Annunciator User Manual

By: Jim Schrempp and Bob Glicksman; v1, 7/28/2025

TABLE OF CONTENTS.

OVERVIEW.

The **Annunciator** is a device that subscribes to cloud events and plays pre-recorded audio clips that correspond to data in the event that it subscribes to. It is intended to be used on a variety of projects that utilize Particle's[1] cloud-based publish and subscribe mechanism. An example of such a system is the Maker Nexus Help Button system that is depicted in figure 1, below.

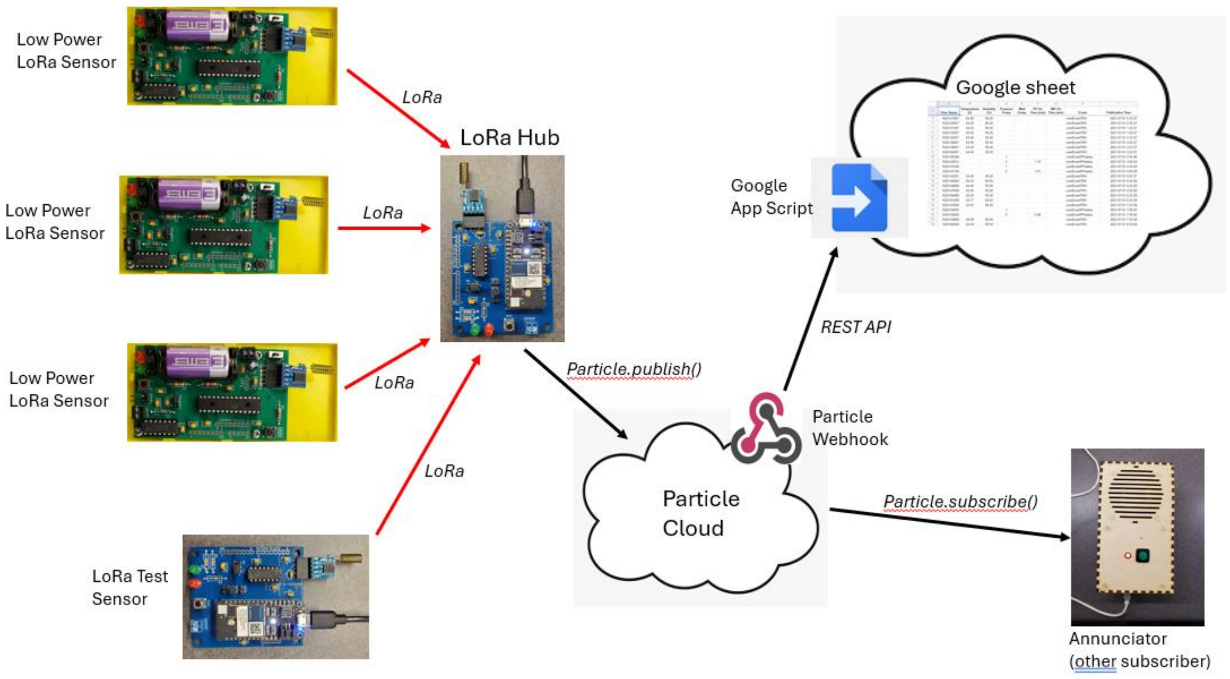

Figure 1. Help Button System.

In this system, a Particle Photon 2 based "Hub" receives LoRa[2] messages from low power sensors that are triggered by a user depressing a button on the face of the sensor's enclosure. Pressing the button sends a short message via LoRa to the Hub which, in turn, publishes an event to the Particle Cloud. Several Annunciators subscribe to this event, decode the event data to extract the deviceID of the sending device, and play a pre-recorded MP3 audio clip requesting help to the sender's location.

Annunciators are general in nature. They are compatible with any system that uses Particle's publish and subscribe capability. They may easily be customized to any such system by changing functions in the software that:

1. Subscribe to the event (changing the event name that is subscribed to).
2. Parse the event data (usually a String) to determine the deviceID in the event.

[1] Particle.io
[2] https://en.wikipedia.org/wiki/LoRa

The Annunciator code includes a separate file (ClipsList.h) that determines the file name of a clip to be played based upon the deviceID in the event.

The Annunciator uses a DFRobotDFPlayer mini MP3 Player[3] module that plays audio clips recorded on a micro SD card inserted into the mini MP3 Player. The mini MP3 Player contains a 3 watt audio amplifier that can directly drive a small 8 ohm speaker for audio output. The mini MP3 Player also provides stereo audio output signals that are brought out to a stereo phono jack that is included on the printed circuit board. This jack provides an alternative means of audio output , e.g. an amplified stereo speaker system.

