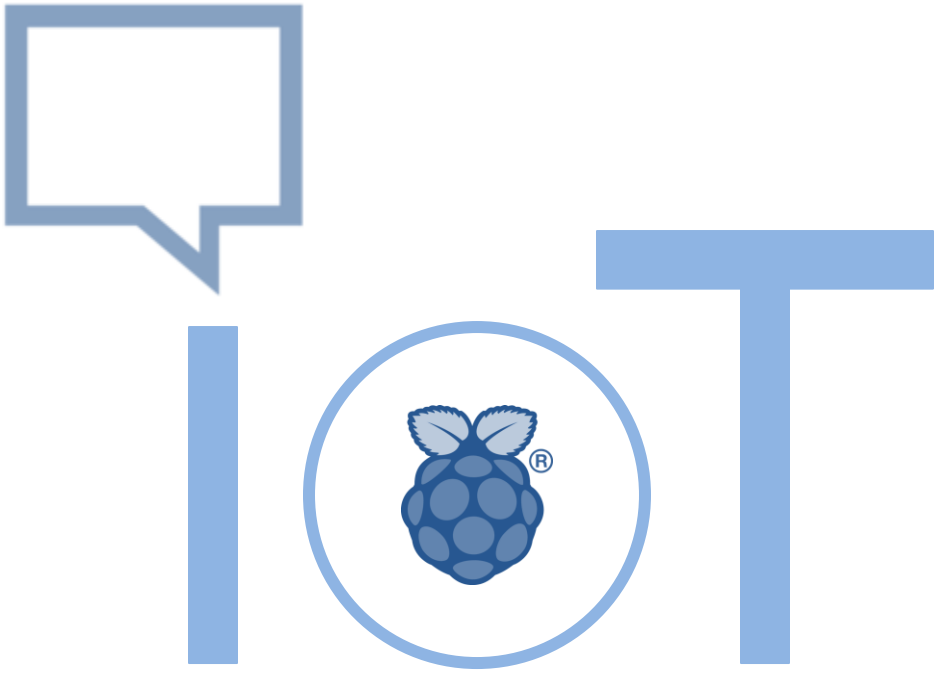


Connecting to ThingSpeak

A Step-By-Step Tutorial

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

Central Michigan University

Tutorial Overview

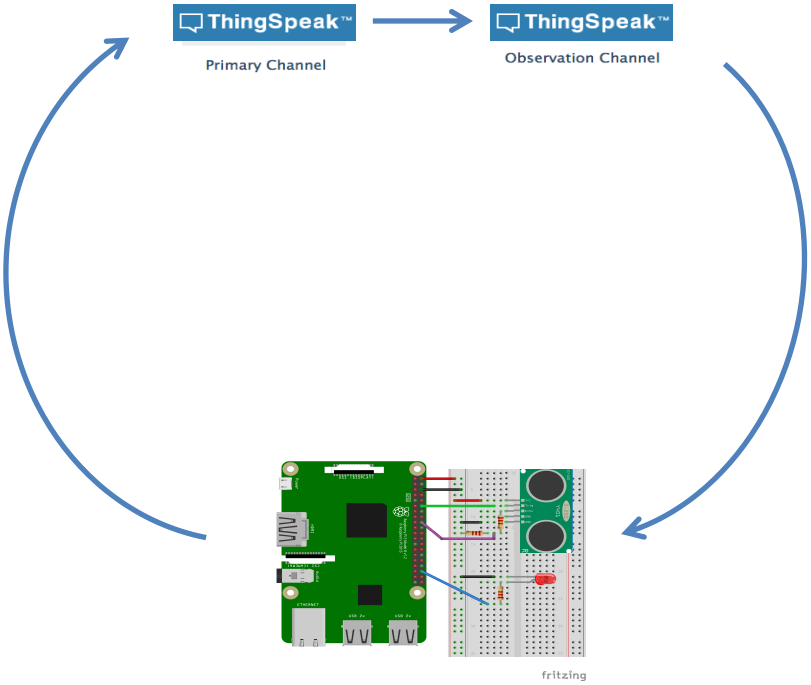
This is a complete guide to sending sensor data to using ThingSpeak, an Internet of Things (IoT) platform, to create a reaction based on the data.

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

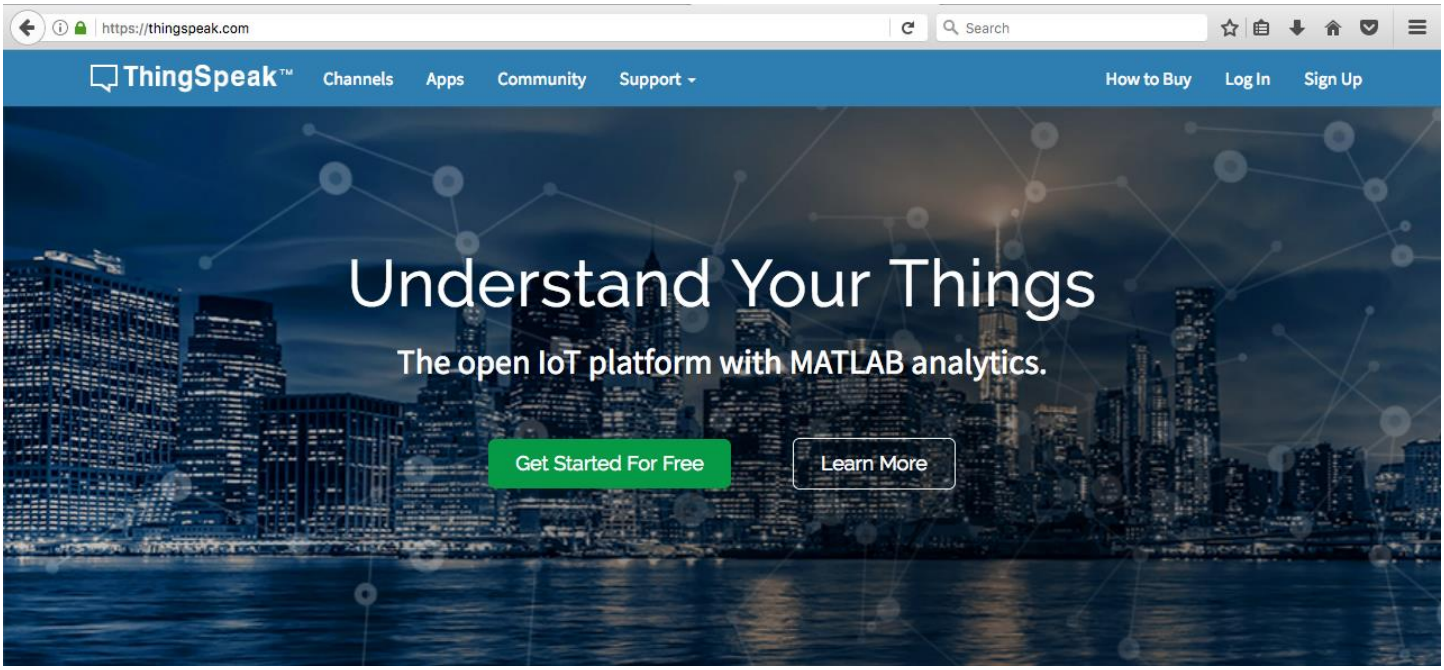
This tutorial is broken into four sections with various parts. The table of contents is listed below.

The Finished Product

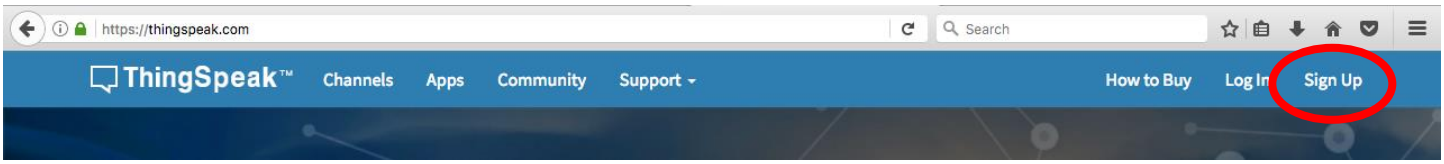


Section 1: Signing Up for ThingSpeak

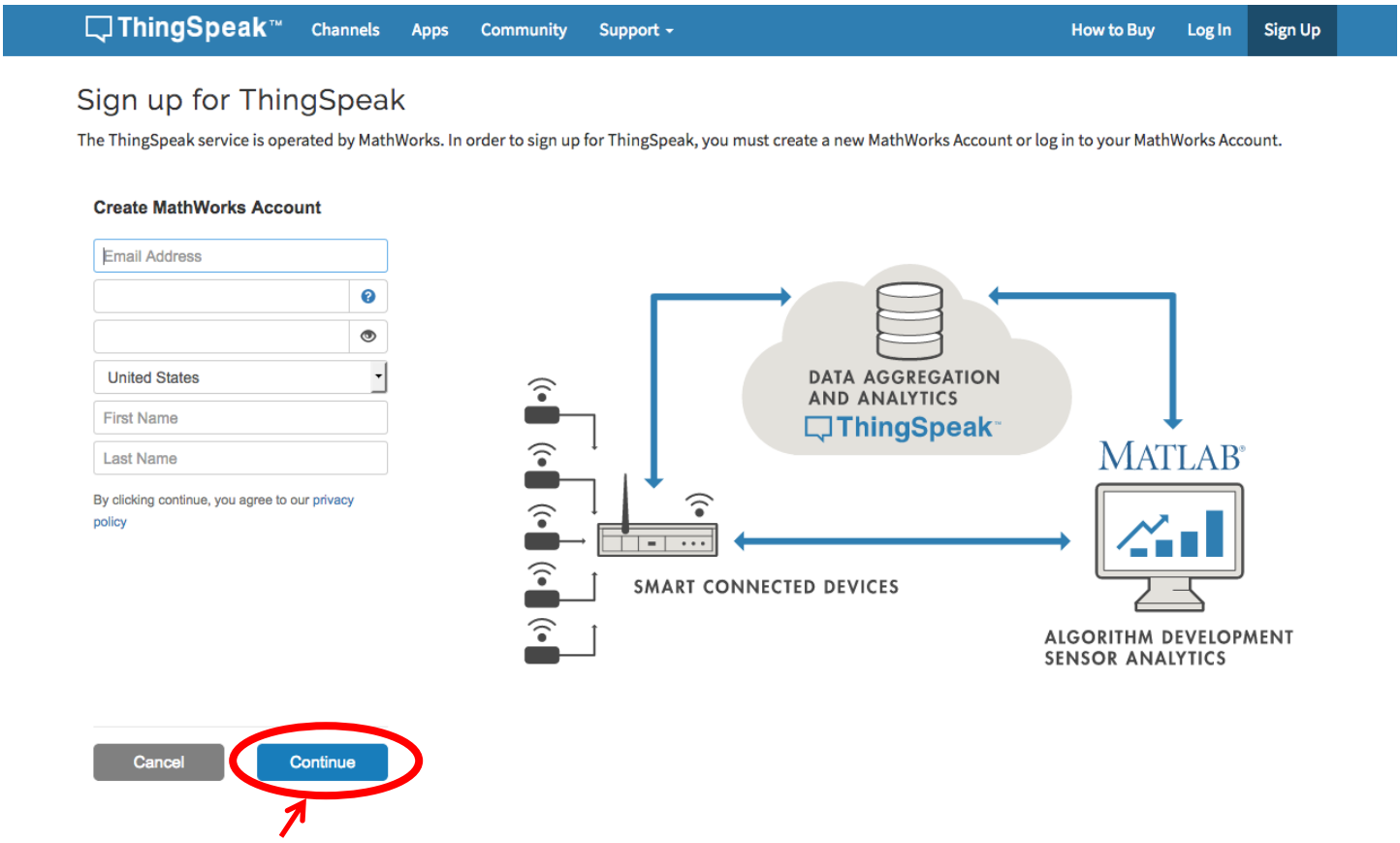
Step 1: Go to www.thingspeak.com. Your browser should look like this:



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

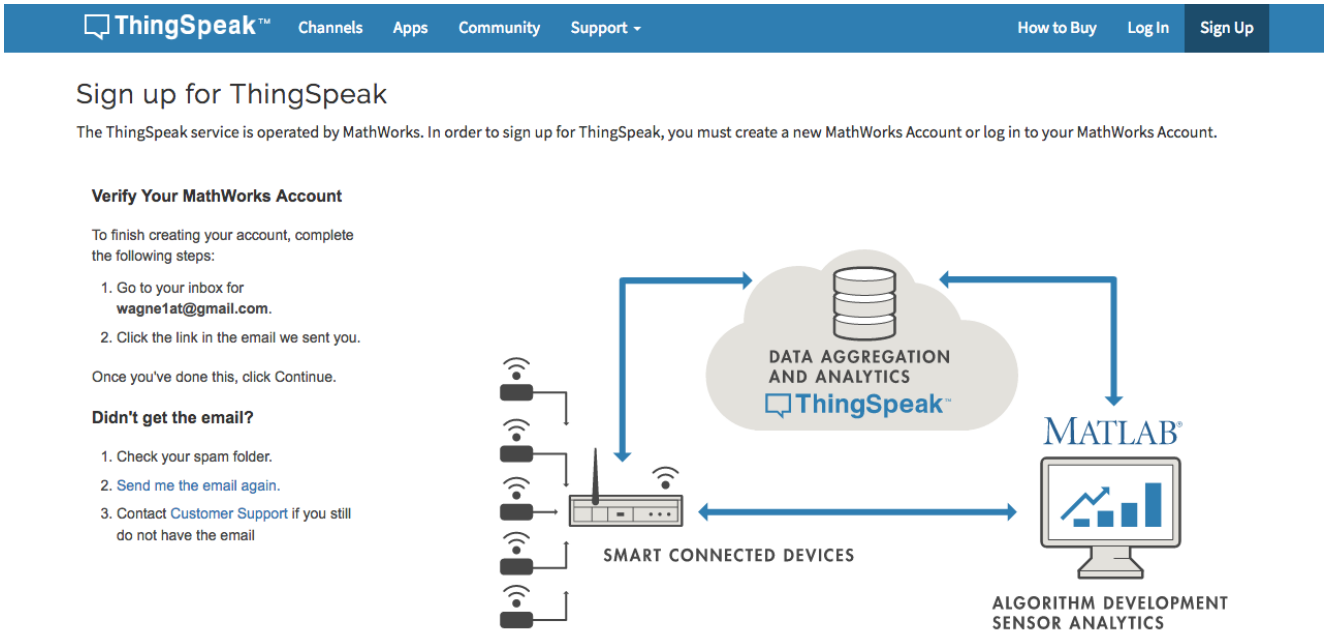


Step 3: Fill out your email address, username, password, country, first name, and last name.



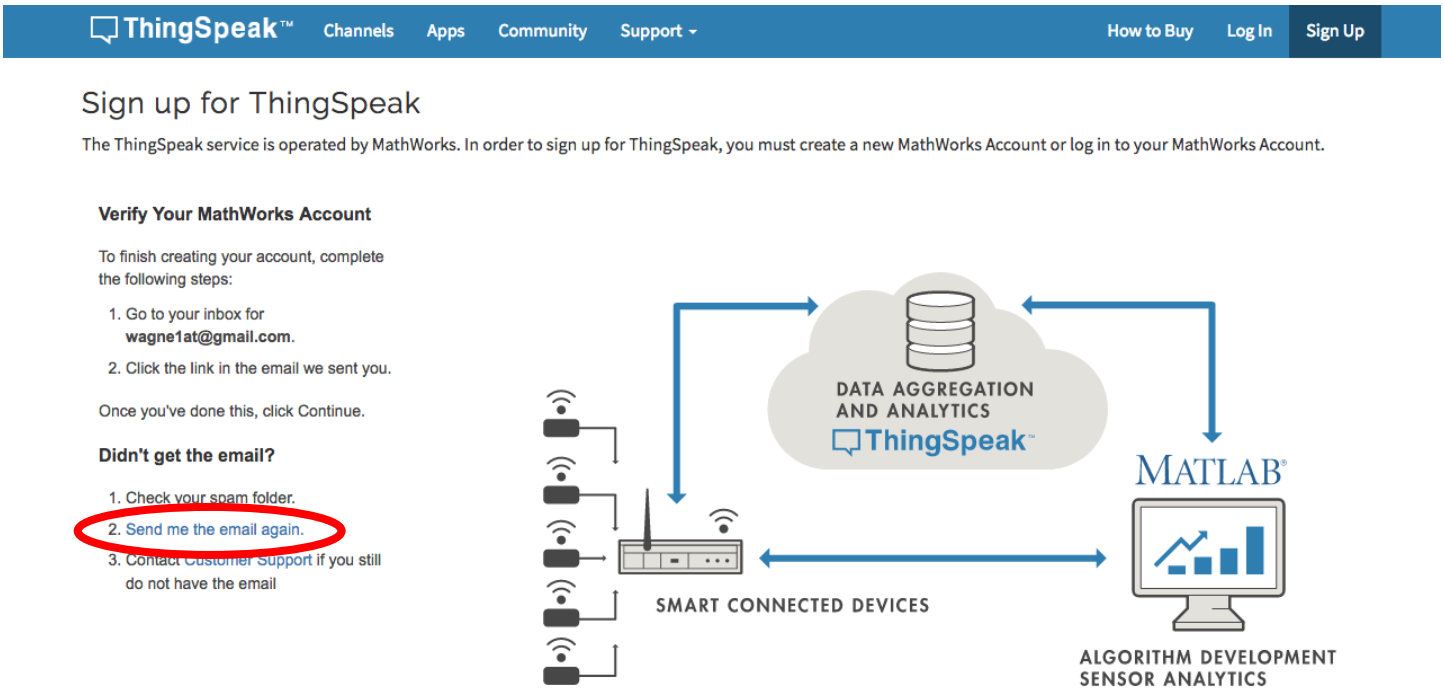
Step 4: Click Continue.

Step 5: Your screen will look like this:

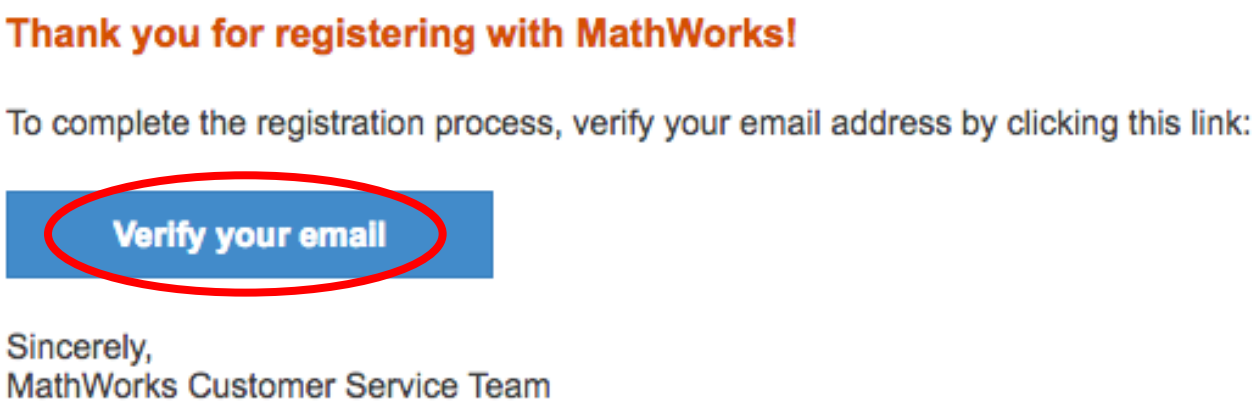


Step 6: Open a new tab. Check your email address for a new message from "service" with a subject line reading "Verify email address." Open this email and move to Step 7.

