International Journal of Trendy Research in Engineering and Technology
Volume 4 Issue 4 August 2020
IJTRET
ISSN NO 2582-0958

Tutorial Article

SENSOR DATA COMMUNICATION TO THNIGSPEAK IOT PLATFORM USING RASPBERRY PI

P.Sureshkumar
So-Fi Infotech, Kolathur, Chennai - 600099
cto@so-fi.in

ABSTRACT

The aim of this article is to explain the readers the technique of sending data from the sensor through the Raspberry pi and communicating it to the Thingspeak cloud which is an IOT platform. To explain the procedure a simple ultrasonic sensor HC-SR-04 which senses objects up to a distance of 13 ft connected with a raspberry Pi board is used.

This tutorial describe about Raspberry Pi board, Ultrasonic sensor, circuit design for sensing of data, Python programming script with step by step interpretation. Finally procedure for creating a channel in ThinSpeak for uploading our data and to read the uploaded data from a remote desktop/mobile is also included in this tutorial.

I Raspberry Pi

The Raspberry Pi3 Model B is a wonderful platform that can be used to build automation systems. Clearly,the Raspberry Pi3 model B board is perfect when being used as a “hub” for automation systems, connecting to other open-source hardware parts like sensors. Raspberry Pi is a small sized single board computer which is capable of doing the entire job that an average desktop computer does Like spread sheets, word processing, Internet, Program ming, Games etc.

Raspberry Pi 3 ModelB Built on the latest Broadcom 2837 ARMv8 64bit processor, the new generation Raspberry Pi4 Model B is faster and more powerful than its predecessors. With built-in wireless and Bluetooth connectivity, it becomes the ideal IoT ready solution. It consists of 1.2GHz QUAD Core Broadcom BCM2837 64bit ARMv8 processor,BCM43438 Wi-Fi on board, Bluetooth Low
Energy (BLE) on board,1GB RAM,4x USB 2 ports,40pin extended GPIO,HDMI and RCA video output. The Raspberry Pi3B model is shown in fig.1.

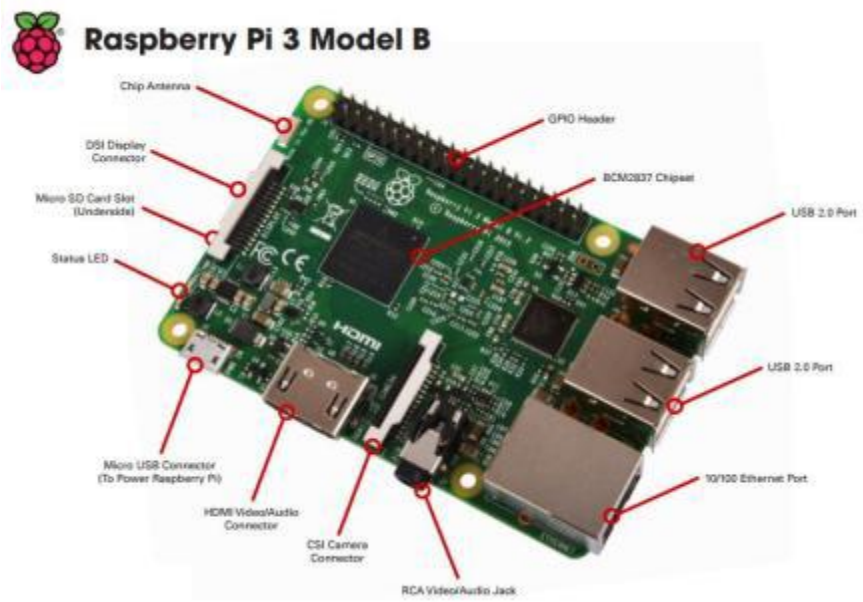

Fig. 1 Raspberry Pi 3 Model B module

Raspberry Pi3Model B runs on Linux kernel basedoperating systems. It boots and runs from the SD card. It does not have any internal memory other than the ROM. It has an SD card slot which is capable of reading up to 32 GB. The GPIO pins of the raspberry Pi3 Model B are programmed using Python programming language. The I/O devices like sensors are given to GPIO pins whenever needed.

