

Embedded System Software Final Project - ootdPT

Junhyeok Park
20171643



Motivation



IoT (*Internet on Things*)

Motivation



Ex) Intercom

COMMAX
SmartHome & Security

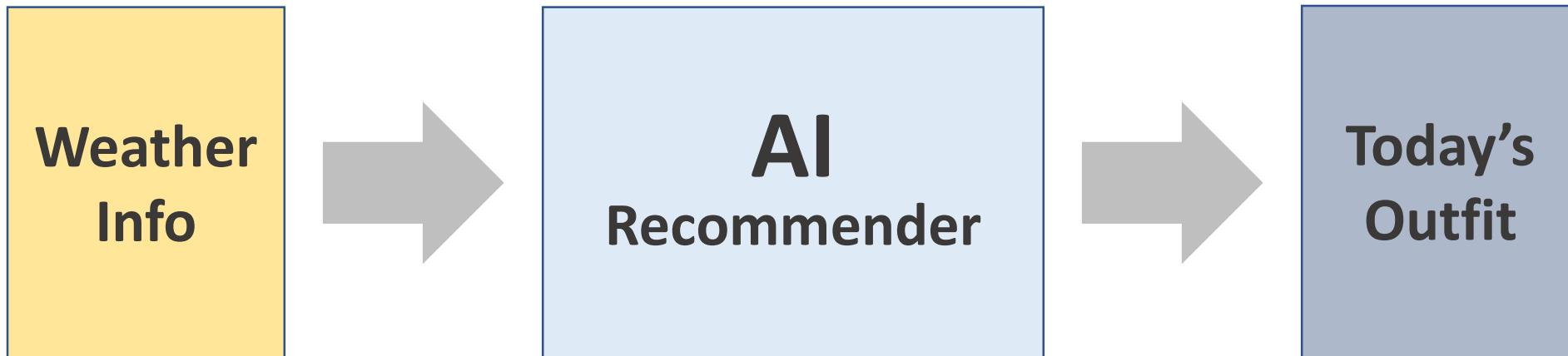
KOCOM

Motivation

*What about
outfit recommendation?*

Motivation

*What about
outfit recommendation?*





Idea

ootdPT

Outfit Of The Day

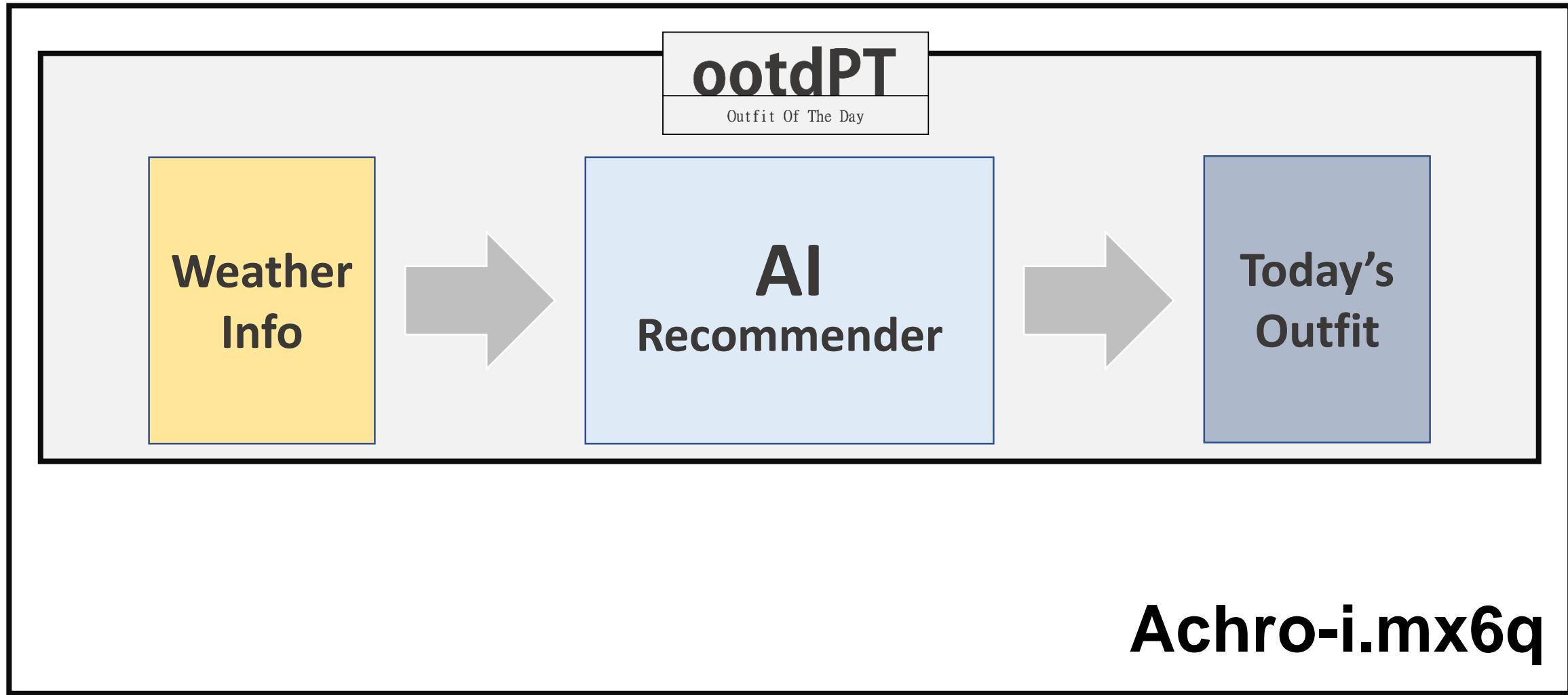
Idea

ootdPT

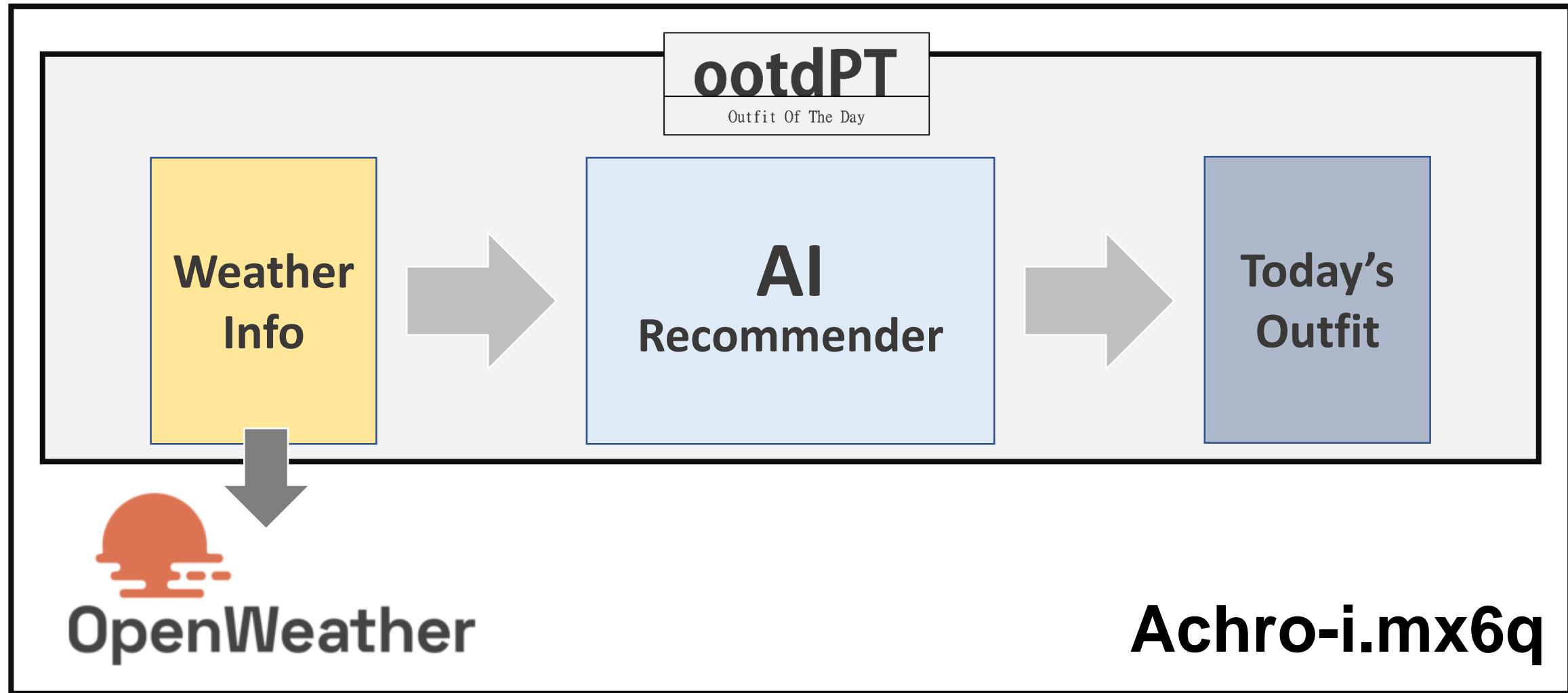
