CordieBot Design and Software

Photo: Jennifer Patselas

A creation by Hal Breidenbach
2020

Table of Contents

Background

This project was one of the most complex that I have worked on as an individual. The initial concept was simple, and contained in the first version. Using a repurposed lamp as a body, make a talking head which would tell time, make some inane comments in a somewhat mechanical voice, and display a light show.

The first version, simply named CordieBot, was based on an Arduino computer. It did the light show, had a number of comments to recite, and some time after my granddaughter Cordie received it she delegated it to back of a bookshelf.

Figure 1 The Lamp

Clearly, a more interesting bot was needed. But the capacity of the Arduino had been exhausted. Enter the RaspberryPi 3 A+. This computer could fit inside the lamp body, and had plenty of horsepower to do more processing. In addition, it has built in WiFi.

The CordieBot now can give the time of day along with informative or inane comments in a reasonably natural sounding voice, give the temperature and weather outside, recite a quote for the day, and when it gets warm inside its body it can turn on a cooling fan. It has a comparable light show to its predecessor, it can be updated using a USB drive, and the messages it recites can be updated over the web using secure communications.

Additionally, since the messages will be spoken, I wrote a script to run on the RaspberryPi to simply state the message so I can listen to it before I insert it into the file of messages that are played after the time is spoken.

The new capabilities were initially all programmed using Python, but when checking memory usage, it was clear that there was a memory leak, and a significant one. The memory on the RaspberryPi was totally filled in less than 24 hours. I was using a number of packages downloaded from free sites, and I could not locate the leak. After some effort at fixing the problem, I decided that the easier path would be to rewrite the code in c++.

Overall, the current CordieBot is 3-1/2 years in the making, although there have been long gaps in the development process. It contains about 3700 lines of original code in addition to the downloaded libraries.

CordieBot components

Hardware—a repurposed lamp, RaspberryPi, lights and sound.
CordieBot software—resident on the CordieBot, written in c++.
CordieBot message management software—resident on the CordieBot, written in c++.
CordieBot update scripts—resident on the CordieBot, written in shell script.
Message web page—running on a host computer, written with JavaScript with a CSS file.
Message test software—running on a RaspberryPi so messages can be heard before being sent to the CordieBot. Written in c++.

The learning curve

The CordieBot software challenged me to learn new applications, tools, and techniques:

1. Getting PubNub (the communications software) to work on the RaspberryPi.
2. Writing a web page to communicate with the CordieBot 2.
3. Using json to format and decode the messages.
4. Interfacing an Analog to Digital translation chip to the RaspberryPi and generally using the R-Pi hardware I/O.
5. Programming the speech functions and multi threading the software to overlap the light show and speech.
6. Updating the WiFi access from a USB drive.
7. Updating its own programs from a USB drive.
8. Using a single button to control the program functions.

Speech

For the natural sounding speech that CordieBot 2 has I purchased the Callie voice from Cepstral (cepstral.com). The speech is actually delivered through a program called *Swift*, and is accessed using aoss on the R-Pi. I slowed the speech down about 1/3 to make it clearer and easier to understand. It is possible to insert pauses, stress and pitch changes, and phonetic strings into the speech commands. I found the pauses useful, the others not as much.

Light show

While the speech capability is a major feature of the CordieBot, the light show adds drama. As a talking head, it is important to know when it is awake. When requested to speak, the CordieBot LED eyes are ramped up in a soft blue. Then an LED in the interior (brain) and on the top (head) are ramped through a series of random colors. When the speech is done, the eyes are ramped down to dark. A class is defined to handle the light show and ramping the eyes.

Controlling the CordieBot—the touch button

On the front of the CordieBot is a blue touch sensitive button which selects the function. Based on the selection, the CordieBot speaks a desired response. The selection is based on the number of touches in a short period of time.

