# Table of Contents

## Overview

## Build It

### Code It
- Arduino Code
- Adafruit IO
- Create the Song Feed
- Dashboard
- Create Dashboard
- Create Song Blocks
- Create Volume Slider
- Pause/Play Block
- Stream Block
- If This then That - IFTTT

## Use It
Overview

You can turn the Internet of Things into the Internet of Tunes! With this easy-to-build project you can use online events to trigger your favorite songs.

The Feather ESP8266 Huzzah is a WiFi connected board that can use the Music Maker FeatherWing with amp to play music at your command. Using the Adafruit IO () service, a small bit of Arduino code and IFTTT () (If This then That), events such as Twitter mentions, weather reports, YouTube uploads, news alerts, and more can trigger songs to play that you have stored on the included SD memory card.
There are endless possibilities for you to create a sweet enclosure for your connected Feather, FeatherWing, and speaker. We've included this lovely little cardboard box for just this purpose. You can decorate it, cut it, modify it, bedazzle it, and make it your own!

This is just one way to get your system set up for WiFi enabled playback. You can modify this to suit your needs, such as adding the breadboard, IR receiver, and potentiometer for volume control. It's as simple as marking and cutting the cardboard with a hobby knife and then mounting your parts.
Decide where to mount your speaker.

Mark the position with a pencil.
Cut inside the lines a bit for a snug fit.
Pop in your speaker.

Feed the wires through the hole first, then tip the left side in, followed by the right -- this will allow for the cleanest fit of the wires and speaker!
For an extra secure build, use the supplied screws and nuts.
Poke a hole through at each corner.
Push the screws through, and then secure the nuts on the other side.
Insert the USB cable into the box through the side flap as seen here. You'll use it to program and power the Feather.

NOTE: Due to fluctuations in the organic, artisanal cardboard box harvest, your box may look different than the one pictured here. It is fine to modify yours as needed to route the wiring, such as the one seen in the last picture here.
Wire the speaker to the screw terminals of the Music Maker MP3 FeatherWing, and tighten the screws.
If you’re mounting the Feather HUZZAH to the breadboard, do so now. You can also go breadboard-less.
Weather or not you use the breadboard, you'll want to insert your SD card. Make sure you can remove it easily for adding files to it later.
Fit the Music Maker FeatherWing onto the Feather HUZZAH.
Now, you can close up the box.
The USB cable can be used both to program and power the device. To power it from a wall outlet, plug it into the provided 5V transformer.
Feeling fancy? Great, me too! Decorate your player with the provided collectable enamel pin!!
You can also get super duper fancy and use different speakers and boxes for your WiFi Music Alert Box, as seen here.
Code It

There are three areas to "code" for this project.

• First, we have the Arduino code that'll run on the Feather board.
• Second, we have Adafruit IO feeds and dashboard to configure.
• And lastly, we'll got to IFTTT.com to set up applets that can trigger the Adafruit IO feeds.
Arduino Code

The program running on the Feather is pretty simple. It is adapted from two sample sketches. One that plays music using the Music Maker FeatherWing, and another that connects the board to WiFi and waits for commands from Adafruit IO.

First, make sure you're familiar with the setup and coding of the Feather board, by going through [this guide](#) and connecting to WiFi successfully.

Next, become familiar with playing MP3s with the Music Maker FeatherWing by following the setup in [this guide](#).

Now, you're ready to download the Music Alert Box code and configure it to your needs.

Download this file and then unzip it and move the directory into your Arduino sketch folder.

The sketch contains two files. First, we'll look at the config.h file. Here, you need to change the IO_USERNAME to your Adafruit IO username, and the key to your Adafruit IO key.

