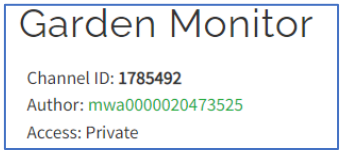
Challenge 2

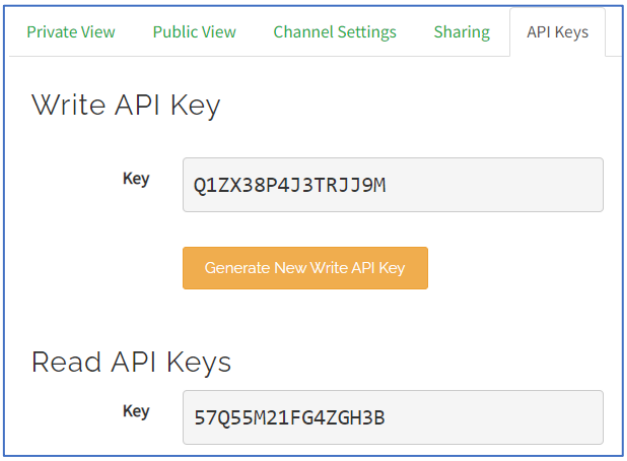
Daily Garden Report

Step 1: Find your channel ID API keys

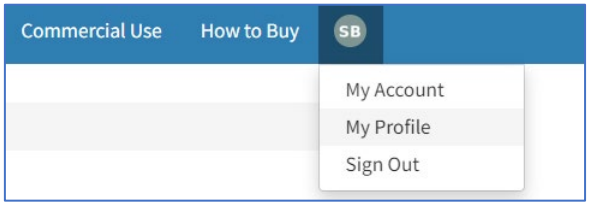
For this challenge activity, you will need your channel ID and the two keys that will help with reading data and sending emails. If you haven't already, copy down your channel ID:



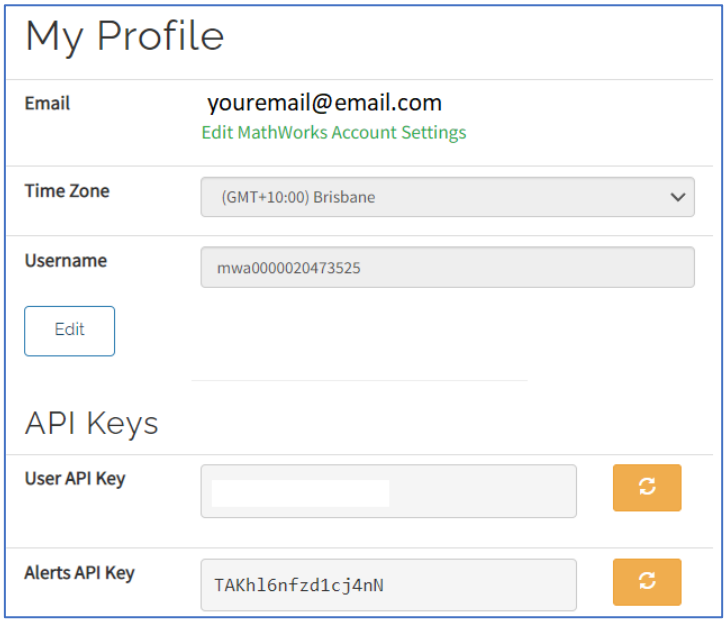
The first one is the Read Key for getting data from your channel. Navigate to the API Keys tab and make a note of the Read Key (I suggest copying it to a digital notebook)



The email alert key is a little harder to find, click on the little circle in the top right corner to navigate to 'My Profile'.

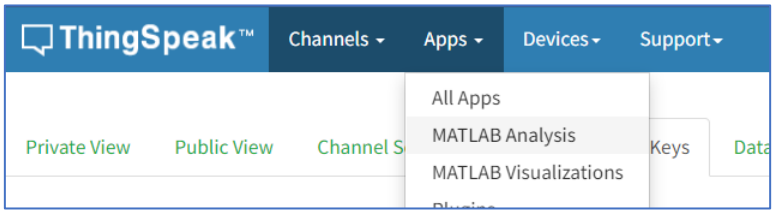


You should now see a screen that looks something like the following image. Copy the 'Alerts API Key' into your OneNote or Word file. If there is no 'Alerts API Key' when you open this page, just click the little orange refresh button next to it and a key should appear.

