# ThingKit: a context-awareness IoT programming environment

Carnegie Mellon University

Junrui (Jackie) Yang jackieyang51@gmail.com
(Mentor: Adrian de Freitas Advisor: Prof. Anind K. Dey)
Peking University - Carnegie Mellon University Summer Program

## Background and Challenge

- IoT is involved with ...
  Complexity IoT devices is highly heterogeneous, each kind of devices has its own capabilities.
  Scalability The number of IoT devices is booming.
  Duplicity IoT devices need to face the trade off of accuracy and usability and usually involved many duplicate code segments for data processing.
- However IoT middle-ware is ...
  - unique for each device in the way of communication, other programs/devices can hardly share their information.
  - being focused on the management side instead of the user side.
- Luckily ThingKit is ...
  Context-awaring Programs/devices know not only what is around but also what they can do.
  Capabilities-oriented ThingKit focuses on what the `Thing` can do instead of what is the identity of `Thing`.
  Multifunctional `Thing` can have more than traditional functions. e.g. A microphone can also sense if there is people around according to sound.

## Introduction

- ThingKit is a Python programming library to dynamically acquire IoT devices and services with context awareness.
- Everything is absracted as a `Thing`. Method is dynamically discovered and bound to the instance.
- Bluetooth is used to provide context information so that each device(program) can know the IoT context.
- Futher abstraction is achieved by using virutal devices and virtual functions.

## Solution

- Technology based on
  MQTT MQTT is an IoT chat room, where ThingKit gathers capabilities and executes functions.
  Bluetooth Using Bluetooth sensing and broadcasting unique identifier to locate the communication channel
- Features and Implementations
  Discover `Things` around
  1. Bluetooth gathers device ID
  2. ThingKit using ID to establish a connection to the device specific channel and gather capabilities.

  Gather capabilities
  1. ThingKit establishes a query about capabilities, and subscribes to all result.
  2. Other ThingKit providing one or more functionality to this device respond to the query.
  3. ThingKit capture the response and dynamically added it to the instance.
  4. ThingKit providing new functionality will broadcast their new functionality no matter whether there is a query so that other ThingKit already subscribed to this `Thing` can also get the latest profile
  5. ThingKit will re-request capabilities to delete the expired capabilities.

  Select according to capabilities
  1. ThingKit has a list of nearby devices.
  2. ThingKit looks through that list to find out if there is any device matches the type
  3. ThingKit returns a special class `ThingsList`, which shares the common capabilities in that list, so that users can easily batch call function.

## Prototype

```
import IOT
IOT.init()

def toggle():
  light = IOT.selectNearest
       ("light")
  light.setLight(!light.
       getLight())
```

**Fig. 1:** A simple application using ThingKit: Universal toggle

```
import IOT
IOT.init()

for light in IOT.selectAll("light"):
  def blink(): light.on(); light.off(); light.on();
  light._createMethod("notify", blink, {}, "notify the
       user")

for speaker in IOT.selectAll("speaker"):
  def say(): speaker.speak("You have a notification!")
  speaker._createMethod("notify", say, {}, "notify the
       user")

def notify():
  IOT.selectAll().notify()
```

**Fig. 2:** A program adds notify functionality to nearby devices, and uses it to notify the user

## Interface

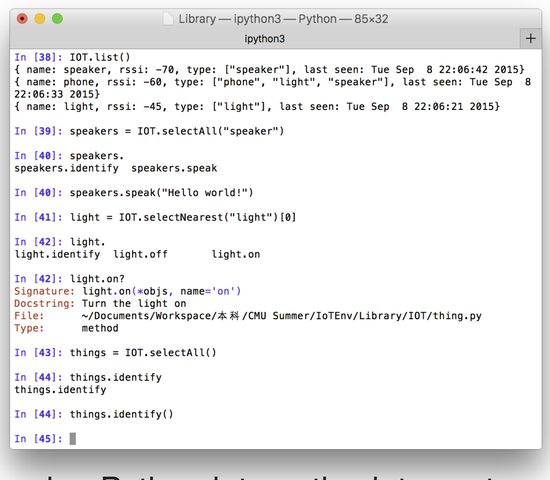


**Fig. 3:** A running Python interactive interpreter with ThingKit

## Progress

- ☑ Basic functionality
  - ☑ Discovery devices
  - ☑ Gather capabilities
  - ☑ Select devices
- ☐ Advanced functionality
  - ☑ Virtual capabilities
  - ☑ Virtual devices
  - ☐ Global select
- ☐ Portable
  - ☑ Linux
  - ☐ OS X (currently impossible)
  - ☐ Android

## Smart sign [Side project]

Background Facial recognition: a method to provide personalized service in public environment, which also requires protection from unauthorized data access.

Solution Bluewave is a framework that allows device to opportunistically share their context within a confined range. Smart sign registers the face image shared in the context within its own recognition model along with their personal data (in this case, name). Once the smart sign recognizes the person, it can use the person's data to provide personalized service.

```
{
  "name" : "Finoa Lee",
  "urls" : [
    "example.com/face1.jpg",
    "example.com/face2.jpg",
    "example.com/face3.jpg"
  ]
}
```

**Fig. 4:** A context profile used by smartsign

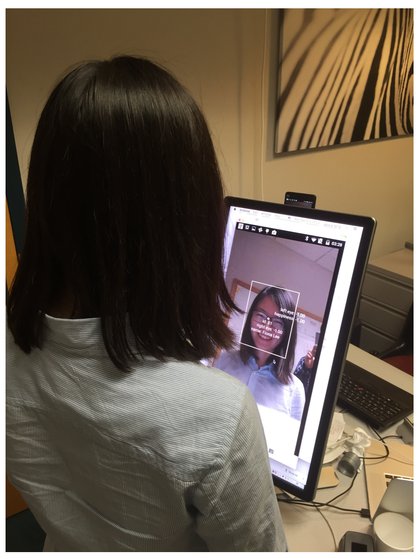

**Fig. 5:** A woman in front of a smart sign