

Connecting to ThingSpeak

A Step-By-Step Tutorial

For Creating a Reaction Using an Ultrasonic Sensor and an LED Light

Tutorial Overview

This is a complete guide to sending sensor data to using ThingSpeak, an Internet of Things (IoT) platform, to create a reaction based on the data.

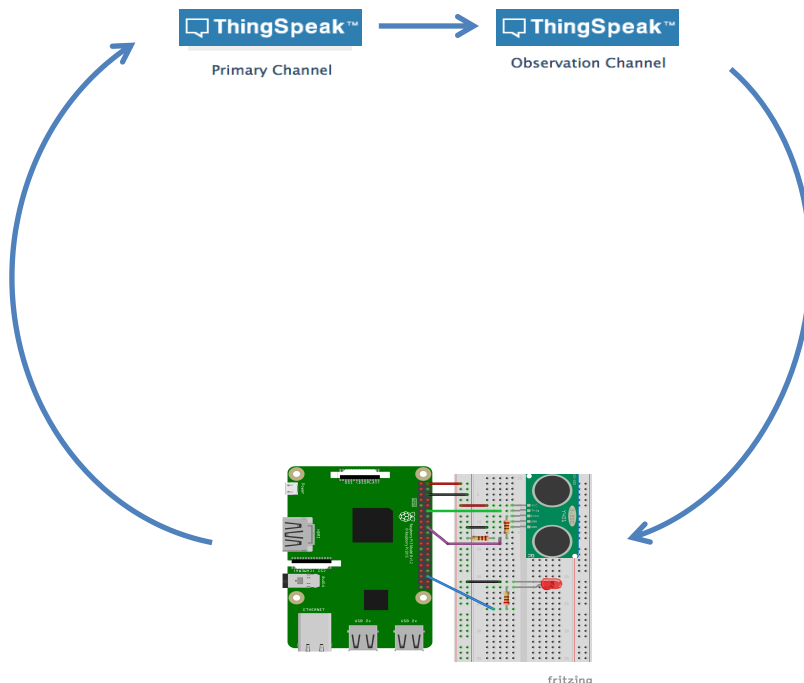
This tutorial is written for beginners with no coding, wiring, or programming experience. By the end of this tutorial, users will be able to turn on an LED light when the ultrasonic sensor senses an object less than twenty centimeters away.

Tutorial Outline

This tutorial is broken into four sections with various parts. The table of contents is listed below.

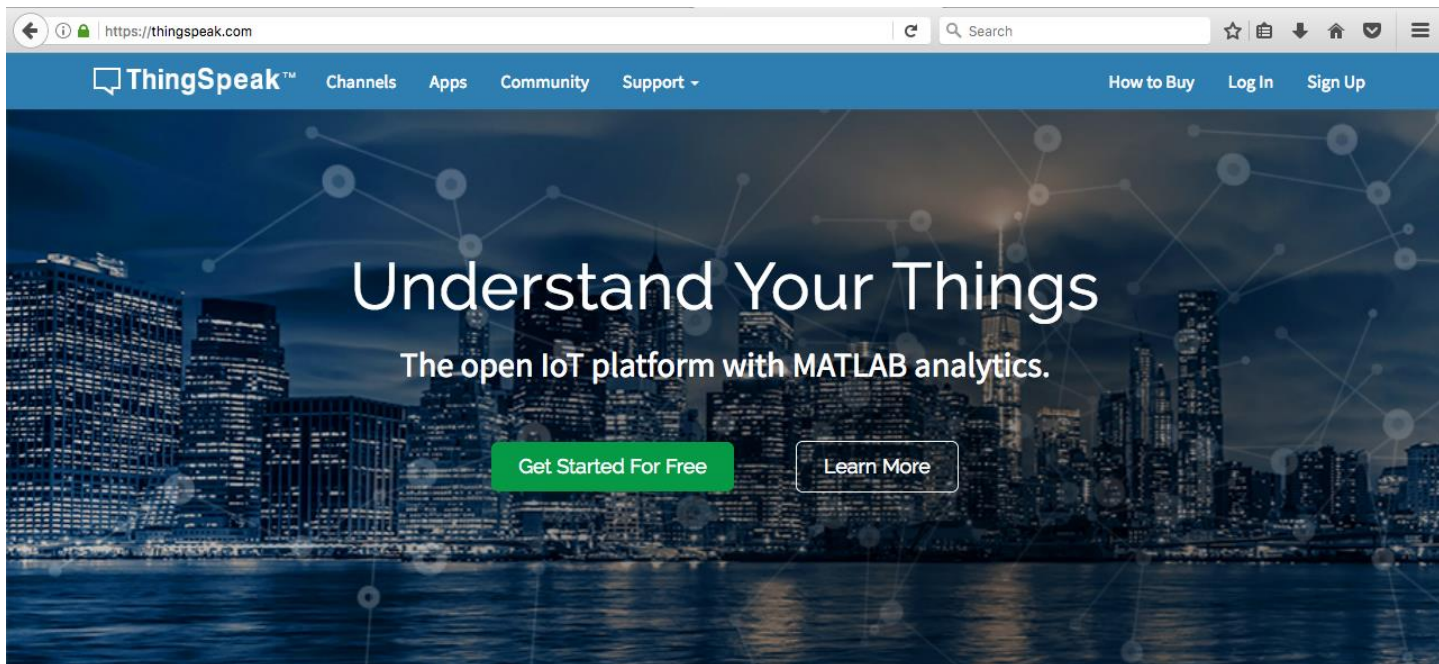
<u>Section Title</u>	<u>Pages</u>
Section 1: Signing Up for ThingSpeak	4-8
Section 2: Setting Up the ThingSpeak Channels	10-14
Part 1: Primary Channel Setup	
Part 2: Observation Channel Setup	
Section 3: Setting Up MATLAB & ThingSpeak Reaction	16-23
Part 1: MATLAB Analysis Setup	
Part 2: Reaction Setup	
Section 4: Code and Sensor Setup	25-39
Part 1: Wiring the Pi	
Part 2: Setting Up the Code	
Part 3: Watching it All Work Together	
Python Code (Text Version)	40-41

The Finished Product

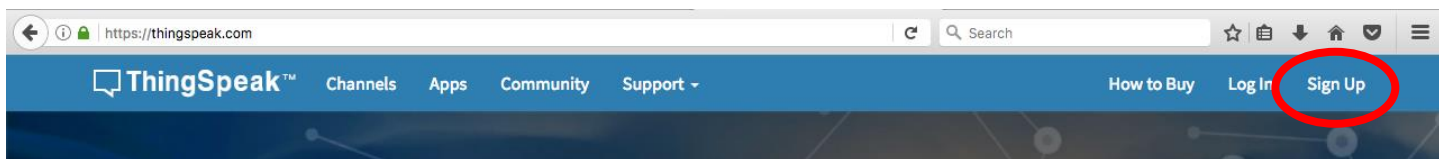


Section 1: Signing Up for ThingSpeak

Step 1: Go to www.thingspeak.com. Your browser should look like this:



Step 2: Click Sign Up in the upper right hand corner.



Step 3: Fill out your email address, username, password, country, first name, and last name.

ThingSpeak™ Channels Apps Community Support ▾ How to Buy Log In Sign Up

Sign up for ThingSpeak

The ThingSpeak service is operated by MathWorks. In order to sign up for ThingSpeak, you must create a new MathWorks Account or log in to your MathWorks Account.

Create MathWorks Account

Email Address
?
United States
First Name
Last Name

By clicking continue, you agree to our [privacy policy](#)

Cancel Continue

The diagram illustrates the ThingSpeak ecosystem. On the left, several smart connected devices (represented by icons with Wi-Fi symbols) are connected to a central router. This router is connected to a cloud labeled 'DATA AGGREGATION AND ANALYTICS' with the ThingSpeak logo. A bidirectional arrow connects this cloud to a computer monitor on the right labeled 'MATLAB' with the text 'ALGORITHM DEVELOPMENT SENSOR ANALYTICS' below it.

Step 4: Click Continue.

Step 5: Your screen will look like this:

ThingSpeak™ Channels Apps Community Support ▾ How to Buy Log In Sign Up

Sign up for ThingSpeak

The ThingSpeak service is operated by MathWorks. In order to sign up for ThingSpeak, you must create a new MathWorks Account or log in to your MathWorks Account.

Verify Your MathWorks Account

To finish creating your account, complete the following steps:

1. Go to your inbox for **wagne1at@gmail.com**.
2. Click the link in the email we sent you.

Once you've done this, click Continue.

Didn't get the email?

1. Check your spam folder.
2. [Send me the email again.](#)
3. Contact Customer Support if you still do not have the email

The diagram is identical to the one in Step 3, showing smart connected devices connected to a central router, which is connected to a cloud for 'DATA AGGREGATION AND ANALYTICS' (ThingSpeak), which in turn is connected to a 'MATLAB' computer monitor for 'ALGORITHM DEVELOPMENT SENSOR ANALYTICS'.

Step 6: Open a new tab. Check your email address for a new message from "service" with a subject line reading "Verify email address." Open this email and move to Step 7.

***Note: If the email is not in your inbox, check your spam folder. If it's not there, choose the "Send me the email again" option in the window with ThingSpeak (see image below).*

Sign up for ThingSpeak

The ThingSpeak service is operated by MathWorks. In order to sign up for ThingSpeak, you must create a new MathWorks Account or log in to your MathWorks Account.

Verify Your MathWorks Account

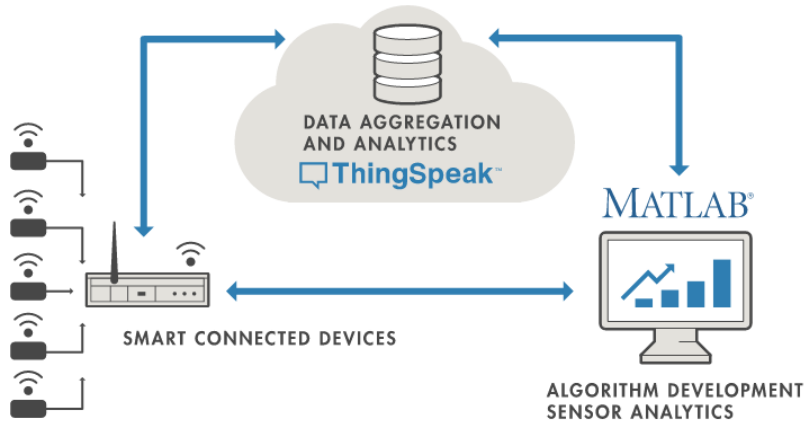
To finish creating your account, complete the following steps:

1. Go to your inbox for **wagne1at@gmail.com**.
2. Click the link in the email we sent you.

Once you've done this, click Continue.

Didn't get the email?

1. Check your spam folder.
2. **Send me the email again.**
3. Contact [Customer Support](#) if you still do not have the email



Step 7: The message should look like the image below. Click "Verify Your Email."

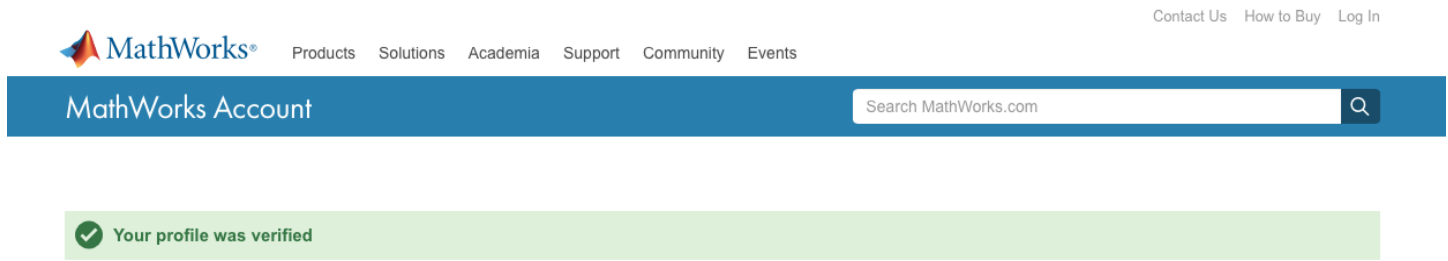
Thank you for registering with MathWorks!

To complete the registration process, verify your email address by clicking this link:

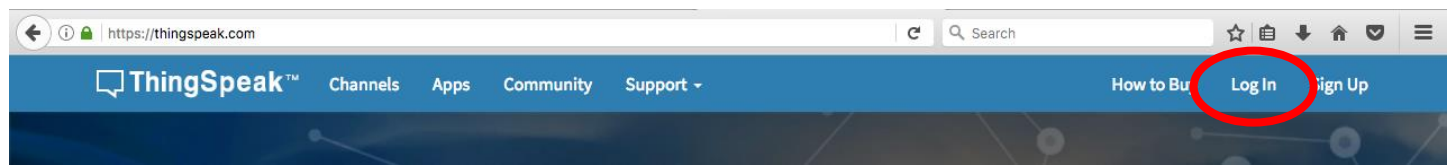
Verify your email

Sincerely,
MathWorks Customer Service Team

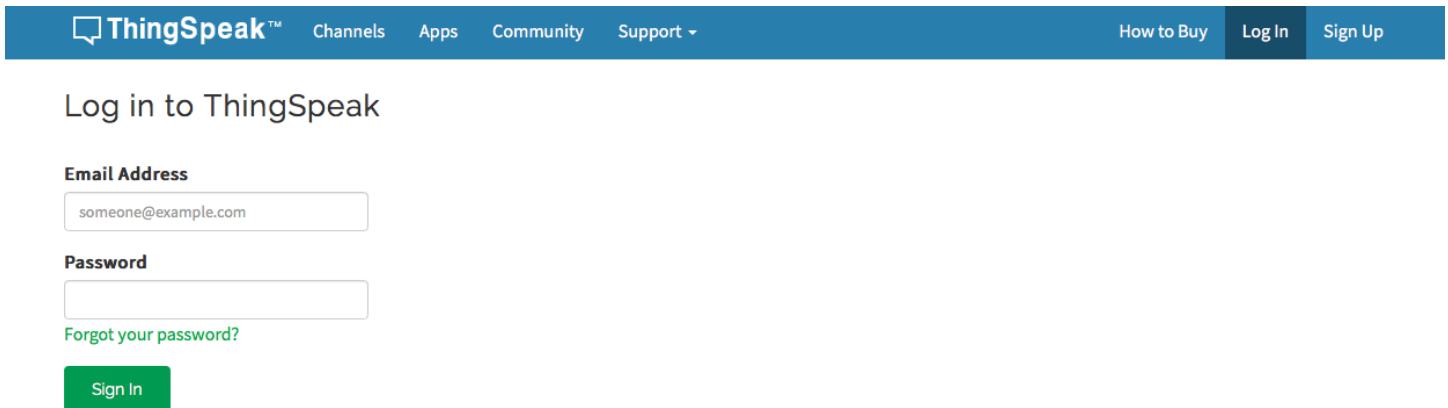
Step 8: A confirmation page will appear. Notice that this has opened a new tab to MathWorks Inc, the parent of ThingSpeak.