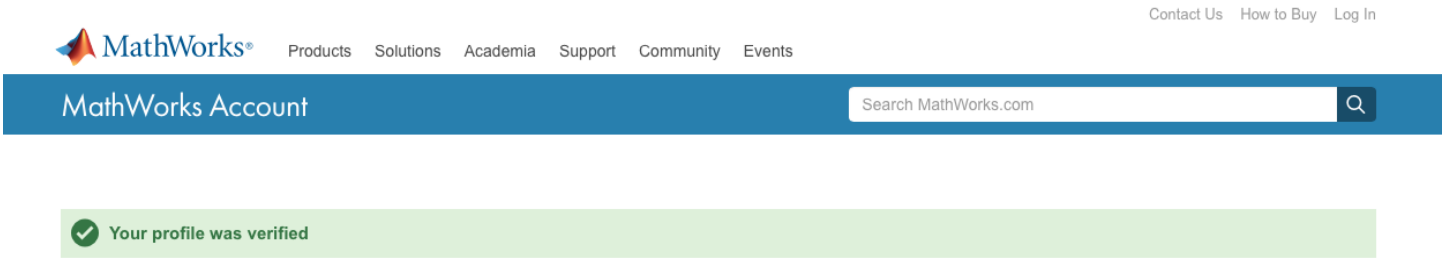
***Note: If the email is not in your inbox, check your spam folder. If it's not there, choose the "Send me the email again" option in the window with ThingSpeak (see image below).*



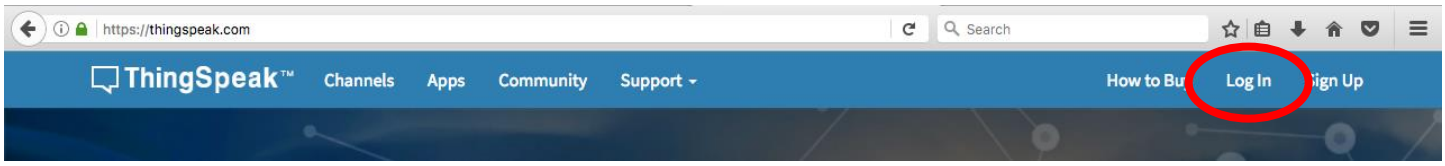
Step 7: The message should look like the image below. Click "Verify Your Email."



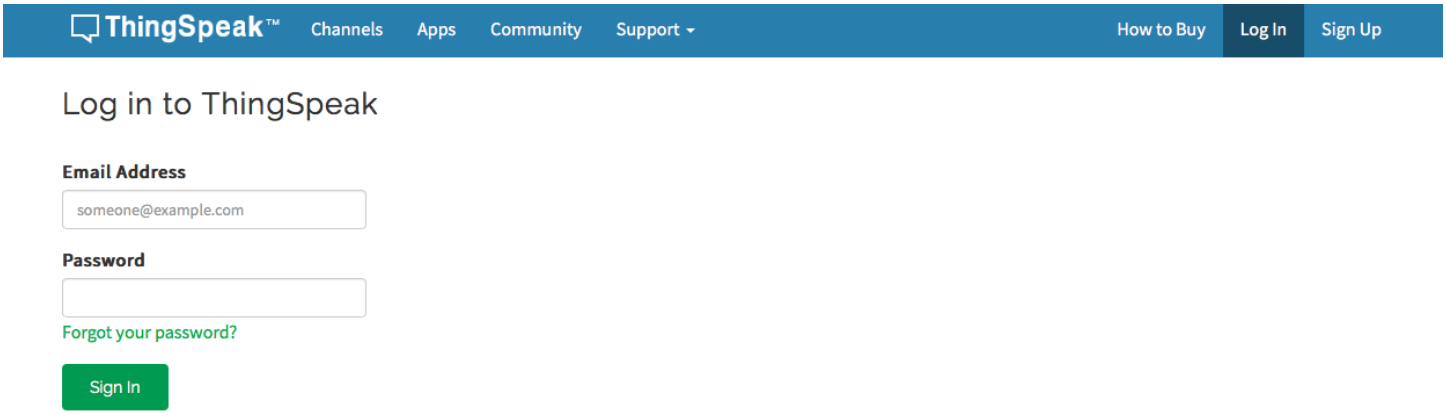
Step 8: A confirmation page will appear. Notice that this has opened a new tab to MathWorks Inc, the parent of ThingSpeak.



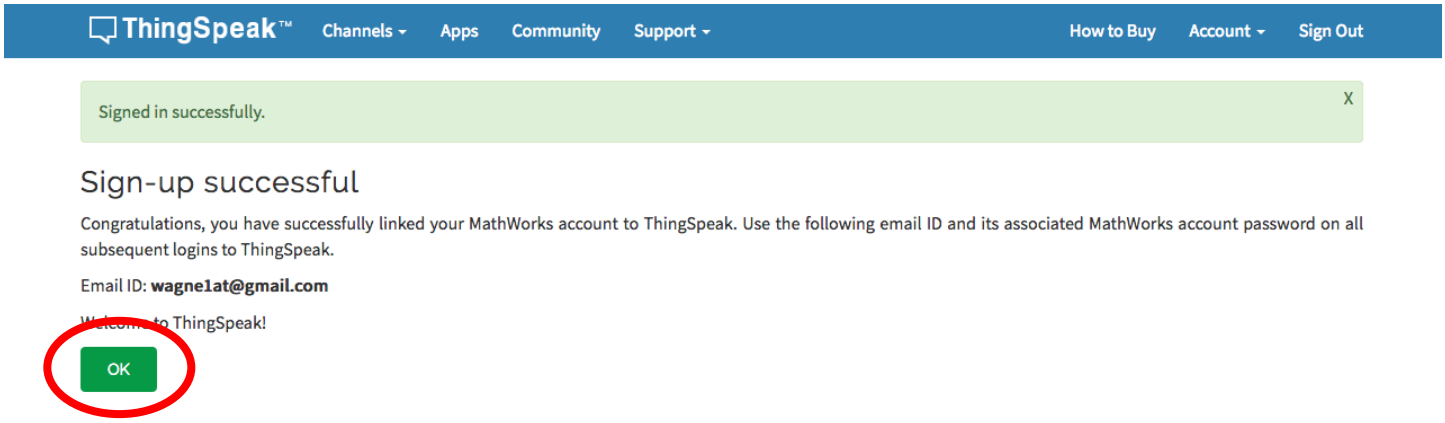
Step 9: Go back to www.thingworks.com. Click Log In.



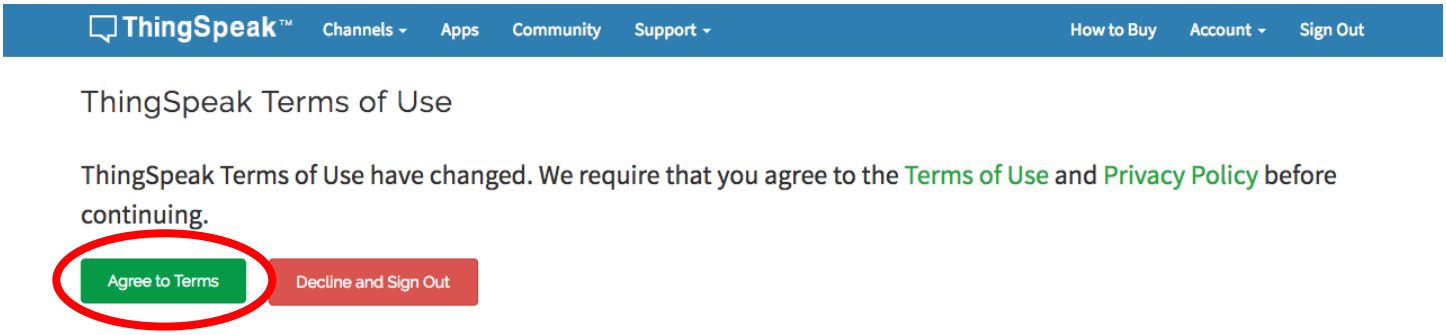
Step 10: Type in your email address and password. Then click "Sign In."



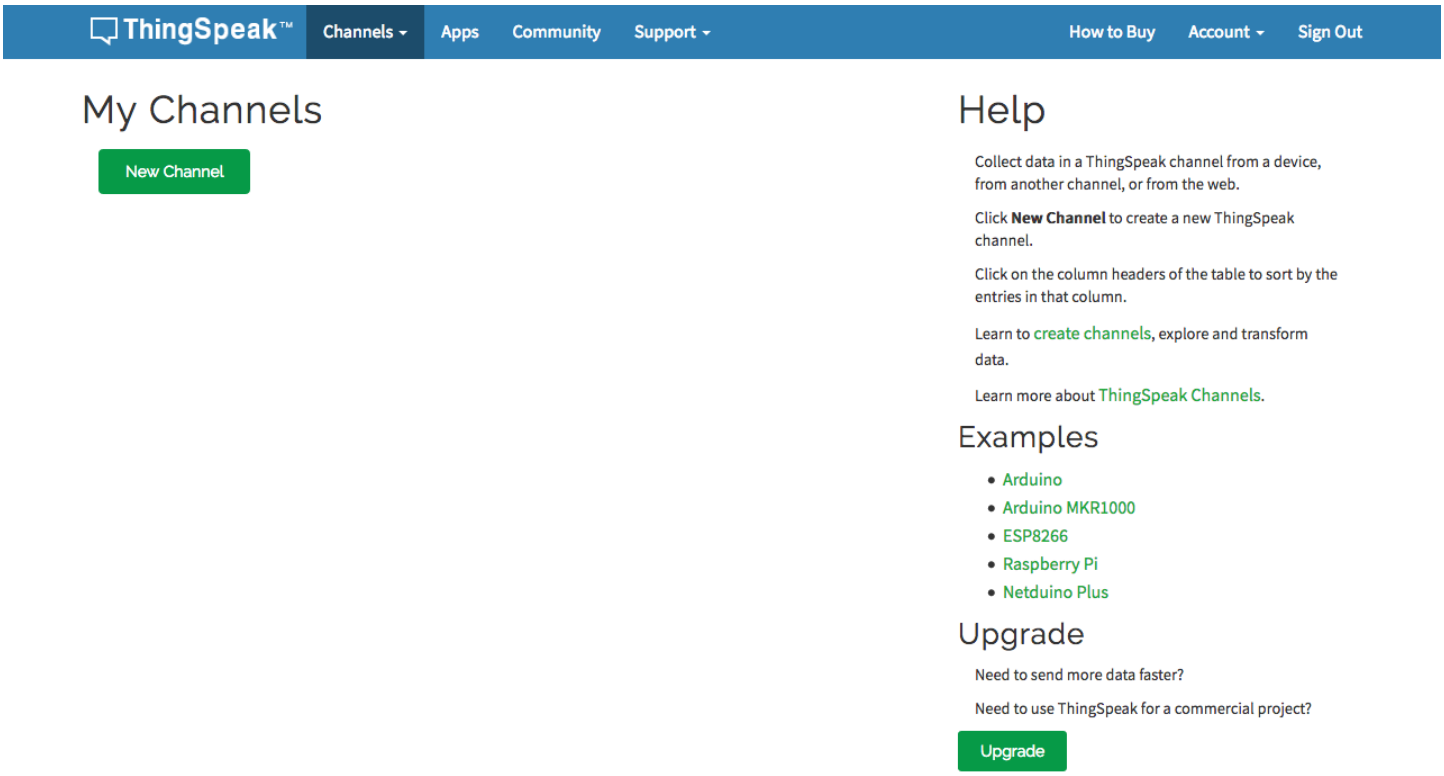
Step 11: A confirmation page should appear. Click OK to proceed.



Step 12: Agree to the terms of use.



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.

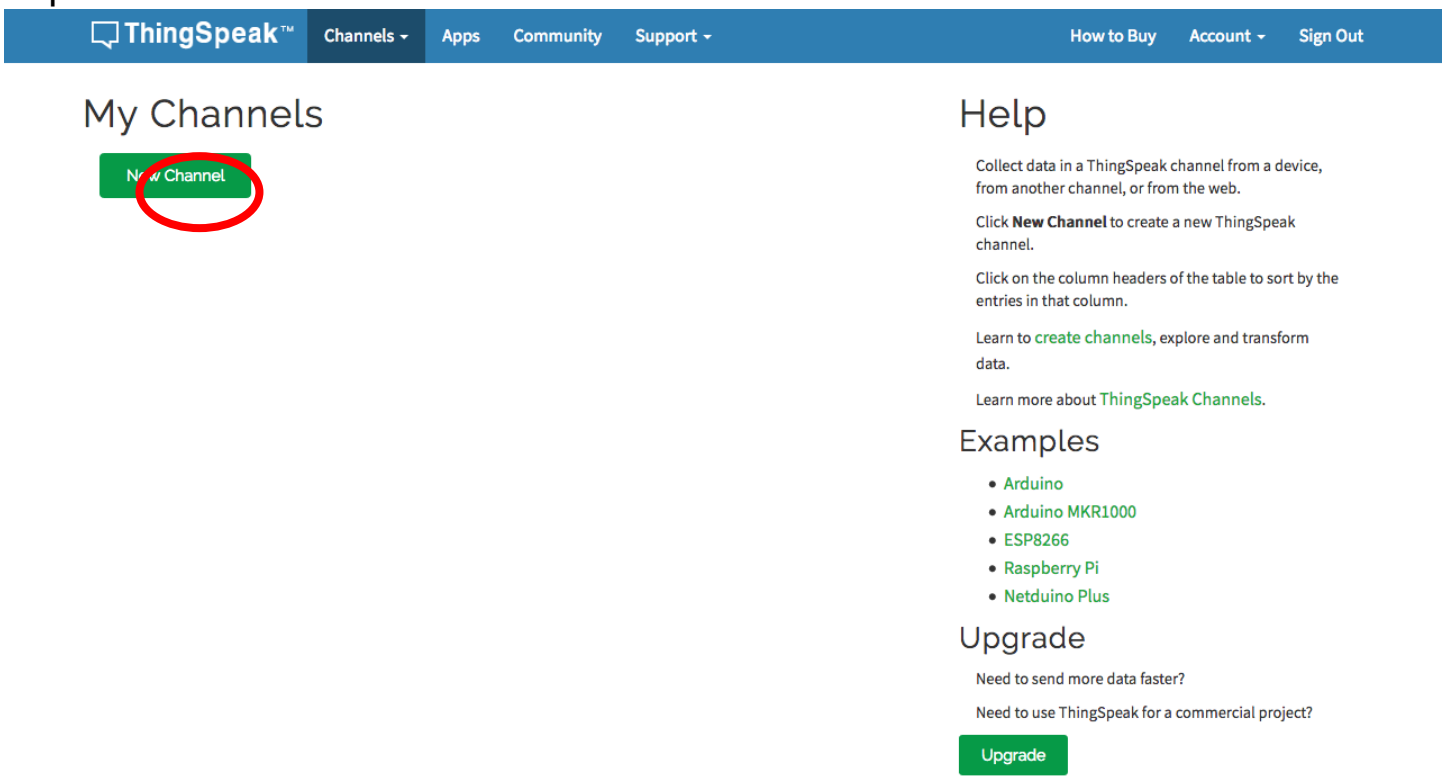


Section 2: Setting Up the ThingSpeak Channels

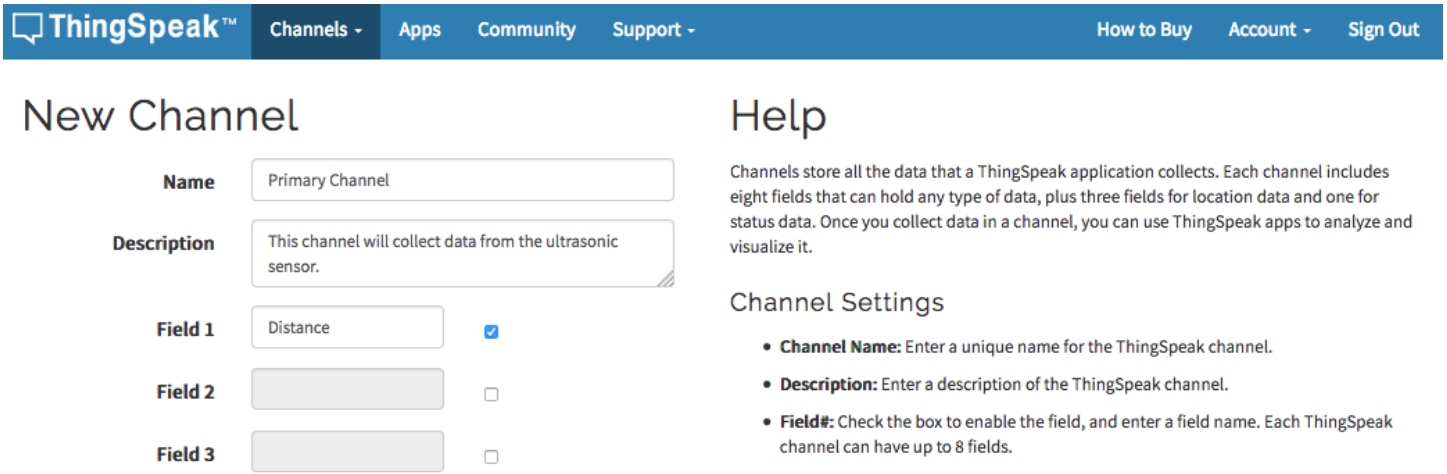
We are going to set up two channels in ThingSpeak. The first channel will be called the "Primary Channel" because it will collect data that it receives from an ultrasonic sensor *and* it will also send data to another channel. The second channel will be called the "Observation Channel" because it will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