GPIO:

One powerful feature of the Raspberry Pi 3 Model B is the row of GPIO (general purpose input/output) pins along the edge of the board.

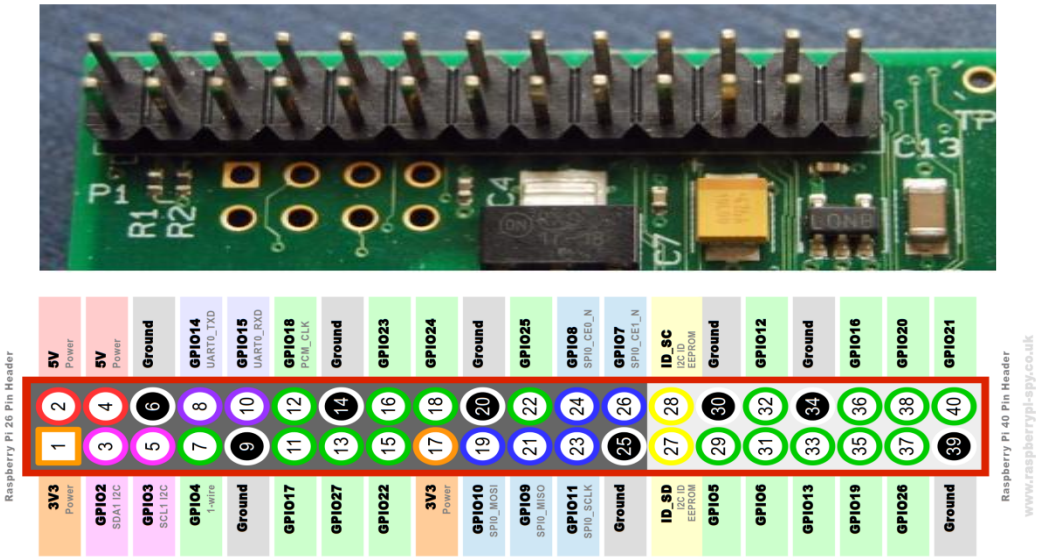

Fig.2. GPIO pins

The program can be written on the pins to interact in amazing ways with the real world. Inputs don't have to come from a physical switch; it could be from a sensor or a signal from another computer or device. For example, the output can do anything, from turning on an LED to sending a signal or data to another device. If the Raspberry Pi3B is on a network, you can control devices that are attached to it from anywhere and those devices can send data back. Connectivity and control of physical devices over the internet is a powerful and exciting thing, and the Raspberry Pi3 model B is ideal for this.

II Ultrasonic Sensor- HC-SR04

Ultrasonic distance sensors are designed to measure distance between the source and target using ultrasonic waves. HC-SR04 is a commonly used sensor for distance measurement in the range 2cm to 400cm.

Like a bat it sends out pulses of ultrasonic sound (at a frequency of 41KHz, higher than humans can hear), and listens for the reflected sound from the near by objects

It consists of an ultrasonic transmitter, receiver and control circuit. The transmitter transmits short bursts which gets reflected by target and are picked up by the receiver.

The time difference between transmission and reception of ultrasonic signals is calculated.

Using the equation '**Speed = Distance/Time**', the distance between the source and target can be easily calculated.

Time taken by pulse is actually for **to and fro** travel of ultrasonic signals, while we need only half of this. Therefore Time is taken as Time/2.

*Distance = Speed * Time/2*

Speed of sound at sea level = 343 m/s or 34300 cm/s

Thus, *Distance* = 34300 * time/2 = *17150 * Time (unit cm)*

HC-SR04 Ultrasonic Sensor Pin Configuration

This sensor includes four pins and the pin configuration of this sensor is discussed below.

