Gro

A Voice Controlled Smart Urban Gardening System

Submitted by:

Nikhil Raghavendra
1617629
DCPE/FT/3B/21

Thanks to Mrs. Chan-Tan Yuek Wee for her guidance in class and allowing me to develop an application for the Android Operating System.

Contents

About

Gro is an intuitive voice-controlled smart gardening system targeting the Android Operating System. It aims to ease the hassle of daily continued management of urban gardens in cities using an Internet-of-Things (IoT) enabled irrigation system and the Gro Android application. Users can install the Gro Android app on their Android smartphones and remotely manage and monitor their garden using simple voice commands. The application also allows the users to set schedules which will trigger the IoT enabled irrigation system to automatically water the plants on a daily basis at a user-defined time.

Gro's user interface is minimalistic and easy to use. Using the application is as simple as talking to someone because users don't have to go through the trouble of learning to navigate the user interface.

The Android application and the IoT firmware were developed using the Android Studio IDE, the Arduino IDE and the object-oriented principles taught in this module.

User Guide

Voice commands issued by the user will be displayed here

Tap on the mic to speak and issue voice commands

To issue voice commands, tap on the mic icon and start speaking. Internet connectivity is necessary in order for speech recognition and speech synthesis to function properly. The Text-To-Speech (TTS) engine has to be set to Google Text-to-speech engine in order for speech synthesis to work. This can be done using the device's Settings app.

Voice Commands

To water the plants forthwith:

> *"Water my plants now"*

To water the plants at a scheduled time:

> *"Water my plants at 4:55 p.m."*

To cancel the schedule:

> *"Cancel my schedules"*

To get the ambient temperature reading:

> *"What is the current temperature?"*

To get the ambient humidity reading:

> "What is the humidity now?

Voice commands are processed by a regular expression (regex) engine that searches for keyword patterns and performs actions accordingly. Thus grammar and word ordering are not important. This makes the application very intuitive because users don't have to remember application specific voice commands.

Gro is smart enough to alert users when rain is forecast and they wish to set a schedule for the day. This feature will check if rain is forecast within the next 24 hours and if users set a schedule within the timeframe, Gro will inform the user that it might be unnecessary to water the plants today because rain is forecast within the next 24 hours.

Project Structure

The MainActivity class interfaces with the RegexEngine class, text-to-speech, and speech-to-text APIs. All voice commands issued by the user are converted to text with the help of the text-to-speech API and passed to the RegexEngine's coreUttProcessor method. The RegexEngine class contains several methods.

The RegexEngine class' constructor calls the OpenWeatherMap API and caches current weather information and forecast information using global variables that are accessible throughout the class. The wordInUtt method checks if a specific keyword pattern can be found in the voice command and returns true if found. The getTime method uses regular expressions to get time strings from voice commands and return the specified time as a string. The setScheduleAtTime method is used to set a schedule at a specified time. The getWeatherInfo method returns the current temperature and humidity readings given keys such as "temp" and "hum" as parameters. The checkForecastForRain method checks if rain is forecast for the day and returns true if forecast.

The Notify class contains the createNotificationChannel method that can be used to create a notification channel for our application, through which we notify our users of completed schedules.

The ScheduleManager class contains two methods, setSchedule and cancelSchedule which set and cancel schedules respectively.

The ScheduleReceiver class is called only when the set time is up. The class' onReceive method is triggered which calls the IOTHelper class' waterPlants method and displays a notification using the displayNotification method.

The IOTHelper class contains the waterPlants method which triggers the Blynk HTTP endpoint, triggering the ESP8266. This lights up the built-in LED of the ESP8266 for five seconds before turning off, simulating an IoT chain of events.