HKIoTDemo Documentation

Release 1.0

Eric Tran, Tyler Freckmann

October 12, 2016

Contents

Welcome to our demo on how the Harman Wireless HD Audio System can be integrated into the Internet of Things. The Wireless HD Audio SDK allows one to develop applications that connect HD Wireless speakers to other devices in the home.

Note: If you would like to check out our source code for this project, click here!

CHAPTER 1

Video of the Demo

Ever tired of reading plain old text? You probably would much rather see with your own eyes what we're doing, so click here for a video showcasing our project as a whole!

CHAPTER 2

About the project

Using Harman SDK, we were able to connect their speakers with other APIs and platforms. We were able to see how wifi enabled speakers can have an impact in a connected lifestyle environment.

CHAPTER 3

Challenges we ran into

Brainstorming and figuring the different possible APIs we could use in this demo was a tough process. The amount of public APIs available to use is outstanding, and filtering out the choices for the purpose of our demos was difficult task. Other challenges we ran into include figuring out what the best platform as a service would be, and that eventually led us to using Facebook's Parse.

Throughout the design phase of this demo, we had issues on how to structure the entire architecture. We wanted an easy to read setup for developers (in terms of code), as well as an easy to use setup for consumers (in terms of usability).

A main challenge we have noted and hope to make more efficient is how authentication was handled in this demo. On the first run, the user had to authenticate with SmartThings and Parse multiple times in order to have the feature of "turning on the light" as he/she wakes up. Having multiple authentication screens lessens the user experience. And additionally, all the third party sensors required a Parse login to connect to the platform and our main "Hub". Our goal in the future is to minimalize the amount of authentication screens used.

CHAPTER 4

Architecture Overview

4.1 Architecture Overview

The entities involved in this demo are:

HK Rules Application iOS application that functions as a central hub for interactingg with the Harman speakers.

Parse Cloud Backend architecture for IoT functionality, and interfaces with third parties.

Shower Sensor iOS application representing a sensor in the home (specifically detecting the lengths of showers).

Speech Sensor iOS application representing a voice recognition sensor.

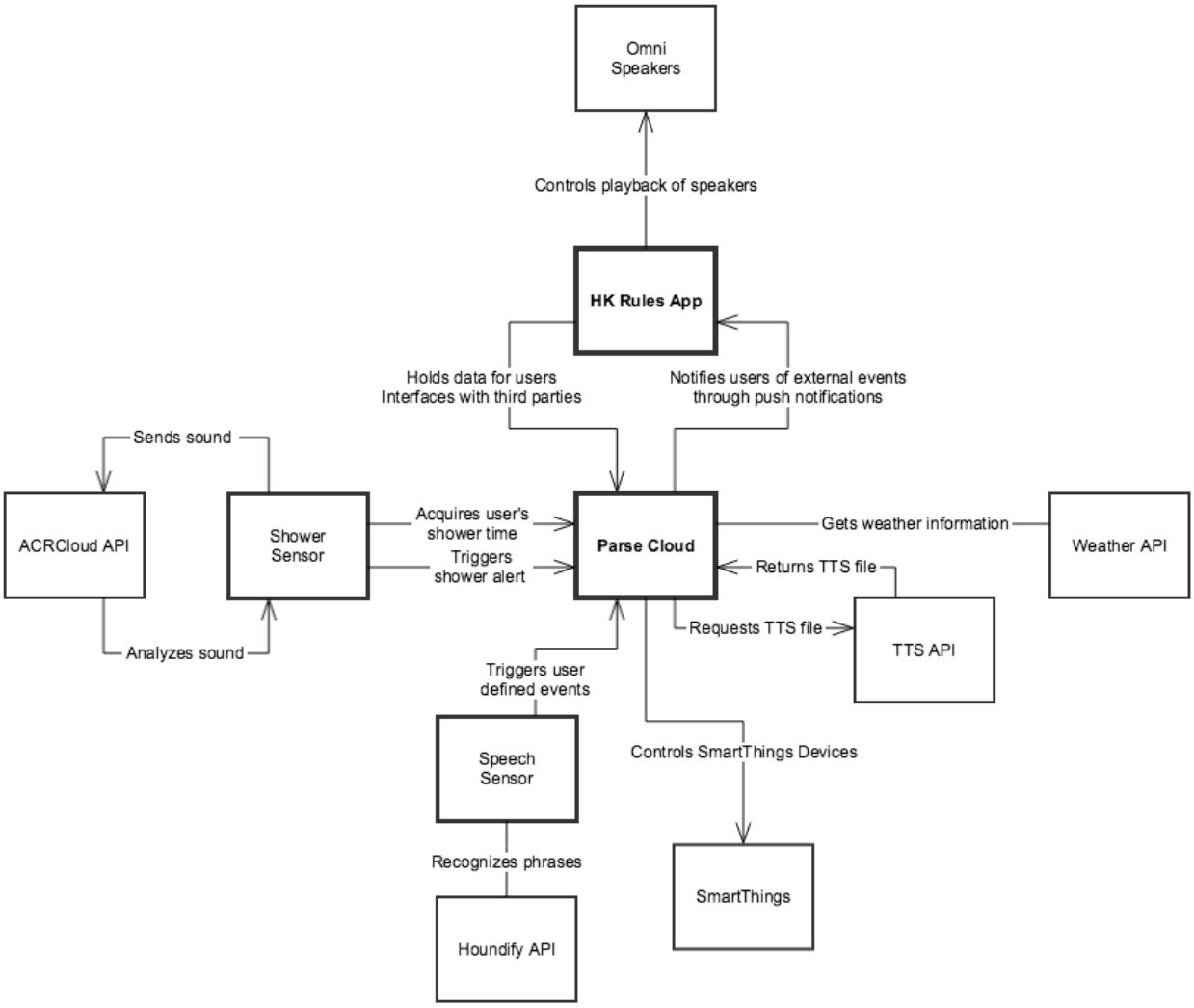
SmartThings Third party IoT devices such as contact sensors, temperature sensors, etc. (SmartThings)