Outfit Of The Day

Android Application
on achro-i.mx6q device

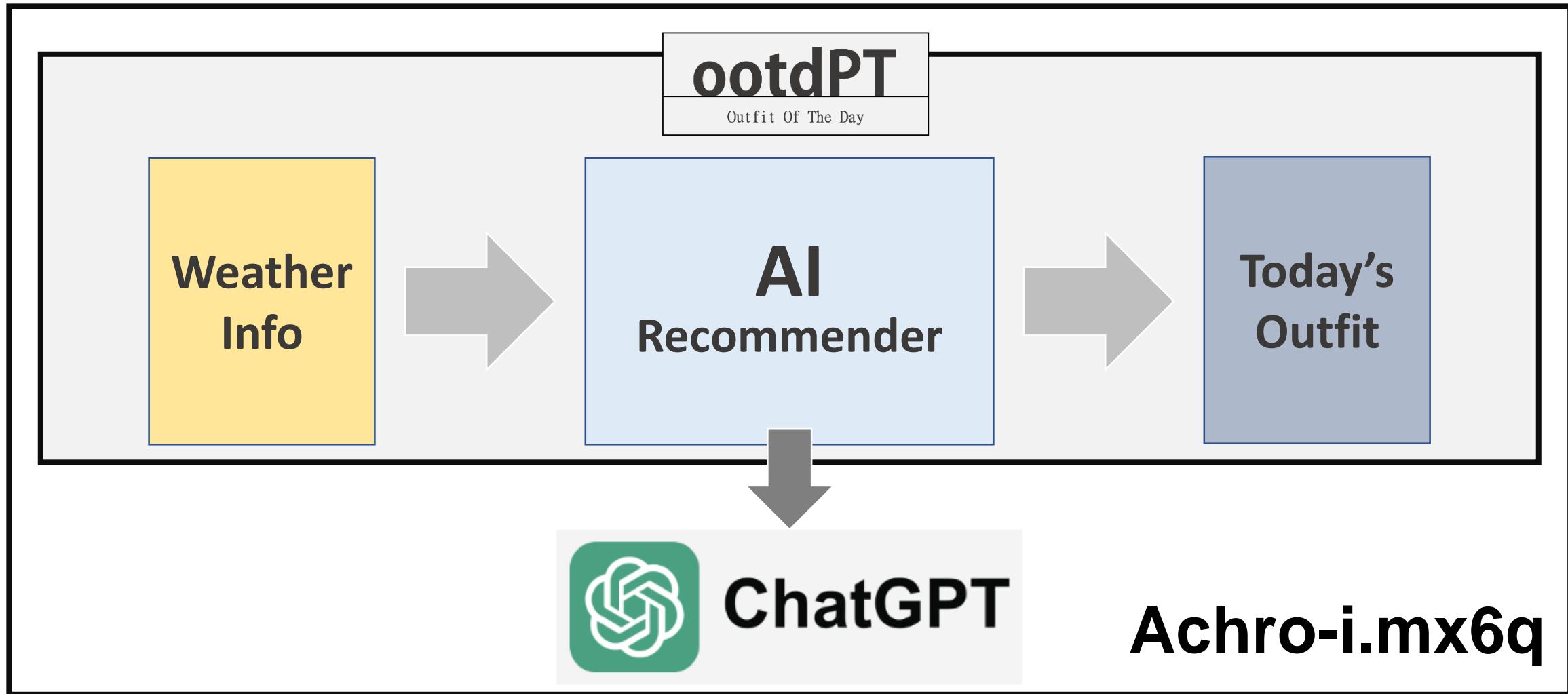
Idea



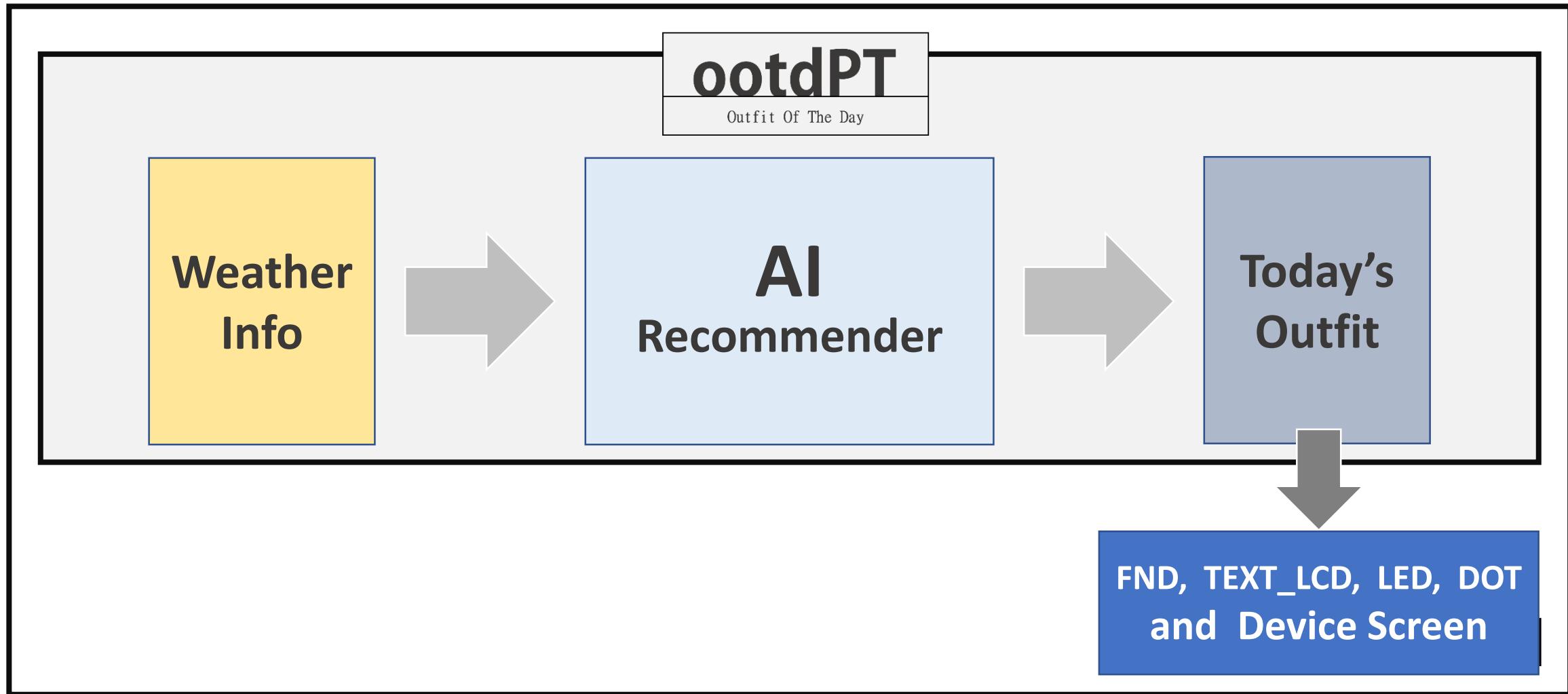
Idea



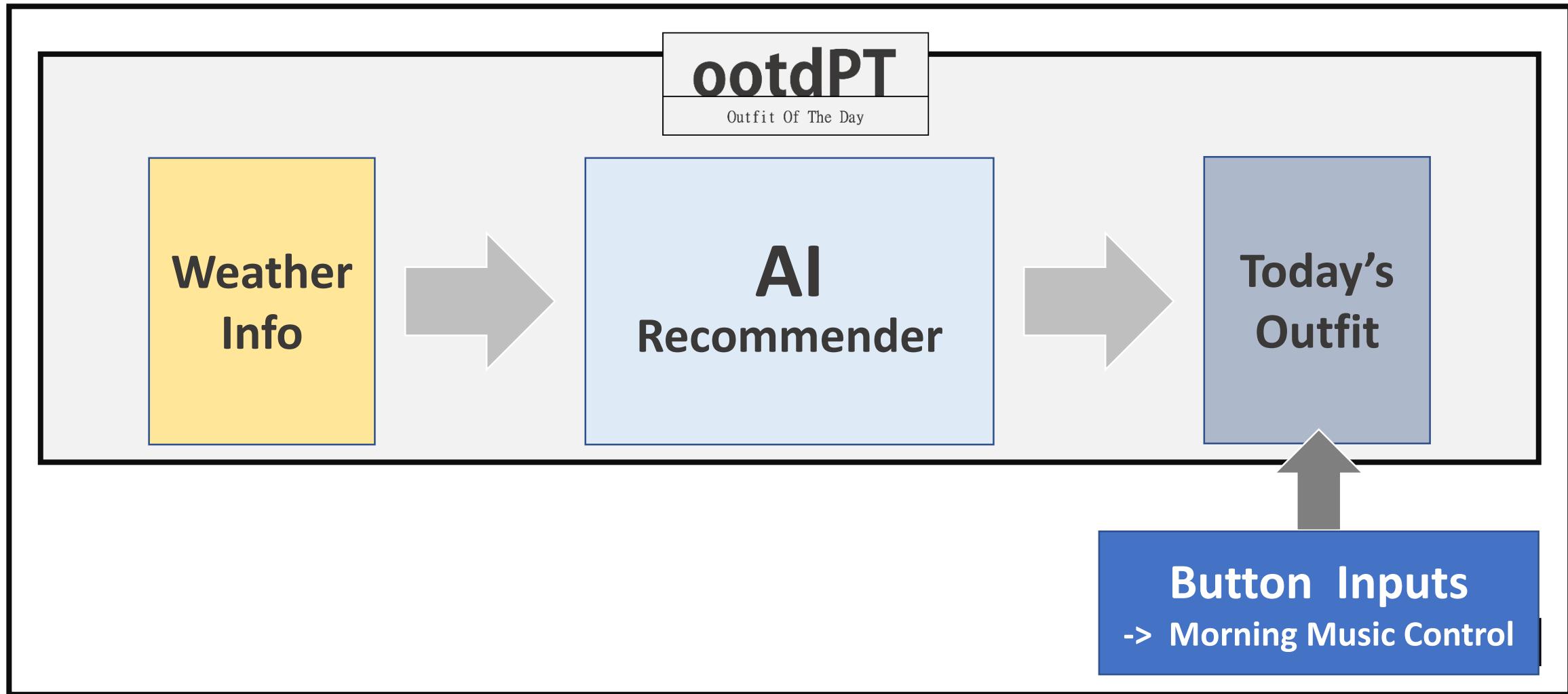
Idea



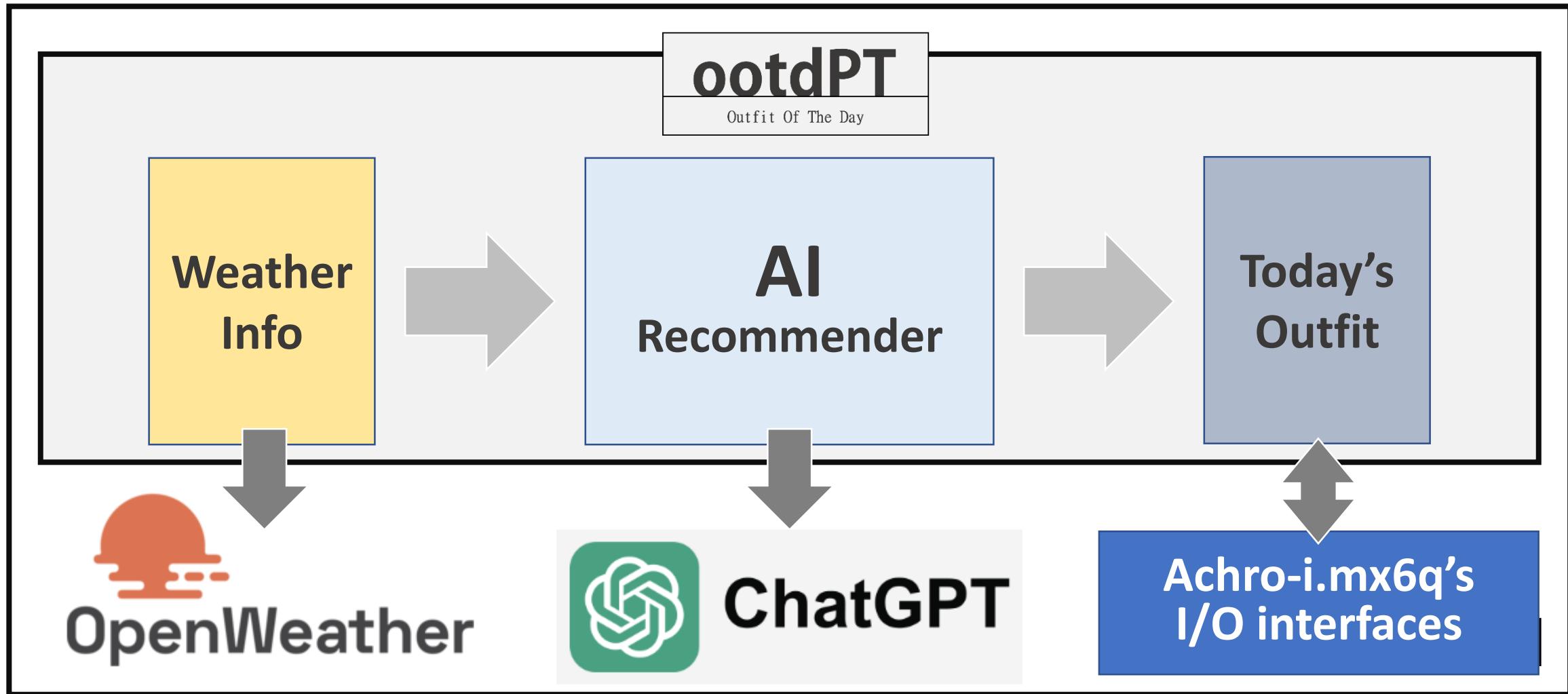
Idea



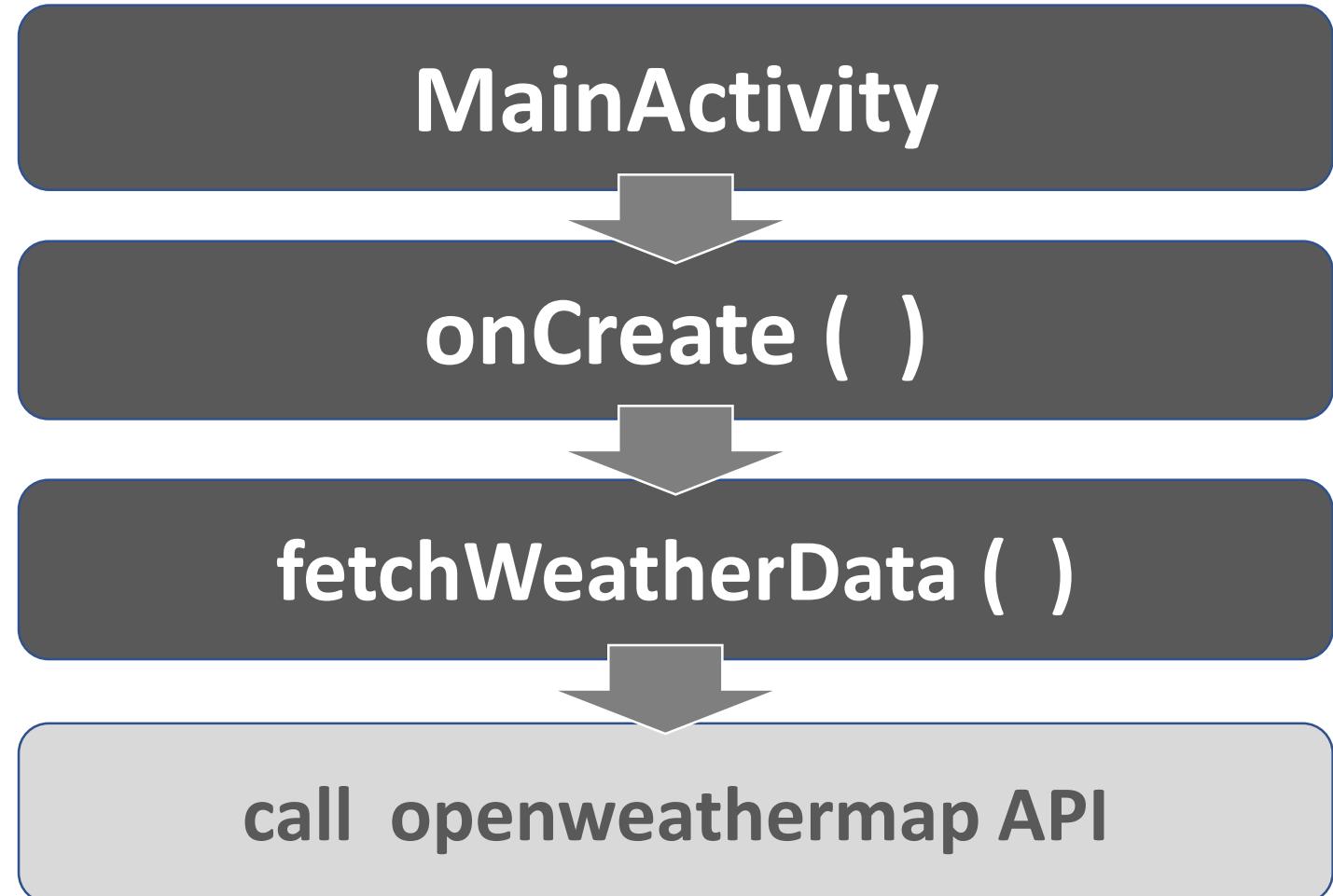
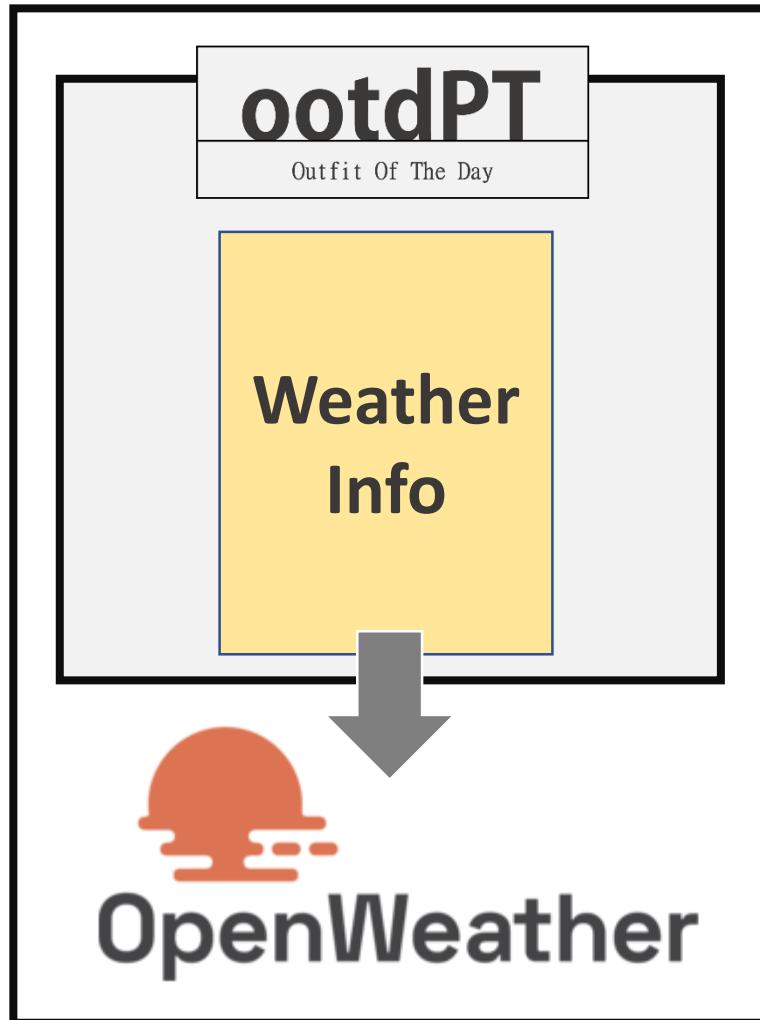
Idea



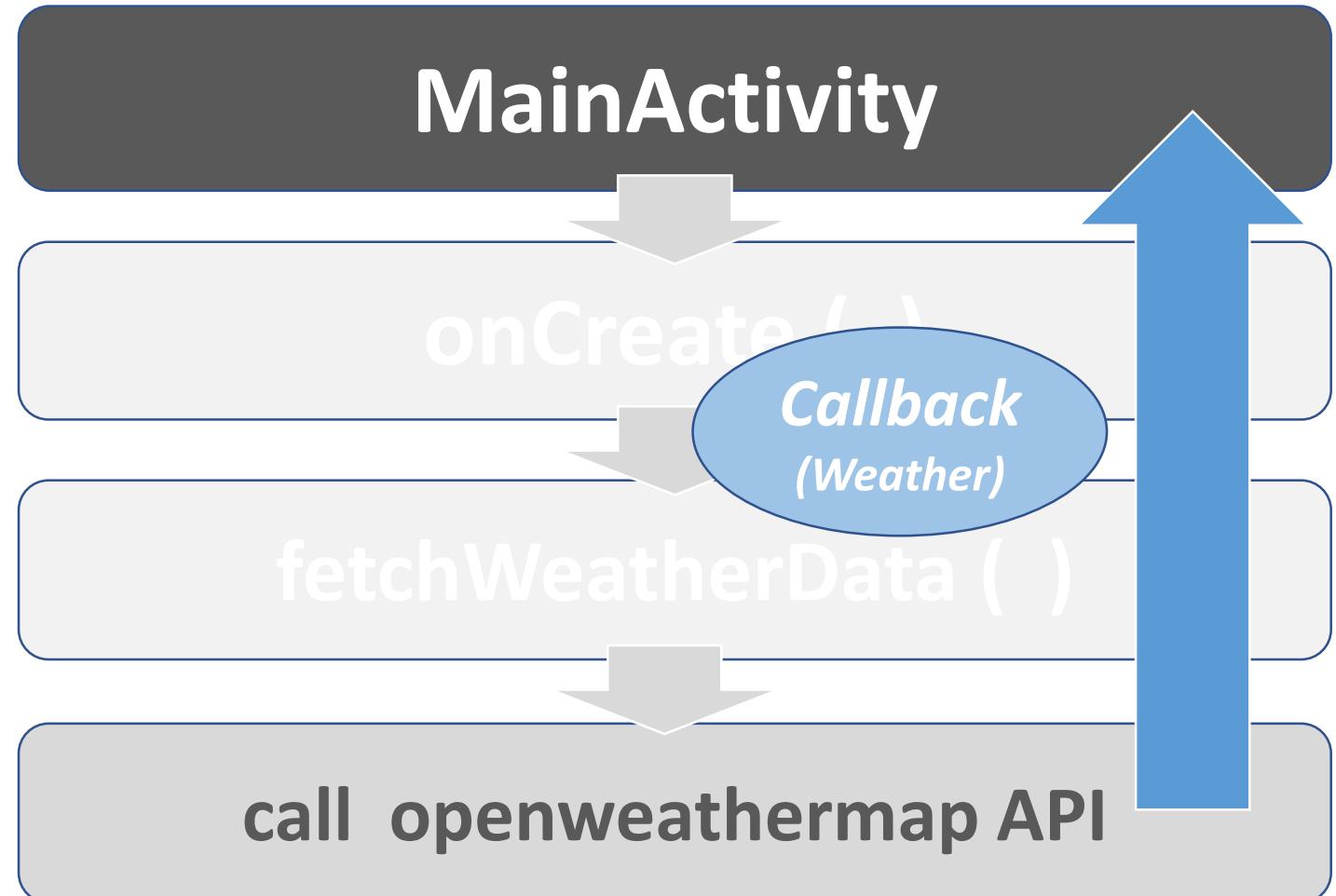
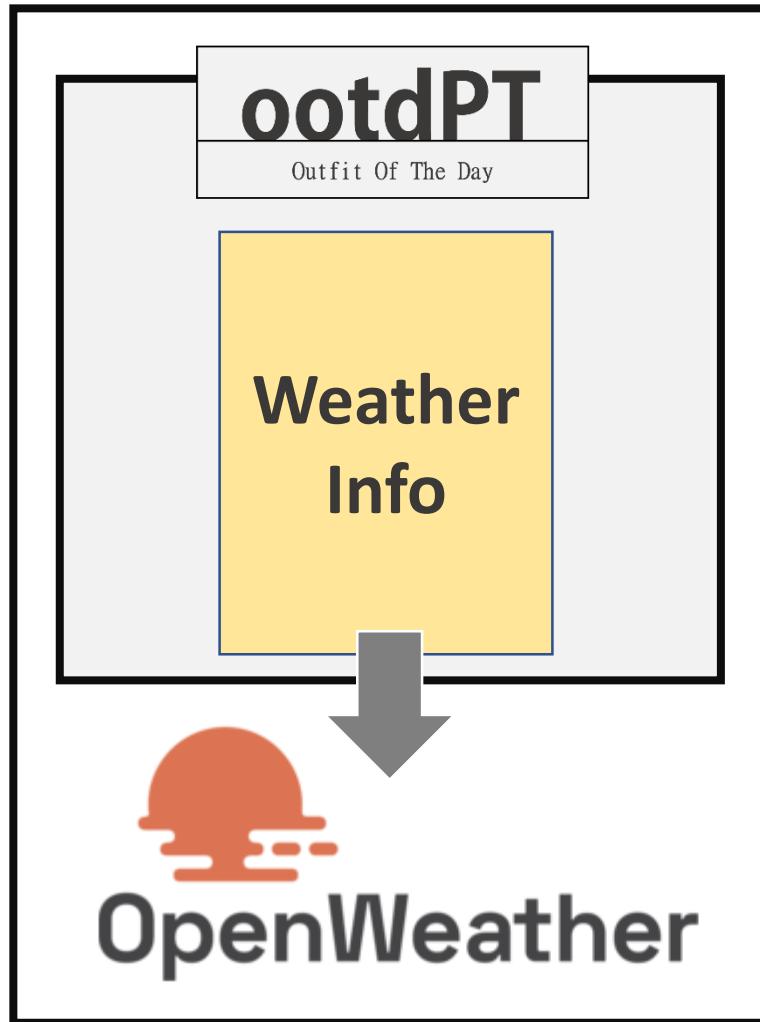
Overall Structure