You can find more info on this [here](#).
Once you've made these changes, save the file.

```cpp
/**************************** WIFI **********************************/

#define WiFi_SSID   "your wifi ssid"
#define WiFi_PASS   "your wifi password"

// comment out the following two lines if you are using fona or ethernet
#include "AdafruitIO_WiFi.h"
AdafruitIO_WiFi io(IO_USERNAME, IO_KEY, WiFi_SSID, WiFi_PASS);

/**************************** FONA **********************************/

// uncomment the following two lines for 32u4 FONA,
// and comment out the AdafruitIO_WiFi client in the WIFI section
#include "AdafruitIO_FONA.h"
AdafruitIO_FONA io(IO_USERNAME, IO_KEY);

/**************************** ETHERNET ******************************

// uncomment the following two lines for ethernet,
// and comment out the AdafruitIO_WiFi client in the WIFI section
#include "AdafruitIO_Ethernet.h"
AdafruitIO_Ethernet io(IO_USERNAME, IO_KEY);
```

Next, you'll change the MusicAlertBox.ino file to include the names of the .mp3 files you're using. Be sure to copy these to the SD card, and keep the names short (eight-
dot-three format is a safe way to go). The names must match exactly between the files and their callouts in the code.

```c
// WiFi Music Alert Box
// Uses Feather Huzzah ESP8266 WiFi board and Music Maker FeatherWing
// along with Adafruit IO and IFTTT to play triggered sound files
// written by John Park
// based on Todd Trece's Digital Input sample code
// MIT license
/************************** Configuration ********************************************/
// edit the config.h tab and enter your Adafruit IO credentials
// and any additional configuration needed for WiFi, cellular,
// or ethernet clients.
#include "config.h"

/************************ libraries ******************************************/
// include SPI, MP3 and SD libraries
#include <SPI.h>
#include <SD.h>
#include <Adafruit_VS1053.h>

/************************ Music Maker definitions **************************/
// These are the pins used
#define VS1053_RESET   -1     // VS1053 reset pin (not used!)
#define VS1053_CS      16     // VS1053 chip select pin (output)
#define VS1053_DCS     15     // VS1053 Data/command select pin (output)
#define CARDCS          2     // Card chip select pin
#define VS1053_DREQ     0     // VS1053 Data request, ideally an Interrupt pin
Adafruit_VS1053_FilePlayer musicPlayer =
  Adafruit_VS1053_FilePlayer(VS1053_RESET, VS1053_CS, VS1053_DCS, VS1053_DREQ, CARDCS);

/************************ Define Feeds**************************************/
// set up the 'song' feed
AdafruitIO_Feed *song = io.feed("music-player-01-song");
// set up the 'volume' feed
AdafruitIO_Feed *volume = io.feed("music-player-01-volume");
// set up the 'pause' feed
AdafruitIO_Feed *pause = io.feed("music-player-01-pause");

/************************ setup ********************************************/
void setup() {
  if (! musicPlayer.begin()) { // initialise the music player
    while (1);
  }

  musicPlayer.sineTest(0x44, 500); // tone to indicate VS1053 is working

  if (!SD.begin(CARDCS)) {
    while (1); // don't do anything more
  }

  // Set volume for left, right channels. lower numbers == louder volume!
  musicPlayer.setVolume(20,20);
  musicPlayer.useInterrupt(VS1053_FILEPLAYER_PIN_INT); // DREQ int

  // set led pin as a digital output
```

©Adafruit Industries
//pinMode(LED_BUILTIN, OUTPUT);

// start the serial connection
Serial.begin(115200);

// wait for serial monitor to open
//While(! Serial);

// connect to io.adafruit.com
Serial.print("Connecting to Adafruit IO");
io.connect();

// set up a message handler for each of the feeds.
// e.g., the handleSongMessage function (defined below)
// will be called whenever a "song message" is
// received from adafruit io

song-&gt;onMessage(handleSongMessage);
volume-&gt;onMessage(handleVolumeMessage);
pause-&gt;onMessage(handlePauseMessage);

// wait for a connection
while(io.status() &lt; AIO_CONNECTED) {
    Serial.print(".");
    Serial.println(io.statusText());
delay(500);
}

// we are connected
Serial.println();
Serial.println(io.statusText());

}*="/************************ loop ***************************************************/
void loop() {
    // io.run(); is required for all sketches.
    // it should always be present at the top of your loop
    // function. it keeps the client connected to
    // io.adafruit.com, and processes any incoming data
    io.run();
}

="/********************** handleSongMessage ***************************************************/
// this function is called whenever a 'song' feed message
// is received from Adafruit IO. it was attached to
// the 'song' feed in the setup() function above.
void handleSongMessage(AdafruitIO_Data *data) {

    Serial.print("song received &lt;- ");