Each of the things we used in this demo served a purpose. We tried to incorporate as many IoT platforms and devices as we can, but they're are just so much!

4.1.1 Context Diagram

Below is a diagram displaying how each of how entities are connected to each other.

Is the image above too hard to read for you? Click Here!


Omni Speakers
Controls playback of speakers
HK Rules App
Holds data for users
Interfaces with third parties
Notifies users of external events
through push notifications
Sends sound
ACRCloud API
Shower Sensor
Acquires user's shower time
Triggers shower alert
Parse Cloud
Gets weather information
Weather API
Returns TTS file
Requests TTS file
TTS API
Analyzes sound
Triggers user defined events
Speech Sensor
Controls SmartThings Devices
Recognizes phrases
SmartThings
Houndify API

CHAPTER 5

Overview of Classes

5.1 Overview of Classes

The three main scenarios that demonstrate Harman's integration into the IoT space are:

- Wake Up
- Take a Shower
- Leave the Home

The classes we designed and created all pertain to one or more of these scenarios. We hope it's clear how each class is connected entirely to this project, but if it isn't, please definitely continue reading!

5.1.1 HK Rules Application

The HK Rules iOS application allows the user to configure their preferences regarding each scenario. These preferences are stored in the Parse Cloud associated with a "User" object that represents the actual person using the HK Rules app. The user configures their "Wake Up" preferences in a "WakeConfig" object and their "Take a Shower" preferences in a "ShowerConfig" object.

HK Rules also controls the Harman speakers through the HKWControlHandler, from the HK Wireless SDK. Whenever we needed the speakers to play something, whether it was a Text-To-Speech weather update or a song as an alarm, the HK Rules app handles that.

5.1.2 Parse Platform

Parse notifies the user of events occuring in the home through push notifications. These notifications are sent to the HK Rules app, which handles them in the AppDelegate. The AppDelegate then activates the speakers accordingly.

The Parse Cloud also allows the HK Rules system to integrate with other services, such as weather updates, text-to-speech, and other IoT platforms like SmartThings. The way that Parse interacts with these third parties is through "Cloud Code".