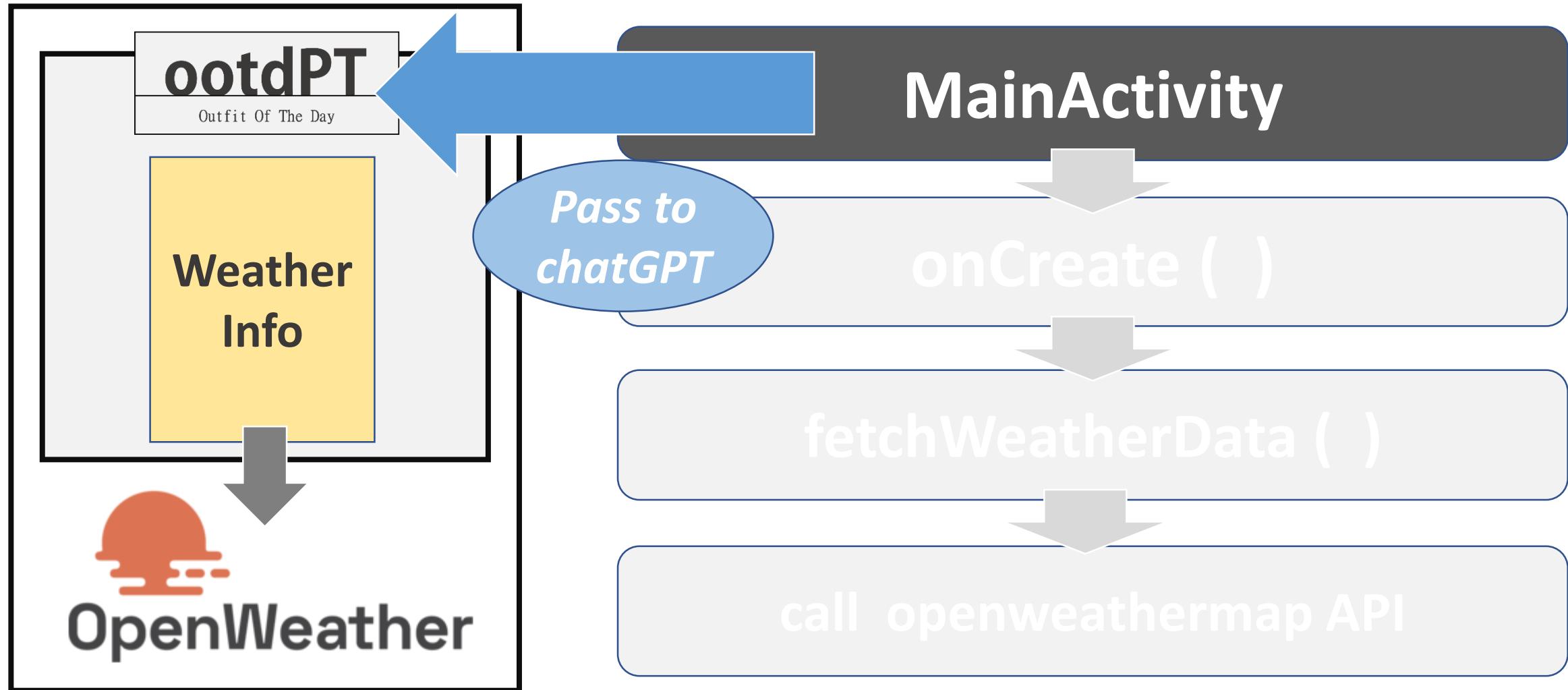
Weather Information



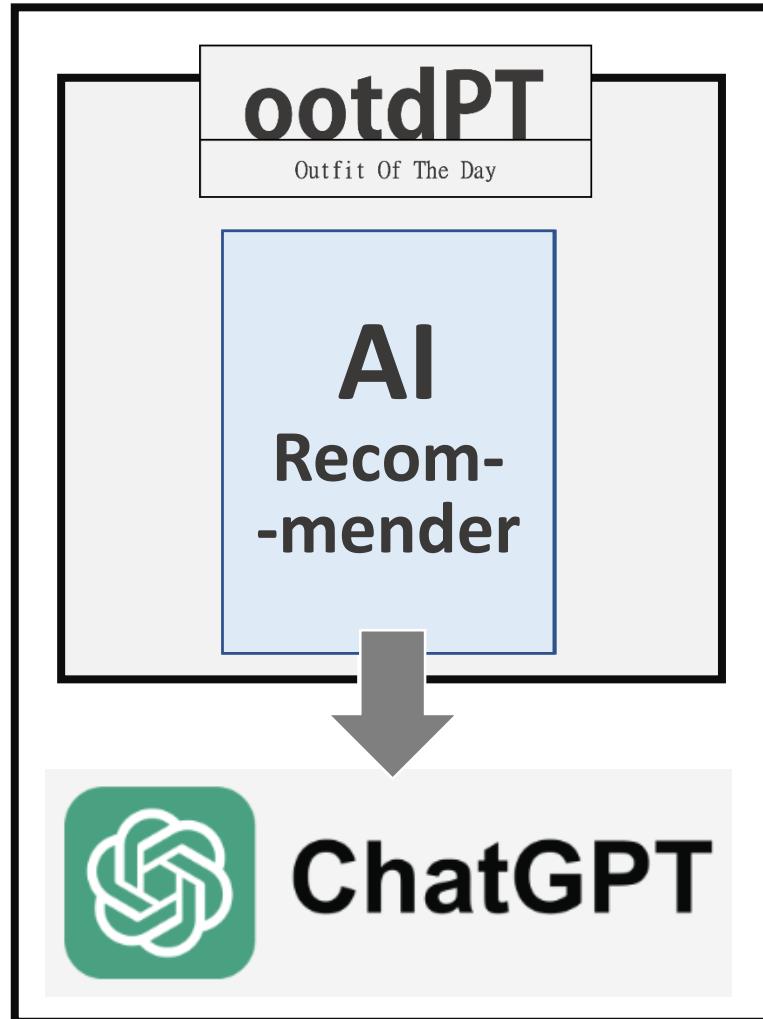
Weather Information



Weather Information



Outfit Recommendation

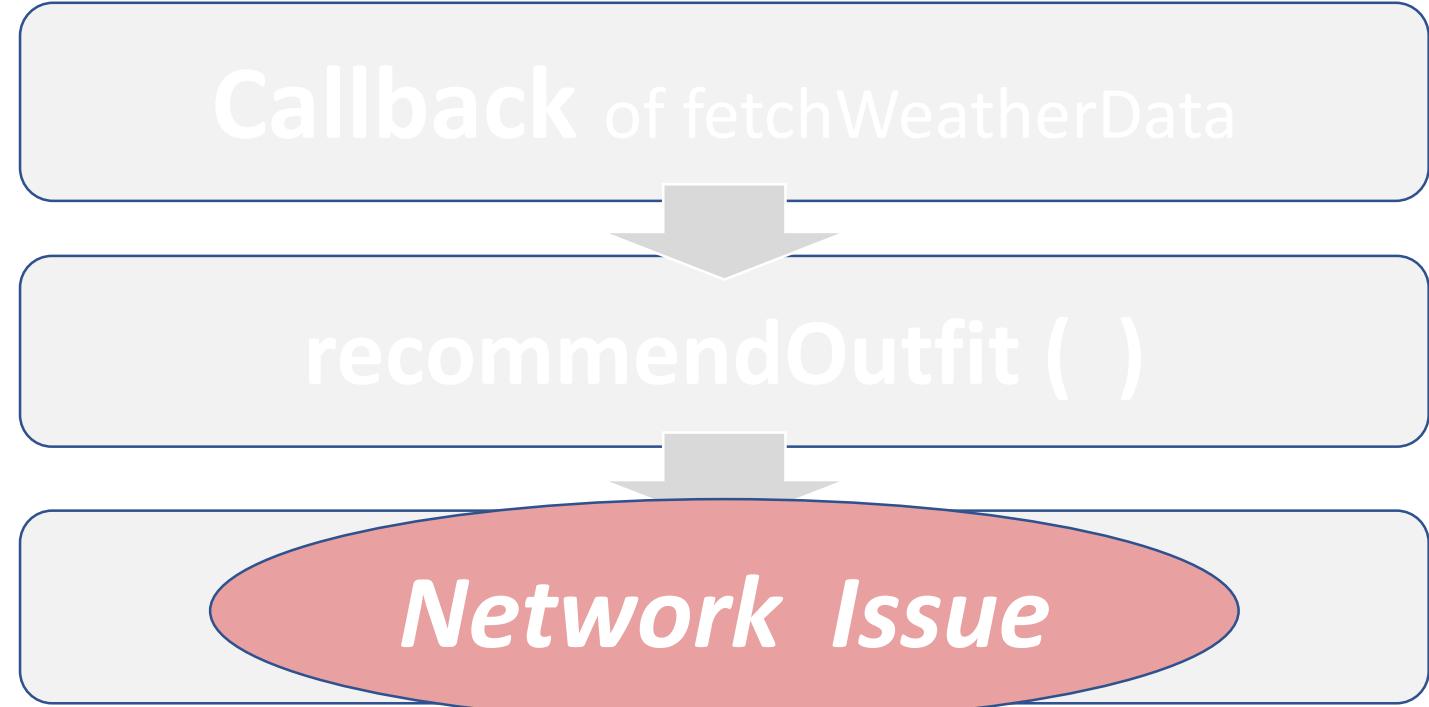
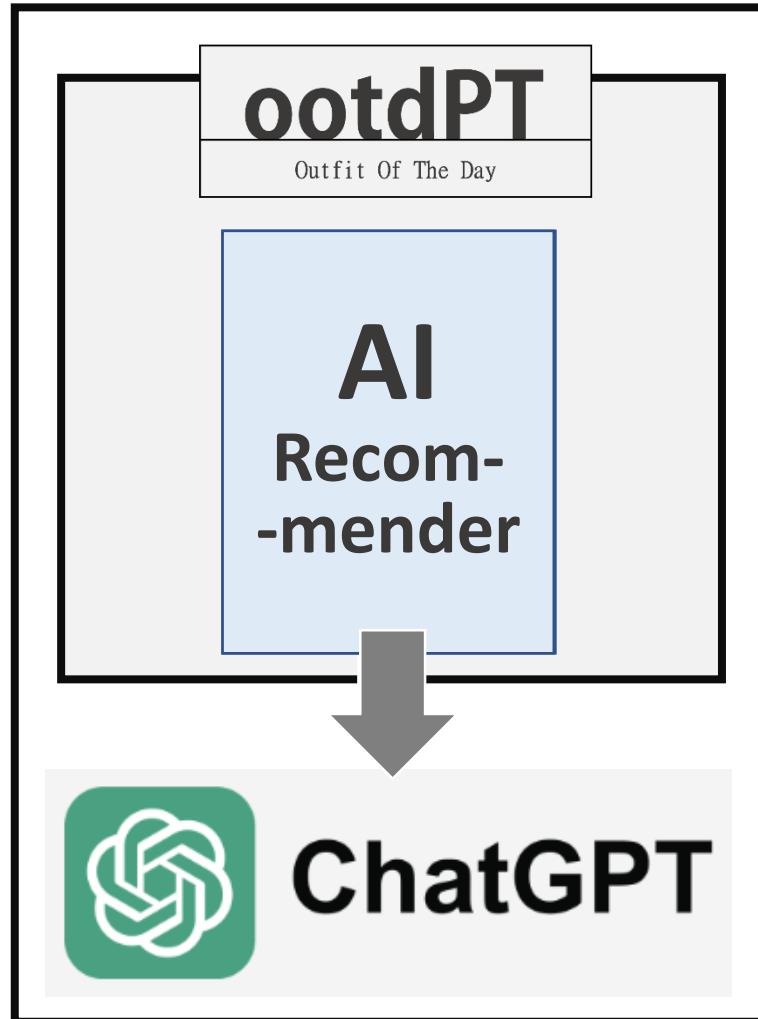


Callback of `fetchWeatherData`

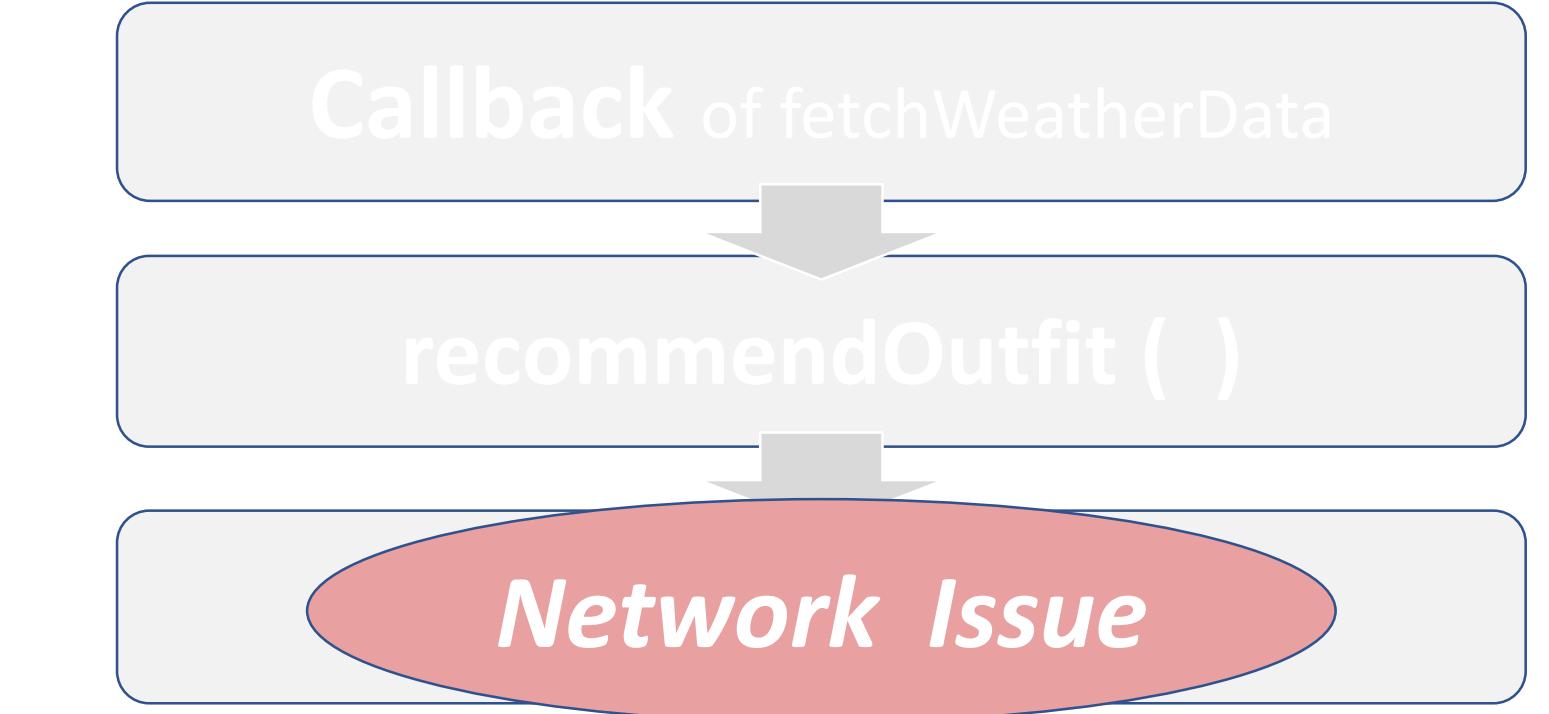
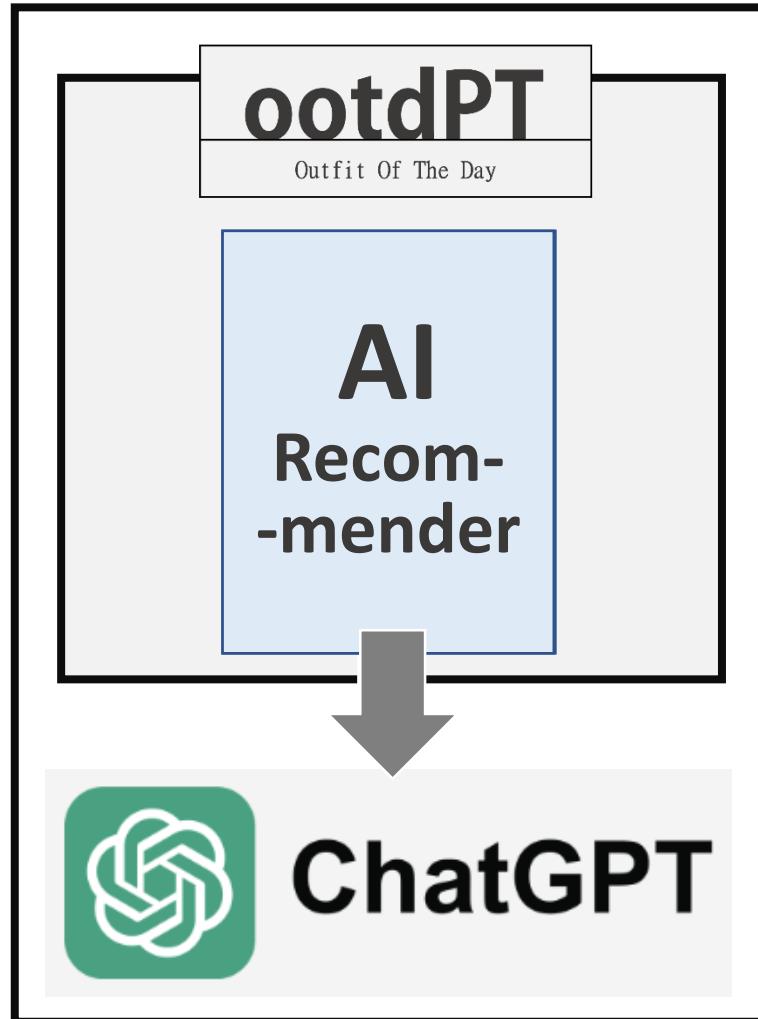
`recommendOutfit()`

`call chatGPT API`

Outfit Recommendation

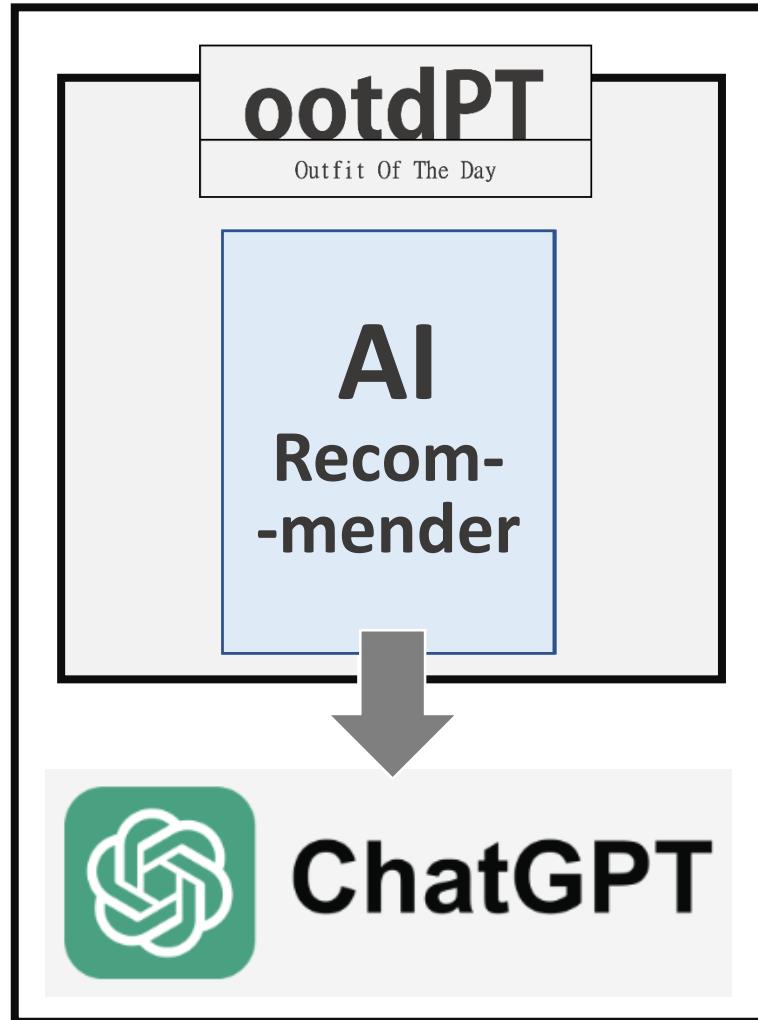


Outfit Recommendation



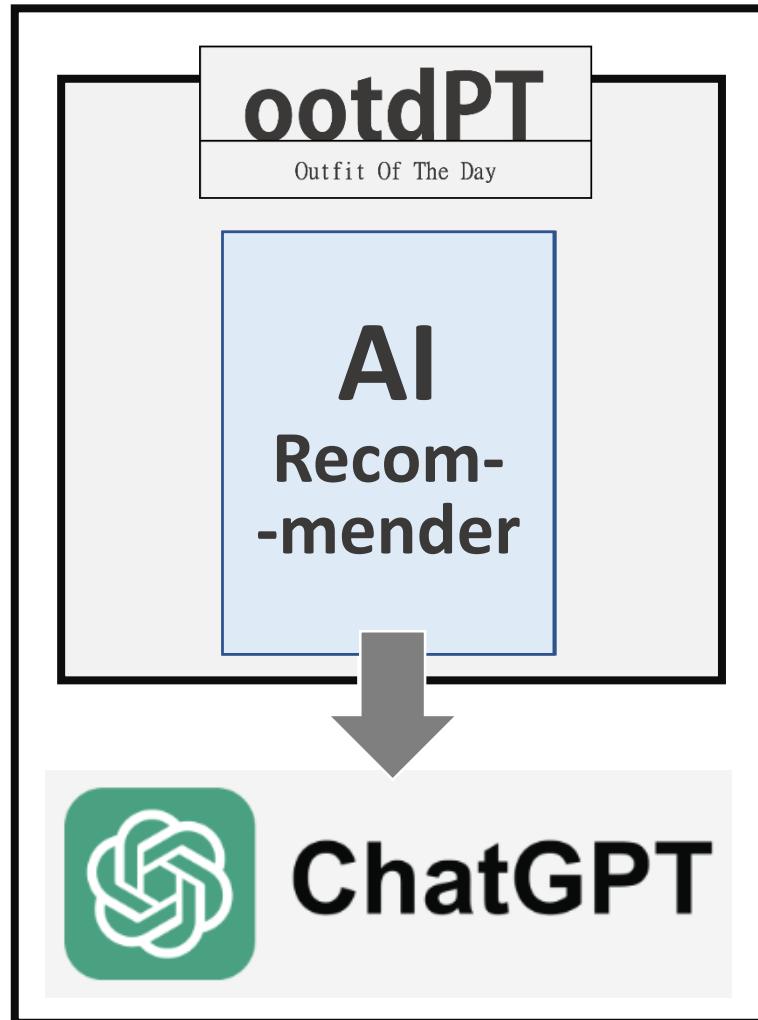
Android version4 **doesn't support**
a network connection to chatGPT API

