ASAP: Automatic Status Adjustment APP for Firefox OS

Shih-Hao Huang and Chow-Sing Lin and*
Jian-Jhe Hong and Cheng-Han Wu and Yu-Chi Hsien

Dept. of Computer Science and Information Engineering, National University of Tainan, Taiwan

Abstract* — Most of applications (APP) for automatic status adjustment on mobile phone have been implemented for Android and IOS systems. In this work, we specifically develop the APP called ASAP to import such a functionality to the promising Firefox OS. Firefox OS has been developed by Mozilla in recent years, using web APIs to directly communicate with the core of the operating system and the underneath hardware of a mobile phone. In ASAP, the automatic status adjustment is achieved by system monitoring with scenario scripts. When the condition of a certain script is met, the corresponding pre-defined activities are executed to instantly alter the status of a mobile phone. There are six conditions and six activities defined in the scenario, and the satisfaction of one condition can trig multiple activities to execute. In addition to listening the activity events, we also use a timer to periodically scan the pre-defined scenario scripts to trig the activities. The experimental results show that the ASAP can successfully trig the corresponding activities when a pre-defined scenario is detected.

Index Terms — **Automatic Status Adjustment, Script, ASAP, Firefox OS, HTML5**

---

* Corresponding author:
mikelin@mail.nutn.edu.tw
DOI：10.6159/IJSE.2015.(5-3).03

ASAP: Firefox OS之裝置自動化狀態調整APP ASAP

黃詩豪、林朝興*、洪健哲、吳承翰、洗鈺淇
國立臺南大學 資訊工程學系

摘要

大部分的裝置自動化狀態調整應用程式，已經被實作於 Android 及 iOS 系統上。在本研究中，我們開發一款裝置自動化狀態調整 APP 將其命名為 ASAP，使其能夠在 Firefox OS 平台之手機上自動改變狀態。Firefox OS 是 Mozilla 近年來之發展，利用 Web APIs 與裝置作業系統及底層硬體溝通。在 ASAP 中，我們利用預先定義之情境活動及系統監控情境腳本，當條件符合時，執行對應的活動以達到自動狀態調整。情境中有六種條件與活動，當滿足所設定之其一條件時可執行多種活動。此外，藉由監看活動事件，使用計時器週期性掃描已設定之情境腳本，並且執行腳本中的活動。本研究實驗結果顯示出 ASAP 能夠成功判讀情境條件並且執行與其相符之情境活動。

關鍵字：自動化狀態調整、腳本、ASAP、Firefox OS、HTML5

壹、前言

現今生活中，幾乎人手一支智慧型手機，而隨著智慧型手機普及，使用上往往會在特定之場合發生尷尬的情形，例如：開會忘記將手機靜音、看電影手機突然鈴聲大作、開車使用手機造成的不安全等，我們可以藉由手機狀態的自動改變或者自定義情境腳本，在所符合的情境下，自動改變手機狀態。

傳統在 Android 上的 Tasker[24]應用程式，在 Google Play Store 上以 NT$99.9 價格販售，且下載次數高達 35,993，使用者評價中，五顆星評價佔總評價的比率為 77%，然後在 iOS 上的 Workflow[25]，超過 4.5 顆星之評價也有 873 筆，可見任務型管理之應用程式在 Android 及 iOS 系統上甚為普通。但在 Firefox OS 上，尚無此類型之應用程式[7]，因此本文提出裝置自動化狀態調整應用程式（Automatic Status Adjustment APP，簡稱 ASAP）之研究，主要目的在於將以往實作於其他行動作業系統上的任務管理型應用程式，以 Firefox OS 為架構，建立在此系統上第一個任務型管理應用程式之框架，然後利用條件與活動之組合建立情境腳本，並且能依照使用者設定之情境腳本改變手機狀態，例如：利用機器學習[22]，當使用者呈現站姿，將手機鈴聲音量調高、加入適地性服務[1][2][3]（Location Based Service，LBS），當使用者在博物館中，將手機調整為靜音或震動。

在 ASAP 中，建立情境腳本系統結構主要以人類最直觀的邏輯想法—IF〈條件（condition）〉THEN〈活動（Activities）〉—組成。條件（Conditions）：時間（Time）、電池（Battery）、地點（Location）、速度（Speed）、駕駛（Driver）、活動感知（Activity Recognition）。活動（Activities）：聲音（Sound）、震動（Vibration）、螢幕亮度（Screen Brightness）、手機設定（Settings）、撥號（Phone Call）、睡眠模式（Sleep Mode）。以上當條件符合時，便執行活動，執行完畢則刪除情境，加上自動監控系統與情境腳本搭配，能夠讓 ASAP 達成自動狀態改變的應用程式。

