ESYS Final Report

BUG

Be Ur Guard

Han Jian b05901044@ntu.edu.tw

Table of Content

Why Making BUG

When it comes to home safety, security camera is always the top choice because for human it is easy to read the abnormal, however the recorded videos are hard for resource constrained devices to store or analyze at realtime. So BUG makes use of other sensor-able materials such as sound, position and luminosity to record signals from environment as events.

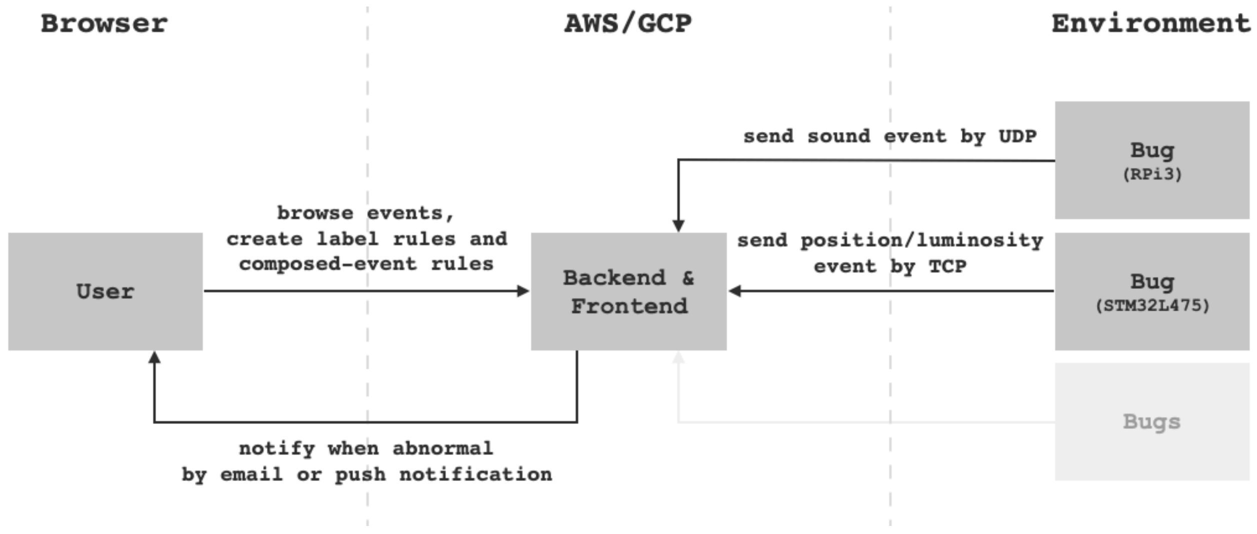
Most time a meaningful event for human is composed by multiple single-material events. For example, opening a door might be accompanied with:

- Specific Sound
- Different Position
- Different Luminosity

So BUG provides user an interface to define rule about how to label single-material events and how to compose the labelled events to be human readable ones as composed event.

How BUG Works

BUG consists of 3 parts: Bug, Backend, Frontend.


Browser
AWS/GCP
Environment
send sound event by UDP
Bug (RPi3)
browse events, create label rules and composed-event rules
send position/luminosity event by TCP
User
Backend & Frontend
Bug (STM32L475)
Bugs
notify when abnormal by email or push notification

BUG - Bug

Bug represents the devices recording events. For this proof of concept, we have two MCU serving as Bug

- STM32L475
  - Utilize acceleration sensor on MCU to detect whether self position has difference
  - Utilize photoresistor sensor to detect whether luminosity has difference (not implemented)
  - Send position, luminosity event to Backend through TCP
- RPi3
  - Use USB microphone to sense sound around it
  - Send sound event to Backend through UDP

BUG - Backend

Backend serves as a TCP/UDP server for devices to report events

Backend also serves as a RESTful API server for user at browser to interact with

- Create/Read/Delete on devices
- Read/Subscription on events
- Create/Read/Delete on labels

After label created, the following events will be checked whether match the rule, if matched, it will be labelled automatically. (not implemented for all events yet)

BUG - Frontend

Frontend serves web assets for browser to render. Users can browser the event history day by day, manage devices on it, create label specifications and composed events specifications.

Proof of Concept

Demonstration video is available here.

Challenge

Keep TCP Connection Alive

When integrated with wifi module wifi-ism43362, it has no way to know whether MCU is connected to remote socket. After digging into the source code, I found there is a private member `connected` recording the state of it. So I try to inherit the class `TCPSocket` and utilize the struct layout to peek the state to ensure the connection is alive.

For example:

```
#include "TCPSocket.h"

// Copy from ISM43362Interface.cpp
struct ISM43362_socket {
  int _;
  nsapi_protocol_t __;
  volatile bool connected;
};

class TCPClient : protected TCPSocket {
  bool connected() {
    return ((struct ISM43362_socket*)_socket)->connected;
  }
};
```

Another challenge I found that I haven't found a way to reconnect to wifi after access point restarts. After digging into the source, I found there is an api to reset the ISM43362 module, after reset and reconnecting works! So I forked the repository to expose the function `reset` of `ISM43362` on class `ISM43362Interface`. Here is the patch commit.

Data Filtering

For data filtering, I use noise gate algorithm to filter noise. It is implemented as a finite state machine with 3 state Opened, Closing, Closed. With the Closing state design, it can effectively prevent signal from split into too much fragment. Here is the pseudo code for next state:

```
enum State { Opened, Closing, Closed }

def next_state(value):
  over_threshold = value > threshold

  if state is Opened and not over_threshold:
    state = Closing
    remaining_release_count = release_count
  elif state is Closing:
    if over_threshold:
      state = Opened
    else:
      if remaining_release_count == 0:
        state = Closed
      else:
        remaining_release_count -= 1
  elif state is Closed and over_threshold:
      state = Opened
```

To Improve

- Implement luminosity tracker
- Implement auto event labelling
- Implement composed event composing
- Implement email notification and web push notification
- Use TLS/DTLS between Bug and Backend
- Offline mode for Bug

Credits

- Audio Processing for Dummies

Miscellaneous

Github page is ready online! FYI:

- Title

  BUG - Be Ur Guard
- Description

  BUG utilizes kinds of event from environment to detect anomalies for home security.
- Member

  Han Jian (簡廷翰)
- Github Repo
- Github Page
- Thumbnail