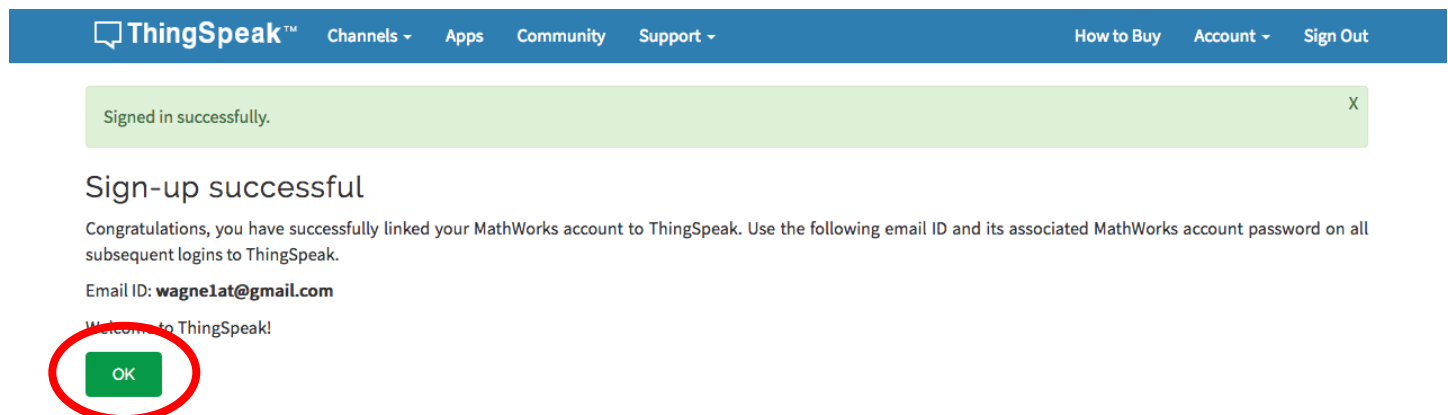
Step 9: Go back to www.thingworks.com. Click Log In.



Step 10: Type in your email address and password. Then click "Sign In."



Step 11: A confirmation page should appear. Click OK to proceed.



Step 12: Agree to the terms of use.

The screenshot shows the ThingSpeak website's Terms of Use page. At the top is a blue navigation bar with the ThingSpeak logo and links for Channels, Apps, Community, Support, How to Buy, Account, and Sign Out. Below the navigation bar, the page title is "ThingSpeak Terms of Use". The main text states: "ThingSpeak Terms of Use have changed. We require that you agree to the [Terms of Use](#) and [Privacy Policy](#) before continuing." Below this text are two buttons: a green "Agree to Terms" button and a red "Decline and Sign Out" button. The "Agree to Terms" button is circled in red.

Step 13: Congratulations! If your screen looks like the one below, you have successfully registered for ThingSpeak. You are ready to proceed to Section 2.

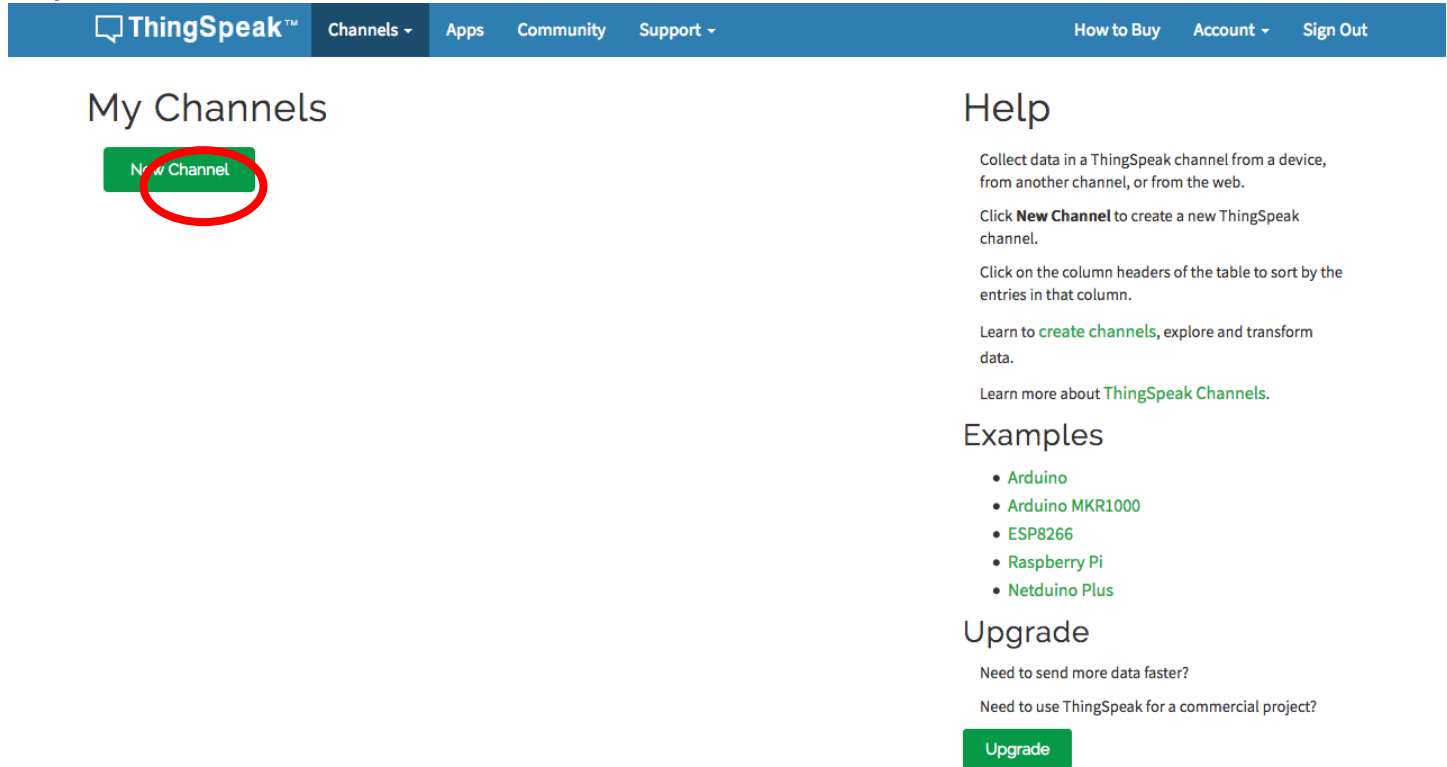
The screenshot shows the ThingSpeak website's "My Channels" page. At the top is a blue navigation bar with the ThingSpeak logo and links for Channels, Apps, Community, Support, How to Buy, Account, and Sign Out. Below the navigation bar, the page title is "My Channels". There is a green "New Channel" button. To the right of the "My Channels" section is a "Help" section with the following text: "Collect data in a ThingSpeak channel from a device, from another channel, or from the web. Click **New Channel** to create a new ThingSpeak channel. Click on the column headers of the table to sort by the entries in that column. Learn to [create channels](#), explore and transform data. Learn more about [ThingSpeak Channels](#)." Below the "Help" section is an "Examples" section with a list of links: "Arduino", "Arduino MKR1000", "ESP8266", "Raspberry Pi", and "Netduino Plus". Below the "Examples" section is an "Upgrade" section with the text: "Need to send more data faster? Need to use ThingSpeak for a commercial project?" and a green "Upgrade" button.

Section 2: Setting Up the ThingSpeak Channels

We are going to set up two channels in ThingSpeak. The first channel will be called the “Primary Channel” because it will collect data that it receives from an ultrasonic sensor *and* it will also send data to another channel. The second channel will be called the “Observation Channel” because it will allow us to react and observe the data.

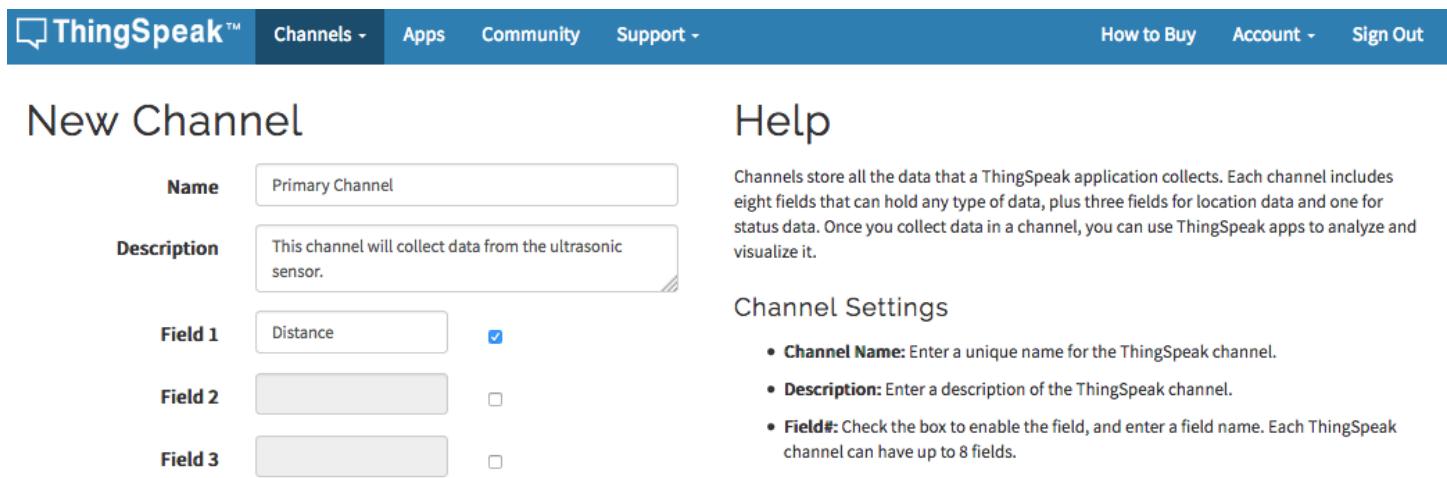
Part 1: Primary Channel Setup

Step 1: Click New Channel



The screenshot shows the ThingSpeak user interface. At the top is a blue navigation bar with the ThingSpeak logo and links for Channels, Apps, Community, Support, How to Buy, Account, and Sign Out. Below the navigation bar, the page is titled "My Channels". A green button labeled "New Channel" is highlighted with a red circle. To the right of the "My Channels" section is a "Help" section with text explaining how to collect data and create channels, followed by "Examples" (Arduino, Arduino MKR1000, ESP8266, Raspberry Pi, Netduino Plus) and an "Upgrade" section with an "Upgrade" button.

Step 2: Name the channel, write its description, and create its field. In this tutorial, we will only receive data on one variable (distance), so only one field is needed.



The screenshot shows the "New Channel" form in the ThingSpeak interface. The "Name" field contains "Primary Channel". The "Description" field contains "This channel will collect data from the ultrasonic sensor." Below the description are three "Field" entries. "Field 1" has the name "Distance" and a checked checkbox. "Field 2" and "Field 3" have empty input boxes and unchecked checkboxes. To the right of the form is a "Help" section explaining that channels store all data and include eight fields. Below the help is a "Channel Settings" section with three bullet points: "Channel Name" (unique name), "Description" (description), and "Field#" (checkbox to enable field and enter name, up to 8 fields).

Step 3: Scroll to the bottom and click "Save Channel."

The screenshot shows the 'Save Channel' form in the ThingSpeak interface. At the top, there is a navigation bar with 'ThingSpeak™' and menu items: 'Channels', 'Apps', 'Community', 'Support', 'How to Buy', 'Account', and 'Sign Out'. The form includes several sections: 'Show Location' with 'Latitude' and 'Longitude' input fields (both containing '0.0'); 'Show Video' with radio buttons for 'YouTube' (selected) and 'Vimeo', and a 'Video ID' input field; and 'Show Status' with an unchecked checkbox. At the bottom of the form, a green 'Save Channel' button is circled in red.

Step 4: Your screen should look like the screen below. Next, click the "API Keys" button.

The screenshot shows the 'Primary Channel' page in the ThingSpeak interface. The navigation bar at the top is identical to the previous screenshot. Below the navigation bar, the page title is 'Primary Channel'. On the left, there is a summary of channel information: 'Channel ID: [redacted]', 'Author: [redacted]', and 'Access: Private'. On the right, a description reads: 'This channel will collect data from the ultrasonic sensor.' Below this information is a horizontal menu with buttons for 'Private View', 'Public View', 'Channel Settings', 'Sharing', 'API Keys' (circled in red), and 'Data Import / Export'. Underneath the menu are two buttons: '+ Add Visualizations' and 'Data Export'. To the right of these are two green buttons: 'MATLAB Analysis' and 'MATLAB Visualization'. Below the menu is the 'Channel Stats' section, which shows: 'Created: less than a minute ago', 'Updated: less than a minute ago', and 'Entries: 0'. At the bottom, there is a preview of a 'Field 1 Chart' for the 'Primary Channel', showing a vertical axis labeled 'Distance'.

Step 5: This is where the API Keys and the Channel ID for this particular channel are located. We will need all three of these keys (the Channel ID, the Write API Key and the Read API Key) for our code.

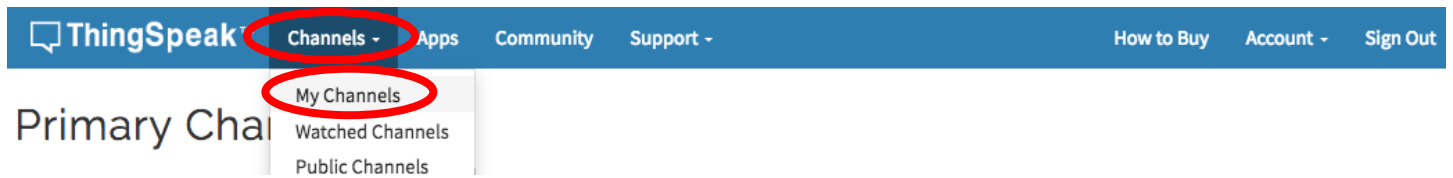
The screenshot shows the 'Primary Channel' page on ThingSpeak. At the top, there is a navigation bar with 'ThingSpeak™', 'Channels -', 'Apps', 'Community', 'Support -', 'How to Buy', 'Account -', and 'Sign Out'. Below the navigation bar, the page title is 'Primary Channel'. Underneath, it displays 'Channel ID: 306964' (circled in red), 'Author:', and 'Access: Private'. A note states 'This will process data from the ultrasonic sensor.' Below this, there are tabs for 'Private View', 'Public View', 'Channel Settings', 'Sharing', 'API Keys' (selected), and 'Data Import / Export'. The 'Write API Key' section shows a key 'FYLP4EYMV9ICFMFQ' (circled in red) and a 'Generate New Write API Key' button. The 'Read API Keys' section shows a key '3S6H9W7ZNN2CWUTY' (circled in red). To the right, there is a 'Help' section with text: 'API keys enable you to write data to a channel or read data from a private channel. API keys are auto-generated when you create a new channel.' Below the help text is the 'API Keys Settings' section with a bulleted list: '• Write API Key: Use this key to write data to a channel. If you feel your key has been compromised, click Generate New Write API Key.' '• Read API Keys: Use this key to allow other people to view your private channel feeds and charts. Click Generate New Read API Key to generate an additional read key for the channel.' '• Note: Use this field to enter information about channel read keys. For example, add notes to keep track of users with access to your channel.' At the bottom right of the page, there is a section for 'API Requests'.

