Squeeze Box Controller

Summary

This project builds on work started by Mike Maxwell, additional functions have been added:

HENDRE: Support for 2 additional plays a total of 5

LeeC77: Control of volume for each individual Squeeze Box

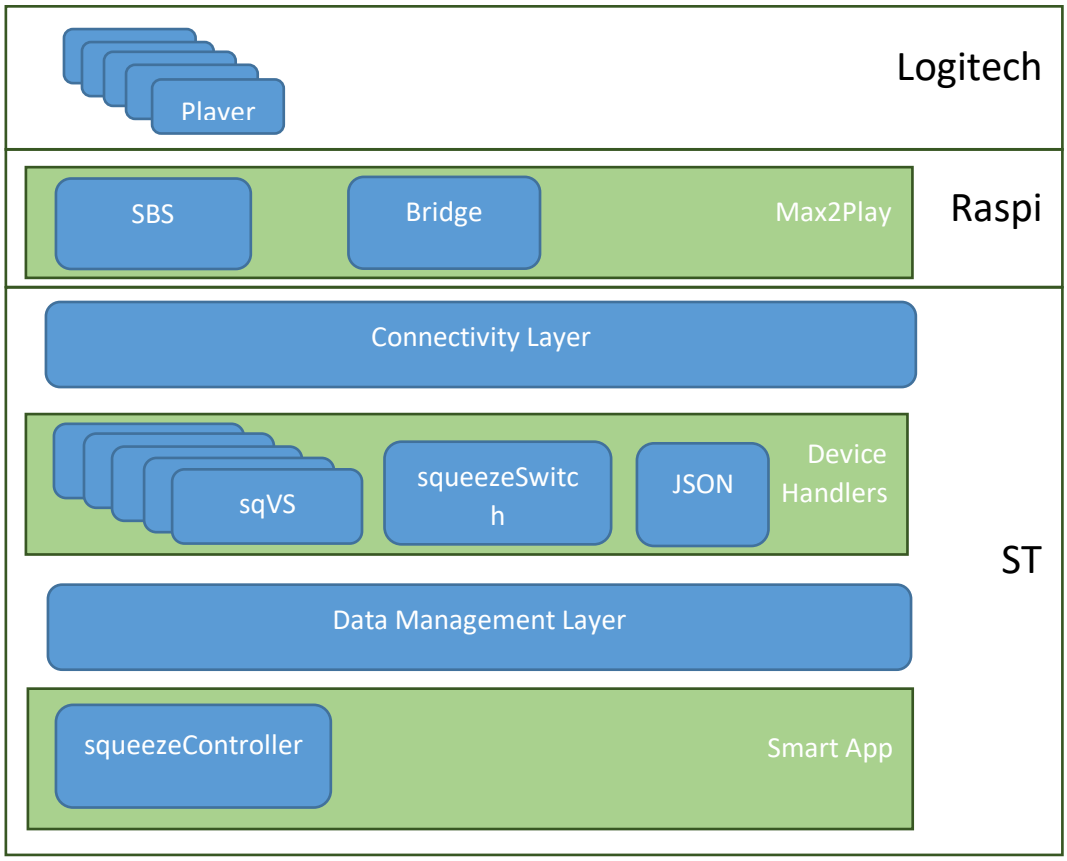
LeeC77: Status feedback (requires a bridge program)

LeeC77: Player group synchronisation and unsync all

LeeC77: Play a playlist.

If you use it please consider donating to my favourite charity:
https://www.nowdonate.com/checkout/pv0j03m4s1o1x60o6bh2

The diagram below gives you a logical overview of the various components with a physical overlay on the right, this will help orientate you as to where the separate installations need to be.



Installation

So the coding is by no means polished and is probably a bit of mess, I only dabble, but it works for me and I hope it works for you.

The installation is not simple, you will need to be confident with installing 3rd party Apps and device drivers in ST, plus a little knowledge of Linux, SBS settings and confidence in changing IP settings in your router.

