Campus Welcome System

Zhang Zhonghan,
Lan Xijing,
Zhang Zhiyu
Instructor: Wang Jiabin

High school attached to Zhejiang University,
Hangzhou, China

May 2021

Introduction

If you are a student, new to campus, you may want to quickly understand the place of life. On campus, you may not want to be obsessed with small things, and you may also want to have a little luck in your campus life.

If you are a school official, you may want to reduce the repetitive and mechanized introduction done by humans and to promote special activities in the school.

Our intelligent *Campus Welcome System* can meet the expectations above to a certain extent.

Function Overview

Levels	Function Name and Descriptions
Session Starting	**Greeting** attract passers-by's attention by greeting and guide them to start the session.

Levels	Function Name and Descriptions
Introduction	**Campus Introduction** introduce the campus's basic information in conversations
	Navigation make the navigation in multi interaction dimension
Personal Service	**Daily Suggestion** - Take meals for example As an extension of the introduction of campus facilities, it is also a small suggestion for daily choice
	Basic User System - *Make a Friend* Establishing the user system through face recognition to lay a foundation to provide personal services
	Festival Promotion Learn about campus activities through conversation
humanistic concern	**Basic Reminder** record the daily activities into the personal reminder
	Morning Inspiration Provide an inspirational quote every morning
	Weather suggestions Remind people to take an umbrella or something else after school according to the weather tomorrow
Demonstrate Technique	**Demo time simulation** Toggle the time in the program to demonstrate the functions

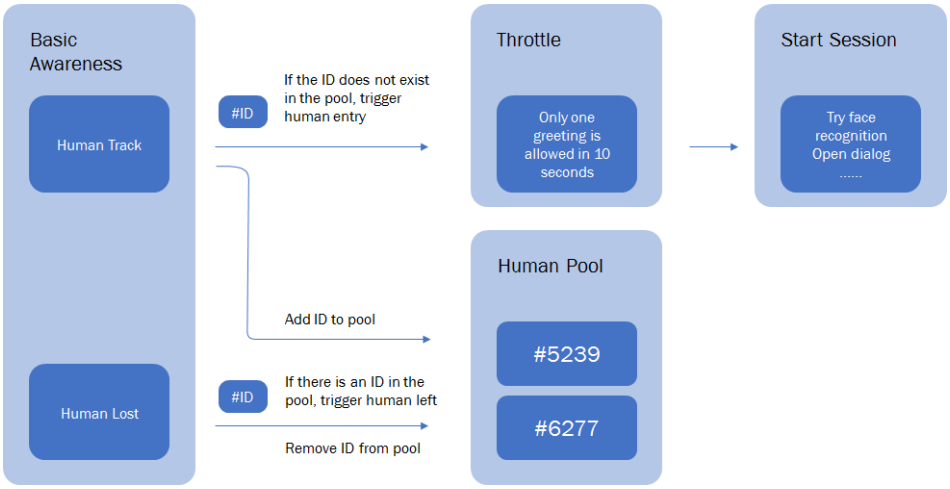
Detailed Description and implementation

1. Greeting and Session Starting

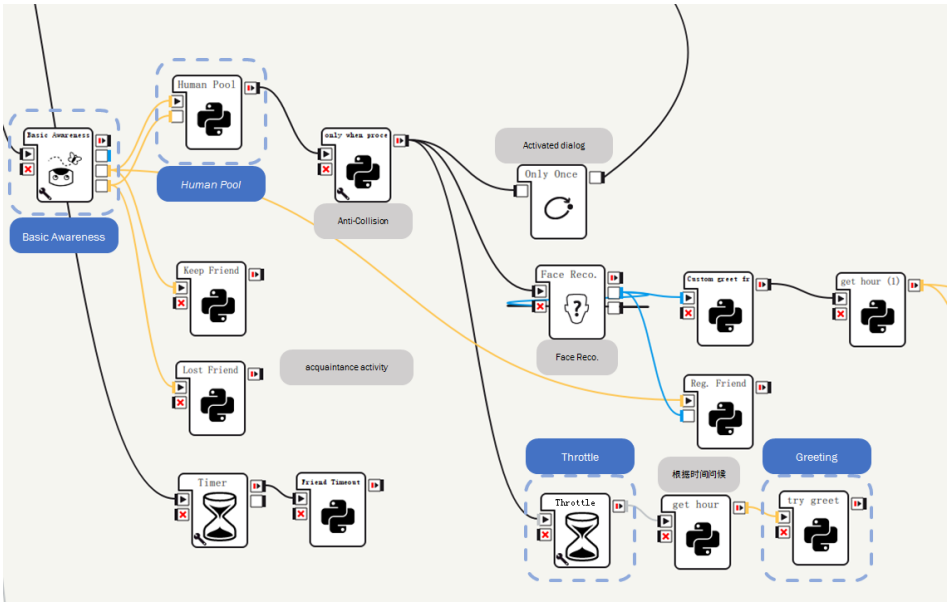
H1

The robot can detect passers-by and greeting them to guide them start session.

