Design And Implementation Motorcycle Security System Using Internet Based On Raspberry-Pi

Muhammad Aditya Taufik
S1 Telecommunication Engineering
Telkom University
Bandung, Indonesia
adityataufik16@gmail.com

Leanna Vidya Yovita, M.T
Faculty of Electrical Engineering
Telkom University
Bandung, Indonesia
Leanna.vidya@gmail.com

Ramdhan Nugraha, M.T
Faculty of Electrical Engineering
Telkom University
Bandung, Indonesia
ramdhan1305@gmail.com

***Abstract*— Security system on motorcycle today are still very less attention. At the same time it needs a Security system for vehicles, especially motorcycles. By using raspberry-pi would be made a security system for a motorcycle that is capable of controlling a motorcycle with the Internet as the interface. With this system the user can controlled the system. Communicate with system with two-way communication using a 'telegram' chatting application, and tracking location using geolocation technology.**

Keywords—Raspberry-Pi, Internet, Telegram, Geolocation

I. INTRODUCTION

Indonesia, which is one of the most densely populated countries in the world was to have a very high crime rate. Particularly the number of motorcycle theft. With motorcycle users are more and more, plus the people who are still less sensitive to the importance of motorcycle safety, the perpetrators curanmor (motor vehicle theft) more freely perform the action. In addition to the lack of awareness among owners of motorcycles, many other factors that cause high levels of theft of a motorcycle. One of them is the security system or security system on the motorcycle itself. With rapid technological advances, there are still a few people who try to improve the security system on a motorcycle. Even factories motorcycle manufacturer still subordinated security system that in fact it is one of the most important systems in motor vehicles.

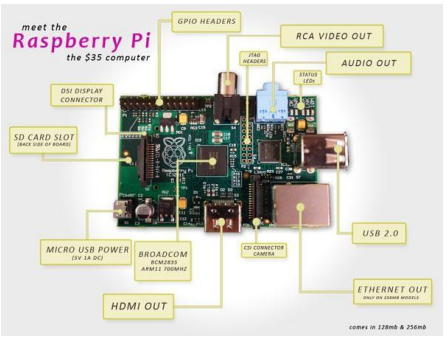
Motorcycle security systems existing today is in the form of a regular alarm system. Such as alarm system with shock sensor, which will detect the vibration on the motorcycle when it gets stolen. However, these systems often detect vibrations were not for the motorcycle to be stolen but because tersenggol or shifted. And the accident will cause the siren sounded. And the system is still considered less effective. Or a remote alarm system which uses RF (Radio Frequency). RF alarm system is able to lock and unlock system on the machine for a considerable distance still within range of the RF system. When the owner is beyond the range of the RF system can not work even then.

Therefore, it takes a motorcycle security system that can not only be controlled by their owners, but also able to communicate with their owners even though from a great distance. With such a system, the owner can control the vehicle manually or automatically from a great distance, whether it sends a command such as turning off the engine or receiving a report about the location of the motorcycle, and the owner will be free to control and monitor the vehicle anywhere and anytime.

II. BASIC THEORY

A. *Raspberry-Pi*

Raspberry Pi[1] is a small-sized computer of a credit card. Raspberry Pi is not different functions of the computer in general. There is a section for inserting a USB keyboard and mouse. There are two models of the raspberry pi, models A and B models. Some of the differences between the two lies in the size of memory (256 MB to 512 MB the model A and model B) as well as the availability of network adapters that exist only in model B. Arsitktur Raspberry Pi (model B) can be seen in the picture below.



Gambar 2.1 Raspberry-Pi Architecture

By setting a certain overclock mode, raspberry-pi can run the processor at speeds above 700MHz with this kofigurasi. In the turbo mode, the processor-pi raspberr able to work up to 1000MHz.

Here is a table that shows some overclocking mode on raspberry pi.

Overclocking Mode	ARM (MHz)	Core (MHz)	SDRAM (MHz)	Overvolt
None	700	250	400	0
Modest	800	250	400	0
Medium	900	250	450	2
High	950	250	450	6
Turbo	1000	500	600	6
Pi2	1000	500	500	2

Tabel 2.1 Overclocking Mode

B. *Telegram*

Telegram[2] is a chat application for mobile phones and desktop cloud-based degan focus on safety and speed. This chat application such as a combination of SMS and E-Mail.Aplikasi was very quick, simple and free. Telegrams can also be used in a variety of different devices at the same time as samrtphone with different OS (Android, IOS, Windows), tablet and computer.

This application can send different types of files such as photos, videos and files in any format with a capacity of up to 1GB. And the application of user telegram can form groups of up to 200 members.

The advantages offered by the chat application telegram is a telegram Very Fast, telegram-based cloud (cloud computing services) and highly encrypted. As a result, Telegram become one of the fastest chat application. Telegram Highly Secure, still associated cloud, data bases there makes Telegram safer. Messages sent will not be recorded or stored entirely on the server, but in the cloud. Telegram Easily Accessible from Various devices, because of the basic cloud, telegram allows the user to access messages from a variety of devices (including PC) and share photos, videos and documents indefinitely. Various Can Telegram File Size By GigaByte, this application allows users to share files and video to the size Giga Byte. Telegram Free Ads, developer telegram guaranteeing "100% free and no ads". Number of Group Members Telegram Very Large, telegram allows its user to create a group with up to 200 people. Open Source nature, with its open source telegram can be developed by anyone who wants to develop it.

C. *Geolocation*

Geolocation[3] is the process of determining the location. Although the concept of geolocation often associated with GPS, there are actually more than one way to determine the location while being connected to the internet. For example, your Internet address (IP address) can be used to define the outline of where you are, even without having to use sophisticated methods.