Part 1: Primary Channel Setup

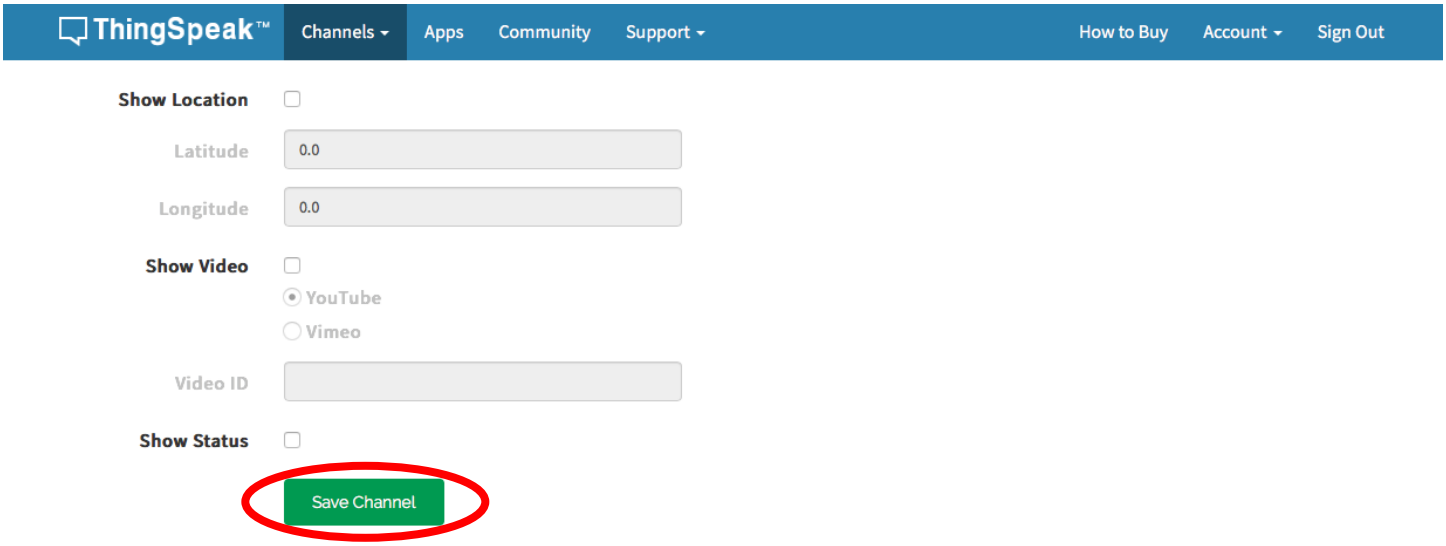
Step 1: Click New Channel



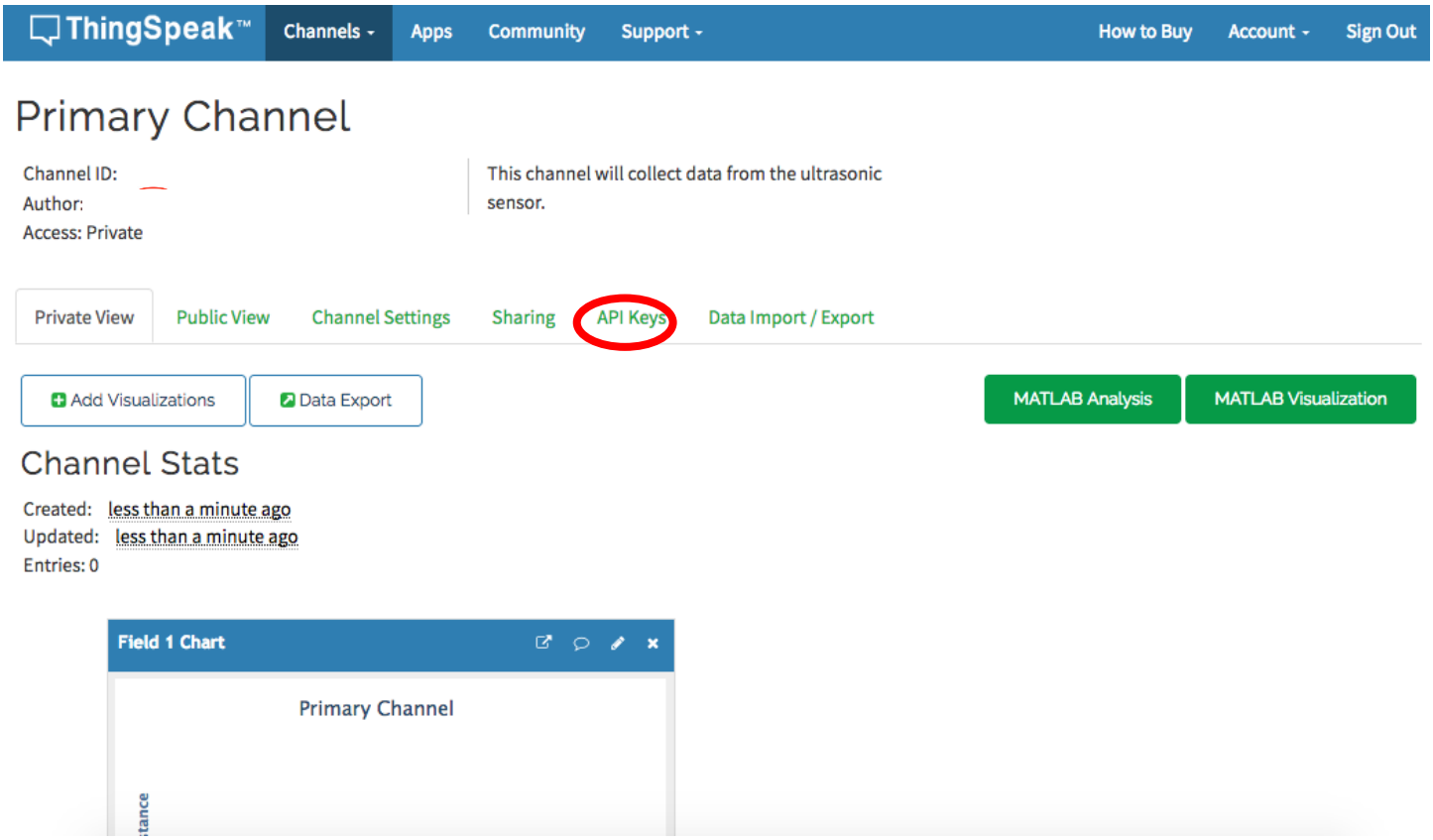
Step 2: Name the channel, write its description, and create its field. In this tutorial, we will only receive data on one variable (distance), so only one field is needed.



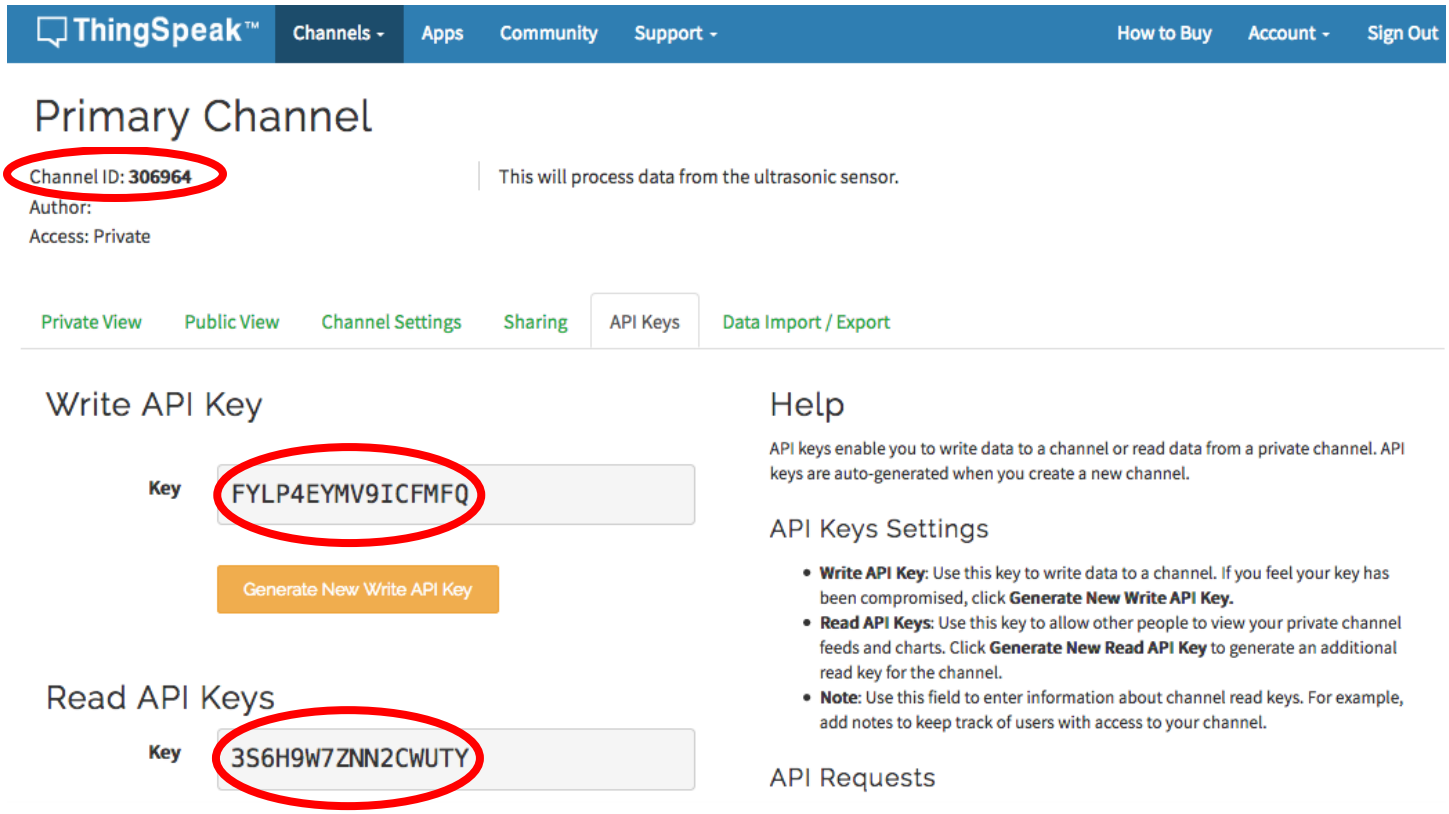
Step 3: Scroll to the bottom and click "Save Channel."



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



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.

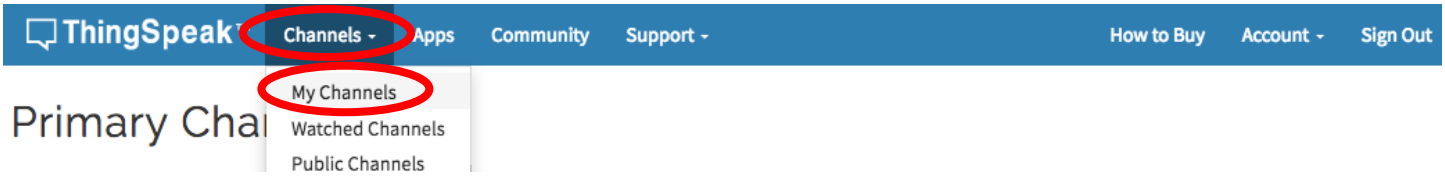


Congratulations! The first channel has been successfully set up. Proceed to Part 2 below.

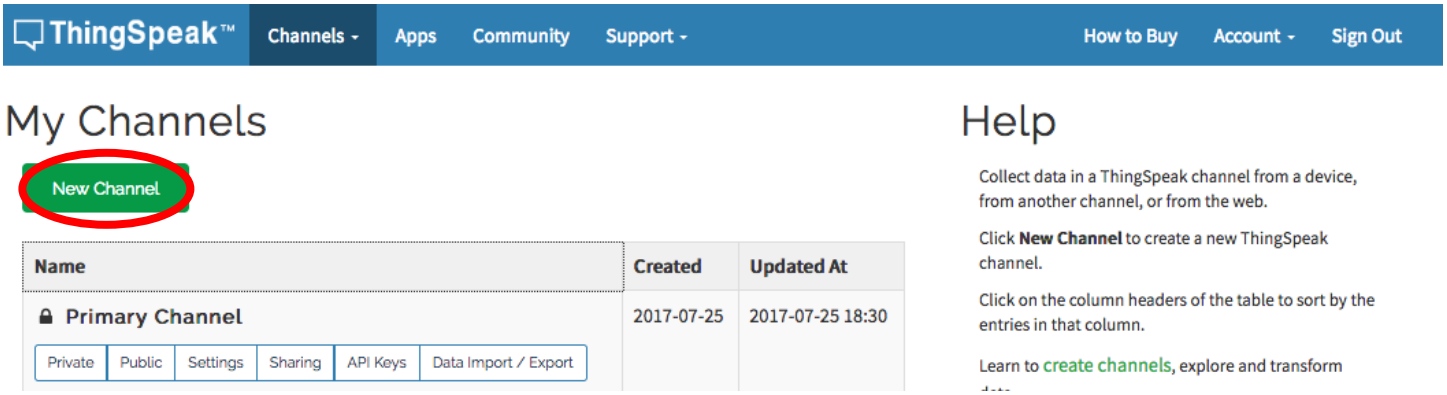
Part 2: Observation Channel Setup

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

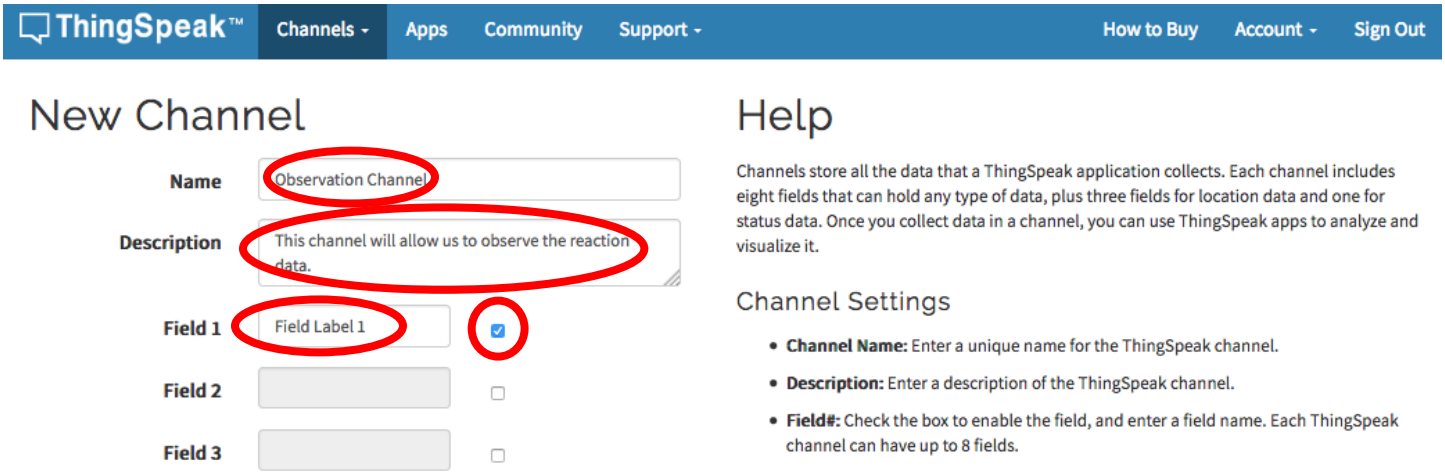
Step 1: Click Channels > My Channels.



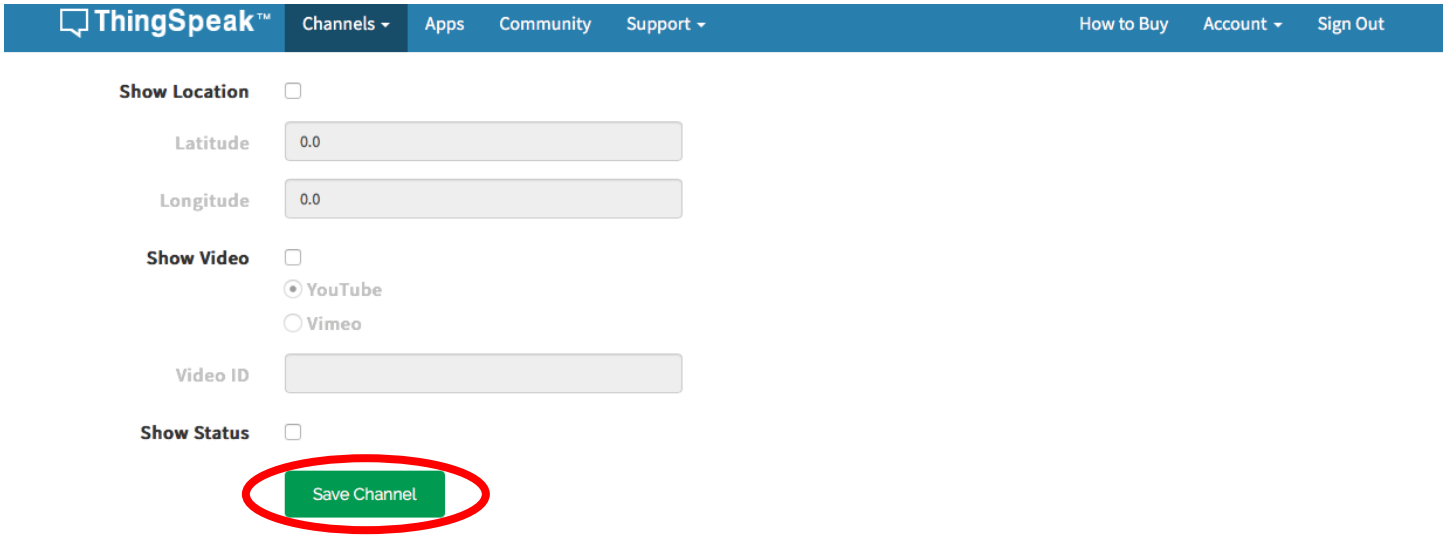
Step 2: Click New Channel.



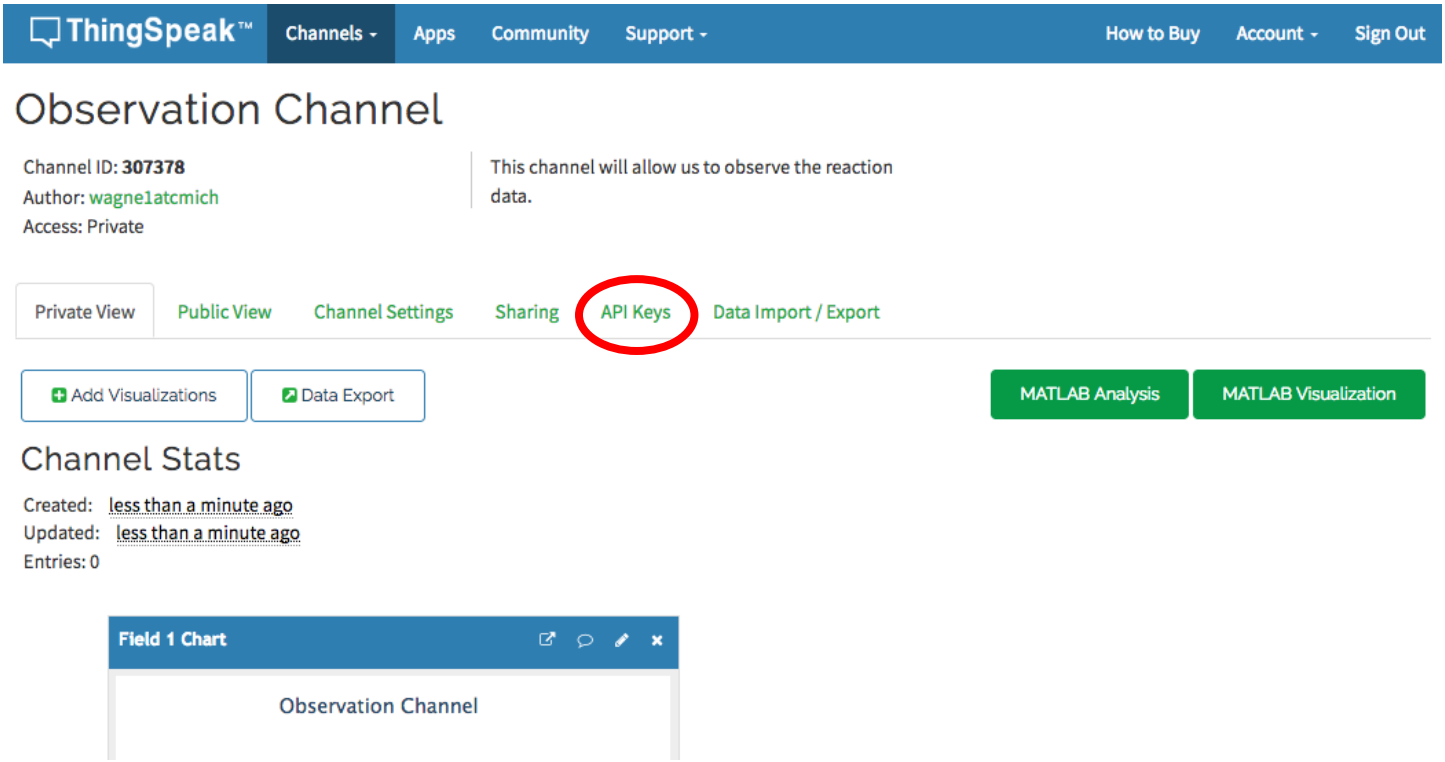
Step 3: Create the name, description, and field label for this channel.