H2 A balance between avoiding missing the people and avoiding greeting repeatedly to the same person should be kept. We design a mechanism as follows.



Simplified Diagram



Implement in Choregraphe

H3 1.1 Human Pool Module

Create a global list of `activehumans`:

- When receiving the `ontrack` signal, if the ID (not - 1 and not in the list), add it and call `onstopped`
- When receiving the `onlost` signal, if the ID is in the list, the ID will be removed from the list

```
global activeHumans
class MyClass(GeneratedClass):
    def __init__(self):
        GeneratedClass.__init__(self)

    def onLoad(self):
        global activeHumans
```

```
        activeHumans = []
        pass

    def onUnload(self):
        pass

    def onInput_onTrack(self, p):
        if  p != -1 and activeHumans.count(p) == 0:
            activeHumans.append(p)
            self.onStopped()
        pass

    def onInput_onLost(self, p):
        if activeHumans.count(p) != 0:
            activeHumans.remove(p)
```

1.2 Throttle

We modified the build-in `Delay` box to create our `Throttle` box.

```
class MyClass(GeneratedClass):
    def __init__(self):
        GeneratedClass.__init__(self, False)

    def onLoad(self):
        self.coolDown = False # True for cooling
        self.delayed = []
        self.lastInput = 0

    def onUnload(self):
        self.cancelDelays()

    def cancelDelays(self):
        cancel_list = list(self.delayed)
        for d in cancel_list:
            d.cancel()

    def cleanDelay(self, fut, fut_ref):
        self.delayed.remove(fut)

    def triggerOutput(self):
        self.coolDown = False # Reset cooldown at the end of the
timer

    def onInput_onStart(self, p):
        import qi
        import functools
        if not self.coolDown: # Only if not in cooling
```

```
            delay_future = qi.async(self.triggerOutput,
delay=int(self.getParameter("Timeout (s)") * 1000 * 1000))
            self.delayed.append(delay_future)
            bound_clean = functools.partial(self.cleanDelay,
delay_future)
            delay_future.addCallback(bound_clean)
            self.coolDown = True # set cooldown to true
            self.timerOutput(p) # call onstopped instantly

    def onInput_onStop(self):
        if self.getParameter("Trigger timerOutput if cancelled")
and self.delayed:
            self.timerOutput()
        self.onUnload()
```

2. Basic Introduction

After starting the session, pepper can answer some simple questions about the campus, such as:

H2

- How long is the history of this school?
- Who is the founder of this school?

In order to solve the hit problem better, we use the concept syntax in dialog.

```
# questions
concept:(oursch) ["这个学校" "我们学校" "浙大附中"]
concept:(qend) ["是谁" "是哪位" "是哪个" "是什么"]

u: (~oursch 的创建者 ~qend) 是丰子恺,潘天寿等明远学社成员呀
u: (~oursch 的校长 ~qend) 是申屠校长啊 $headmaster=1
u: (~oursch 有多少年了) 1947到2021,一共75年了
```

3. Navigation

Campus may be the area that navigation software can't cover accurately, so it's very important to add the navigation function to the robot. For better experience, we give directions from the three dimensions of voice, graphics and posture, so as to ensure the accuracy and communicability of the navigation.

H2

3.1 Trigger

The event syntax in dialog is used to trigger. After triggering, the `Timeline` and `show web view` are used to complete the indication of body actions and graphics respectively.

```
concept:(where) ["在哪" "在哪里"]
u: (食堂 ~where) 右转上坡道，食堂在左手边 $pointRest=1

# - Where is the canteen?
# - Turn right up the ramp and the canteen is on the left
```

3.2 Basic User Interface

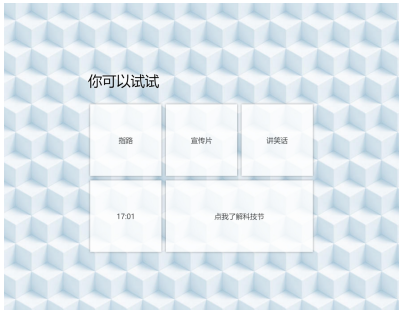
Set up an HTTP server, and display the corresponding pages according to the request parameters

Show Map

A simple JavaScript code snippet:

```
window.onload = function () {
    if (location.search === "?action=restaurant") {
        popup("去食堂", "<div id=\"map-container\"><img
src=\"./assets/map-rest.jpg\" height=300></div>");
        // call popup
    }
};
```

Using front-end technology, we can also add more functions to the interface

Picture	Description
	Initial state, Every button can be clicked except time
	Click to show the campus plan it can provide some guidance even if the dialog does not work properly
	show detailed information