Note that many vendors sell cheap modules named "DFRobotDFPlayer mini MP3 Player" but they are not compatible. We recommend purchasing modules directly from DFRobot.

The printed circuit board used in the Annunciator was originally developed for an earlier project; see:

> https://github.com/TeamPracticalProjects/Animatronics

Some of the circuitry supported by this circuit board is not needed for the Annunciator. See BUILD INSTRUCTIONS, below, for details.

The Annunciator uses the original Particle Photon (Photon 1) Wi-Fi enabled microcontroller. This device is currently deprecated in favor of the Photon 2. It should be possible to use an adapter socket to fit a Photon 2 to this board; *however, this has not been tested*. An adapter socket can be purchased from:

https://store.particle.io/products/particle-classic-adapter?_pos=1&_sid=a70fab1fa&_ss=r

A pin mapping analysis from this printed circuit board to the Photon 2 is included in this repository, see:

> Hardware / Photon adaptor mapping.pdf

OPERATING INSTRUCTIONS.

Overview of Operation.

[3] https://www.dfrobot.com/product-1121.html?srsltid=AfmBOorNRD3TM47Q7XdCygqQFtwkM0e6PAb-3TCUPtaLV12wPd9M4MyM

The Annunciator is very simple to operate. User operation is limited to a small red LED on the front panel and a larger, green backlit pushbutton; see figure 2.

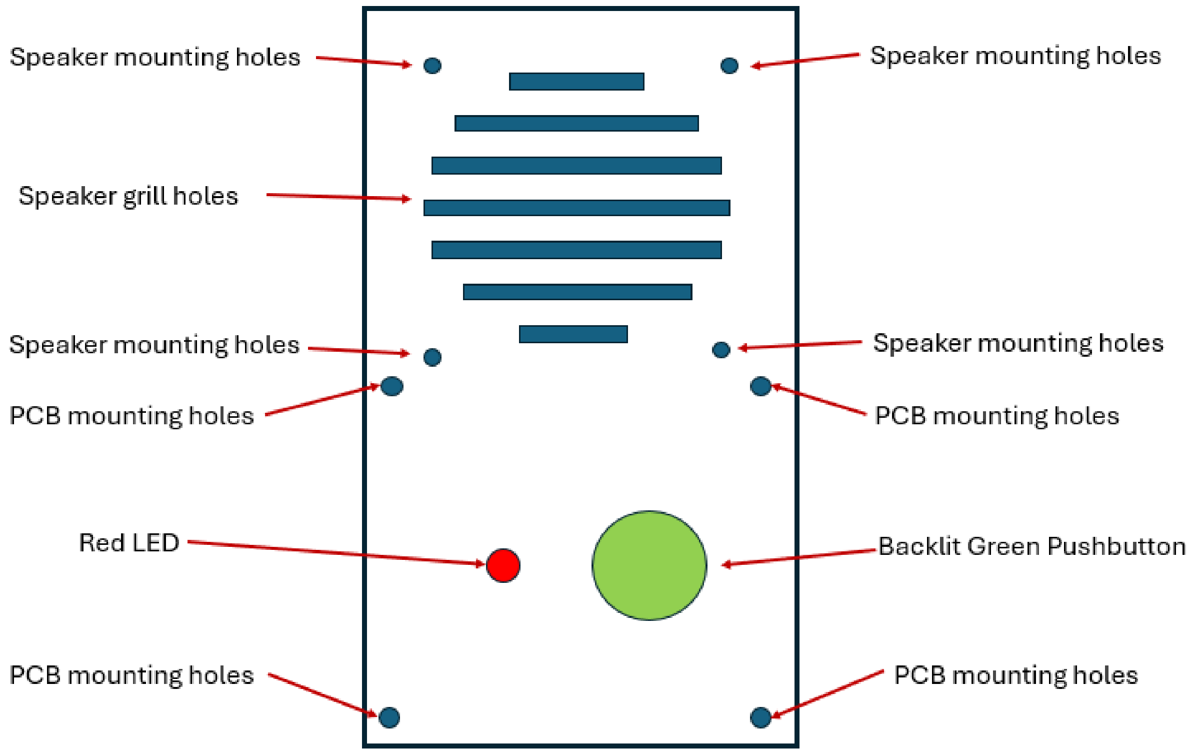

Figure 2. Annunciator front panel.