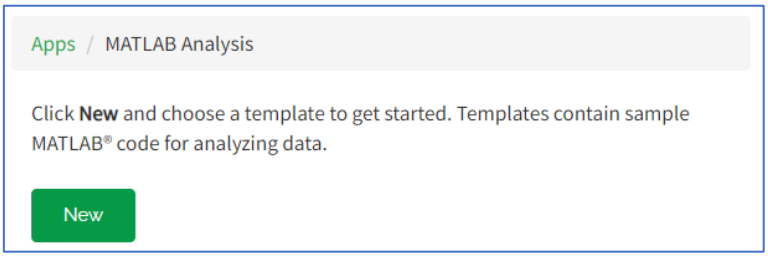


Step 2: Make a weather report

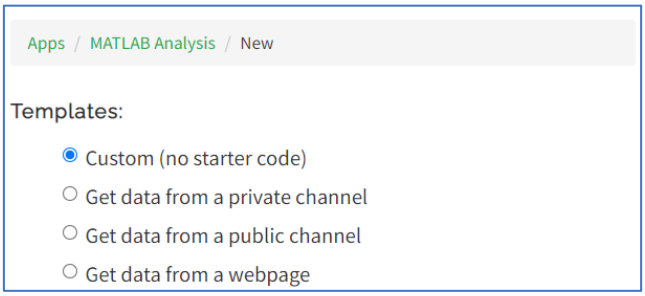
To send a weather report email, we are going to write some code in a programming language called MATLAB. This language is widely used in engineering and is taught in first year at JCU. Start by using the top navigation bar to navigate to Apps > Matlab Analysis



Then click “New” to make a blank MATLAB file:



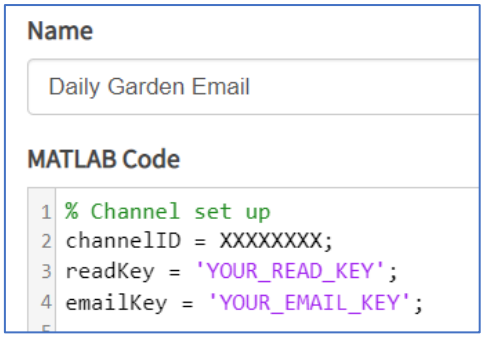
We like a challenge so select ‘Custom (no starter code)’ from the templates options, then scroll down and click ‘Create’.



This will open up a blank file for writing code. Give it a title, and create some variables to store your channel ID, read key, and alert key. Make sure you have copied your channel ID and keys correctly, otherwise you’ll run into problems.

A few notes about the Matlab language at this point:

- You can use the % sign to write comments
- Each line of code should end with a semicolon ; (otherwise it will print stuff out for every line!)



To read data from your channel, you will need to use the 'thingSpeakRead' function. Here's an example:

```
% Read in the data
data = thingSpeakRead(channelID, 'ReadKey', readKey, 'NumPoints', 48);
disp(data);
```

Let's break the **thingSpeakRead** function down so that we can understand how it works.

- The first input is the **channelID**, which is your variable where you stored your channel ID earlier.
- After this first input, we have to explicitly tell the **thingSpeakRead** function what the rest of the inputs are. The second input '**ReadKey'** tells Matlab that the third input **readKey** is the key it should read data from.
- The fourth input '**NumPoints'** tells Matlab that the sixth input **48** is the number of data points it should collect. This means that Matlab will gather the last 48 values. If you wanted to collect more values, you can simply increase this number.

To summarise: this line of code tells Matlab to read 48 data point from all fields, and that these are located at a particular channel with a particular read key.