Outfit Recommendation



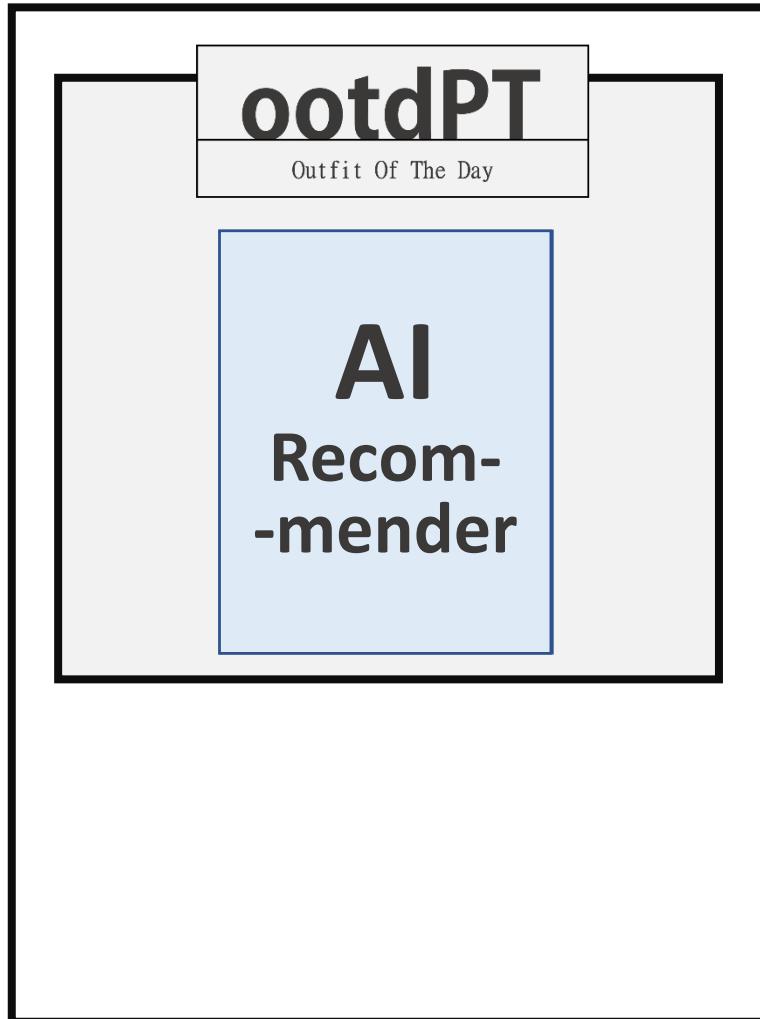
- Option1) In-device AI-like Recommendation Engine
- Option2) Intermediate Server Approach (Serial Port)
- Option3) Device Android Version Upgrade

Outfit Recommendation



- Option1) In-device chatGPT-like Recommendation Engine
- Option2) Intermediate Server Approach (Serial Port)
- Option3) Device Android Version Upgrade

