TeraSoft 鈦思科技
www.terasoft.com.tw

以 MATLAB、機器學習與 ThingSpeak 平台開發物聯網解析系統

By Robert S. Mawrey, MathWorks

智慧裝置與資料解析、機器學習技術的結合，可以被廣範地運用在各個領域，從自行開發的交通監測，到先進的預測維護系統，以及未來各式各樣的消費性產品，比如 Amazon Echo 及 Google Nest 等。

雖然物聯網(Internet of Things, IoT)潛能無窮，但物聯網系統的設計及開發需要複雜網路架構以及多領域的專業，可能讓人感到氣餒。

本篇文章將介紹如何利用資料解析（data analytics）技術，且在無需開發客製化網路軟體或伺服器的情況下來設計開發 IoT 系統並進行原型化及轉檔佈署。這項工作流程以 MATLAB®、以及可以依需求在雲端執行 MATLAB 程式碼的 IoT 解析平台 - ThingSpeak™為基礎。為了說明本工作流程，我們將建立一個潮汐預測系統做為例子，該系統可以讓船員預測風對潮水水位所造成的影響。

水的深度隨著潮汐有所不同，但同樣也受到風的強度、時間、與方向的影響。風成的水位預測通常需要先進的流體力學模型，並且須對當地海灣與海床形狀具備相當深入的了解。美國國家海洋暨大氣總署(NOAA)與其他組織利用這些資源來預測主要港口的水深，但次要的港口與海灣則無法正當利用這些資源。

我們將要開發的這個系統，不需要精密計算的流體力學模型與複雜的設備，只需要基於類神經網絡技術以及低成本的硬體裝置，就可提供一套可行且經濟實惠的方式給予次要港灣進行潮汐預測。

本案例所使用的程式碼可以從 https://www.mathworks.com/matlabcentral/fileexchange/60344-thingspeak-tidal-and-wind-surge-forecasting-example 下載。

潮汐風浪預測範例

本範例以從美國東北部麻薩諸塞州鱈魚角的一個海灣所收集到的資料為基礎(圖 1)。

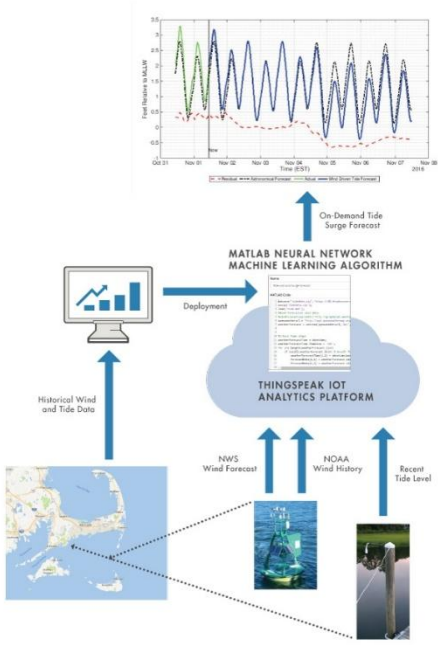

圖 1. 潮汐預測系統。

在 MATLAB 中執行程式碼，可以讀取從 ThingSpeak 以及其他線上資料來源的風及潮汐的資料，接著執行可以預測潮湧並依需求產生潮湧預測圖(圖 2)的潮汐預測與類神經網絡演算法。

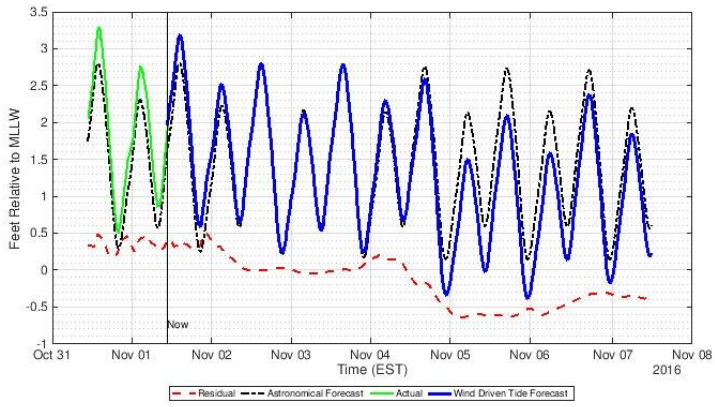

圖 2. 以 MATLAB 機器學習與 ThingSpeak 為基礎的風成潮汐預測圖。圖中可看到潮汐的實際量測值、天文潮汐預測(不考慮風的因素)、以類神經網絡預測的風成潮汐預測(天文加上風)、以及殘差(實際量測值或風成預測與天文預測之間的差距)。