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



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

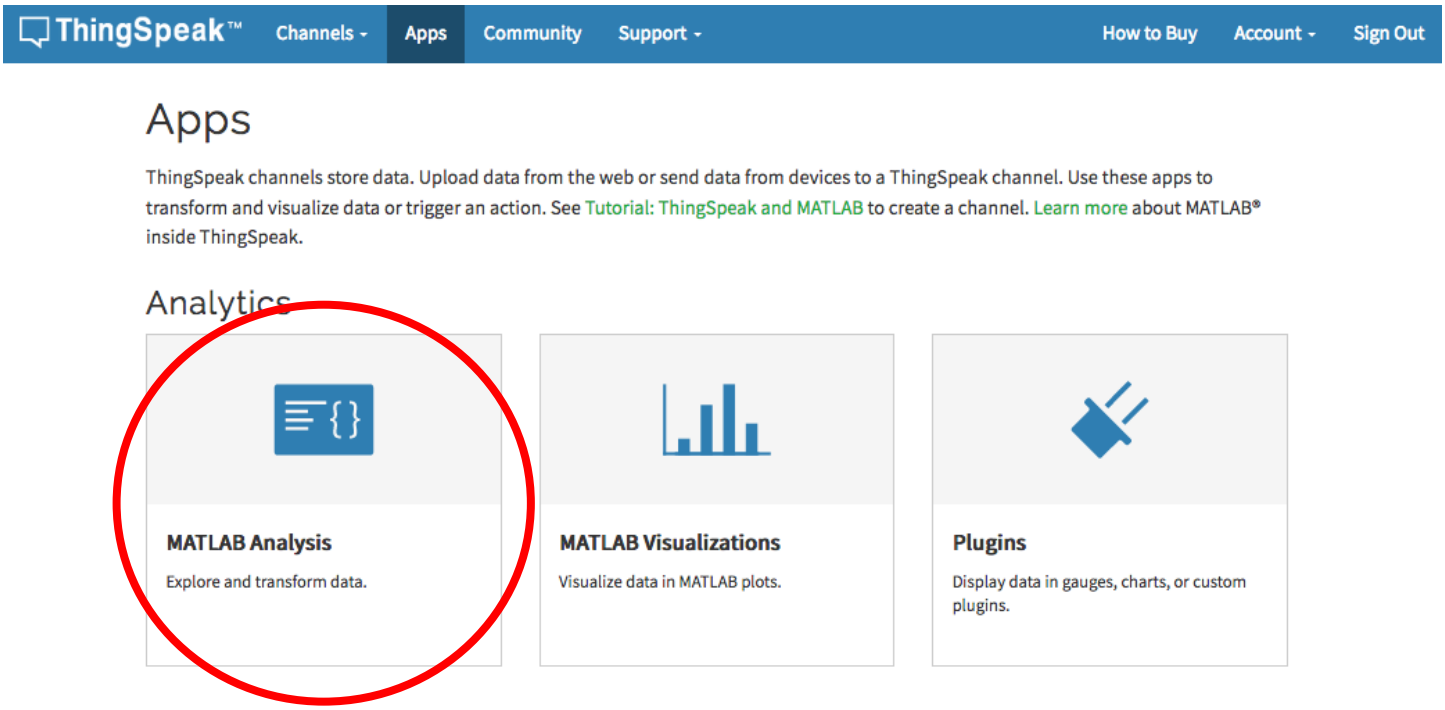
Section 3: Setting Up MATLAB & ThingSpeak Reactions

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.

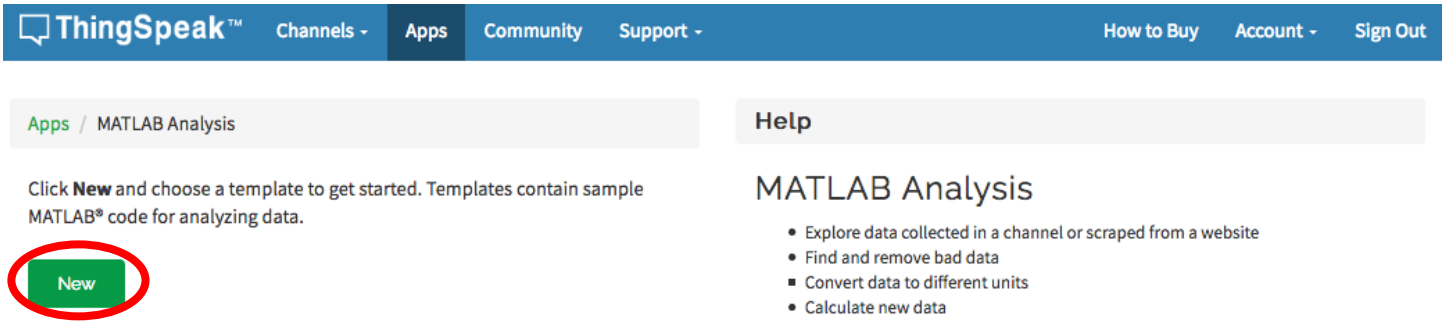
Part 1: MATLAB Analysis Setup

Step 1: Click Apps.

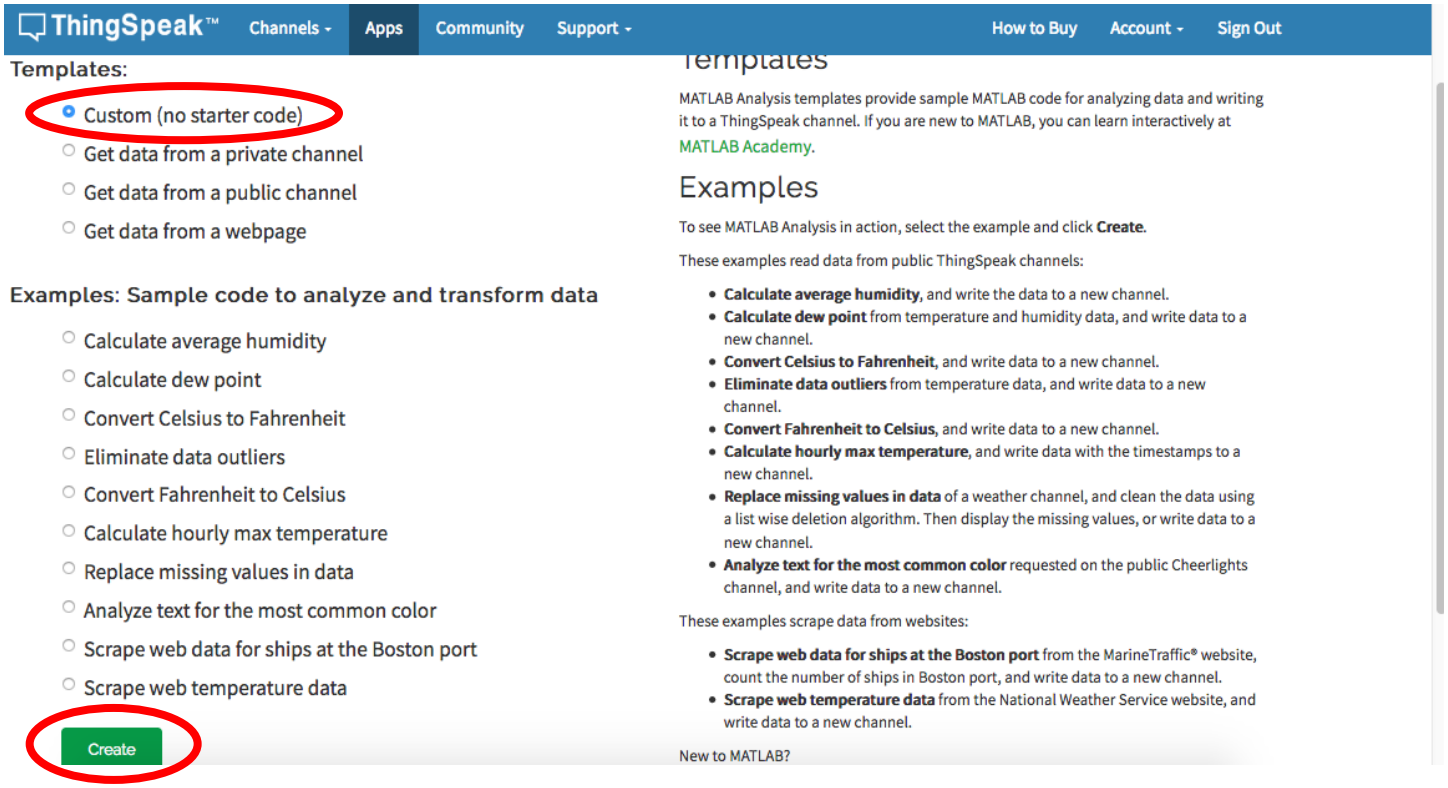
Step 2: Click MATLAB Analysis.



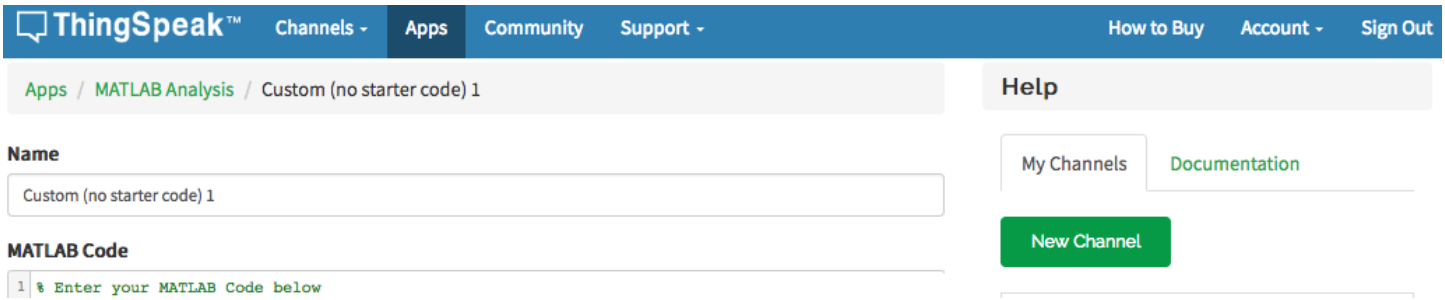
Step 3: Click "New."



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



Step 5: Our browser now looks like this:

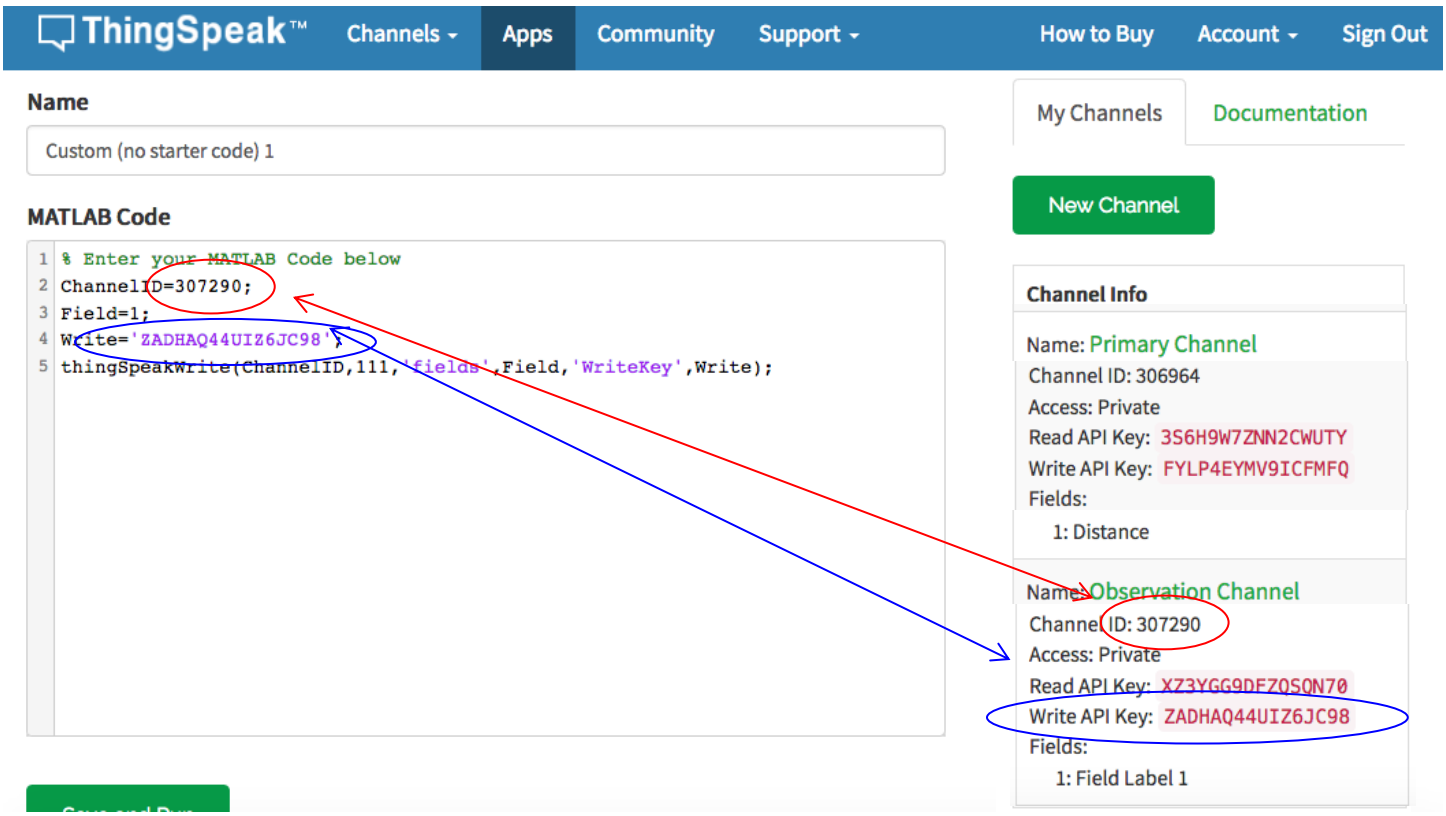


Step 6: We will now enter our MATLAB Code. The portions circled in red indicate information that you will need to fill in from Google Sheets that correspond to your unique channels. *The blue circles (●) indicate where the proper API Keys must be added (see Step 7).*

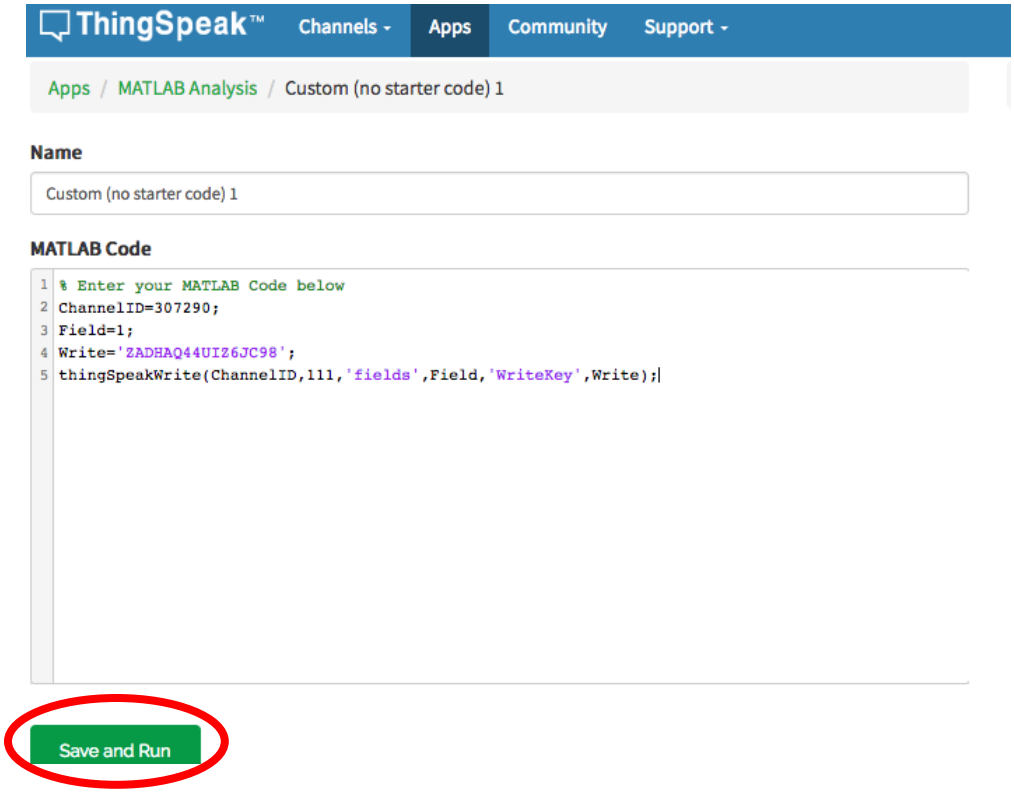
***Note: We recommend typing in the following code by hand to minimize syntax errors on MATLAB.*
***Also Note: Type the code in exactly as it appears (including all punctuation). It is case sensitive.*

```
ChannelID=●;
Field=1;
Write='●';
thingSpeakWrite(ChannelID,111,'fields',Field,'WriteKey',Write);
```

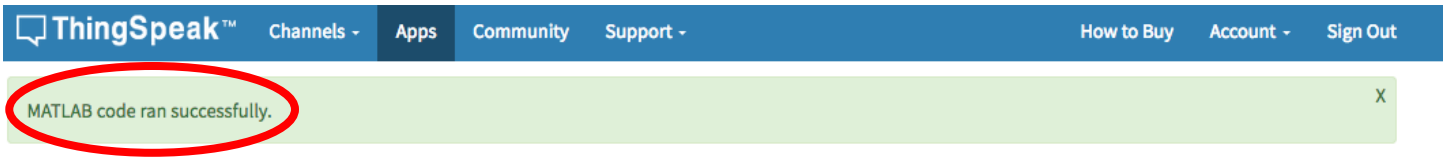
Step 7: Once your code is in place, we now need to input the correct API Keys.



Step 7: Click "Save and Run."



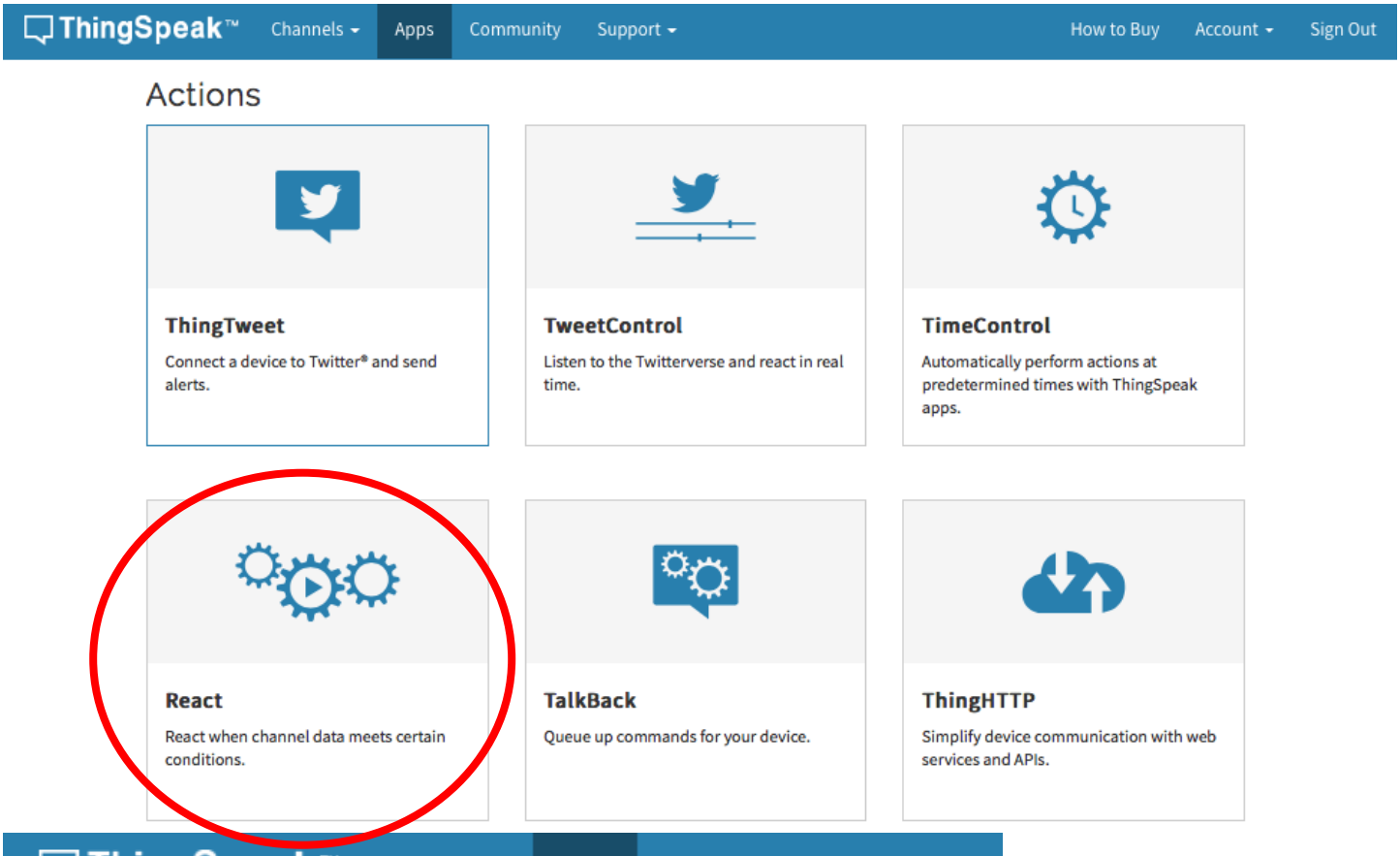
Step 8: If you did this correctly, a green message should appear at the top of the screen (you may need to scroll up to the top), reading "MATLAB code ran successfully."