Outfit Recommendation – Option1

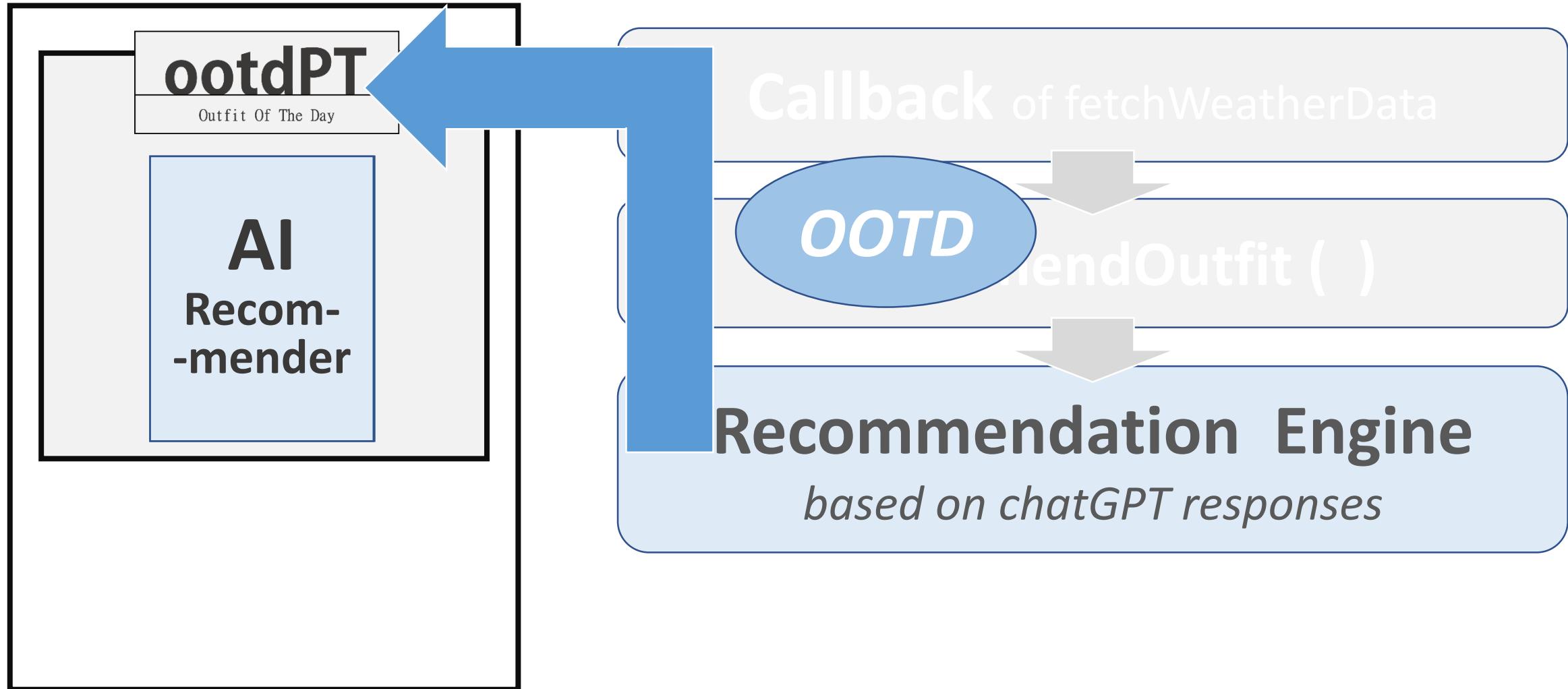


Callback of `fetchWeatherData`

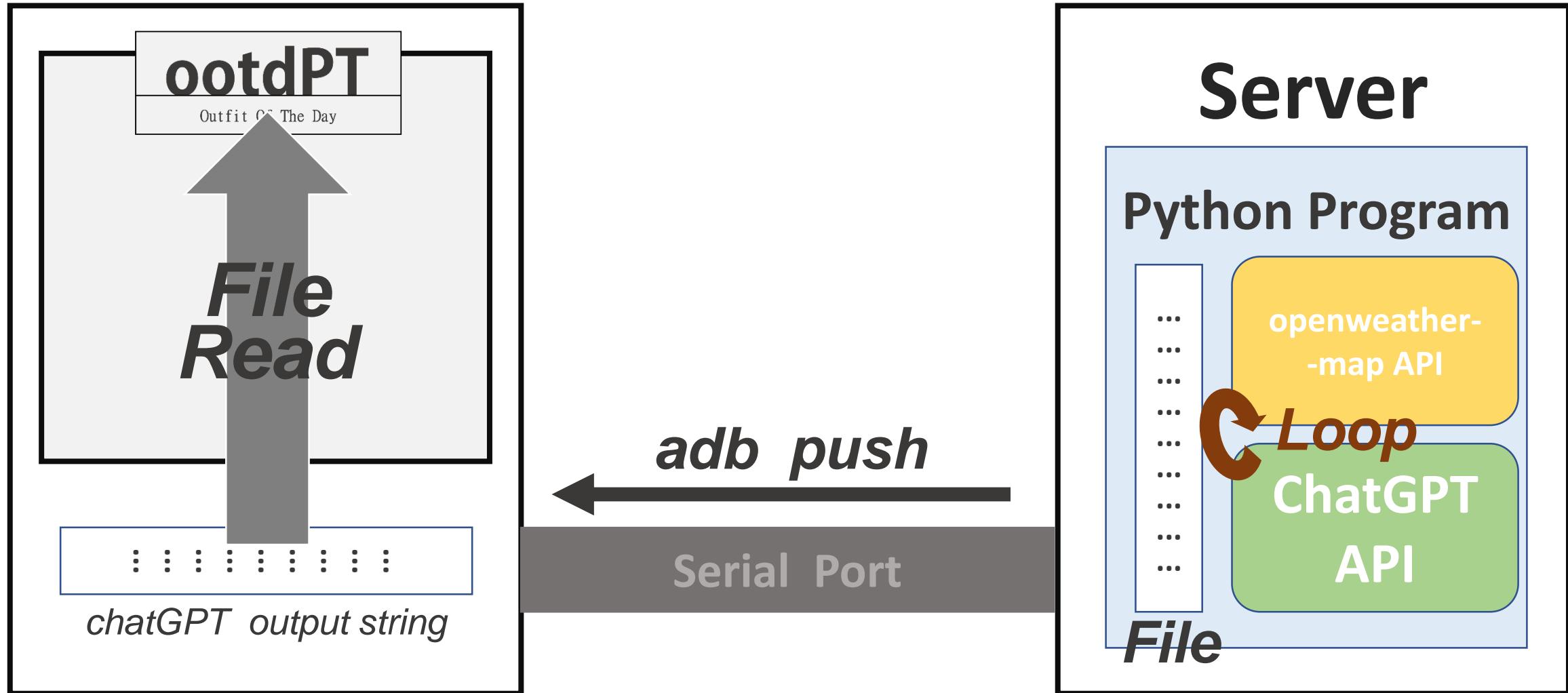
`recommendOutfit()`

Recommendation Engine
based on chatGPT responses

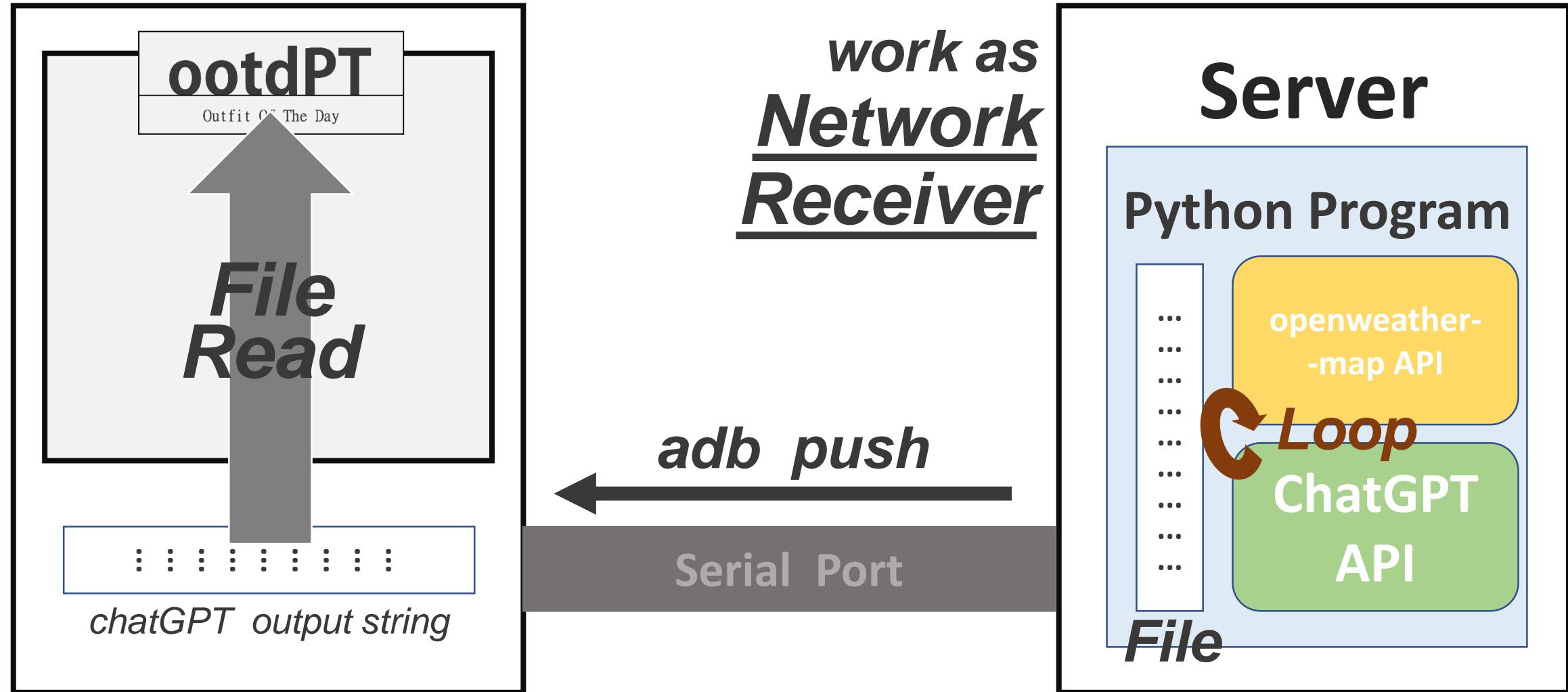
Outfit Recommendation – Option1



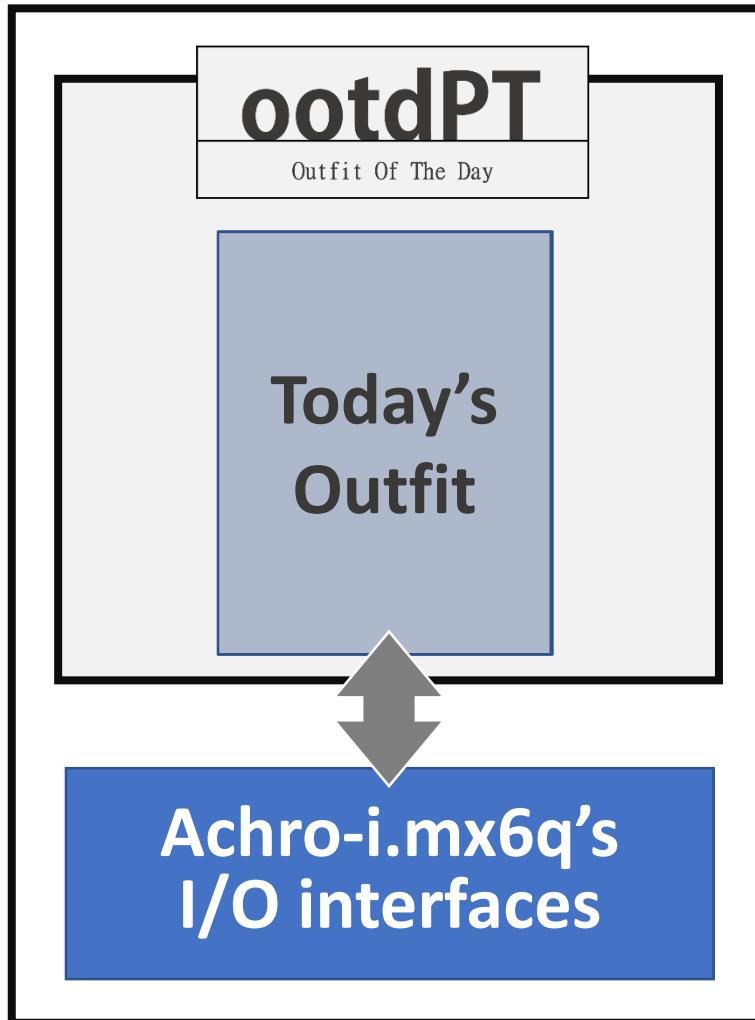
Outfit Recommendation – Option2



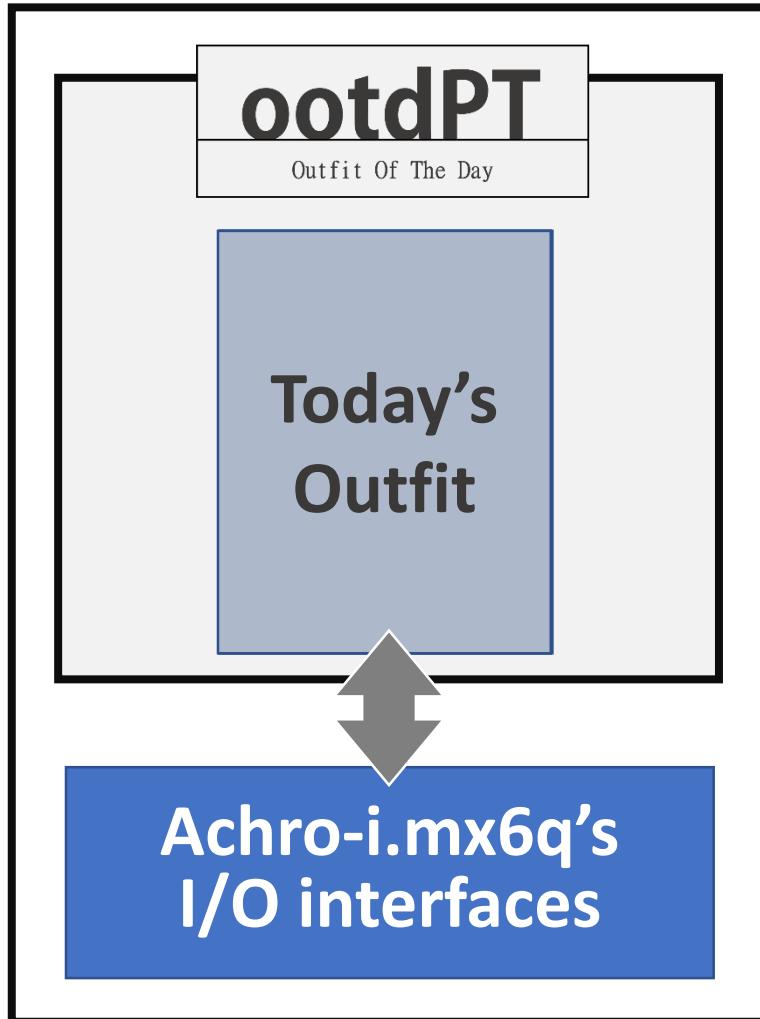
Outfit Recommendation – Option2



I/O with Device

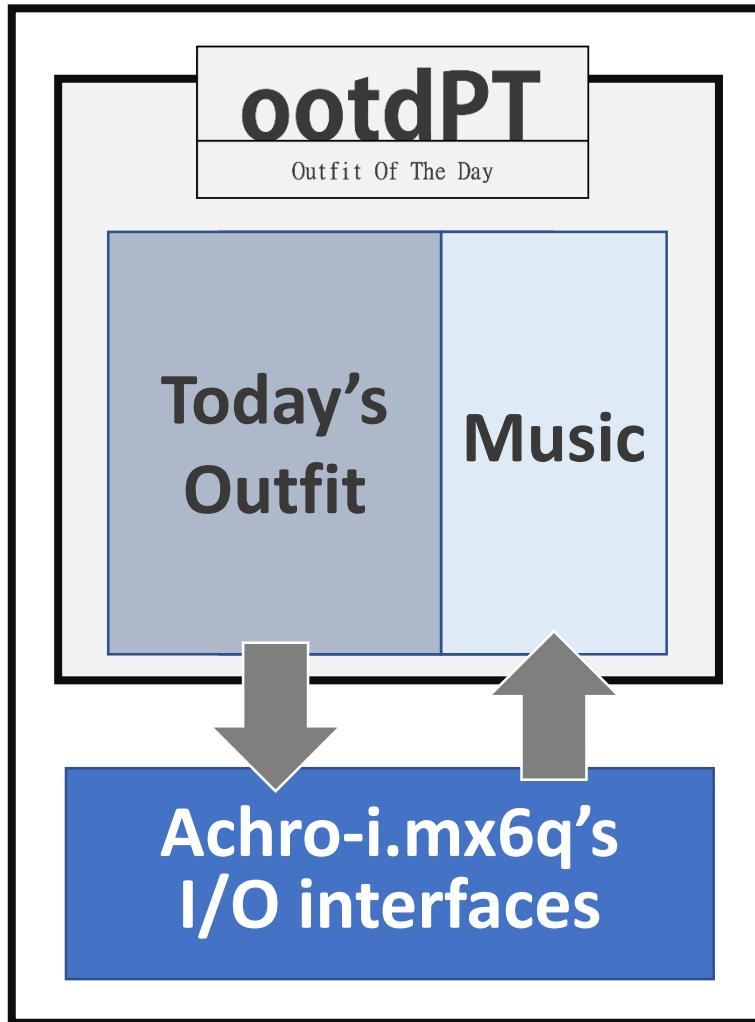


I/O with Device



Additional Feature
Background Music
(since it targets 'home intercom')

I/O with Device

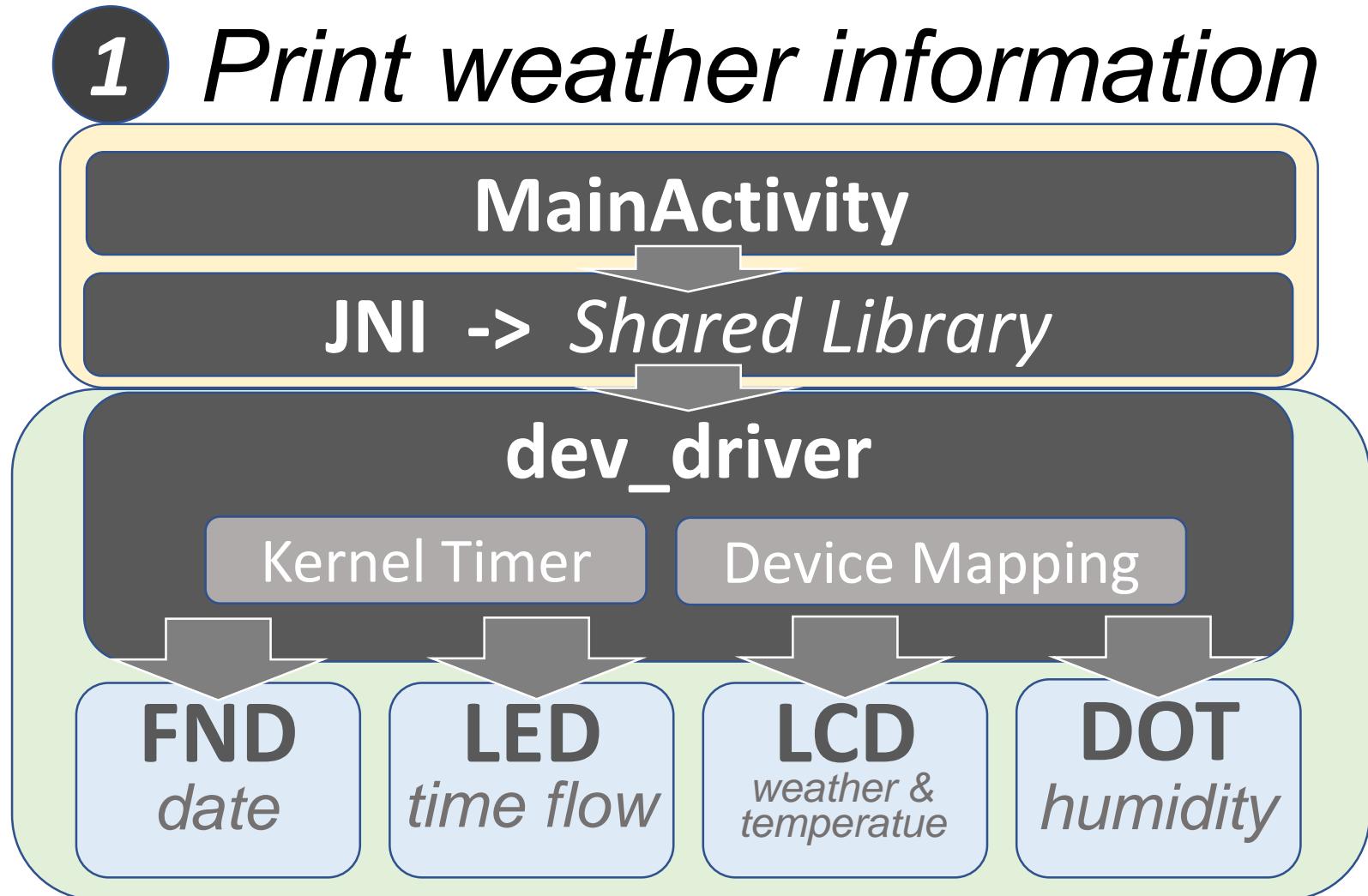
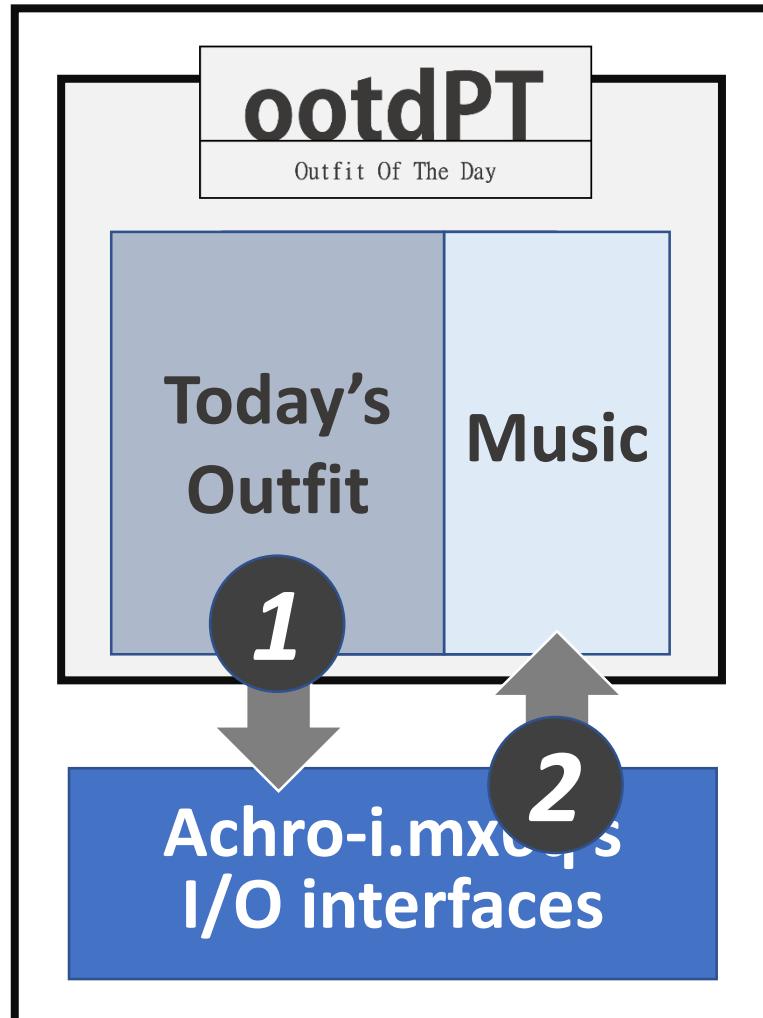


Additional Feature
Background Music
(since it targets 'home intercom')

I/O with Device

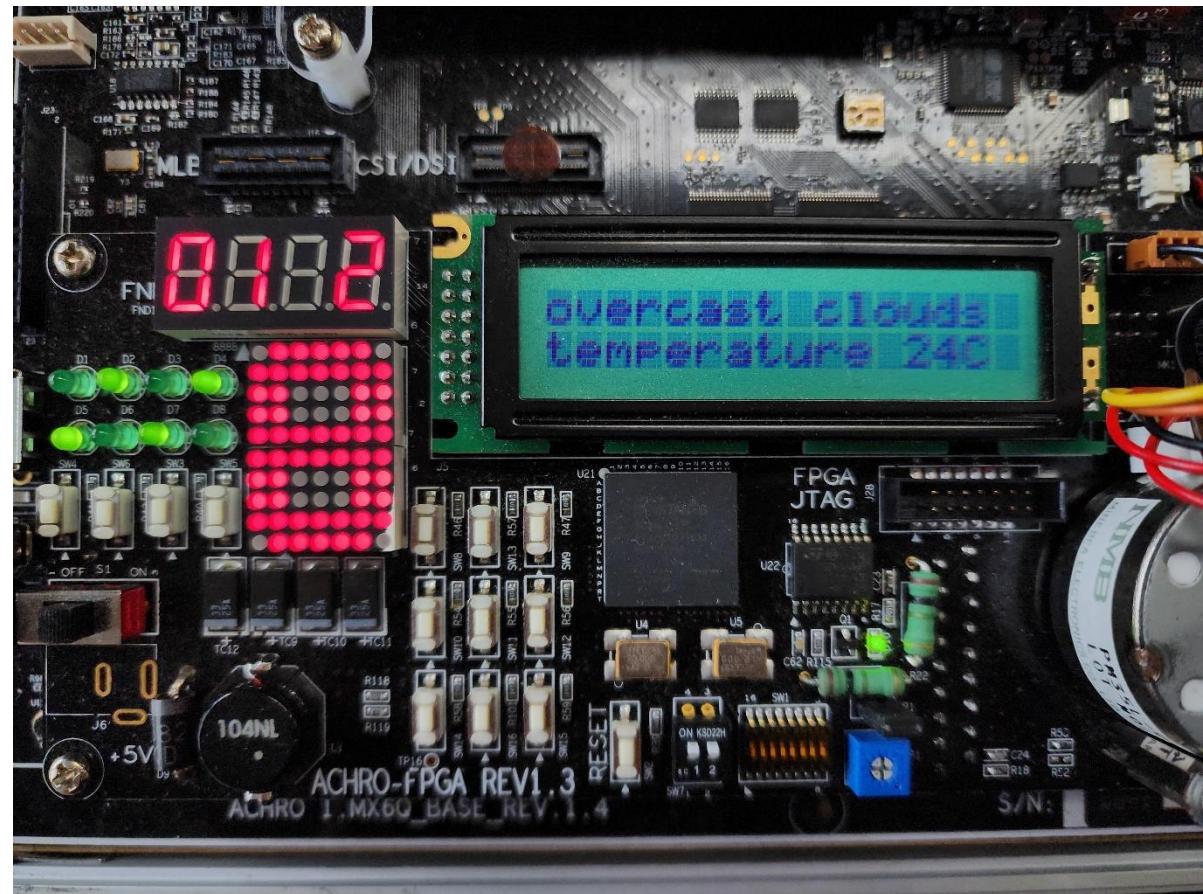
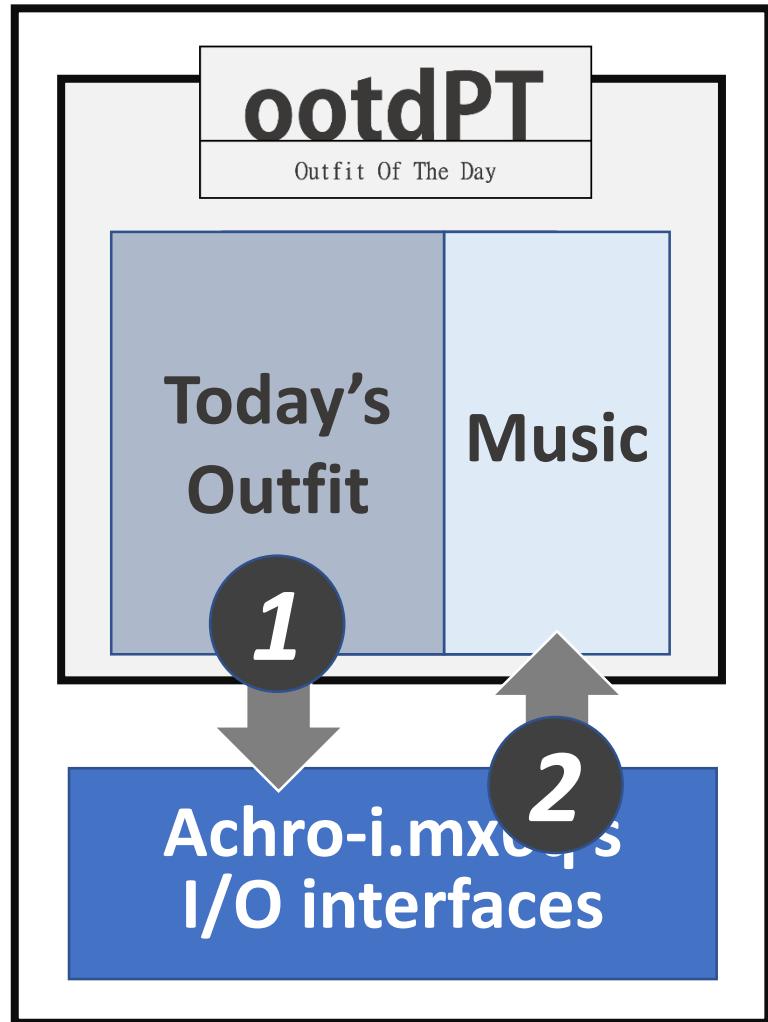
1