我們將透過五個步驟來建立預測系統。

1. 收集歷史資料
2. 分析資料
3. 開發預測演算法
4. 將解析結果佈署到雲端
5. 依需求進行資料的解析與視覺化。

步驟 1：收集資料

一開始，我們以 ThingSpeak 以及低成本的測潮計來收集潮汐水位的歷史資料。測潮計的韌體是以 C 程式碼編寫，我們使用了 ThingSpeak 的開放源嵌入式裝置之通訊函式庫（open-source 4ThingSpeak communication libraries）發佈到 GitHub。

步驟 2：分析歷史資料

我們將歷史資料從 ThingSpeak 下載到桌上型電腦上的 MATLAB 來執行分析工作。在桌上型電腦，我們可以檢驗及清理資料，來移除雜訊、異常值、缺漏值、錯誤值及其他反常情形。

```
% Download the tide data using a custom function

[tideTime,tideRangemm] = readAllMyTideData();

% Convert the range to water depth from the mud

disttomud = 3449; %Measured distance from the gauge to the mud.

depthFeet = (disttomud - tideRangemm) / 25.4 / 12;

[tideTime,duptimeindex] = unique(tideTime);

depthFeet = depthFeet(duptimeindex);
```

```
depthFeetNoisy = depthFeet;

% Remove outliers (could also use movmedian)

depthFeet = hampel(depthFeet,121,2);

depthFeet = hampel(depthFeet,11,1);
```

我們將資料繪製成圖表，確認異常值與其他反常的情況都已被移除(圖 3)。

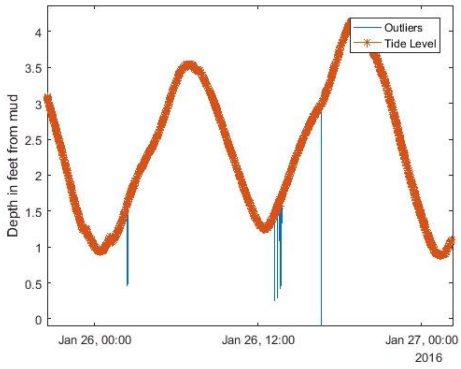

圖 3. 圖表顯示資料中的雜訊及異常值已被移除。

潮汐水位的量測與平均較低低潮位(*mean lower low water,* MLLW)或太陰日(lunar day)當中的水位最低潮位(lowest level)有關。我們利用訊號處理工具箱(Signal Processing Toolbox™)中的 `findpeaks` 函式計算 LLW 水位。

```
%Invert the data to use findpeaks to find the valleys

negdepthFeet = -depthFeet;

dur = duration(15,0,0);

% Use findpeaks

[lowerLowWater,lowerLowUTC] = findpeaks(negdepthFeet,

tideTime,'MinPeakProminence',10/12,'MinPeakDistance',dur);
```

```
lowerLowWater = -lowerLowWater;

MLLWmud = mean(lowerLowWater);

MLLW = 0;

% Tide level in feet relative to MLLW

feetMLLW = (depthFeet-MLLWmud);
```