- Vcc: This pin provides a +5V power supply to the sensor.
- TRIG: This is an input pin, used to initialize measurement by transmitting ultrasonic waves by keeping this pin high for 10us.
- ECHO:This is an output pin, which goes high for a specific time period and it will be
- equivalent to the duration of the time for the wave to return back to the sensor.

- GND: This is pin used to connect to the ground (GND) of the system

Fig.3. Ultrasonic Sensor HC-SR04

Working principle of HC-SR04

Ultrasonic ranging module HC - SR04 provides 2cm - 400cm non-contact measurement function, the ranging accuracy can reach to 3mm. The modules includes ultrasonic transmitters, receiver and control circuit. The basic principle of working :

1.Give trigger signal of High level to TRIG input for atleast 10us

2. The Module automatically sends eight 40 kHz of ultrasonic burst

3.If there is an obstacle in-front of the sensor, the ultrasonic waves will be reflect back

4. If the signal comes back, the ECHO output of the module will be HIGH for a duration of time taken for sending and receiving ultrasonic signals.

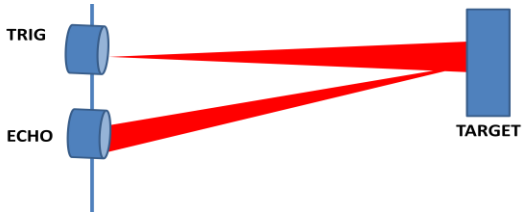

Fig.4. Working principle of HC-SR04

III Circuit Design for sensing data from sensor using Rpi

Connect the Trig Pin of the HC-SR04 Ultrasonic Sensor to the Physical Pin 16 (GPIO23) of the Raspberry Pi. The ECHO output of sensor module is 5v. The input pin of Raspberry Pi GPIO is rated at 3.3v. So 5v cannot be directly given to the unprotected 3.3v input pin. Therefore we use

a voltage divider circuit using a combination of resistors 680Ω and 1.5 KΩ as shown in the following figure to bring down the voltage to 3.3V and then connect it to physical Pin 18(GPIO24) of Raspberry pi

Finally, provide the +5V and GND connections to the Ultrasonic Sensor from the Raspberry Pi Pins.

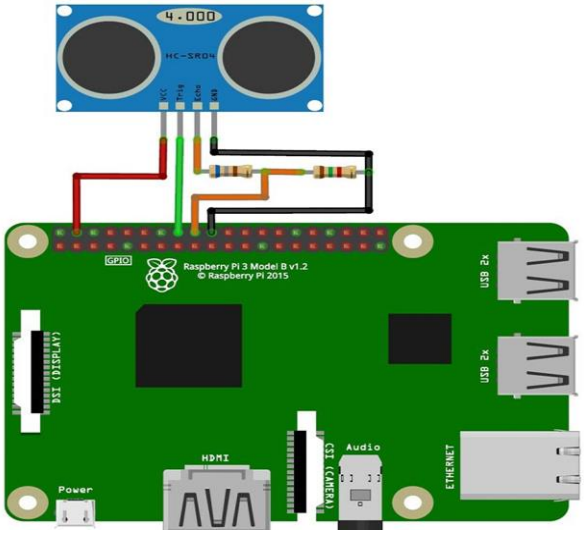

Fig.5. Circuit connecting Raspberry pi & ultrasonic sensor

Use of an ultrasonic Sensor with a Raspberry Pi allows for inexpensive monitoring of water levels in rivers, lakes, height measurement, as well as countless other applications that require distance/ level measurement.

IV Python Programming Script

Ultrasonic Sensor is connected to Raspberry Pi, and now we need to program a Python script to measure distance.

The Ultrasonic sensor output (ECHO) will always output low (0V) unless it's been triggered in which case it will output 5V (3.3V with our voltage divider!). We therefore need to set one GPIO pin as an output, to trigger the sensor, and one as an input to detect the ECHO voltage change.

Python script should first start with the following initial steps

1.Import python GPIO library

2.Import Time library

3.Set GPIO pin numbers

The detailed python programming to detect distance is given below.