Print weather information



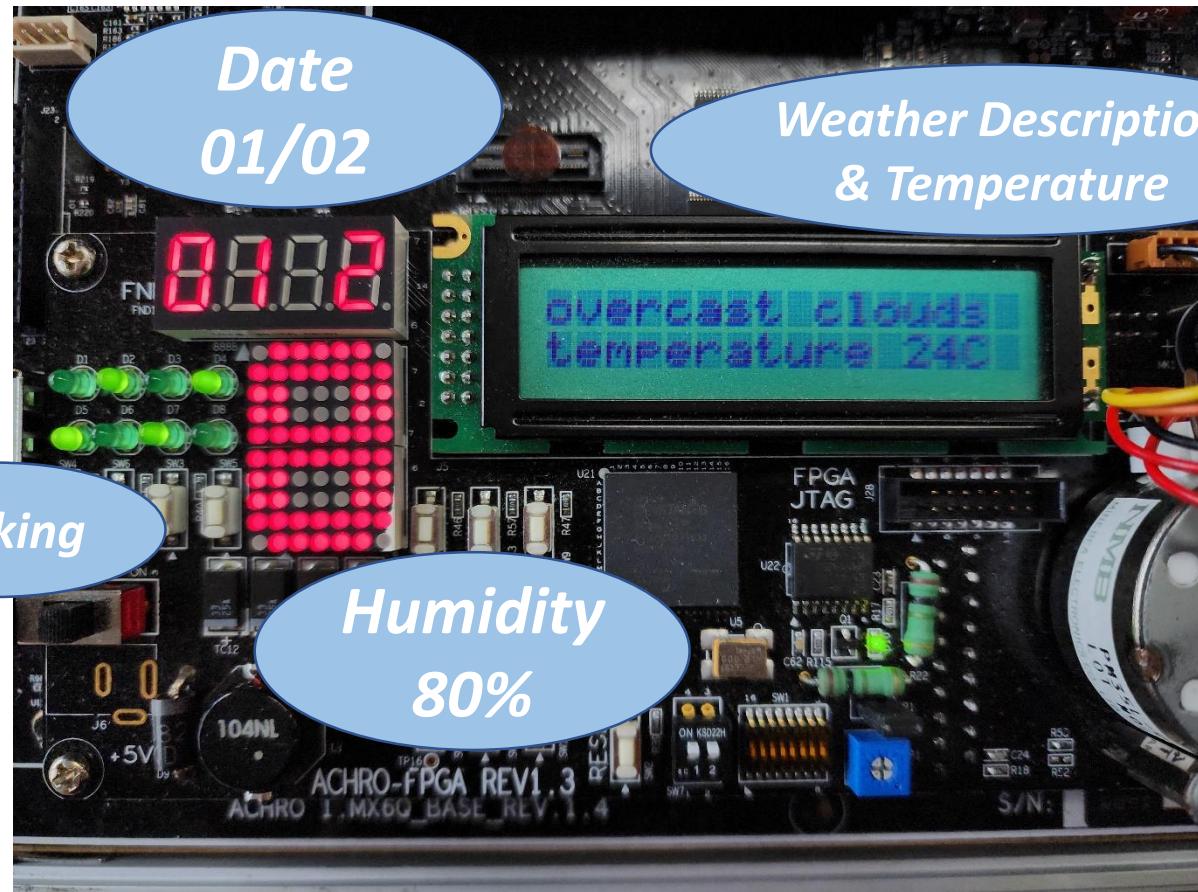
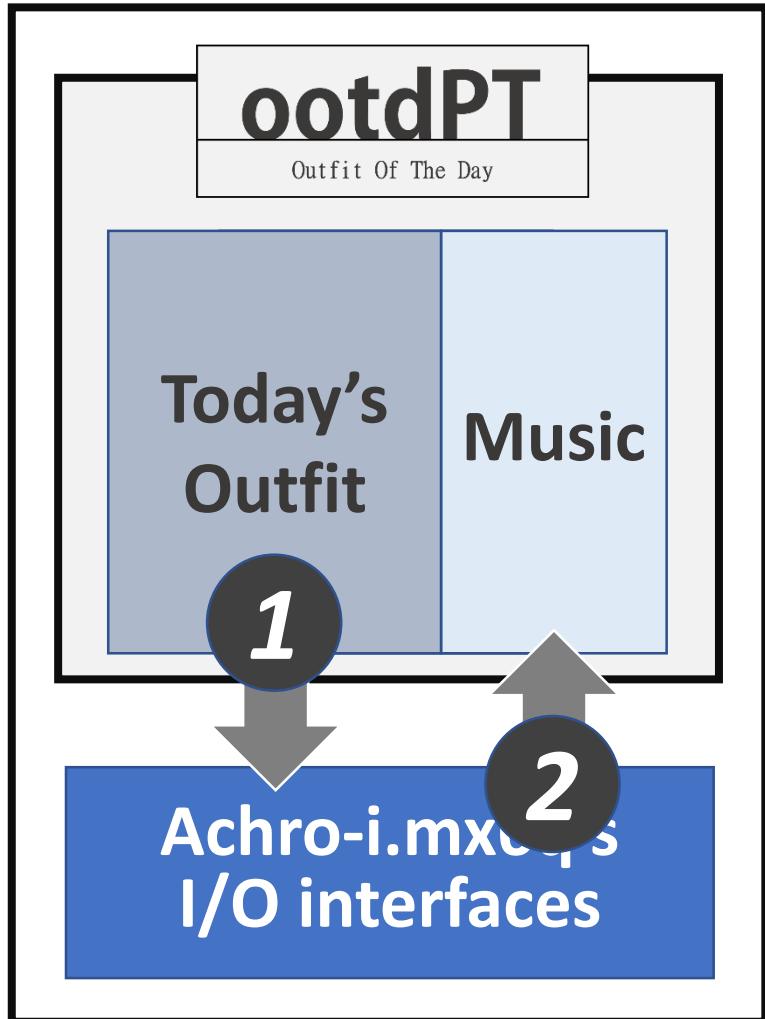
I/O with Device

1 Print weather information



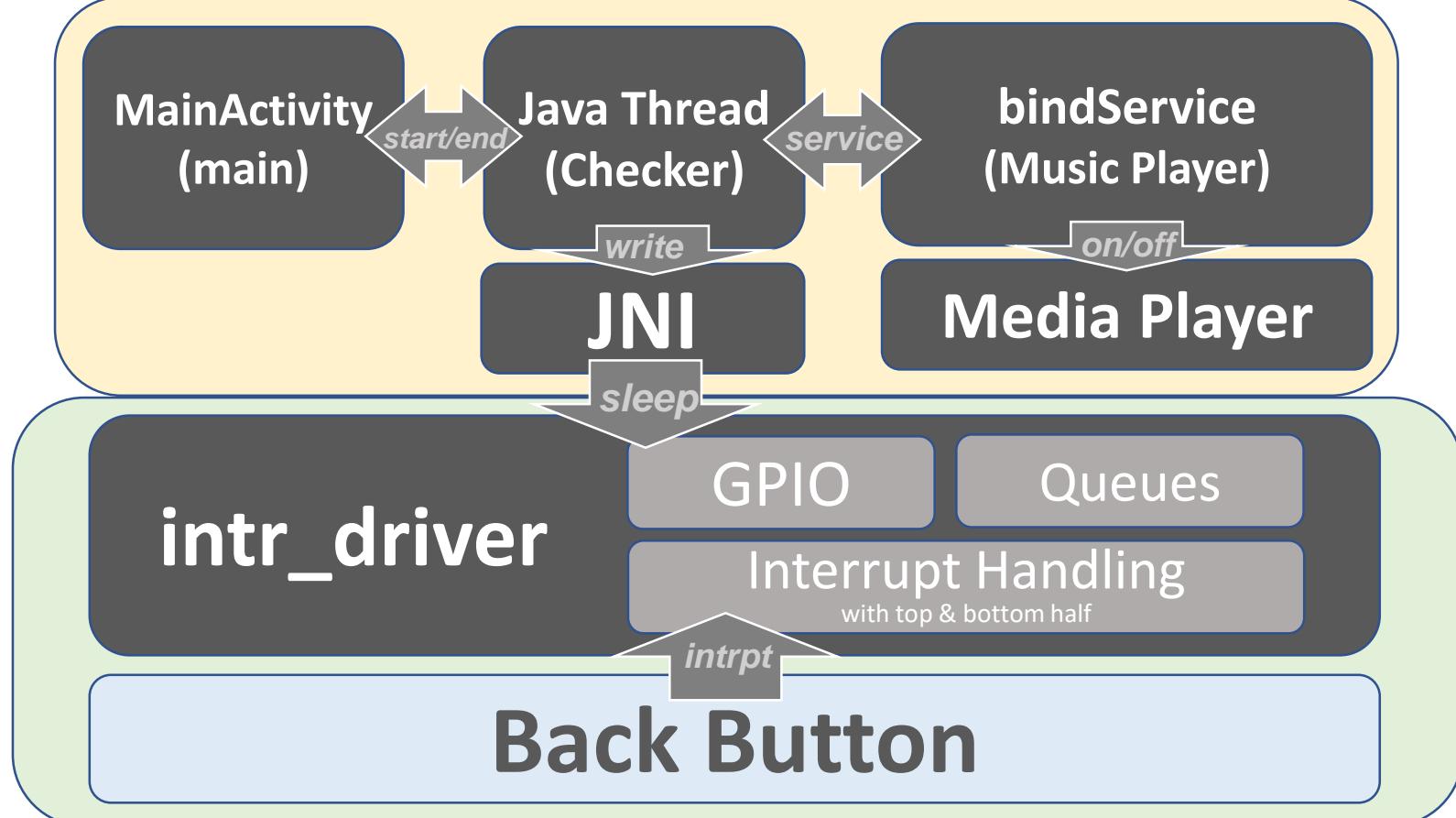
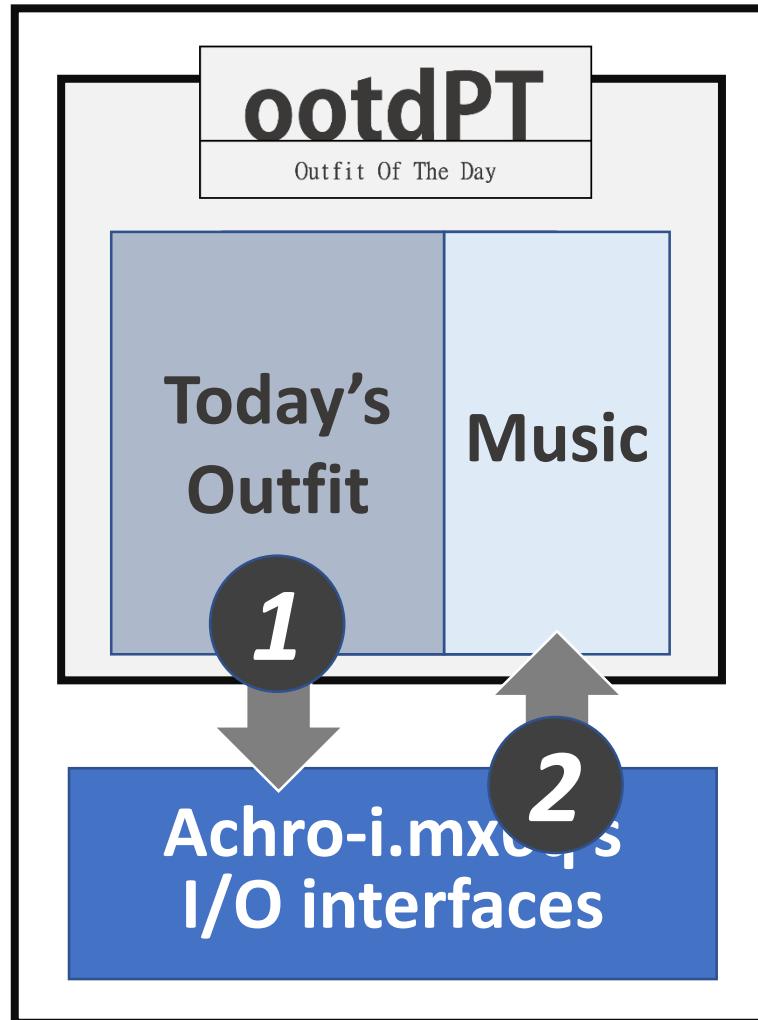
I/O with Device

1

Print weather information

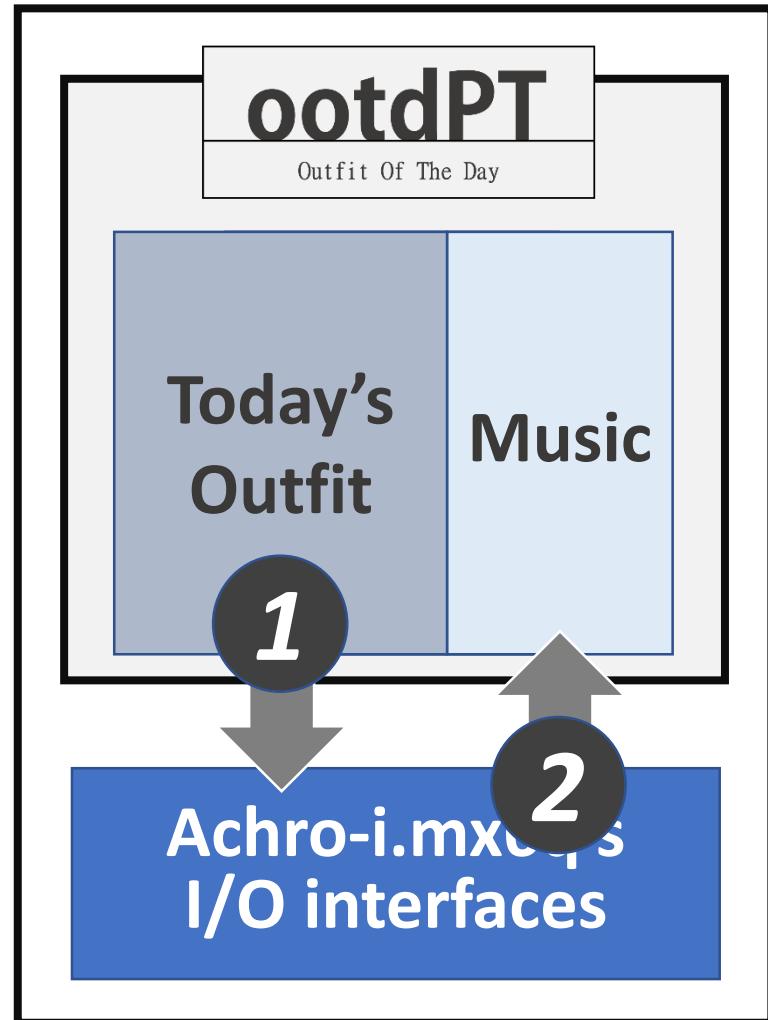
I/O with Device

2 *Background Music Playing*



I/O with Device

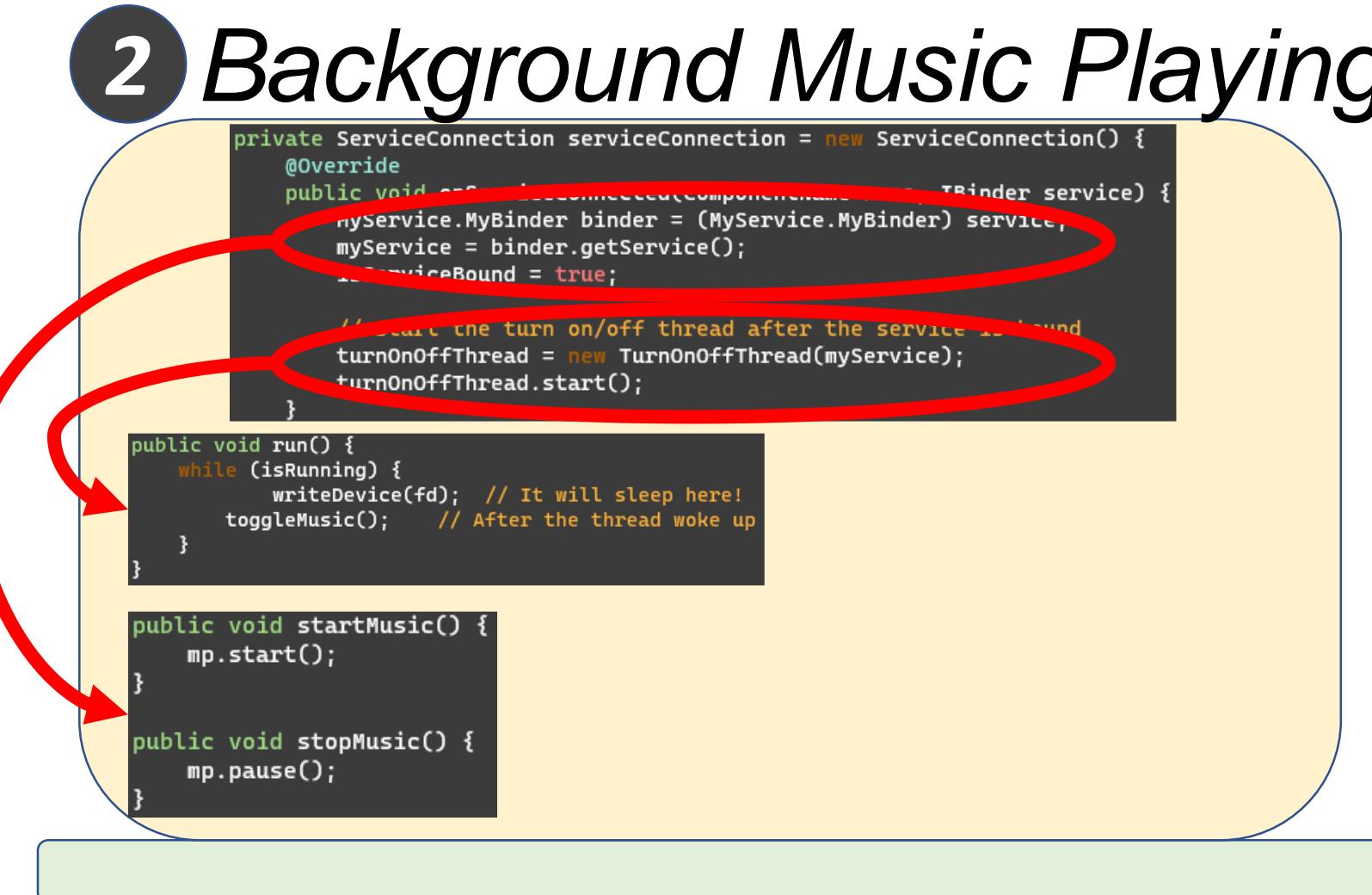
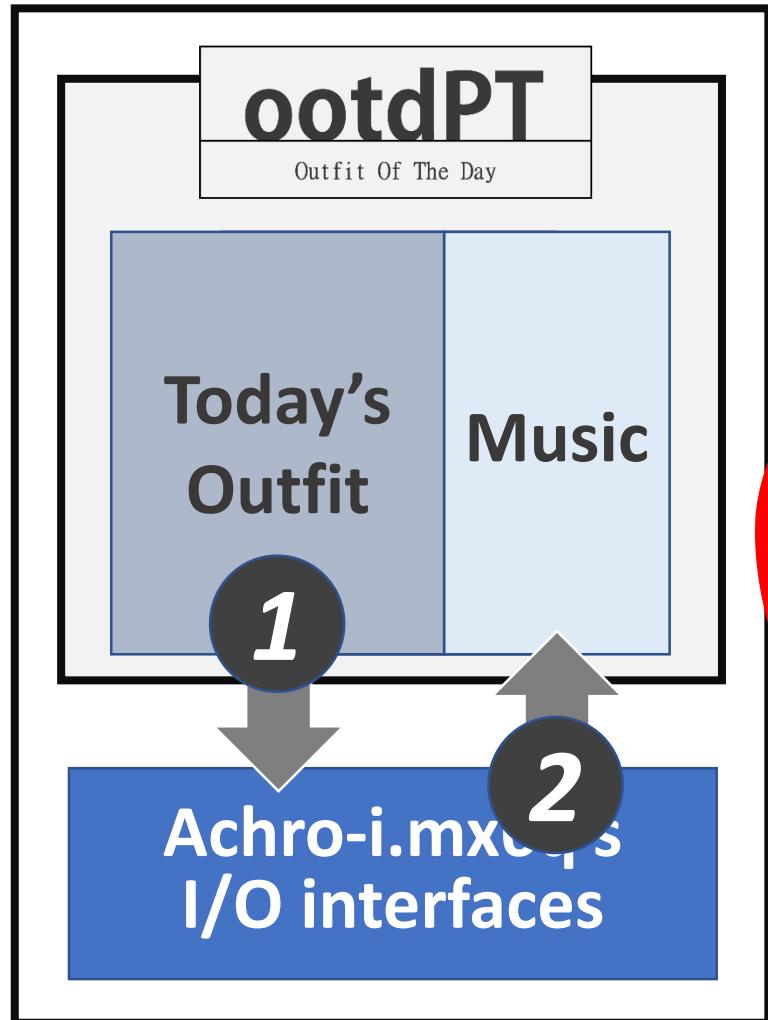
2 Background Music Playing



```
private ServiceConnection serviceConnection = new ServiceConnection() {  
    @Override  
    public void onServiceConnected(ComponentName name, IBinder service) {  
        MyService.MyBinder binder = (MyService.MyBinder) service;  
        myService = binder.getService();  
        isServiceBound = true;  
  
        // Start the turn on/off thread after the service is bound  
        turnOnOffThread = new TurnOnOffThread(myService);  
        turnOnOffThread.start();  
    }  
}
```