我們繪製較低低潮位值來檢驗計算結果(圖 4)。

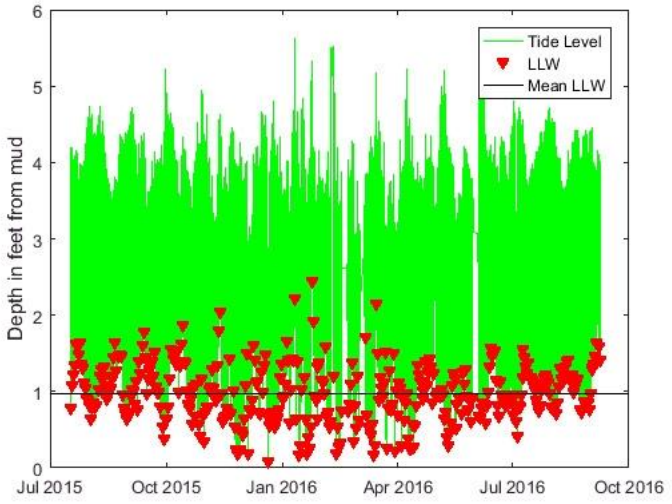

圖 4. 平均較低低潮位(mean lower low water, MLLW)的計算。

接下來，利用清理過的潮汐資料，以 File Exchange 中先進的潮水分析函式 - UTide 來預測未來潮水水位。透過 UTide，我們可以計算許多在特定地點形成潮汐的正弦(sinusoidal)函式的相位與波幅。

```
% Predict Tides. Downsample to avoid memory issues.

sampledTide = feetMLLW(1:15:end);

sampledTime = tideTime(1:15:end);
```

```
MashpeeLatitude = 41.6483;

% Calculate the tidal coefficients using UTide's ut_solv function

      tideCoefWithSurge = ut_solv(datenum(sampledTime),sampledTide,[],...

      MashpeeLatitude,'auto','notrend','DiagnPlots');



% Use ut-reconstr to forecast the astronomical tides at the same times as the

% measuredtides

forecastedFeetMLLW = ut_reconstr(datenum(tideTime),tidecoefNoSurge);



% Calculate the residual error

residualFeetMLLW = feetMLLW - forecastedFeetMLLW;

residualFeetMLLWhourly =

interp1(datenum(tideTime),residualFeetMLLW,datenum(tideTimeHourly));

residualFeetMLLWhourly = fillgaps(residualFeetMLLWhourly);
```

將天文預測值與量測資料值做比較，可看到一個隨著時間而變動的殘差錯誤(圖5)。

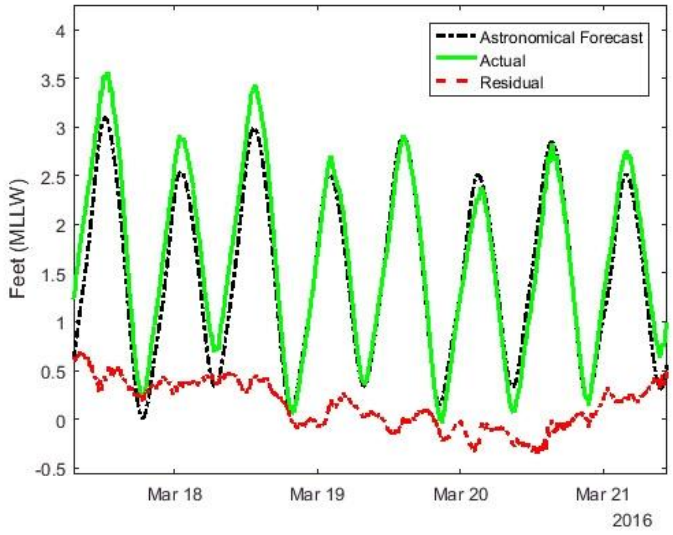

圖 5. 潮汐量測值與天文預測之間的殘差錯誤。

我們注意到殘差錯誤與風的強度成比例：有風在吹送的時後，水會被推進或推出海灣。當風從特定方向強力吹送，會造成潮水水位上升或下降將近一呎(圖 6)。

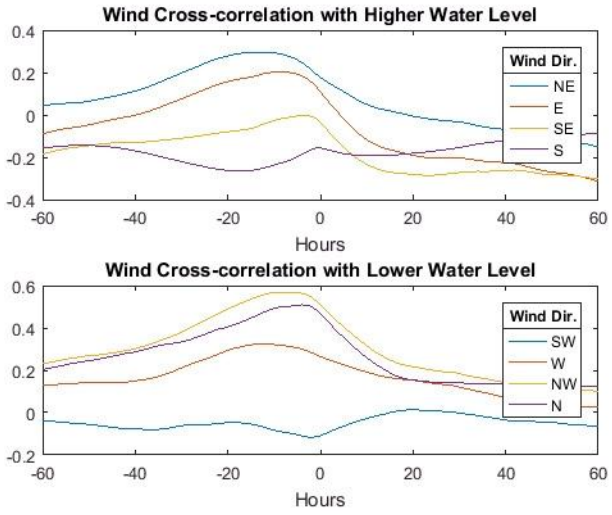

圖 6. 風與水位的交叉相關。

很明顯地，我們需要將風的強度列入考量。在下一個階段，我們將會說明如何預測風對水位的影響。

步驟 3：開發預測演算法

在以各種輸入資料集嘗試使用 Fitting 與 NARX 類神經網絡之後，我們發現 NARX 類神經網絡擅長較近期的預測，而 Fitting 類神經網絡對於近期及長期的預測都很有效。