`import RPi.GPIO as GPIO`	`#Import python GPIO library`
`import time`	`#Import time library to make raspberry pi to wait between steps`
`GPIO.setmode(GPIO.BCM)`	`#Set GPIO pin numbering in BCM (This is different from Physical pin numbering)`
`TRIG = 23`	`#GPIO 23 represents output pin which triggers the sensor`
`ECHO = 24`	`GPIO 24 reads the return signal from the sensor as ECHO`
`print "Distance measurement in progress"`	`To inform the user that measurement is in progress`
`GPIO.setup(TRIG,GPIO.OUT)`	`#Set pin GPIO 23 as output`
`GPIO.setup(ECHO,GPIO.IN)`	`#Set pin GPIO 24 as input`

`while True:`
`GPIO.output(TRIG, False)`
`print "Waitng For Sensor To Settle"`
`time.sleep(2)`
	`#Set TRIG pin as LOW`
	
	`#Delay of 2 seconds for the sensor to settel`

`GPIO.output(TRIG, True)`
`time.sleep(0.00001)`
`GPIO.output(TRIG, False)`
	`#Set TRIG as HIGH`
	`#Delay of 0.00001 seconds (10µS)`
	`#Set TRIG as LOW`

Now we've sent our pulse signal we need to listen to our input pin, which is connected to ECHO.

The sensor sets ECHO to high for the amount of time it takes for the pulse to go and come back,

So our code therefore needs to measure the amount of time that the ECHO pin stays high. We use the "while" string to ensure that each signal timestamp is recorded in the correct order.

The time.time() function will record the latest timestamp for a given condition. Example: if a pin goes from low to high, and we're recording the low condition using the time.time() function, the recorded timestamp will be the latest time at which that pin was low.

So our first step should be to record the last low timestamp for ECHO (pulse_start)

`while GPIO.input(ECHO)==0:`
`pulse_start = time.time()`
	`While" string is used to ensure that each signal timestamp is recorded in the correct order`
	`#Check whether the ECHO is LOW`
	`#Saves the last known time of LOW pulse`

Once a signal is received, the value changes from low (0) to high (1), and the signal will remain high for the duration of the echo pulse. We therefore also need the last high timestamp for ECHO (pulse_end).

`while GPIO.input(ECHO)==1:` `pulse_end = time.time()`	`#Check whether the ECHO is HIGH` `#Saves the last known time of HIGH pulse`
`pulse_duration = pulse_end - pulse_start`	`#Get pulse duration to a variable`
`distance = pulse_duration * 17150`	`#Multiply pulse duration by 17150 to get distance`
`distance = round(distance, 2)`	`#Round to two decimal points`
`Print"Distance:",distance,cm`	`# print Distance:--cm`

The above python script may be names as “distance_ sensor.py”
On running this python script with the following command , the distance can be measured .

```
pi@raspberrypi ~ $ sudo python distance _sensor.py
Distance Measurement In Progress
Waiting For Sensor To Settle
Distance: 12.52 cm
pi@raspberrypi ~ $
```

V . ThingSpeak

ThingSpeak is an open source Internet of Things (IoT) application and API to store and retrieve data from things-sensors using the HTTP protocol over the Internet . Thing Speak gets coordinated help from the numerical registering programming MATLAB from Math Works. It permits ThingSpeak clients to investigate and envision transferred information utilizing Matlab without requiring the buy of a Matlab permit from Math works.

Procedure for creating a Channel in ThingSpeak

1.Sign In to ThingSpeak™ using your Math Works® Account, or create a new Math Works account.

