# Meow Meow

## Smart Pet Interaction System

Embedded Systems Labs, Spring 2017
B02901061 鄧郁璇 B02901080 董皓文

## Section 1 - Meow Meow

***Meow Meow*** is composed of two separate systems, as illustrated below.

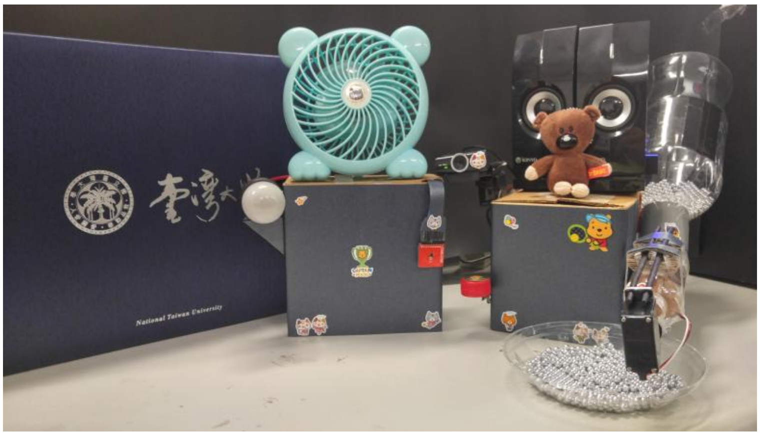


**Fig. 1** ***Meow Meow,* a smart pet interaction system**

## Section 2 - Environment Monitoring System

- **Climate module**
  - monitor the temperature and humidity
  - update *climate/temperature* and *climate/humidity* in Firebase realtime database
  - update *climate/hot* to true when the temperature rises over 28°C and update *climate/hot* to false when the temperature drops below 25 °C.

- **Relay module**
  - control the fan and the light bulb
  - automatically turn on the fan when temperature rises
  - automatically turn on the light at night(18:00~06:00) and turn it off at daytime(06:00~18:00)
  - enable clients to send request to control the fan and the light bulb