輸進類神經網絡的資料包含了風的歷史與預測資料、潮汐水位量測值、潮汐水位的天文預測值。我們利用類神經網絡工具箱(Neural Network Toolbox™)中的 Neural Net Fitting 與 Neural Net Time Series 應用程式設計並測試類神經網絡。

```
% Use the readNOAABuoy function to read the data from NOAA buoy 44020

% readNOAABuoy(buoyNumber,years)

[buoyTime,buoyWSPD,buoyWDIR,buoyPRES]=readNOAABuoy(44020,1);



tideTimeHourly.TimeZone = 'UTC';

windSpeedResampled = interp1(buoyTime,buoyWSPD,tideTimeHourly,'linear','extrap');

windDirResampled = interp1(buoyTime,buoyWDIR,tideTimeHourly,'linear','extrap');

windSpeedSquared = windSpeedResampled.^2;



% Prepare the data to create arrays to train a NARX neural network

tideForecastHourly = ut_reconstr(datenum(tideTimeHourly),tidecoefNoSurge);

stressN = (windSpeedResampled.^2).*cosd(windDirResampled);

stressE = (windSpeedResampled.^2).*sind(windDirResampled);

neuralInputNarx = [tideForecastHourly,stressN,stressE];

neuralOutputNarx = tideResampled - MLLWmud;
```

```
neuralInputNarxTrain = neuralInputNarx(300:4950,:);

neuralOutputNarxTrain = neuralOutputNarx(300:4950);
```