Congratulations! The first channel has been successfully set up. Proceed to Part 2 below.

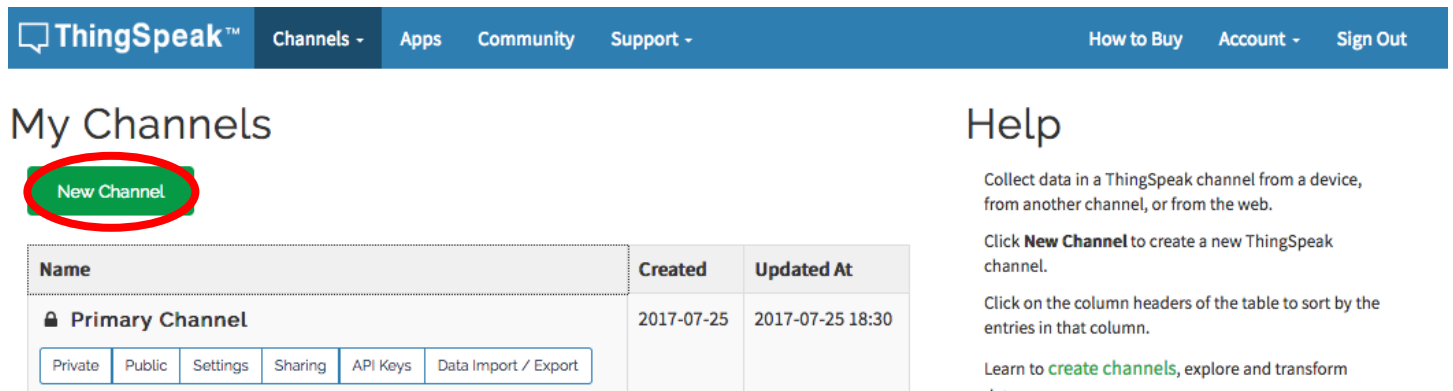
Part 2: Observation Channel Setup

We will now setup the second channel, which we will call the Observation Channel.

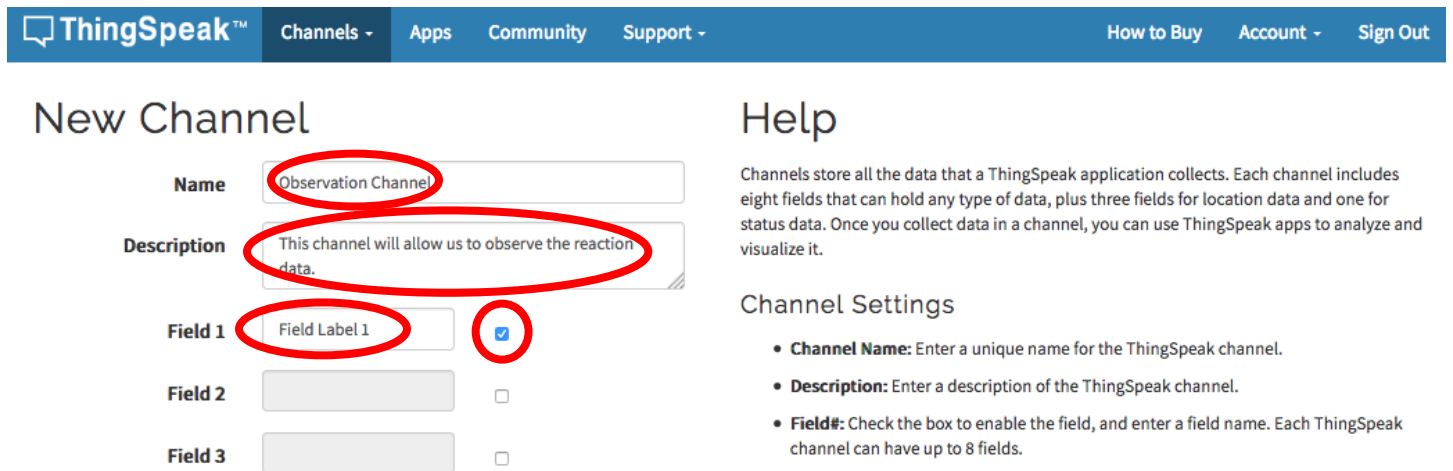
Step 1: Click Channels > My Channels.



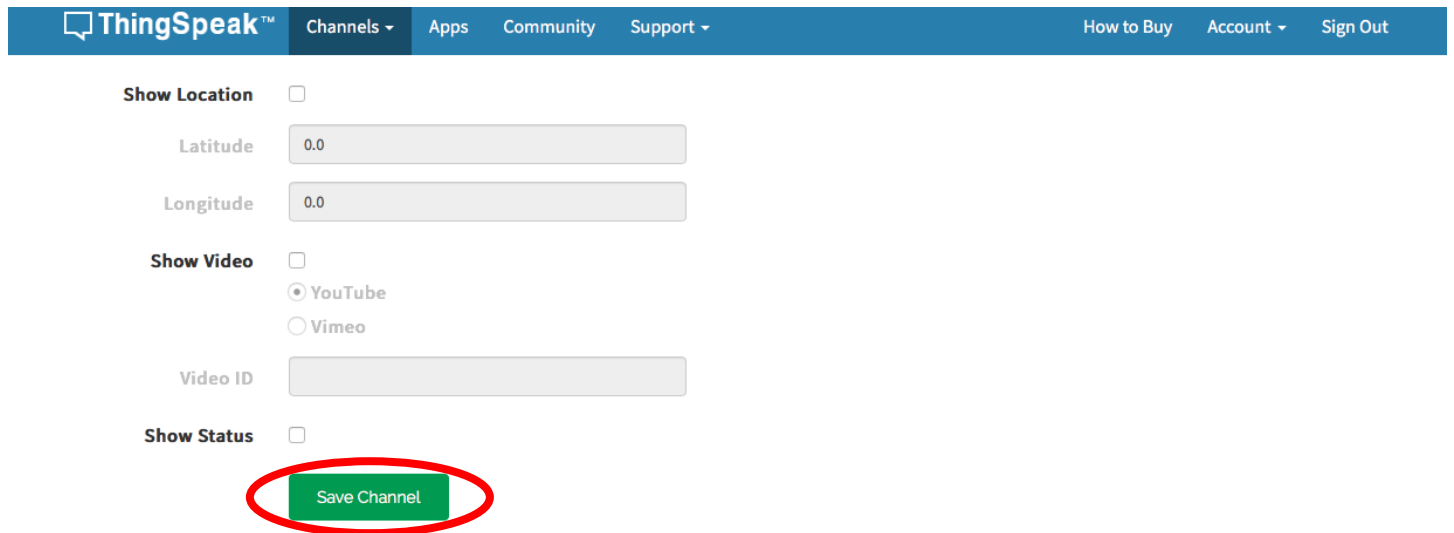
Step 2: Click New Channel.



Step 3: Create the name, description, and field label for this channel.



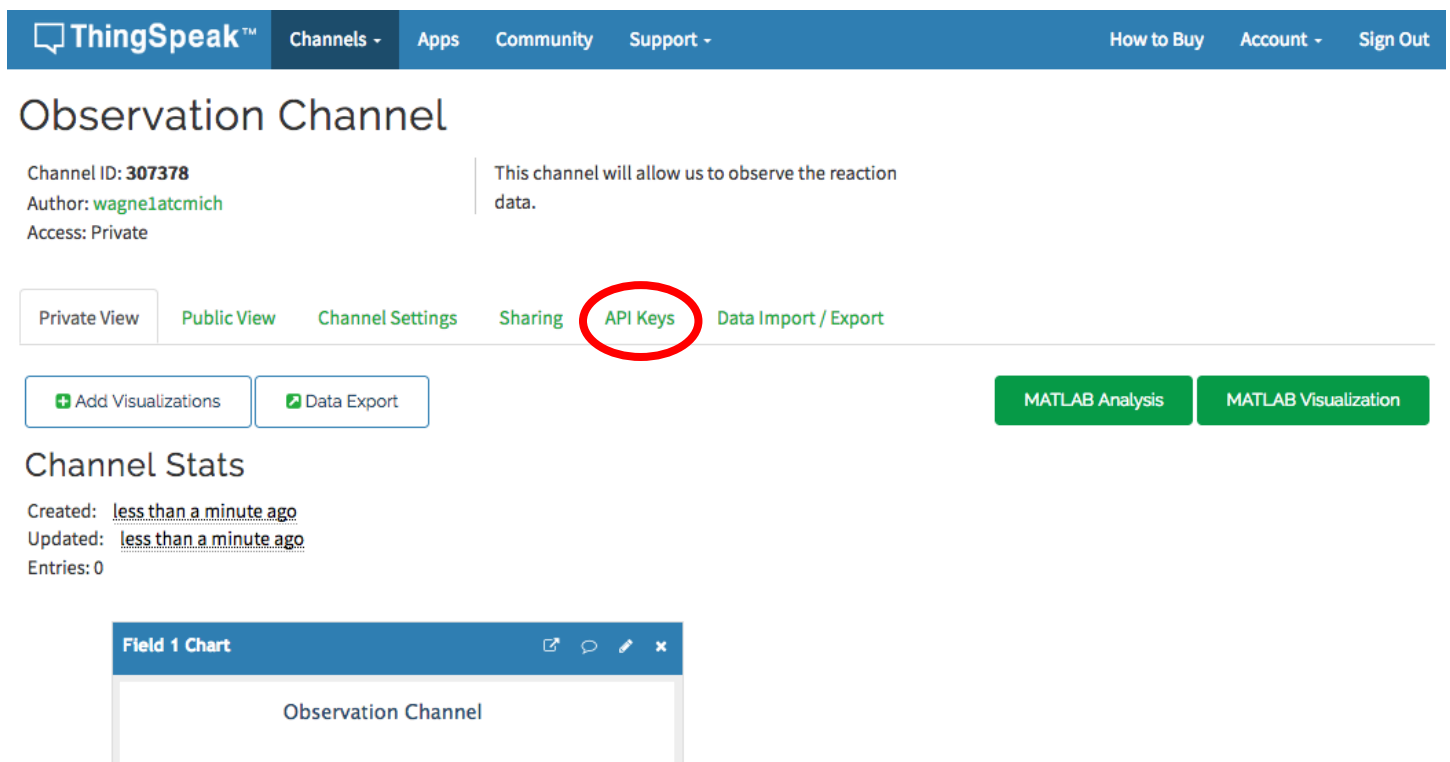
Step 4: Scroll to the bottom and click "Save Channel."



The screenshot shows the 'Save Channel' form in the ThingSpeak interface. The form includes the following fields and options:

- Show Location:** (unchecked)
- Latitude:**
- Longitude:**
- Show Video:** (unchecked)
- Video Source:** YouTube, Vimeo
- Video ID:**
- Show Status:** (unchecked)
- Save Channel:** A green button circled in red.

Step 5: Click API Keys. We will again record the API Keys and Channel ID for this channel.



The screenshot shows the 'Observation Channel' page in the ThingSpeak interface. The page includes the following information and navigation options:

- Channel ID:** 307378
- Author:** wagne1atcmich
- Access:** Private
- Description:** This channel will allow us to observe the reaction data.
- Navigation Tabs:** Private View, Public View, Channel Settings, Sharing, **API Keys** (circled in red), Data Import / Export
- Buttons:** Add Visualizations, Data Export, MATLAB Analysis, MATLAB Visualization
- Channel Stats:** Created: less than a minute ago, Updated: less than a minute ago, Entries: 0
- Field 1 Chart:** A placeholder for a chart titled 'Field 1 Chart' for the 'Observation Channel'.

Both channels have successfully been set up. In the next section, we will setup an app that will allow us to receive, process, and send the data.

Section 3: Setting Up MATLAB & ThingSpeak Reactions

This section sets up a new reaction. Reactions allow us to set a condition to produce a result. In our case, we want to stop our car when it senses an oncoming object. Our condition is: "*If* **the channel reads a distance less than 20**, *then* **stop the car.**" The underlined words are part of the condition statement. The bolded words indicate the variables that we program and change. In order to set up the condition, we need to first write a MATLAB Analysis code (Part 1) below. Once this is setup, we can set up the condition statement in the Reaction portion (Part 2) below.

Part 1: MATLAB Analysis Setup

Step 1: Click Apps.



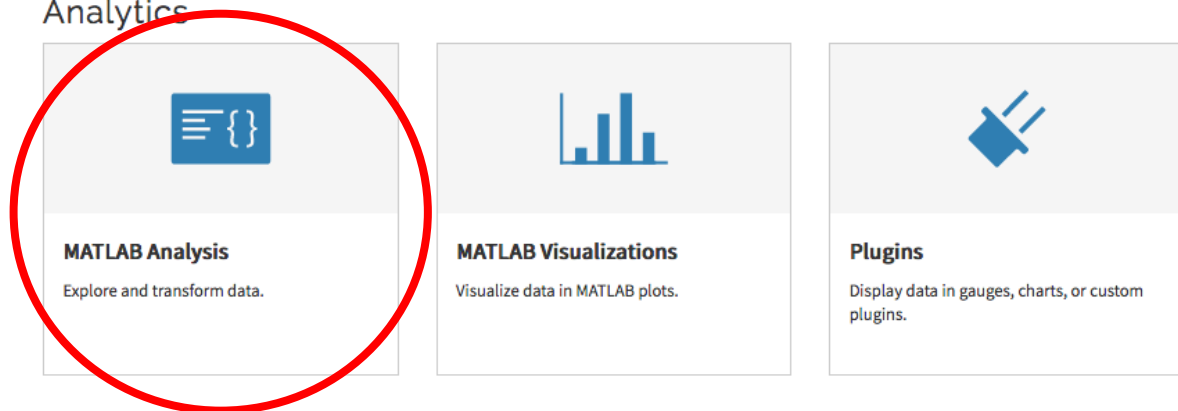
Step 2: Click MATLAB Analysis.



