Maker Nexus Help System
Technical Description Document

By: Jim Schrempp and Bob Glicksman; v1, 9/5/2025

NOTICE: Use of this document is subject to the terms of use described in the document "Terms_of_Use_License_and_Disclaimer" that is included in this release package. This document can be found at:

https://github.com/TeamPracticalProjects/MN_Help_System/tree/main/Documents/Terms_of_Use_License_and_Disclaimer.pdf

TABLE OF CONTENTS.

INTRODUCTION.

This document describes the Help Button System that was developed for Maker Nexus[1]. The system is easily modified for use in other venues.

The Help Button system is based upon developmental work by Team Practical Projects and heavily leverages hardware and software developed for these projects; specifically:

https://github.com/TeamPracticalProjects/LoRa_experiments

https://github.com/TeamPracticalProjects/Annunciator

https://github.com/TeamPracticalProjects/Event_Timer

This document provides complete instructions for building, deploying and using the Help Button system. *Much of the material in this document is included by reference to the aforementioned GitHub repositories.*

Key technologies used in the Help Button system are LoRa[2] and Particle[3] IoT. LoRa technology provides the system with the ability to send short messages over long distances with limited power requirements. Particle IoT technology provides a cloud-based publish and subscribe capability and well as a REST API to access Particle devices from the outside world.

HELP BUTTON SYSTEM DESCRIPTION.

System Requirements and Operation.

Maker Nexus is a 501c3 educational non-profit makerspace located in Sunnyvale CA. The Maker Nexus facility comprises about 28,000 square feet of floor space that is sectioned into various operational areas, including:

- Woodshop
- Metal shop (cold shop)
- Welding and Plasma Cutting (hot shop)
- Electronics-Robotics Lab
- Textiles

[1] https://www.makernexus.org

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

[3] https://www.particle.io

- Laser cutting/engraving
- 3D Printing
- Dye sublimation printing and vinyl cutting
- Classrooms and meeting rooms
- General crafts working area

Most of these areas are not visible from other areas within the facility.

Maker Nexus stresses safety and security. At least one staff member must be present in the facility anytime that it is open for use by members. It is Maker Nexus policy that members remain in their work area anytime that power equipment is in use. This means that members generally cannot see a staff member to summon assistance when it is needed.

The Help Button system was developed to allow members to summon staff assistance from within sight of their in-process work, regardless of where staff members might be in the facility. Help Buttons (figure 1) are deployed throughout the facility for this purpose.

Figure 1. Help Button.

Each Help Button contains a backlit pushbutton on its face. A member who needs assistance simply presses the push button. The push button is backlit and flashes once to indicate that a help request message has been sent, and then three times quickly to indicate that the message has been received and processed by the system. The Help Buttons are self contained and battery operated so that they may be placed anywhere necessary without regard to the availability of power, Internet, or any other external support.

Devices called “Annunciators” (figure 2) are placed throughout the facility. There are enough Annunciators so that staff members can hear an announcement from any location.

Each Annunciator contains an internal loudspeaker and a backlit push button. Whenever a member presses a Help Button, all Annunciators in Maker Nexus flash their backlit pushbuttons and play an appropriate audio clip, e.g. “Assistance needed in the Woodshop”. These announcements summon a staff person to the location of the member, fulfilling the policy that members remain by their in-process work.

Figure 2. Annunciator.

Staff members can re-play the previous announcement by pressing the backlit push button on an Annunciator.

After initial deployment of the Help Button system, Maker Nexus staff requested support for playing pre-recorded administrative audio clips on the Annunciators, e.g. “Maker Nexus is closing in 30 minutes. Please begin packing your materials and cleaning up your work area.” An Android app (figure 3) was created to allow staff personnel to play these messages.

Figure 3. Android App Screen Photo.

This app is deployed on an Android smart phone that the staff member on-duty always carries with them. The staff member simply taps a button on the app to cause the appropriate pre-recorded message to play on all Annunciators.

In order to automate playing certain recurring administrative audio clips at the appropriate times of the day, an additional component called the “Event Timer” was added to the Help Button system, see:

https://github.com/TeamPracticalProjects/Event_Timer

The Event Timer is an independent component of the system that publishes its own events to the Particle Cloud that the Annunciators subscribe to.