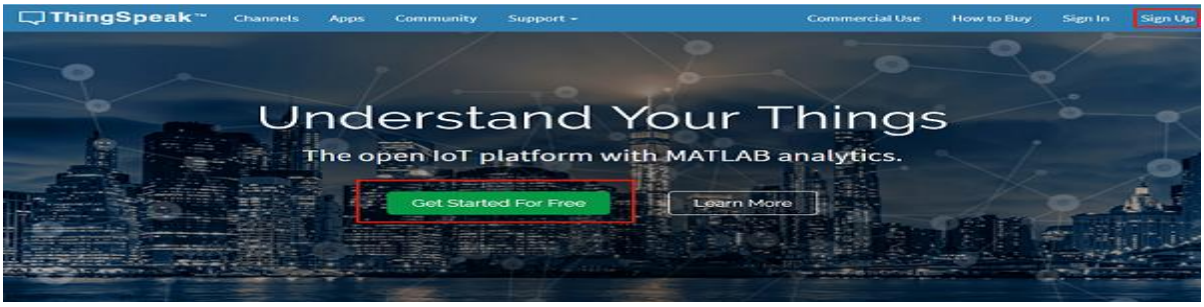

Fig.6. Opening menu for creating new Thingspeak /Math Work account

After creating account your new channel is to be created for your project as given below.

2. Click Channels>My Channels.

- Click on “MY Channels” and click on **New Channel** for your new project.

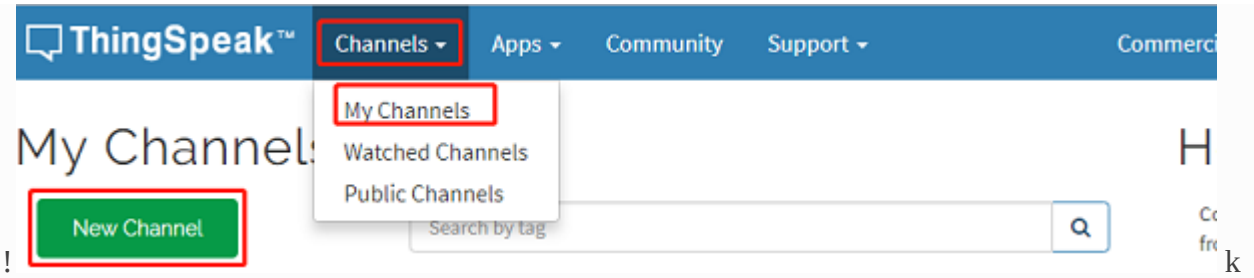

Fig.7a.ThinSpeak screen shot showing New Channel creation process.

3.Project parameters are to be entered in the list below. If there are more parameters, you need to choose more fields

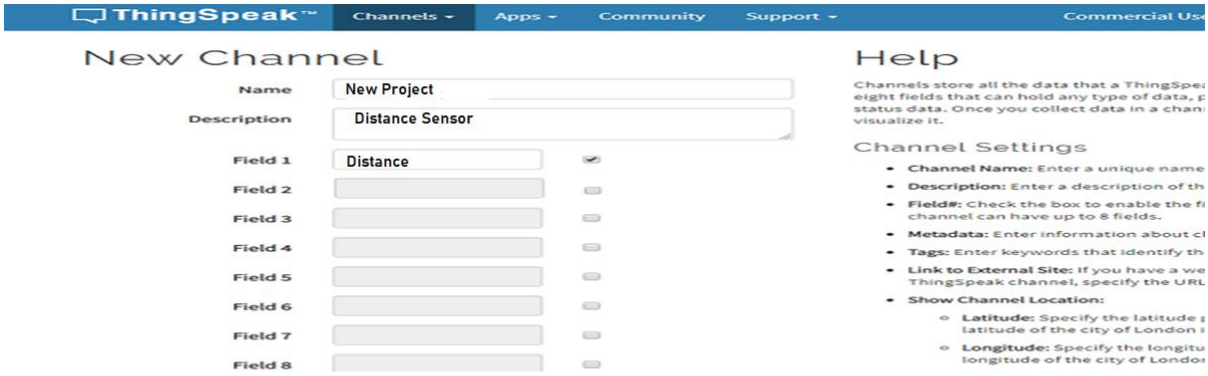

Fig7.b.ThinSpeak screen shot showing New Channel creation process.

4. Check the boxes next to Fields 1–3. Enter these channel setting values:

Name: new project

Field 1: distance in m

Field 2:

Click Save Channel at the bottom of the settings.