5.1.3 External Sensors

Parse is notified of events occuring in the home through external sensors. The sensors in the demo are the "Shower Sensor" and the "Speech Sensor".

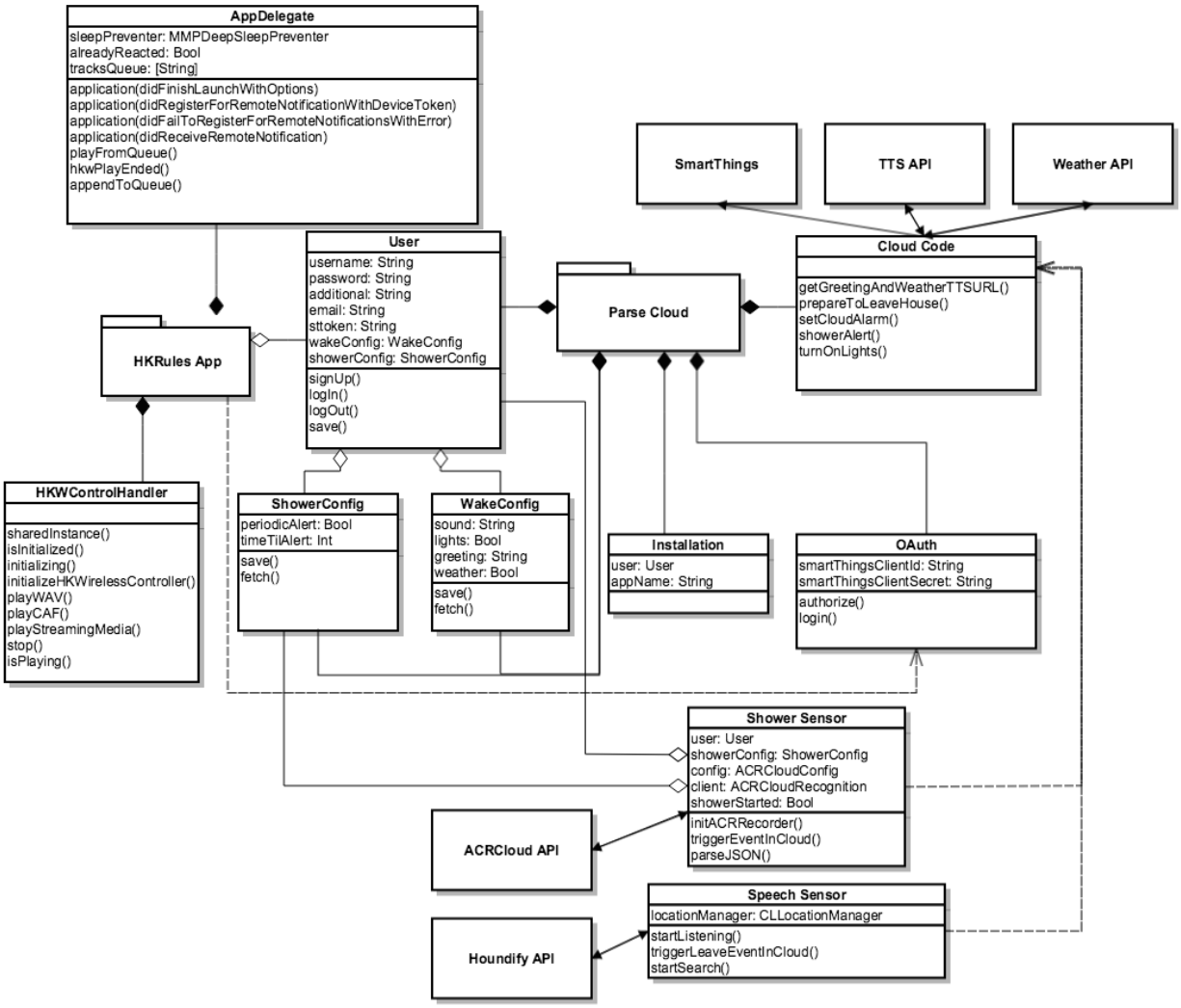
Since we didn't have an actual "Shower Sensor", we wrote an iOS application that emulated one. We spent days on different approaches, from checking FFT plots, to trying to detect ambient white noise in the background. We settled on using a sound fingerprinting platform as a basis for the application.