The **disp(data);** line will allow you to view the structure of the 'data' variable. Click "Save and Run" at the bottom of this code and you will see an output that looks like this:

Output

```
data =

   501    29    81
   500    29    72
   593    30    71
   599    25    74
   575    26    77
   572    27    81
```

We can see that the 'data' variable has three columns: one for soil moisture, one for temperature, and one for humidity. The order may be different for you depending on how you set up your fields. Now we want to grab each of these out of the 'data' variable, so add the following code:

```
% Read in the data
data = thingSpeakRead(channelID, 'ReadKey', readKey, 'NumPoints', 48);
soilMoisture = data(:, 1)
temperature = data(:, 2);
humidity = data(:, 3);
```

The **data(:, x)** notation can be interpreted as "all rows from column x".

Now let's get some basic statistics to use for our weather report. In this example, we'll gather the minimum, maximum, and average values for each of our three parameters. Feel free to explore other statistics functions if you'd like to. Here is the code for getting these statistics for soil moisture:

```
minMoisture = min(soilMoisture);
maxMoisture = max(soilMoisture);
avgMoisture = mean(soilMoisture);
```

Now repeat this for temperature and humidity as well!

You can check that your code is working by displaying the values. One way to do this is using **fprintf**:

```
16 fprintf("Soil moisture min: %d, max: %d, avg: %.2f\n", minMoisture, maxMoisture, avgMoisture);
```

The %d tells Matlab that a whole number will go in that spot, while the %.2f says a number with two decimal places will go in that spot. When the code runs, 'minMoisture' is inserted into the first spot, 'maxMoisture' to the second spot and 'avgMoisture' to the third. The \n parameter just goes to a new line after the message is printed, which looks neater when showing multiple messages.

Now let's start building the message. One approach to doing this is shown below:

```
23 % Build the message
24 message = sprintf("SOIL MOISTURE:\n");
25 message = strcat(message, sprintf("Maximum = %d\n", maxMoisture));
26 message = strcat(message, sprintf("Minimum = %d\n", minMoisture));
27 message = strcat(message, sprintf("Average = %.2f\n\n", avgMoisture));
```

Let's break this code down:

- Line 24 starts the message by creating a heading. The \n parameter goes to a new line.
- Line 25 uses 'strcat' to combine the previous message with another message, in this case the maximum moisture information. The 'sprintf' function works in the same way as fprintf, but saves the message rather than displaying it.
- Line 26 and 27 continue building in the same way as line 25.

You can continue to repeat this process to build up a report that includes temperature and humidity as well. Once you've constructed your message, you can use the **disp** function to check that it looks the way you want it to:

```
37 disp(message);
```

This will print out the message in the Output box:

```
SOIL MOISTURE:
Maximum = 600
Minimum = 500
Average = 543.81

TEMPERATURE:
Maximum = 31
Minimum = 25
Average = 27.79

HUMIDITY:
Maximum = 85
Minimum = 70
Average = 77.71
```

Step 3: Send the email!

Reading from the channel achieved! Now let's turn it into an email. Carefully copy the following code:

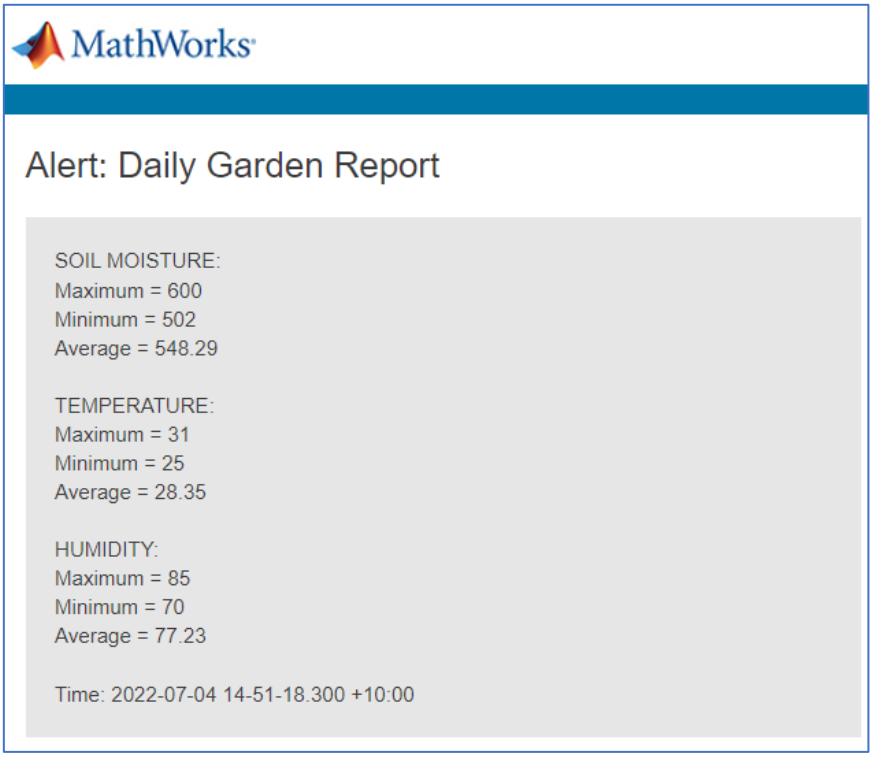
```
37 % Send an email
38 emailSubject = "Daily Garden Report";
39 emailBody = message;
40 alertUrl= "https://api.thingspeak.com/alerts/send";
41 options = weboptions("HeaderFields", {'Thingspeak-Alerts-API-Key', emailKey})
42 result = webwrite(alertUrl, 'subject', emailSubject, 'body', emailBody, options);
```

Now this code is quite complex, so let's briefly look at each line to understand what it's doing.

- Line 38 – we are creating the title for the email we are writing
- Line 39 – we are creating the message body for the email we are sending, in this case this is the message we have already built
- Line 40 – this is the URL that our message will be given to, so that it can then be emailed to us
- Line 41 – we are telling Matlab what our personal emailKey is (that's how it knows which email address to use)
- Line 42 – we are sending our message to the URL that handles ThingSpeak emails, along with some information about the type of message

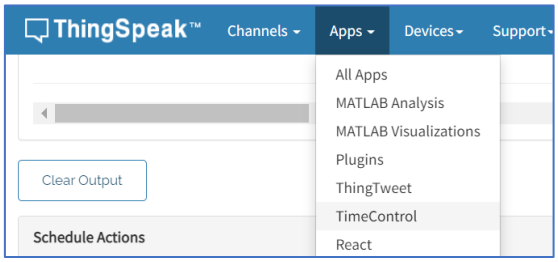
BE VERY CAREFUL THAT YOUR CODE IS CORRECT. ThingSpeak only lets you send an email request twice every 30 minutes. If you make a mistake, you could be waiting for a while to try again! **Check your code with an instructor to be super safe.** Once you have done this, you can click "Save and Run".

After a few moments, you should receive an email. It will go to the email address you used to set up your ThingSpeak account. If you don't receive it, check your Junk/Spam/Clutter folder.

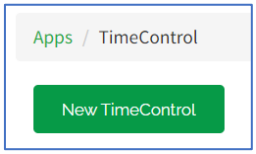


Step 4: Trigger the code to run once per day

At the moment we have to press a button to trigger an email, so it's not very useful for receiving a daily weather report. In this section, we will set up a TimeControl to trigger it automatically at a time that suits you. Navigate to Apps > TimeControl

