

# 基於雲端服務之物聯網害蟲即時監測系統

R09631007 吳乙澤 R09631011 陳玟綏 R09631020 何宜臻

Github 連結：[https://github.com/csinrn/CCACS\\_Final](https://github.com/csinrn/CCACS_Final)

Demo 影片連結：<https://reurl.cc/7oyyAb>

## 摘要

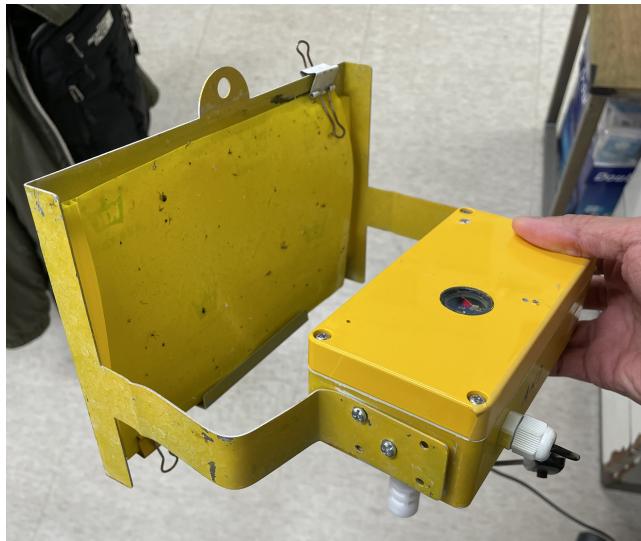
農業現場的傳感器部署與資料處理在精準農業的蓬勃發展下日益受到重視。本專題善用嵌入式系統元件與雲端計算平台Aws，架設出易佈署的農場害蟲監控系統。我們利用樹莓派(RaspberryPi)、光照感測器(BH1750)、溫濕度感測器(SHT20)、相機模組等傳感器收集現場資料，使用雲端服務Aws IoT Core, IoT Analytics, Lambda 等服務處理回傳之環境因子，並在雲端平台上架設Node.js後端處理資料，與實作Html前端來實現資訊的易取得性，以期輔助第一線農民快速掌握農地蟲害狀態，以做出及時對策，降低不必要的農業損失。

## 研究材料與方法

該章講述本專題所用到的硬體材料，以及實作本害蟲監控系統的方法，方法包括系統架構、資料處理、後端、前端.....等四項。另外，本系統完成後，有把它放在實驗室空間做測試。測試結果包含系統的上傳資料、網頁等兩項成果，這兩項成果將於「結果與討論」中做更詳細的說明。

## 硬體材料

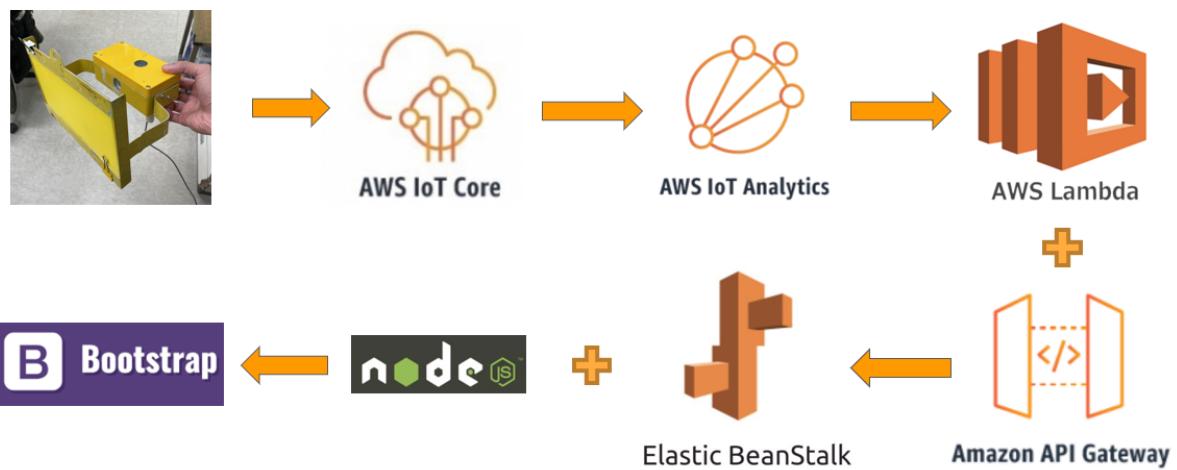
主要用到樹莓派(RaspberryPi)、光照感測器(BH1750)、溫濕度感測器(SHT20)、空氣品質感測器(SGP30)、相機模組(Pi Camera V2)、黏蟲紙、防水盒等材料，組合出一個嵌入式系統。搭配前端、後端、資料庫的使用，便可將該嵌入式系統所收到的資料呈現在網頁上。見下圖一，該圖為本嵌入式系統的組裝成果。右方盒狀物為防水盒，盒內裝有樹梅派、各種感測器與相機，相機可透過盒內的鏡片，拍攝到外部的黃色黏蟲紙。由此圖片，便能獲得害蟲數量。最後，分析感測器資料與害蟲數量，比較兩者的關係，便能獲得相關性(Correlation)、線性迴歸(Linear Regression)方程式等資訊。