Apps

ThingSpeak channels store data. Upload data from the web or send data from devices to a ThingSpeak channel. Use these apps to transform and visualize data or trigger an action. See [Tutorial: ThingSpeak and MATLAB](#) to create a channel. [Learn more](#) about MATLAB® inside ThingSpeak.

Analytics



Step 3: Click "New."



[Apps](#) / [MATLAB Analysis](#)

Click **New** and choose a template to get started. Templates contain sample MATLAB® code for analyzing data.



Help

MATLAB Analysis

- Explore data collected in a channel or scraped from a website
- Find and remove bad data
- Convert data to different units
- Calculate new data

Step 4: Choose the template to be "Custom (no starter code)." Then click "Create."

ThingSpeak™ Channels - Apps Community Support - How to Buy Account - Sign Out

Templates:

- Custom (no starter code)
- Get data from a private channel
- Get data from a public channel
- Get data from a webpage

Examples: Sample code to analyze and transform data

- Calculate average humidity
- Calculate dew point
- Convert Celsius to Fahrenheit
- Eliminate data outliers
- Convert Fahrenheit to Celsius
- Calculate hourly max temperature
- Replace missing values in data
- Analyze text for the most common color
- Scrape web data for ships at the Boston port
- Scrape web temperature data

Templates

MATLAB Analysis templates provide sample MATLAB code for analyzing data and writing it to a ThingSpeak channel. If you are new to MATLAB, you can learn interactively at [MATLAB Academy](#).

Examples

To see MATLAB Analysis in action, select the example and click **Create**.

These examples read data from public ThingSpeak channels:

- **Calculate average humidity**, and write the data to a new channel.
- **Calculate dew point** from temperature and humidity data, and write data to a new channel.
- **Convert Celsius to Fahrenheit**, and write data to a new channel.
- **Eliminate data outliers** from temperature data, and write data to a new channel.
- **Convert Fahrenheit to Celsius**, and write data to a new channel.
- **Calculate hourly max temperature**, and write data with the timestamps to a new channel.
- **Replace missing values in data** of a weather channel, and clean the data using a list wise deletion algorithm. Then display the missing values, or write data to a new channel.
- **Analyze text for the most common color** requested on the public Cheerlights channel, and write data to a new channel.

These examples scrape data from websites:

- **Scrape web data for ships at the Boston port** from the MarineTraffic® website, count the number of ships in Boston port, and write data to a new channel.
- **Scrape web temperature data** from the National Weather Service website, and write data to a new channel.

New to MATLAB?

Step 5: Our browser now looks like this:

ThingSpeak™ Channels - Apps Community Support - How to Buy Account - Sign Out

Apps / MATLAB Analysis / Custom (no starter code) 1

Help

My Channels Documentation

Name

Custom (no starter code) 1

MATLAB Code

1 % Enter your MATLAB Code below

Step 6: We will now enter our MATLAB Code. The portions circled in red indicate information that you will need to fill in from Google Sheets that correspond to your unique channels. The blue circles (●) indicate where the proper API Keys must be added (see Step 7).

****Note:** We recommend typing in the following code by hand to minimize syntax errors on MATLAB.

****Also Note:** Type the code in exactly as it appears (including all punctuation). It is case sensitive.

```
ChannelID=●;  
Field=1;  
write='●';  
thingSpeakwrite(ChannelID,111,'fields',Field,'writeKey',write);
```

Step 7: Once your code is in place, we now need to input the correct API Keys.

The screenshot shows the ThingSpeak interface. At the top is a navigation bar with 'Channels', 'Apps', 'Community', 'Support', 'How to Buy', 'Account', and 'Sign Out'. Below the navigation bar, there are tabs for 'My Channels' and 'Documentation'. A 'New Channel' button is visible. The main content area is divided into two sections: 'Name' and 'MATLAB Code'. The 'Name' section has a text input field containing 'Custom (no starter code) 1'. The 'MATLAB Code' section has a code editor with the following code:

```
1 % Enter your MATLAB Code below  
2 ChannelID=307290;  
3 Field=1;  
4 write='ZADHAQ44UIZ6JC98';  
5 thingSpeakwrite(ChannelID,111,'fields',Field,'writeKey',write);
```

Red circles highlight '307290' in line 2 and 'ZADHAQ44UIZ6JC98' in line 4. Blue circles highlight 'ZADHAQ44UIZ6JC98' in line 4 and 'ZADHAQ44UIZ6JC98' in the 'Write API Key' field of the 'Observation Channel' in the 'Channel Info' section. Red arrows point from the red circles in the code to the corresponding fields in the 'Channel Info' section. Blue arrows point from the blue circles in the code to the corresponding fields in the 'Channel Info' section.

The 'Channel Info' section shows two channels:

- Primary Channel**
Channel ID: 306964
Access: Private
Read API Key: 3S6H9W7ZNN2CWUTY
Write API Key: FYLP4EYMV9ICFMFQ
Fields:
1: Distance
- Observation Channel**
Channel ID: 307290
Access: Private
Read API Key: XZ3YGG9DFZQSQN70
Write API Key: ZADHAQ44UIZ6JC98
Fields:
1: Field Label 1

Step 7: Click "Save and Run."

ThingSpeak™ Channels - Apps Community Support -

Apps / MATLAB Analysis / Custom (no starter code) 1

Name

Custom (no starter code) 1

MATLAB Code

```
1 % Enter your MATLAB Code below
2 ChannelID=307290;
3 Field=1;
4 Write='ZADHAQ44UIZ6JC98';
5 thingSpeakWrite(ChannelID,111,'fields',Field,'WriteKey',Write);|
```

Save and Run

Step 8: If you did this correctly, a green message should appear at the top of the screen (you may need to scroll up to the top), reading "MATLAB code ran successfully."

ThingSpeak™ Channels - Apps Community Support - How to Buy Account - Sign Out

MATLAB code ran successfully. X

Part 2: Reaction Setup

Step 1: We now need to set up a reaction that uses our new MATLAB code. This will allow us to program changes to our car based on the data that it receives. To do this, first click "Apps."

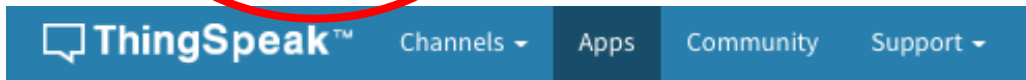


Step 2: Scroll to the bottom and click "React."



Actions

A grid of six action cards. Each card has a blue icon at the top, a title, and a brief description. The cards are: 1. ThingTweet (Twitter icon): Connect a device to Twitter® and send alerts. 2. TweetControl (Twitter icon with sliders): Listen to the Twittersverse and react in real time. 3. TimeControl (Gear and clock icon): Automatically perform actions at predetermined times with ThingSpeak apps. 4. React (Three gears with a play button): React when channel data meets certain conditions. This card is circled in red. 5. TalkBack (Speech bubble with gears): Queue up commands for your device. 6. ThingHTTP (Cloud with up and down arrows): Simplify device communication with web services and APIs.



Apps / React

New React

Step 4: Fill out the name of this reaction. We called ours Action 1 (Stop Car).



Apps / React / New

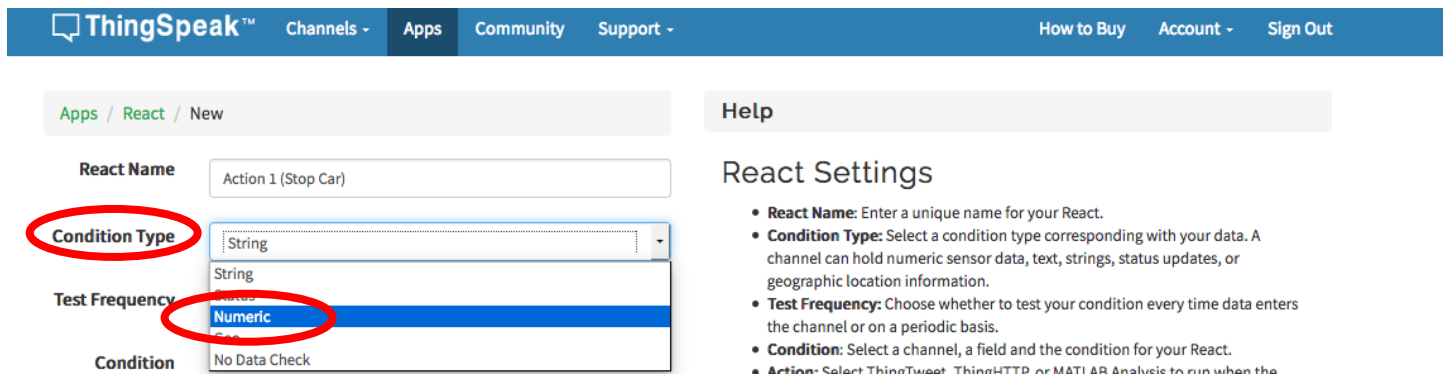
React Name

Help

React Settings

React Name: Enter unique name for your React

Step 5: Change the condition type from String to Numeric.



Apps / React / New

React Name: Action 1 (Stop Car)

Condition Type: String (dropdown menu open, Numeric selected)

Test Frequency: No Data Check

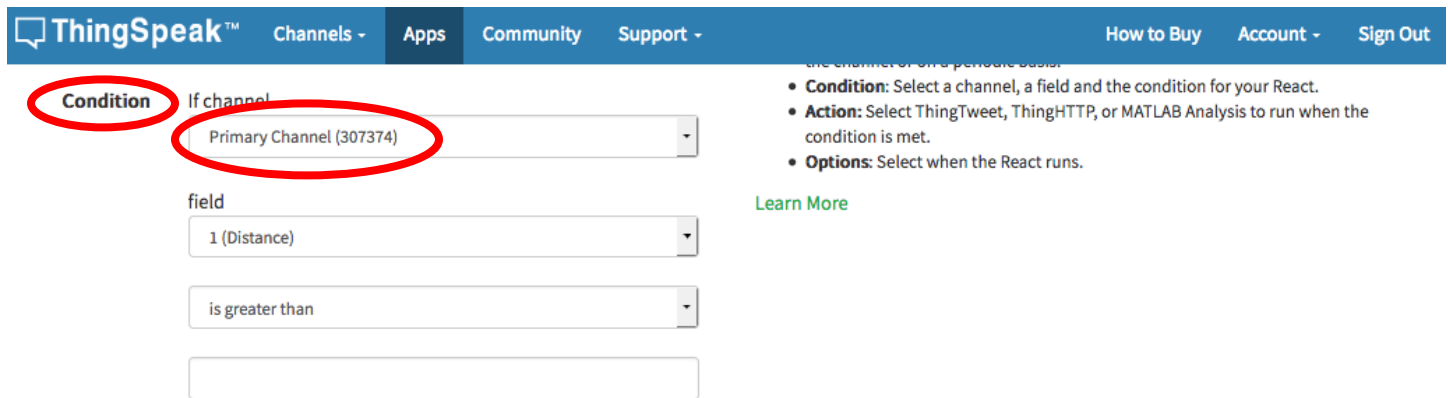
Condition: No Data Check

Help

React Settings

- **React Name:** Enter a unique name for your React.
- **Condition Type:** Select a condition type corresponding with your data. A channel can hold numeric sensor data, text, strings, status updates, or geographic location information.
- **Test Frequency:** Choose whether to test your condition every time data enters the channel or on a periodic basis.
- **Condition:** Select a channel, a field and the condition for your React.
- **Action:** Select ThingTweet, ThingHTTP, or MATLAB Analysis to run when the

Step 6: Set the condition. This will perform the action that we want the car to make. It is set up as an "if, then" logic command. We first set the "If channel" to the Primary Channel.



Condition: If channel

If channel: Primary Channel (307374)

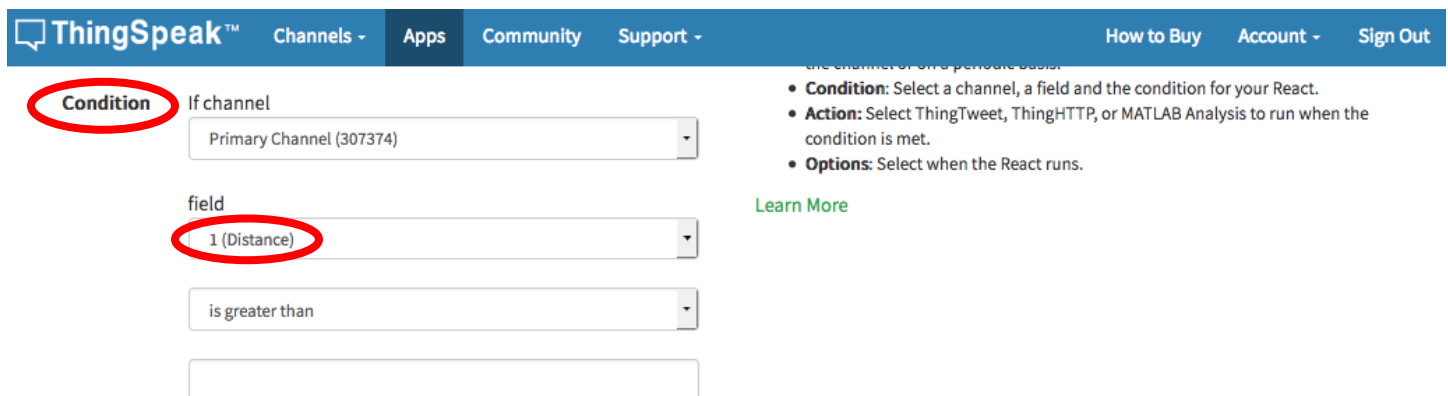
field: 1 (Distance)

is greater than

- **Condition:** Select a channel, a field and the condition for your React.
- **Action:** Select ThingTweet, ThingHTTP, or MATLAB Analysis to run when the condition is met.
- **Options:** Select when the React runs.

[Learn More](#)

Step 7: We then choose the field. Since we want to analyze the distance, we will leave this field portion as it is.



Condition: If channel

If channel: Primary Channel (307374)

field: 1 (Distance)

is greater than

- **Condition:** Select a channel, a field and the condition for your React.
- **Action:** Select ThingTweet, ThingHTTP, or MATLAB Analysis to run when the condition is met.
- **Options:** Select when the React runs.

