Jeffrey J. Bradley
Teammate: Thomas W. Branson
CS370, Colorado State University
Professor Shrideep Pallickara
12/5/18

Term Project – Deliverable 3

Introduction

Large events with music usually come with flashing colorful lights: DJ lights. Companies everywhere are employing advanced methods to control these lights using remotes, smartphones, music pulses, etc. Voice control is also used for products like Amazon Alexa and smart home lighting. Our goal was to control a set of lights from a smartphone in a simple and user-friendly way. Telegram and the telepot library are what this project utilizes to receive smartphone (or any smart device) commands.

The problem we have solved is remote lighting. The problem is being solved everywhere currently for more than DJ lights. Smart Lighting is becoming a big industry through companies like Amazon and Philips. The most cutting-edge of smart lighting bulbs these companies offer include LIFX, Leviton, Lutron, Ecobee and TP-Link, all of which rely on Wi-Fi to signal the bulbs (1). Our solution was no exception, because it needs a Wi-Fi connection as well. Our problem of greatest priority was user-friendliness.

We used a tutorial to set up the connection to the Telegram messaging service. The app for it can be downloaded on any smart device, and the bot can be created and accessed from anywhere (2). We implemented various patterns and commands to control the lights through text input. Tom and I set up our own Pi's, wiring, and breadboards as per the same specifications, so the code was portable between the 2 of us. The user-friendly command parser reduces command line confusion greatly since it looks for keywords, like how an Amazon Echo listens for keywords. For example, "[Play][Jazz]" translates to 'start music in the jazz genre category' (3). This was a great baseline for the solution to user-friendliness.

Problems

Some smart home control-methods are not simple enough for any given person to use them. Amazon Alexa and Amazon Echo are great examples of very easy smart home control methods. One could argue that the language must be changed within these in order for a non-English-speaker to use an Alexa setup by an English-speaker.

Additionally, the Raspberry Pi does not have the same hardware as Amazon uses, so our code must be able to communicate with the Pi in predictable ways.

- Bilingual communication
- GPIO & python communication method

Solutions

Our solution and in turn, what we implemented throughout the project using the knowledge discussed above is as follows:

- Wireless message transfer to Pi
- Keyword command parser
- Help command
- Command compounding
- De-obfuscation of GPIO output indices
- Demo mode

Wireless communication

The telepot library enables communication between the Telegram messaging app and a python message handler. According to the documentation, one must import the library, obtain a Telegram bot API token and initialize a new 'Bot' object with the token as a parameter (5). The object is now listening to that Telegram bot's chat streams.

Keyword command parsing

The message handler searches the whole command for keywords such as "red", "green", "all", "on", and "off" in addition to patterns like "dance", "bounce", "strobe", and "demo". This enables the user to enter the command in English if they wanted to. For example, "Turn on red" is equivalent to "red on ", and "Dance then Bounce then Turn on all" is equivalent to "dance bounce all on ". This uses the methodology Amazon employs when designing their Echo's (3).

The solution to the bilingual communication problem should now be somewhat more obvious. Any smart device is very likely to have translation functionality. If non-English-speaking users can translate their commands to English before sending it through Telegram easily and quickly, then this problem is solved.

Help

We added a help command for further simplification of the bot. All the user must do is type help, and the telegram bot will reply a list of every possible command. This is very useful because the user does not have to go digging through documentation.

Command Compounding

The other perk of keyword handling is what the name entails. Commands can be compounded. The parser checks for every word regardless of if others are present, so every command found will be executed. One caveat is that commands cannot be used more than once in the same message.

De-obfuscation of GPIO output indices

WiringPi.h was our solution to assigning individual GPIO's to lights. It is a library that allows us to control individual GPIO pins from our Pi using a single line in Python. According to the wiringpi.com: "[WiringPi] is a simplified numbering scheme which provides a mapping from virtual pin numbers 0 through 16 to the real underlying Broadcom GPIO pin numbers." (4).

Demo

If a user cannot translate their commands and simply wants to see the lights without specifying anything, they can just send the command "demo" through Telegram and the pi will go into demo mode, where every light sequence is put on display.

Conclusion

Despite the relative simplicity of this program, a lot of work went into it to getting it to work as we envisioned. We even had additional goals that we did not meet in time for the deadline. Rather than implement 100+ commands, we stuck to about 10 to preserve user-friendliness, since this was the main problem we had hoped to solve. Our solution required the use of Telegram, and the telepot and wiringpi libraries, and most notably: Wi-Fi. This seems to be a standard requirement for smart lighting, so we could not find a reliable solution that did not need Wi-Fi. This program is a baseline for something much bigger. Ideas we had for future implementation were:

- Add a temperature sensor and turn on corresponding heat level light
- Add a distance sensor and display distance in binary across lights
- Add a visual display element using a 3.5' LCD screen
- Add more lights
- Add more patterns
- Parse duplicate commands in the same message

I plan to implement some if not all of these in the future, since the project is currently just a proof of concept. It is however quite portable in terms of starter code, so this would serve well as an open-source project on GitHub potentially.

Bibliography

1. Null, Christopher. "Smart Home Guide for Beginners: Make Your Home More Convenient to Live in without Spending Lots of Time or Money." TechHive, TechHive, 15 Aug. 2018, www.techhive.com/article/3297744/connected-home/smart-home-guide-for-beginners-how-to-make-your-home-more-convenient-to-live-in.html#toc-1.

2. Raj, Aswinth. "Using Telegram Bot with Raspberry Pi: Sharing Text and Files." Arduino Line Follower Robot Code and Circuit Diagram, Circuit Digest, 10 Oct. 2017, circuitdigest.com/microcontroller-projects/raspberry-pi-telegram-bot.

3. Amazon.com, Inc. “Alexa and Alexa Device FAQs.” Amazon, Amazon, www.amazon.com/gpm/help/customer/display.html?nodeId=201602230.

4. Wiring Pi, Setup, wiringpi.com/reference/setup/.

5. Nickoala. “Nickoala/Telepot.” GitHub, 26 May 2018, github.com/nickoala/telepot.