圖一、嵌入式系統組裝成果

## 系統架構

本專題主要以雲端計算平台 AWS(Amazon Web Service) 的各項服務來實現，包括 AWS IoT Core、IoT Analytics、Lambda …… 等服務。見下圖二，該圖為本專題的架構圖。首先，將圖一的嵌入式系統放在特定場域中，該系統便能持續地收集系統周邊的資料，並透過 AWS IoT Core 的 MQTT 協定，將感測器資料傳送到雲端。傳送完畢後，再以 AWS IoT Analytics 將資料儲存成 CSV 的格式。爾後，為使其他開發者能方便地取得資料，使用 AWS Lambda 和 API Gateway 產生一組 Restful API。其他開發者收到該組 API 後，便能利用該 API 下載資料以做線性迴歸。最後，我們還建立了一個網頁，後端是 Node.js，前端是 HTML。後端用來做線性迴歸運算，前端則將系統蒐集到的資料、迴歸的運算結果視覺化，以製作成圖表、圖形或表格以利查看。若有時間的話，會再嘗試以 AWS Elastic Beanstalk 快速部署網站。



圖二、系統架構圖

## 資料處理

資料處理方面利用 AWS Lambda，從資料庫導出現場設備儲存的資料，進一步處理成有利用價值的資訊。根據研究，害蟲數量會受到溫度積、光度等因素影響。利用現場設備收集到的環境資料紀錄當地長時間的環境資訊與對應的害蟲數目變化，即可利用機器學習方法來分類甚至預測當前硬體設備附近的害蟲成長趨勢，並量化為可能性數值，以網頁的方式呈現數據，輔助第一線農民快速準確地做出決策。

## 後端

後端使用Node.js來串連硬體服務收集之資料與前端顯示，善用express framework精簡、靈活的特性，架設API提供前端處理後的數據。我們使用THI指數做x軸，害蟲數目做y軸做regression，作為以環境資訊預測害蟲數量的根據。