[Learn More](#)

Step 8: We now choose the condition statement. In our case, we want our car to stop when the ultrasonic sensor senses an object less than twenty units away. We set the condition as follows:

ThingSpeak™ Channels - Apps Community Support - How to Buy Account - Sign Out

Condition Type: Numeric

Test Frequency: On Data Insertion

Condition If channel: Primary Channel (307374)

field: 1 (Distance)

is greater than
is greater than or equal to
is less than
is less than or equal to
is equal to
is not equal to

- **React Name:** Enter a unique name for your React.
- **Condition Type:** Select a condition type corresponding with your data. A channel can hold numeric sensor data, text, strings, status updates, or geographic location information.
- **Test Frequency:** Choose whether to test your condition every time data enters the channel or on a periodic basis.
- **Condition:** Select a channel, a field and the condition for your React.
- **Action:** Select ThingTweet, ThingHTTP, or MATLAB Analysis to run when the condition is met.
- **Options:** Select when the React runs.

[Learn More](#)

Step 9: Fill in the numeric condition. Because we want our car to stop when the ultrasonic sensor senses an object less than twenty units away, we enter "20" into the field.

ThingSpeak™ Channels - Apps Community Support - How to Buy Account - Sign Out

Condition Type: Numeric

Test Frequency: On Data Insertion

Condition If channel: Primary Channel (307374)

field: 1 (Distance)

is less than

20

- **React Name:** Enter a unique name for your React.
- **Condition Type:** Select a condition type corresponding with your data. A channel can hold numeric sensor data, text, strings, status updates, or geographic location information.
- **Test Frequency:** Choose whether to test your condition every time data enters the channel or on a periodic basis.
- **Condition:** Select a channel, a field and the condition for your React.
- **Action:** Select ThingTweet, ThingHTTP, or MATLAB Analysis to run when the condition is met.
- **Options:** Select when the React runs.

[Learn More](#)

Step 10: We now need to set the action using MATLAB Analysis. First click on the dropdown menu, then choose MATLAB Analysis.

is less than

20

Action ThingHTTP
ThingHTTP
MATLAB Analysis
ThingTweet
Add ThingHTTP request

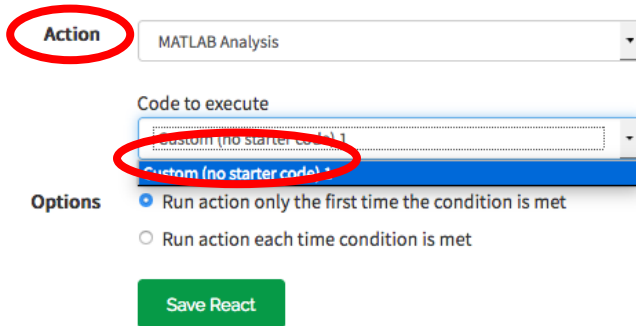
Options

Run action only the first time the condition is met

Run action each time condition is met

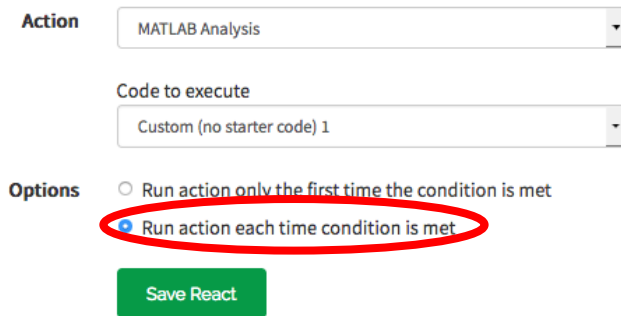
Save React

Step 11: Set the Code to execute "Custom (no starter code) 1."



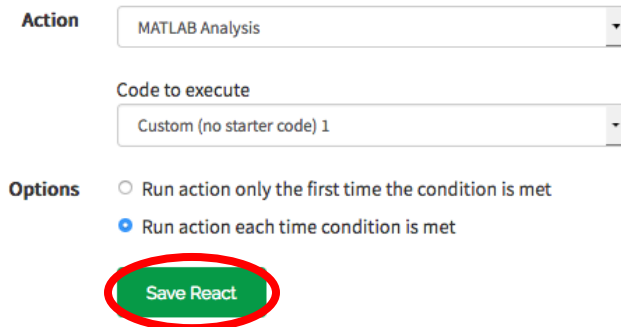
The screenshot shows a configuration interface for a MATLAB Analysis action. The "Action" dropdown is set to "MATLAB Analysis". The "Code to execute" dropdown is set to "Custom (no starter code) 1". Under "Options", the radio button for "Run action only the first time the condition is met" is selected. A green "Save React" button is at the bottom.

Step 12: Set the option to read "Run action each time condition is met."



The screenshot shows the same configuration interface as Step 11. The "Action" dropdown is "MATLAB Analysis" and the "Code to execute" dropdown is "Custom (no starter code) 1". Under "Options", the radio button for "Run action each time condition is met" is now selected. A green "Save React" button is at the bottom.

Step 13: Click "Save React."



The screenshot shows the same configuration interface as Step 12. The "Action" dropdown is "MATLAB Analysis" and the "Code to execute" dropdown is "Custom (no starter code) 1". Under "Options", the radio button for "Run action each time condition is met" is selected. The green "Save React" button is circled in red.

Section 4: Code & Sensor Setup

This tutorial will now transfer data from an ultrasonic sensor to ThingSpeak using a Raspberry Pi. In the last tutorial, we setup a Reaction using MATLAB to generate a response. Here's what it will do:

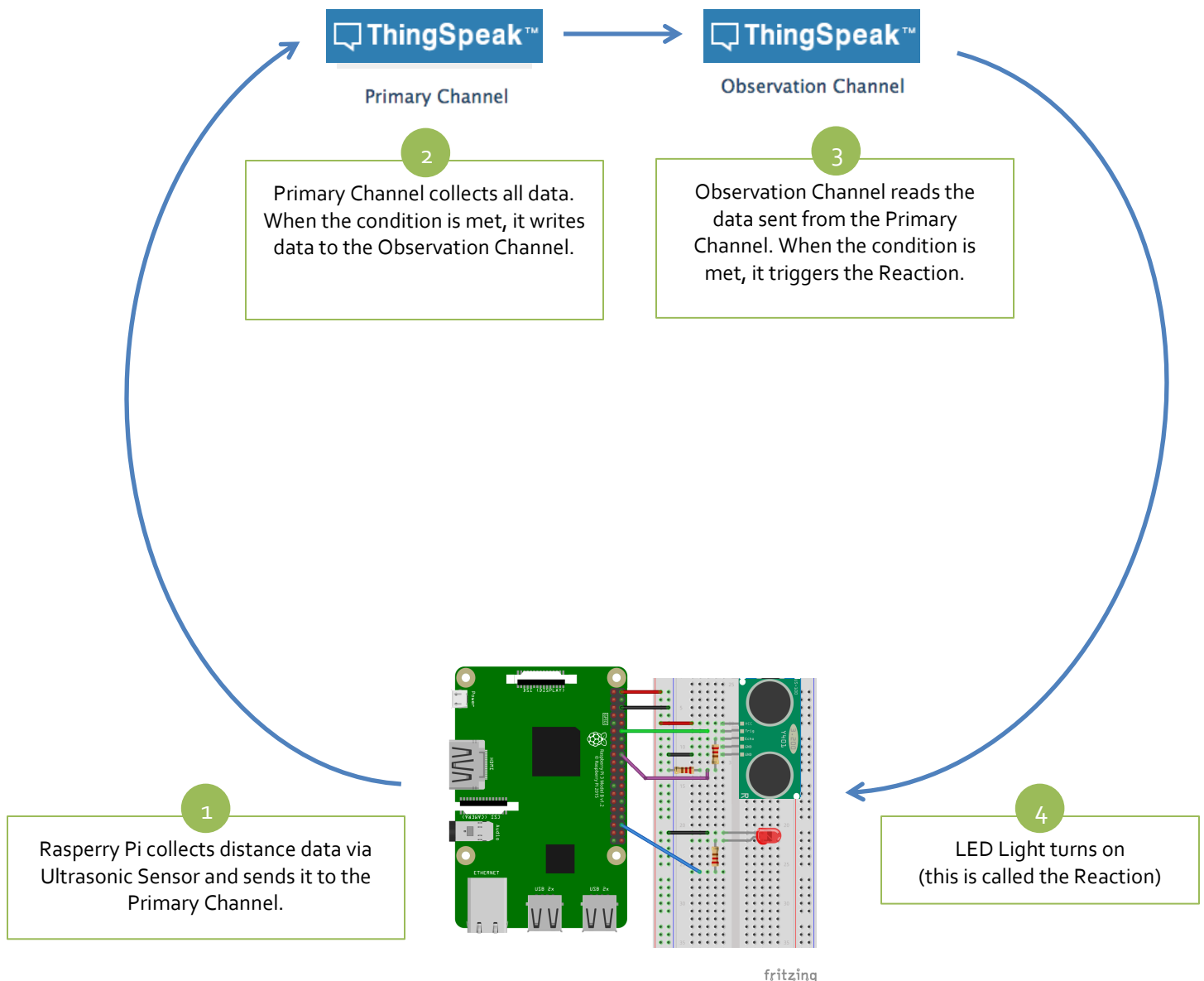
- Collect distance data using an ultrasonic sensor powered by a Raspberry Pi
- Send the distance data from the Raspberry Pi up to the Primary Channel in ThingSpeak
- Analyze the data using the Primary Channel. When an object less than 20cm away is sensed, it will send a command to the Observation Channel.
- The Observation Channel will send feedback to the Raspberry Pi which will turn on an LED light. (We call this the *reaction*. We set the condition for the reaction in Section 3 of the tutorial using the Reaction condition. Our condition is < 20 . Other conditions may be set.)

We will complete this section in three parts.

Part 1. Wiring the Pi with an LED light and an ultrasonic sensor.

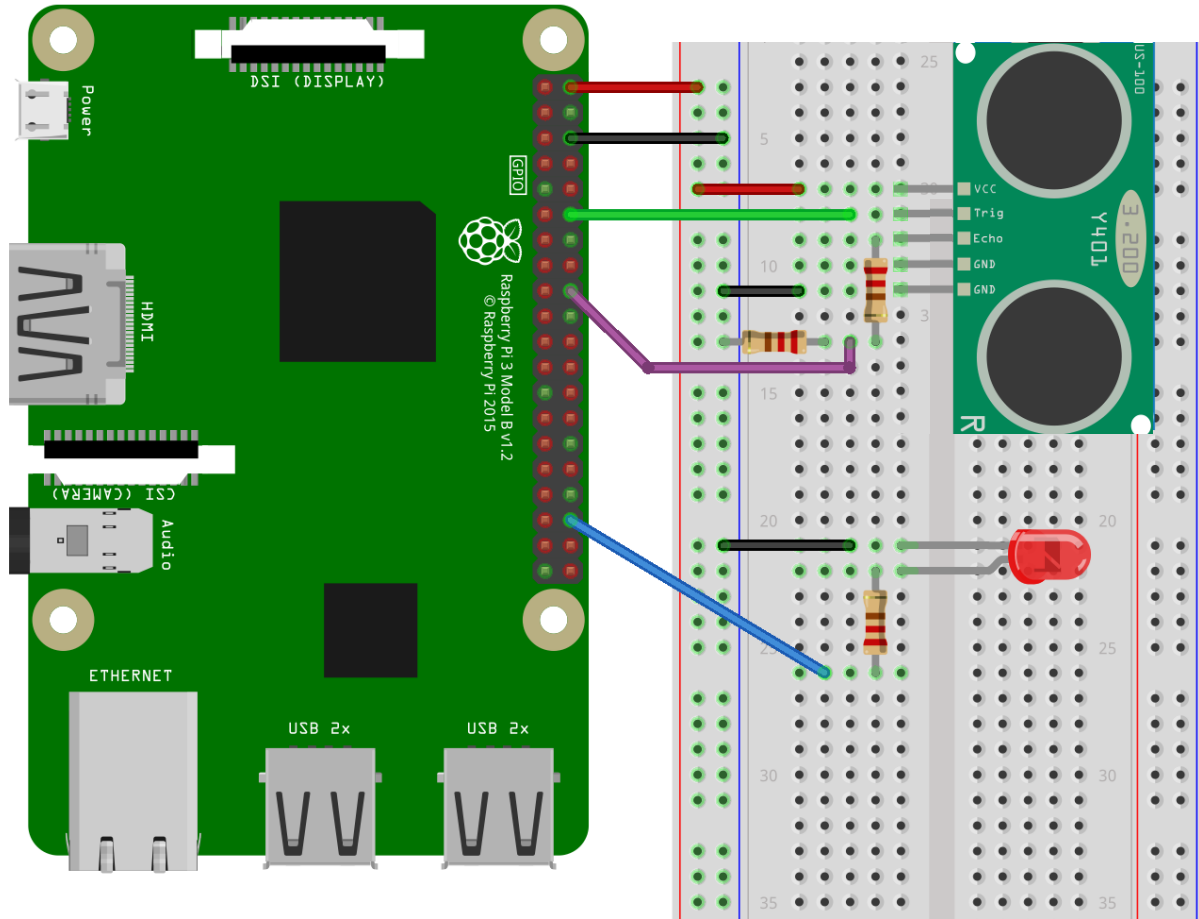
Part 2. Setting up the code.

Part 3. Watching it all work together.



Part 1: Wiring the Pi

Step 1: Hookup the ultrasonic sensor and LED light using the schematic below. (For a complete explanation on breadboards and wiring for beginners, view [this](#) tutorial.)



fritzing

Note: Depending on the type of sensor that you use, be sure that the Trigger corresponds to the Trigger in the image and that the Echo corresponds to the Echo in the image.

Part 2: Setting Up the Code

The libraries that you need to run this code should already be installed on your Raspberry Pi 3.

To install `http.client` type the following command into the terminal window:

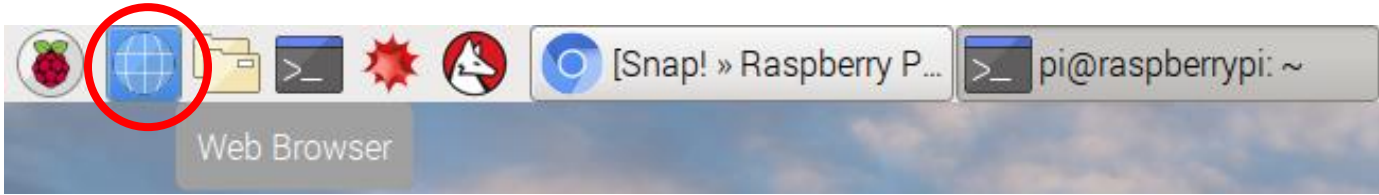
```
pip3 install http.client
```

We now just need to install the file to run our ultrasonic sensor.