ASAP 是在開放式原始碼行動作業系統（Open Source Operating System）—Firefox OS 平台開發，用網頁技術（HTML5、CSS、JavaScript）開發，並使用 Web API 取得手機感測器感測的資訊，例如：三軸加速器、全球定位系統（GPS）、光源感應器。ASAP 使用 Firefox 瀏覽器提供的 Firefox OS Simulator[21]進行實測，最後能成功執行、判讀情境腳本，展現以網頁技術與 Web API 結合的手機應用程式。

本篇論文編排如下，第二章節我們將介紹 Firefox OS 大致的架構與 APP 的權限層級。第三章節則是會對我們提出的系統方法進行說明，分為三

大部份，情境腳本、黑名單系統、自動偵測。第四章節是我們在模擬器上執行情境條件判讀與活動執行之實驗。第五章節是目前 APP 的結論以及未來的展望。

貳、Firefox OS

Firefox OS是Mozilla對此系統命名的產品名稱，而在開發此系統的專案名稱為Boot to Gecko（B2G），且專案的原始碼發佈在開放原始碼社群（Github）。『Firefox OS App 同樣是以標準的 Web 技術 (如 HTML、CSS、JavaScript 等等) 所打造而成。如果你本來就是Web 開發者，就已經具備了大部分的技術。另必須知道數個特定的 JavaScript API，以利存取裝置的硬體與重要功能(如相機、陀螺儀、光線感測器、聯絡人資訊、系統警示通知.....)。這些也同樣能參閱應用程式中心與 Web 平台頁面獲得更多資訊。』[4]這段話引用自Firefox OS開發者網站的敘述，清楚地說明在此平台上的APP是以網頁技術與Web API開發。

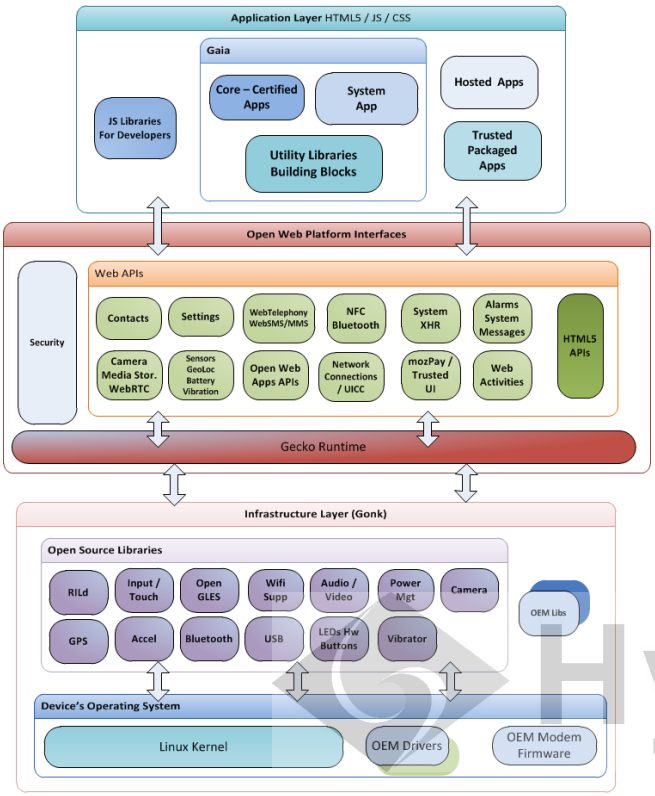


圖1 Firefox OS架構圖[5]

依照Firefox OS開發者網站的描述，系統的主要架構是Gonk、Gecko、Gaia三大區塊。其中的Gonk是此平台的底層作業系統，以Linux為核心（基於Android Open Source Project, AOSP）與使用者空間硬體抽象層（Hardware Abstraction Layer, HAL）組成，兩者皆來自常見的開放原始碼專案。

Gecko是應用程式的運行層，主要支援應用程式中使用的網頁技術和Web API。本層確保前述之API能夠完整運作在每一個Gecko支援的作業系統，也包含網路、圖形、排版堆疊以及JavaScript虛擬機。

最上層Gaia屬於使用者介面層，凡舉在Firefox OS開啓後看到的畫面皆是由此層所產生的，像是其它智慧型手機的畫面，也皆能在Gaia上實現。本層也同樣使用網頁技術實作，並開放與底層系統溝通，另外，第三方應用程式皆被安裝於此層。

Firefox OS的應用程式皆是在Gaia層實作，對於應用程式的權限分層共有三層，Web APP、Privileged和Certified[6]：

- **Web APP.** 未受信任層級。屬於一般網頁內容，包含網頁主要程式與安裝檔案（manifest）。
- **Privileged APP.** 受信任層級。此類APP較Web APP多一些權限能使用API。
- **Certified APP.** 受高度信任層級。這類系統App需經過電信服務商或OEM核可，且在手機出廠時，已經與手機系統一起安裝於手機。此類型APP能使用系統API（Settings API）改變手機設定，但不允許從外部導入JavaScript，而ASAP屬於此類型APP。

