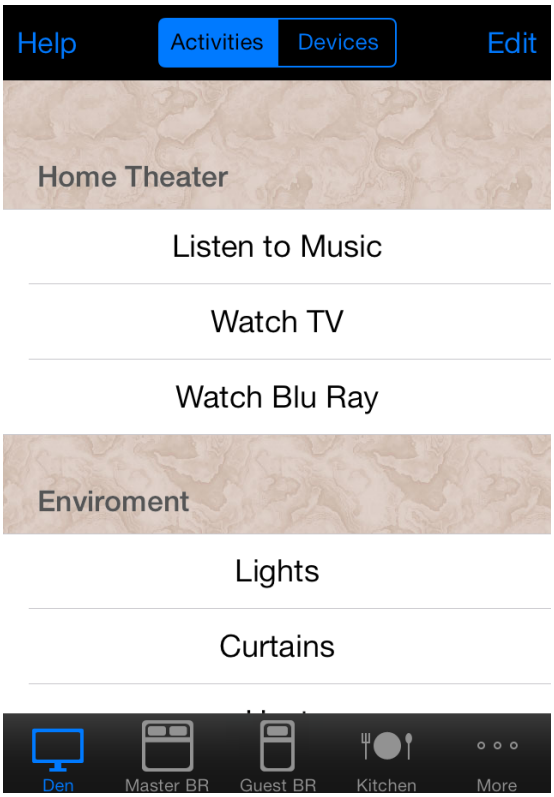

Figure 32.

Just touch the activity or device you wish to use. Its first portrait panel of buttons will appear. To change to another page, touch on either side of the page indicator at the bottom of the screen, or, if you haven't defined commands to the swipe right or swipe left gestures for that activity/device, then just swipe right or left. To go to the first landscape panel of buttons for that activity/device, just rotate your iPhone/iPad/iPod. When a button is pressed, it will highlight while sending the command sequence.

NOTE: When iPhone applications go into the background or are terminated, they are allowed roughly 10 seconds to finish what they are doing. Thus, if you hit the home button to suspend or terminate ***Mote*** while it is sending a sequence of commands, even though the application appears to quit it will continue to send the commands.

6. Settings

Selecting ***Mote*** in your iPhone/iPad/iPod's system "Settings" application displays the set of preferences in Figure 33.

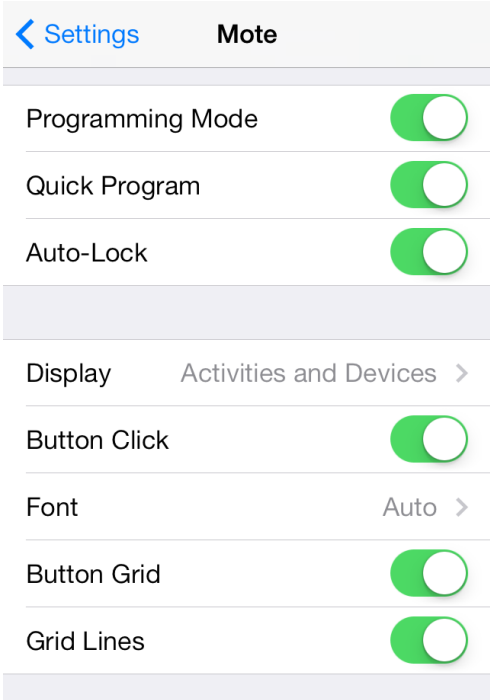

Figure 33.

