* 如果有時間可以再做的功能:
  + 在遇到小偷時，觸發樹梅派，跟他講現在是哪一台洗衣機遭竊
  + 樹梅派會做的動作:
    - 把攝影機的方向轉向目標洗衣機(使用步進馬達，每一台洗衣機都會在對應位置)
    - 觸發攝影機拍照，為了避免小偷在剛破壞機器時先馬上逃到別的角度而拍不到，因此除了一開始拍一張，可能還需要在隔個幾秒再拍幾張
    - 每次拍完照片都把照片存到S3
    - 之後APP就可以到S3中抓照片，同時顯示給業者
  + <https://blog.gtwang.org/linux/linux-aws-command-upload-download-s3-file-tutorial-examples/>
* 區塊鏈維修上鍊:
  + 平常會先一直去檢查機器(060319FFFF1C)有沒有問題，如果有問題(060319”55”xxxx，其中有故障時中間會是55，所以可以用那個55判斷)就會回傳資料，這時我就把這個資訊傳到雲端
  + 然後在雲端那邊就會上鍊
  + 之後就等待有人來維修，如果廠商修好了，就按某個按鈕觸發他去做自我檢查，如果檢查到修好了(060319000000)，就傳代表通過的資料到雲端
* 偵測洗衣機的運轉狀態，將其分為運轉中、待機中，當洗衣機的狀態改變就傳送一個訊息到雲端，APP在收到訊息之後就會顯示目前的狀態
  + 如何偵測??
    - 使用查看運轉情報的指令(060350FFFF55)，每間隔一段時間就看一下現在洗衣機的狀態
    - 假如剛剛是待機(060350000100)現在是運轉(060350000257)，就Set一個旗標，代表現在是剛啟動
    - 反之，假如剛剛是運轉(060350000257)，現在是結束洗衣(060350000550)，就Set一個旗標，代表現在是剛洗完衣服
    - 有個地方一直去偵測旗標的狀態，假設「運轉旗標」=1，就做對應的動作:
      * 傳訊息到雲端跟他講說現在開始運行了
      * 跟他講這次洗衣服的給水情報(060357FFFF52)
      * 把運轉旗標清為零
    - 有個地方一直去偵測旗標的狀態，假設「運轉結束旗標」=1，就做對應的動作:
      * (待填)
* 因為使用者在預約的時候，會需要有一組密碼來打開洗衣機，因此需要每次都產生一組新的密碼給使用者(沒有預約時電控鎖也是鎖著的，但密碼可能為空的，因此在按解鎖按鈕時直接按就可以了，相當於沒有輸入密碼)(這組密碼也同時用來解鎖，假如不適用預約的時候可以透過鍵盤自己設定密碼，假如客人沒有設密碼就開始洗衣，就需要傳一個訊息到雲端，透過簡訊通知使用者我們用亂數幫他產生的密碼，之後要解鎖就用這組密碼)
  + 使用者透過APP進行預約
  + 透過亂數產生一組密碼
  + 將此密碼透過APP顯示給使用者看，同時將密碼存入對應時段的資料庫中
  + 在預約時間快到的時候，使用Shadow的desired來更新目標洗衣機的「預約密碼」
  + 在收到更新密碼的通知後，更改存在嵌入式裝置中的某個變數，之後透過reported回報影子已經按照要求完成更新
  + 如果嵌入式板子在過程中重新啟動會有密碼消失的問題，解決方案:
    - 每次嵌入式Reset後就會透過reported回報預設的密碼到雲端(在建立好MQTT連線過後，否則無法送出/收到)
    - 因為他傳送到了更新的Topic(update/documents)，而我們也訂閱了一樣的Topic(檢查一下是否有改掉變成我reported的收不到)，因此我們會收到目前Shadow的狀態
    - 將儲存密碼的變數改成desired的密碼
    - 透過reported再回傳已經更改密碼
* Shadow總共會存以下的東西:
  + 業者密碼
  + 使用者預約密碼
* 鍵盤掃描、超音波感測器就開一個新的Thread，寫在同一個地方
  + 鍵盤掃描:
    - 使用到的變數:
      * 輸入狀態(char)
        + 會有a,b,c,d幾個狀態，看要不要再加一個預設狀態，代表的是按下幾個功能鍵後改變的狀態
      * 零錢箱密碼(char \*)
        + 會有預設值1234之類的
        + 可以讓業者更改
        + 使用Shadow存放，每次開機時先透過Shadow更新
      * 輸入零錢箱密碼(char \*)
    - 用在以下地方:
      * 業者設定密碼
        + 先按下「更改零錢箱密碼」
        + LCD顯示「請輸入舊密碼」(使用密碼變數去接收)，按Enter鍵送出(更改零錢箱密碼是一個狀態，裡面還會分成幾個小狀態)
        + 若正確，則LCD清除改顯示「輸入新密碼」，並且將「零錢箱密碼」變數清為空字串(這個會在Enter那邊判斷，如果cmpState == enterOldPassword，則判斷剛輸入的enterPassword是否與customerPassword相同，若相同則將cmpState改成enterNewPassword)
        + 使用者輸入新密碼(使用「零錢箱密碼」變數去接收，每次往後新增字元)
        + 使用者按下「Enter」鍵送出
        + 透過Shadow去更新業者密碼，並且LCD顯示「密碼更新完成」