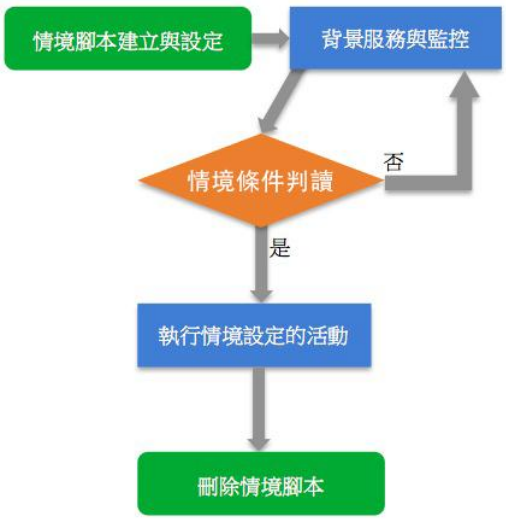


圖2 情境腳本執行流程圖

參、系統方法

ASAP 的系統主要由情境腳本、黑名單系統、自動偵測系統組成，以下分別介紹此三個系統。

3.1 情境腳本

情境腳本是改變手機狀態的一種任務系統，建立腳本前必須先對其命名，然後再選擇任務的條件以及執行的活動。腳本新增完畢時，將會在 ASAP 的首頁顯示建立的腳本，並且能夠編輯、查看、刪除。目前的設定為腳本只能設定一種條件且活動的設定不限制，所以目前情境的組合共有 4320 種。

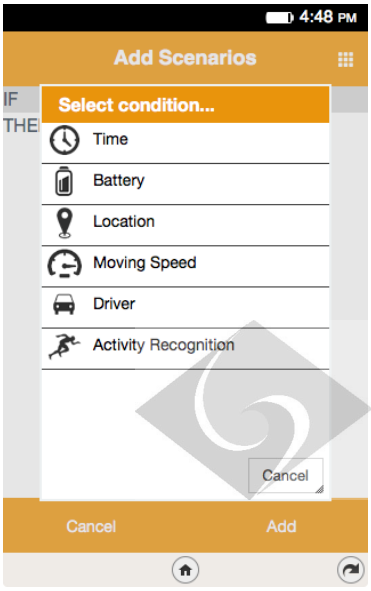


圖 3 條件選擇圖

3.1.1 情境的條件

時間

手機時間與設定值相同或在區間內，表示此種情境發生。加入時間情境的方法可以選擇時間點或區間，ASAP 利用 HTML5 中的 input 標籤（tag），類型（type）為 number，讓使用者輸入時間，將輸入的資料以 24 小時制的標準檢查是否正確，最後將小時與分鐘的數值以 data-phour（小時）和 data-pminute（分鐘）儲存至 IfTime tag 物件中。如果使用者輸入的是一段時間，則以時間的下界 data-boundlhr、data- boundlmin，時間的上界 data-bounduhr、data-boundumin 四種屬性儲存。

電池

包含兩種主要的參數，電池電量（level）以及電池充放電狀態（charging）。讓使用者加入的條件為：電量可以選擇「>=、>、==、<、<=」和百分比兩種設定，前者使用 HTML5 中的 select tag 讓使用者選取；後者以 input tag 輸入框輸入（預設情況為「==」與 0%）。輸入完之後會檢查輸入的資料，再以 data-blevel（數字）、data-blevelcondition（數字 1~5 對應前面敘述的五種情況）存入 IfBattery tag 物件。電池狀態有兩種，充電（True）、放電（False），使用 select tag 讓使用者選擇，將 data-bcharging（充放電情況）存入 IfBattery tag 物件。

地點

使用者選取地點並設定範圍（單位：公尺）當手機在此範圍內，就判定裝置在該地點。例如：設定台北的麥當勞為條件的地點，且範圍 100 公尺內，表示當裝置偵測到的地點屬於上述範圍內，就判定其所在位置為麥當勞。

因為 ASAP 屬於 Certified APP 無法外部嵌入 JS 腳本，所以利用 Activity API[14]與我們另外開發的選擇地點的 APP（名稱：ASAPLocation）完成加入地點情境條件。先定義產生的活動「var

locActivity=newMozActivity({name: ‘ASAPLocation’, data: loc});」，當此物件產生時，執行名稱為 ASAPLocation 的 APP，則 ASAPLocation 回傳的資料將會儲存在 loc 物件中，而且 locActivity 物件中需處理 onsuccess 與 onerror 的回呼函式，利用 onsuccess 回呼函式處理從 ASAPLocation 回傳的 loc 物件，即是使用者選擇的地理位置資訊。

