University of Colorado Boulder

Magic Wand Super Project

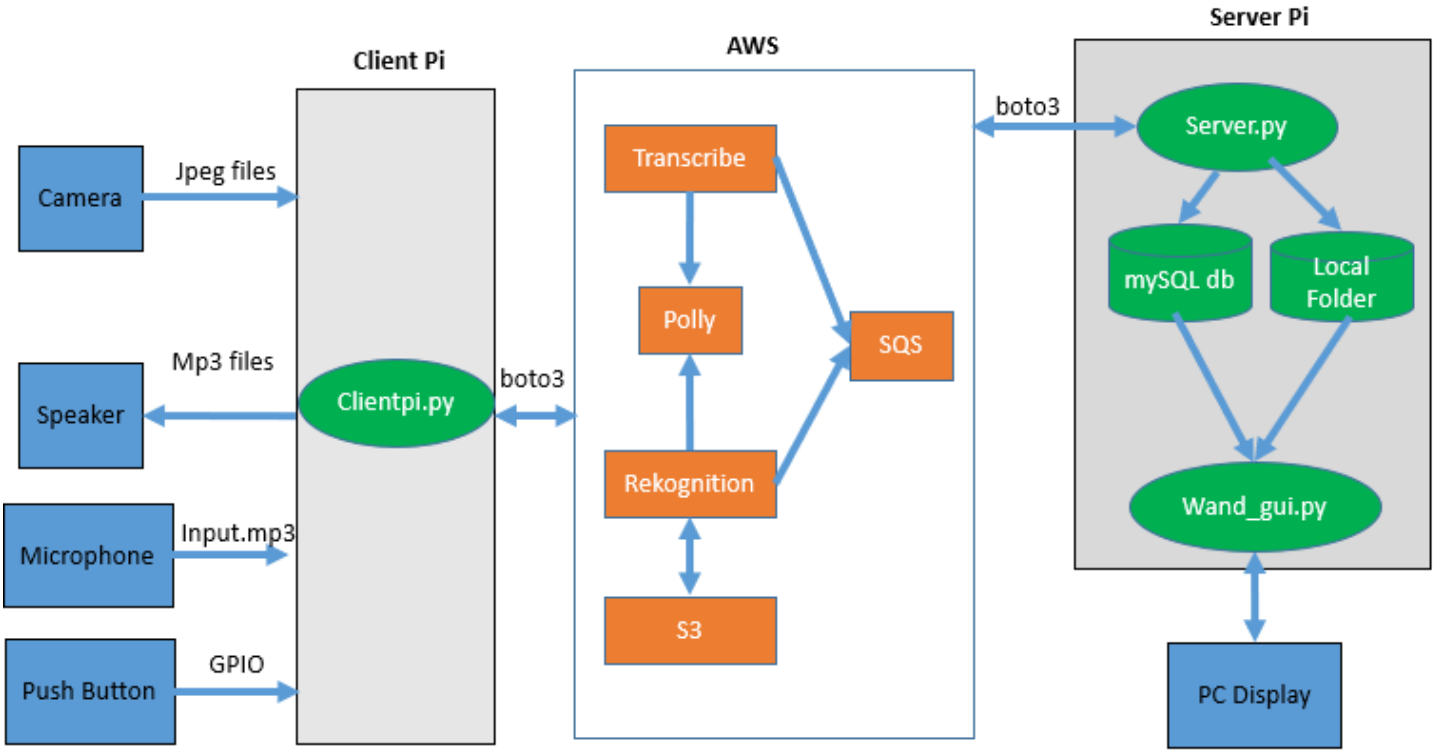
Project 6

Michael Finale
12-10-2019

Team Members:

This project was developed by Michael Finale.

Architecture Diagram



The diagram above shows the main components and their relation to other subcomponents for the Magic Wand System. Two Raspberry Pi's running seprate python scripts interface with AWS and other devices in order to operate the Magic Wand System.