    // write the current state to the led
    if(data-&gt;toInt()==1){
        musicPlayer.startPlayingFile("sloww.mp3");
    }
    else if(data-&gt;toInt()==2){
        musicPlayer.startPlayingFile("mm.mp3");
    }
    else if(data-&gt;toInt()==3){
        musicPlayer.startPlayingFile("alon.mp3");
    }
    else if(data-&gt;toInt()==4){
        musicPlayer.startPlayingFile("breaths.mp3");
    }
    else if(data-&gt;toInt()==5){
        musicPlayer.startPlayingFile("happy.mp3");
    }
    else if(data-&gt;toInt()==6){
"
musicPlayer.startPlayingFile("kittin.mp3");
} else if(data-&gt;toInt()==7){
    musicPlayer.startPlayingFile("new1.mp3");
} else if(data-&gt;toInt()==8){
    musicPlayer.startPlayingFile("new2.mp3");
} else if(data-&gt;toInt()==9){
    musicPlayer.startPlayingFile("pop_1.mp3");
} else if(data-&gt;toInt()==10){
    musicPlayer.startPlayingFile("wawawa.mp3");
} else if(data-&gt;toInt()==11){
    musicPlayer.startPlayingFile("zoom1.mp3");
} else if(data-&gt;toInt()==12){
    musicPlayer.startPlayingFile("metroly.mp3");
} else if(data-&gt;toInt()==13){
    musicPlayer.startPlayingFile("pascal.mp3");
} else if(data-&gt;toInt()==14){
    musicPlayer.startPlayingFile("solder.mp3");
}
Serial.print(data-&gt;feedName());
Serial.print(" ");
Serial.println(data-&gt;value());

/************************ handleVolume ************************************/
// this function is called whenever a 'volume' feed message
// is received from Adafruit IO. it was attached to
// the 'volume' feed in the setup() function above.
void handleVolumeMessage(AdafruitIO_Data *data) {
    Serial.print("volume received <- ");
    // since we are using the same function to handle
    // messages for two feeds, we can use feedName() in
    // order to find out which feed the message came from.
    Serial.print(data-&gt;feedName());
    Serial.print(" ");
    Serial.println(data-&gt;value());

    musicPlayer.setVolume(data-&gt;toInt(), data-&gt;toInt());
}

/************************ handlePauseMessage *******************************/
// this function is called whenever a 'pause' feed message
// is received from Adafruit IO. it was attached to
// the 'pause' feed in the setup() function above.
void handlePauseMessage(AdafruitIO_Data *data) {
    Serial.print("pause received &lt;- ");
    // write the current state to the led
    if(data-&gt;toInt()==1){
        musicPlayer.pausePlaying(false);
    } else if(data-&gt;toInt()==0){
        musicPlayer.pausePlaying(true);
    }
    Serial.print(data-&gt;feedName());
}
Save and then upload the sketch to your Feather ESP8266 Huzzah and test it out. You should see the board connect to your WiFi network by checking the serial monitor.

![Serial Monitor screenshot](image)

**Adafruit IO**

Now, we can set up Adafruit IO to act as a WiFi connected playback UI for the Feather and its songs.

The [Adafruit IO Basics: Digital Input guide](https://learn.adafruit.com/adafruit-io-basics-digital-input) is an excellent place to start to get an understanding of how the system works. Familiarize yourself with the guide, end even try out the example to get started. Then, we'll modify the feeds and dashboard for our needs.
For the WiFi Music Alert Box we'll set up three feeds: Song, Volume, and Pause.

Create the Song Feed

Start by creating a new feed Named Music Alert Box Song. If you need help getting started with creating feeds on Adafruit IO, check out the Adafruit IO Feed Basics guide.
Edit the feed to adjust the key name if needed -- this is the name that will be used by the Arduino sketch to interact with the feed. In this case, the key is set to music-player-01-song. You can see this in the Arduino sketch where it says:

```
// set up the 'song' feed
AdafruitIO_Feed *song = io.feed("music-player-01-song");
```

Create two more feeds, Music Alert Box Volume, and Music Alert Box Pause. Each of these will be used to handle different kinds of data. Use the key names seen here.