5. You now see these tabs:

6. Private View: This tab displays information about your channel that only you can see.

7. Public View: If you choose to make your channel publicly available, use this tab to display selected fields and channel visualizations.

8. Channel Settings: This tab shows all the channel options you set at creation. You can edit, clear, or delete the channel from this tab.

9. Sharing: This tab shows channel sharing options. You can set a channel as private, shared with everyone (public), or shared with specific users.

10. API Keys: This tab displays your channel API keys. Use the keys to read from and write to your channel.

11. Data Import/Export: This tab enables you to import and export channel data.

Next Steps Your channel is available for future use by clicking Channels>My Channels.

You can find the data of what we have uploaded here:

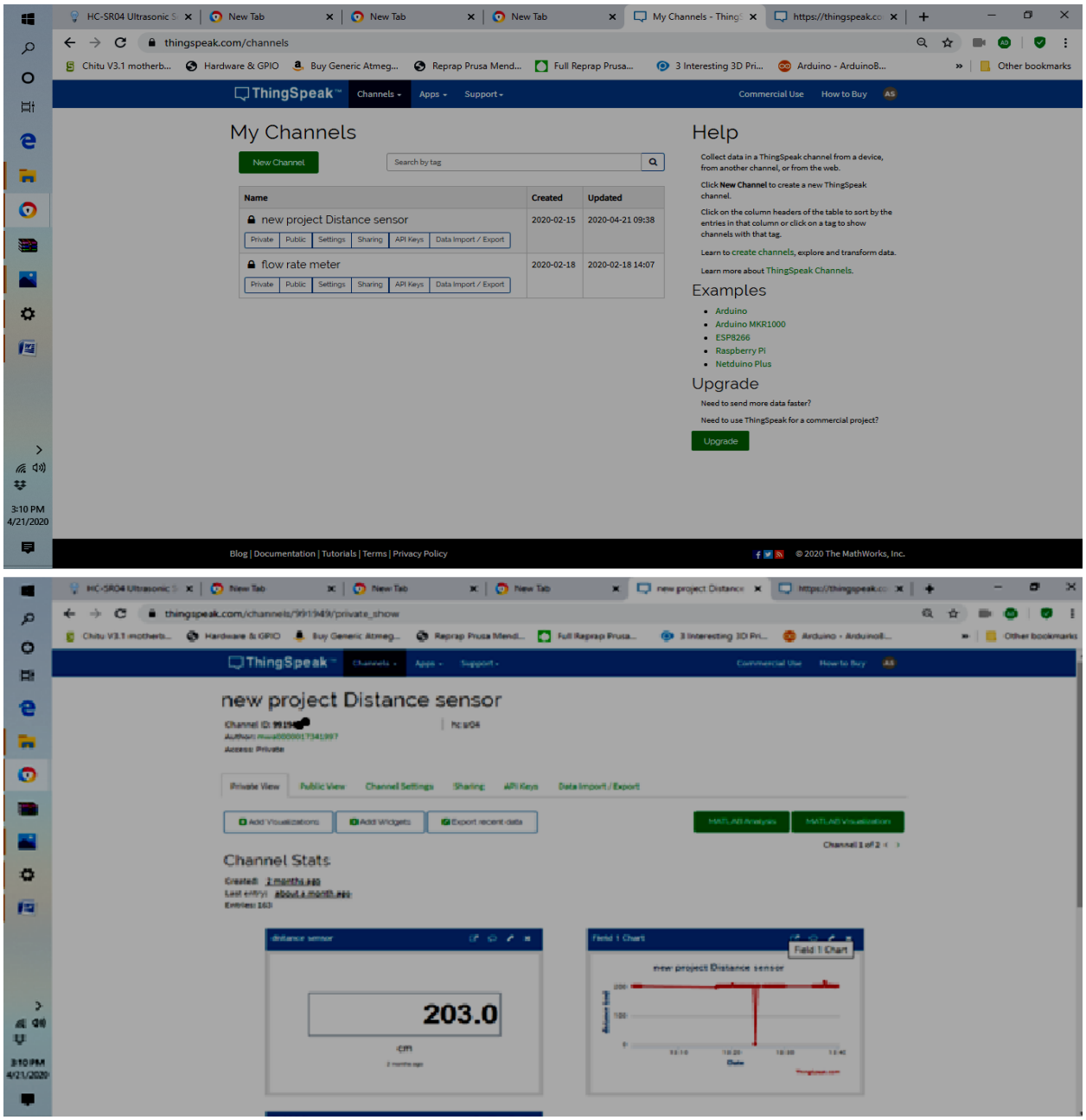

Fig.8. ThinSpeak screen shots showing uploaded data

"Write API key"is to be entered in our python script for uploading the data in the cloud whereas the "**Read APi key" /"Chennel id")**is to be entered to read data from the remote device.