All announcements, either from Help Buttons or the App, are logged to a Google spreadsheet so that system utilization can be monitored and tracked.

Technical Overview.

Figure 4 is an overview of the components and data flows that comprise the Maker Nexus Help Button system.

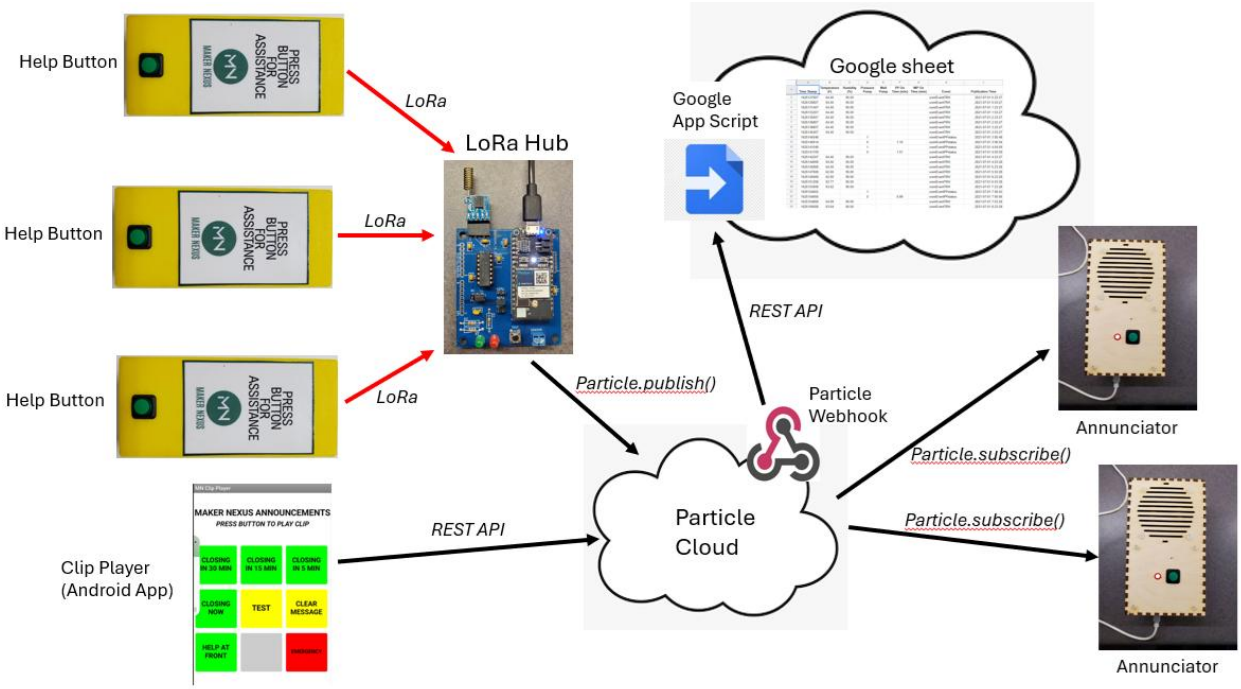

Figure 4. Help Button System Overview.

Help Buttons are battery powered devices that are located at strategically selected locations throughout the Maker Nexus facility. Pressing a Help Button wakes up its internal circuitry from a very low power “deep sleep” state. The button sends out a LoRa message. The message includes the device ID that is configured on the Help Button and identifies the area where the Help Button is deployed. Jumpers on the Help Button printed circuit board allow the Help Button to be set to one of 8 different device IDs that are associated with a base ID configured in the Help Button firmware.

LoRa messages sent from Help Buttons have a fixed destination ID which is the ID of the singular LoRa Hub that is an integral part of the system. The Hub is based on the Particle Photon 2 WiFi microcontroller.

The Hub is always powered up and listening for new LoRa messages. When the Hub receives a LoRa message, it:

- Validates the message as coming from a Maker Nexus Help Button.
- Acknowledges the message with a reply message back to the Help Button.
- Publishes an event to the Particle Cloud. The event data contains the device ID of the sending Help Button.