在 ASAPLocation 當中，畫面顯示手機當前偵測之地點的地圖，再利用 Google Map API 中 Place Library 的 SearchBox[17]方法(此方法要求傳入 HTML5 的 input tag object)，對 input tag 加入 event listener，當 input 內容改變，顯示搜尋到的地點。由於找到的地點可能有多個，因此加入 markers[18]和 infowindows [19]類別，讓使用者藉由點選 marker 顯示 infowindows 資訊，得知需求的地點，如圖 5 所示。使用者點選 marker，跳出 infowindows，點選 Submit 便會將此地點資訊回傳至 ASAP。收到回傳的物件時，將此物件的訊息顯示在視窗中（如圖 4），按下 add，便可將物件中的 name、address、latitude、longitude 存入 IfLoc tag 物件。

速度

判斷裝置移動的速度之條件。利用 HTML5 input tag，取得內容與檢查輸入的正確性，將資料以 data-speed 的屬性加入 IfSpeed tag 物件。判斷方式則利用 Geolocation API[8] 中的方法 getCurrentPosition 中的 onsuccess 回呼函式，取得 position 物件中 speed 屬性，然後再與情境中設定的數值比較是否執行腳本。

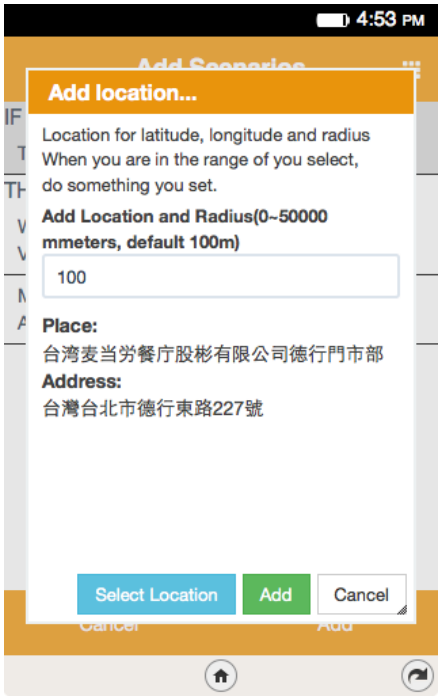


圖 4 位置條件加入圖

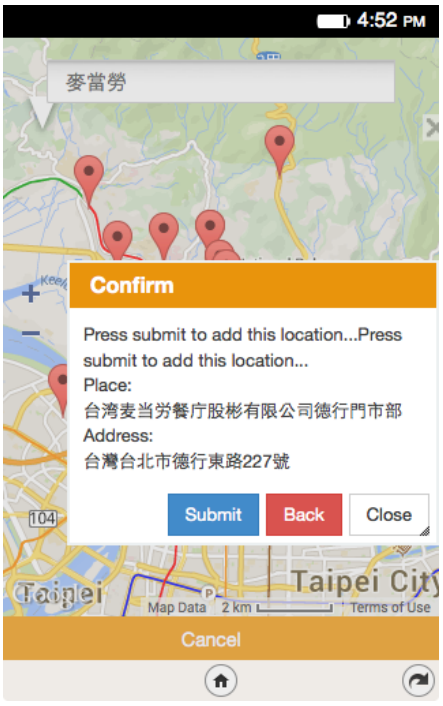


圖 5 選擇位置條件圖

駕駛

在判斷使用者當下是否為駕駛的情境，本文是假設鑰匙與手機結合，利用 WiFi 連結，然後將 Mac Address 作為鑰匙的 key[23]。當進入車內，手機便與車子配對，按下車上啓動開關，比對 Mac Address 是否相同，相同便表示使用者為駕駛。

實作的方法是利用 navigator.mozWifiManager[15]物件產生對手機WiFi狀態監看的 statuschange 事件，當手機連接上 WiFi 分享器設備，便觸發 statuschange 事件，執行監看式中的回呼函式，從函式的 evt 物件，將 evt.target.macAddress 與預設的 key 比對，如果相同，即代表使用者為駕駛。

活動感知

感知使用者當下的身體活動是坐下、站立、走路、跑步之其中一種。利用下拉式選單，供使用者選擇情境條件。使用此功能前，必須先訓練好四種動作資料。本文利用手機的三軸加速感測器取得 x、y、z 軸之加速變化量，然後計算出判斷當前手機方向的 42 個特徵值（x、y、z 均有值，以及 y 等於零且 x、z 均有值兩種情況，並計算每種情況各 21 個特徵值）。特徵值包括：平均值，標準差，最大值，最小值，第 10%、25%、50%、75%、90%值，1%~5%、1%~10%、1%~25%、1%~75%、1%~90%、1%~95%資料累加值，和 1%~5%、1%~10%、1%~25%、1%~75%、1%~90%、1%~95%資料累加平方和之值。