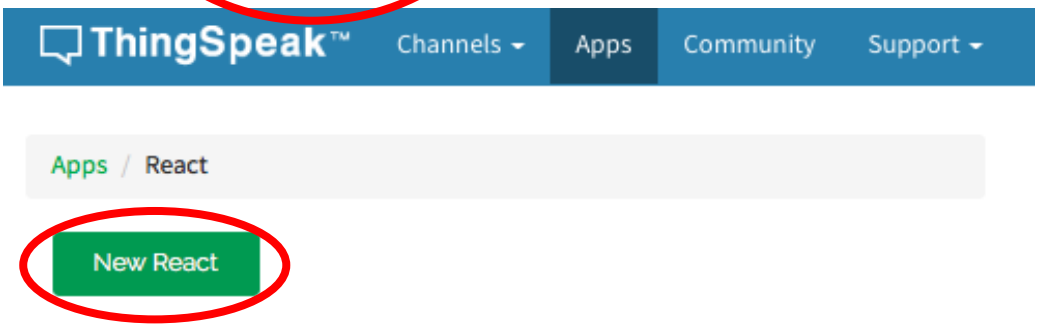


Part 2: Reaction Setup

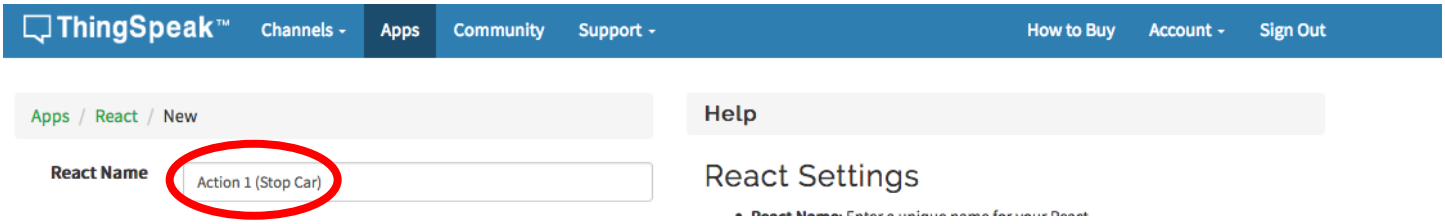
Step 1: We now need to set up a reaction that uses our new MATLAB code. This will allow us to program changes to our car based on the data that it receives. To do this, first click "Apps."

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

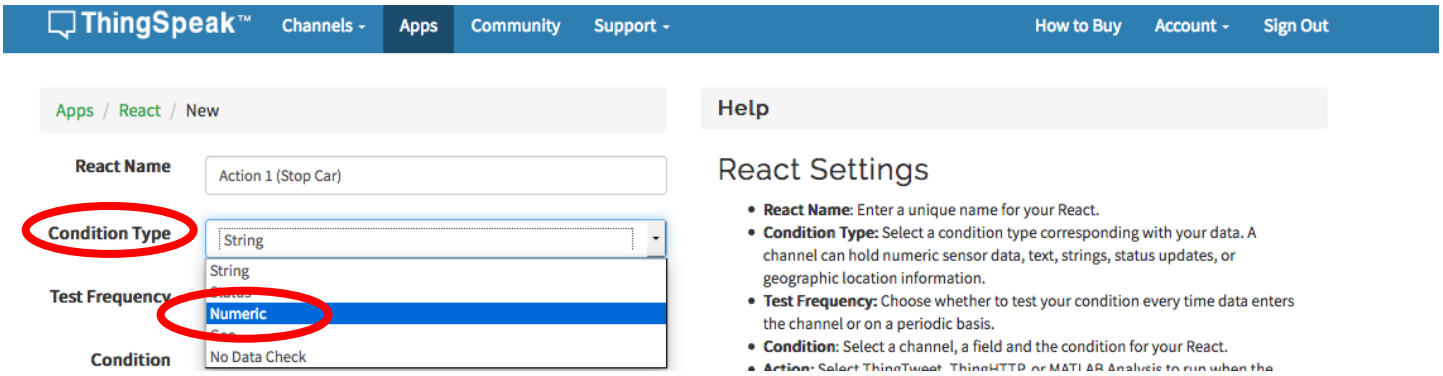




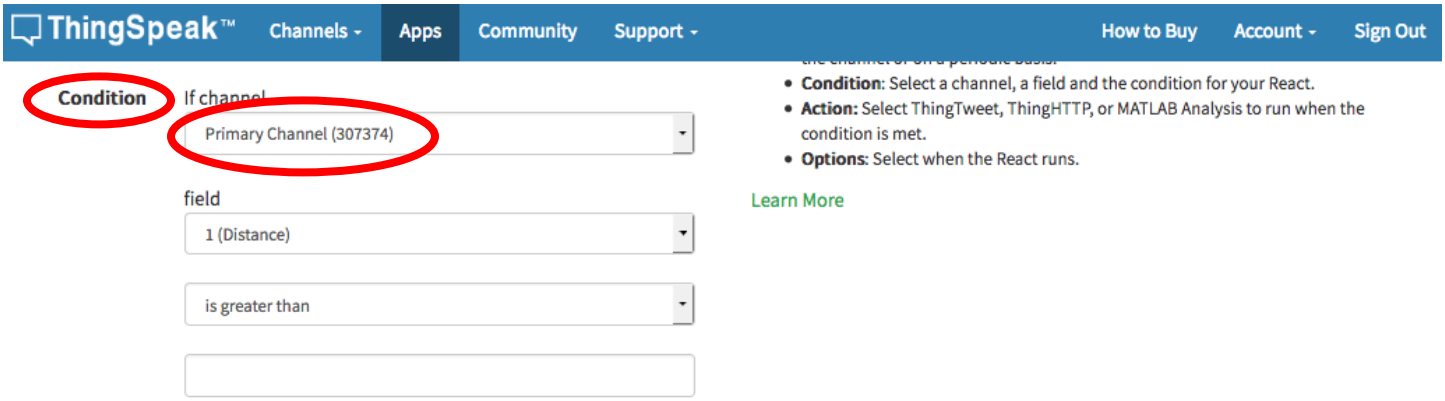
Step 4: Fill out the name of this reaction. We called ours Action 1 (Stop Car).



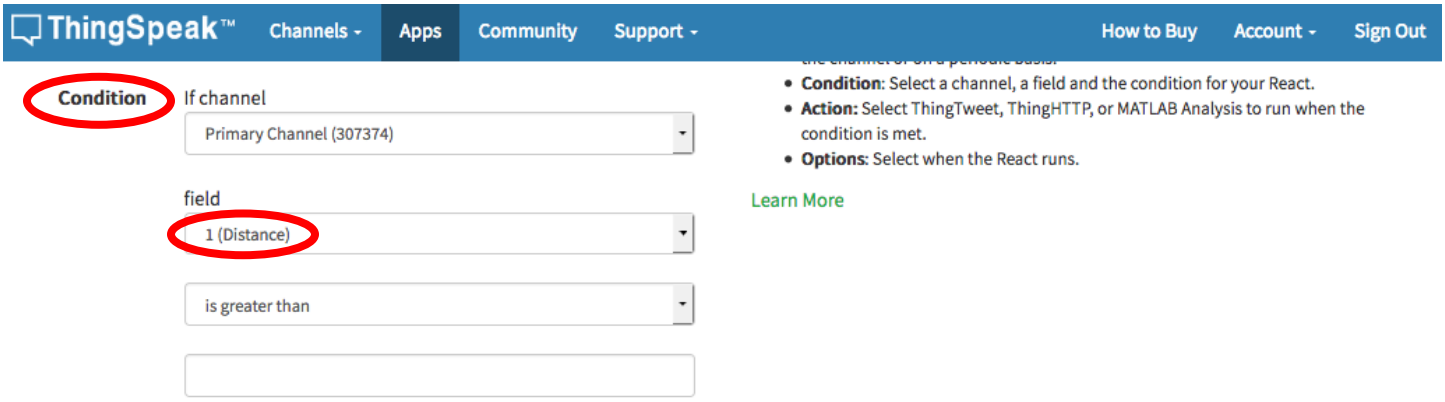
Step 5: Change the condition type from String to Numeric.



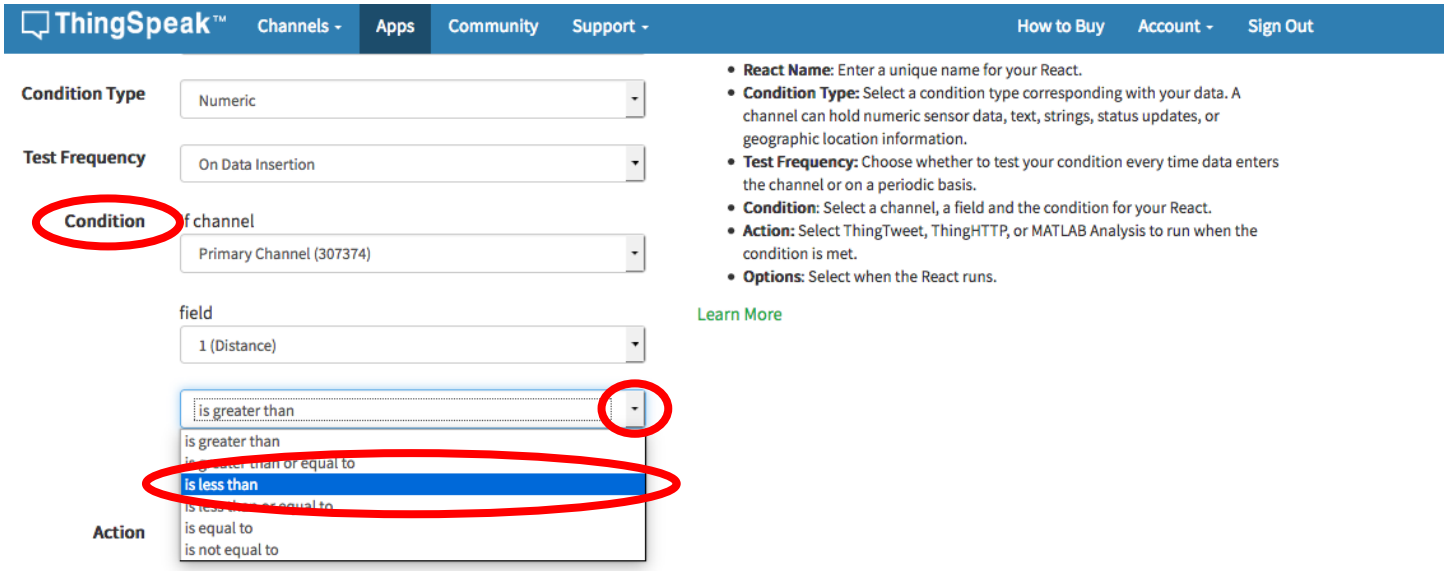
Step 6: Set the condition. This will perform the action that we want the car to make. It is set up as an "if, then" logic command. We first set the "If channel" to the Primary Channel.



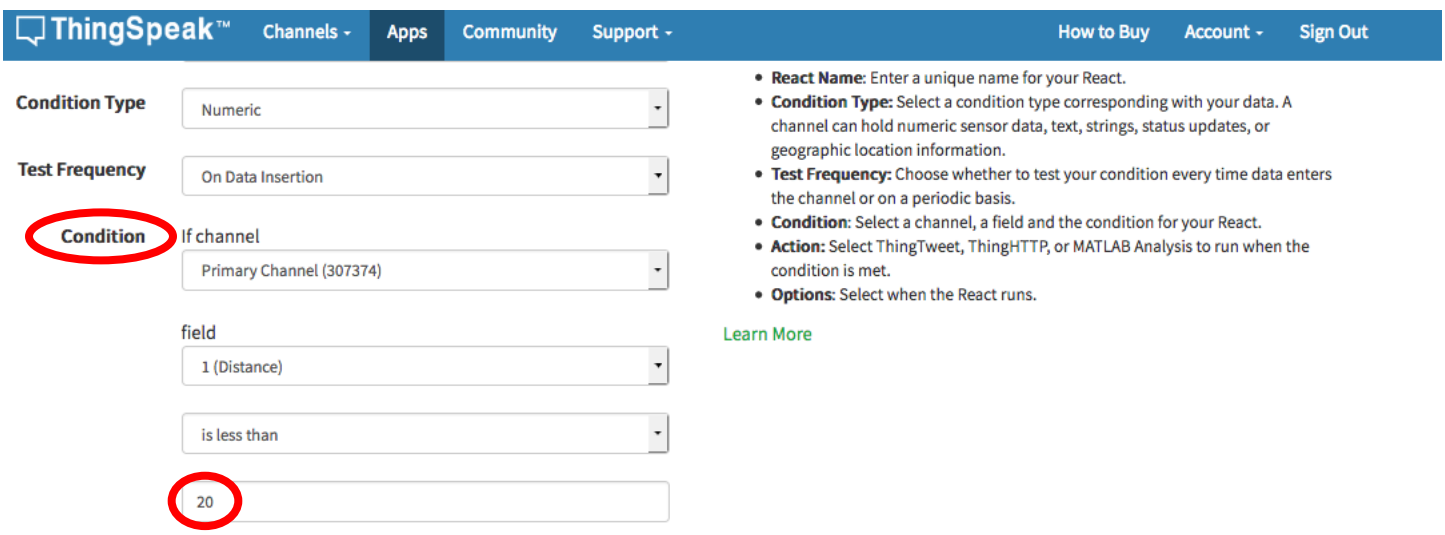
Step 7: We then choose the field. Since we want to analyze the distance, we will leave this field portion as it is.



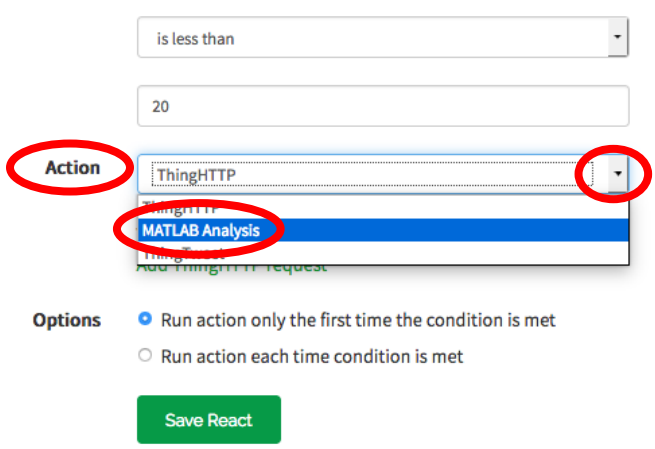
Step 8: We now choose the condition statement. In our case, we want our car to stop when the ultrasonic sensor senses an object less than twenty units away. We set the condition as follows:



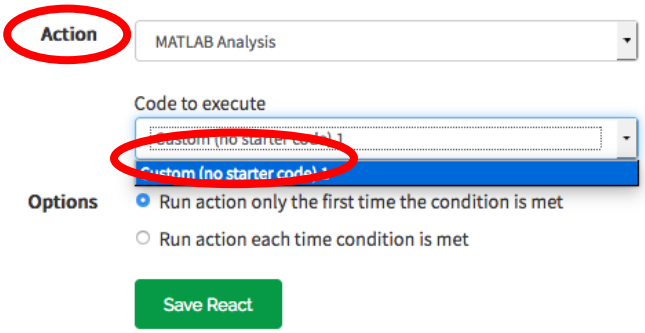
Step 9: Fill in the numeric condition. Because we want our car to stop when the ultrasonic sensor senses an object less than twenty units away, we enter "20" into the field.



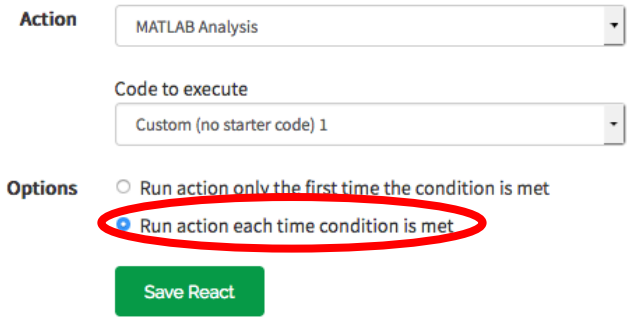
Step 10: We now need to set the action using MATLAB Analysis. First click on the dropdown menu, then choose MATLAB Analysis.



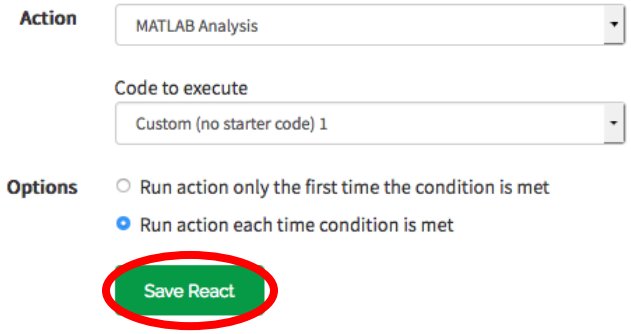
Step 11: Set the Code to execute "Custom (no starter code) 1."



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



Step 13: Click "Save React."



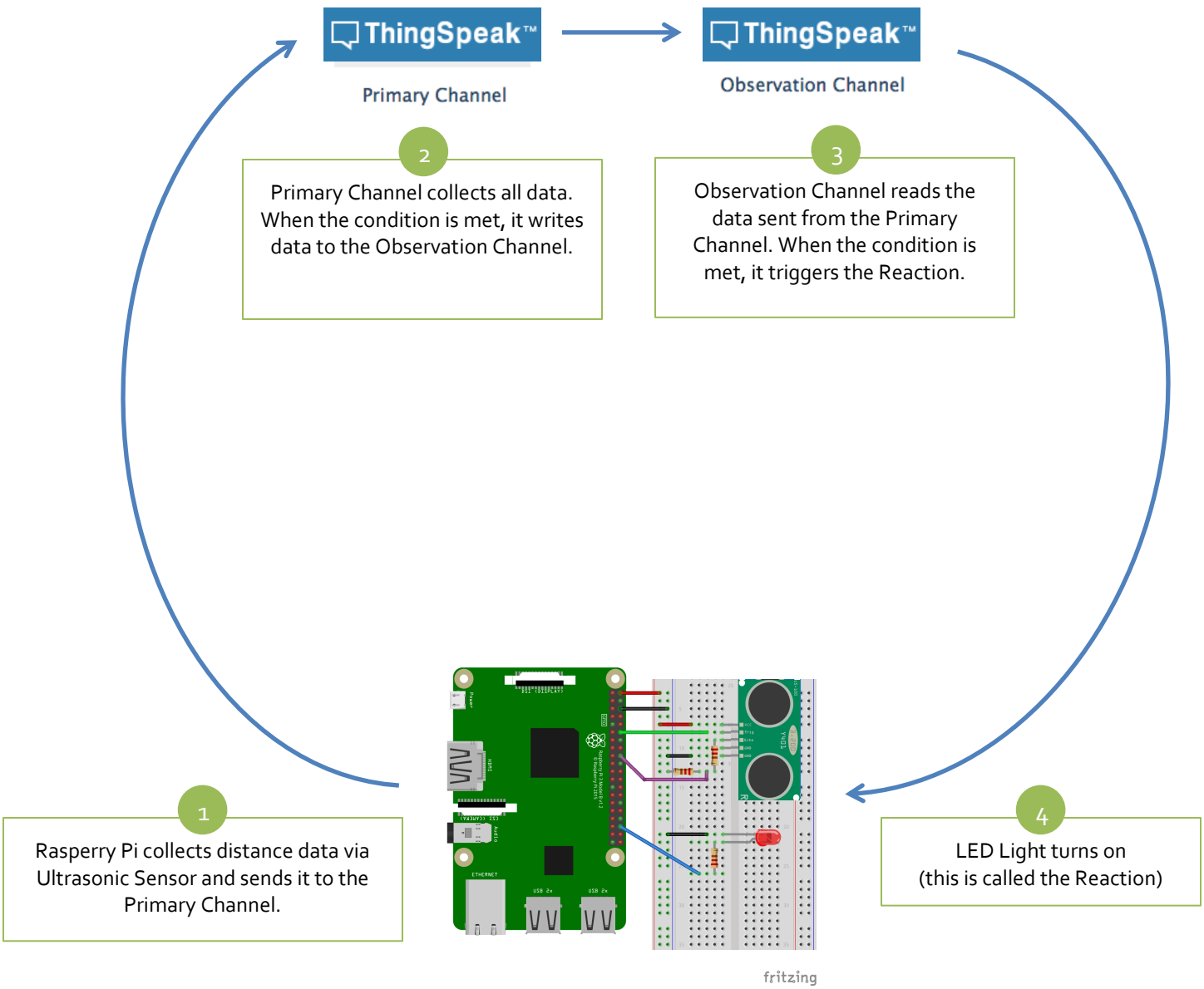
Section 4: Code & Sensor Setup

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.)