更新密碼為密碼變數裡面存放的字串

實際做法為使用sprintf去改變業者密碼對應的欄位

* + - * + 之後APP也可以透過去抓Shadow的值，把密碼顯示在APP中(目的是為了讓業者知道他的密碼變成怎樣)
      * 業者輸入驗證碼(零錢箱防盜)
        + 業者按下「零錢箱驗證」按鈕
        + 更改「輸入狀態」變數為b
        + 在輸入按鍵的副程式中，會判斷輸入狀態是甚麼，在狀態為b的區塊更新「輸入零錢箱密碼」變數
        + 密碼輸入完成後按下「Enter」鍵
        + 比較「輸入零錢箱密碼」變數、「零錢箱密碼」變數
        + 若相同則:

將「驗證成功」變數設為「True」

清除「輸入零錢箱密碼」變數

LCD顯示「驗證成功」

* + - * + 若不同則:

LCD顯示「驗證失敗」

清除「輸入零錢箱密碼」變數

* + - * 更改使用者密碼部分，應該要先檢查洗衣機有沒有在運轉，假如沒運轉就不能設密碼，不然就會發生洗衣機沒人使用，卻鎖起來的情形發生。
      * 使用者輸入密碼來解鎖洗衣機(預約上鎖)
        + 平常沒有預約，也沒人使用時，會沒有密碼，可以直接打開使用
        + 使用者在預約的時候會獲得一組密碼
        + 在預約時間快到的時候，雲端那邊會去更改Shadow，把洗衣蓋密碼改成使用者之前獲得的密碼
        + 地端在雲端使用desired更新影子之後，把「洗衣機密碼」變數改成desired的密碼
        + 使用reported回傳更新影子

----------------------------------- 使用者嘗試解鎖 ------------------------------

* + - * + 使用者按下「洗衣門蓋解鎖」，這時輸入狀態變數更新為c
        + 這時候「洗衣機密碼」變數是有數值的，因此判斷假設「洗衣機密碼」變數不是空字串才執行以下功能，否則直接將洗衣機打開
        + 在輸入按鍵的副程式中，會判斷輸入狀態是甚麼，在狀態為c的區塊更新「輸入洗衣機密碼」變數
        + 使用者按下Enter鍵(d):

在副程式中判斷得知目前輸入的字元是d，就再接著判斷「輸入狀態」變數，假如是c就把剛剛的「輸入洗衣機密碼」變數跟「洗衣機密碼」變數做比較，假設相等就:

將電控鎖解鎖

將「洗衣機密碼」變數清除為空字串

假設不相同就在LCD上顯示「密碼輸入錯誤」

* + - * 使用者直接按開啟洗衣機蓋的按鈕(平時洗衣機上鎖，但沒密碼)
        + 「洗衣機密碼」變數、「輸入洗衣機密碼」變數平常都是空字串，有預約的時候「洗衣機密碼」變數才不會是空字串
        + 依此一開始判斷，假設按下的是c，而且「洗衣機密碼」變數是空字串，就直接將電控鎖打開

|  |  |  |  |
| --- | --- | --- | --- |
| 7 | 8 | 9 | 更改零錢箱密碼(A) |
| 4 | 5 | 6 | 更改使用者密碼(B) |
| 1 | 2 | 3 | 零錢箱解鎖(C) |
| \* | 0 | Enter(E) | 開啟洗衣機(D) |

* + - 先寫一個副程式，他使用的參數是一個字元，代表的是掃描到輸入的字元，假如輸入的字是a~c之間，則將「代表輸入狀態的變數」更改為對應的字元(例如按下「更改零錢箱密碼」對應到字元’a’，這時就將狀態變數改為a，之後就可以判斷假如狀態變數==a，就代表現在是要改密碼)
* LCD顯示(OK):
  + 預先設計以下幾個副程式:
    - 將LCD清除
    - 將Cursor移到初始位置
    - 寫入想要的指令，參數為指令字串、RS的類別(要做指令控制/顯示資料)
      * (Enable本來都是0)
      * 設定RS
      * 把Enable拉成1
      * Delay 1mS
      * 對各個接角寫值(D0~D7)
      * Delay 1mS
      * 把Enable設回0
      * 再一個一個字元顯示出來(好像有個模式是可以顯示完Cursor自己移動)
  + RW永遠設成0，因為我們只要寫
  + RS是設定模式，資料或指令
  + <https://controllerstech.com/interface-lcd-16x2-with-stm32-without-i2c/>
  + <https://www.openhacks.com/uploadsproductos/eone-1602a1.pdf>
  + 日立 HD44780 相容的 LCD 有 4-bit 和 8-bit 兩種使用模式，使用 4-bit 模式主要的好處是節省 I/O 腳位，通訊的時候只會用到 4 個高位元 (D4-D7)，D0-D3 這四支腳位可以不用接。每個送到 LCD 的資料會被分成兩次傳送 – 先送 4 個高位元，然後才送 4 個低位元。
  + 這次使用的是4位元模式，可以減少GPIO的量
    - 若要顯示字元’A’，就要去看他的ASCII Code，是0x41(01000001)
    - 在寫入資料的時候是，先讓D7~D4 = 0100，然後Enable一次
    - 接著再讓D7~D4 = 0001，一樣這兩次都要Enable
* 超音波感測器原理(OK):
  + 給Trig腳一個至少10us的高電位信號，HC-SR04會發送8個40KHz的方波，並自動偵測是否有返回信號，若有信號返回，則透過Echo腳位送出一個高電位的訊號，高電位持續的時間就是超音波從發射到返回的時間。
  + 由此資訊就可以計算出距離: 測試距離 = { 高電位 x 聲速(340m/s) } / 2
  + 每秒34000公分
  + 300uS = 300/1000000秒 = 300 \* 0.034公分
  + <https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf>
* 現在已經改成訂閱Document，因此不管是使用desired或reported都會收到整個影子資訊(不包含delta)，之後要做的是:
  + 一開始先抓出current>>state>>desired、reported的資料，將兩個進行比對，假如兩個內容有不同材執行下一步
  + 再從desired中抓出所有value，更新地端對應的變數為該值
  + 使用reported更新Shadow
  + (雖然也訂閱了document，但因為更新完後照理來講desired、reported的資料應該會相同，所以不會執行任何內容)
  + (假如有需要在每次更新內容時都做出某些反應，則可以在後面再加一個else，並且在if、else中都做出那些反應)
* 假如要用stm32搭配ESP8266可以參考這個網址:
  + <https://github.com/nimaltd/ESP8266>
* WICED3好像有問題