To facilitate the setting of server address, we exposed a global variable:

```
global tabletServer
tabletServer = "http://my-server.com"
```

4. Daily Suggestion

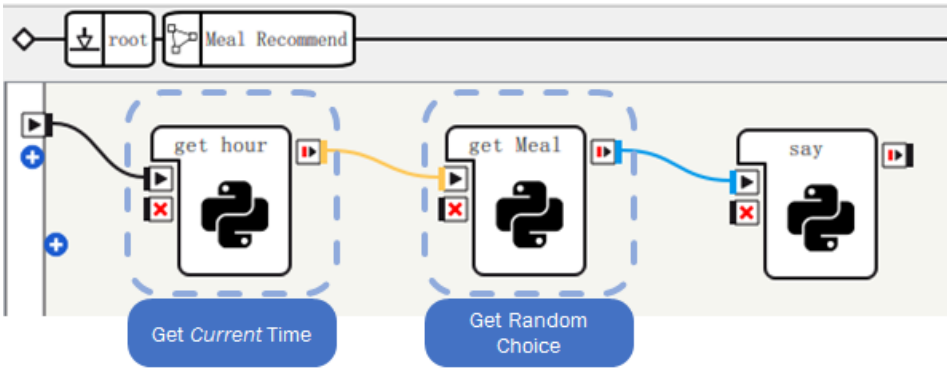
Provide suggestions to some unimportant and small problems in daily life. For example, let it help decide what to eat at noon today.

H2

After being triggered in the Dialog, the program will get the current time and the content of the meal, and provide some suggestions.

```
# meal
concept:(meal) ["吃什么" "有什么好吃的" "推荐点啥好吃的"]
u: (~meal) $mealRcm=1
```

Once triggered, the `Meal Recommend` process will be called to handle the `mealRcm` event, and a dish is randomly selected according to the time.



the internal structure of Meal Recommend Diagram

The implementation of part of the Python code is as follows:

```
import datetime
import random

breakfast = ["葱油饼", "肉包", "土豆饼", "油条", "鸡蛋饼", "梅干菜肉饼"]
dinner = ["牛腩面", "10元套餐", "12元套餐"]

if h >= 6 and h <= 8:
    self.onStopped("可能可以试试" + random.choice(breakfast))
elif (h >= 11 and h <= 13) or (h >= 16 and h <= 17):
    self.onStopped("不妨试试" + random.choice(dinner))
elif (h >= 19 and h <= 22):
    self.onStopped("我还没去看过夜宵呢")
else:
    self.onStopped("现在食堂应该还没饭")

pass
```

5. Simple User System

We have created a user system with face recognition as the core to screen out users who desire to interact and provide personalized services.

H2

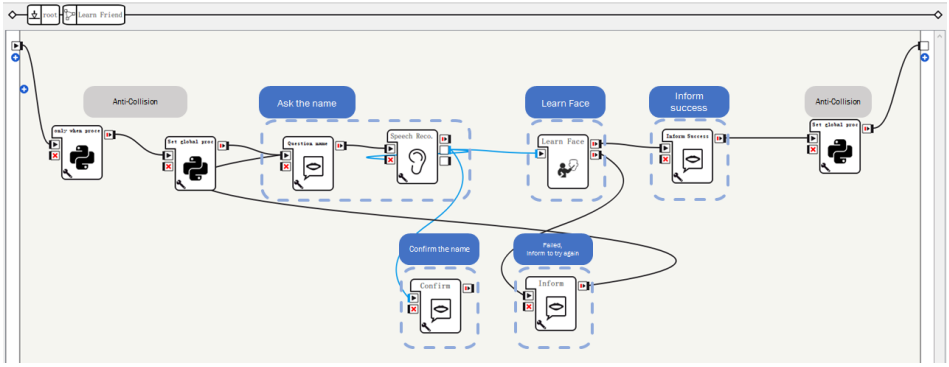
- Registration: "Make a friend" is involved in the conversation to let the robot learn the face
- Connect: Connect through face recognition when you see you
- Read and write user data: create a name-data mapping inside the module to store and access user data

H3 # 5.1 Registration

The Registration module called by the `lrnFace` event in the Dialog.