- **Programming Mode** — Turn "ON" to allow programming ***Mote***. This will put the "Edit" and "Help" buttons on the home screen. Once you are done programming ***Mote***, you may want to set this to "OFF" so that other users can't easily change the programming.
- **Quick Program** — Turn "ON" to add an "Edit" button to each button panel, allowing you to quickly program the buttons for that device/template/activity, without going through the programming home screen. Once you are done programming ***Mote***, you may want to set this to "OFF" so that other users can't easily change the programming.
- **Auto-Lock** — Turn "OFF" to disable the system auto-lock feature. In this case, the screen will not automatically blank and lock after the specified time of inactivity.
- **Display** — Use this setting to specify whether activities only, devices only, or both activities and devices are displayed when not in programming mode. This has no effect on programming mode, when both activities and devices are always displayed.
- **Button Click** — If turned "ON", then a button click sound will be emitted when a button is pressed (and continue to click if the button repeats and is held).
- **Font** — The font size to use for button labels. If set to "Auto", then the font size for each button will be adjusted so that the label fits the button. Otherwise a selection of four fixed font sizes are available (which may cause a label to be truncated on a button if it doesn't fit).
- **Button Grid** — Set to "ON" to force the resizing and placement of buttons to lie on a grid, which aids button layout.
- **Grid Lines** — If set to "ON", a grid of lines will be displayed when editing button panels, to facilitate button layout. The lines are separated by four times the spacing used when "Button Grid" is on.

7. Custom Icons

Users may add their own custom icons, for use both as button icons (the image on a button, not the image of the button itself) and room icons. Any icons you add will appear in the image selector when assigning an image to a button or a room. The following rules apply:

- The required size for icons depends on whether your device has a retina display or not. Images for retina display devices need to have twice as many pixels in both width and height as images for non-retina display devices. The minimum size for button icons are 30 x 30 pixels for non-retina display devices, and 60 x 60 pixels for retina display devices. The maximum size is 200 x 200 pixels for non-retina display devices, and 400 x 400 pixels for retina display devices. The size of the pre-supplied icons is 30 x 30 pixels for non-retina display devices, and 60 x 60 pixels for retina display devices, so that is the recommended size. Icons need not be square.

- Room icons must be 30 x 30 pixels for non-retina display devices, and 60 x 60 for retina display devices. They cannot be any larger.

- Your images must be png files.

- Images must be named using the ".png" extension. For retina display devices, they must be named using "@2x" before the extension. That is an Apple convention for identifying high-resolution images. For example, say you have an image that represents a DVD player, and you want to name it "dvd". Assume it is the minimum allowed size. For a non-retina display device, the image would be 30 x 30 pixels, and you would name it "dvd.png". For a retina display device, the image would be 60 x 60 pixels, and you would name it "dvd@2x.png".

- Images intended for a non-retina display will show on a retina display, however since they are at a lower resolution than the display they will appear blocky.

- Images should use a transparent background.

- Any images you add will be available both as button icons and room icons (though they will be available as a room icon only if they are the minimum size). However, for room icons, the images will be turned into monochromatic gray images, no matter their color. This is Apple's convention for icons that appear on a tab bar at the bottom of a screen, which is how rooms are selected in ***Mote***. Thus, your image will appear in full color when you select it as a button icon, but in gray when you select it as a room icon. You need only provide one image; the conversion to monochromatic gray for a room icon will occur automatically.

- Your images must not be named the same as one of the pre-supplied icons, or they will be ignored. The following names are currently used for pre-supplied icons (the ".png" and "@2x" extensions have been dropped in this list): right, left, up, down, arrow-right, arrow-left,

arrow-up, arrow-down, arrow-right-2, arrow-left-2, arrow-up-2, arrow-down-2, arrow-right-3, arrow-left-3, arrow-up-3, arrow-down-3, power, power-green, power-red, plus, plus-thick, minus, minus-thick, stop, pause, fbw, ffw, bskip, fskip, sbskip, sfskip, rec, rec-red, rec-green, slow, louder, softer, mute, eject, search, zoom-plus, zoom-minus, check, menu, shuffle, brighter, dimmer, repeat, zero, one, two, three, four, five, six, seven, eight, nine, ok, enter, set, clock, sun, moon, snow, lock-open, lock-closed, house, window-open, window-closed, door-open, door-closed, fan, light, light-on, open, close, no, heart, paw, double-bed, single-bed, plate, wine-glass, cherries, tv-1, tv-2, bathtub, garage, music, scissors, butterfly.

To upload a custom image into ***Mote***, use iTunes File Sharing. Start up iTunes, and connect your device. Click on the device in iTunes to display its properties. Click on the “Apps” tab. Under “File Sharing”, select ***Mote*** from the list of apps. Below the panel entitled “Mote Documents”, click the “Add...” button to add one of your custom images.