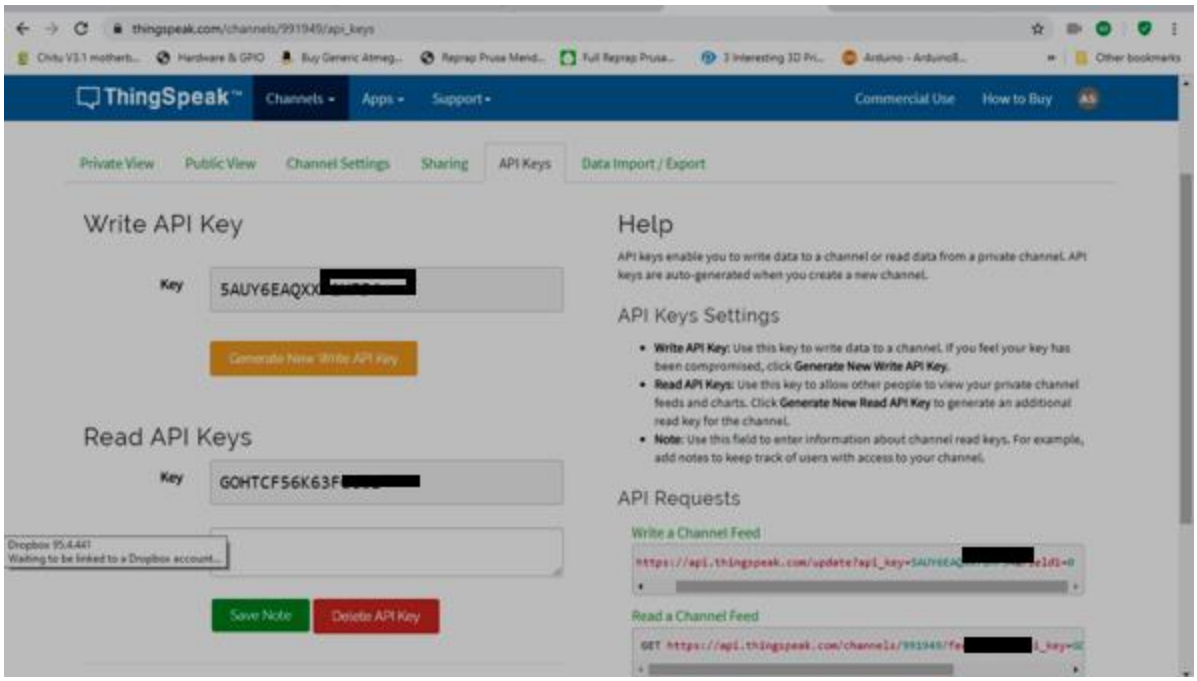

Fig.9. Thingspeak screen shot- Showing **Write API key** and **Read APi key**

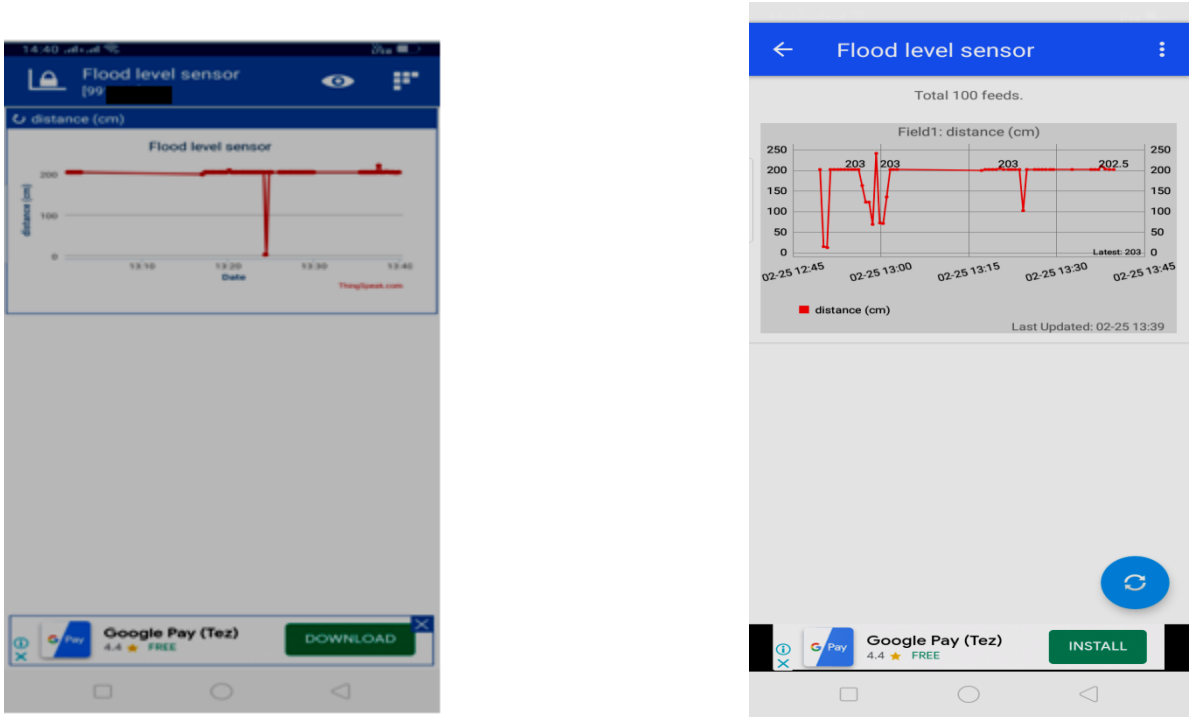

Fig10. a)Mobile app-ThingShow senor data b) Mobile app Things chart showing same data when channel Id is entered.

The simplest way to use urllib.request in python programming script is as follows:

```
from urllib.request import urlopen
```

Channel address and locations are updated using the following python code

```
baseURL= 'https://api.thingspeak.com/update?api_key=%s' % "5AU[redacted]"
f=urlopen(baseURL + "&field1=%d"%(distance))
print(f.read())
f.close()
```

VI Conclusion

This project has been with a simple ultrasonic sensor and Raspberri pi and it was found to be an easy way to communicate with the cloud with some storage and visualization facilities. In this project Raw data loaded in the cloud can be visualized in graphical format with in a very short span of time at a remote desk/ mobile app anywhere in the world.

The project can be done cheaply and also scaled up to a larger implementation. In the free version of ThingSpeak we can only see two channels at a time . Since for visualization of more channels one should go for paid options, we can try other products like AWS IoT(Amazon), GCloud IoT core(Google), Microsoft Azure IoT Suite which allow storage Database, processing alerts machine learning and data visualization.

List of URLs for Reference

https://www.thingiverse.com/tag:viewer

https://thingspeak.com/

https://apkpure.com/thingshow-thingspeak-visualizer/com.devinterestdev.thingshow

https://docs.python.org/3/howto/urllib2.html

https://www.elprocus.com/hc-sr04-ultrasonic-sensor-working-and-its-applications/

https://tutorials-raspberrypi.com/raspberry-pi-ultrasonic-sensor-hc-sr04/