1	Speak the time, with a possible message.
2	Announce the local weather and temperature.
3	Read the "Quote of the Day" off the Quote of the Day website.
4	Repeat the last message spoken.
5	Say its version number.
6	Check for updates.
8	Recite the origin story of the CordieBot
9	Say its internal temperature.
17	Say "You found an Easter egg, good job". This last command is not publicly documented, I am curious if it will be found.

The button has its own class to read the button and determine the number of touches. There is an allowable interval separating the touches, and when no touches have been sensed for a longer period of time the number of touches that has been sensed is returned to the calling routine.

CordieBot Communications

The CordieBot communicates with a server computer to update a message file. The components of the communication capability include and html script running in a browser on the server, a program on the Raspberry Pi, and a communication link.

Messages

The message feature of the CordieBot was the driver behind much of its other capabilities. The messages are kept in a file (proclamations.txt) on the CordieBot in json format. Three c++ classes, Message, MsgList, and MessageBank, provide the ability to update and access the messages from a resident vector. At start-up the messages are read into the vector, and the messages are re-read whenever the file is updated. The message management software receives updates from the host computer and updates the file as needed. The message management software, accessed from another computer, has the capability to:

Add messages
Change messages
Delete messages
Retrieve a message (back to the web process)
Retrieve all messages
Resequence the messages

The messages are formatted to contain the following:

ID – a unique identification number
Year, Month, and Day
Type
Content – what will actually be spoken

The message type describes how the message will be used.

Type 1 – the message has a year, month, and day on which it will be used (this can be used to announce a special event that only occurs once).

Type 2 – the message has a month and day and may be used on that day any year (this can be used for announcing a birthday or anniversary).

Type 3 – contains no date values, and may be used any time.

The messages are used randomly, and many times the time will be spoken with no message at all. The software is structured to prefer the type 1 message if one is present for that date, then a type 2 message if one is present for that day of the year, and if neither of the above a type 3 message.

PubNub

PubNub is a system of software that can be used for a chat application (plus many other uses). For individual non-commercial use it is free. It uses keys that are many characters long so it is very secure. And it has sample software for web pages and for C++. The communication from the host computer is published on a channel (in CordieBot's case, from an html web page) and received by the CordieBot which subscribes to that channel. Both the channel name and the keys must match. The CordieBot can then publish a response on that channel, and the web page, subscribing to that channel, can receive the response and display it. The PubNub keys are kept in a separate file, keys.js, which can be used by any of the programs that contain PubNub functions: the update program on the CordieBot, the html script on the host computer, and the speak program on another R-Pi.