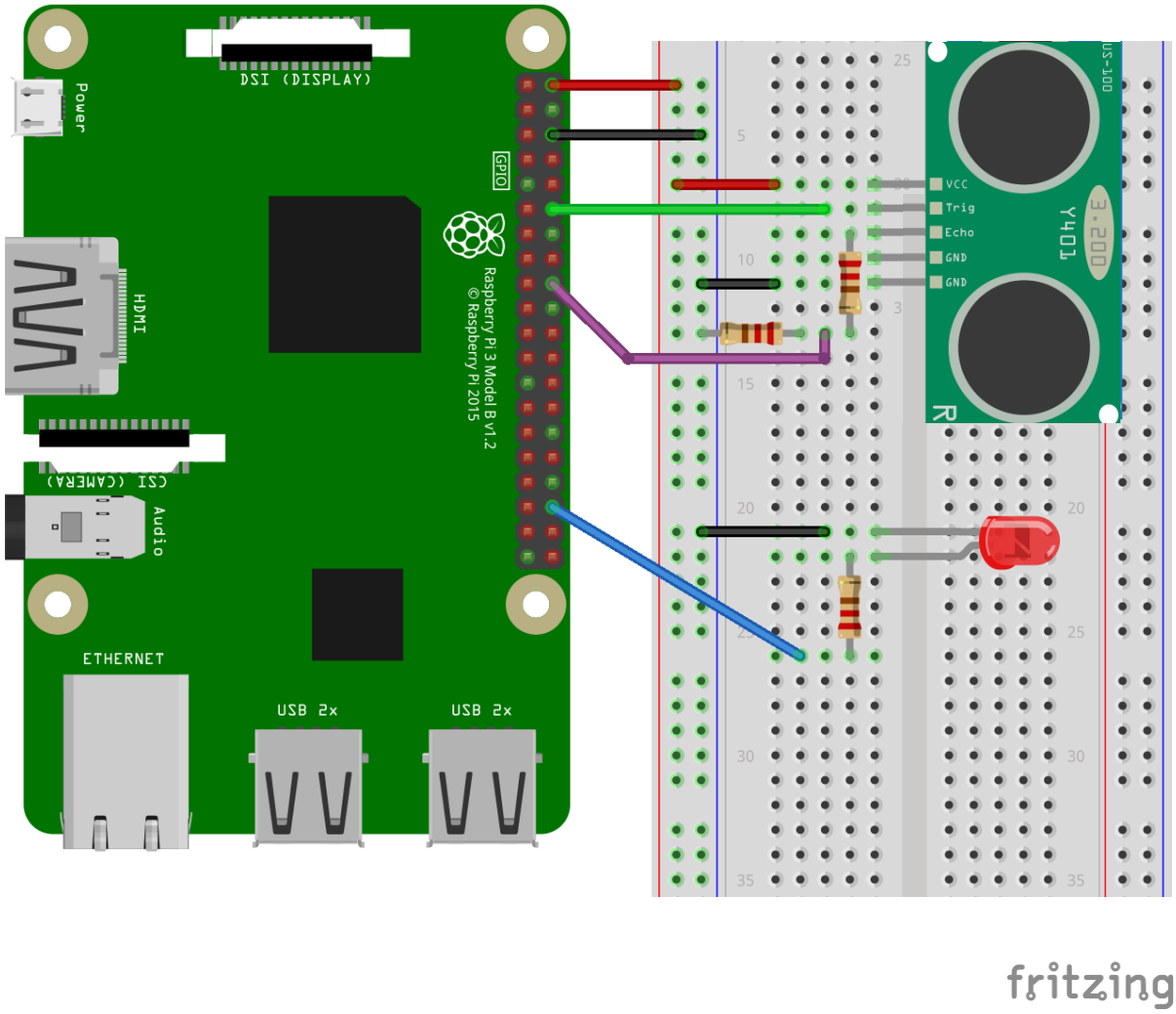
We will complete this section in three parts.

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.)



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:

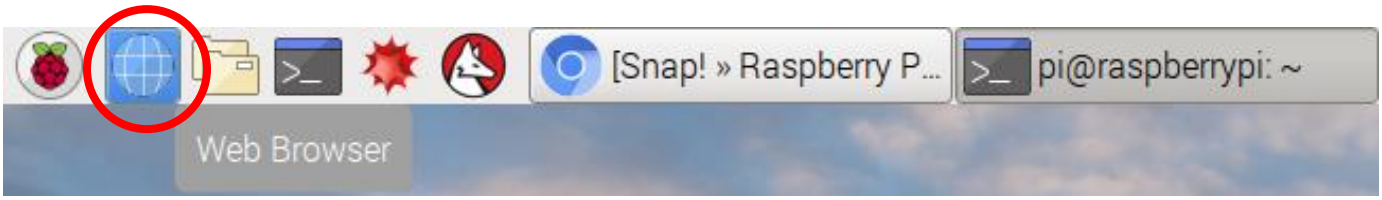
pip3 install http.client

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

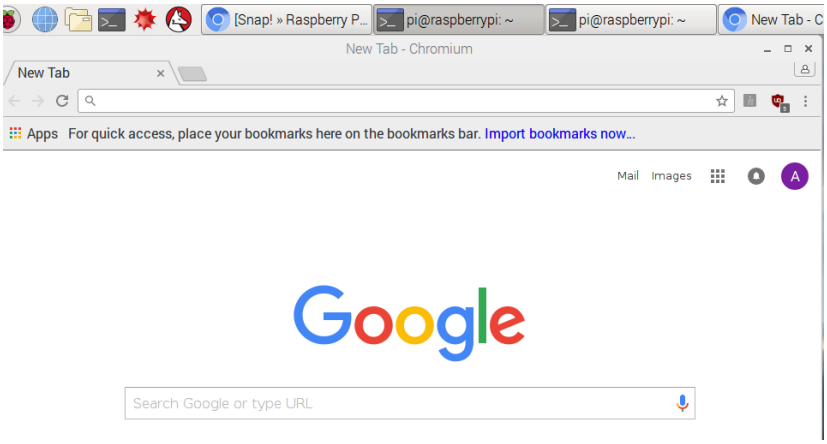
tutorial_test_sensor.py

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

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

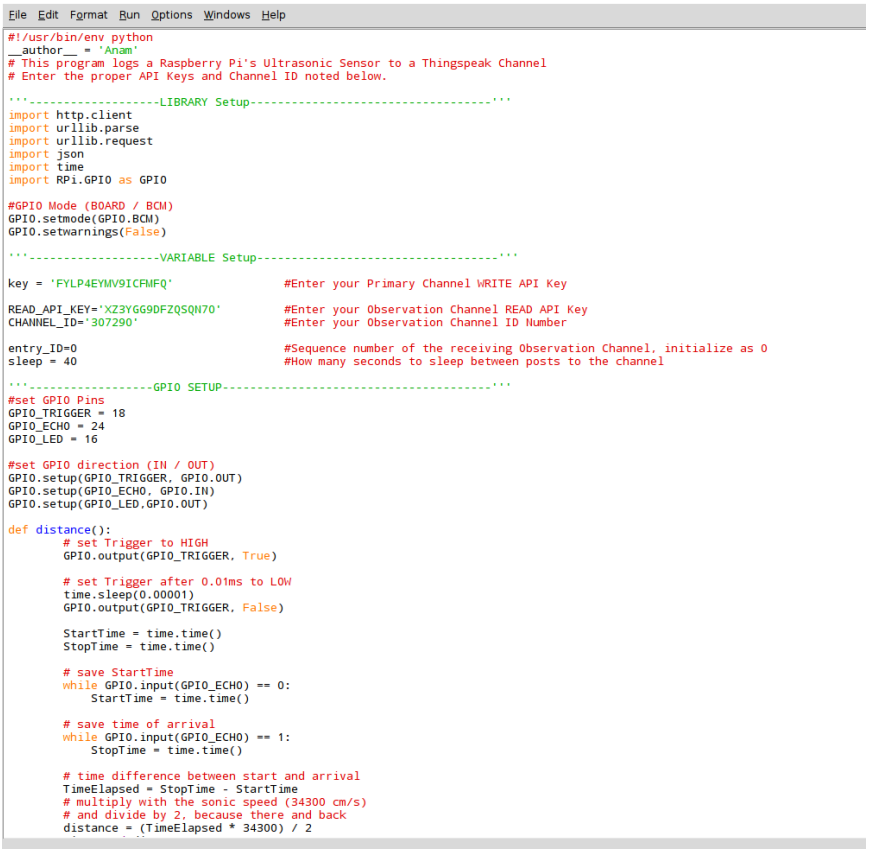


Step 2: It should appear like the screen below.



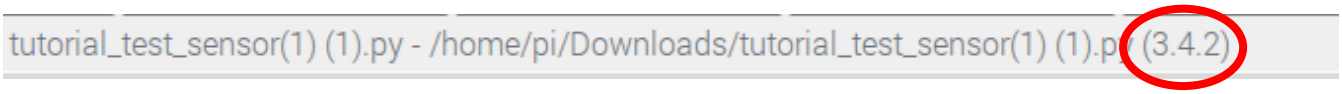
Step 3: Navigate to the document (either via email or link). Then download the file. It should appear at the bottom of your Web Browser. Click on the file.

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 = 'FYLP4EYMV9ICFMFQ'                #Enter your Primary Channel WRITE API Key