Five volt power needs to be supplied to the Annunciator via a USB type B connector that is exposed on the bottom of the enclosure. The five volt power source must drive all of the electronics and the internal speaker and should be rated at 1 ampere or higher.

When five volt power is applied, the Annunciator's internal Photon microcontroller automatically boots up and connects to the local Wi-Fi network. This process typically takes a few seconds. After connecting to the Internet via the local Wi-Fi network, the microcontroller initializes its internal parameters and subscribes to the Particle event that is configured in its internal software. At this point, the red LED turns on, indicating that the Annunciator is online and awaiting event publications.

When a subscribed-to event occurs, the internal software receives the event data and parses the data to identify the device that originated the event publication (deviceID). The Annunciator software then determines the appropriate clip to play (via information in the *ClipsList.h* file) and initiates the following sequence of events:

- The backlit pushbutton flashes its green LED for 1 second
- The clip plays through the internal speaker while the green backlight continues to flash

- When the clip is complete, the green backlight continues to flash for 1 second

A user can re-play the last clip that the Annunciator played by pressing the front panel pushbutton. The clip re-plays with the same sequence as above.

Setting the Playback Volume.

The playback volume can be adjusted by calling a Particle Cloud function. The Particle Console may be used to read out the current volume level and to set a new volume level that is appropriate to the environment where the Annunciator is to be placed – see TESTING below for details. Volume is relative to the maximum of 100 (%). Clips should be recorded at the maximum volume possible and the playback volume then used to reduce the volume to an appropriate level.

Recording and Installing Audio Files.

The mini MP3 Player has a micro SD slot that accommodates micro SD cards of up to 32 GB. The Annunciator software expects to see pre-recorded clip files in a folder directly under the root: /MP3/

The audio files in this folder must be named with a 4 digit number that can then be appended with any valid text, e.g.:

0021ReceptionDesk.mp3

Note that the clip files are nominally in MP3 format; however, the miniMP3 Player data sheet says that other formats such as .wav files may be used.

The clip files in the /MP3/ folder are identified by the Annunciator software using the 4 digit number at the beginning of the files name. The rest of the file name is not relevant to the software.

The means of recording clip files for the Annunciator are beyond the scope of this document. Any voice recorder (such as Windows voice recorder) may be used to create clip files. Alternatively, there are several on-line services that produce synthesized voice files from text data. If the files produced by some app or service are not in MP3 format, there are many free online converters available that can convert non-compatible formats (e.g. MP4A) to mp3.

Audio editing software packages such as Audacity may be used to edit recorded clips before writing them to a micro SD card. The clips may need to be shortened, to have blank leaders removed, to have the volume adjusted, etc. After the audio files have been suitably reviewed and processed, they should be named according to the 4 digit leading number convention and written to the /MP3/ Folder on the micro SD card that is to be inserted into the mini MP3 Player module.

The pre-recorded clips may be played through the Annunciator for testing purposes, prior to deployment. See the TESTING section of this document for details.

THEORY OF OPERATION.

Hardware.

Figure 3 shows the schematic for the Annunciator electronics. Note that the printed circuit board housing the electronics is re-purposed from another project (https://github.com/TeamPracticalProjects/Animatronics). The Animatronics project has circuitry on the printed circuit board that scales, filters and extracts the “envelope” of an audio clip during playback. This functionality is not required for the Annunciator project and the circuitry indicated on figure 3 should not be installed on the circuit board.