Create Dashboard

In order to assign meaning to the feeds you’ve created, make a new Dashboard called Music Alert Box. If you need help getting started with Dashboards on Adafruit IO, check out the [Adafruit IO Dashboard Basics guide](https://www.adafruit.com/learning/tutorial/180).
Create Song Blocks

Create a new block that will play a song when clicked. For this, you can use a momentary button block.

Create a new block

Click on the block you would like to add to your dashboard. You can always come back and switch the block type later if you change your mind.

The pop up window will ask you to select to which feed to connect the button. In this case, we'll choose Music Alert Box Song feed by clicking the group/feed checkbox for that line, and then clicking Next Step.
In the Block Settings window that pops up, you can name the block, adjust the button text to the name of the song you want it to trigger, and give the Press Value a number that correlates to your Arduino sketch if statement. You can delete the Release Value, we won't use it in this case, and then click Create block.

In this final step, you can give your block a title and see a preview of how it will look. Customize the look and feel of your block with the remaining settings. When you are ready, click the "Create Block" button to send it to your dashboard.

Your dashboard will now have on button on it. Click it and your song will play!
Add more buttons to correlate to each song, and then you can arrange them by clicking the gear icon on your dashboard.

Don’t need a name over each button? Edit the button's Block Title to be a space character!
Create Volume Slider

You can create a slider element to control the volume.

Create a new block

Click on the block you would like to add to your dashboard. You can always come back and switch the block type later if you change your mind.

For this block, select the Music Alert Box Volume feed.
Choose feed

The slider works well if you have a range of values you need to send. If you have a lot of feeds, you may want to use the search field. You can also create a feed quickly below.

<table>
<thead>
<tr>
<th>Group / Feed</th>
<th>Last value</th>
<th>Recorded</th>
</tr>
</thead>
<tbody>
<tr>
<td>Welcome Feed</td>
<td>100</td>
<td>2 hours ago</td>
</tr>
<tr>
<td>Onoff</td>
<td>ON</td>
<td>6 months ago</td>
</tr>
<tr>
<td>shoptemperature</td>
<td>70</td>
<td>6 months ago</td>
</tr>
<tr>
<td>counter</td>
<td>11544</td>
<td>6 months ago</td>
</tr>
<tr>
<td>light</td>
<td>1</td>
<td>6 months ago</td>
</tr>
<tr>
<td>Music Alert Box Song</td>
<td>2</td>
<td>an hour ago</td>
</tr>
<tr>
<td>Music Alert Box Volume</td>
<td>8.6</td>
<td>an hour ago</td>
</tr>
<tr>
<td>Music Alert Box Pause</td>
<td>1</td>
<td>an hour ago</td>
</tr>
</tbody>
</table>

Adjust the block settings in the next popup window to match these.

Block settings

In this final step, you can give your block a title and see a preview of how it will look. Customize the look and feel of your block with the remaining settings. When you are ready, click the "Create Block" button to send it to your dashboard.

Block Title
Music Alert Box Volume

Slider Min Value
1

Slider Max Value
80

Slider Step Size
.1

Slider Label
Volume

Note: Volume on the Music Maker FeatherWing gets louder as the numbers get lower!

Test this out -- now as you play back songs you can adjust the volume.
Pause/Play Block

For fun, let's set up a block that can pause and play the songs! Create a new toggle block.

Choose the Music Alert Box Volume feed.
Choose feed

A toggle button is useful if you have an ON or OFF type of state. You can configure what values are sent on press and release. If you have lots of feeds, you may want to use the search field. You can also create a feed quickly below.

<table>
<thead>
<tr>
<th>Group / Feed</th>
<th>Last value</th>
<th>Recorded</th>
</tr>
</thead>
<tbody>
<tr>
<td>Welcome Feed</td>
<td>100</td>
<td>3 hours ago</td>
</tr>
<tr>
<td>On/off</td>
<td>ON</td>
<td>6 months ago</td>
</tr>
<tr>
<td>shop temperature</td>
<td>70</td>
<td>6 months ago</td>
</tr>
<tr>
<td>counter</td>
<td>1144</td>
<td>6 months ago</td>
</tr>
<tr>
<td>light</td>
<td>1</td>
<td>6 months ago</td>
</tr>
<tr>
<td>Music Alert Box Song</td>
<td>2</td>
<td>an hour ago</td>
</tr>
<tr>
<td>Music Alert Box Volume</td>
<td>8.6</td>
<td>2 hours ago</td>
</tr>
</tbody>
</table>

Use the title Pause/Play to label the block on the UI. Set the On text to 1 and the Off text to 0. These are the values that will be sent to the Feather when the block is clicked.