我們利用類神經網絡應用程式來訓練 NARX 網絡(圖 7)。

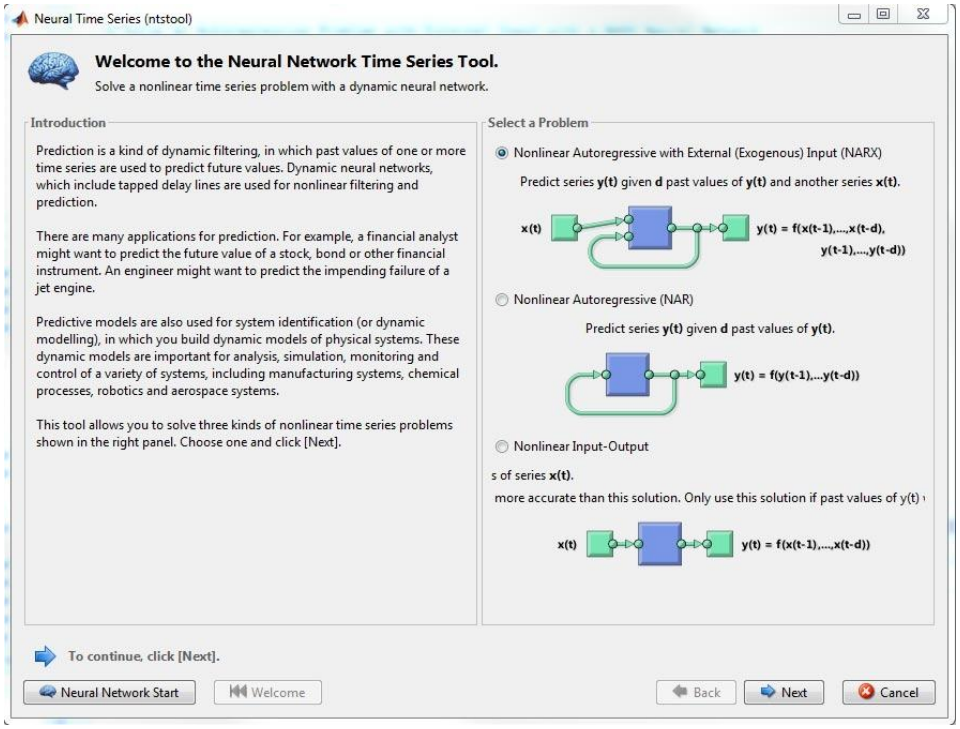

圖 7. 利用 Neural Time Series 應用程式訓練一個 NARX 網絡。

我們利用 Neural Time Series 工具訓練 NARX 網絡，並產生以下程式碼：

```
% Solve an Autoregression Problem with External Input with a NARX Neural

% Network

% Script generated by Neural Time Series app



X = tonndata(neuralInputNarxTrain,false,false);

T = tonndata(neuralOutputNarxTrain,false,false);
```

```
% Choose a Training Function

trainFcn = 'trainlm';    % Levenberg-Marquardt backpropagation.



% Create a Nonlinear Autoregressive Network with External Input

inputDelays = 1:24;

feedbackDelays = 1:24;

hiddenLayerSize = 9;

net = narxnet(inputDelays,feedbackDelays,hiddenLayerSize,'open',trainFcn);



% Choose Input and Feedback Pre/Post-Processing Functions

net.inputs{1}.processFcns = {'removeconstantrows','mapminmax'};

net.inputs{2}.processFcns = {'removeconstantrows','mapminmax'};



% Prepare the Data for Training and Simulation

  [x,xi,ai,t] = preparets(net,X,{},T);



% Setup Division of Data for Training, Validation, Testing

net.divideFcn = 'dividerand';    % Divide data randomly

net.divideMode = 'time';    % Divide up every sample

net.divideParam.trainRatio = 70/100;
```

```
net.divideParam.valRatio = 15/100;

net.divideParam.testRatio = 15/100;



% Choose a Performance Function

net.performFcn = 'mse';    % Mean Squared Error



% Train the Network

[net,tr] = train(net,x,t,xi,ai);

netNarx = net;
```

我們利用 Neural Fitting 應用程式同樣地訓練一個雙層的前饋(feed-forward)網絡。

藉由比較 NARX 網絡與 Fitting 網絡在各種潮湧的表現，我們可以評估類神經網絡在風浪事件時的表現(圖 8)。

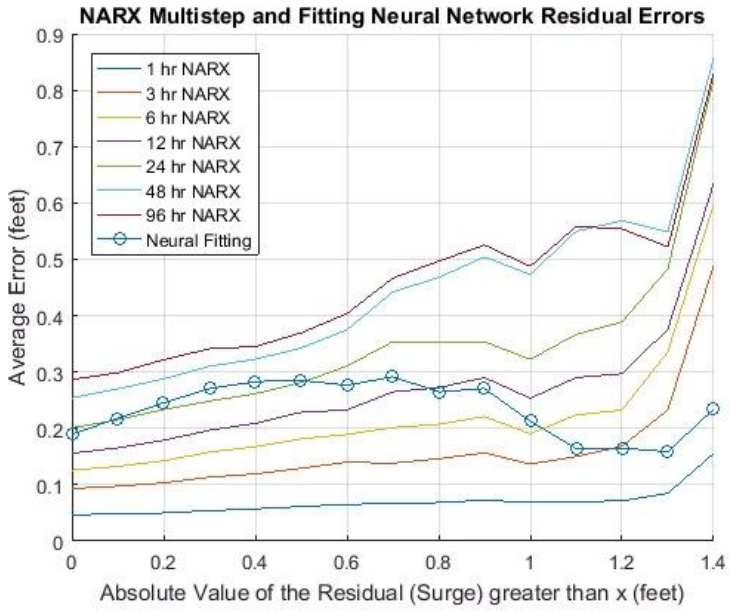

圖 8. Fitting 與 NARX 類神經網絡在潮湧函式的表現。

可以看到在預測 12 小時內的中度潮湧時，NARX 比 Fitting 網絡早了幾個步驟提出較低平均值錯誤。但在對 24 小時以上的預測，NARX 網絡的表現則不如 Fitting 網絡。

以風來預測前 12 個小時的潮水水位時，使用的是 NARX 網絡。在第 12 至 24 小時之間，我們將 NARX 與 Fitting 網絡的結果進行線性內插。而在第 24 小時之後，我們則採用 Fitting 網絡。