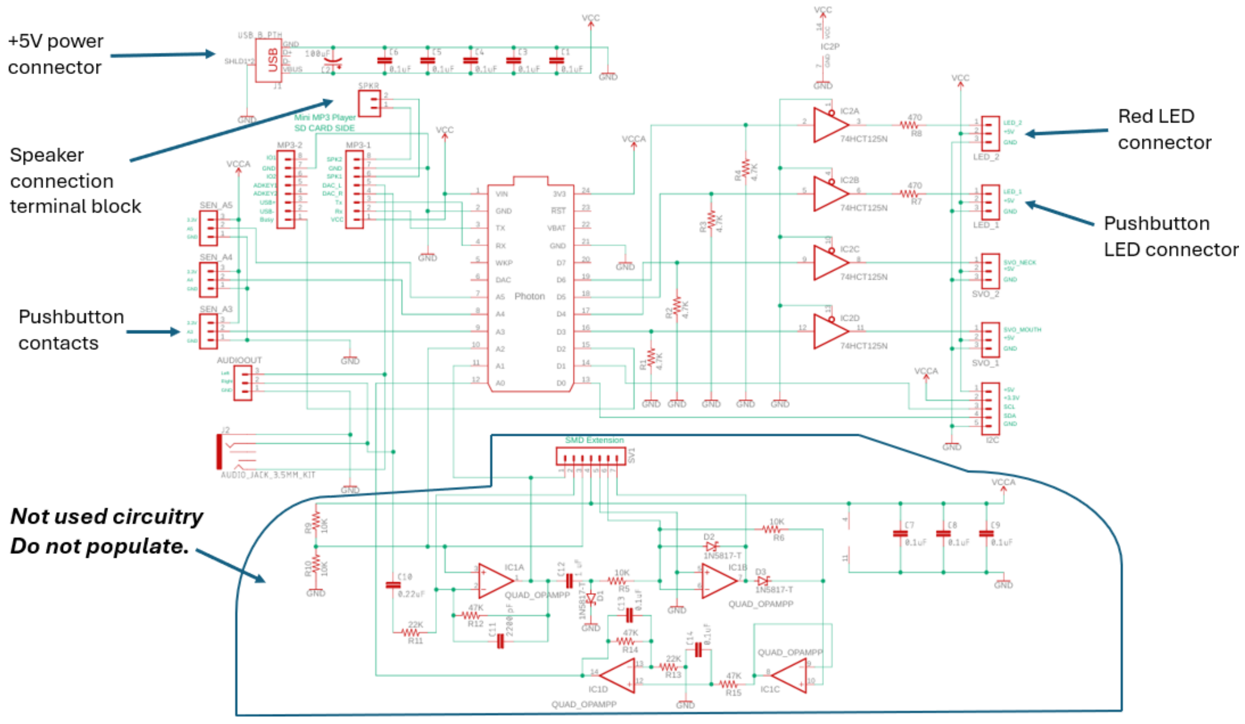

Figure 3. Annunciator Electronics Schematic.

The Annunciator uses a Particle Photon[4] Wi-Fi enabled microcontroller. The Photon microcontroller communicates with the mini MP3 Player module via serial I/O using the Photon's *Serial1* port pins:

- Photon Tx to mini MP3 Player Rx
- Photon Rx to mini MP3 Player Tx
- miniMP3 Player *busy* pin to Photon pin D2

Note that both the Photon and the miniMP3 Player are powered from the 5 volt input (supplied via the USB-B connector); however, both devices contain onboard regulators and operate at 3.3 volt logic levels.

The mini MP3 Player contains an on-module 3 watt audio amplifier. SPK1 and SPK2 pins connect to a 2 pin terminal block for connection of a small loudspeaker. The mini MP3 Player DAC_L and DAC_R lines are stereo outputs and are connected (with GND) to the phono jack on the printed circuit board.

A 3 pin male pin header installed on SEN_A3 is used to connect the pushbutton contacts to the Photon using Photon pin A3. One pushbutton contact is wired to the A3 input and the other is wired to the GND input.

IC2 is a 74HCT125 quad driver chip. It is used to convert 3.3 volt logic levels from Photon pins D6, D5, D4 and D3 to 5 volts for driving external LEDs. 470 ohm current limiting resistors on the D6 and D5 driver outputs provide for direct connection of red and green LEDs. The LED_2 pin header LED_2 pin connects to the anode (+ side) of the red LED and the cathode (- side) of the LED is connected to GND. Similarly, The LED_1 pin header LED_1 pin connects to the anode (+ side) of the green backlight LED and the cathode (- side) of the LED is connected to GND.

Software.

The Photon software for this project is in this repository:

Software / PhotonSoftware / Annunciator_code / src

There are two source code files in the src folder:

- **Annunciator_code.ino**: This is the main source code for the Annunciator
- **ClipsList.h**: This file is included in the main code and contains mapping information between deviceID data from an event to the clip file number to be played.

[4] Original Photon: Photon 1, now deprecated

Annunciator_code.ino subscribes to a Particle Cloud event using a *Particle.subscribe()* statement in *setup()* (line # 405 in the code). This causes the device to subscribe to an event called "LoRaHubLogging" and to execute the handler function called "particleCallbackEventPublish" (line # 361 in the code). This event handler function processes the event data to extract the deviceID that triggered publication of the event. It then uses the array in the included file "ClipsList.h" to determine the clip that is to be played and it sets a flag to indicate that a new clip is to be played.