在監測的過程中，利用 Device Orientation API[9]，在 window 監看式中加入“devicemotion”事件，當事件改變時呼叫 handleMotion（此方法接收 event 的物件，透過此物件取得 acceleration.x，acceleration.y，acceleration.z 數值），而且設定每 50 毫秒取得三軸加速器感測的 x，y 和 z 數值。利用 Sliding Window 資料流量控管的概念，將 0～4 秒偵測的數據設為第一組資料，2～6 秒偵測的數據設為第二組資料，4～8 秒偵測的數據設為第三組資料，以上總共花費 8 秒取得三組資料。然後將訓練資料與偵測資料正規化，分別計算出 42 個特徵值，再將兩者使用 KNN 演算法（k-Nearest Neighbors Algorithm，最近鄰居法）找出最接近的動作，並執行投票活動，每次投票三次，三次之中票數最高的動作，即判斷使用者當下為此動作。

動作成功偵測之後，將掃描每一個情境腳本，有符合此情境條件的腳本就判斷動作與設定的動作是否一致，如果相同就執行此情境。

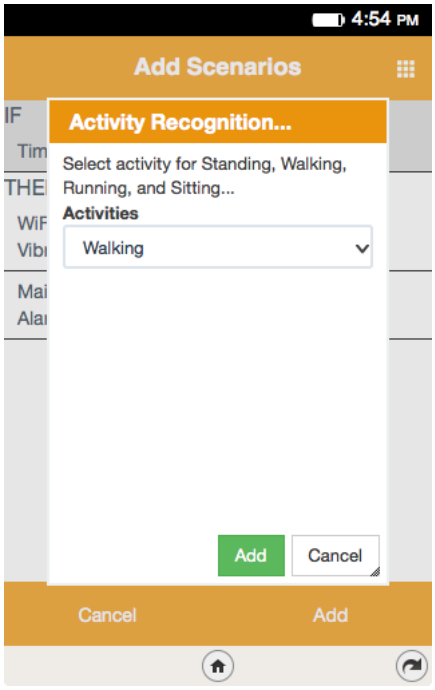


圖 6 活動感知條件圖

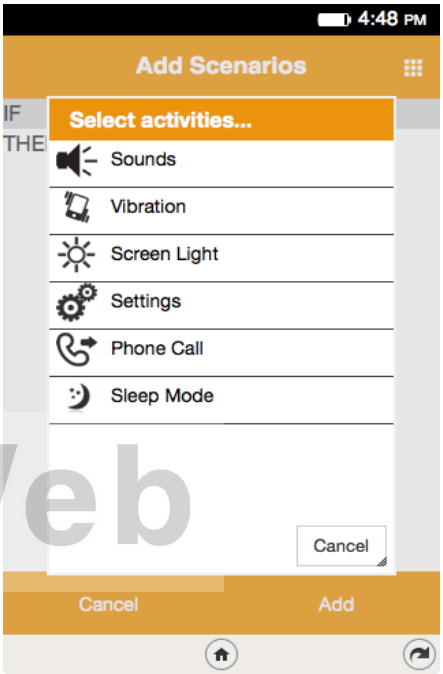


圖 7 活動選擇圖

3.1.2 腳本的活動

聲音

手機的聲音有很多種，本文中將聲音分為兩類，第一類為警告聲音（Alarm），當手機有發出警告的時候，產生的聲音；第二類為主聲音（Main）由雙音多頻 DTMF（Dual Tone Multi Frequency）、內容（Content）、通知（Notification）、語音 TTS（Text to Speech）四種組成。以上兩類的聲音設定範圍是 0~15，0 表示靜音，15 表示最大聲。

實作改變音量的方法是由 navigator.mozSettings[13]做物件管理，執行中需要先建立 Lock，使用 mozSettings.createLock() 方法，再對 Lock.set(object)方法傳入 object，依照使用者設定數值而改變之屬性只有 audio.volume 中的 dtmf、content、notification、tts 和 alarm 此五項。

震動

當情境條件符合，可以讓手機產生震動的效果。震動的活動設定為震動持續的時間（單位：秒）和是否持續震動至使用者取消此兩種。此活動的情況將會有三種組合：第一種是預設，震動持續兩秒，且震動三次即停止；第二種是設定持續秒數，且震動次數為三次；第三種設定持續秒數與是否按取消才停止。

實作方法利用 navigator.vibrate(time)[10]的 time 參數控制震動時間，利用 setInterval 與 clearInterval 控制震動執行的區間，達成前述之三種效果，加上計算次數的變數 vibrateTimes 便能控制震動次數。

螢幕亮度

調整手機螢幕背光亮度的活動，且使用者可以設定兩種情況。第一種是直接對手機螢幕亮度調整，由弱到強排序：極暗、暗、普通、亮、極亮；第二種是開啓或關閉手機內建的螢幕亮度自動調節功能。

實作手機亮度改變的方法，先建立 navigator.mozSettings.createLock()，再將{‘screen.brightness’: IntegerNumber}傳入 Lock.set()裡，其中 IntegerNumber 是第一段敘述的五種亮度對應的數值。若要開啓手機內建自動調節螢幕亮度則是使用 Lock.set({‘screen.automatic-brightness’:Boolean})方法。