## 前端

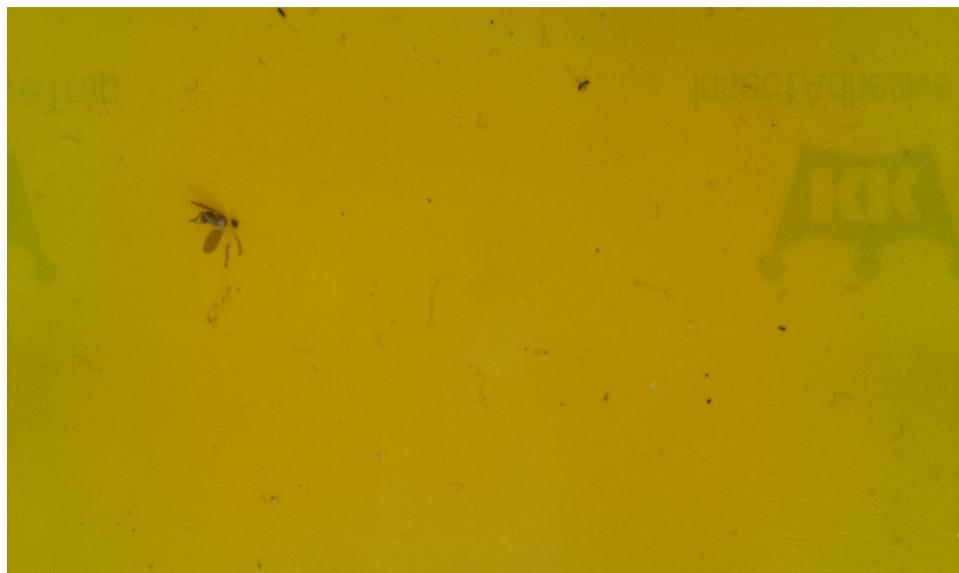
以 HTML 的架構設計網頁，並用 JavaScript 的 API 去跟伺服器要資料後，顯示在網頁上。之後再以 bootstrap 前端框架和 Google Charts 等免費框架來盡可能地美化網頁。下圖三為利用 Google Charts 中 trendLine 來做第一張 Linear Regression 動態圖的簡化版 Code。

```
1 <html>
2   <head>
3     <script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>
4     <script type="text/javascript">
5       google.charts.load("current", {packages:["corechart"]});
6       google.charts.setOnLoadCallback(drawChart);
7       function drawChart() {
8         var data = google.visualization.arrayToDataTable([
9           ['THI', 'Bugs Number'],
10          [ 8,      12],
11          [ 4,      5.5],
12          [ 11,     14],
13          [ 4,      5],
14          [ 3,      3.5],
15          [ 6.5,    7]
16        ]);
17
18        var options = {
19          title: 'Linear Regression of THI and Bugs Number',
20          legend: 'none',
21          crosshair: { trigger: 'both', orientation: 'both' },
22          trendlines: {
23            0: {
24              type: 'polynomial',
25              degree: 3,
26              visibleInLegend: true,
27            }
28          }
29        };
30
31        var chart = new google.visualization.ScatterChart(document.getElementById('polynomial2_div'));
32        chart.draw(data, options);
33      }
34    </script>
35  </head>
36  <body>
37    <div id='polynomial2_div' style='width: 900px; height: 500px;'></div>
38  </body>
39 </html>
```

圖三、前端程式碼

## 結果與討論

見下圖四，該圖是置於本嵌入式系統內部的相機所拍攝到的黏蟲紙照片。該照片是系統上傳到雲端資料庫的。上傳完成後，再使用 HTML 和雲端資料庫所提供的 URL，便能將照片顯示在網頁上。拍攝完成後，透過 YOLO 模型，能偵測到黏蟲紙上的害蟲數量。將害蟲數量與感測器資料做分析後，便能獲得下表一。



圖四、相機所拍到的黏蟲紙圖片(可於網站上顯示)

見表一，該表顯示以線性迴歸方法，處理嵌入式系統在實驗室蒐集到的資料後，所算得的線性方程式與該式的決定係數 (coefficient of determination)。見直線方程式，該式的  $x$  為 THI 值， $y$  為害蟲數量。見決定係數，該係數表示  $x$  與  $y$  資料的相關程度。該值為 0.04，表示在實驗室環境下所測得的 THI 值與害蟲數量的關聯性極低，幾乎不存在關聯性。推測原因是在實驗室這種密閉空間中，原本就不太容易抓捕蟲類，導致蟲類數量與感測器所獲得的資料對不起來，進而在跑完線性迴歸後，得到極低的決定係數 0.04。