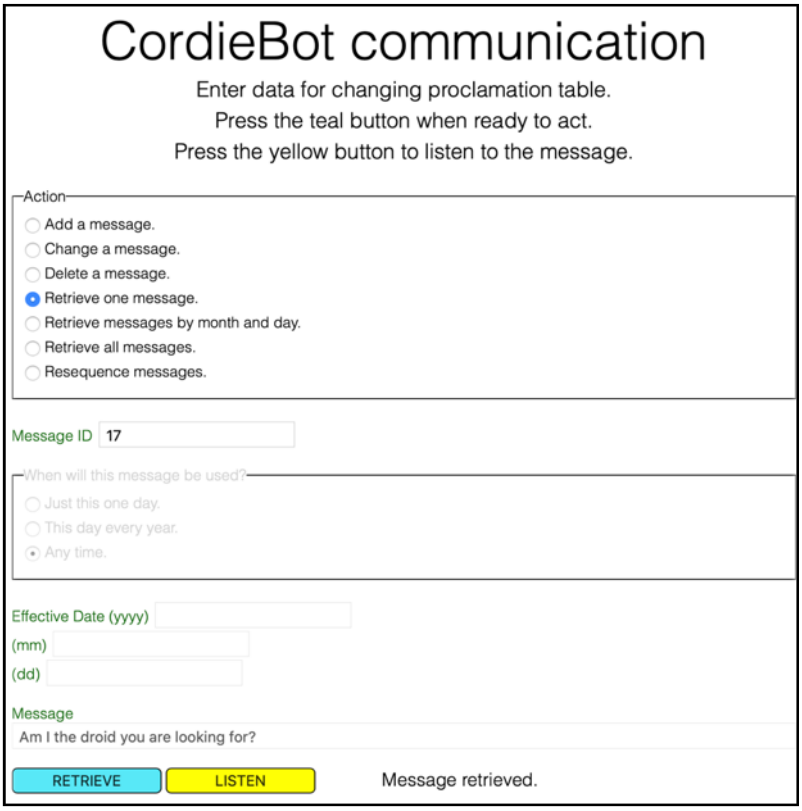

Figure 2 Screen shot of the message management page.

The host computer web page

On the host computer I wanted an application that would let me do any maintenance on the list of messages located on the CordieBot. This, with PubNub, lets me add, change, delete, and display messages remotely, with a secure connection (the PubNub keys contain 32 unique characters). With this connection, I can add from home new messages for my granddaughter to hear the next time she checks the time. I can have a local R-Pi set up to listen to the messages before they are sent. The web page (figure 2) is written in JavaScript and uses a CSS file that my daughter created.

"Headless" updates to the CordieBot software

The CordieBot application can be commanded to check for a USB memory stick, and if present will check for a wpa_conf file to add or change a WiFi connection, as well as checking for updated application software. If the software is being update, the application will move the new version to the appropriate directory and then reboot.

Construction

When I found this brass lamp in the Habitat Restore, I had just a glimmer of what it would become. The rings on the sides do look a bit like earrings, and the body something like a face, so I decided to take that idea wherever it would lead me.

The following description of how the CordieBot was constructed is not intended to be a how-to. It is unlikely that you will find a lamp exactly like the one I found. But perhaps you can get some ideas of what you can do from the way I constructed the CordieBot.

The separate parts of the lamp are held together by two 3/8" hollow threaded rods that are common to most lamps. When the parts are re-ordered, the same rod may be used to hold the new arrangement together. Since it is hollow, the wiring can be run from one segment to the next.

Note that the very top of the lamp is inverted, making a space at the top that is open. Inside this open space is an LED which I refer to as the brain lamp. At the top of the hollow tube that holds the upper part to the main part of the head is another 10mm LED which fits nicely the end of that tube. I refer to it as the head lamp.

The eyes are attached to a piece of perf board set behind the eyeholes, which are bordered by brass washer held to the face by 6 2-28x1/8" bolts giving a steampunk look to the construction.
The lips are a cutout piece of sheet brass over brass screen obscuring a hole to the interior. The speaker is behind the hole, and the lips are held to the face by 4 of the 2-56 x 1/8" bolts. An interior bracket holds the speaker in place,

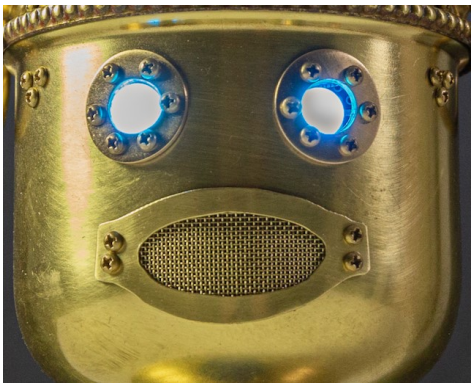

Figure 3 Face detail.

and supports a small piece of perf board which holds the eyes and the LED PWM board.