手機設定

使用者可以開關無線網路（WiFi）、藍牙（Bluetooth）、地理位置偵測（Geolocation）、資料傳輸（Data Network）、省電模式（Power Saving）、震動開關（Vibration），或不設定。

更改手機功能設定方法是使用 mozSettings 建立 Lock 後，各別控制項的屬性是 wifi.enabled（WiFi 開關）、bluetooth.enabled（藍牙開關）、geolocation.enabled（GPS 開關）、ril.data.enabled（行動數據開關）、powersave.enabled（省電模式開關）、vibration.enabled（震動開關），將以上傳入 Lock.set({控制項: Boolean})，依照使用者所設定之屬性，便能更改手機功能之設定。

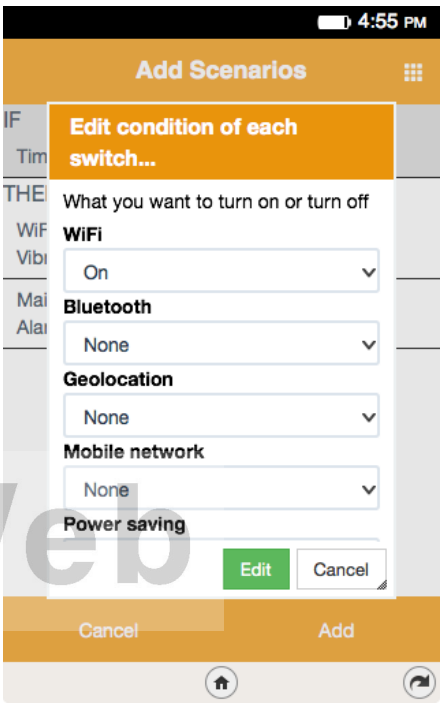


圖 8 手機設定活動選擇圖

撥號

依照情境腳本活動設定的電話號碼撥號。此活動需設定兩個屬性，姓名與電話號碼。實作撥號的方法，利用 Telephony API[16]宣告 window.navigator.mozTelephony 物件，呼叫 ASAP 定義的函式 dialPhoneCall 傳入電話號碼，再使用物件中的方法 telephoneCall.dial(TelephoneNumber) 方法撥號。

睡眠模式

睡眠模式的內容是開啓省電模式，以及將手機設定為靜音。當使用者設定睡眠模式為關閉，手機的省電模式便會關閉。ASAP 定義一個方法，此方法呼叫 setPowerSaving（開啓省電模式）和 setSound（設定音量大小）方法。setPowerSaving 方法是開啟或關閉手機內建的省電模式，setSound 方法則是將手機音量設定為 0。

3.2 黑名單系統

目前在 Firefox OS 系統上並沒有黑名單的系統，因此 ASAP 開發此系統，讓使用者將想拒接之電話號碼加入名單，當此號碼撥入時則自動取消通話。

圖 9 黑名單圖

黑名單系統實作方法，是從 input tag 取得輸入的姓名和電話，然後檢查電話號碼格式是否正確，如果正確就將資料儲存至 IndexedDB[12]。利用 ASAP 定義的 addData-ToBL() 方法，使用 transaction.objectStore(DB_STORE_NAME).add(DATA)加入資料，新增成功便執行 onsuccess 方法將成功加入之資料顯示於畫面中。加入成功後，發現錯誤，可以對加入的資料編輯，ASAP 使用 transaction.objectStore(DB_STORE_NAME).put 方法，將更新的資料 put 至資料庫中。刪除資料是利用 delete(刪除資料的 key)方法。開啟資料庫讀取資料需要較多時間，因此 ASAP 在應用程式開啟時與資料變更時，預先載入資料存入至陣列 dbBlackListNumberStore。利用 window.navigator.mozTelephony 物件執行自動拒接電話方法，主要是對 incoming 事件監看，當來電時執行回呼函式，從函式中的自帶參數之 event 中的 call.id.number 與 dbBlackListNumberStore[]中的號碼比對，相同則呼叫 event.call.hangUp()結束通話。

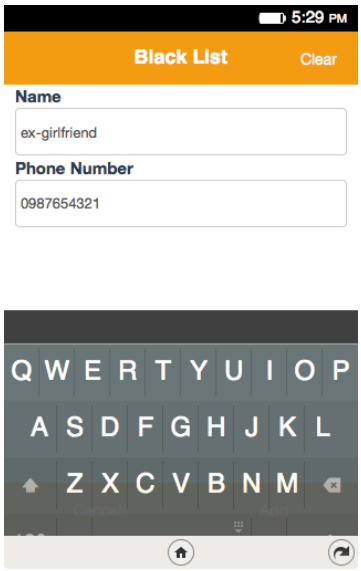


圖 10 新增來電拒接對象至黑名單圖

3.3 自動偵測

3.3.1 時間自動偵測

ASAP 加入計時器方法（scenarioTimeListener），用以每分鐘掃描存在的情境腳本，然後判斷情境條件是否成立，如成立便執行腳本。

計時器是當 ASAP 開啓的當下，先取得手機目前秒數（currentSecond），然後計算出延遲時間（delayTimer = 60 – currentSecond），使用 setTimeout 延遲執行 scenarioTimeListener，再加入 setInterval 方法，以 60000 毫秒執行計時器。