Please, first read the original thread for guidance on how to install, here's the link to the original project at the ST Community:
https://community.smartthings.com/t/squeezebox-and-smarthings/1622

You will need to know the MAC address of each of your players and the network address of your Squeezebox Server, how to install the sqVS, squeezeSwitch and squeezeController is covered in the original thread, it's probably worth getting those installed first. The squeezeSwitch network ID is that of the bridge program and not the SBS, see below.

Next is to install the bridge program 'SBBridge.py' on your RasPi or other Linux machine. I use Max2Play on a RasPi B. The bridge is written in Python. The job of the bridge is to receive commands from the squeezeSwitch and forward them onto the SBS to control the player and to receive replies from the SBS and send them back to a JSON Slurper, where they are forwarded to the ST platform. This allows the status buttons in to reflect completion of a command rather tha

NOTE: The **bridge expects to receive on port 39500**, so this has to be HEX encoded as the device ID for squeezeSwitch.

The bridge program also expects the squeeze box server to be on 192.168.1.85:9090. If yours is not then you will need to edit this. Similarly with the ST hub, it's expected to be at 192.168.1.119

It's a good idea to statically assign the IP address of your RasPi and ST Hub so that the addresses don't move round on power cycle, I did this in my router.

Make a note of the directory path and name of your bridge program. Mine was:

/home/pi/winshare/SBBridge/ SBBridge.py

Now install the JSONSlurper. My JSONSlurper handles incoming data from several sources for convenience, so I have provided an edited copy below. Just install it as a device handler like you did for sqVS.

The job of the JSONSlurper is to receive messages from the bridge (forwarded from the SBS) as JSON and raise events on the ST platform.

```
import groovy.json.JsonSlurper

metadata {
        definition (name: "LAN Slurper V2", namespace: "LeeC77", author: "Lee Charlton") {
                capability "Sensor"
                attribute "hubInfo", "string"
                attribute "sbsresponse", "string"
                }
```

```
	// define tiles use attribute name as device.
	// define the relationship between state and label.https://graph-eu01-euwest1.api.smartthings.com/ide/device/editor/65942be7-08eb-4105-b889-1af8b1daeccf#
	tiles (scale: 2){
		valueTile("hubInfo", "device.hubInfo", decoration: "flat", height: 2, width: 6, inactiveLabel: false, canChangeBackground: true) {
			state "hubInfo", label:'${currentValue}'
			}
		}
	// Tile Layouts:
	main(["hubInfo"])
	}
def parse(description) {
	def descMap = parseDescriptionAsMap(description)
	def body = new String(descMap["body"].decodeBase64())
	def slurper = new JsonSlurper()
	def result = slurper.parseText(body)
	//log.debug result
/* section added to catch messages from bridge */
	if (result.containsKey("SBSResponse")) {
		sendEvent(name:"hubInfo", value:result.SBSResponse)
		 log.debug "SBServer response ${value:result.SBSResponse}"
	 sendEvent (name: "sbsresponse", value:result.SBSResponse)
	}
/*********************************************/
def parseDescriptionAsMap(description) {
	description.split(",").inject([:]) { map, param ->
		def nameAndValue = param.split(":")
		if (nameAndValue.length == 2) map += [(nameAndValue[0].trim()):nameAndValue[1].trim()]
		else map += [(nameAndValue[0].trim()):""]
	}
}
```

So hopefully you will now have the following ST devices installed

squeezeSwitch

sqVS (one for each physical device, up to 5 are supported by the Smart)

JSONSLurper

You should also have the ST Smart App squeezeController installed.

On the RasPi you should have the SBS and the bridge program.

You will need to configure squeezeController and execute the bridge program to test it all out.

Once you have got it working, the RasPi will need to be configured to run the bridge program at boot up after the SBS has started, here is how I did it.

