# Harry Potter Bar Interactive Architecture Approaches Dave Price 02/2017

This document lays out the architectural designs for controlling the interactive portions of Adam and Savannah’s Harry Potter Bar. In it I lay out the base architecture for the approaches and then details as to how each approach would be implemented. The approaches I considered are:

1. DMX Software Controlled
2. Full DMX Controlled
3. Arduino Controlled
4. DMX Controlled Lighting with Arduino Controlled Animation

## Terms Used in this Document

Scene – A scene corresponds to a combination of sensors, actuators, lighting, and sound that make up an experience in the Harry Potter Bar. The scenes consist of Lachlan, the shrunken heads, Twilfit & Tattings Window, the door, and the Ice Cream Shop window. Scenes are controlled by a “Scene Controller”, which is described below.

Master Scene – A master scene is an overarching scene that affects all of the scenes above. The “Voldemort Scene” is a master scene. Master Scenes are controlled by the “Master Controller”, which is described below.

Scene Sequence – A scene sequence is a DMX sequence that sends signals over DMX channels to the scenes. This applies only to the DMX based options explained below. In it there would be a scene sequence for each scene controller and a master sequence for the master controller.

Scene Scripts – A scene script is code that controls the sensors and actuators within a scene. Scene scripts run in the scene controllers. Master scene scripts run in the master controller.

## Basic Architecture

The scenes in the Harry Potter Bar are independent in that they are separate experiences. However, because of the need for master scenes, they need to be centrally controlled. Therefore, the basis for each of my architecture approaches consists of a master controller and multiple scene controllers.

Scene Controller – There will be a scene controller for each scene in the Harry Potter Bar. Each scene controller would handle the sensors and actuators in the corresponding scene.

Master Controller - This controller will handle any overarching master scene functionality and stop/start the scene controllers as necessary.

Communication Between Controllers – signals between master controller and scene controller depends upon the approach taken. Communication may be DMX, serial over wire, or wireless. Ideally, we would set this up wirelessly and my testing experience so far with wireless responsiveness has been great. If we go with a wired approach, the amount of wiring will be impacted by the chosen design.

## Controller Design Options

The architecture above is the basis for each of the approaches considered. Details for each design are described below.

### 1 – DMX Software Controlled

The approach implemented in the design is a centralized controller model in which the master controller is a DMX software controller scripting execution of all scenes. DMX scene sequences will be created for each scene in the DMX software controller. When the sequences run, the resulting DMX channel signals are sent to the master controller and onto the respective scene controllers. In this approach, rather than using DMX cabling and devices, DMX channel signals will be communicated serially.

Master Controller – The master controller will consist of a PC running the DMX controller application and an Arduino to direct the DMX signals to each scene controller. The DMX application would send DMX sequences to the master controller Arduino, which in turn directs the scene specific DMX packets to the appropriate scene controller.

I’ve researched a really nice open source software controller called Vixen, which is typically used to control Christmas sound and light shows. DMX sequences can be created in Vixen to drive multiple devices within a scene like lighting, servos, motors, and sound.

Scene Controllers – the scripts running on the scene controllers will receive signals from the master controller and trigger actuators accordingly.

Lighting – because the DMX software is running on the pc connected to the master controller, it could either control the lighting independently of the sequences sent to the scene controllers, or the scene controllers can control the lighting based upon the sequences they receive from the master controller.

Voldemort Scene – The Voldemort master scene would be another DMX sequence running under Vixen.

Scene Script Updates – In this approach, the scene scripts are DMX sequences running in Vixen so updating the scene scripts is a matter of updating the Vixen sequence.

Wiring – Serial communication between computer running Vixen, master controller, and scene controllers. Connection between the master controller and scene controllers can be wired or wireless.

#### Positives

Sequence Creation - The DMX software UI allows for easy configuration of sequences.

#### Challenges

Sensors - How to receive sensor input to DMX controller to trigger scene specific sequences. I’ve only begun learning about DMX, but it appears to be an output focused protocol designed to drive devices based upon the sequence. I’ve not seen where sensors can provide input to the DMX controller triggering sequences to be run. We need to capture input from the sensors on the scene controllers to drive the corresponding actuators and we need to trigger the “Voldemort sequence” from the master controller.

Split Scripts – because the sequences don’t support triggering, actuator logic (in the sequences) will be separate from the sensor logic (in the Arduinos). The sensor logic in the Arduino would still need to be maintained using the Arduino IDE.

Switching Sequences – The sensor challenge above drives how to control sequences between scene controllers, but we would also need to determine how to trigger the “Voldemort” master scene at the DMX controller in order to pause individual scenes, start the master scene, and resume.

Concurrent Sequences - Coordination of multiple scene sequences within the DMX software. Since all scenes are managed centrally and distributed to scene controllers as they execute, they must all be managed and run at the same time. Vixen has the capability to run multiple sequences in parallel as a “show”, but I’d be concerned as to whether running concurrently would impact the timing of each individual sequence.

All scene sequences are triggered; either by proximity sensors, wand, or manually (by Adam). In contrast, the DMX software runs sequences repeatedly sending DMX signals to the scene controllers; however the scene need only actually run when triggered. To “trigger” a continually running sequence from a sensor, the only solution I can come up with is to have the scene controller would run the sequence it is receiving only following a trigger from a sensor. There could be a potentially noticeable delay in the timing between the sensor and the beginning of the next scene sequence cycle.

Performance with all sequences running concurrently on master controller - Without an ability to trigger the scene sequences from the sensors, all 8 scene sequences will be running in parallel on the DMX software being sent to the scene controllers serially. This concurrent execution may be taxing on the master controller resources and/or may affect the performance of the individual sequences.