3.3.2 地點自動偵測

固定時間範圍內自動偵測當前地理位置。使用的方法是 Geolocation API，提供的 navigator.geolocation.getCurrentPosition(successCallback,error Callback, options)，當成功取得位置後，執行 successCallback ，然後從接收的 position 的物件中，取得 longitude 及 latitude，表示當前裝置所在的經緯度。errorCallback 回呼函式處理偵測不到位置的狀況，有以下三種情況：PERMISSION_DENIED（拒絕存取）、POSITION_UNAVAILABLE（位置未獲取）、TIMEOUT（超時）。options 物件包含的屬性有，enableHighAccuracy 是否高度精準測量、timeout 當偵測的時間超過此屬性的值就停止，以及 maximumAge 是 position 的物件最大存在的時間。

時區使用 Google TimeZone API 以跨站請求的方式，用 JQuery 中的$.getJSON(URL, callbackFunction(Data))方法取得時區的物件。URL 是 https://maps.googleapis.com/maps/api/timezone/json?location=**<latitude>**,**<longitude>**&timestamp=**<timestamp>**&key=**<key>**，其中的參數分別是緯度、經度、當前手機時間戳記及 API 的 KEY。利用 callbackFunction(Data)方法，取得的 Data 物件包含五個屬性：**dstOffset**、**rawOffset**、**status**、**timeZoneId**、**timeZoneName**，分別為**日光節約時間**、**UTC 時間偏移量**、**API 請求狀態**、**時區 ID** 及**時區名稱**，從 **status** 屬性先判定請求是否成功，便能得到當地時間：*timestamp + dstOffset + rawOffset*。

3.3.3 電池自動監控

ASAP 使用 Battery API[11]中的電池電量(levelchange)與電池充放電狀態(chargingchange)事件監看電池資訊。宣告電池的物件 var battey = navigator.batter ，再利用物件的監看式 battery.addEventListener(“事件”,levelChangeListener,false)和 battery. addEventListener(“事件 ”,chargingChangeListener,false) 。當 levelchange 或 chargingchange 事件被觸發則呼叫 levelChangeListener 或 chargingChangeListener。電量改變時執行 levelChangeListener 方法，掃描情境腳本中電池電量之情境條件；電池狀態改變時執行 chargingChangeListener 方法，掃描情境腳本中電池狀態改變之情境條件，並判斷與執行活動。

3.3.4 執行情境腳本的活動

情境發生便執行腳本活動，因此 ASAP 定義 executeActs 方法，傳入$(scenario ID)物件，從此物件找出是否符合在 3.1.2 中提到之六種活動，如果符合就執行。

肆、系統實驗

ASAP 是在 Firefox OS 平台開發是近幾年由 Mozilla 公司開發，不同於 Android、iOS 系統，使用網頁技術與 Web API 開發。我們開發所使用工具為 Sublime Text 3，實測則是使用 Firefox 瀏覽器提供的 Firefox OS Simulator，系統為 Firefox OS 2.0[20]。

4.1 情境腳本實驗

4.1.1 條件判讀結果

時間、電池、駕駛此三種情境判讀成功率為 100%。時間是由手機內部時間取得，藉由計時器的方式進行監測與比對，因此能達到 100%的判讀成功率；電池則是由 Battery API 對電池監控，當電量改變或充放電狀態改變時也同樣能達到 100%的判讀成功率；而駕駛條件之判斷，當手機與汽車連接成功，便判斷 MAC Address 是否與手機上預設之相同。

位置與速度之情境條件判讀成功率為 100%，因為使用 Geolocation API，在 getCurrentPosition 方法之成功回呼函式接收的 position 物件已經包含經緯度與速度。活動感知

使用 Device Orientation API 取得手機三軸加速器的資料，透過 KNN 找出最接近的動作，但由於不同品牌的手機的三軸加速器取得的值有所差異，因此成功率為 80%。

4.1.2 活動執行結果

聲音、震動、螢幕亮度、手機設定、撥號、睡眠模式，此六種的活動皆能正常執行。

表一 情境條件與活動執行成功率

條件	成功率	活動	成功率
時間	100%	聲音	100%
電池	100%	震動	100%
地點	100%	螢幕亮度	100%
速度	100%	手機設定	100%
駕駛	100%	撥號	100%
活動感知	80%	睡眠模式	100%