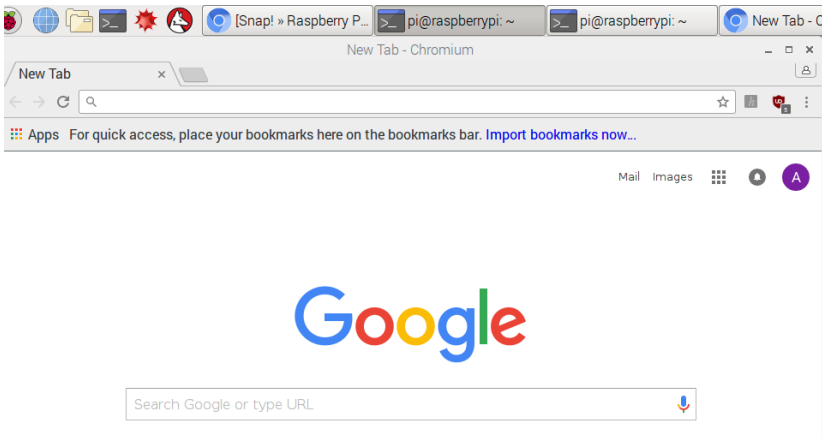
```
tutorial_test_sensor.py
```

To download a file using a Raspberry Pi, follow the steps below.

Step 1: Open a web browser on your Raspberry Pi.



Step 2: It should appear like the screen below.



Step 3: Navigate to the document (either via email or link). Then download the file. It should appear at the bottom of your Web Browser. Click on the file.

Step 4: A new window should open like the one shown below.

```
File Edit Format Run Options Windows Help
#!/usr/bin/env python
__author__ = 'Anam'
# This program logs a Raspberry Pi's Ultrasonic Sensor to a Thingspeak Channel
# Enter the proper API Keys and Channel ID noted below.

'''-----LIBRARY Setup-----'''
import http.client
import urllib.parse
import urllib.request
import json
import time
import RPi.GPIO as GPIO

#GPIO Mode (BOARD / BCM)
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

'''-----VARIABLE Setup-----'''
key = 'FVLP4EYMV9ICFMFQ' #Enter your Primary Channel WRITE API Key
READ_API_KEY='XZ3YG69DFZQ5QN70' #Enter your Observation Channel READ API Key
CHANNEL_ID='307290' #Enter your Observation Channel ID Number

entry_ID=0 #Sequence number of the receiving Observation Channel, initialize as 0
sleep = 40 #How many seconds to sleep between posts to the channel

'''-----GPIO SETUP-----'''
#set GPIO Pins
GPIO_TRIGGER = 18
GPIO_ECHO = 24
GPIO_LED = 16

#set GPIO direction (IN / OUT)
GPIO.setup(GPIO_TRIGGER, GPIO.OUT)
GPIO.setup(GPIO_ECHO, GPIO.IN)
GPIO.setup(GPIO_LED, GPIO.OUT)

def distance():
    # set Trigger to HIGH
    GPIO.output(GPIO_TRIGGER, True)

    # set Trigger after 0.01ms to LOW
    time.sleep(0.00001)
    GPIO.output(GPIO_TRIGGER, False)

    StartTime = time.time()
    StopTime = time.time()

    # save StartTime
    while GPIO.input(GPIO_ECHO) == 0:
        StartTime = time.time()

    # save time of arrival
    while GPIO.input(GPIO_ECHO) == 1:
        StopTime = time.time()

    # time difference between start and arrival
    TimeElapsed = StopTime - StartTime
    # multiply with the sonic speed (34300 cm/s)
    # and divide by 2, because there and back
    distance = (TimeElapsed * 34300) / 2
```

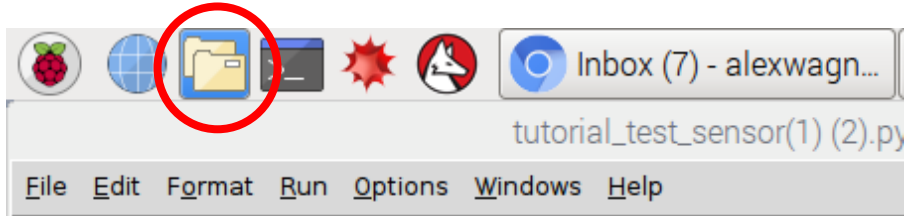
Note: This code must be run in Python 3. To be sure you are in Python 3, check the file name at the top of the window. It should begin with "3" like this:

```
tutorial_test_sensor(1) (1).py - /home/pi/Downloads/tutorial_test_sensor(1) (1).p(3.4.2)
```

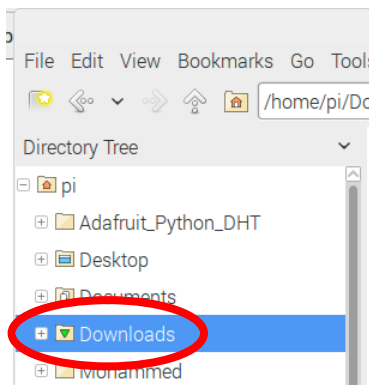
If your file name looks like this, you may proceed to Step 5.

If your file begins with a "2" you must open the file following these instructions:

Mini Step 1: Click the Folder Icon.



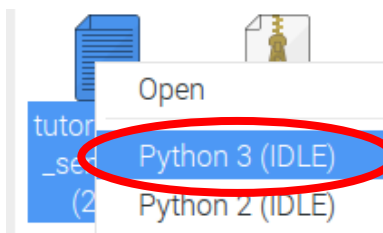
Mini Step 2: Click Downloads



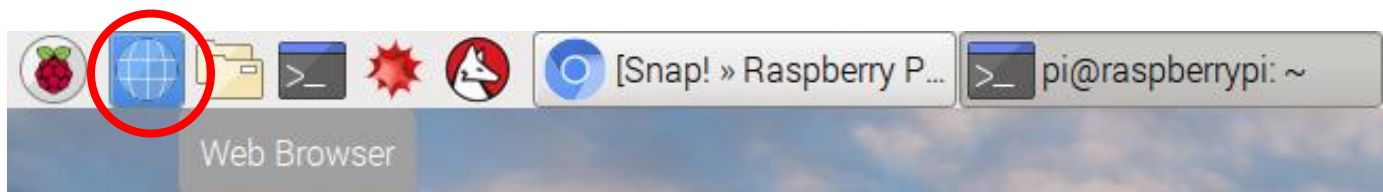
Mini Step 3: Locate your file

and right click on it.

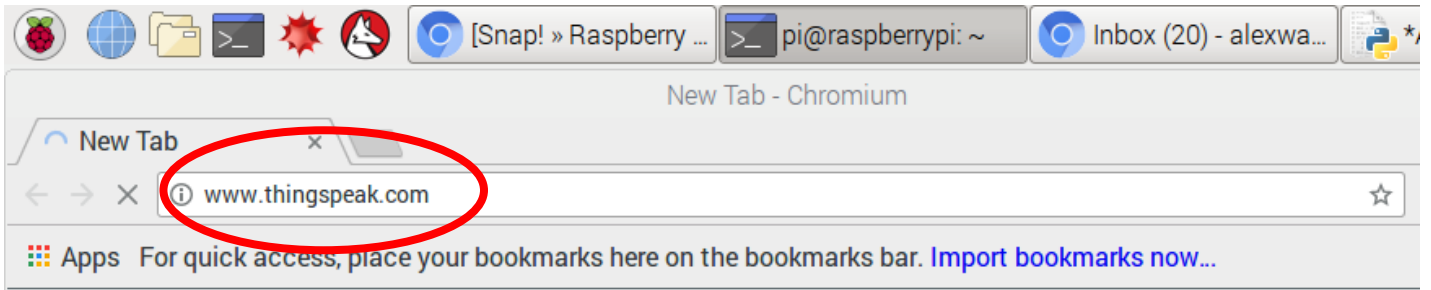
Mini Step 4: Click "Python 3 (IDLE)"



Step 5: We now need to enter our custom Channel ID and API Keys. We will need to log back onto ThingSpeak to access this information. First, open a new Chromium browser.



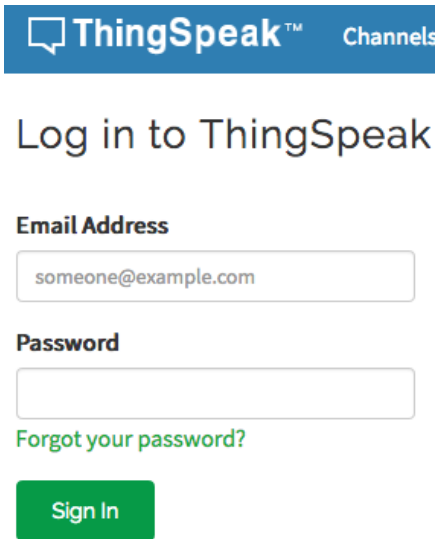
Step 6: Navigate to www.thingspeak.com



Step 7: Once there, click Log In.



Step 8: Enter your account information.



Step 9: To easily view all of the API Keys and Channel IDs, we will navigate to a MATLAB Analysis screen, which has all information listed at the right hand side of the page. First, click Apps.



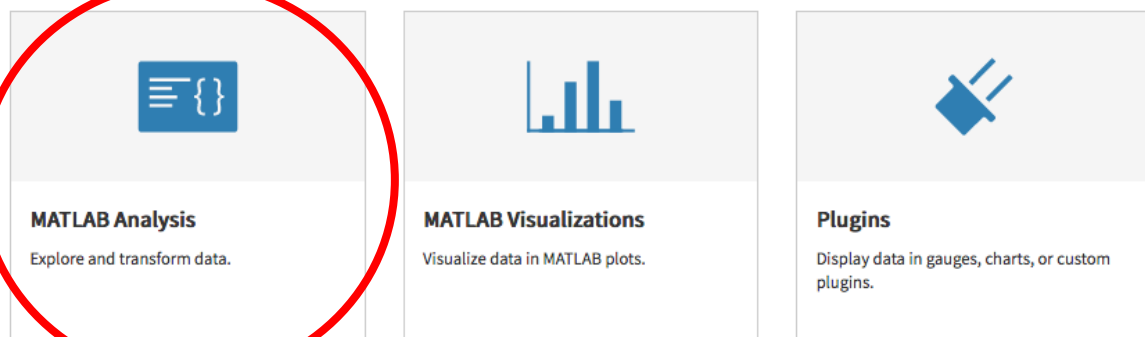
Step 10: Click MATLAB Analysis.



Apps

ThingSpeak channels store data. Upload data from the web or send data from devices to a ThingSpeak channel. Use these apps to transform and visualize data or trigger an action. See [Tutorial: ThingSpeak and MATLAB](#) to create a channel. [Learn more](#) about MATLAB® inside ThingSpeak.

Analytics



Step 11: Click "New."

Apps / MATLAB Analysis

Click **New** and choose a template to get started. Templates contain sample MATLAB® code for analyzing data.

New

Help

MATLAB Analysis

- Explore data collected in a channel or scraped from a website
- Find and remove bad data
- Convert data to different units
- Calculate new data

Step 12: Click "Create."

Templates:

- Custom (no starter code)
- Get data from a private channel
- Get data from a public channel
- Get data from a webpage

Examples: Sample code to analyze and transform data

- Calculate average humidity
- Calculate dew point
- Convert Celsius to Fahrenheit
- Eliminate data outliers
- Convert Fahrenheit to Celsius
- Calculate hourly max temperature
- Replace missing values in data
- Analyze text for the most common color
- Scrape web data for ships at the Boston port
- Scrape web temperature data

Create

Templates

MATLAB Analysis templates provide sample MATLAB code for analyzing data and writing it to a ThingSpeak channel. If you are new to MATLAB, you can learn interactively at [MATLAB Academy](#).

Examples

To see MATLAB Analysis in action, select the example and click **Create**.

These examples read data from public ThingSpeak channels:

- **Calculate average humidity**, and write the data to a new channel.
- **Calculate dew point** from temperature and humidity data, and write data to a new channel.
- **Convert Celsius to Fahrenheit**, and write data to a new channel.
- **Eliminate data outliers** from temperature data, and write data to a new channel.
- **Convert Fahrenheit to Celsius**, and write data to a new channel.
- **Calculate hourly max temperature**, and write data with the timestamps to a new channel.
- **Replace missing values in data** of a weather channel, and clean the data using a list wise deletion algorithm. Then display the missing values, or write data to a new channel.
- **Analyze text for the most common color** requested on the public Cheerlights channel, and write data to a new channel.

These examples scrape data from websites:

- **Scrape web data for ships at the Boston port** from the MarineTraffic® website, count the number of ships in Boston port, and write data to a new channel.
- **Scrape web temperature data** from the National Weather Service website, and write data to a new channel.

New to MATLAB?

Step 13: We can now see all of the API Keys and Channel IDs listed at the right. We will enter this information into the `tutorial_test_sensor.py` file.

Name

Custom (no starter code) 1

MATLAB Code

```
1 % Enter your MATLAB Code below
2 ChannelID=307290;
3 Field=1;
4 Write='ZADHAQ44UIZ6JC98';
5 thingSpeakWrite(ChannelID,111,'fields',Field,'WriteKey',Write);
```

My Channels

Documentation

New Channel

Channel Info

Name: **Primary Channel**
Channel ID: 306964
Access: Private
Read API Key: **356H9W7ZNN2CWUTY**
Write API Key: **FYLP4EYMV9ICFMFQ**
Fields:
1: Distance

Name: **Observation Channel**
Channel ID: 307290

Step 14: Copy the correct API Keys and Channel ID into the `ultrasonic-ts.py` file. Note: Be sure to keep the apostrophes in place. Your key should appear as: 'API KEY'

The screenshot shows the Thingspeak web interface on the left and a terminal window on the right. The Thingspeak interface displays two channels:

- Primary Channel:** Channel ID: 306964, Access: Private, Read API Key: 3S6H9W7ZNN2CWJTY, Write API Key: FYLP4EYMV9ICFMFQ.
- Observation Channel:** Channel ID: 307290, Access: Private, Read API Key: XZ3YGG9DFZQSQN70, Write API Key: ZADHAQ44UIZ6JC98.

The terminal window shows the Python script `Alex_tutorial.py` with the following code:

```
#!/usr/bin/env python
__author__ = 'Md Anam'
# This program logs a Raspberry Pi's Ultrasonic Sensor to a Thingspeak
# Enter the proper API Keys and Channel ID noted below.

'''-----LIBRARY Setup-----'''
import http.client
import urllib.parse
import urllib.request
import json
import time

'''-----VARIABLE Setup-----'''
key = 'FLYP4EYMV9ICFMFQ'
READ_API_KEY = 'XZ3YGG9DFZQSQN70'
CHANNEL_ID = '307290'

entry_ID=0
sleep = 40

#Enter your Primary Channel WRI
#Enter your Observation Channel
#Enter your Observation Channel
#Sequence number of the receivi
#How many seconds to sleep betw
```

Red, orange, and blue arrows indicate the mapping of values from the Thingspeak interface to the script variables.