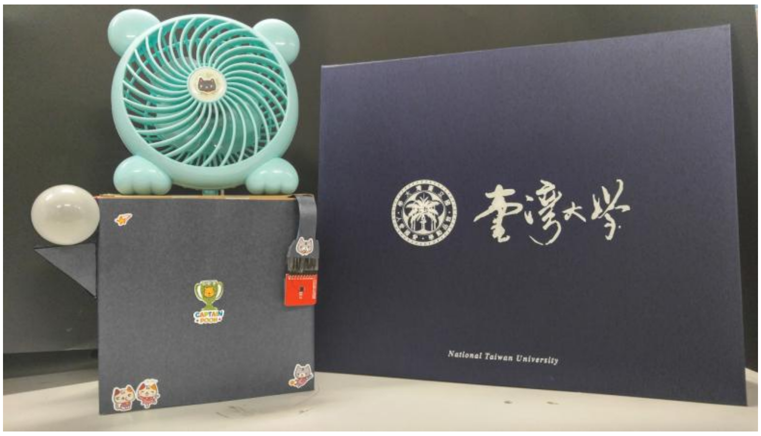


**Fig. 2 Environment monitoring system**

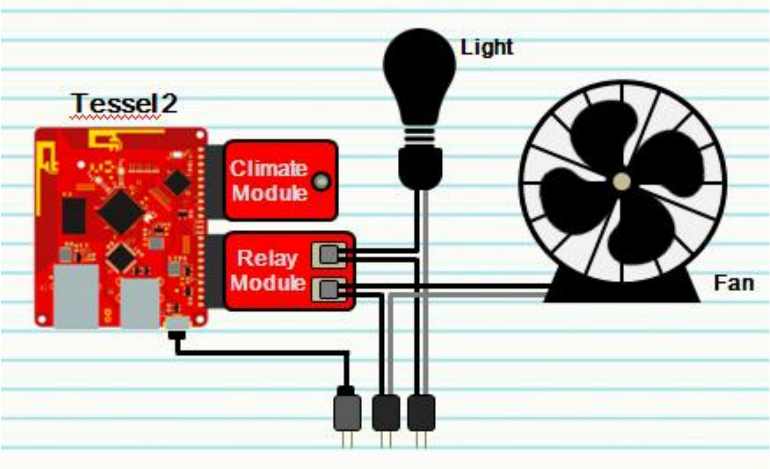


**Fig. 3 Environment monitoring system structure**

## Section 3 - Interactive Feeding System

- **Server**
  - establish a local host using socket.io
- **Button(via GPIO)**
  - enable your pet to call you
- **Servo module**
  - control the direction of the camera
  - control the pet food dispenser
- **USB camera module**
  - stream live video
- **USB audio module**
  - play recorded voice message when you call
  - play recorded voice message when you feed remotely

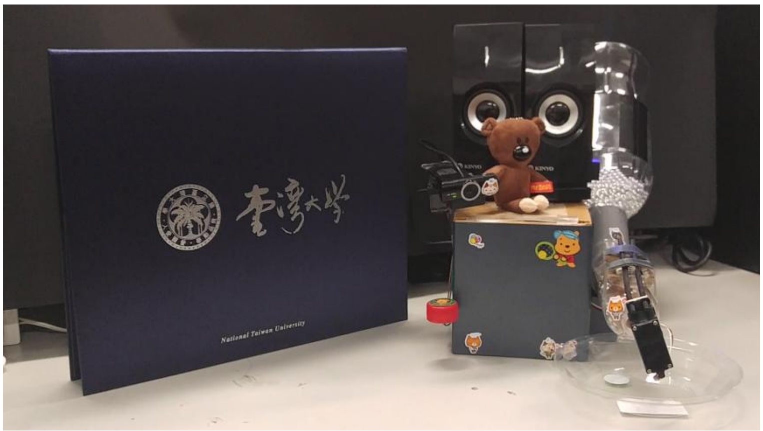


**Fig. 4 Interactive feeding system**

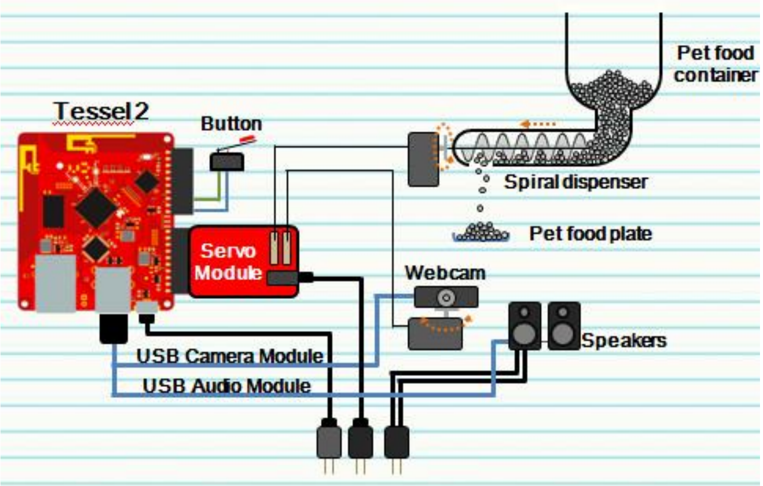
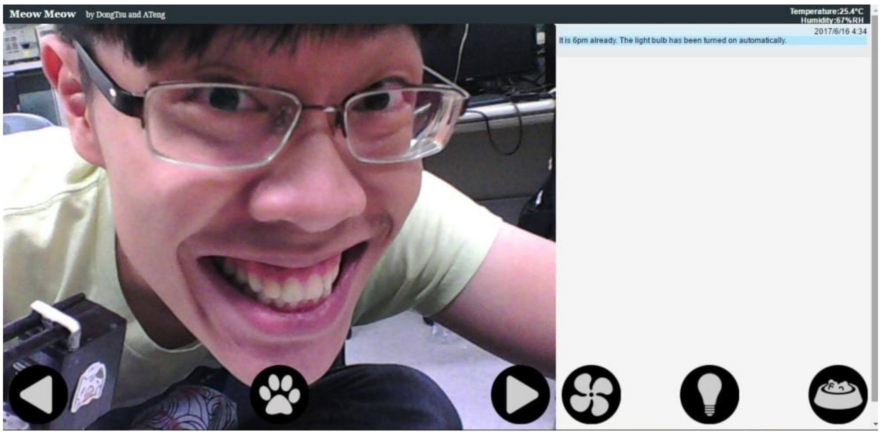