How it works is the "Shower Sensor" detects when a shower is running by using the ACRCloud API to capture audio from the environment and check if it is representative of a shower. For audio comparsion, we used an mp3 file of a shower downloaded directly from youtube. If the sound from the microphone captured is equal to the mp3 file, we knew it was a shower and could start the timer.

The "Speech Sensor" is a voice recognition iOS application that we wrote ourselves as well. It uses the Houndify API to recognize speech and convert it to usable data. Our sensor looked for specific phrases that we hardcoded in the application; specifically "I'm leaving", "I am leaving", or "I'm leaving now". Once it recognizes one of these phrases, the application would trigger an event in the cloud automatically. Something cool that we did not have time to implement, but would have loved to have was something like "Smart Phrase Learning", where if you say anything along the lines of depature of the home, it would trigger the event.

5.1.4 Class Diagram

Below is an diagram of all the different classes we designed for the demo:



Have trouble reading the diagram? Click Here!

CHAPTER 6

Wake Up Scenario

6.1 Wake Up Scenario

The "Wake Up" scenario is the first scenario of our IoT demo. It is a simulation of some of the neat things we can do with Harman speakers as a user wakes up in the morning.

The 3rd party things we used for this scenario are:

SmartThings for turning on the lights automatically in the morning, making it easier to wake up.

Weather API for weather forecast as you wake up, so you know how to dress yourself out the door.

Text-To-Speech (TTS) API for a custom greeting or reminder for youself, such as "You have a dentist appointment today!"

We will be leading you through the Wake Up Scenario in the best way that we can. The numbers correspond to the numbering on the sequence diagram below.

6.1.1 Sequence Diagram

Is the image above too hard for you to read? Click Here!

6.1.2 Initial Setup

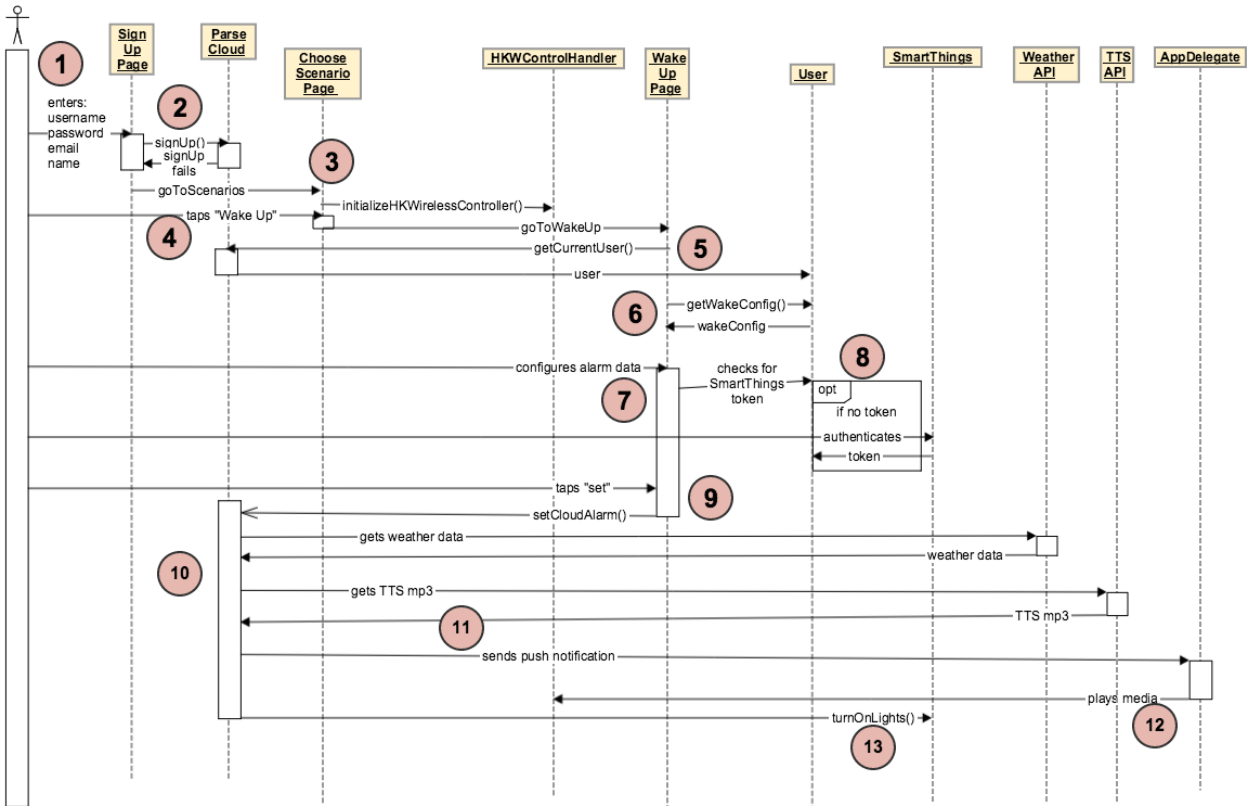
Here, we have to set up the initial settings. We sign up once so we have access to all the features, afterwards, you never have to login again, unless of course, you logged out for some reason.