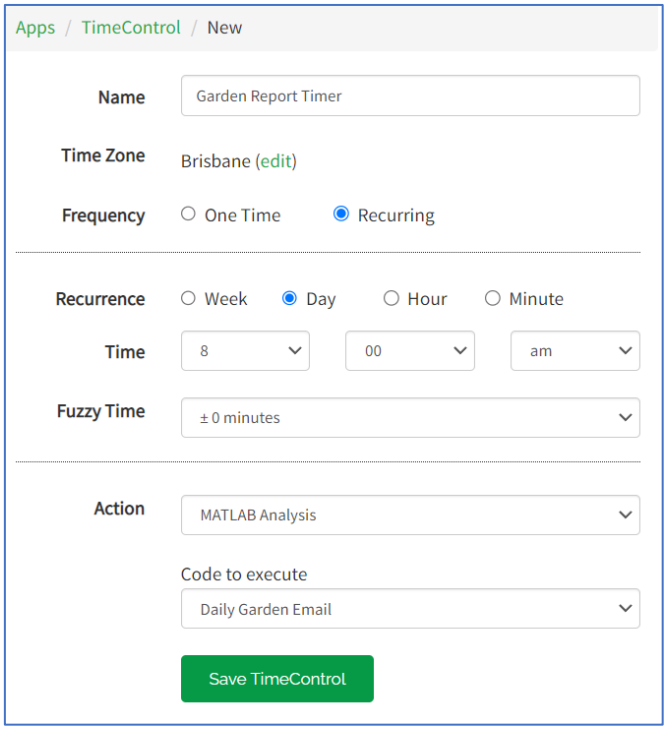


Then click "New TimeControl".



Now we have to set up our 'TimeControl'. Give your TimeControl a name and select "Recurring" for the frequency. Then you can choose how often you want to receive reports. In this example, we will create daily weather reports.

You will eventually want to set a time that you think is convenient, but for now select a time a couple of minutes into the future – that way you can see whether it's working.



When you have finished configuring the settings, click "Save TimeControl" at the bottom of your screen. Once the time you set is reached, you should receive an email shortly after.

When you want to edit your TimeControl, navigate back to Apps > TimeControl. You will see your TimeControl there with an option to 'Edit'. Follow the prompts and you'll be able to change it to a more suitable time.

Step 5: Adjusting the number of data points

Back in step 2, we wrote our code to gather 48 data points. This would assume that there is only one channel update every 30 minute (i.e. the D1 mini only posts data once every 30 minutes).

```
% Read in the data
data = thingSpeakRead(channelID, 'ReadKey', readKey, 'NumPoints', 48);
```

If you want to update the channel more frequently, say 15 minutes, then you will need to change the 48. You can calculate the number of points needed by:

$$numPoints = \frac{24 * 60}{updateTime}$$

Where 24 x 60 is the number of minutes per day and 'updateTime' is the time between two updates in minutes.

You will likely also need to revisit your D1 mini code to adjust the sleep/delay time between measurements to the required interval. Garden parameters won't change too quickly, so consider this when choosing a time interval.

Additional Activities

Got everything working but still have time to spare? Here's some suggested extensions to add to your system:

- Reduce repetition in your code through using functions to gather the statistics of interest and/or for building the message. Information about how to write functions in Matlab can be found here: https://au.mathworks.com/help/matlab/ref/function.html (*this extension is only recommended if you've used functions in this or other programming languages*)
- Generate a weekly report that shows the statistics for each day over the week in a neat format.
- Calculate the *heat index* (which gives a 'feels like' temperature metric that indicates how intense the heat is). Include this in your weather report. The heat index formula is shown below.

Heat Index in degrees Fahrenheit

The heat index in degrees Fahrenheit can be calculated as

$$t_{HI} = -42.379 + 2.04901523\,t + 10.14333127\,\varphi$$
$$- 0.22475541\,t\,\varphi - 0.00683783\,t^2 - 0.05481717\,\varphi^2$$
$$+ 0.00122874\,t^2\,\varphi + 0.00085282\,t\,\varphi^2 - 0.00000199\,(T\,\varphi)^2$$

where

t_{HI} = heat index (°F)

t = air temperature (°F) ($t > 57$°F)

φ = relative humidity (%)

Noting that Celsius can be converted to Fahrenheit (and vice versa) using:

$$°C = (°F - 32) \times \frac{5}{9}$$

The heat index can be interpreted based on the following table. You could rate the heat based on this!

Heat Index	Risk Level
Less than 91°F (32°C)	Lower (caution)
91° to 103°F (32° to 39°C)	Moderate
103° to 115°F (39° to 46°C)	High
Greater than 115°F (46°C)	Very high to extreme