Block settings

In this final step, you can give your block a title and see a preview of how it will look. Customize the look and feel of your block with the remaining settings. When you are ready, click the “Create Block” button to send it to your dashboard.

Block Title
Pause/Play

Button On Text
1

Button Off Text
0
Lastly, we'll create a data stream block. This will allow us to see the values flowing from our button presses to the Feather.

Select all three Music Alert Box feeds.
Choose up to 5 feeds

A stream block can be used to view the rolling history of data for multiple feeds. Up to 5 of your feeds can send data to this block. If you have lots of feeds, you may want to use the search field. You can also create a feed quickly below.

- **Welcome Feed**: Last value 100, Recorded 3 hours ago
- **Onoff**: Last value ON, Recorded 6 months ago
- **shop temperature**: Last value 70, Recorded 6 months ago
- **counter**: Last value 11544, Recorded 6 months ago
- **light**: Last value 1, Recorded 6 months ago

- **Music Alert Box Song**: Last value 2, Recorded an hour ago
- **Music Alert Box Volume**: Last value 8.6, Recorded an hour ago
- **Music Alert Box Pause**: Last value 1, Recorded an hour ago

Place the data feed block where you like. Now, you can see updates as your feed data changes.

**If This then That - IFTTT**

The third part of the setup is to trigger the songs using Internet alerts via IFTTT applets. These can be any of hundreds of possible triggering services! For example, Twitter, Date & Time, Flickr, YouTube, just to name a few.
If you haven't already, create an account on IFTTT.com.

Once you have logged in, click on your user profile and choose, Services from the drop-down menu.

Click on "All service", then type "Adafruit" into the search bar, and then click on the Adafruit service icon.
Click the big, huge Connect button. Because how can you resist?

If you are logged into your Adafruit.com account, you'll be asked to authorized the IFTTT connection. (If not, you need to first log into Adafruit.)
Authorize IFTTT

The application IFTTT is requesting the following information for your account. Would you like to grant access?

- Name
- username
- Adafruit IO Dashboard URL
- Read and write to your feed data

By granting access, you'll be able to connect your Adafruit account to IFTTT, and enable any integrations that have been provided.

[Authorize] [Cancel]

Now, back in IFTTT, you can create a new Applet. Click on [My Applets ()] and then click New Applet.

[Link to New Applet]

Click on the +this link.
Pick the Date & Time service.

Choose the Every hour at trigger
Leave it set to 00 and click Create trigger.

That takes care of the if this, so on to the then that. Click the +that link.

Type adafruit into the search, then click on the Adafruit icon.
There's just the one option, Send data to Adafruit IO.

From the dropdown for Feed names, pick Music Alert Box Song. This tells the action to send data to that Adafruit IO feed. Then, type in a 1 in the Data to save field. This is the number of the song you want to play on the hour. Then click Create action.
Complete action fields
Step 5 of 6

Send data to Adafruit IO
This Action will send data to a feed in your Adafruit IO account.

Feed name (required)
Music Alert Box Song

The name of the feed to save data to.

Data to save (required)
1

The data to be saved to your feed.

Add ingredient

Create action
Click **Finish** because you're all done!

You can check that all of the connections between services are working properly by clicking the applet to expand its info, and then clicking **Check now**.
If every hour at 00 minutes past the hour, then send data to Music Alert Box Song feed. Play song 1.

by 🌸 jp_adafruit_com

On

• Created on May 26, 2017
• Never run

This Applet usually runs within a minute

Check now

works with 🕒
Use It

Now that you know how to set up an IFTTT applet connected to Adafruit IO service, you can make many more triggered by anything that's available, from social media mentions, to stock market fluctuations, to smart home sensor, and even voice commands from Amazon Alexa.

Feel like going bit more mobile with your Music Alerts? You can add a battery to your project, such as the 2200MAh LiPoly seen here. As long as you're in WiFi range, you can take your IoT tunes from room to room of your house or office.