I made a script file, called in the background from rc.local, that delays the start-up of the bridge by 200 seconds. The file is /home/pi/winshare/SBBridge/BridgeStart.sh

And simply contains the following script commands.

```
#!/bin/sh

sleep 200

python /home/pi/winshare/SBBridge/SBBridge.py &
```

Don't forget to give it executable privileges

I then edited rc.local to point at this start up script using the command:

sudo nano /etc/rc.local

I then added the line below towards the end and before the 'exit 0'

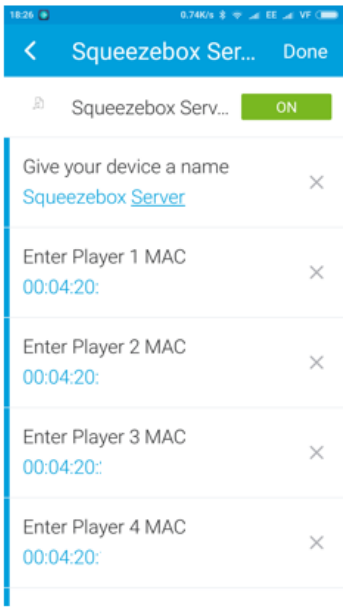
/home/pi/winshare/SBBridge/BridgeStart.sh &

Don't forget you will need to modify your path and filenames to match your installation and also remember to use the '&' at the end of the rc.local entry and the shell script bridge path.

Configuration

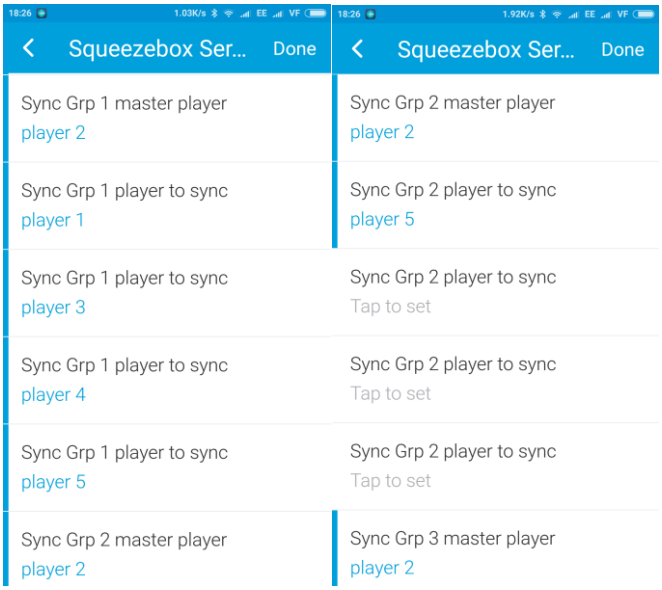
The Smart App is used to tie it all together; configure the smart app to point at the devices you installed, the JSONSlurper, up to five players and the squeeze switch.

The squeezeSwitch is where the majic happens; remember those MAC addresses from earlier, they need to go into the settings 1 for each player. See the figure below *(I have masked my MAC addresses)*