Step 15: Enter Channel ID and Read API Key into the code. Note: The apostrophe (') after `READ_API_KEY` should be left there. An example of the full web address is shown below. The Channel ID and API Key are highlighted in yellow.

'http://api.thingspeak.com/channels/307290/feeds/last.json?api_key=XZ3YGG9DFZQSQN70'

```

'''-----DATA sending to ThingSpeak-----'''
def sending(distance):
    while True:
        INPUT=distance
        params = urllib.parse.urlencode({'field1': INPUT, 'key':key})
        headers = {"Content-type": "application/x-www-form-urlencoded","Accept": "text/plain"}
        conn = http.client.HTTPConnection("api.thingspeak.com:80")
        try:
            conn.request("POST", "/update", params, headers)
            response = conn.getresponse()
            #print (temp)
            print (response.status, response.reason)
            data = response.read()
            #conn.close()
        except:
            print ("connection failed")
        break

'''-----DATA receiving from ThingSpeak-----'''
def receiving(ID):
    conn=urllib.request.urlopen('http://api.thingspeak.com/channels/CHANNEL_ID/feeds/last.json?api_key=READ_API_KEY')
    response = conn.read()
    print("http status code=%s" % (conn.getcode()))
    check=len(response)

```

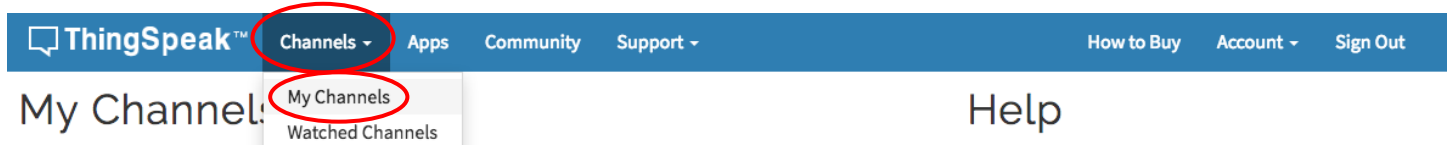
The screenshot shows the Thingspeak user interface. At the top, there are navigation links: "How to Buy", "Account", and "Sign Out". Below these are "My Channels" and "Documentation" buttons. A prominent green "New Channel" button is visible. The main content area displays "Channel Info" for two channels:

- Primary Channel:** Channel ID: 306964, Read API Key: 3S6H9W7ZNN2CWJTY, Write API Key: FYLP4EYMV9ICFMFQ. The Read API Key is circled in red.
- Observation Channel:** Channel ID: 307290, Read API Key: XZ3YGG9DFZQSQN70, Write API Key: ZADHAQ44UIZ6JC98. Both the Channel ID and Read API Key are circled in red.

Two arrows originate from the screenshot: a red arrow points from the Read API Key of the Primary Channel to the `READ_API_KEY` variable in the code, and a blue arrow points from the Read API Key of the Observation Channel to the `READ_API_KEY` variable in the code.

Part 3: Watching it All Work Together

Step 1: Open ThingSpeak and click on "Channels" > "My Channels."



Step 2: Click "Private" under Primary Channel.

A screenshot of the 'My Channels' page. At the top is a 'New Channel' button. Below it is a table with two channels: 'Primary Channel' and 'Observation Channel'. The 'Private' button for the Primary Channel is circled in red. The table has columns for 'Name', 'Created', and 'Updated At'.

Name	Created	Updated At
Primary Channel Private Public Settings Sharing API Keys Data Import / Export	2017-07-25	2017-07-25 18:30
Observation Channel Private Public Settings Sharing API Keys Data Import / Export	2017-07-25	2017-07-25 19:08

Help

Collect data in a ThingSpeak channel from a device, from another channel, or from the web.

Click **New Channel** to create a new ThingSpeak channel.

Click on the column headers of the table to sort by the entries in that column.

Learn to [create channels](#), explore and transform data.

Learn more about [ThingSpeak Channels](#).

Examples

Step 3: A blank graph should appear like this:

A screenshot of the ThingSpeak channel page. At the top is a navigation bar with 'Channels', 'Apps', 'Community', 'Support', 'How to Buy', 'Account', and 'Sign Out'. Below the navigation bar are buttons for 'Add Visualizations', 'Data Export', 'MATLAB Analysis', and 'MATLAB Visualization'. The main content area is titled 'Channel Stats' and shows 'Created: 7 days ago', 'Updated: 7 days ago', and 'Entries: 0'. Below the stats is a 'Field 1 Chart' which is currently blank. The chart has a y-axis labeled 'Distance' and an x-axis labeled 'Date'. The ThingSpeak logo is visible in the bottom right corner of the chart area.

Step 4: Open a new window. Navigate to your ThingSpeak account. This time, click “Private” under “Observation Channel.”

The screenshot shows the ThingSpeak 'My Channels' page. At the top, there is a navigation bar with 'Channels', 'Apps', 'Community', and 'Support' menus, and 'How to Buy', 'Account', and 'Sign Out' links. Below the navigation bar, the page is titled 'My Channels' and features a 'New Channel' button. A table lists two channels:

Name	Created	Updated At
Primary Channel Private Public Settings Sharing API Keys Data Import / Export	2017-07-25	2017-07-25 18:30
Observation Channel Private Public Settings Sharing API Keys Data Import / Export	2017-07-25	2017-07-25 19:08

The 'Private' button for the 'Observation Channel' is circled in red. To the right of the table, there is a 'Help' section with instructions on how to create channels and sort data, and an 'Examples' section.

Step 5: Put these two viewing windows side-by-side on your screen, as shown below.

The screenshot shows two browser windows side-by-side. The left window displays the 'Primary Channel' view, and the right window displays the 'Observation Channel' view. Both windows show a 'Field 1 Chart' with a y-axis labeled 'Distance' and an x-axis labeled 'Date'. The 'Observation Channel' view also shows a y-axis labeled 'Field Label 1'. Both channels are listed as 'Created: 7 days ago', 'Updated: 7 days ago', and 'Entries: 0'.

Step 6: Make sure there are no objects within 20cm of your ultrasonic sensor. (You may find it helpful to point it to the ceiling.)

Step 7: On your Raspberry Pi, make sure you are in the `tutorial_test_sensor.py` window (shown below).

```
File Edit Format Run Options Windows Help
#!/usr/bin/env python
__author__ = 'Anam'
# This program logs a Raspberry Pi's Ultrasonic Sensor to a Thingspeak Channel
# Enter the proper API Keys and Channel ID noted below.

'''-----LIBRARY Setup-----'''
import http.client
import urllib.parse
import urllib.request
import json
import time
import RPi.GPIO as GPIO

#GPIO Mode (BOARD / BCM)
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

'''-----VARIABLE Setup-----'''
key = 'FYLP4EYV9ICFMFQ'          #Enter your Primary Channel WRITE API Key
READ_API_KEY='XZ3YG69DFZQ5QN70' #Enter your Observation Channel READ API Key
CHANNEL_ID='307290'              #Enter your Observation Channel ID Number

entry_ID=0                       #Sequence number of the receiving Observation Channel, initialize as 0
sleep = 40                        #How many seconds to sleep between posts to the channel

'''-----GPIO SETUP-----'''
#set GPIO Pins
GPIO_TRIGGER = 18
GPIO_ECHO = 24
GPIO_LED = 16

#set GPIO direction (IN / OUT)
GPIO.setup(GPIO_TRIGGER, GPIO.OUT)
GPIO.setup(GPIO_ECHO, GPIO.IN)
GPIO.setup(GPIO_LED, GPIO.OUT)

def distance():
    # set Trigger to HIGH
    GPIO.output(GPIO_TRIGGER, True)

    # set Trigger after 0.01ms to LOW
    time.sleep(0.00001)
    GPIO.output(GPIO_TRIGGER, False)

    StartTime = time.time()
    StopTime = time.time()

    # save StartTime
    while GPIO.input(GPIO_ECHO) == 0:
        StartTime = time.time()

    # save time of arrival
    while GPIO.input(GPIO_ECHO) == 1:
        StopTime = time.time()

    # time difference between start and arrival
    TimeElapsed = StopTime - StartTime
    # multiply with the sonic speed (34300 cm/s)
    # and divide by 2, because there and back
    distance = (TimeElapsed * 34300) / 2
```

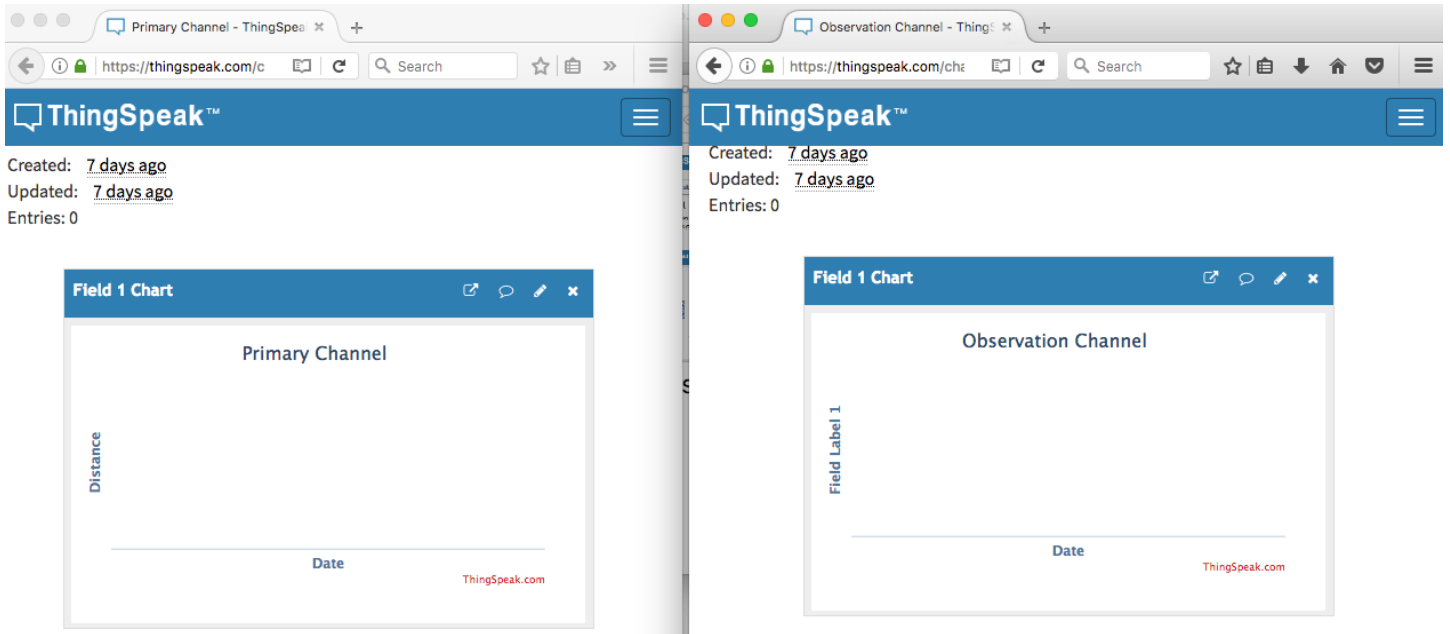
Step 8: Press F5 to run your program. A window should appear like this:

```
*Python 3.4.2 Shell*
File Edit Shell Debug Options Windows Help
time.sleep(10)
KeyboardInterrupt
>>> ===== RESTART =====
>>>
Distance 77
200 OK
pre_entry_ID= 0
http status code=200
65
```

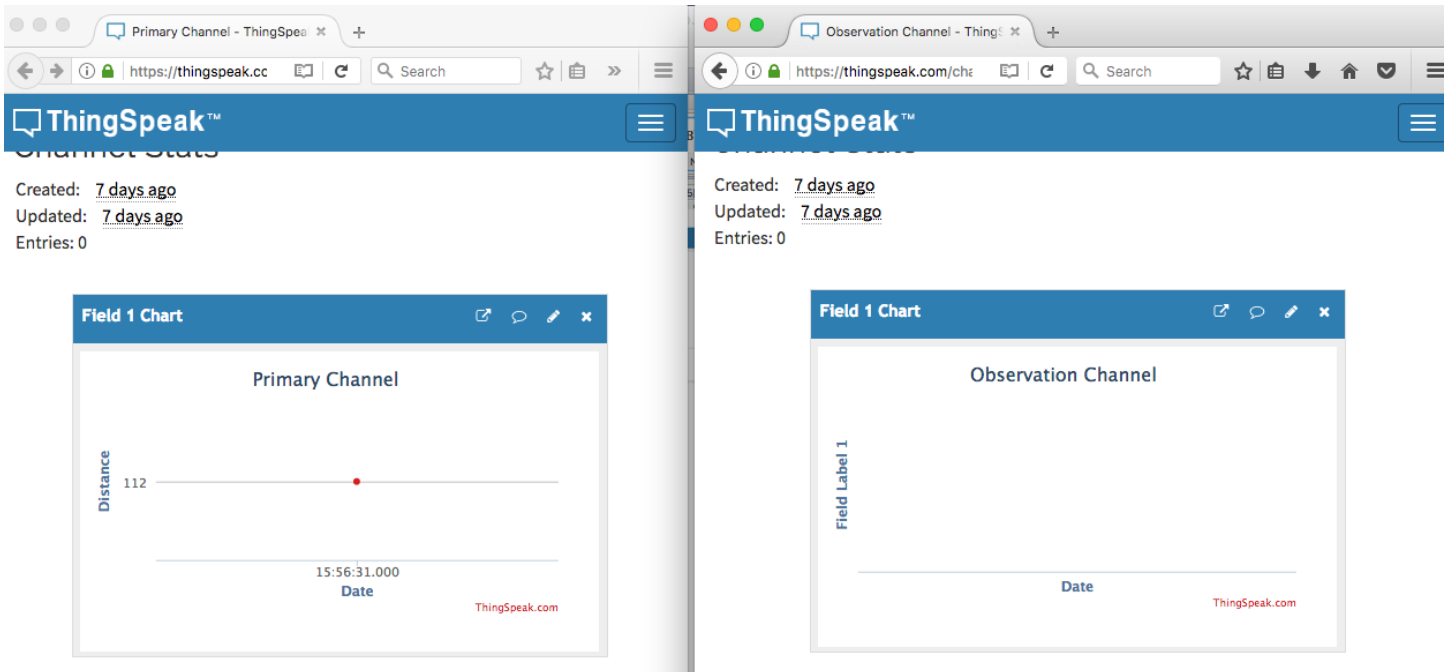
The line that reads `200 OK` indicates that the data that was sent through a proper connection. Allow the code to keep running.