Interior Bracket

The interior bracket is the only hardware construction inside the CordieBot. It is sized to fit the space in the lamp. The bottom of the bracket has a 3/8" hole and fits over the rod connecting the base to the head. This allows the bracket to be held by the same nut that keeps the head attached to the base. The bracket has a U shaped holder to keep the speaker from sliding. The bracket is bent so that the eyes are held in place behind the eyeholes, and there is tension on the speaker to hold it in place. A small piece of foam on the back of the speaker helps keep it steady without affecting the sound. (Figures 4 and 5)

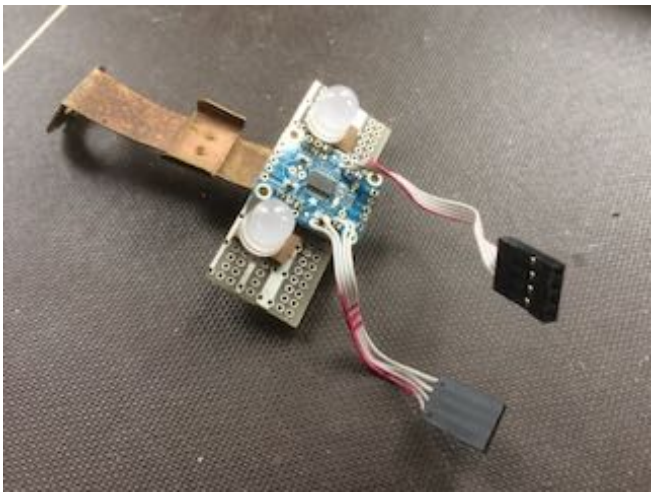

Figure 4 Lamp and speaker assembly showing eyes and PWM driver board. The flanges at the midpoint hold the speaker in place.

The wires from the eye LEDs on the perf board are woven through holes in the board and back up to the LED PWM board. A small piece of 1/8" Masonite spaces pushes the eyes out a bit to let them project through the eyeholes better. The head and brain lamps connect to connectors wired to the LED PWM board. (Figure 6)

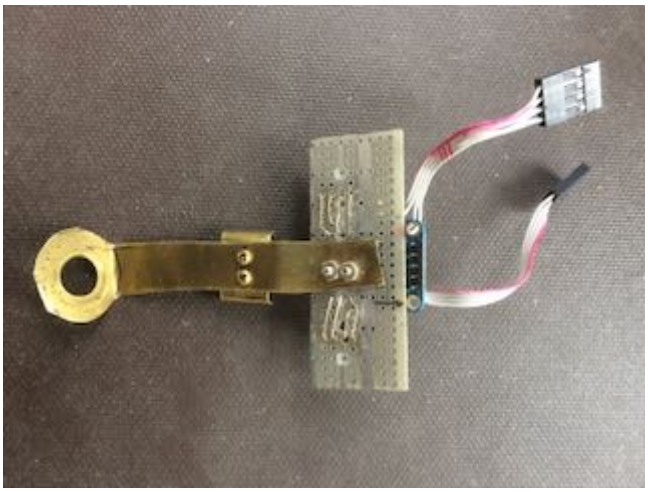

Figure 5 Rear side of the lamp and speaker assembly. Note that the perf board is connected to the bracket by two 2-56 bolts.

Figure 6 Interior of the "head" showing the installed eye bracket and wires from the base.

The Base

The base of the CordieBot holds the input button, the volume control, power input, and a USB connector that directly connects to the R-Pi.

Figure 7 Rear of the base, showing the volume control, USB port, and power connector.

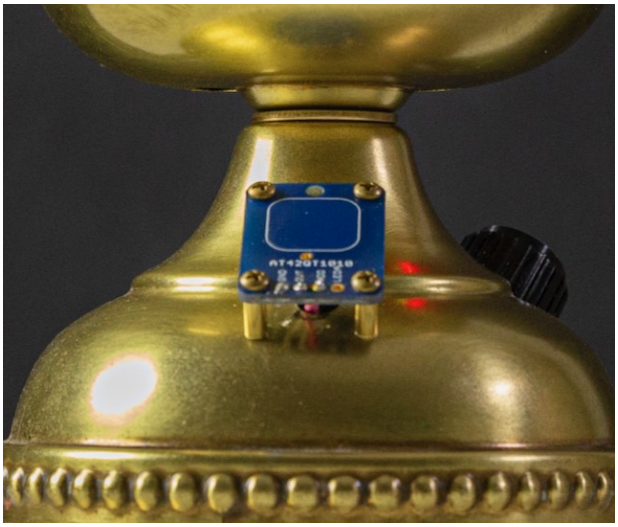

Figure 8 Front of the base, showing the touch button.