5.1.1 Main Registration Process

H4



Face recognition main program

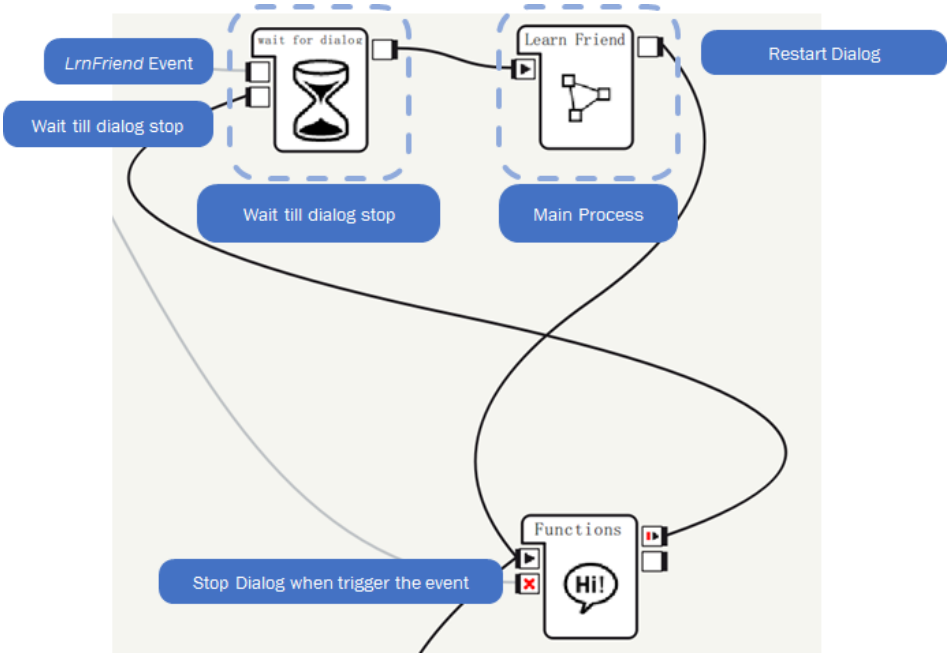
During this process, we set up a lot of prompts to ensure the smooth progress of user registration. The following is a sample dialogue:

```
[用户]：交个朋友吧
[Pepper]：好啊，交个朋友什么的，最喜欢了，你叫什么
# 提示交互流程开启

[用户]：用户姓名（方便起见为李华）
[Pepper]：是李华对吧，我要开始认人啦
# 提示正确听到了姓名

[Pepper}：（人脸学习中）

[Pepper]：我记住你啦
# 提示成功
```

5.1.2 Anti-Collision

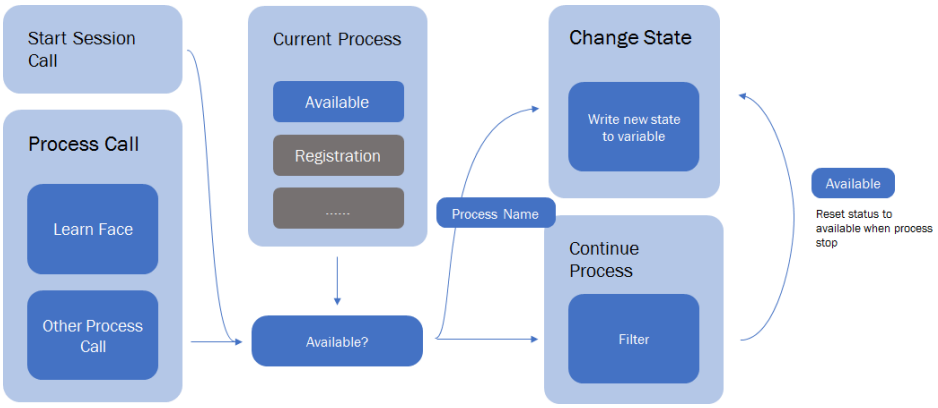
In order to prevent the voice recognition in Dialog from conflicting with the voice recognition in face learning, we adopted the following process to avoid conflict.