READ_API_KEY='XZ3YGG9DFZQSQN7O'         #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
```

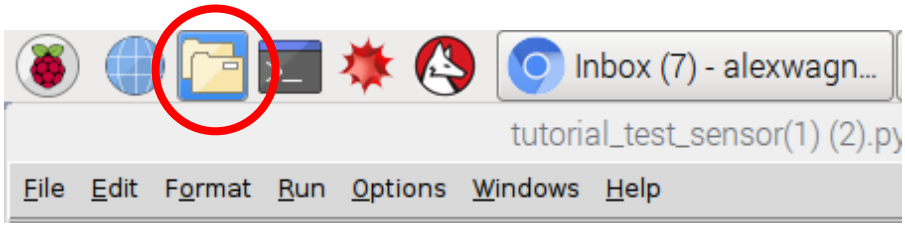
Note: This code must be run in Python 3. To be sure you are in Python 3, check the file name at the top of the window. It should begin with "3" like this:



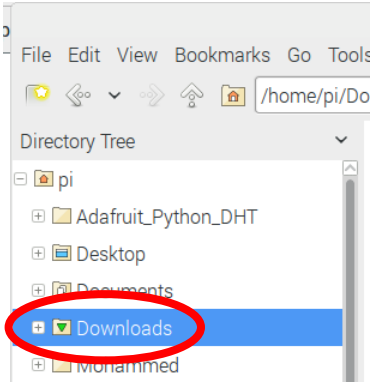
If your file name looks like this, you may proceed to Step 5.

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

Mini Step 1: Click the Folder Icon.

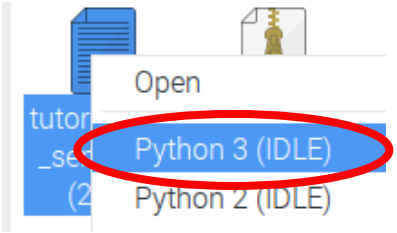


Mini Step 2: Click Downloads

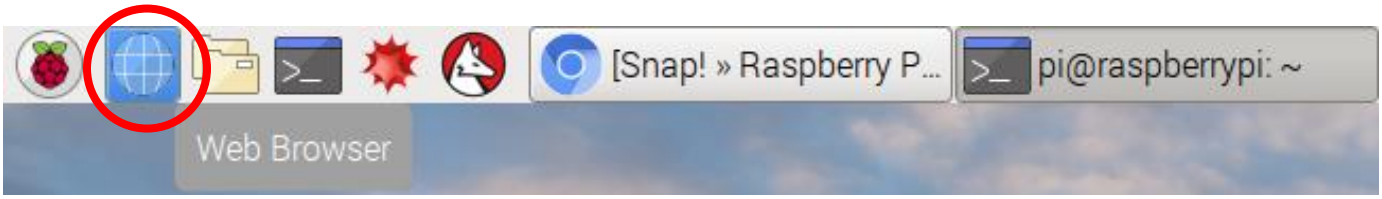


Mini Step 3: Locate your file and right click on it.

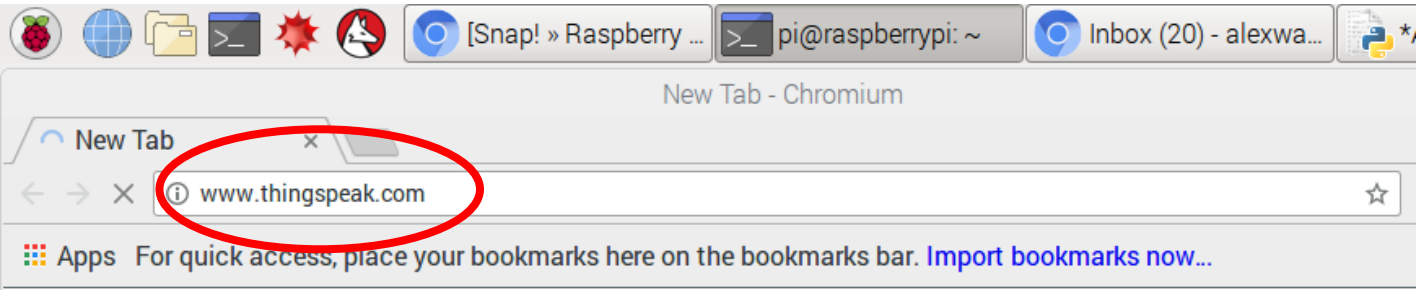
Mini Step 4: Click "Python 3 (IDLE)"



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.

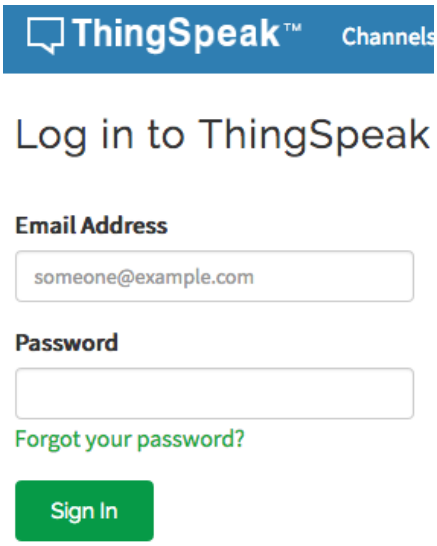


Step 6: Navigate to www.thingspeak.com



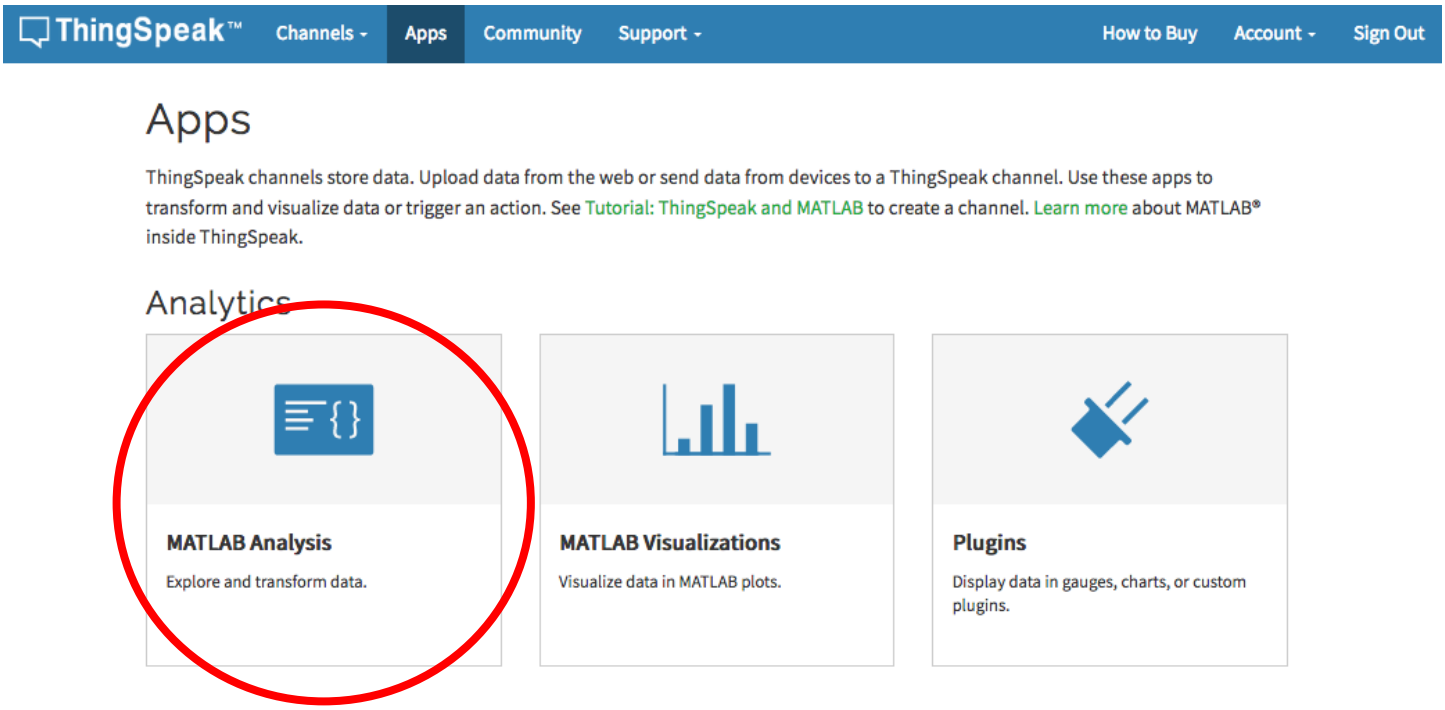
Step 7: Once there, click Log In.

Step 8: Enter your account information.

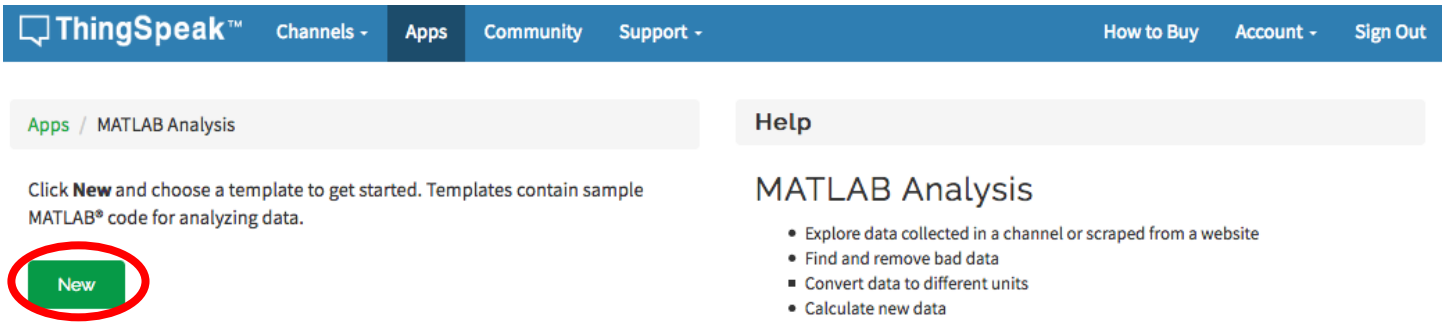


Step 9: To easily view all of the API Keys and Channel IDs, we will navigate to a MATLAB Analysis screen, which has all information listed at the right hand side of the page. First, click Apps.

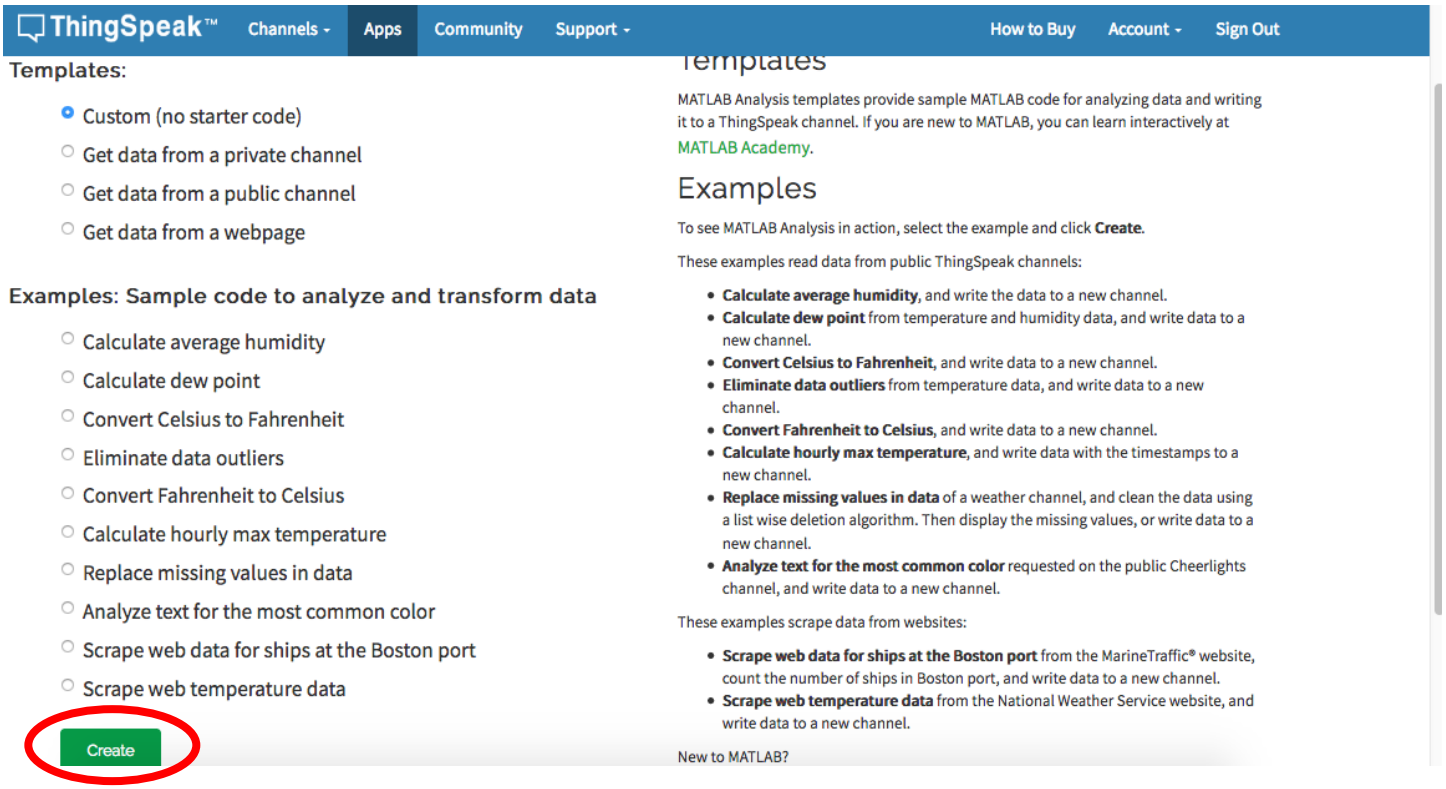
Step 10: Click MATLAB Analysis.



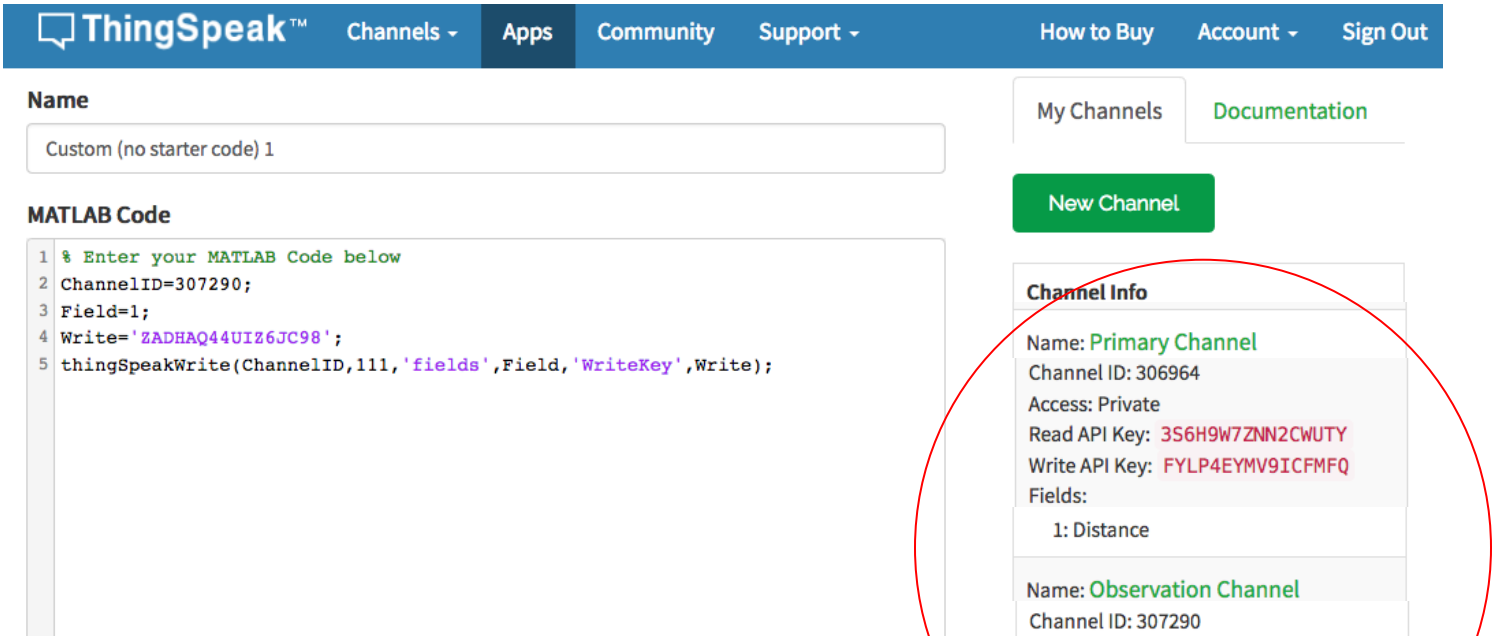
Step 11: Click "New."



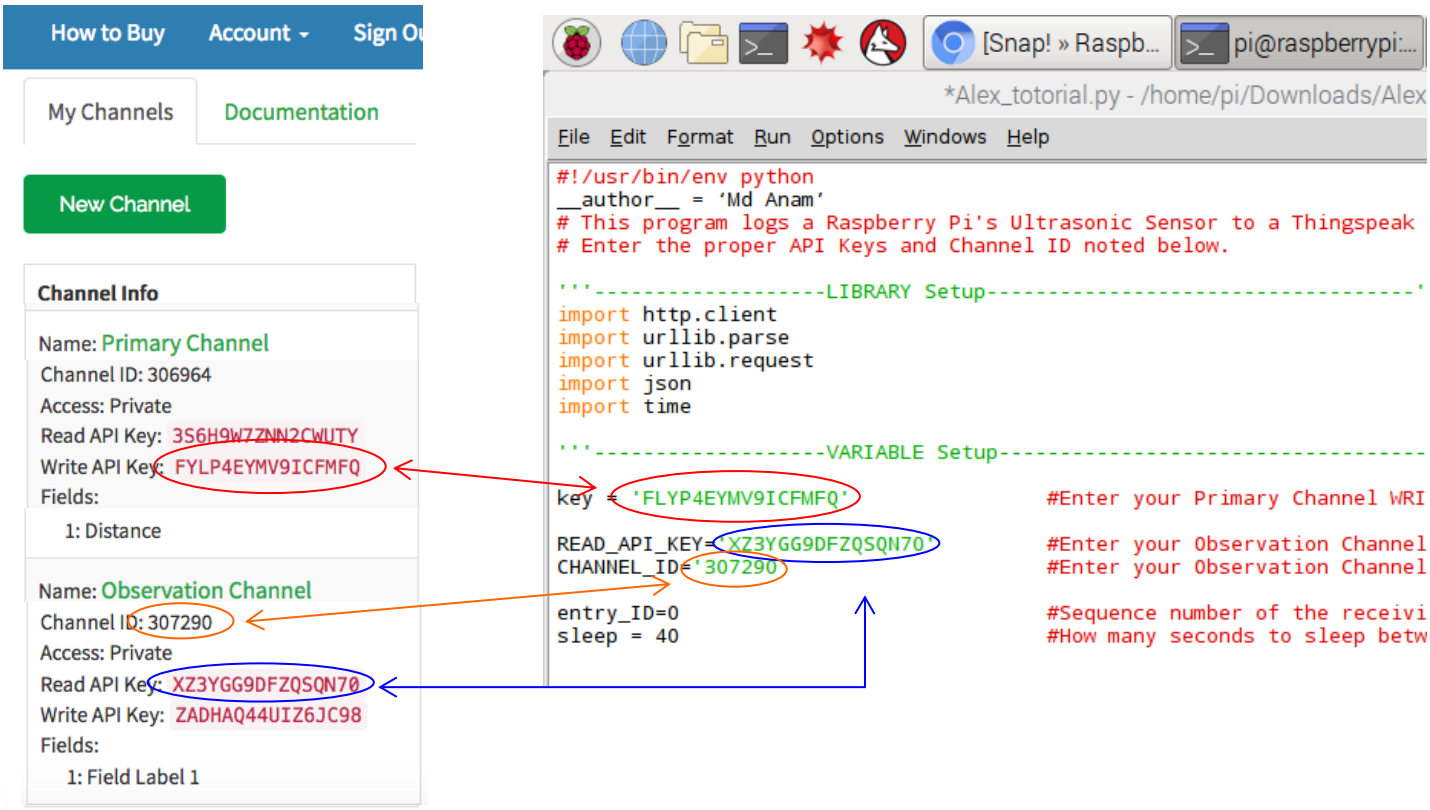
Step 12: Click "Create."



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.



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'`

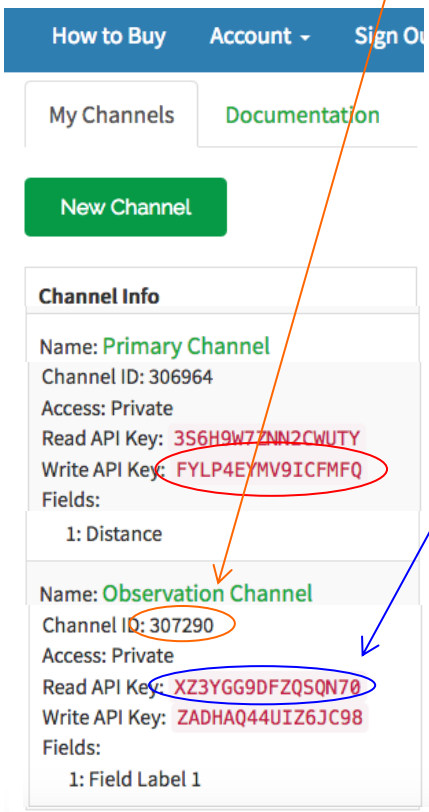


Step 15: Enter Channel ID and Read API Key into the code. *Note: The apostrophe (') after READ_API_KEY should be left there. An example of the full web address is shown below. The Channel ID and API Key are highlighted in yellow.*

`'http://api.thingspeak.com/channels/307290/feeds/last.json?api_key=XZ3YGG9DFZQSQN7Q')`

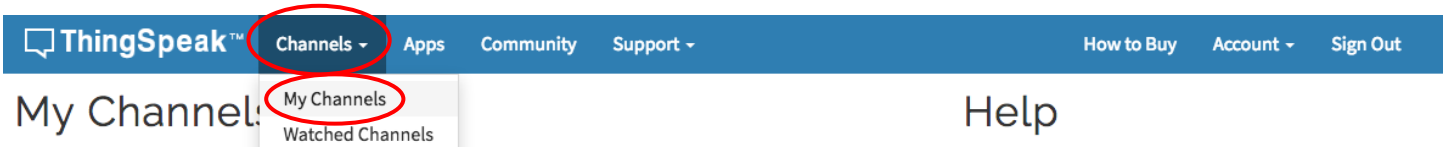
File Edit Format Run Options Windows Help

```
'''-------------------DATA sending to ThingSpeak----------------------------------'''
def sending(distance):
    while True:
        INPUT=distance
        params = urllib.parse.urlencode({'field1': INPUT, 'key':key})
        headers = {"Content-typZZe": "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)
```

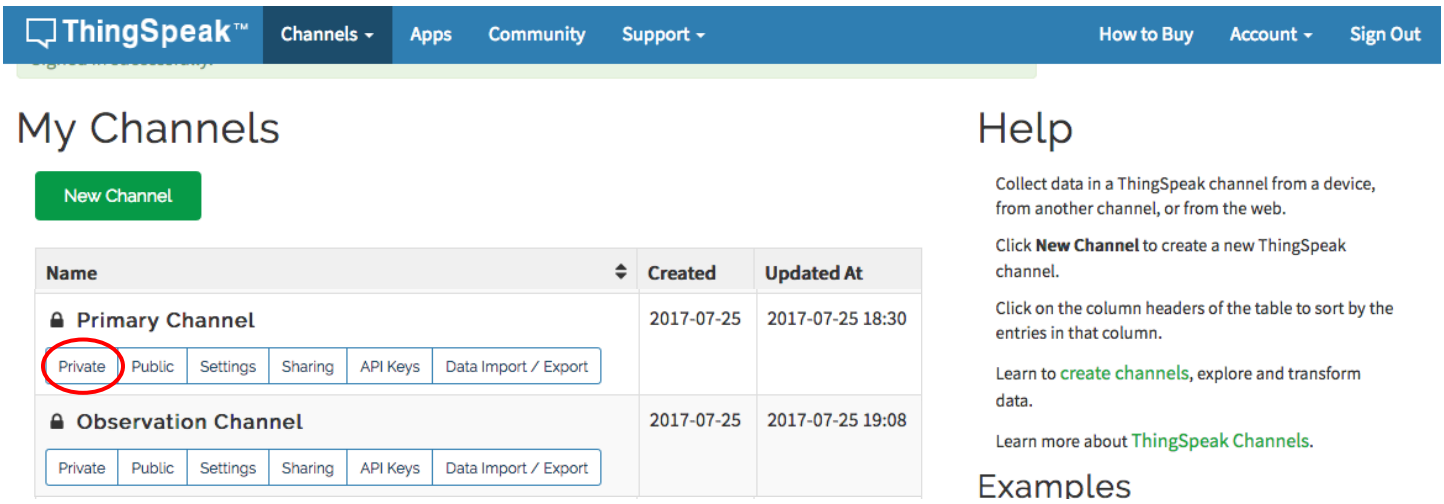


Part 3: Watching it All Work Together

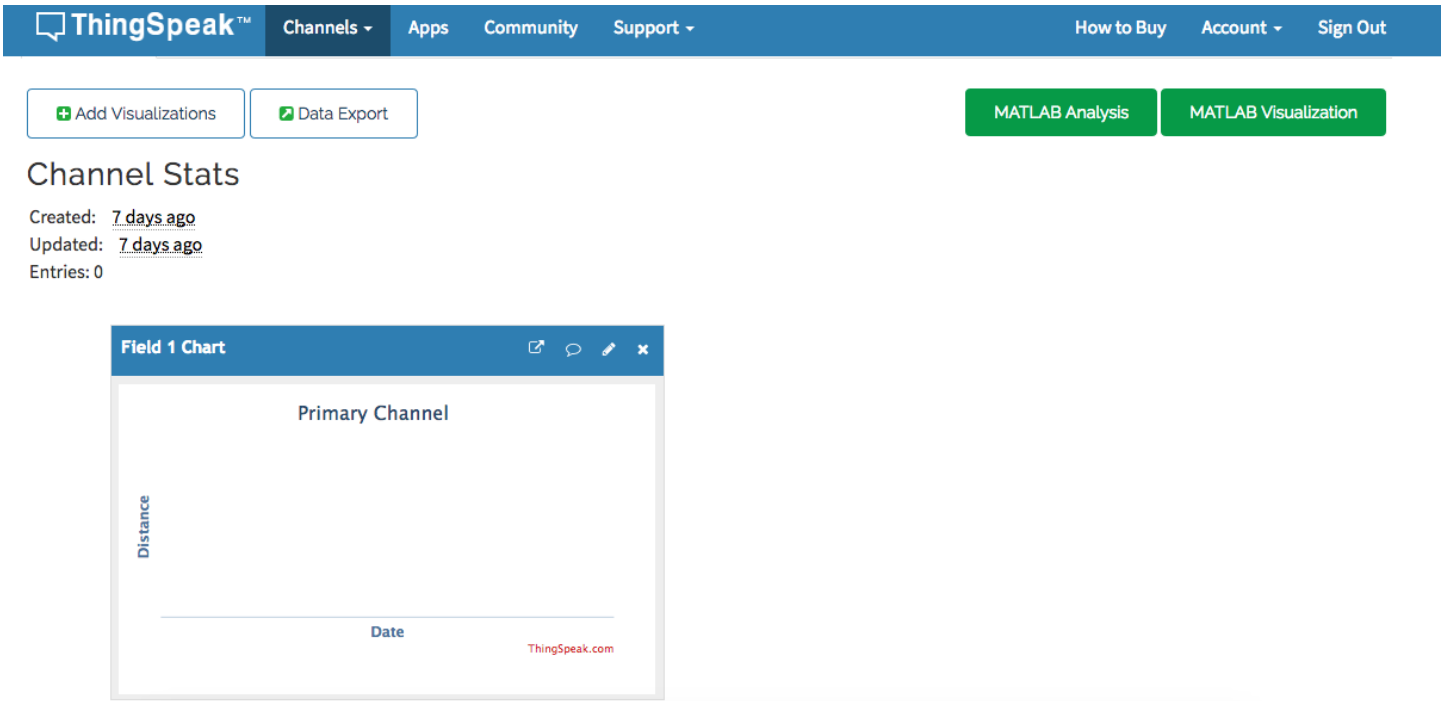
Step 1: Open ThingSpeak and click on "Channels" > "My Channels."



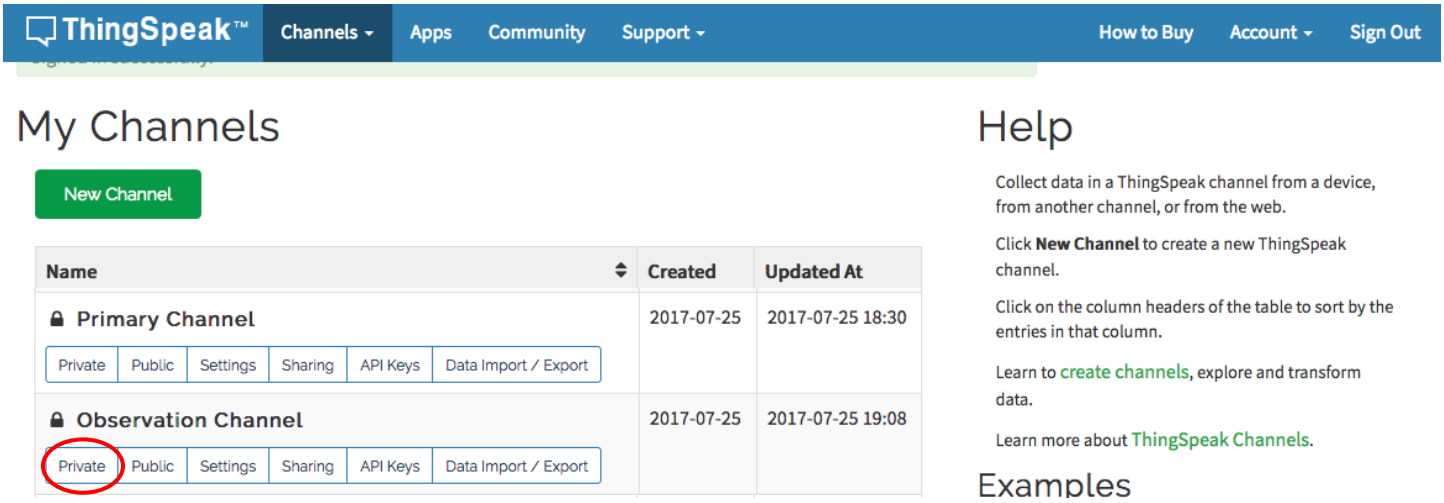
Step 2: Click "Private" under Primary Channel.



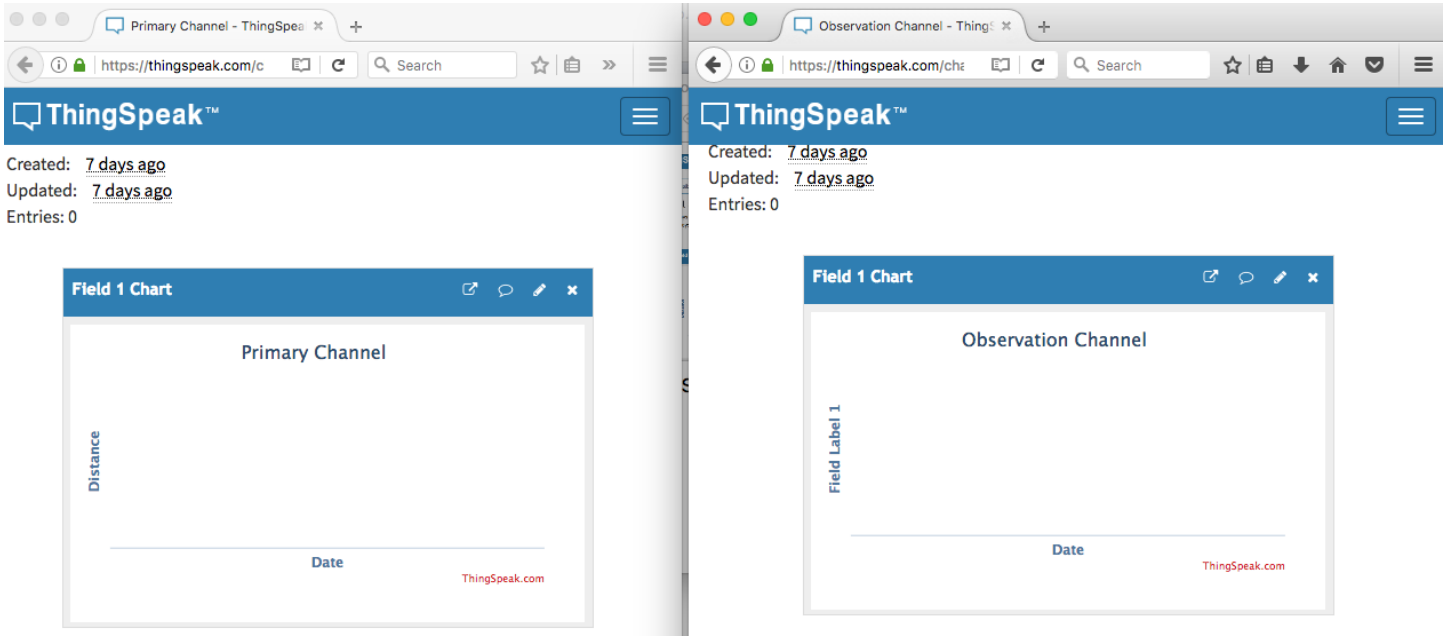
Step 3: A blank graph should appear like this:



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



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



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 = 'FYLP4EYMV9ICFMFQ'                #Enter your Primary Channel WRITE API Key

READ_API_KEY='XZ3YGG9DFZQSQN7O'         #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
```

