Archer:

- Created the automatic setup system (turns a base PuppyPi into a cloud PuppyPi)
- Created network interoperability for the PuppyPi (connect to WPA Enterprise servers AND WPA Personal servers)
- Created automatic git synchronization system so that all changes on the PuppyPi are linked to the GitHub
- Worked with Danny to make the majority of documentation (Read The Docs) for the project
- Created the voice data compression system so that the information we send to the cloud is not huge – decreased payload interaction with external networks
- Created architecture-wide error handling within AWS
- Created the silence recognition system – recognizes when someone has stopped speaking instead of a time-based system
- Extracted the MP3 control system from the Docker and made it compatible with the "external" Raspberry Pi system
- Created the WebSocket connection architecture that allows us to send commands to the PuppyPi from outside the ROS system
- Worked with Danny on creating functionality for complex (chained) commands.
- Interlinked the three working systems of the PuppyPi (cloud, controller, and ROS docker).
- Created the entire cloud architecture
  - PuppyPi → API Gateway → Lambda → OpenAI Whisper → Lambda → GPT → Lambda → PuppyPi
  - Used REST APIs and Lambda functions to manage our work
  - Includes transcription service and command interpretation service (both using OpenAI models) as well as the pipeline linking them
- Handled cybersecurity concerns for both the cloud and the PuppyPi (hiding credentials within environment files, securing the API with an API key, etc…)

Alicia:

- Retrieved an APK file of the PuppyPi controller app and used JadX to extract the source code
- Analyzed the app source code to learn what logic was done on the PuppyPi vs the app and what information was sent to the PuppyPi and how
- Used Wireshark and other packet capture programs to investigate all of the messages being sent to the PuppyPi
- Found which ROS messages that the app sent to the PuppyPi for its activity modes were required for the activity mode functionality
- Created color detection functionality
- Helped implement april tag detection functionality

Danny:

- Created and continually worked on AI prompt to grab specific commands based on input text from audio transcription
- Created wake word program to listen to user audio when a specific word (“PuppyPi”, “Scuffy”) is said
- Edited user voice input to max out at 10 seconds to help with processing and background noise
- Implemented the ability to say multiple commands in a single phrase with a queue connected via threads to the websocket
- Debugged Lambda function to fix incorrect outputs or miscellaneous error commands being sent to the puppypi
- Cleaned and reconfigured controller.py (main function that implements features) to make it more readable
- Extensive debugging on controller.py
- Created stop feature allowing new commands to interrupt and clear the queue for running commands specifically for continual commands such as walk and line following
- Created a variable time for commands where the user can suggest a number of seconds for the command to run, i.e. walk for 5 seconds
- Contributed to Read the Docs mainly with outlining the features of the project

Eli:

- Connected ROS and cloud services
- Configured microphone and sending of audio file to the cloud from the puppypi
- Configured speakers to play barks and other communication
- Implemented timing functionality so commands could last for a set duration
- Implemented walking and turning commands
- Created the payload dictionary to convert JSON from cloud service into an executable set of websocket commands
- Implemented a command interrupt system such that PuppyPi stops what it is doing and restarts communication with cloud services upon wake word

Olivia:

- Created & solely coded GitHub website for team, kept it updated throughout the semester
- Created all Sprint Planning Documents, worked with Archer to make Sprint Backlogs
- Kept GitHub issues/backlog up to date
- Created all presentations
- Created & linked ReadtheDocs and helped write documentation
- Figured out how to edit and run Python/ROS programs in VNC using docker cp to move files
- Found ROS commands to get the PuppyPi to perform action groups (sit, moonwalk, stand up, etc)
- Worked with Eli on getting PuppyPi to perform action group commands using websocket

- Figured out how to get PuppyPi to walk using ROS commands and websocket commands, and altered the controller to handle walking
- Worked with Eli on turning right & left
- Worked with Eli on “waiting” between websocket payloads
- Created face detection, line following, apriltag detection, and lidar functionality in websocket