H4

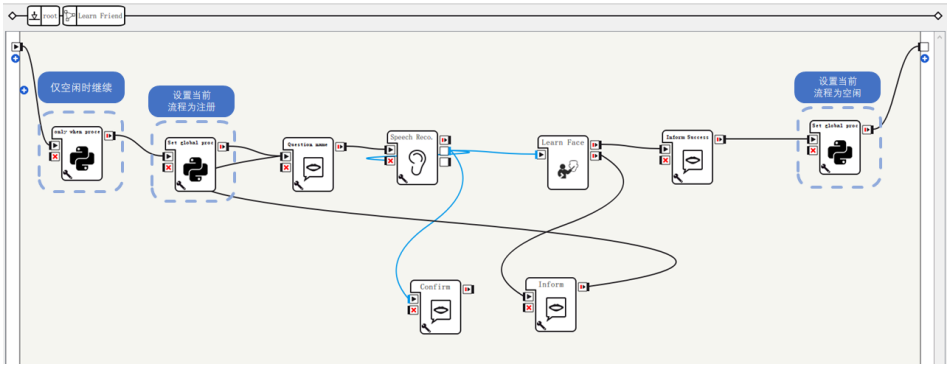


Solve the conflict between Dialog and speech recognition

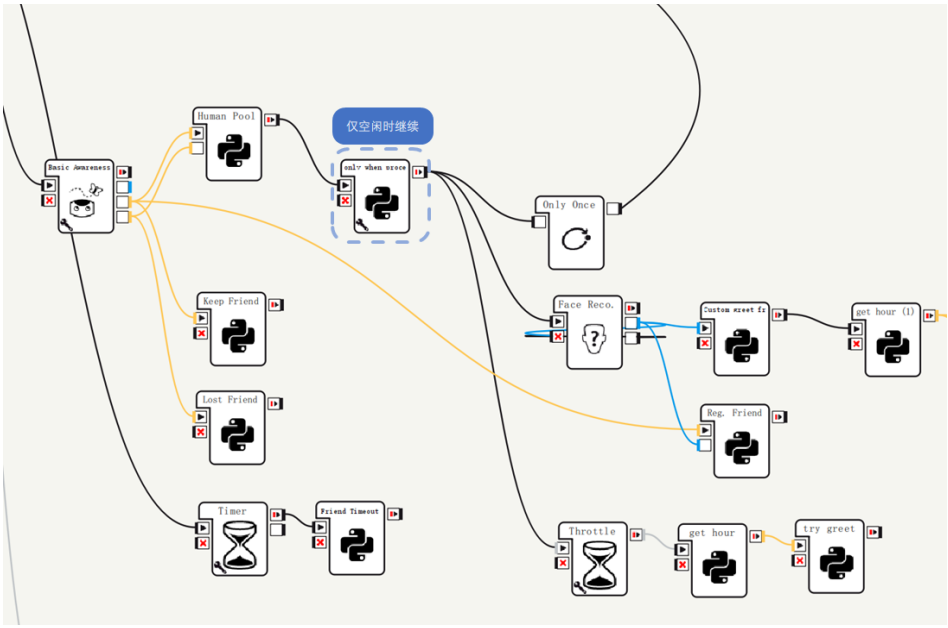
In order to prevent face recognition from being interrupted by greetings and to prevent face recognition from being accidentally triggered during other processes, we adopted the following solutions to avoid conflicts:



Use global process state to implement conflict avoidance



Anti-collision handling in event call



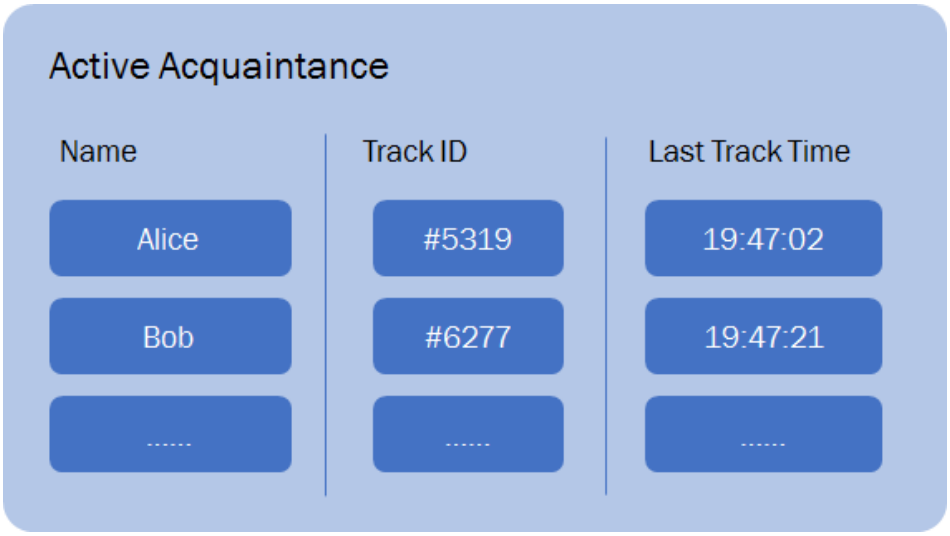
Anti-collision handling in greeting process

H3 5.2 Login

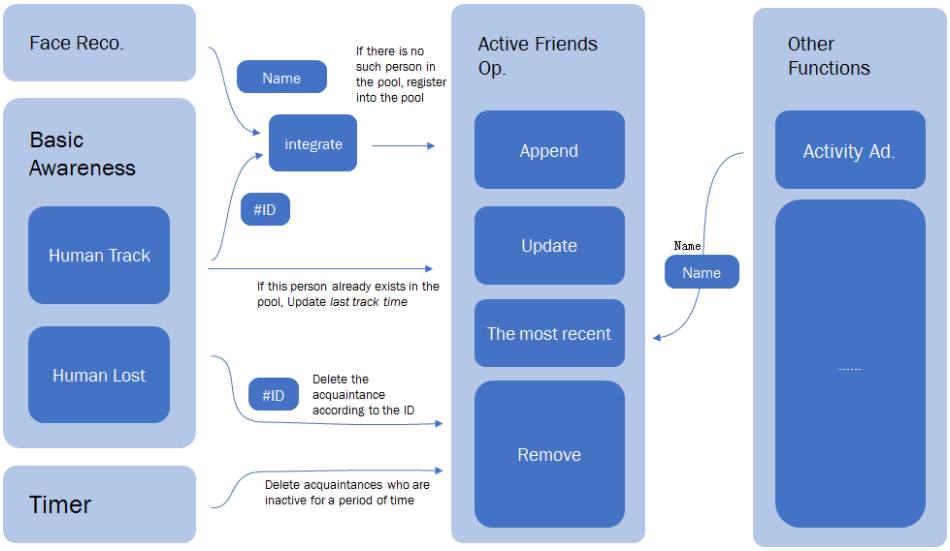
After Pepper recognizes the user, record the user's name for use by other services. Because the use of `Face Reco.` for every personalized service will bring a lot of uncontrollable factors, including but not limited to:

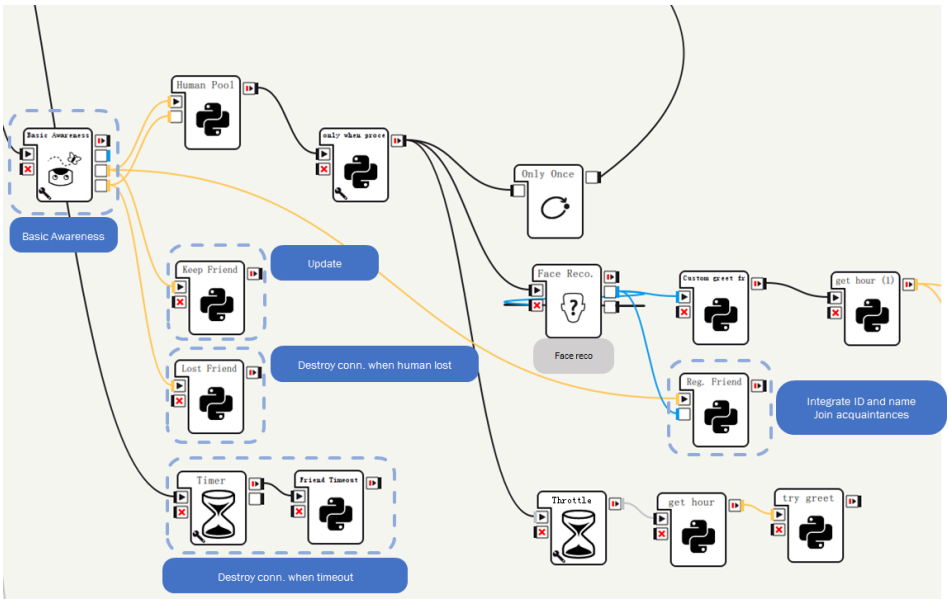
1. If it is a stranger, then the unrecognizable output will have to wait for `8s`, but this is a repeated attempt and can be optimized
2. The output of `Face Reco.` is not stable
3. Every time you add an `Face Reco.` box, the code-readability will decrease

We use the acquaintance pool mechanism to overcome these factors. Its data structure is as follows. The code uses `List` nested `Dict` to describe



Its specific implementation mechanism is as follows:





Join/update/destroy of acquaintance activity

The following code snippet implements the function of getting the last acquaintance:

```
global activeFriends
maxTime = 0
lastName = ""
for info in activeFriends:
    self.logger.info("info: " + str(info))
    if info['lastNotice'] > maxTime:
    lastName = info['name']

    if lastName != "":
        self.onStopped(lastName)
        return
    else:
        self.onNoFriend()
```

6. Festival Promotion & Basic Reminder

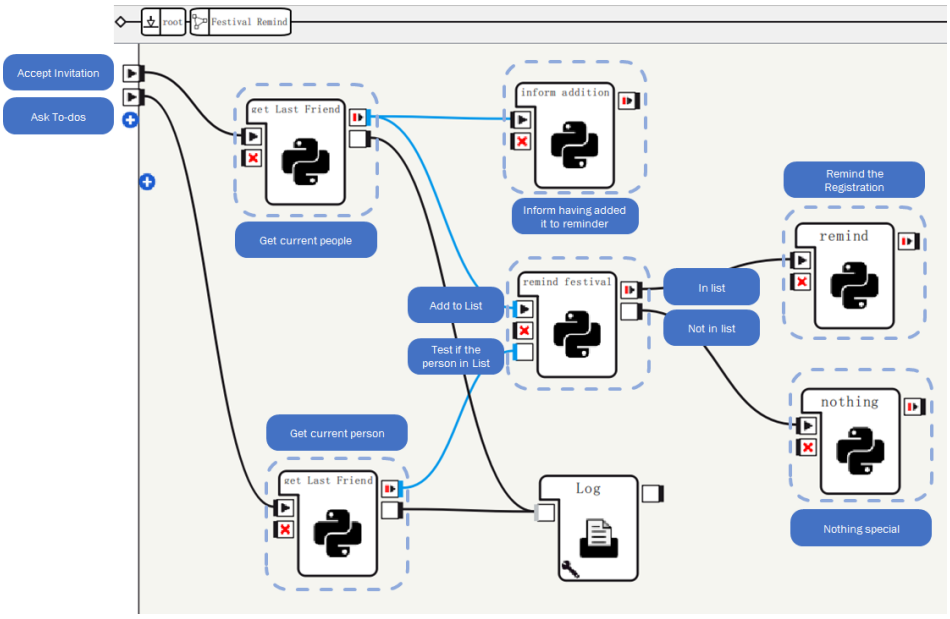
On campus, you can get recent activities in small chats with robots. If you are an acquaintance, you can also add the corresponding items to the memo.