The volume, power, and USB are on the back, while the input button is on the front. The button is held in place by four 2-56 x 1/2" bolts and is held away from the base by 1/4" pieces of 1/6" brass tubing.

To make the USB cable fit into the circular form of the base, I cut the corners off of the cable. (Figure 10)

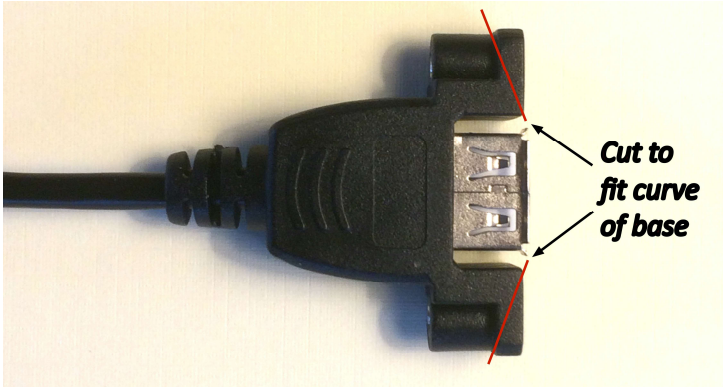

Figure 10 Cuts to end of USB cable.

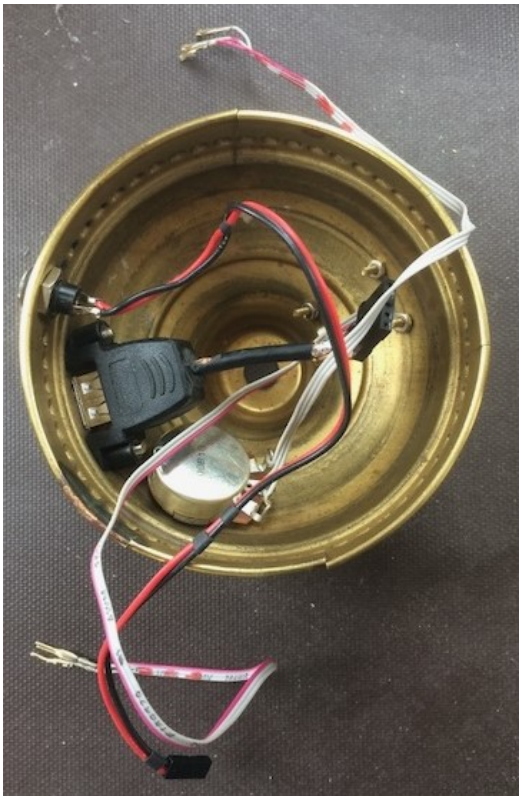

Figure 9 Inside the base.

The wires will be fed up through the 3/8" threaded tube that holds the base to the head. After the wires are fed through the tube they may be inserted into their connectors.

The Upper Head

Two parts from the original lamp make up the upper head. The very top part is inverted from the original lamp. With its open cutout design it seemed just the place to put a brain light. The two pieces are held together with a length of the standard 3/8" threaded rod. The very top light, or head light, is glued to the end of the threaded rod. The wires for this light are soldered to the LED leads and insulated with pieces of heat shrink tubing. The head lamp and brain lamp are programmed to do a light show any the CordieBot speaks.