伍、結論

ASAP 在 Firefox OS 開發，本質是使用網頁技術（HTML5、CSS、JavaScript、Web API），也是第一個任務管理類型 APP。藉由新增情境腳本並改變手機狀態，以致不讓使用者在特定的場合下產生尷尬。

未來將在情境腳本加入多情境條件判讀與情境腳本存在之週期。另外，在情境條件與情境活動的選擇上加入更多選項，讓 ASAP 之情境腳本系統更完善。

陸、致謝

感謝鴻海精密工業股份有限公司 黃宇新經理在本論文研究期間，給予諸多幫忙與意見，讓本研究可以順利進行並得到良好的實驗結果。

參考文獻

[1] Cho, Eunjoon, Seth A. Myers, and Jure Leskovec. "Friendship and mobility: user movement in location-based social networks." Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2011.

[2] Gruteser, Marco, and Dirk Grunwald. "Anonymous usage of location-based services through spatial and temporal cloaking." Proceedings of the 1st international conference on Mobile systems, applications and services. ACM, 2003.

[3] Virrantaus, Kirsi, et al. "Developing GIS-supported location-based services." Web Information Systems Engineering, 2001. Proceedings of the Second International Conference on. Vol. 2. IEEE, 2001.

[4] Firefox OS 簡介, https://developer.mozilla.org/zh-TW/Firefox_OS/Introduction, 2014

[5] Firefox OS 架構, https://developer.mozilla.org/zh-TW/Firefox_OS/Platform/Architecture, 2014

[6] Firefox OS 安全性概述, https://developer.mozilla.org/zh-TW/Firefox_OS/Security/Security_model, 2014

[7] Grønli, Tor-Morten, et al. "Mobile application platform heterogeneity: Android vs Windows Phone vs iOS vs Firefox OS." Advanced Information Networking and Applications (AINA), 2014 IEEE 28th International Conference on. IEEE, 2014.

[8] Popescu, Andrei. "Geolocation api specification." World Wide Web Consortium, Candidate Recom-mendation CR-geolocation-API-20100907 (2010).

[9] DeviceOrientation Event Specification, http://www.w3.org/TR/orientation-event, 2011

[10] Vibration API, http://www.w3.org/TR/vibration, 2014

[11] Battery Status API, http://www.w3.org/TR/battery-status, 2014

[12] Collins, Mark J. "Indexed DB." Pro HTML5 with Visual Studio 2012. Apress, 2012. 255-279.

[13] Settings API, https://developer.mozilla.org/zh-TW/docs/WebAPI/Settings, 2013

[14] Web Activities, https://developer.mozilla.org/en-US/docs/Web/API/Web_Activities, 2014

[15] WifiManager, https://developer.mozilla.org/en-US/docs/Web/API/WifiManager, 2014

[16] Telephony API, https://developer.mozilla.org/en-US/docs/Web/API/Telephony, 2014

[17] Google Map API – SearchBox, https://developers.

google.com/maps/documentation/javascript/reference?hl=zh-tw-SearchBox, 2014

[18] Google Map API – Marker, https://developers.google.com/maps/documentation/javascript/reference?hl=zh-tw - Marker, 2014

[19] Google Map API – InfoWindow, https://developers.google.com/maps/documentation/javascript/reference?hl=zh-tw - InfoWindow, 2014

[20] Firefox OS 2.0 Developer, https://developer.mozilla.org/en-US/Firefox_OS/Releases/2.0, 2014

[21] Firefox OS 2.0 Simulator, https://ftp.mozilla.org/pub/mozilla.org/labs/fxos-simulator/, 2014

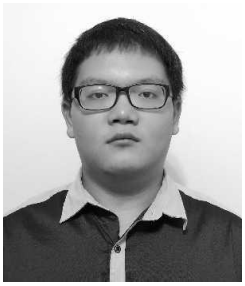
[22] P. Siirtola and J. Röning,"Recognizing human activities user-independently on smartphones based on accelerometer data,"International Journal of Interactive Multimedia and Artificial Intelligence, vol. 1, no. 5, pp. 38–45, 2012.

[23] Busold, Christoph, et al. "Smart keys for cyber-cars: secure smartphone-based NFC-enabled car immobilizer." Proceedings of the third ACM conference on Data and application security and privacy. ACM, 2013.

[24] Tasker, http://tasker.dinglisch.net, 2014

[25] Workflow, https://my.workflow.is, 2014

作者簡介



黃詩豪 於西元 2013 年進入國立臺南大學就讀，目前為資訊工程學系三年級學生，研究興趣為雲端運算，網路通訊，行動計算，物聯網。

林朝興 教授於西元 2000 年取得美國中佛羅里達州立大學電腦工程博士，目前任職於國立臺南大學資訊工程學系，研究興趣為網路媒體傳輸、同儕式媒體串流、行動計算、及媒體內容分析。