H2

We use two events in the conversation to expose the entrance:

```
concept:(accept) ["好啊" "好" "行"]
concept:(actbegin) ["最近" "浙大附中" "我们学校" "最近几天"]
u: (~actbegin 有什么活动吗) 有一个科技节,在那里你可以发挥你的才华,把理想变为现实,想要参加吗
  u1: (~accept) 好的 $festivalSpecial=1
  u2: (不想) 好吧

u: (今天有什么事情吗) $reportEvent=1
```

If you are an acquaintance, and the answer is *yes* (means you are interesting in the festival), then the registration will be included in the to-do list.



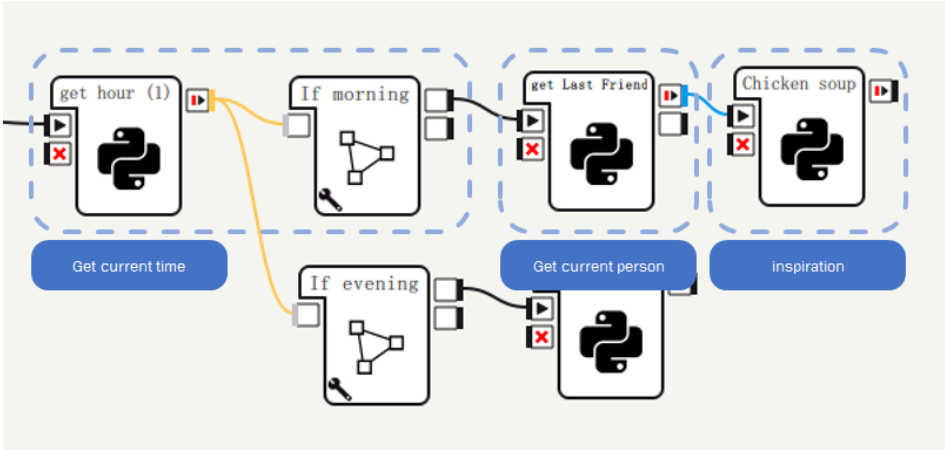
Task promotion and to-do realization

7. Greeting Function Additional

H3 7.1 Morning Inspiration

H2

Judging by time, Robot will give its acquaintances a random sentence of inspiration every morning, and in order to prevent interruption, the robot only greet the same person once.