Figure 11 Upper head detail

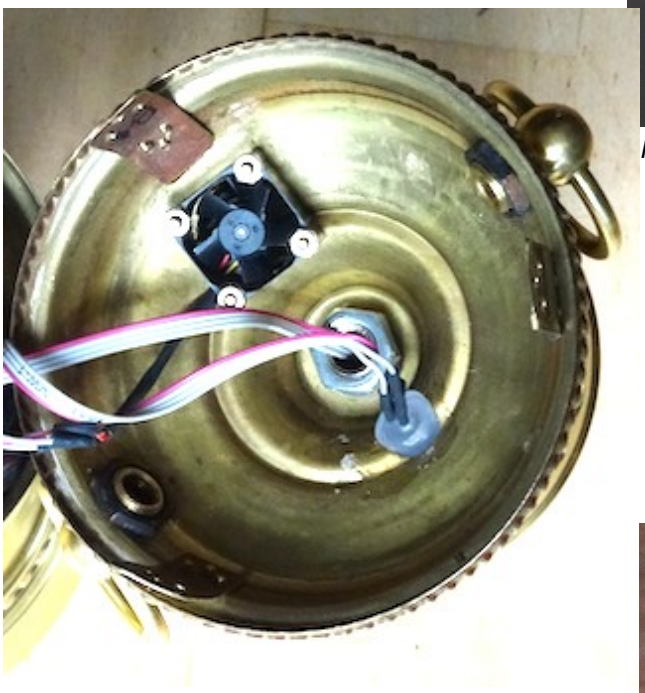

Figure 12 Interior of the top pieces.

threaded rod. Note that there are ventilation holes in the body of the head, which can be seen in figure 7.

Since the Raspberry Pi will be inside the head, the typical threaded rod structure of a lamp cannot be used to hold the parts together. Figure 13 shows the location of the bracket, which is held by six 2-56 x 1/8" screws. The bracket holes for the screws are tapped so no nuts are required. Six screws are hardly necessary, but adds to the steam punky look of the CordieBot. There are three brackets, one at each side just in front of the loops, and one in the middle in the back. The brackets may also be seen in Figure 12 (the back side is at the top left of the picture).

In addition, there is a miniature fan which is operated when the temperature gets too high in the CordieBot to cool the interior. The fan moves air from the interior through a hole in the bottom of the two pieces and is hidden by the grated portion of the top of the two. It is held in place by four 2-56 x 1/2" bolts. The brain lamp is glued into a 3/8" hole which is also inside the perimeter of the grated portion of the top. Figure 12 shows the locations of these components, as well as the wires going to the head lamp through the hollow

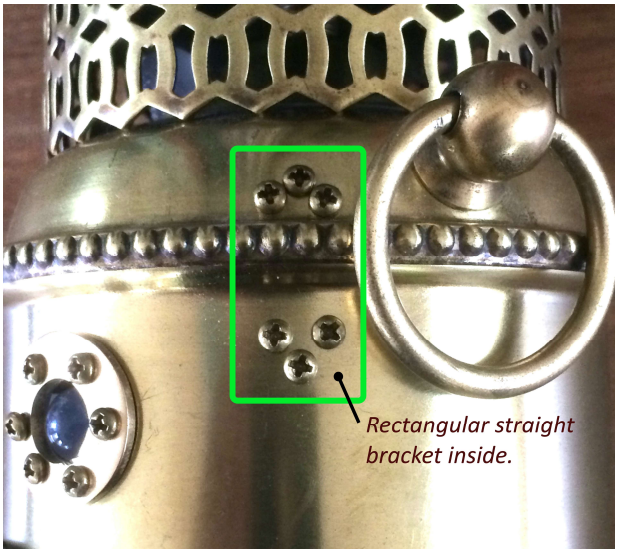

Figure 13 Bracket to hold the top on.

Electronics and the Raspberry Pi

The electronics interfacing the Raspberry Pi with the daughter boards and other components are on a custom PC board. It is similar to the Raspberry Pi HAT, but lacks the ROM to tell the R-Pi what the board is. Since this is a unique and dedicated application, I didn't feel it was necessary to have the ROM ID. I also didn't feel a need to go through the learning curve of including it in the design and altering the software to look for the ID. The populated interface board is shown in Figure 14.