表一、實驗室資料的迴歸結果

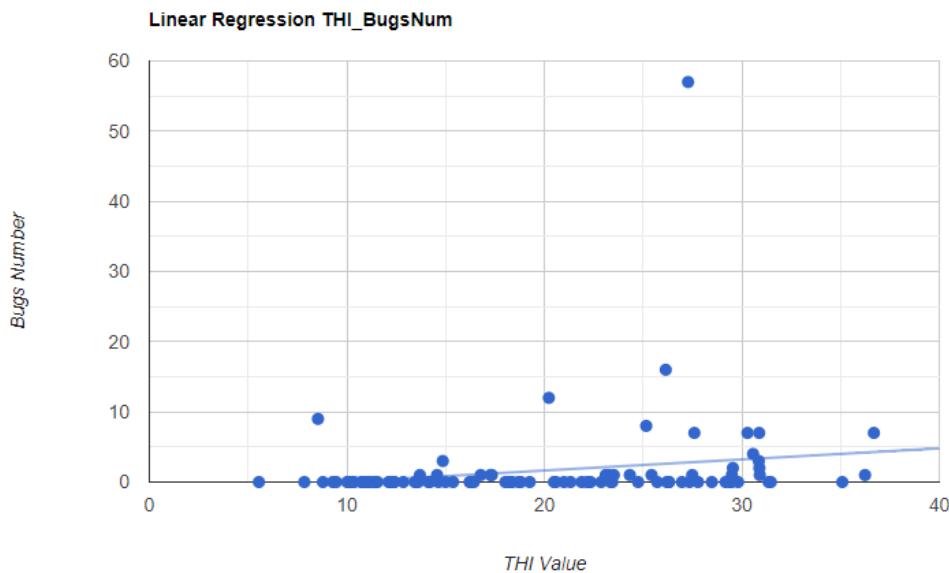
直線方程式	決定係數
$y = -0.29x + 9.42$	0.04

因此，以線性回歸分析在實驗室所獲得的資料是不合理的。想把該套系統帶到實際的農園做測試，但又沒有足夠時間去蒐集足量的資料以做線性迴歸。於是，考慮到本專題為課程的期末報告，為了 Demo 方便，我們便以人工的方式生成資料集，以彌補實驗室資料集相關性不足的問題。見下表二，該表為自行生成資料集的回歸結果。決定係數 0.7，較實驗室資料的回歸結果好上不少，代表 THI、害蟲數量兩者間有關聯性。

表二、自行生成資料集的迴歸結果

直線方程式	決定係數
$y = 0.2x + -2.05$	0.7

見下圖五，該圖為自行生成資料集的分布圖與線性回歸的直線方程式。可以看出  $x$  為 27 處， $y$  為 56 處有個偏離其他數值的點。該點為離群值 (Outlier)，在一般正常的數據分佈也很常見，表示我們自行生成的資料集挺符合正常數據的趨勢。



圖五、自行生成資料集於網站上的結果

## 結論

本專題提出一個以雲端計算平台為基礎的物聯網監控系統，除了來不及將系統放置到實際農園蒐集資料外，其他部分都運作的挺正常的。系統偵測到的資料能順利地透過 AWS 服務傳到雲端，後端也能簡便地使用 Restful API 取得雲端上的資料。若最後的資料集不必自行生成的話，該專題便能如數完成先前設定的各個預期目標。

## 參考資料

1. Aws IoT Core ([https://docs.aws.amazon.com/zh\\_tw/iot/latest/developerguide/mqtt.html](https://docs.aws.amazon.com/zh_tw/iot/latest/developerguide/mqtt.html))
2. Aws Lambda ([https://docs.aws.amazon.com/zh\\_tw/lambda/latest/dg/getting-started.html](https://docs.aws.amazon.com/zh_tw/lambda/latest/dg/getting-started.html))
3. Bootstrap (<https://codeofaninja.com/2014/05/bootstrap-tutorial-beginners-step-step.html>)
4. Google Charts (<https://developers.google.com/chart>)