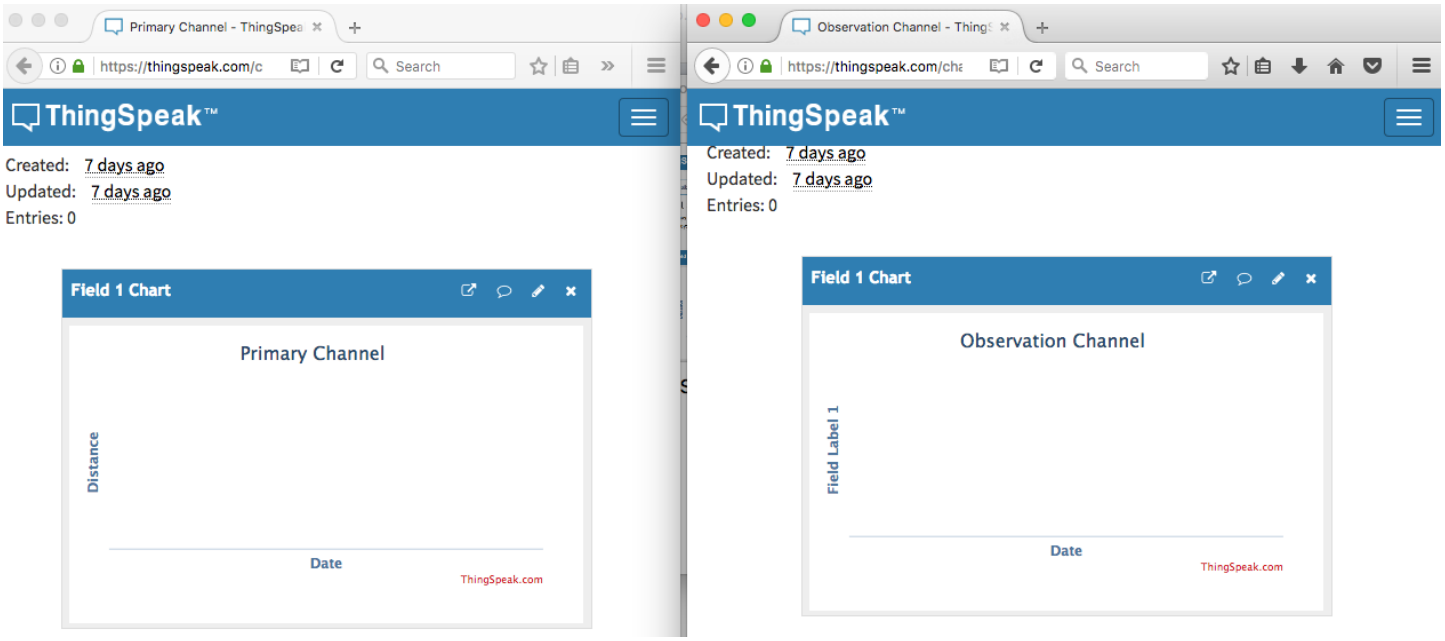
Step 8: Press F5 to run your program. A window should appear like this:

```
*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
```

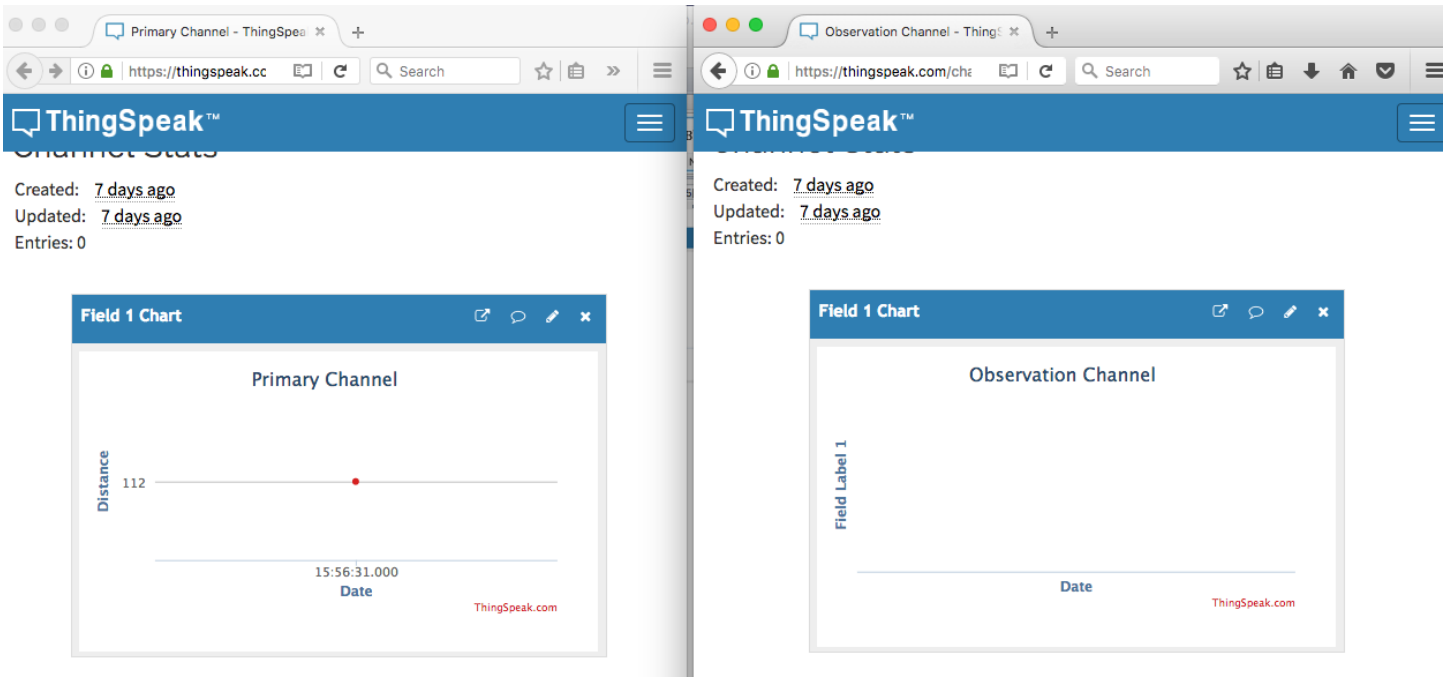
The line that reads 200 OK indicates that the data that was sent through a proper connection. Allow the code to keep running.

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……

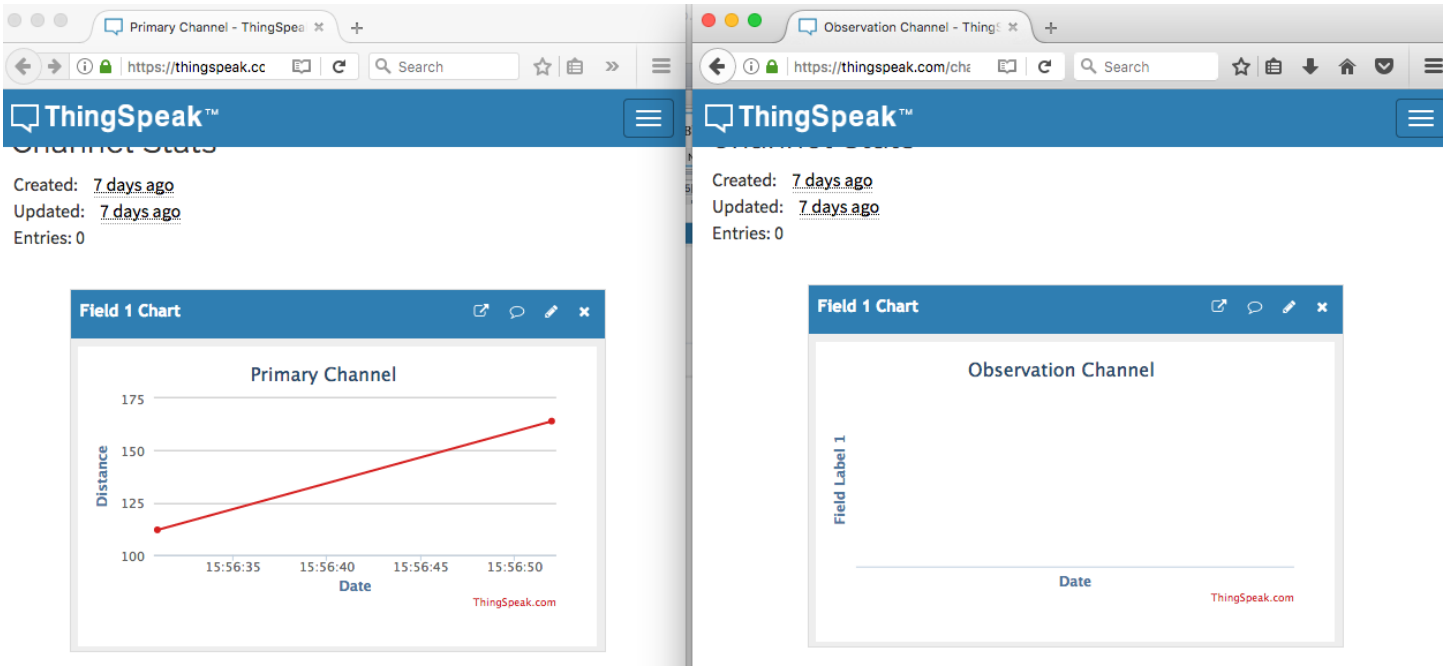


…to this



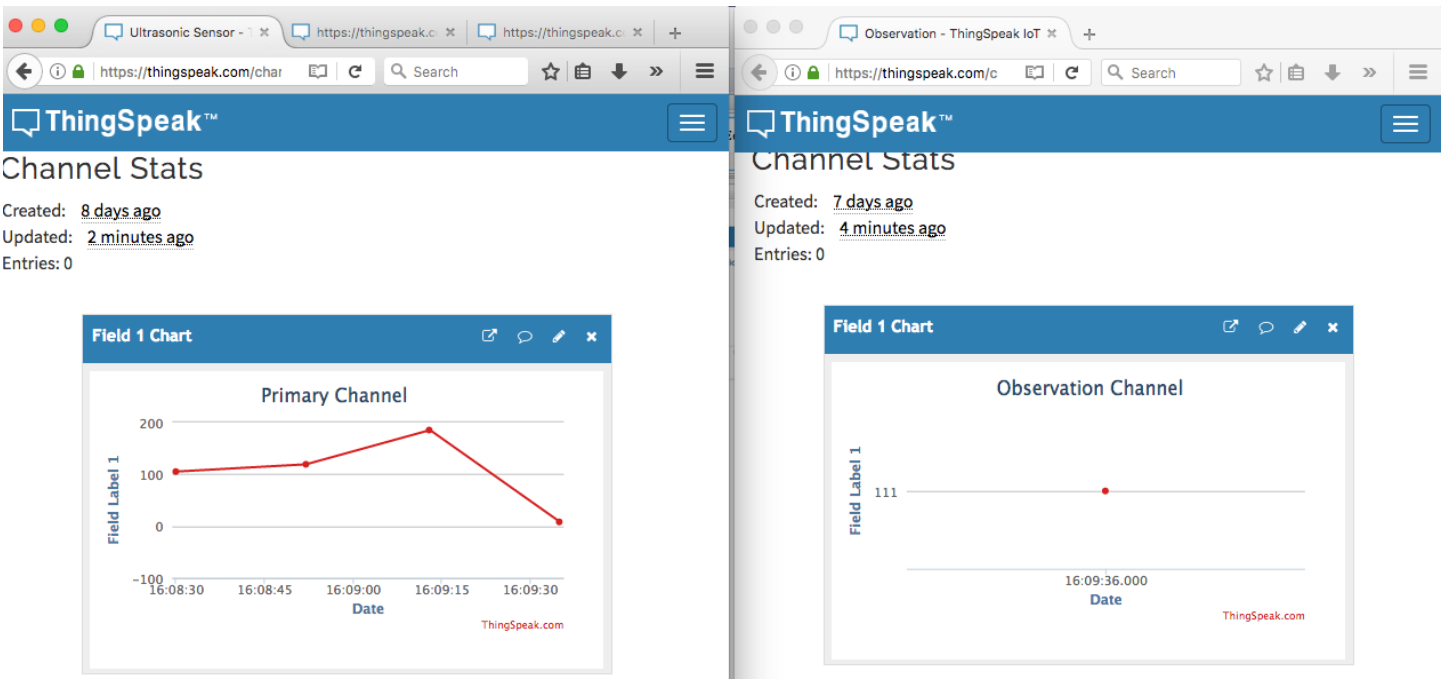
***Note: The specific values on your graph will be different than the ones shown. For example, our first data point was 112. Yours may be 100 or 63 or another value.*

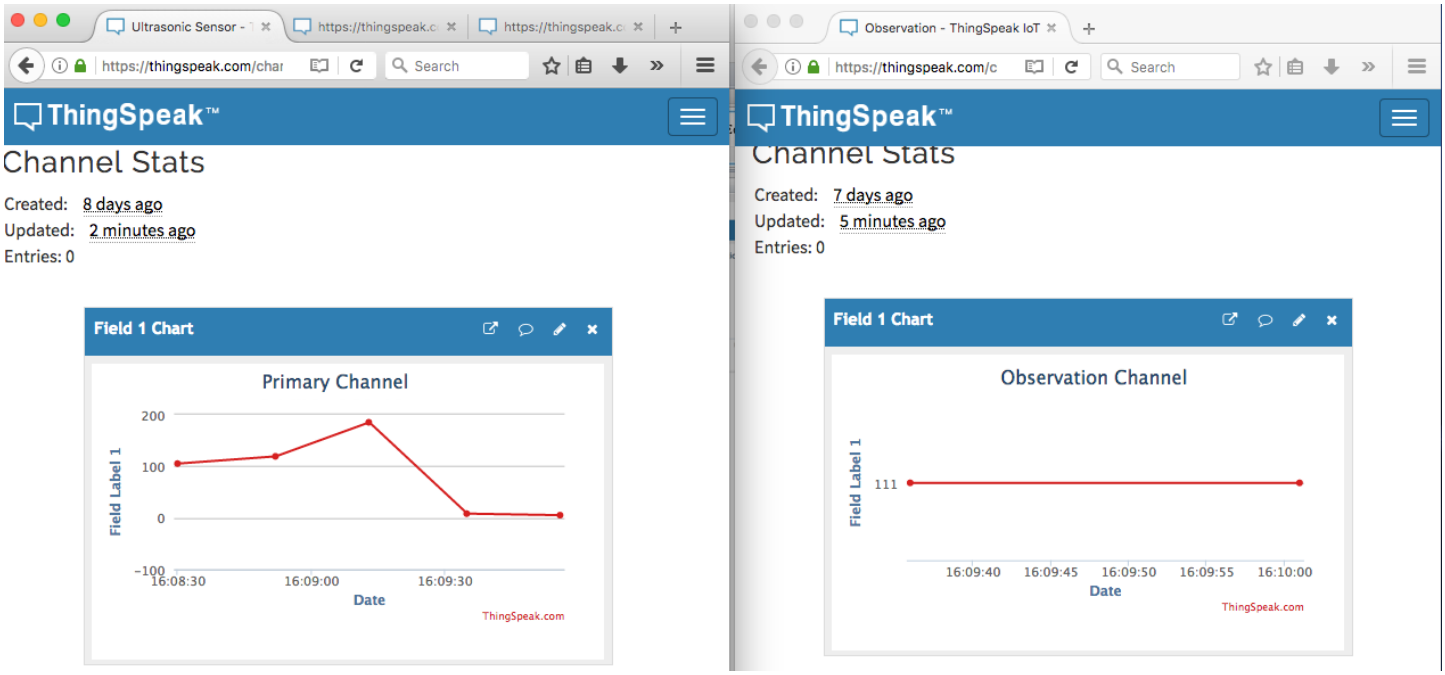
Step 11: Continue watching. The program will continue running. Another data point will be sent to ThingSpeak and look like this.



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='XZ3YGG9DFZQSQN70'         #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-typZZe": "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=XZ3YGG9DFZQSQN70')
    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()
```