The main *loop()* is primarily a large state machine whose states are defined in the *enum StateVariable {}* at line # 131 in the code. The use of a state machine provides a completely non-blocking operation of *loop().*

A second state machine is for non-blocking debouncing and processing of the pushbutton switch. The states for this state machine are defined in *enum ButtonStates {}* at line # 142 in the code.

BUILD INSTRUCTIONS.

Hardware Build Instructions.

Figure 4 shows a populated printed circuit board with major components connected to it.

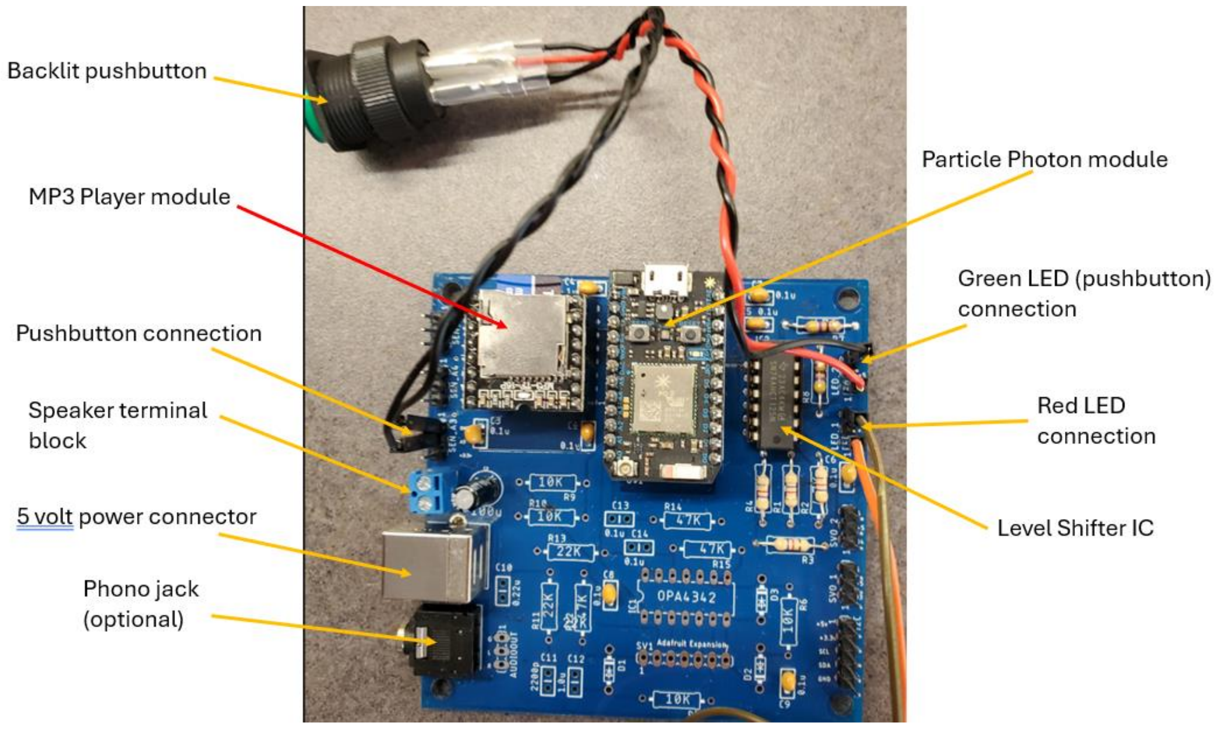

Figure 4. Major Components.

CADsoft Eagle source files for the printed circuity board are contained in the repository:

https://github.com/TeamPracticalProjects/Animatronics/tree/main/Hardware/Eagle_Files

The zip file contains the manufacturing files that can be used to obtain boards from JLCPCB.com or similar on-line vendors.

A parts list for these components is in this repository under:

Hardware / Annunciator parts list.pdf

We suggest using female pin headers for mounting the Photon and the mini MP3 Player to the printed circuit board. Solder all other components to the board as shown in figure 4.

Connections for the red LED and the pushbutton are shown in figure 4 as well, and are as described in the THEORY OF OPERATION section of this document.

The pushbutton switch can be obtained as follows:

https://www.adafruit.com/product/1440

This device has 4 tab connections on the back. The two larger connections are for the pushbutton contacts and are not polarized. One goes to the A3 pin of the SENS_A3 pin header and the other goes to GND. The two smaller connections are labeled + and – and are for connecting the backlight LED. The + connection goes to the LED_1 pin of the LED_1 pin header and the – connection goes to GND of this pin header.

