IOT

ASSIGNMENT 3

Raspberry Pi: Hall effect sensor integration and cloud uploading report

INTRODUCTION

Raspberry Pi is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation in association with Broadcom. The Raspberry Pi project originally leaned towards the promotion of teaching basic computer science in schools and in developing countries.

ThingSpeak is an open IoT platform for monitoring your data online. In the ThingSpeak channel you can set the data as private or public according to your choice. ThingSpeak takes a minimum of 15 seconds to update your readings. It's a great and very easy to use platform for building IOT projects.It is an IoT analytics platform service that allows you to aggregate, visualize, and analyze live data streams in the cloud. You can send data to ThingSpeak from your devices, create instant visualization of live data, and send alerts.

Hall sensors are sensors which produce an electrical signal at its output when it comes in contact with a magnetic field. The analog value of the electric signal at the output of the sensor is a function of the strength of the magnetic field.

COMPONENTS REQUIRED

1. Raspberry Pi
2. Power Cable
3. WiFi or Internet
4. Hall effect sensor

STEPS FOR BUILDING RASPBERRY PI CLOUD

Step 1: Signup for ThingSpeak

For creating your channel on ThingSpeak you first need to sign up on ThingSpeak. In case if you already have an account on ThingSpeak just sign in using your id and password.

Step 2: Create a Channel for Your Data

Once you Sign in after your account verification, Create a new channel by clicking “New Channel” button

Step 3: Getting API Key in ThingSpeak

To send data to ThingSpeak, we need an unique API key, which we will use later in our python code to upload our CPU data to the ThingSpeak Website.

THINGS SPEAK CHANNEL ID: 1965760

A **channel** is **where we send our data to store**. Each channel includes 8 fields for any type of data, 3 location fields, and 1 status field. Once we have a ThingSpeak Channel we publish data to the channel, have ThingSpeak process the data, then have our application retrieve the data.

The 16-digit **API key** allows you to read from a private channel and write to a channel

PROTOCOLS

GET:

GET is used to request data from a specified resource

- GET requests can be cached
- GET requests remain in the browser history
- GET requests can be bookmarked
- GET requests should never be used when dealing with sensitive data
- GET requests have length restrictions
- GET requests are only used to request data (not modify)

POST:

POST is used to send data to a server to create/update a resource.

- POST requests are never cached
- POST requests do not remain in the browser history
- POST requests cannot be bookmarked
- POST requests have no restrictions on data length