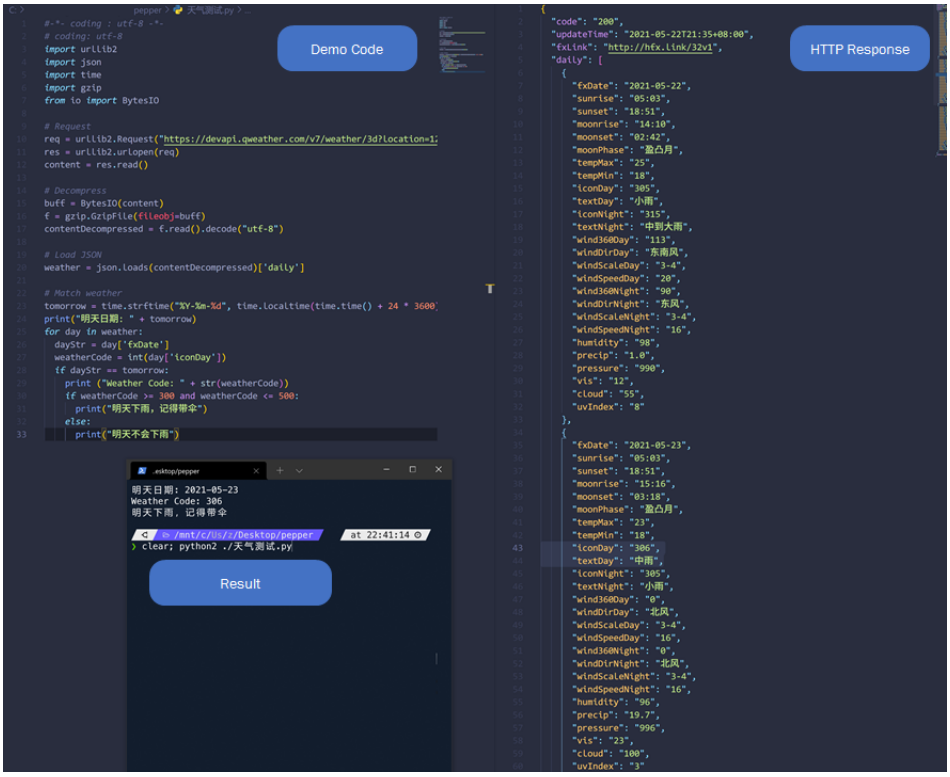
H3 7.2 Weather reminder

A weather reminder interface is reserved, and the weather API will be requested in real time in the future for complete implementation. Below is a demo implemented by python

```
import urllib2
import json
import time
import gzip
from io import BytesIO

# Request
req = urllib2.Request("https://devapi.qweather.com/v7/weather/3d?location=120,30&key=<MyAppKey>")
res = urllib2.urlopen(req)
content = res.read()

# Decompress
buff = BytesIO(content)
f = gzip.GzipFile(fileobj=buff)
contentDecompressed = f.read().decode("utf-8")

# Load JSON
weather = json.loads(contentDecompressed)['daily']

# Match weather
tomorrow = time.strftime("%Y-%m-%d", time.localtime(time.time() + 24 * 3600))
print("明天日期: " + tomorrow)
for day in weather:
    dayStr = day['fxDate']
    weatherCode = int(day['iconDay'])
    if dayStr == tomorrow:
        print ("Weather Code: " + str(weatherCode))
        if weatherCode >= 300 and weatherCode <= 500:
            print("明天下雨，记得带伞")
```

```
        else:
            print("明天不会下雨")
```

The test is successful, and the results are as follows:



8. Demo time simulation

In order to demonstrate all the time-related function of the robot (such as *Morning Inspiration*), time simulation is of great necessity.

H2

We use two global variables in implementation.

```
global lastSyncRealTime # Real time when last synced
global lastSyncSimuTime # Simulation time at the last
synchronization
```

The simulation time can be calculated by only one line:

```
time.time() - lastSyncRealTime + lastSyncSimuTime
```

There are two ways to switch the simulation time:

H3 8.1 By touching head

By adding the `MiddleTactilTouched` event listener to realize the cycle switching between morning and night

code segment:

```
self.timeSimu = (self.timeSimu + 1) % 3

self.tts = ALProxy('ALTextToSpeech')

global lastSyncRealTime
global lastSyncSimuTime

if self.timeSimu == 0:
    self.tts.say("假装现在是早晨")
    lastSyncRealTime = time.time()
    lastSyncSimuTime = 1621549219
elif self.timeSimu == 1:
    self.tts.say("假装现在是中午")
    lastSyncRealTime = time.time()
    lastSyncSimuTime = 1621570819
elif self.timeSimu == 2:
    self.tts.say("假装现在是晚上")
    lastSyncRealTime = time.time()
    lastSyncSimuTime = 1621603219
```

H3 8.2 By setting up a background server

We use `Node.js` to implement a simple time server and background interface to complete the synchronization: the front end uses `/getSimuTime` to change the simulation time, and Pepper uses polling `/getSimuTime` to complete the time synchronization

Server-side code:

```
(function(app) {

  // init times
  let realTime = (new Date().getTime()) / 1000,
      simuTime = (new Date().getTime()) / 1000;

  // `/setSimuTime` API
  app.get('/setSimuTime', (req, res) => {
    try {
      const reqRealTime = +req.query.realTime,
            reqSimuTime = +req.query.simuTime,
            valiStr = +req.query.valiStr;

      if(!vali(
         reqRealTime.toString()
         + reqSimuTime.toString(),
          valiStr
      )) {
        throw new Error("validate the request failed")
```

```
      }

      realTime = reqRealTime,
      simuTime = reqSimuTime;;
      res.status(200);
    } catch (e) {
      res.status(400).end();
    }
  })

  // `/getSimuTime` API
  app.get('/getSimuTime', (req, res) => {
    res.send(JSON.stringify({realTime, simuTime}))
    res.end();
  })

})(app);
```

Pepper-side code:

```
try:
    global lastSyncRealTime
    global lastSyncSimuTime
    global tabletServer
    req = urllib2.Request(tabletServer + "/getSimuTime")
    res = urllib2.urlopen(req)
    j = json.loads(res.read().decode('utf-8'))
    lastSyncRealTime = j['realTime']
    lastSyncSimuTime = j['simuTime']
    self.logger.info("[time server] update successful" + str(j))
except:
    self.logger.warn("[time server] failed to fetch time status")
```