### 2 - Full DMX Controlled

This approach is similar to the DMX software approach, however rather than an Arduino master controller, there would be a DMX hardware controller and all communication from the master controller to the scene controllers would be done over DMX cables. Each scene controller would require a DMX shields to make it a DMX slave. The scene controllers would be connected in series, like other DMX devices and would process the DMX packets addressed to it.

DMX sequences could be created that result in channel signals that orchestrate the scene scripts. The Arduino scene controllers “execute” the scene scripts by interpreting the DMX channel signals and acting accordingly (make frog hop, ring bell, trigger door knocker, etc.)

Master Controller – The master controller will consist of a PC running the DMX controller application connected to a DMX controller to direct the DMX signals to the scene controllers.

Scene Controllers – the scripts running on the scene controllers will receive DMX packets from the DMX master controller and trigger actuators accordingly.

Lighting – same as in DMX Software Controlled approach.

Voldemort Scene – same as in DMX Software Controlled approach.

Scene Script Updates – same as in DMX Software Controlled approach.

#### Positives

Sequence Creation – same as in the DMX Software Controlled approach.

#### Challenges

Sensors - same as in DMX Software Controlled approach.

Split Sequences – same as in DMX Software Controlled approach.

Switching Sequences – same as in DMX Software Controlled approach.

Concurrent Scenes - same as in DMX Software Controlled approach.

Performance with all sequences running concurrently on master controller - same as in DMX Software Controlled approach.

Wiring – Because communication between master and scene controllers is over DMX cable DMX shields would have to be purchased for each Scene controller and the Arduino logic would have to be written to leverage the DMX shields.

### 3 – Arduino Controlled

The approach implemented in this design modularizes the functionality of the entire setup into scene scripts running on respective scene controllers and master scenes governed by the master controller.

Master Controller - The master controller will consist of a single board Windows machine with integrated Arduino (e.g. LattePanda). The master controller is responsible for running master scenes and orchestrating the scene scripts as a whole when a master scene is to be run. Details are explained in the topics below.

Scene Controllers - In this approach, the scene controllers will be Arduinos running local scene specific scripts to handle the sensors and actuators making up the scene they control.

Lighting – scene lighting will be controlled by the corresponding scene controller. While running a scene script, the lighting will be controlled accordingly. When a master scene is triggered, the scene lighting will be adjusted by each scene controller and restored once the master scene has ended.

Sensors – Sensor input will be captured by the controller (scene or master) to which they are connected. The sensor signal will be handled by the scene script (or master script) running on the controller.

Voldemort Scene –This master scene script will be triggered by a sensor (button, phone, timer, etc) on the master controller. The master controller will send a “standby” signal to all of the scene controllers and then will execute the master scene. When the scene controllers receive this “standby” signal, they will stop their scripts and adjust lighting until told to resume by the master controller once the Voldemort master scene concludes.

Wiring – since the scene scripts are running locally to the scene controller, if we go with a wired solution, there will only need to be 1 wire from master controller to each scene controller.

#### Positives

Performance – the work will be divided up among all of the scene controllers. Each scene controller will truly control its scene by running their corresponding scene script. The master controller will be busy during startup providing scene scripts to each scene controller, but will then remain essentially idle waiting for a master scene to be invoked (see Voldemort Scene).

Evolutionary Build – because scripts run locally, they can be written, tested, and deployed individually. Then enhanced to include pulling updates from the master controller later.

Complete Scripts – because the scripts are code rather than sequences, the related sensor and actuator logic would all be handled in one script.

#### Challenges

Scene Script Updates – Because scripts are running locally on the scene controller or master controller, a method to update scene scripts must be considered.

The standard way of updating scripts on an Arduino is by connecting the Arduino to a laptop, editing code in the Arduino IDE, and publishing the code to the Arduino to execute. This approach would require connecting a laptop to each of the scene controllers any time a script needs updating.

To provide the ability to dynamically update scripts on the controllers, I’ve designed an approach that allows the scene controllers scripts to be managed by the master controller. When the scene controllers start up, they will send a notification to the master controller letting it know that they are online. If there is an updated script for a scene controller, the master controller will send the updated script to the scene controller, which will be cached and executed locally. Changes to scene controller scripts are made on the master controller to be pulled down when the scene controller restarts next.

Scene Scripts – in this approach the scene scripts are Arduino code rather than sequences created in the DMX software. The code is straightforward, but is code rather than configuration.

### 4 – DMX Controlled Lighting, Arduino Controlled Animation

This approach is essentially a combination of Arduino Controlled and DMX Controlled. All lighting would be controlled via DMX, but orchestration of the scene scripts would be via Arduino code.

Master Controller – the master controller would own the master scene scripts and would provide scripts to the scene controllers as above.

Scene Controllers – the scene controllers would obtain their scripts following the same approach as outlined in the Arduino Controlled approach.

Voldemort Scene - DMX controller would initiate the master scene by sending DMX signals to the lights and the master Arduino. The master Arduino would send a signal to all scene controllers to go into “standby”. When the master scene sequence completes, the DMX controller would send DMX signals to the lights and master Arduino to resume normal operation. The master Arduino would send a signal to all scene controllers to “resume” with their locally running scripts.

#### Positives

DMX Lighting – All scene lighting, or master scene lighting, could be controlled via DMX, which is what DMX is designed for.

Arduino Managed Code – in this approach logic would be controlled via the Arduino

#### Challenges

Triggering Master Scene – As with the other DMX approaches, a mechanism to trigger the master scene via the DMX controller would have to be identified.

Scene Script Updates – same as with the Arduino Controlled approach.

Syncing of Lighting and Logic – because lighting and logic are managed separately, synching them might be a challenge.