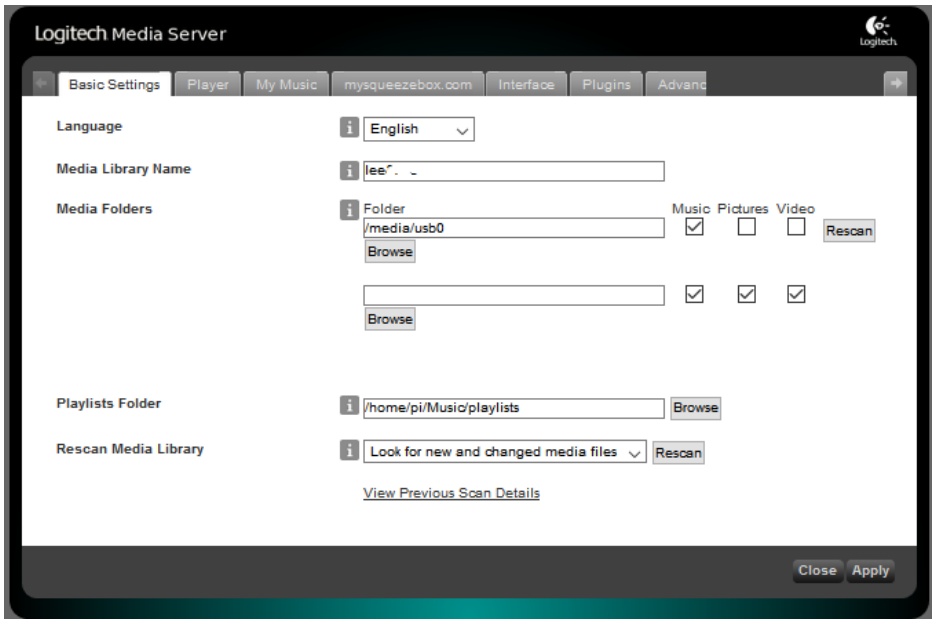
It's worth knowing which MAC address is what device, as in the next section you are going to organise your sync group.

There are three groups of 5 players. The fist player in each group is the master player to which the others in the group are synchronised later. The master ie effectively the player that determines the playlist that persists during the sync process, so if you have a favourite player then assign this as the master. See thr left hand figure below.

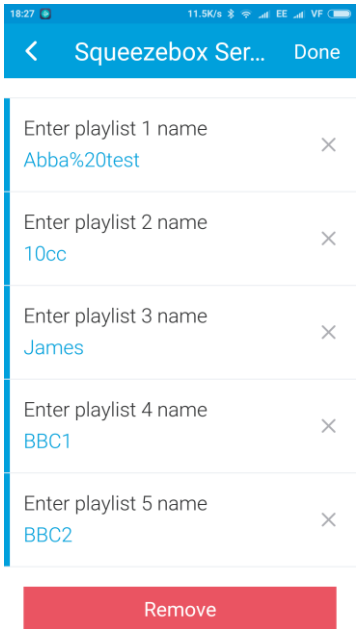


You do not have to fill in every player if you only want a couple or so in the group, see the right hand figure above.

Configuring playlists. The playlist must be stored in the playlists folder configured in the Basic Settings of your SBS, See figure below.

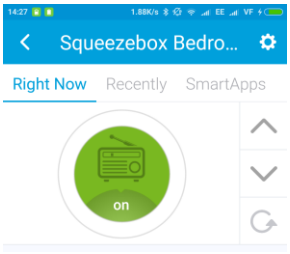


The next section in the Smart App is where you add playlists, there are 5 in total, you need to type the name of your favourite playlists as they appear on your Squeeze Box player but escape encoded (as the SBS converts some characters to space. E.g. '_'). See the figure below.



Using the App and Devices

So each of the sqVS you installed should appear as a device, with on off and volume control. See below. The up and down arrows adjust the volume in equal steps.



The squeezeSwitch device has a number of controls, described below left to right top to bottom.

1. Sync the connected play of group 1 any sync is applied in addition to any existing sync. When the icon is white it is possible to sync, when the icon is green, pressing will unsync all players (not just those in the group).
2. Sync the connected players of group 2, as 1.
3. Sync the connected players of group 3, as 1.
4. Up arrow toggles through the five playlists entered in the settings
5. Text field shows the selected playlist name.
6. Up arrow toggles through the five players entered in the settings.
7. Text field shows the selected player MAC address *(a friendly player name could be a future enhancement)*.
8. Apply button applies the playlist to the selected player.
9. On button, currently no function *(it could be possible to use this to reboot the RasPi or SBS rescan of media)*
10. Refresh, queries the on/off state status of the connected physical players and adjusts the sqVS device state to be in step.