CODE

Import required libraries

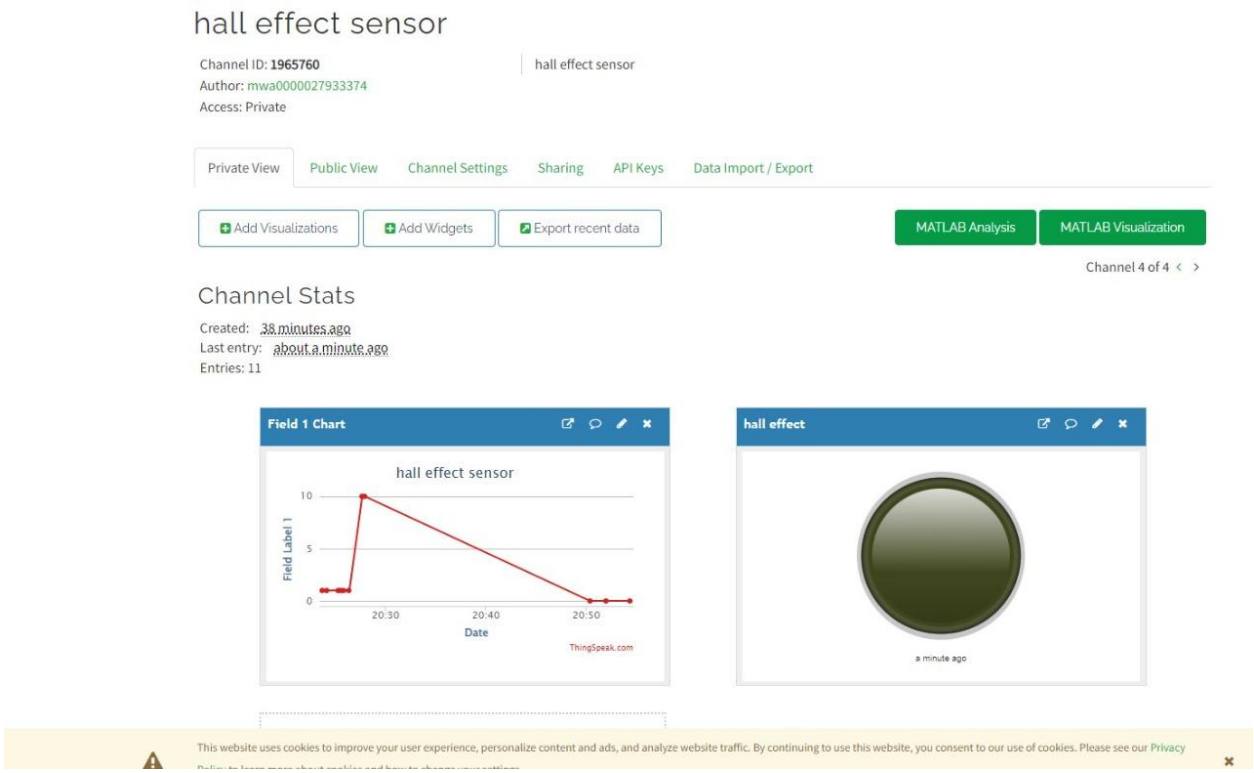
```
import time

import datetime

import RPi.GPIO as GPIO

import http.client

import urllib

key = "6OCD8S2JJ9K2J433"# Import Raspberry Pi GPIO library


def sensorCallback(channel):

  # Called if sensor output changes

  timestamp = time.time()

  stamp =
datetime.datetime.fromtimestamp(timestamp).strftime('%H:%M:
%S')

  if GPIO.input(channel):

      # No magnet

      print("Sensor HIGH " + stamp)

      params = urllib.parse.urlencode({'field1': 0,'key':key})

      headers = {"Content-typZZe":
"application/x-www-form-urlencoded","Accept": "text/plain"}
```

```
        conn =
http.client.HTTPConnection("api.thingspeak.com:80")
        conn.request("POST", "/update", params, headers)
        response = conn.getresponse()
        data = response.read()
        conn.close()
  else:
        # Magnet
        print("Sensor LOW " + stamp)
        params = urllib.parse.urlencode({'field1': 1,'key':key})
        headers = {"Content-typZZe":
"application/x-www-form-urlencoded","Accept": "text/plain"}
        conn =
http.client.HTTPConnection("api.thingspeak.com:80")
        conn.request("POST", "/update", params, headers)
        response = conn.getresponse()
        data = response.read()
        conn.close()
```

```
def main():
  # Wrap main content in a try block so we can
  # catch the user pressing CTRL-C and run the
  # GPIO cleanup function. This will also prevent
  # the user seeing lots of unnecessary error
  # messages.

  # Get initial reading
  sensorCallback(17)

  try:
        # Loop until users quits with CTRL-C
        while True :
        time.sleep(0.1)

  except KeyboardInterrupt:
        # Reset GPIO settings
```

```
        GPIO.cleanup()


# Tell GPIO library to use GPIO references

GPIO.setmode(GPIO.BCM)


print("Setup GPIO pin as input on GPIO17")


# Set Switch GPIO as input

# Pull high by default

GPIO.setup(17 , GPIO.IN, pull_up_down=GPIO.PUD_UP)

GPIO.add_event_detect(17, GPIO.BOTH,
callback=sensorCallback, bouncetime=200)


if __name__=="__main__":

   main()
```

OUTPUT


hall effect sensor
Channel ID: 1965760
Author: mwa0000027933374
Access: Private
hall effect sensor
Private View
Public View
Channel Settings
Sharing
API Keys
Data Import / Export
Add Visualizations
Add Widgets
Export recent data
MATLAB Analysis
MATLAB Visualization
Channel 4 of 4
Channel Stats
Created: 38 minutes ago
Last entry: about a minute ago
Entries: 11
Field 1 Chart
hall effect sensor
Field Label 1
10
5
0
20:30
20:40
20:50
Date
ThingSpeak.com
hall effect
a minute ago
This website uses cookies to improve your user experience, personalize content and ads, and analyze website traffic. By continuing to use this website, you consent to our use of cookies. Please see our Privacy