After being pressed, Help Buttons remain powered up briefly, listening for a response message from the Hub. When a response message is received, the Help Button that sent the original LoRa message flashes its backlit pushbutton 3 short times and then goes back into very low power “deep sleep”.

Annunciators are Particle-based devices that subscribe to LoRa Help System events published to the Particle Cloud. Normally, the Help Button System Hub publishes these events. The Event Timer component can also publish these events.

Annunciators use the older Particle Photon 1 WiFi microcontroller and a DFRobot mini MP3 player that can play pre-recorded audio clips through an internal 3” loudspeaker. Annunciators are placed at strategic locations throughout Maker Nexus so that staff personnel can hear any announcement played through the Annunciator’s speaker wherever they may be within the facility.

Event handler software within each Annunciator runs every time a Help Button System event is published to the Particle Cloud. Annunciator software decodes the data associated with the published event and extracts the device ID of the initiating device. Each unique device ID is mapped (in the Annunciator software) to a pre-recorded audio clip that is stored on a microSD card inserted into the mini MP3 player module in the Annunciator. The LED in the backlit pushbutton on each Annunciator flashes while the audio clip is played. The pre-recorded audio voice clips summon a staff member to the location of the Help Button that originally sent out the LoRa help message.

Publication of LoRa Help System events to the Particle Cloud also fires off a Particle Webhook. The Webhook, in turn, POST’s data to a Google Apps script that is associated with a Google cloud spreadsheet. The script appends a log entry to the spreadsheet so that system operation can be monitored and mined for data about system utilization.

In addition to pre-recorded help messages, various administrative audio clips are recorded on the micro SD cards in each Annunciator. These clips play general announcements, e.g. announcements made daily near closing time that remind members to wrap up their work-in-process and clean up their work area. These administrative audio clips are also mapped to device IDs; however, physical Help Buttons are not deployed with these device IDs. Rather, an Android app uses Particle REST technology to simulate a Help Button in the Hub software and cause the Hub to publish an appropriate event to the Particle Cloud. Maker Nexus staff members can use the app to manually trigger playing of these administrative messages as needed.

An additional component of the system called an Event Timer works independently of the LoRa Help Buttons and Hub. The Event Timer is not shown in figure 4. The Event Timer automates the daily playing of closing-related audio announcements.

Help Button System Technology.

Help Buttons are battery operated low power devices that power up and send out a LoRa message whenever the button on the front of the device is pressed. The LoRa message contains a device ID that is configured, via jumpers on the Help Button printed circuit board, to one of 8 values relative to a base device address. Multiple Help Buttons with the same device ID are allowed. If more than 8 distinct device IDs are needed in a system, the base address of a set of 8 IDs can be changed in the Help Button software. A Help Button must be reset after changing device ID jumpers in order for the new device ID to take effect.

The Help Buttons contain low power circuitry to detect a change (opening or closing) of a contact. The polarity of the action (opening or closing) that triggers the device to send out a LoRa message can be set via an on-board jumper.

Each Help Button sends out a LoRa message when it is triggered. The LoRa messages sent out from each Sensor contains:

- The unit's device ID
- A very brief code indicating the device type.
- A one-up message count number that starts with zero (after a reset) and increments thereafter.
- RSSI and SNR readings for the previous response message from the Hub. These are logged and used to ensure that communication remains reliable over time.

There is one Hub for any given system. The Hub operates on LoRa Channel 18, and all Help Buttons must use this same channel (and the same LoRa settings) in order to communicate with the Hub. The Hub is configured (in the Hub software) with a specific LoRa device ID. The Hub device ID must be different from each Help Button's device ID and must be unique for any given system.

When the Hub receives a LoRa message, the Hub software decodes the message, checks the message for validity, responds to the sending device with a response message, and publishes an event to the Particle Cloud. The published event contains information about the received message, including:

- The device ID of the sensor sending of the message.
- The message type and message count information from the received message.
- The Hub's recorded RSSI and SNR values for the received message.

