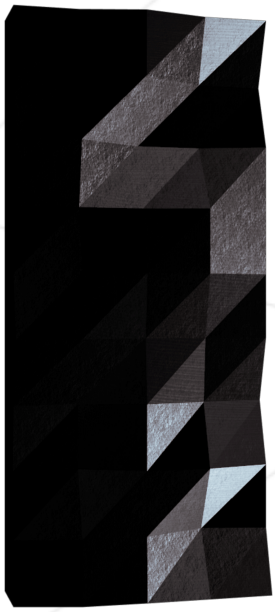


K.I.F.T. aka Knight Coders

Knight Industries Forty Two

Gaetan gaetan@42.us.org
qst0 qst0@42.us.org

Summary:
The future of voice commands and super cool robot assistants starts with you!

Contents

Chapter I

Foreword

Michael Knight: Lets play a little 'hooky'.

K.I.T.T.: 'Hooky?' I'm not familiar with that term.

Michael Knight: Then allow me to educate you in one of life's finer pleasures.
K.I.T.T.: [as speed builds up] Oh, this does feel good!

Michael Knight: [laughs, raises voice] How 'bout a little turbo boost?

K.I.T.T.: Dare we? Without sufficient reason?

Michael Knight: Without sufficient reason' is the definition of hooky! Shall we, as they say, 'go for it?

K.I.T.T.: Let's!

Although it's one of the finer pleasures in life,
Playing Hooky will be detrimental to finishing this project.

Chapter II

Goals

Knight Industries needs C programmers to prototype and implement a new `VUI` for the famous `K.I.T.T.` "Knight Industries Two Thousand". Since the previous version of the system is only compatible with Michael Knight a few things should be changed. Teach the voice recognition library `Sphinx` to recognize your voice and commands. The system should be able to respond with a computer generated voice and sound clips. For this use something like `SAM` "Software Automatic Mouth", or create something new. Log all the words and sounds the system hears, and the commands it responds to. Finally, make a server for storing and reacting to the data, and a client for connecting to the system. We encouraged the use of `web development` skills.

PocketSphinx

https://cmusphinx.github.io/

Software Automatic Mouth

Chapter III

General instructions

Create a `VUI` (Voice User Interface) using Sphinx.
You must use the `C programing language`.

Create a `server` program and a `client` program to use your system.
The client should allow a user to send `voice commands` and the server should process them.

Have the system respond with it's own `voice` and `audio clips`.
`Log` all the words and commands the system hears.
Create an interface for the user to view these commands and words.

Make the program accessible for it's users.
Teach Sphinx to better recognize your voice and commands.

Chapter IV

Mandatory part

You must use the `Sphinx` library, and you must program in `C`.
You must teach Sphinx to better recognize your voice and commands.
Your system must be able to hear your commands.
You must follow the norm, but can defend reasonable deviation.
Here are ideas for commands:

1. Set an alarm / timer.
2. Turn on / off lights.
3. Send email / SMS.
4. Check events / weather / traffic.
5. Who is connected?
6. Where is (user) connected?
7. Play music.
8. Search the web for (term).
9. Check history.

Implement these commands and some of your own.

Your system must be able to respond with it's own voice.
We suggest usings something like `SAM`
You can name your system whatever you would like.
You must create a server and client for the system.

You must use PocketSphinx

You must use some of your own data to teach your voice recognition.

Chapter V

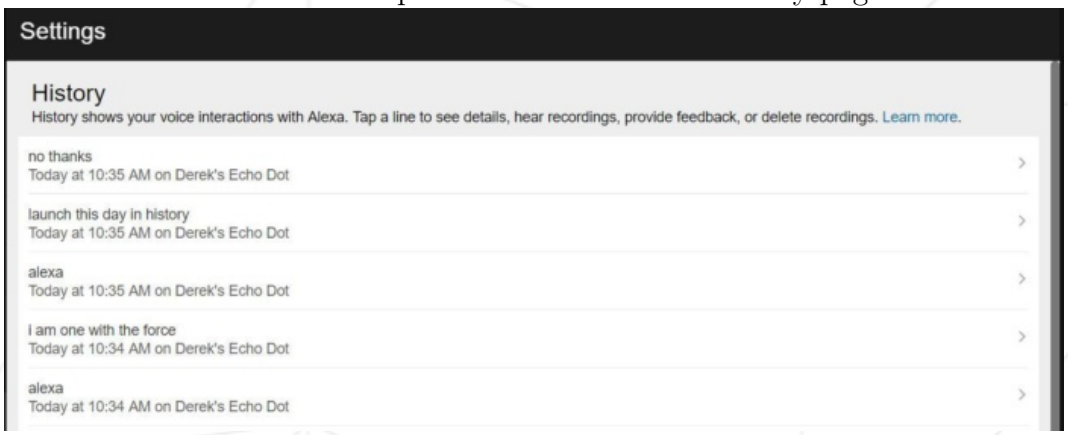
Bonus part

When you are 100% certain you have accomplished the mandatory part...

Try the roads less traveled:

Create a web client.

Here is an example of Amazon Alexa's history page:



House your system on a raspberry pi or similar micro-computer.

Make your assistant more conversational.

Copy more features from Google Home, Amazon Alexa and K.I.T.T.

Lots of easter eggs.

Biometry.

Sentients.

Chapter VI

Turn-in and peer-evaluation

Turn your work in using your `git` repository, as usual.
Only work present on your repository will be graded in defense.