Note that the power to the whole system attaches to this board, and that a connector from this board supplies power to the R-Pi. While there are 5 volt power lines on the 40 pin connector, these are not protected and it is better to supply power to the R-Pi as if it were connected to the power input jack.

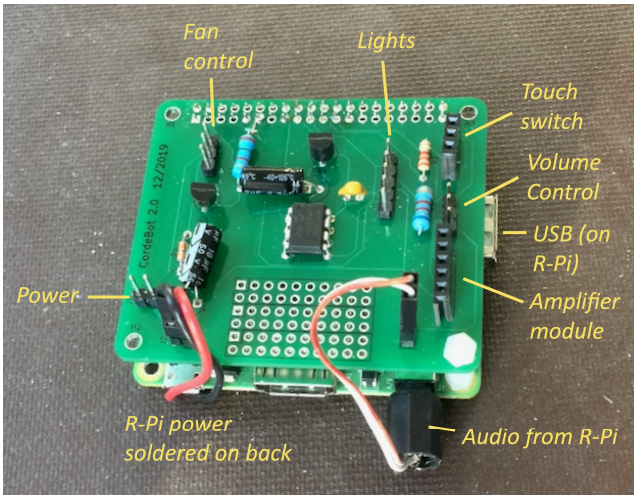

Figure 14 Raspberry Pi 3A+ and "HAT"

Figure 15 Raspberry Pi inside the head.

A complete set of schematics and a design of the PCB board is available in another document.

The Raspberry Pi is simply inserted into the head of the CordieBot. I originally planned to connect it with bolts to the back of the head, bur realized that it wasn't necessary. I did cover anything conductive on the edges of the Raspberry Pi with electrical tape to insure nothing could get shorted out if it did get close to the brass case.

All of my wiring is color coded with the wire for pin 1 marked in red.

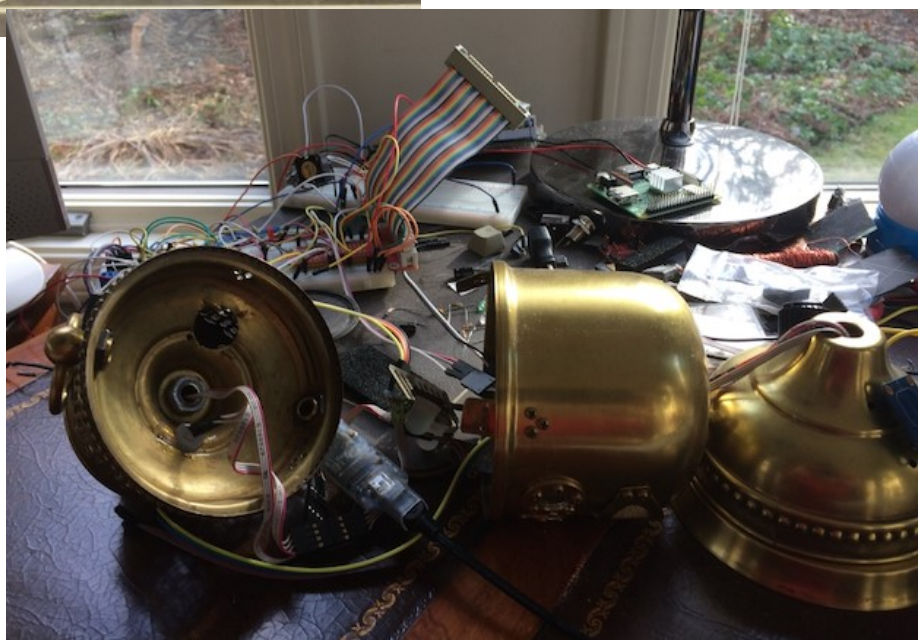

Figure 16 Under construction. Yes, the rats nest of wiring did get confusing from time to time, but it all worked out in the end.