In addition, the Hub contains a Particle Cloud function that can be called (e.g. from the Particle Console) to simulate receipt of a LoRa message. This capability is useful for testing as well as for activation of simulated Help Buttons from an App. MIT App Inventor 2 code for such an App is included in the LoRa_Experiments GitHub repository:

https://github.com/TeamPracticalProjects/LoRa_experiments/tree/main/Range_Testing/Range_Test_Hub/Clip%20Play%20App

A Particle Webhook is configured to subscribe to received LoRa Hub events. This webhook does an https: POST to a Google Apps Script that is bound to a Google sheet. A configurable Google sheet is used to log messages received by the Hub.

LoRa Technology.

LoRa communication is via Reyax RYLR998 LoRa modules. These modules are low cost (approximately $12 each) and contain a LoRa transceiver and a low power microcontroller. The Reyax RYLR998 module acts as a “LoRa Modem” and uses serial asynchronous “AT” commands to communicate with a host computer or microcontroller.

Figure 5. Reyax RYLR998 LoRa Module.

The “AT” commands are documented in the repository:

https://github.com/TeamPracticalProjects/LoRa_experiments/blob/main/Documents/LoRa_AT_Command_RYLR998_RYLR498_EN.pdf

BUILDING AND DEPLOYING HELP BUTTONS.

Help Buttons consist of three parts:

- LoRa Low Power Sensor electronic printed circuit board.
- Backlit push button switch.

- 3D Printed Enclosure.

Fusion 360 CAD files for the 3D Printed enclosure can be found here:

https://github.com/TeamPracticalProjects/LoRa_experiments/tree/main/Hardware/Help%20Button%20Case

The enclosure consists of two parts:

- Wall Mount: the back of the enclosure that is designed to be mounted on a wall.
- PCB Mount: the front of the enclosure that mounts the electronics (printed circuit board and backlit push button switch).

The PCB Mount slides over the Wall Mount part to secure the Help Button to a wall. We printed both parts on an FDM 3D printer using PLA filament.

The backlit pushbutton switch can be purchased here:

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

The back of the push button looks like this:

Figure 6. Back of Pushbutton Switch.

There are 4 terminals on the push button. The terminals labeled + and – are for the LED that is the backlight. The two larger terminals that are unlabeled are the switch contacts. All 4 terminals need to be connected to the printed circuit board by wires. The wires can be soldered

on to these terminals or else spade connectors can be crimped to the wires. Suitable spade crimp connectors can be purchased from:

https://www.adafruit.com/product/4748

We recommend color coding the switch wires as follows:

- - LED terminal: black wire
- + LED terminal: red wire
- Switch terminals: green or other color wires.

The other end of these wires (the end that connects to the printed circuit board) should have standard female Dupont header pins (2.54 mm spacing) crimped onto them.

Figure 7, below, shows the mounting and wiring of the push button switch to the printed circuit board in the Help Button enclosure.

Figure 7. Mounting and Wiring Help Button Components.

Note that the wires from the switch terminals (green wires) are not polarized. They are connected to the right angle sensor contacts on the printed circuit board. The red and black backlight wires are polarized and must be wired as shown in the figure: red wire to the inside of the printed circuit board and black wire to the outside edge of the printed circuit board.

Instructions for assembling the Help Button printed circuit board can be found in the document:

https://github.com/TeamPracticalProjects/LoRa_experiments/blob/main/Documents/Low_Power_LoRa_Sensor_User_Manual.pdf

There is one exception to these instructions when the printed circuit board is to be used as a Help Button: *the green LED is NOT SOLDERED to the board*. Rather, a 2 pin male pin header is soldered where the green LED normally goes. The red and black wires from the push button switch connect to these pin header terminals, as the green LED is the backlight of the pushbutton in this application.

The printed circuit board is connected to the enclosure using M3 screws.

A jumper must be placed between the NO and center contact of the Polarity jumper field.

The address jumpers need to be placed according to the deployed location of the Help Button. See the following document for details:

https://github.com/TeamPracticalProjects/MN_Help_System/blob/main/Documents/Configuring%20a%20Help%20Button%20for%20MN.pdf

Please be sure to follow the complete instructions for building the printed circuit board, including the SOFTWARE INSTALLATION AND SETUP INSTRUCTIONS.

BUILDING AND DEPLOYING THE HUB.