I/O with Device

2 Background Music Playing



The code snippet shows a ServiceConnection implementation for a music service:

```
private ServiceConnection serviceConnection = new ServiceConnection() {
    @Override
    public void onServiceConnected(ComponentName componentName, IBinder binder) {
        myService.MyBinder binder = (MyService.MyBinder) binder;
        myService = binder.getService();
        isServiceBound = true;

        // Start the turn on/off thread after the service is bound
        turnOnOffThread = new TurnOnOffThread(myService);
        turnOnOffThread.start();
    }

    public void run() {
        while (isRunning) {
            writeDevice(fd); // It will sleep here!
            toggleMusic(); // After the thread woke up
        }
    }

    public void startMusic() {
        mp.start();
    }

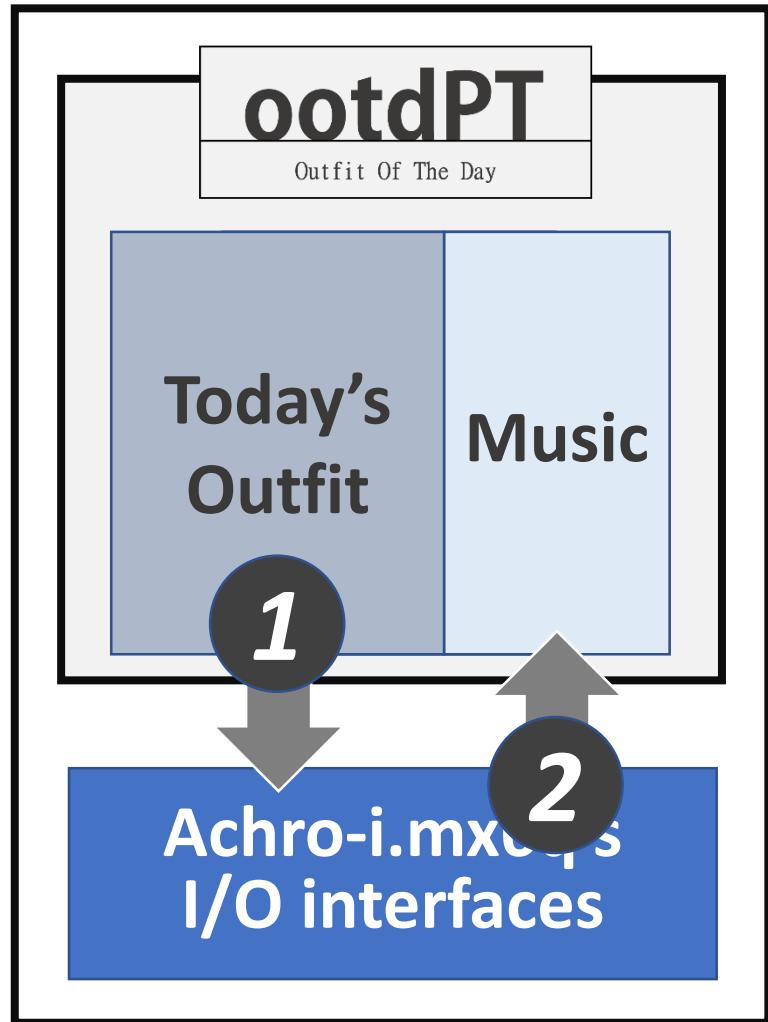
    public void stopMusic() {
        mp.pause();
    }
}
```

Red arrows highlight the following sections of the code:

- A red circle highlights the `onServiceConnected` method, which starts the `turnOnOffThread`.
- Two red arrows point to the `run` method, specifically to the `while` loop where it sleeps and then toggles the music.
- Two red arrows point to the `startMusic` and `stopMusic` methods.

I/O with Device

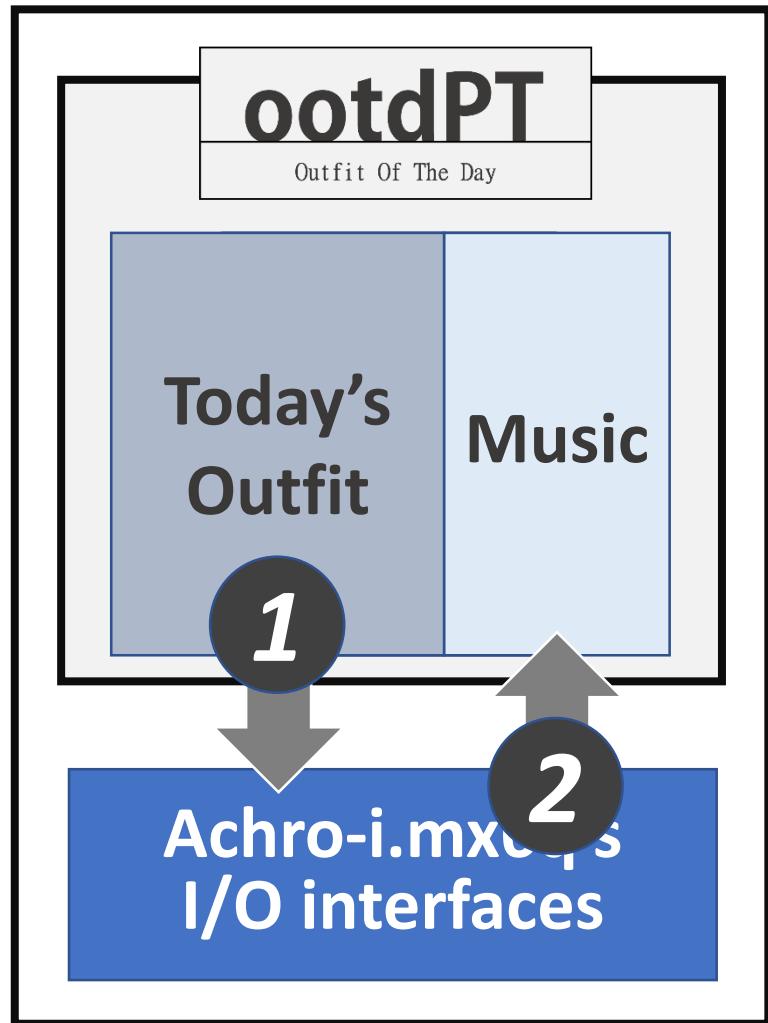
2 Background Music Playing



```
private ServiceConnection serviceConnection = new ServiceConnection() {  
    @Override  
    public void onServiceConnected(ComponentName name, IBinder service) {  
        MyService.MyBinder binder = (MyService.MyBinder) service;  
        myService = binder.getService();  
        isServiceBound = true;  
  
        // Start the turn on/off thread after the service is bound  
        turnOnOffThread = new TurnOnOffThread(myService);  
        turnOnOffThread.start();  
    }  
  
    public void run() {  
        while (true) {  
            writeDevice(fd); // It will sleep here!  
            toggleButtons(); // After the thread woke up  
        }  
    }  
  
    public void startMusic() {  
        mp.start();  
    }  
  
    public void stopMusic() {  
        mp.pause();  
    }  
  
/* write() Handler for this driver */  
static int intr_driver_write(  
    struct file *file, const char __user *buf, size_t count, loff_t *f_pos  
) {  
    printk("[INTR_DRIVER] %s called\n", __func__);  
  
    /* Put the user thread on a sleep */  
    printk("[INTR_DRIVER] Make user thread(java thread) sleep\n");  
    interruptible_sleep_on(&intrq);  
  
    return 0;  
}
```

I/O with Device

2 Background Music Playing



```
private ServiceConnection serviceConnection = new ServiceConnection() {  
    @Override  
    public void onServiceConnected(ComponentName name, IBinder service) {  
        MyService.MyBinder binder = (MyService.MyBinder) service;  
        myService = binder.getService();  
        isServiceBound = true;  
  
        // Start the turn on/off thread after the service is bound  
        turnOnOffThread = new TurnOnOffThread(myService);  
        turnOnOffThread.start();  
    }  
}
```

```
public void run() {  
    while (isRunning) {  
        writeDevice(fd); // It will sleep here!  
        toggleMusic(); // After the thread woke up  
    }  
}
```

```
public void startMusic() {  
    mp.start();  
}
```

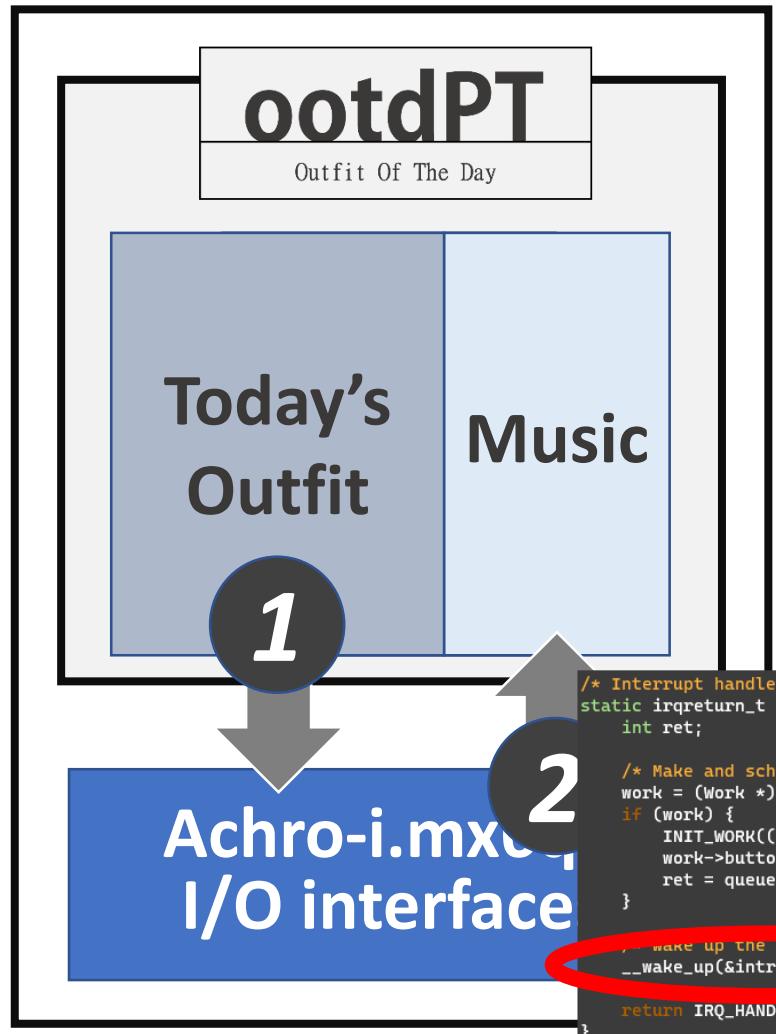
```
public void stopMusic() {  
    mp.pause();  
}
```

```
/* write() Handler for this driver */  
static int intr_driver_write(  
    struct file *file, const char __user *buf, size_t count, loff_t *f_pos  
) {  
    printk("[INTR_DRIVER] %s called\n", __func__);  
  
    /* Put the user thread on a sleep */  
    if (intrrq->rq->rq_flags & RQF_INTERRUPTIBLE)  
        interruptible_sleep_on(&intrrq);  
  
    return 0;  
}
```

Sleep

I/O with Device

2 Background Music Playing



```
private ServiceConnection serviceConnection = new ServiceConnection() {
    @Override
    public void onServiceConnected(ComponentName name, IBinder service) {
        MyService.MyBinder binder = (MyService.MyBinder) service;
        myService = binder.getService();
        isServiceBound = true;

        // Start the turn on/off thread after the service is bound
        turnOnOffThread = new TurnOnOffThread(myService);
        turnOnOffThread.start();
    }

    public void run() {
        while (isRunning) {
            writeDevice(fd); // It will sleep here!
            toggleMusic(); // After the thread woke up
        }
    }
}
```

```
/* Interrupt handler for 'back' button, it will trigger the sleeping java thread */
static irqreturn_t _back_button_handler(int irq, void* dev_id) {
    int ret;

    /* Make and schedule the bottom half (printks) */
    work = (Work *)kmalloc(sizeof(Work), GFP_KERNEL);
    if (work) {
        INIT_WORK((struct work_struct *)work, work_function);
        work->button = BACK;
        ret = queue_work(workq, (struct work_struct *)work);
    }

    /* wake up the sleeping thread (which is a java thread in here) */
    __wake_up(&intrq, 1, 1, NULL);

    return IRQ_HANDLED;
}
```

Wake up

if 'back' clicked and interrupt generated

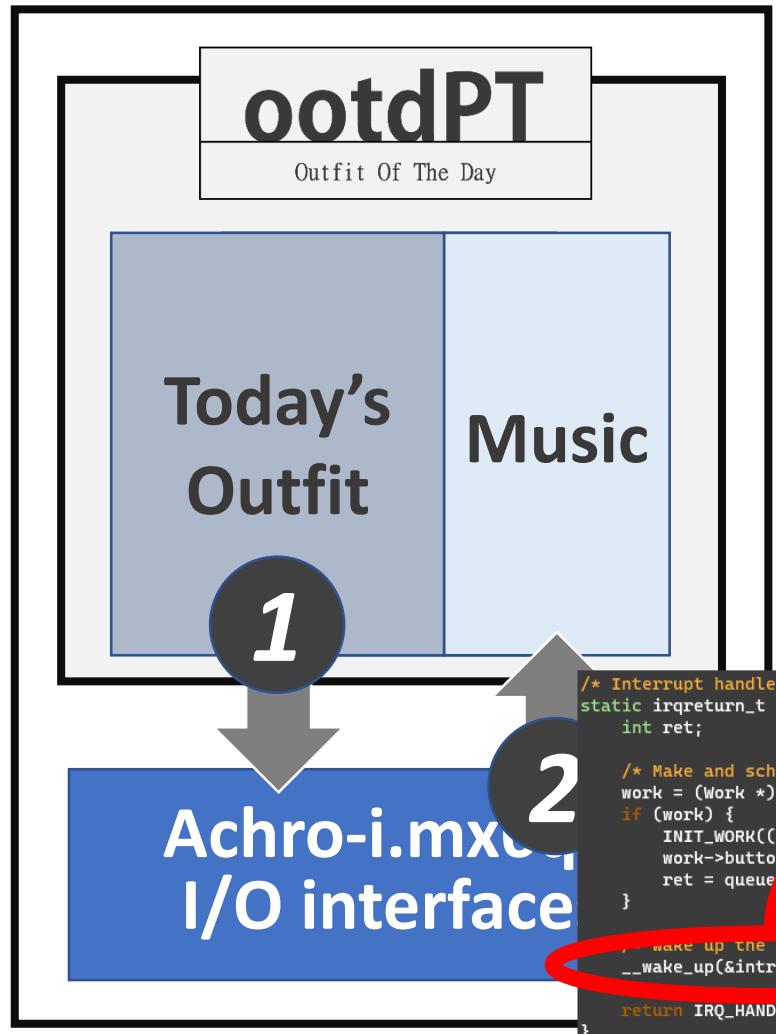
```
/* write() Handler for this driver */
static int intr_driver_write(
    struct file *file, const char __user *buf, size_t count, loff_t *f_pos
) {
    printk("[INTR_DRIVER] %s called\n", __func__);

    /* Put the user thread on a sleep */
    printk("[INTR_DRIVER] Make user thread(java thread) sleep\n");
    interruptible_sleep_on(&intrq);

    return 0;
}
```

I/O with Device

2 Background Music Playing



```

private ServiceConnection serviceConnection = new ServiceConnection() {
    @Override
    public void onServiceConnected(ComponentName name, IBinder service) {
        MyService.MyBinder binder = (MyService.MyBinder) service;
        myService = binder.getService();
        isServiceBound = true;

        // Start the turn on/off thread after the service is bound
        turnOnOffThread = new TurnOnOffThread(myService);
        turnOnOffThread.start();
    }

    public void run() {
        while (isRunning) {
            writeToDevice("turnOn"); // It will sleep here!
            toggleMusic(); // After this thread woke up
        }
    }
}

private void toggleMusic() {
    isMusicOn = !isMusicOn;
    // Control the music playback
    if (isMusicOn) {
        myService.startMusic();
        Log.d("TurnOnOffThread", "Music turned on");
    } else {
        myService.stopMusic();
        Log.d("TurnOnOffThread", "Music turned off");
    }
}

/* write() Handler for this driver */
static int intr_driver_write(
    struct file *file, const char __user *buf, size_t count, loff_t *f_pos
) {
    printk("[INTR_DRIVER] %s called\n", __func__);

    /* Put the user thread on a sleep */
    printk("[INTR_DRIVER] Make user thread(java thread) sleep\n");
    interruptible_sleep_on(&intrq);

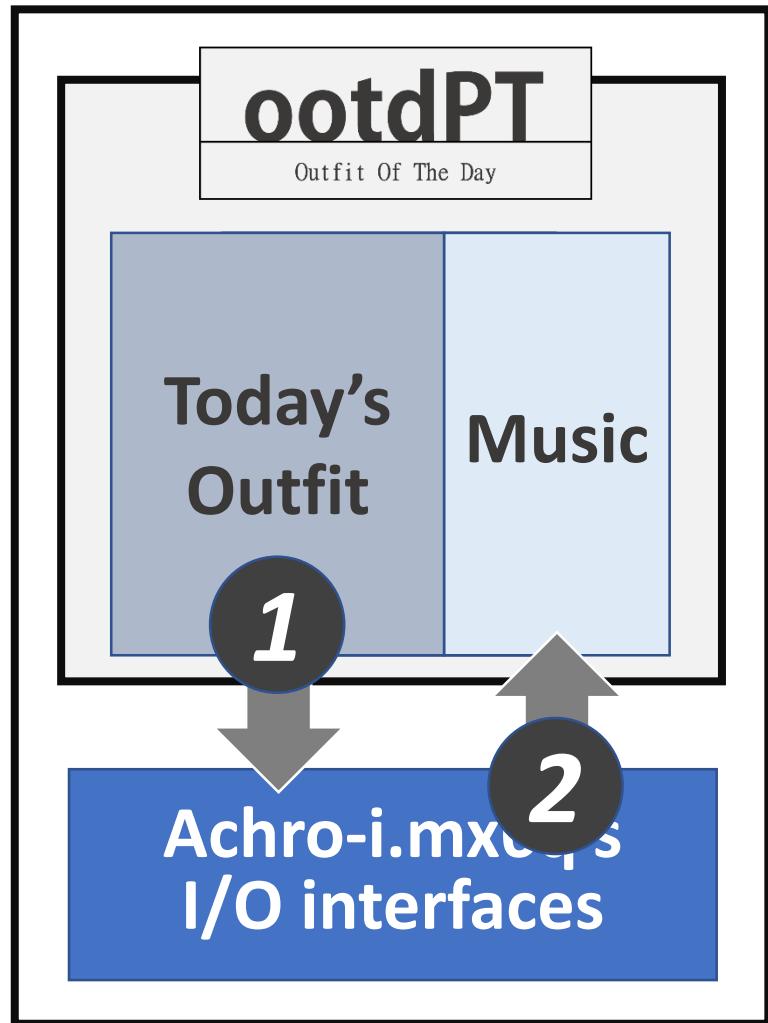
    return 0;
}

```

The code shows the implementation of a service connection and a background thread for toggling music. A red arrow highlights the sleep operation in the `run()` method, and another red arrow highlights the wake-up operation in the `intr_driver_write()` handler.