1. The user starts the HK Rules iOS app and enters their username, password, email, and name into the "Sign Up" page. The information is used to associate the user with Parse for configuration purposes.
2. The "Sign Up" page signs the user up in the Parse Cloud to create their "HK Rules account", which creates a "User" object representing that user. If the sign-up fails, the user is redirected back to the "Sign Up" page.
3. If the sign-up is successful, the user is directed to the "Choose Scenario" page, which initializes the HKWControlHandler object, which controls audio playback of the speakers.

6.1.3 Configuring For Wake Up

Here are the steps that lead to choosing all the different settings for the wake up scenario as mentioned before.

4. On the "Choose Scenario" page, the user taps "Wake Up", which brings them to the "Wake Up" page.



5. The "Wake Up" page requests the currentUser from Parse.
6. The "Wake Up" page queries the currentUser for the WakeConfig object.
7. The user then configures the wakeConfig alarm data.
8. If the user chooses the "Turn on lights" option for their alarm, the "Wake Up" page checks to see if the current user has a SmartThings token. If it doesn't, then the user is redirected to SmartThings where they can authenticate their SmartThings account and gain a token for future control of their SmartThings devices.
9. Once the user has configured all their alarm settings, he or she taps "Set", which will trigger the "setCloudAlarm()" function on the Parse Cloud.

Note: If you need a SmartThings token, you will have to go through multiple authentication pages, but rest assured, you will only have to do this once as well! We have been trying to find a more user friendly way of handling this authentication process, but bear with us in the meantime.

6.1.4 Now We Wait...

After you've "set" the alarm, the wait begins. Everything is done behind the scenes from the user perspective.

10. During the "setCloudAlarm()" function, the Parse Cloud gets weather and TTS data from external APIs to send back to the user during the alarm.
11. At the designated alarm time, Parse sends a push notification to AppDelegate running in the HK Rules app on the user's iOS device that includes all the configuration data concerning the alarm (alarm sound, weather/tts data, etc.).

12. When the AppDelegate receives the push notification, it tells the HKWControlHandler to play the alarm media through the Harman speakers.
13. When Parse sends the push notification, it also tells the SmartThings platform to turn on the user's lights (using the User's SmartThings authentication token from step 8).

And voila! The wake up scenario is done. Wasn't that cool?!

CHAPTER 7

Shower Scenario

7.1 Shower Scenario

The "Shower" scenario is the second scenario of our IoT demo. It is based around the idea of water conservation.

As mentioned earlier in this documentation, we didn't have an actual "Shower Sensor" device, so we wrote an iOS application that emulated one. We went through many different approaches, from checking FFT plots, to trying to detect ambient white noise in the background. But ultimately, we settled on using a sound fingerprinting platform as a basis for the application.

The 3rd party things we used for this scenario are:

ACRCloud API for sound fingerprinting. Used to differentiate when a shower is running.