Step 9: Wait (a delay is built into the code).

Step 10: While you are waiting, observe what happens on your ThingSpeak channels. They should go from this.....

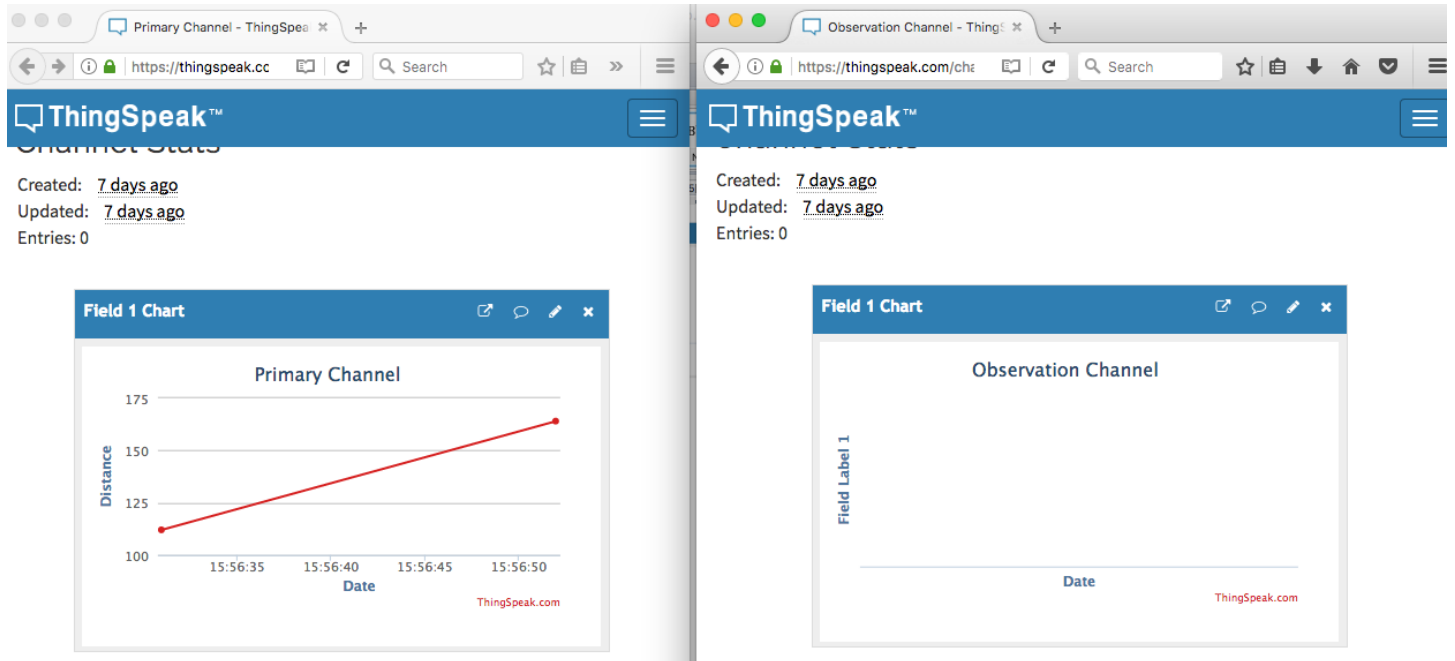


...to this



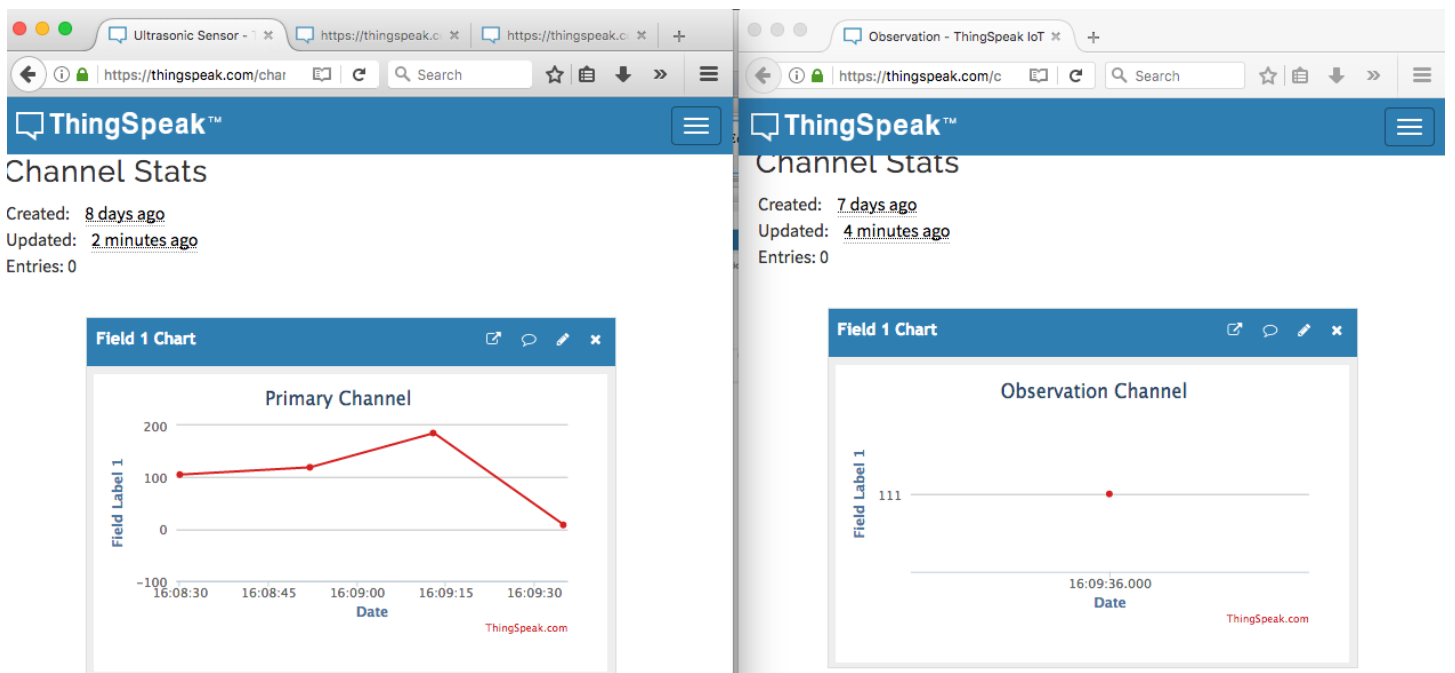
***Note: The specific values on your graph will be different than the ones shown. For example, our first data point was 112. Yours may be 100 or 63 or another value.*

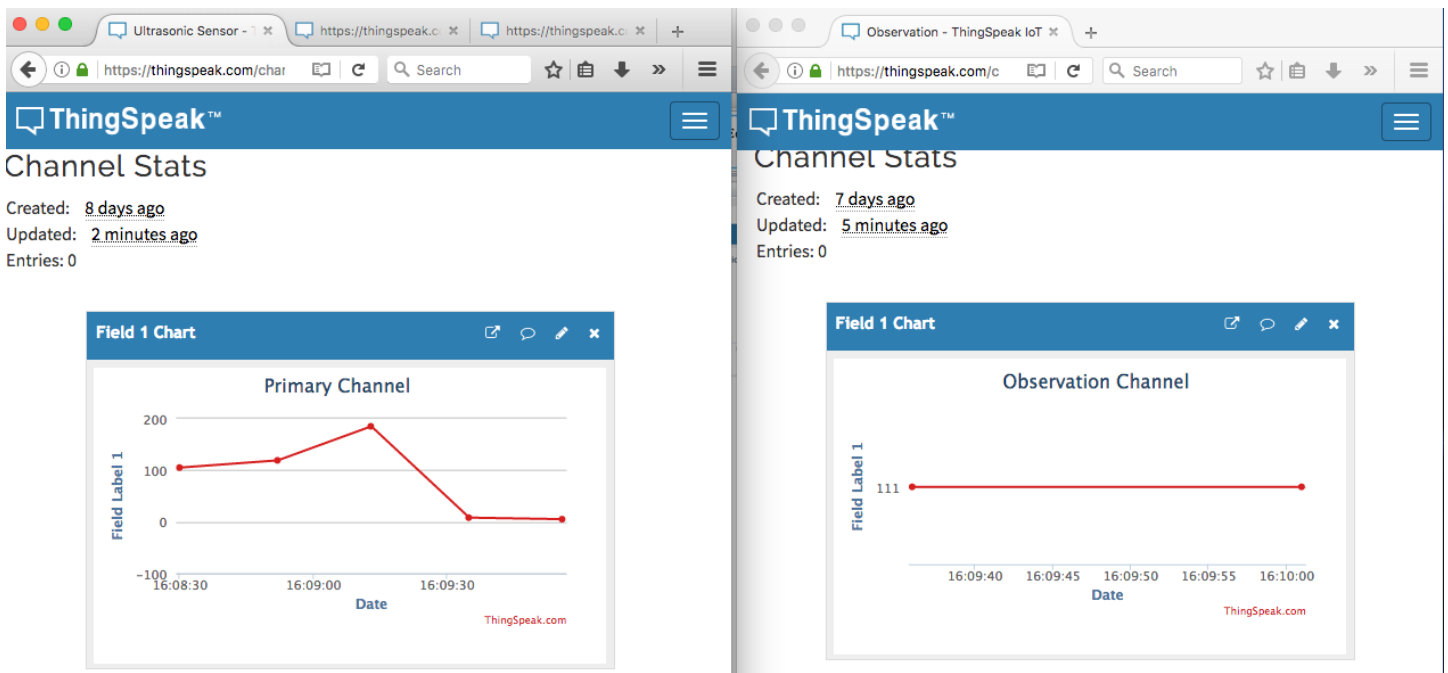
Step 11: Continue watching. The program will continue running. Another data point will be sent to ThingSpeak and look like this.



Step 12: Place an object near your ultrasonic sensor (within 20 cm). Recall that in Section 3 of the tutorial we set up the Reaction condition to have a reaction when a numeric value < 20 is received (this was in Section 3 > Part 2 > Steps 5-9).

Because we set the condition to respond this way, our Observation Channel will show data points *only* when the condition is met. In this step, we have now placed an object in front of the ultrasonic sensor, so the condition will be met. Watch your Observation Channel. It should show a data point:





Step 13: Data points will keep sending to ThingSpeak as long as your program is running. To stop your code, click somewhere in the run window and press Ctrl + C.

Congratulations! We have successfully configured our Raspberry Pi to communicate with ThingSpeak.

Python Code (Text Version)

```
#!/usr/bin/env python
__author__ = 'Anam'
# This program logs a Raspberry Pi's Ultrasonic Sensor to a Thingspeak Channel
# Enter the proper API Keys and Channel ID noted below.

'''-----LIBRARY Setup-----'''
import http.client
import urllib.parse
import urllib.request
import json
import time
import RPi.GPIO as GPIO

#GPIO Mode (BOARD / BCM)
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

'''-----VARIABLE Setup-----'''
key = 'FYLP4EYMV9ICFMFQ'          #Enter your Primary Channel WRITE API Key
READ_API_KEY='XZ3YGG9DFZSQSN70'  #Enter your Observation Channel READ API Key
CHANNEL_ID='307290'              #Enter your Observation Channel ID Number

entry_ID=0                       #Sequence number of the receiving Observation Channel,
initialize as 0
sleep = 40                        #How many seconds to sleep between posts to the channel

'''-----GPIO SETUP-----'''
#set GPIO Pins
GPIO_TRIGGER = 18
GPIO_ECHO = 24
GPIO_LED = 16

#set GPIO direction (IN / OUT)
GPIO.setup(GPIO_TRIGGER, GPIO.OUT)
GPIO.setup(GPIO_ECHO, GPIO.IN)
GPIO.setup(GPIO_LED,GPIO.OUT)

def distance():
    # set Trigger to HIGH
    GPIO.output(GPIO_TRIGGER, True)

    # set Trigger after 0.01ms to LOW
    time.sleep(0.00001)
    GPIO.output(GPIO_TRIGGER, False)

    StartTime = time.time()
    StopTime = time.time()

    # save StartTime
    while GPIO.input(GPIO_ECHO) == 0:
        StartTime = time.time()

    # save time of arrival
    while GPIO.input(GPIO_ECHO) == 1:
        StopTime = time.time()

    # time difference between start and arrival
    TimeElapsed = StopTime - StartTime
    # multiply with the sonic speed (34300 cm/s)
    # and divide by 2, because there and back
    distance = (TimeElapsed * 34300) / 2
    Dis=round(distance)
    print("Distance",Dis)

    return Dis
```

```

'''-----DATA sending to ThingSpeak-----'''
def sending(distance):
    while True:
        INPUT=distance
        params = urllib.parse.urlencode({'field1': INPUT, 'key':key})
        headers = {"Content-type": "application/x-www-form-urlencoded","Accept":
"text/plain"}
        conn = http.client.HTTPConnection("api.thingspeak.com:80")
        try:
            conn.request("POST", "/update", params, headers)
            response = conn.getresponse()
            #print (temp)
            print (response.status, response.reason)
            data = response.read()
            #conn.close()
        except:
            print ("connection failed")
        break

'''-----DATA receiving from ThingSpeak-----'''
def receiving(ID):
    conn=urllib.request.urlopen('http://api.thingspeak.com/channels/307290/feeds/last.json?ap
i_key=XZ3YGG9DFZSQN70')
    response = conn.read()
    print("http status code=%s" % (conn.getcode()))
    check=len(response)
    print(len(response))
    pre_entry=ID
    print('pre_entry=',pre_entry)
    data=json.loads(response.decode("utf-8"))
    #entry_ID = data['entry_id']

    if check>4:
        data=json.loads(response.decode("utf-8"))
        entry_ID = data['entry_id']
    else:
        entry_ID=ID

    conn.close()
    return entry_ID

while True:
    d=distance()
    time.sleep(.1)

    sending(d)
    print('pre_entry_ID=',entry_ID)
    last=entry_ID
    entry_ID=receiving(entry_ID)
    print('post_entry_ID=',entry_ID)
    if last!=entry_ID:
        #GPIO.setwarnings(False)
        GPIO.output(GPIO_LED,GPIO.HIGH)
        time.sleep(.1)
        last=entry_ID
    else:
        GPIO.output(GPIO_LED,GPIO.LOW)

    time.sleep(10)
GPIO.cleanup()

```