The red LED is any 3 mm red LED device and is connected to the LED_2 pin header as described in the THEORY OF OPERATION section of this document and shown in figure 4, above.

A suitable speaker can be obtained from:

https://www.adafruit.com/product/1313

Wires are soldered to the speaker terminals and connected to the 2 pin terminal block on the printed circuit board.

The enclosure for the Annunciator is laser cut from 3 mm (1/8") birch plywood stock based upon a custom-designed pattern. The CAD file for this enclosure is found in this repository at:

Hardware / LaserCutEnclosure / annunciator_1.svg

The enclosure was designed in Inkscape and cut using an Epilog Helix 45 watt laser cutter/engraver, but the .svg file should work with any laser cutter.

All components of the annunciator are mounted on the rear of the front panel, as shown in figure 5.

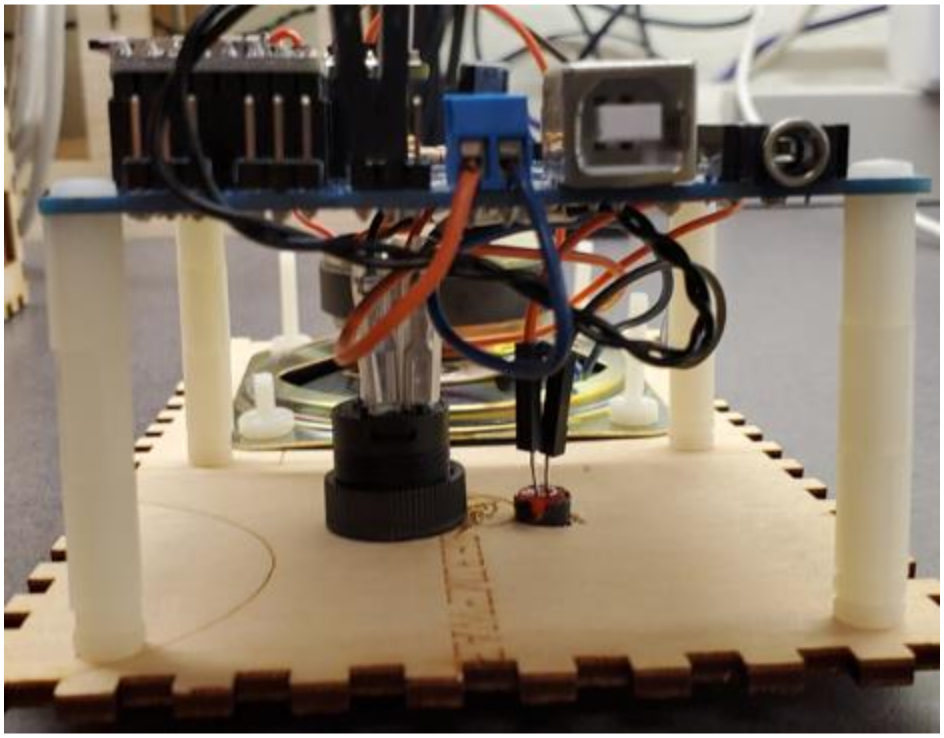

Figure 5. Components Mounted on the Front Panel.

The speaker is directly mounded to the back of the front panel; 6-32 hardware is suggested.

The red LED is push mounted through the front panel using standard 3 mm LED panel mounting hardware. Crimp on Dupont female leads are suggested for wiring to the LED, but the leads can be soldered if preferred.

The pushbutton is inserted through the front panel from the front and secured using the integral plastic nut to the rear of the panel. Crimped swage connectors may be used for connecting to the pushbutton connectors, but wires can be soldered to these connections if preferred.

The assembled and tested printed circuit board is mounted in back of the red LED and pushbutton using 1-3/4" long threaded standoffs, as shown in figure 5. Note the orientation of this board: the USB-B connector and phono jack must be at the bottom of the front panel, as shown in the figure. #4-40 or M3 mounting hardware is suggested. The speaker, pushbutton and red LED are connected to the printed circuit board as described above.

The back, sides, top, and bottom pieces of the enclosure form an empty box when assembled. Care should be taken to ensure that the pieces fit together tightly and that the front panel fits properly on the assembled enclosure box. The laser cut piece with a slot cut into it is the enclosure bottom and when the enclosure is assembled and the front panel is inserted properly, the USB-B and phono jack connectors should be accessible through this slot.