Text-To-Speech (TTS) API for converting an alert text to speech to play back through the speaker.

We will be leading you through the Shower Scenario in the best way that we can. The numbers correspond to the numbering on the sequence diagram below.

7.1.1 Sequence Diagram

Is the diagram too small for you to read? Click Here!

7.1.2 Initial Setup

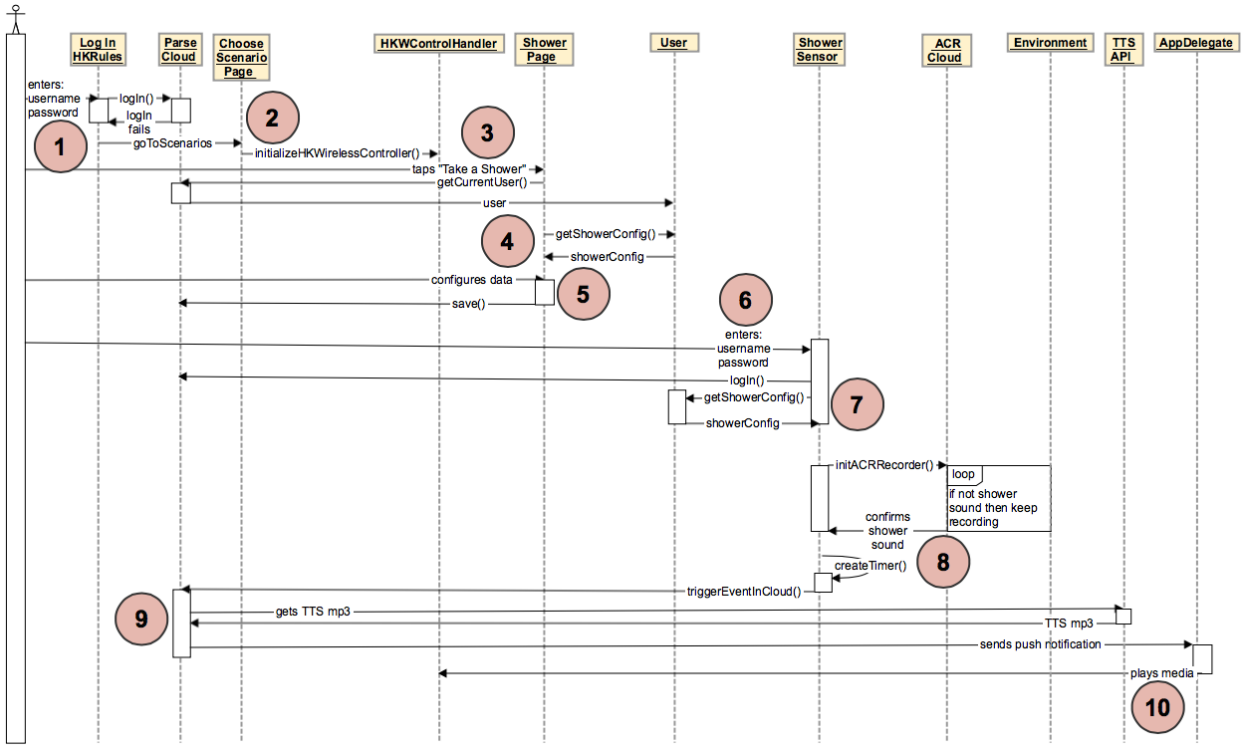
Here, we only have to login to the same user we created in the "Wake Up" scenario. This is the user in which the "Shower Sensor" application will pull the preferred shower time from.

1. The user enters their username and password into the "Log In" page in the HK Rules iOS app, which logs the user in on the Parse side.
2. The user is then directed to the "Choose Scenarios" page which initializes the HKWControlHandler object.
3. The user then taps "Take a Shower" and is directed to the "Shower" page.

7.1.3 Configuring For Shower

The following steps are used for configuring the shower preferences of an individual through the HK Rules application.

4. The currentUser "User" object is retrieved, which returns a "ShowerConfig" object with it.



5. The user configures their preferences for the Shower scenario: how long they want to shower and whether they want periodic alerts.

