GSheetBells

Release 1.0.0

Oct 12, 2020

Sections:

CHAPTER 1

Outline

GSheetBells concept is using the power and hosting of google sheets, to power a bell system that is easy to configure, reliable and perfectly on time, every time.

To build, it requires medium to high levels of technical knowledgeable - as this documentation is most defiantly not perfect. You will need to estimate some things.

If you get stuck, you can contact me, the Creator, at jasper@qrl.nz. You can also make Pull Requests and raise issues with the code and documentation at the Github Repository here.

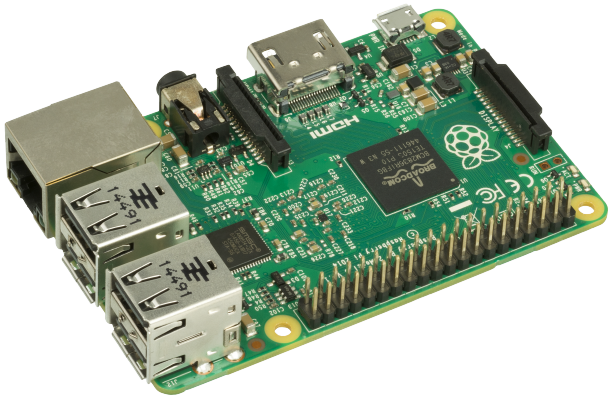
Rough Materials:

- Access to 3D printer
- Raspberry Pi (Types discussed in the next section)
- Momentary switch, normal switch
- led's red + green + suitable resistors
- Relay suited for your existing Bell System (Or tone generator if you have to, but it is not officially supported)
- USB power for the Raspberry Pi + Ethernet cable

1.1 Setting Up The Raspberry pi

This is really up to you on which one you use, you can just use any slow ones sitting around, although make sure it *dose* have an ethernet port (You can use wifi, it's just more prone to password changes etc). I used an old raspberry pi 2 in my case. You also need the SD card to act as the system drive.

Note: If you get stuck anywhere on this section, Search engines, like Google, is your friend and will have the answer if you look hard enough.



1.1.1 Installing Raspbian

Go follow this guide to create a bootable raspbian sd card. Install Raspberry Pi OS Lite, (under the *Raspberry Pi OS Other* tab in there software).

Now with the operating system on the SD card, place a file name `ssh` with no extension into the boot partition on the SD card.

1.1.2 Configuring the Pi for operation

First, we need to connect. Plug in the Raspberry Pi into ethernet. Next on another computer that is on the same network, open a terminal (It's called command line on windows) and try running `ssh pi@raspberrypi` to open a remote session to the Raspberry Pi. The default password *raspberry* which we will change soon.

Note: If this dose not work, then make sure the ethernet is plugged in properly and if it still dose not work, plug in a monitor and keyboard, log in, and run ''ip address" to find the ip address, then try again on the other computer: `ssh pi@[Ip address here]`. If all fails, run the next commands manually and enable manually in the *raspi-config* -> *Interfacing options* menu.

First, configure the operating system by running:

```
sudo raspi-config
```

Use the arrow keys and enter to enter the network sub-menu. Change the Hostname something relevant (make sure to make a note). Wait then enter the Interfacing options and enable **I2C**. Once that is done, the last thing to do in these settings is in the `Boot Options` -> `Desktop / CLI menu` select `B1 Console Autologin` You can then exit raspi-config. Next you want to change the name of the default user by executing:

```
sudo passwd
```

and following the prompts.

Next clone the code into Raspberry Pi's home directory `git clone https://github.com/Fallstop/GSheetBells.git` Then configure permissions

```
chmod +x GSheetBells/BellRinger.sh
chmod +x GSheetBells/InternetStatus.sh
chmod +x GSheetBells/StartScripts.sh
```

After that, install the dependencies using this command:

```
sudo apt update && sudo apt upgrade &&
sudo apt install python3 python3-pip screen &&
sudo pip install -r GSheetBells/requirements.txt
```

Note: This will take ages.

Nice, time to set up the auto start. For this, we are going to use Screen, which allows us to have sessions running in the background that can be connected to. Edit the start processes by running

```
sudo nano /etc/profile
```

and adding this to the end:

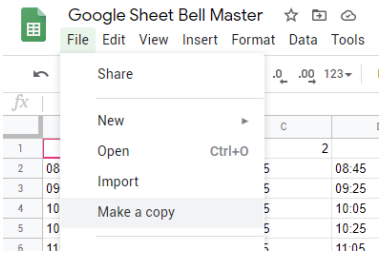
```
cd ~
sh /home/pi/GSheetBells/StartScripts.sh
```

Cool, Next up is setting up Google Sheets

1.2 Connecting to Google Sheets

1.2.1 Getting the Template

Follow this link - https://bit.ly/GSheetBellTemplate and make a copy



You can have an explore how it works, but the time inputs are next:

1.2.2 Layout and Rules

Their is a few custom scripts and formatting in the google sheet that you copied

- The bell times on the main page (`MainTUI`) are taken, sorted and made more machine friendly and placed onto the `DataSorted` page. This process means you can have gaps in your bells and have them in any order. This is done through the use of named ranges. This can sometimes be mucked up by cutting and pasting, so if the second page is missing bells, make sure the Named Ranges look like they cover the range (They are under Data menu).
- There is a hash script that supplies a hashing function used in the `DataSorted` page. It produces a hash that the Raspberry pi uses to work out if anything has changed to it's local copy (To reduce the number of writes to the SD card and chance of a corrupted local copy)

- The bell registering area is capped to the space for a thousand bells, that number is arbitrary but the sensing area has to be capped somewhere. If you want to extend this, you can by moving the hash cap down to where you want it and updating the Named Ranges from ending at 1000, to ending at your new end.
- The bell times them selfs must be a 4 characters long, 24 hour and be separated by a colon. EG: 8:45 am: `08:45` | 2:30 pm: `15:30`

You can add in your bell times now, if you want.

1.2.3 Google Cloud Platform

If you have used GCloud before, you can skip this little section.

Go to GCloud Console, accept the terms and conditions. Now go to the Google Sheet Quickstart, click enable google sheet API. Use the desktop client. Now save the Client ID and Client Secret somewhere and download the Client Configuration.

Next we need to generate the Token and the easiest way to do this is on your computer.

Download the Repository by either Cloning or Downloading a zip.

Open a Terminal (Command Line in windows) and cd to the location of the Local Repository. Make sure you have python3/pip installed and run:

Note: With Windows, use python/pip command instead of python3/pip3

```
pip3 install -r requirements.txt
```

While that is installing, you can place the credentials you downloaded earlier into the Python folder in the Repository and fill out the config file in the same folder.

Once both are done, run the script in the Python folder

```
cd Python
python3 BellRinger.py
```

It will take a bit to load, then it will open a webpage for you to authenticate it with the google account which has the Google Sheet. Check it has an error, if it dose, check your config.py is correct. If it works, your good to go.

Now we need to put the `config.py` and `` `credentials.json `` onto the Raspberry Pi. You can do this how you normally do this kind of thing, or you can take the SD card out of the Raspberry Pi and Dump it into the `home/pi/GSheetBells/Python/` folder. You can also do it via scp (If you have it installed), it would look like this:

```
scp ./token.pickle pi@HostnameHere:~/GSheetBells/Python/token.pickle
```

Time to test it!

```
cd ~/GSheetBells/Python
python3 BellRinger.py
```

Sample Output

```
Output here (Need to run on desktop)
```

If all works, then good, if it doesn't and you get a error message you can't work out, then good luck.

That is all you need to do for setting up Google Sheets!

1.3 Wiring the components

This is one of the more complicated parts of tutorial, but it should be pretty easy once you get a grip on how it works.

Almost all of these stages are optional as you may not want to implement all of the features.

Default features that you can change:

- Status LED's for internet (1 green, 1 red) and power (1 red LED).
- Relay that protects the Raspberry Pi from the power of the existing setup and triggers the bells. This could in theory be simply replaced for a tone generator if your existing setup relies on that.
- Mute bell switch, a switch that mutes the automatic ringing.
- Ring now bell switch, a switch that bypasses the relay and forces the bells to ring.

I have split this into multiple parts that you can include or exclude.

1.3.1 Relay and switches

These is the base parts that you probably want in your bell system, the relay to control actual bell system, and the ever convenient fast access switches on the top. To get started, you need to plug/solder your electronics using this information:

- Connect the data pin to pin 12 on the raspberry pi
- The ring now switch just dose the job of the relay, just connect across the wires the relay uses
- The mute switch just interrupts the wire going to the relays data pin.

This diagram will probably help, top left is nearest to the SD card on your raspberry pi:

1.3.2 Monitoring LED'S

Allows others and you to see what is wrong at a glance.

1.4 3D printing the Case

1.4.1 Case Model

I have made a template box off my design, only thing is it is done in Fusion 360, feel free to redesign it in a superior program (like blender). You will probably need to adjust the deign for your use case (switch sizes, LED's, screw sizes, ect).

The template case is open to any changes, as it as fair load of printing problems.

1.4.2 3D printing

Usually, it's not super hard to find a 3D printers; as it is an expectation to have them at high-schools, universities and large libraries. If you have no experience with 3D printing, it's not hard to learn and usually, there will be a person that will be helpful around these things.

For the filament, use whatever cheapest, easy to print plastic lying around as it is not going to see much wear and tear. PLA is perfect for this kind of project.

Take your best guess at the settings considering your printers abilities, and let it off.

1.4.3 Assembly

Note: Make sure to remove any support

If you have goon for a custom case, well done, and you should prepare the case how you planed. Otherwise, you should no that the screws on the top lid is M5 and it doesn't matter how long they are. A optional decoration is to paint the letters on the top and sides.

Putting all the connected wires and boards should be pretty easy, make sure you get the LED's in the right holes, and the raspberry pi ethernet port