The enclosure back, sides, top, and bottom should be glued together using ordinary wood glue and left to dry. The front panel should fit snugly on the assembled and glued box and does not need further fastening. The front panel should not be glued to the box, as it may need to be removed for maintenance purposes.

Instructions for Flashing the Photon Software.

The source code for the Annunciator's Photon microcontroller is contained in this repository:

Software / PhotonSoftware / Annunciator_code / src

There are two files in this folder:

- **Annunciator_code.ino**: the main software for the Annunciator and the code that gets compiled.
- **ClipsLish.h**: a c++ header file that is included in Annunciator_code.ino

Both of these files need to be in the same folder.

In addition, the code includes the Particle port-in of the Arduino library for the mini MP3 Player, in:

Software / PhotonSoftware / Annunciator_code / lib / DFRobotDFPlayerMini

This folder contains all of the library files:

- *DFRobotDFPlayerMini / src* contains the .cpp and .h source files for the library
- *DFRobotDFPlayerMini / examples* contain subfolders with example files (as is usual with Arduino libraries).

The structure of the *Software / PhotonSoftware / Annunciator_code / src* folder is already set up properly for use with the Particle workbench to compile and over-the-air flash the code to the Annunciator's Photon.

When preparing to compile and flash this code, **Particle OS version 3.0.0 *must be selected***. This is because the DFRobotDFPlayerMini library contains several bugs. There are several functions in this library that are defined as returning *int* values but lack a *return* statement. These functions are not used by our software and the return values are not used by the example code that comes with the library. OS version 3.0.0 treats these errors as warnings and the library code compiles and executes just fine ignoring the compiler warning messages. *Later versions of the Particle OS are stricter and won't compile the code!* If you need to compile with a later version of the OS, you should check and see if a fixed version of the libraries is available. If a fixed version is not available, you can probably fix the libraries yourself by noting the compiler errors and either adding a *return 0;* statement at the end of each offending function or else re-defining the offending functions to return *void* rather than *int*.

MODIFYING THE SOFTWARE.

The Annunciator software in this repository contains some project-specific parts that may need to be modified when the Annunciator is used on a different project. The following sections highlight these areas of the code.

Event handler function.

The event handler that is called whenever there is a new publication to the subscribed event is in the code beginning at line # 359. This code expects the event data to be a String that contains the deviceID as part of this String. The event handler calls another function *eventDataParse()* that parses the deviceID out of the event data String and converts it to an *int*. It then subtracts a base deviceID, which is called `BEGIN_DEV_NUM,` and is defined in the file *ClipsList.h*. This process produces an index into the *ClipList[]* array that is also defined in *ClipsList.h*. The *ClipList[]* array contains the numerical part of the file number of the clip stored on the micro SD card that is inserted into the mini MP3 Player. Some or all of this function may need to be changed to fit the requirements of a different application of the Annunciator.

Event data parsing function.

The function *eventDatParse(String evData)* is responsible for parsing the data String that comes with the event publication in order to extract the deviceID from it. Even if the overall structure of the project uses the current event handler intact, the message String that is the returned event data will likely differ from what this code expects. That is why the functionality of extracting the deviceID from the event data String is isolated to its own function. The code in this repository should be easy to follow and to modify, as needed, to fit the needs of a different project. The event data parsing function *eventDatParse(String evData)* starts at line # 314 in the code.

ClipsList.h.

This file contains the specifics for mapping device IDs extracted from the event data to the actual clip to be played. The actual clip to be played is designated by the leading 4 digit number in its file name in the /MP3/ Folder on the micro SD card.

ClipsList.h defines an array called *ClipList[]*. The device ID, minus the defined constant BEGIN_DEV_NUM is the index into the *ClipList[]* array. The implicit assumption here is that the deviceIDs associated with voice clips that are to be played are numbered sequentially, beginning with BEGIN_DEV_NUM. The defined constant MAX_NUM_CLIPS is the size of the *ClipList[]* array. There are several other defined constants for default messages that should be included on the micro SD card.

TESTING.

A number of Particle Cloud variables and functions are included in the Annunciator code to aid in configuring and debugging the Annunciator. The Particle Console can be used to access these variables and functions. When the Annunciator Photon is selected in the Console, these Cloud variables and functions can be accessed by the controls at the bottom right of the Console screen, as shown in figure 6.