The Client Raspberry Pi runs a python script called "Clientpi.py" which interfaces with AWS using the boto3 python library. This script waits for the user to press the push button that is tied to GPIO pins on the Client Rasberry pi. Once the button is pressed the script synthesizes speech using AWS polly and plays an audio file to ask the user for a voice command. Upon reciving the voice command, the recorded audio file is sent to AWS Transcribe where it is processed and read back in text from to the Clientpi.py script. The script will again use AWS Polly along with the transcribed command to play back the command that was interpreted from the user. Upon receiving the command "Identify" from AWS Transcribe, the script will capture a photo using the installed camera within 10 seconds and will play audio to the user indicating so. The captured image is then sent to AWS S3 where it is processed by AWS Rekognition and given a label. This label is read back to the user followed by a request to answer whether the label was correct or not. The user will then either answer "correct" or "wrong" and a corresponding audio file will be played before the script returns to waiting for the user to press the push button. All recorded commands are stored in SQS as a JSON message that also contains information on whether the command was valid or not. All labels are also stored in SQS as a JSON message thar also contains information on whether the label was correct or not. Audio files that are played on the speaker are either synthesized from AWS Polly or are static audio files that do not change.

The Server Rasberry Pi is a virtual machine running Raspbian OS and has two scripts running in parallel. The first script, Server.py, also interfaces with AWS using the boto3 library and periodically retrieves and deletes messages stored in SQS from the Clientpi.py script as well as the last image stored in S3 that was used by AWS Rekognition. SQS messages are stored onto a local database and are sorted into one of two tables, depending on whether the SQS message contained information for a recorded command or information for a label. Images downloaded fro S3 are downloaded to a local folder shared by the wand_gui.py script. The wand_gui.py script presents a pyQT5 Gui on the pc Monitior that allows the user to retrieve statistics on the recorded commands and generated labels for the images taken. There is also a function to display the last image taken and corresponding label.

Project Deviation Statement

The following list contains the main deviations from the original project design.

- GPIO push button used to initiate wand.
- Image from S3 stored on local folder instead of in a mySQL db
- The portion of code that interfaces the Server Pi with AWS is implemented in Python instead of Node.js
- Error Messages and Status logs are displayed on the status bar as one item instead of two separate UI elements
- A raspberry Pi Virtual Machine was used in place of an actual Rasberry Pi for the Server Pi

Third-party Code Used Statement

Tutorials

- on how to setup AWS CLI https://ownthe.cloud/posts/configure-aws-cli-on-raspberry-pi/
- on how to setup AWS CLI and boto3: http://2017.compciv.org/guide/topics/aws/intro-to-aws-boto3.html#installing-the-aws-tools
- on how to configure a mySQL db: https://pimylifeup.com/raspberry-pi-mysql/

Borrowed Code:

- Code samples for using AWS SQS with boto3: https://docs.aws.amazon.com/code-samples/latest/catalog/python-sqs-send_message.py.html

Libraries:

- Pygame library: http://www.pygame.org/news.html
- Picamera: https://picamera.readthedocs.io/en/release-1.13/
- urllib : https://docs.python.org/3/library/urllib.html
- RPi.GPIO : https://pypi.org/project/RPi.GPIO/
- boto3: https://aws.amazon.com/sdk-for-python/
- pydub: https://github.com/jiaaro/pydub
- mysql.connector : https://dev.mysql.com/doc/connector-python/en/
- PyQt5: https://pypi.org/project/PyQt5/

Other:

- Stock Audio file source: http://www.orangefreesounds.com/

Project Observations Statement

The following observations were made during the development of this project:

- Installation of node.js was simple to execute however the node package manager refused to cooperate no matter what version was installed. This prevented me from installing the AWS-sdk for node.js and therefore forced me to abandon Node.js all together. It was for this reason I used python for the Server.py script.
- Initially I intended to have the Client pi poll for an audio command before beginning it execution, however it was much more simpler to integrate a push button. I assumed this would be a more rational approach if the final product of the Magic wand would be a device that the user would hold in their hand during operation.
- AWS Rekognition proved to be much easier to use than originally anticipated.
- AWS Transcribe proved to be the most difficult AWS service to integrate into the Clientpi.py script, it also is the most time demanding service as I would sometimes have to wait for 5 minutes to process a short 3 second audio file.