The Help Button System Hub consists of a printed circuit electronic board and an optional 3D printed mount. Complete instructions for assembling and programming the printed circuit board can be found in the document:

https://github.com/TeamPracticalProjects/LoRa_experiments/blob/main/Documents/Photon_2_Hub_and_Sensor_User_Manual.pdf

An optional 3D printed mount holds the printed circuit board and protects the LoRa module that is plugged into it. 3D CAD files for this mount can be found at:

https://github.com/TeamPracticalProjects/LoRa_experiments/tree/main/Hardware/Hub%20Case

Please be sure to follow the complete instructions for building the printed circuit board, including the SOFTWARE INSTALLATION AND SETUP INSTRUCTIONS.

NOTE: The Hub Photon 2 device and all Annunciator Photon devices must all be claimed into the same Particle account in order for the Publish/Subscribe mechanism to work.

BUILDING AND DEPLOYING ANNUNCIATORS.

Instructions for building, programming, operating and testing Annunciators can be found at:

> https://github.com/TeamPracticalProjects/Annunciator/blob/main/Documentation/Annunciator_User_Manual.pdf

The Annunciators are housed in a laser cut enclosure. 2D CAD files for this enclosure can be found at:

> https://github.com/TeamPracticalProjects/Annunciator/tree/main/Hardware/LasercCutEnclosure

NOTE: The Hub Photon 2 device and all Annunciator Photon devices must all be claimed into the same Particle account in order for the Publish/Subscribe mechanism to work.

BUILDING AND DEPLOYING THE ANNOUNCEMENT APP.

The Announcement App is an Android app that allows a user to play pre-recorded administrative announcements through the Annunciators of the Help Button System. See figure 3 for the App screen. The app is written in MIT App Inventor 2. The source code (.aia file) and the installation package (.apk file) for this app can be found at:

> https://github.com/TeamPracticalProjects/LoRa_experiments/tree/main/Range_Testing/Range_Test_Hub/Clip%20Play%20App

The .apk file is downloaded or sideloaded to any Android phone or tablet and tapped to install it on that device. *You may get a warning that the app is not verified (it did not come from the Google Play store).* Click on the option to install it anyway.

Once installed, the SETUP button (lower left corner of the main screen) must be clicked to enter the SETUP page. It is necessary to log into your Particle account where the Hub is claimed and then to select the Hub from the list of devices in that account. When successful, SETUP is exited. The app should show the main screen and the box in the lower right hand corner should be green, as shown in figure 3. This indicates that the app has successfully pinged the Hub.

The Hub must be powered on and connected to the Internet in order to connect with the app through the Particle Cloud.

BUILDING AND DEPLOYING AN EVENT TIMER.

An Event Timer may be constructed and deployed as part of the Help Button System. The purpose of the Event Timer would be to publish Help System events to the Particle Cloud, in the same manner as does the Hub. These published events will trigger the Annunciators to play appropriate pre-recorded administrative messages. The system that is installed at Maker Nexus uses an Event Timer to automatically play appropriate closing messages daily at 9:30 pm, 9:45 pm, 9:55 pm and 10:00 pm (10:00 pm is closing time, when all members and staff must be out of the facility). For further information about the Event Timer, see:

https://github.com/TeamPracticalProjects/Event_Timer

LOGGING.

Logging of Help System events to a Google sheet may be accomplished as follows:

- Configure a Particle Webhook that subscribes to the Help Button System event publications (by the Hub). The Webhook should POST to a Google App Script that appends the event data to a Google sheet,
- A Google sheet cloud spreadsheet is created to house the logged event data.
- A Google App Script is created with a doPost(e) method that parses the event data out of the event object “e” that is the argument to the doPost() function.
- The script may append local date/time and otherwise process and parse the event data in order to format it for logging to the Google sheet.
- The url of the Google App Script is configured into the url field of the Particle Webhook.
- The url of the Google sheet spreadsheet and the name of the sheet that the data is to be logged to are configured into the Google App Script.

The details of the script depend upon what information is to be logged and in what format.

More information about this process can be found in:

https://github.com/TeamPracticalProjects/Connectivity_Tools_with_Particle_Devices