7.1.4 Starting and Running the Shower Sensor

These are the steps neccesary to get the shower sensor up and running.

6. The user then logs into the Shower Sensor so that the Shower Sensor can know what the user's shower preferences are (by retrieving them from Parse), and also which user to send the Shower alert to.
7. The shower sensor then retrieves the user's shower preferences in a ShowerConfig object from Parse.
8. The shower sensor begins listening to the environment, and sends a packet of sound data to the ACR Cloud for analysis. If the data resembles a shower sound, then the ACR Cloud sends back a positive response, which activates a timer on the shower sensor. The shower sensor keeps listening to the environment and sending data to ACR Cloud for analysis for the duration of the timer. If the shower sound is still playing after the timer runs out, the shower sensor sends an event to Parse which will trigger a shower alert.
9. If Parse receives a notification from the shower sensor that the shower is running longer than the user had configured, it will get TTS data from the TTS API and send a push notification to the HK Rules App on the user's iOS device.
10. When the HK Rules AppDelegate receives the push notification from Parse, it will play an alert about the shower through the Harman speakers.

And the wraps up the Shower Scenario! With this idea refined, we can start to be more cautious with our water spendings and produce noticable changes.

CHAPTER 8

Leave Home Scenario

8.1 Leave House Scenario

The "Leave House" scenario is the last scenario of our IoT demo. When you are about to leave the house for work or maybe even for a party, it'd be nice for an update as to how your house is looking in terms of security, or whether or not you should bring an umbrella or wear shorts due to weather. That is what this scene represents. You're notified of your home security, as well as given a weather update as you leave for the door.

We also did not have a voice recognization sensor of our own, so we had to emulate that feature as well. Houndify API did exactly what we wanted, and we were very glad to have got that working.

The 3rd party things we used for this scenario are:

Houndify API for speech recognization.

Weather API for weather forecast notification.

Text-To-Speech (TTS) API for converting a text to speech to play back through the speaker.

We will be leading you through the Leave House Scenario in the best way that we can. The numbers correspond to the numbering on the sequence diagram below.

8.1.1 Sequence Diagram

Have trouble seeing the diagram? Click Here!