I/O with Device

2 Background Music Playing



```
private ServiceConnection serviceConnection = new ServiceConnection() {  
    @Override  
    public void onServiceConnected(ComponentName name, IBinder service) {  
        MyService.MyBinder binder = (MyService.MyBinder) service;  
        myService = binder.getService();  
        isServiceBound = true;  
  
        // Start the turn on/off thread after the service is bound  
        turnOnOffThread = new TurnOnOffThread(myService);  
        turnOnOffThread.start();  
    }  
}
```

```
public void run() {  
    while (isRunning) {  
        writeDevice(fd); // It will sleep here!  
        toggleMusic(); // After the thread woke up  
    }  
}
```

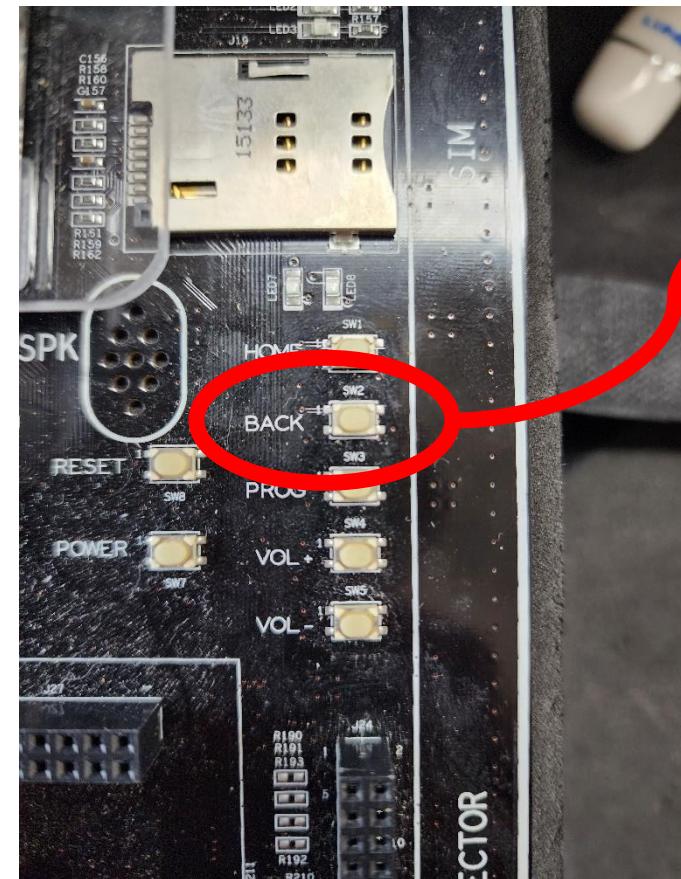
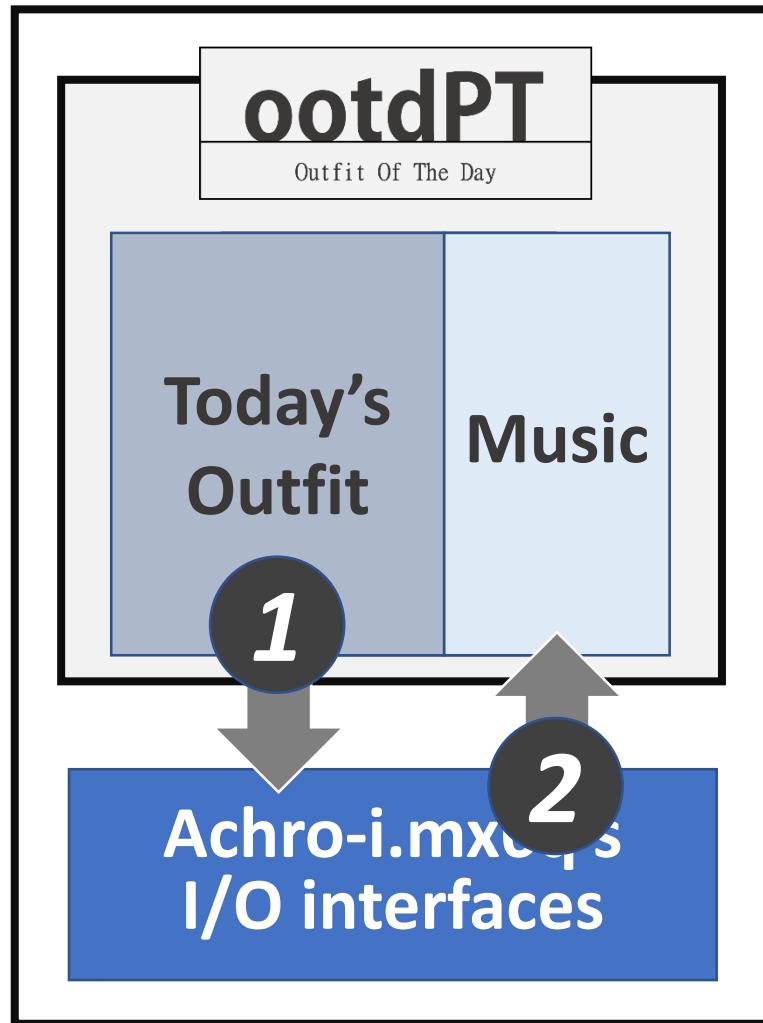
```
public void startMusic() {  
    mp.start();  
}  
  
public void stopMusic() {  
    mp.pause();  
}
```

```
private void toggleMusic() {  
    isMusicOn = !isMusicOn;  
    // Control the music playback  
    if (isMusicOn) {  
        myService.startMusic();  
        Log.d("TurnOnOffThread", "Music turned on");  
    } else {  
        myService.stopMusic();  
        Log.d("TurnOnOffThread", "Music turned off");  
    }  
}
```

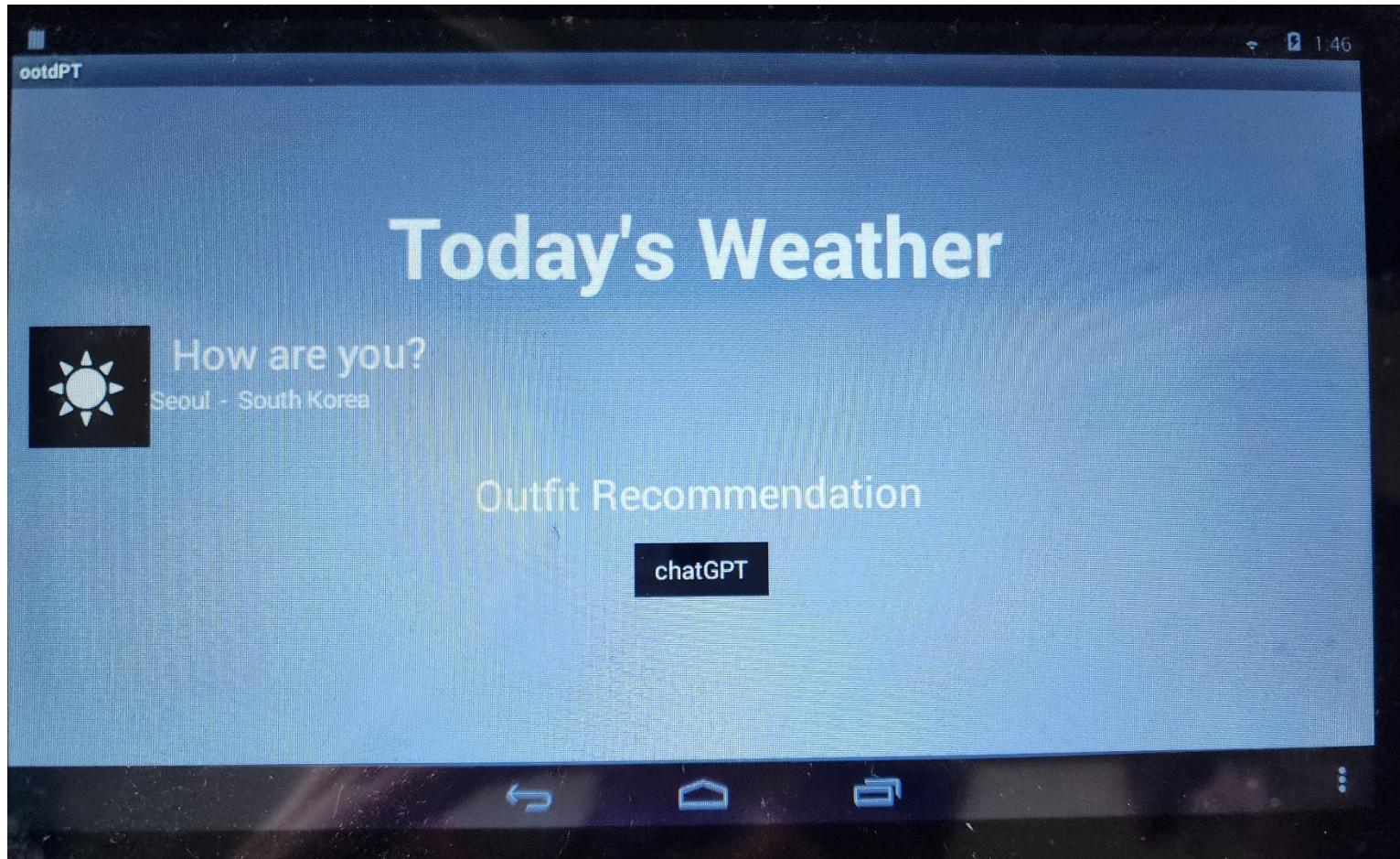
```
/*write() Handler for this driver */  
static int intr_driver_write(  
    struct file *file, const char __user *buf, size_t count, loff_t *f_pos  
) {  
    printk("[INTR_DRIVER] %s called\n", __func__);  
  
    /* Put the user thread on a sleep */  
    printk("[INTR_DRIVER] Make user thread(java thread) sleep\n");  
    interruptible_sleep_on(&intrq);  
  
    return 0;  
}
```

I/O with Device

2 Background Music Playing

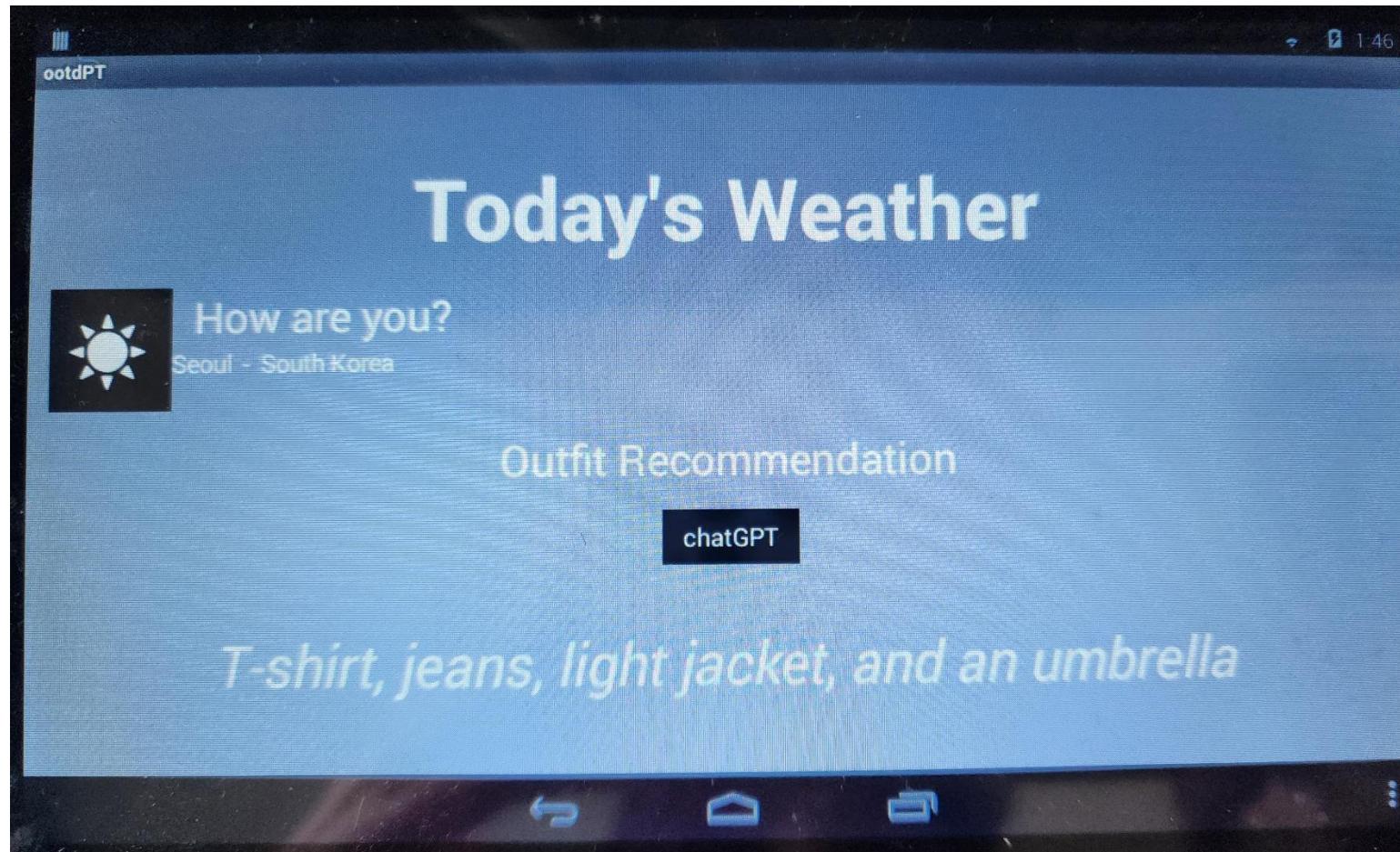


Result - ootdPT



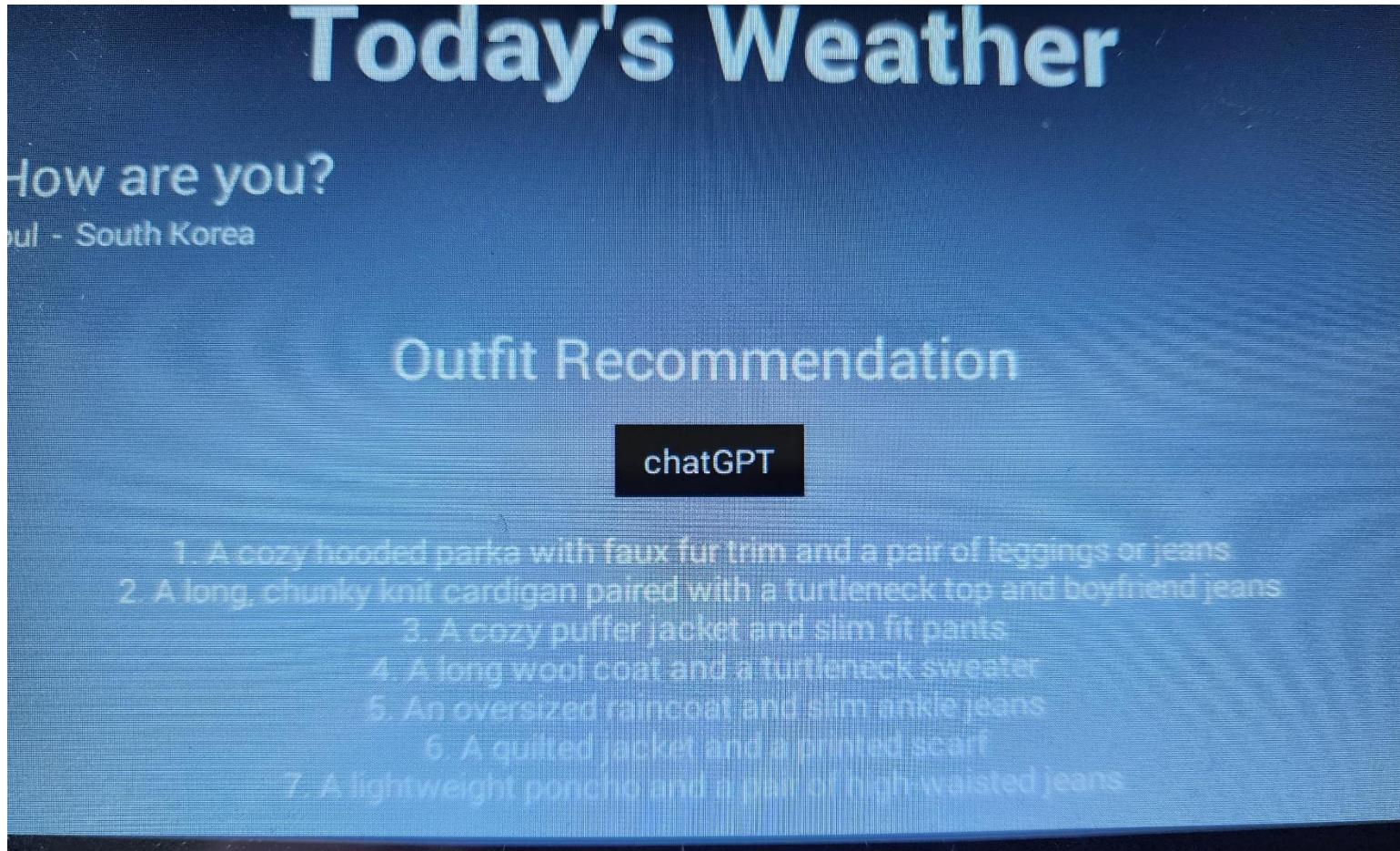
**1. Wait for
API callback**
(takes under 1sec)

Result - ootdPT



**2. In-device
Engine works**

Result - ootdPT



The image shows a screenshot of a mobile application interface. At the top, it says "Today's Weather" and "How are you? Seoul - South Korea". Below that, it says "Outfit Recommendation" and has a "chatGPT" button. A list of 7 outfit suggestions follows:

1. A cozy hooded parka with faux fur trim and a pair of leggings or jeans
2. A long, chunky knit cardigan paired with a turtleneck top and boyfriend jeans
3. A cozy puffer jacket and slim fit pants
4. A long wool coat and a turtleneck sweater
5. An oversized raincoat and slim ankle jeans
6. A quilted jacket and a printed scarf
7. A lightweight poncho and a pair of high-waisted jeans

3. Get *chatGPT* Response
(if you click the button)



Let's run it now!