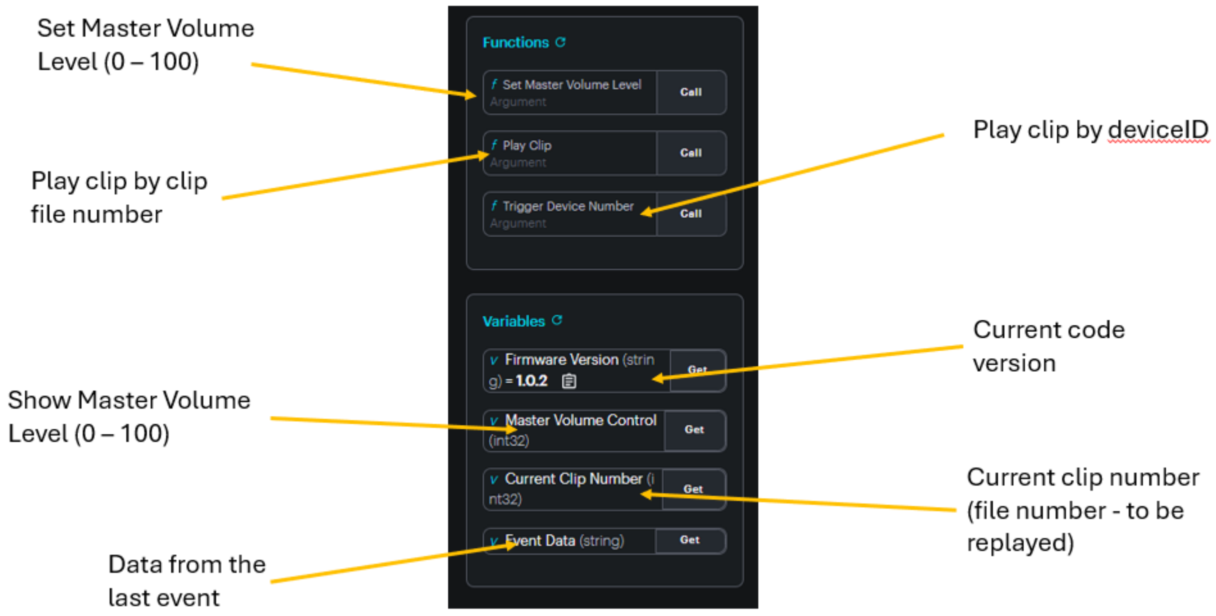
Set Master Volume Level (0 – 100)
Play clip by clip file number
Show Master Volume Level (0 – 100)
Data from the last event
Play clip by deviceID
Current code version
Current clip number (file number - to be replayed)
Functions
Set Master Volume Level
Argument
Call
Play Clip
Argument
Call
Trigger Device Number
Argument
Call
Variables
Firmware Version (string) = 1.0.2
Get
Master Volume Control (int32)
Get
Current Clip Number (int32)
Get
Event Data (string)
Get

Figure 6. Console access to Cloud Variables and Functions.

Cloud Variables.

See the lower part of figure 6 under *Variables*. The designated variable is accessed by clicking on the Get button and the value is displayed under the Variable name.

1. Firmware Version. This displays the version of the Annunciator software that is currently flashed to the selected Photon device. The current version, as of this writing, is 1.0.2.
2. Master Volume Control. This displays the current setting of the playback volume that is stored in non-volatile memory on the Annunciator's Photon. The values are 0 – 100 (%), where 100% is the maximum volume available from the mini MP3 Player.
3. Current Clip Number. This displays the 4 digit prefix of the file name of the file on the micro SD card that was the last clip played. This is the clip that will be replayed if the pushbutton is pressed.
4. Event Data. This displays the String that was received by the Annunciator as the data associated with the subscribed event. This is useful for debugging the parsing of this String to obtain the deviceID.

Cloud Functions.

See the upper part of figure 6 under *Functions*. The designated function is called by clicking on the Call button. The function argument must be entered in the text box to the left of the Call button. The function return value is displayed under the argument in the text box.

1. Set Master Volume Level. A number between 0 and 100 is expected as the function argument. This entry sets the playback volume level for the Annunciator. This value is stored in non-volatile memory so that it may be set once to fit the installed environment and left at that setting thereafter.
2. Play Clip. The argument is the 4 digit prefix of a file number of the clip in the /MP3/ folder of the microSD card. When the function is called, this clip will be played through the Annunciator's speaker. This Cloud function is useful for testing playback, volume and for selecting clips for inclusion in *ClipsLIst.h*.
3. Trigger Device Number. The argument is the deviceID that would be contained in the data String from the subscribed event. It is useful for testing that the correct list of clips is contained in the *ClipsList.h* file.