將結果繪出之後，可以看到在風浪事件期間的預測跟實際水位相符 (圖 9 與圖 10)。

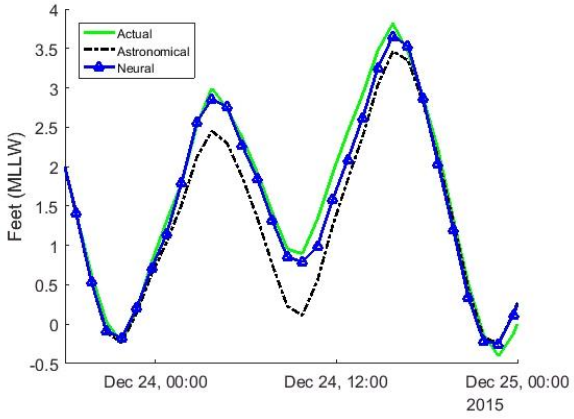

圖 9. 圖中顯示在風浪造成水位上升時，神經網絡預測結果與實際水位相符。

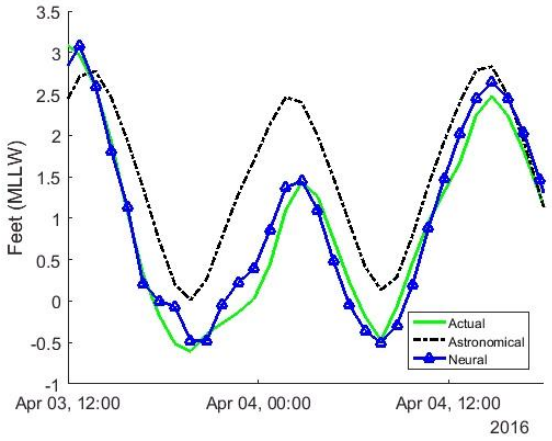

圖 10. 圖上顯示在風浪造成水位下降時，神經網絡預測結果與實際水位相符。

步驟 4：將解析佈署至雲端

我們使用 Fitting 與 NARX 神經網絡來編寫 MATLAB 程式，並將其轉檔佈署到 ThingSpeak 來預測及展示實際測量與預測而來的潮水水位(圖 11)。

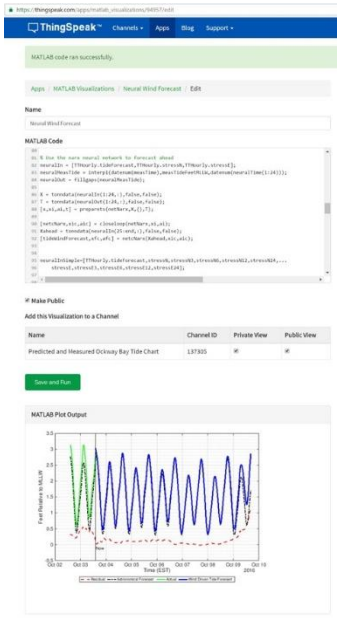

圖 11. ThingSpeak MATLAB 視覺化介面，可以同時查看 MATLAB 程式碼。

步驟 5：依需求執行解析與視覺化

MATLAB 程式碼依需求產生潮湧預測圖(圖 12)。

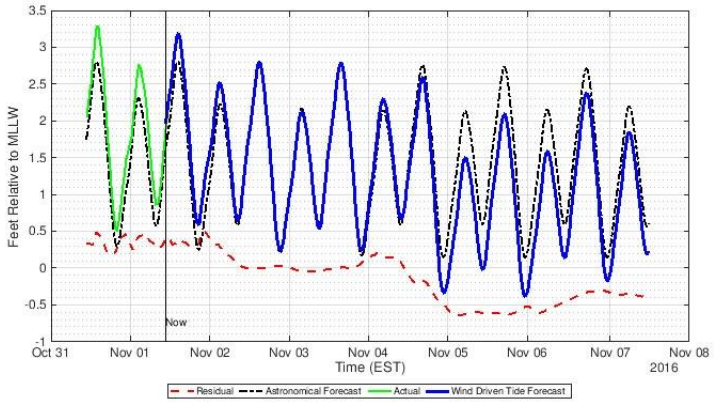

圖 12. 以 MATLAB 機器學習與 ThingSpeak 為基礎所製作的風成潮汐預測圖。神經網絡的輸入資料包含美國國家器象局(National Weather Service)的風速及風向的預測、美國國家海洋暨大氣總署(NOAA)的近期風速與風向資料、利用 UTide 計算出的天文潮汐預測值。

我們在 MATLAB 中，撰寫 thingSpeakRead 程式碼就能從 ThingSpeak 讀取資料，或透過 websave 從其他來源讀取資料。接著利用 timetable 函式將資料結合，並執行類神經網絡來產生預測。.
我們可以選擇是否公開這些圖表，而這些圖表也可以很容易地被嵌入在客製化的網站中。

摘要

我們在這裡展示了如何在不用到網路或佈署到網路設備的情況下，將先進的 IoT 解析系統進行原型化及轉檔佈署的工作。我們發現若是利用 MATLAB 函式，以及像是類神經網絡工具箱(Neural Network Toolbox)應用程式所提供的強大功能，可以降低整體開發耗費的心力，使得單憑一名工程師就可以實現 IoT 解析系統的設計開發，而不需具備特殊的網路開發能力或專門的統計與機器學習知識。

這個 IoT 解析工作流程可以很輕易地被套用在其他資料解析與機器學習應用，像是預測性維護或電力負載預測中。