**Fig. 5 Interactive feeding system structure**

## Section 4 - User Interface



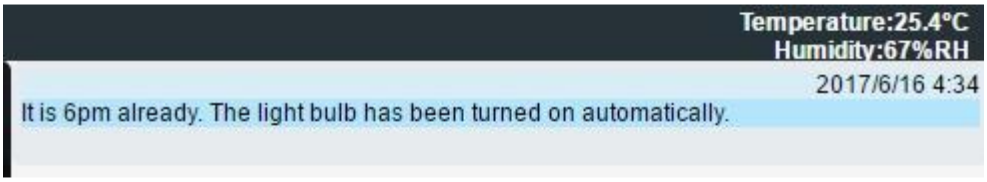
Temperature:25.4°C
Humidity:67%RH

2017/6/16 4:34

It is 6pm already. The light bulb has been turned on automatically.

**Fig. 6 User Interface**

- **Firebase API**
    - connect to Firebase realtime database
    - fetch climate data updated by the environmental monitoring system
    - acquire the state of relay module of the environmental monitoring system
    - send relay control request when the fan/light buttons are clicked(will be elaborated in Sec. 5)
- **Message List**
    - show messages when your pet calls you by pressing the GPIO button
    - show messages when the temperature rises over 28$^{o}$C
    - show messages when the light bulb is turned on automatically at night
    - show date and time information of every message



Temperature:25.4°C
Humidity:67%RH

2017/6/16 4:34

It is 6pm already. The light bulb has been turned on automatically.

**Fig. 7 Message List**

- **Buttons**
  - Camera control ◀ ▶
    - trigger the servo module
    - adjust the direction (left/right) of camera
  - Call meow 🐾
    - trigger the audio module
    - your pets will hear your prerecorded voice
  - Fan switch 🌀 ↔ 🌀
    - trigger the relay module
    - turn on/off the fan
  - Light switch 💡 ↔ 💡
    - trigger the relay module
    - turn on/off the light
  - Feed button 🥣
    - trigger the servo module
    - activate the feeding system
    - trigger the audio module
    - your pets will hear your prerecorded voice

## Section 5 - Implementation Details

### How do we use Firebase API to establish robust connection?

We try to construct a request/response connection model between the browser(which acts like a client) and the environmental monitoring system(which acts like a server). To illustrate the idea, we take light control for example and some trivial checks are skipped. Note that true means 'on'/'turn on' and false means 'off'/'turn off' for *relay/status.lightIsOn and relay/command.switchLight*.

- Whenever the user click the light bulb button, the client will check if *relay/command.switchLight* is equal to *relay/status.lightIsOn*. If so, then the client will change *relay/command.switchLight* into the target state. If *relay/command.switchLight* is not equal to *relay/status.lightIsOn*, it means the server has not completed the last command sent by the clients so we simply skip it..

- The server listens on *relay/command.switchLight*. Whenever its value is changed, the server will change the relay controlling the light bulb into the target state.

- After the server has successfully change the state, the server will change the *relay/status.lightIsOn* into the current state of the relay controlling the light bulb. Note that this is why we say if *relay/command.switchLight* is not equal to *relay/status.lightIsOn*, it means the server has not completed the last command sent by the clients.

- The client listens on *relay/status.lightIsOn*. Whenever its value is changed, the client will change the unlighted light bulb image into lighted light bulb image or in the opposite way.