Abstract

Virtual assistants such as Amazon Alexa have become a common part of people’s everyday life. Additionally, modern hardware has made it extremely cheap to integrate connectivity into numerous smart devices. The Raspberry Pi is a good example of cheap and capable hardware that can be used to easily create small projects. However, it is not very easy for a novice to integrate such a project with voice control services such as Alexa. My project focused on Alexa Gadgets, which are Bluetooth devices that can connect to Amazon Echo devices and use them to facilitate voice control. The aim of the project was to make it as easy as possible for an inexperienced programmer to integrate the Alexa Gadget capabilities into an existing project. I began by working through existing code and documentation to enable a couple of my own projects to utilize these capabilities. In this process, I learned that I had to read lots of documentation and code before I could write the relatively simple functions that I wanted for my projects. Since existing examples proved to be too application-specific and overly complex, I created a tutorial along with a simple set of generalized functions to allow others to easily leverage the capabilities of Alexa Gadgets without needing an in-depth understanding of the whole system. The functions that I created allow users to pass information to their Raspberry Pi using their voice, as well as pass speech commands directly to their Echo from their Raspberry Pi program. To further the ease of implementation, I created a step-by-step tutorial on Instructables.com that walks through every step needed to add this functionality to an existing project.
Deliverables

1. **Library/Script/Other Helpful Software and Documentation**: The set of functions given in the code paired with the tutorial are what I would have found most helpful in adding Alexa Gadget functionality to my own projects. The tutorial gives an explanation of the architecture in "Background", and the starter code allows for a very a huge set of use cases without having to modify the cloud functions. This allows users to focus on the code they are already comfortable with on the Raspberry Pi.

2. **Hello World**: This program is essentially accomplished in step 4 when the program prints out dummy values in response to voice commands.

3. **Example Application**: This is covered in the video shown in step 5 and in the code given where values passed in from voice commands are displayed on the actual SPI display.

4. **Instructables-like Post** Click here.

5. **Discussion and Reflection**: Shown below.

Discussion

Looking back over the project, there are a few things that I might have done differently if I had known more. Amazon has a Github page where they have examples posted to help developers get started with Alexa Gadgets, and it has one primary contributor. I was using these files to help create the code that I wanted to put into my tutorial. In just this last week, a big update was made to the code that I was working from (this was the first update since its release). If I had reached out to the Amazon developer and talked to him about my project, I might have gotten some information on what he was working on releasing, as well as advice. Luckily, this update did not change much for me, but it made me glad that
my tutorial was based on starting with their live-updated code rather a modified copy.

I also think that there is room for improvement going forward. It would likely be possible to automate some of the installation parts of the tutorial using bash scripts. I was a little nervous about this since many of the steps involve installations and small steps that could fail and make it hard to go back halfway through the script. Another thing that could potentially be automated is the setup for the Lambda function. I made a video and scripted walkthrough to show how to setup the Lambda function from the web-based console. However, there is a CLI that could be smoother for this. The main problem is that this would likely take even longer for new users since it is an additional step to get the CLI set up. Nevertheless, I think there would be a way to get the dependencies for this lambda function installed without making the user install them locally and create a zip file. I thought about just providing my .zip file, but I did find that the dependencies it contains changed throughout the project, so this did not seem ideal.

One of the primary challenges that I faced with this project was getting to the point where I felt like I really understood the control flow. There was a significant chunk of time where I was reading reading documentation without absorbing much because I had so few reference points. These pages made much more sense when I reread them later in the project. I think it would have helped me a lot to have known to first try to understand custom skills and then understand gadgets that use custom skills.

Another issue I had was that initially I initially overcomplicated the code that would be necessary to provide versatile functionality in the tutorial. Amazon’s example that used a custom skill was so application specific that it made it seem like the custom skills had to be much more complicated that it really needed to be. Once I really thought this through, I realized that having very simple functions (one intent to send data, and one intent to receive data) on the cloud end of things makes it much easier to do what you want with the Raspberry Pi functions.

Overall, I was very happy with the result. It was really fun to see how easy it is to add voice control to cheap projects, and it makes excited to see what others might do
with this tutorial.