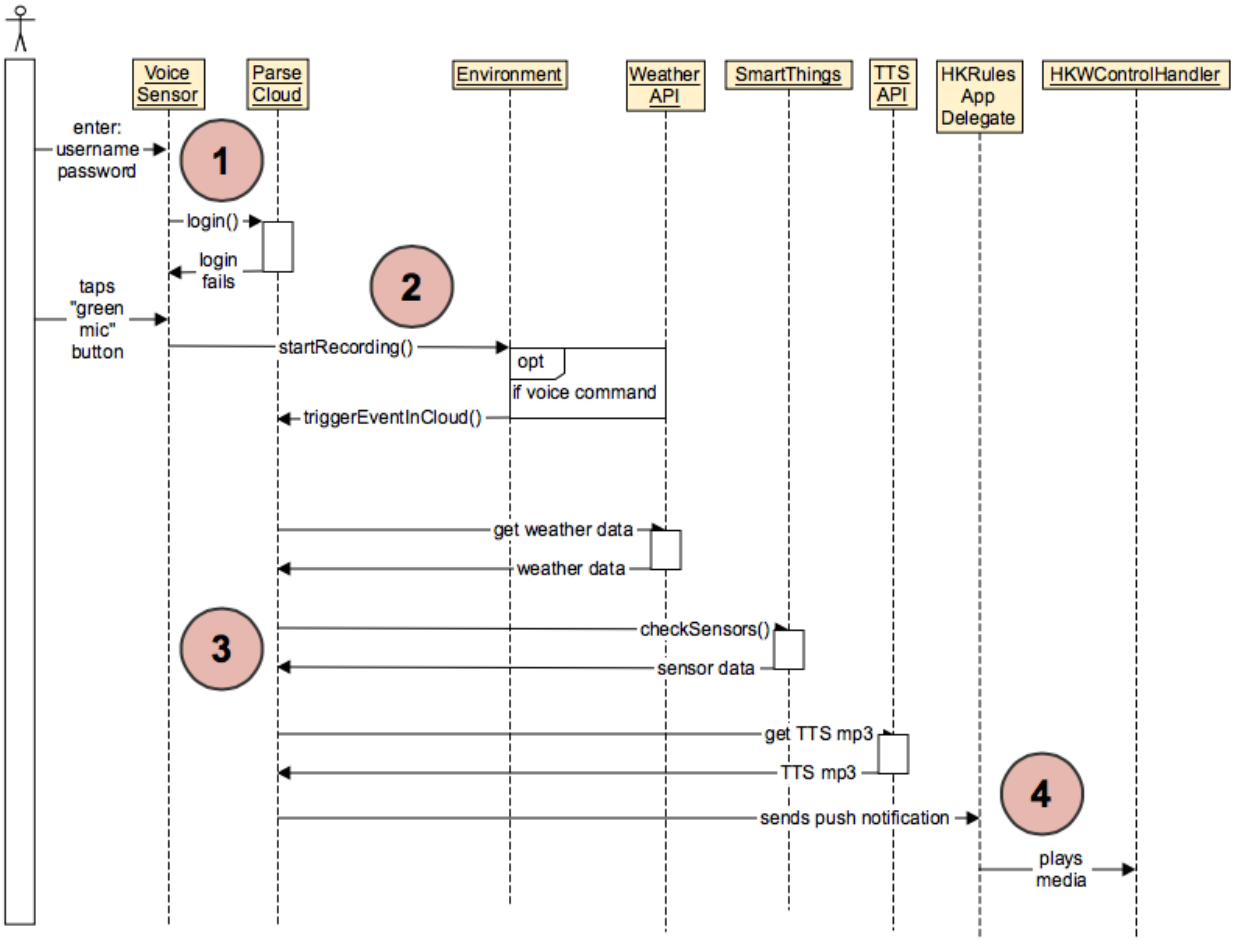
8.1.2 Initial Setup

Here, we only have to login once to the same user we created in the "Wake Up" scenario. This is the user in which the app will send the push notification to once it hears a key phrase.

1. The user enters their username and password into the Voice Sensor and logs in to Parse, so the Voice Sensor knows which user to send push notifications to.

8.1.3 Using the Voice Sensor

2. The user taps the green microphone button on the voice sensor, and if the button turns red, then the sensor is recording. If the voice sensor hears a voice command, it will trigger an event in the Parse Cloud.


Voice Sensor
Parse Cloud
Environment
Weather API
SmartThings
TTS API
HKRules App Delegate
HKWControlHandler
enter: username password
1
login()
login fails
taps "green mic" button
2
startRecording()
opt
if voice command
triggerEventInCloud()
get weather data
weather data
3
checkSensors()
sensor data
get TTS mp3
TTS mp3
4
sends push notification
plays media

3. When Parse receives a notification from the voice sensor that a voice command was given (namely "I'm leaving"), Parse collects weather data, checks the house's security sensors, and compiles that information into a TTS message which it sends in a push notification to the HKRules AppDelegate running on the user's iOS device.
4. When the HKRules AppDelegate receives the push notification, it plays the TTS message through the Harman speakers.

And there we go! That concludes Leave House Scenario, and completes our IoT Demostration! We hope you guys were able to get it up and running, and we hope we were able to help guide you through the process.

CHAPTER 9

AboutUs/References

9.1 References

We would like to say thanks and give credit to all of the different APIs and libraries we used for this demo. After finishing this demo, we felt accomplished to have been able to see how everything worked hand-in-hand with each other.

9.1.1 Resources

Harman/Kardon SDK http://developer.harman.com/

Parse Platform https://parse.com/

SmartThings http://docs.smartthings.com/en/latest/smartapp-web-services-developers-guide/index.html

Houndify https://houndify.com/

ACRCloud https://acrcloud.com/

VoiceRSS http://www.voicerss.org/

Dark Sky Forecast API https://developer.forecast.io/docs/v2

ReadTheDocs https://readthedocs.org/

9.1.2 About Us

This demonstration was designed and implemented by Tyler Freckmann and Eric Tran.

If you would like to see the source code for this project, go here!