For example, if someone wants more details on the determination of your location than the location determination based on IP addresses. For example, a website may want to know your location in order to give local site maps, search for hotels or restaurants nearby.

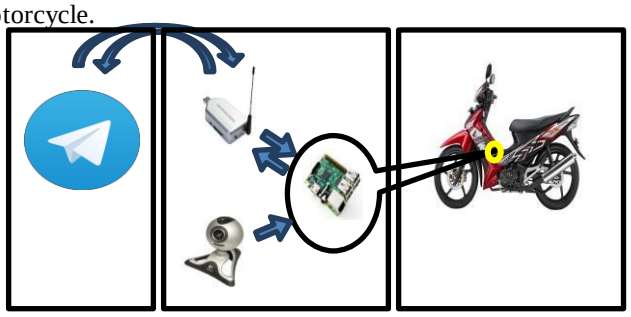
Although the Geolocation specification is quite prescriptive in defining the public API that each browser should give to the developer, it has nothing to say about how Geolocation should be implemented in a browser. General rule of thumb is that each browser must try to balance accuracy with consideration of power consumption, with the aim of providing applications with a set of coordinates accurate enough. In the scenario of a desktop or laptop, most browsers use the same mechanism to provide geolocation data, even though the services they rely background may be different. In the case of Internet Explorer 9 and thereafter, the browser will collect IP addresses or a list of nearby Wi-Fi hotspots and then pass the information to the Microsoft Location Service (the same service built-in facility location on your Windows Phone) to a set of coordinates. Google Chrome and Mozilla Firefox both are also using IP and Wi-Fi hotspot data, but relies on the Google Location Services to the actual demand. Service location method to find the user's location to sacrifice accuracy for speed (and low power consumption), and very useful as a backup, or if the device does not have built-in GPS chip.

For mobile browsers, the situation is slightly more complex, and again to the browser is concerned. Because most modern smartphones have GPS chip, the browser is free to use GPS location services provided by the mobile OS in an attempt to obtain more accurate coordinates. In the case of Internet Explorer 9 on Windows Phone 7.5 and Safari on iOS, that's done by the browser. The result is more accurate location at the expense of additional power consumption, which is usually a trade-off is acceptable in most mobile geolocation scenarios, such as turn-by-turn navigation. It's important to note, however, that the browser is not guaranteed to mobile devices using GPS-based geolocation available only as a chip. Due to the implementation of the specification, up to browser, browser or mobile services underlying OS may choose to fall back to the cell tower triangulation or IP lookup, or use a combination of cell triangulation and GPS. (Mobile OS has a lot to do this for the original application as part of the built-in to their location services.)

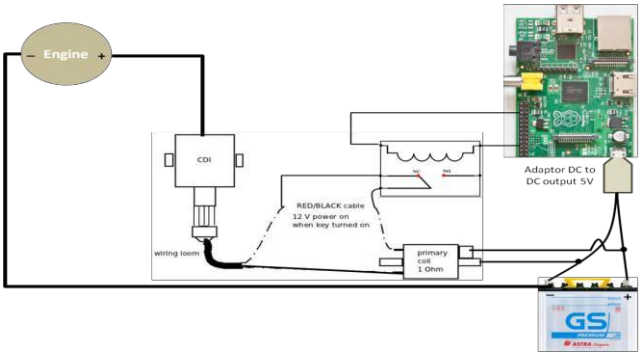
For Web application developers, this is great news, because it means that with geolocation, users can build cross-platform mobile applications which can count on the original location of the facility to the device, such as native-application developers do.

III. SYSTEM MODEL DESIGN

Here is a picture of a model of security systems for motorcycle.



Gambar 3.1 System Model Design



Gambar 3.2 Raspberry-pi installation on motorcycle

Figure 3.1 is a model of the overall system, which consists of a chat application useryang telegram as well as will provide input in the form of commands that will be sent to the raspberry-pi via the Internet. Telegram applications can be run from a smartphone or computer. GSM modem as a liaison between the user and the system at the same time supporting tools geolocation, webcam module as an additional tool supporting security systems, raspberry-pi as a data processor, and a motorcycle as an object that receives the output of the data processed by the raspberry-pi.

In the model of this system occurs one-way communication and two-way between a wire and a GSM modem applications. One-way communication that occurs in this system is when the user through a telegram sent a message in the form of a command (eg a command to turn off the machine), the GSM modem will receive the message which will then be translated by a raspberry-pi and then it will turn off the engine. And two-way communication that occurs in this system is when an application telegram sends a message to request photos from a webcam mounted on a motorcycle, then the message will be sent and received by the GSM modem is then translated by the raspberry-pi that the user requested image from the webcam installed on a motorcycle. Messages that have been translated by raspberry-pi will enable the module so that the module webcam webcam will get a picture of the thief motors or photographs of locations around the motorcycle. Then the photo will immediately be sent back by the GSM modem to the application user telegram. Then the two-way communication also occurs when a user requests location, and then raspberry-pi translate it with the help of GSM modem as a determinant of geolocation and after getting the coordinates of the data will be transmitted by the raspberry pi back to the user.

Figure 3.2 is a model-system installation raspberry pi on a motorcycle engine. Raspberry-pi will get power supply from accumulator motorcycles voltage minimized first using DC adapter. And there will be additional relay between switches and motorcycle engines. Then raspberry-pi will be connected directly to the relay that serves to disconnect the power supply so that there is no power to the engine to the motorcycle. When raspberry-pi get the command to turn off the motorcycle engine power will be cut off and the bike will die.

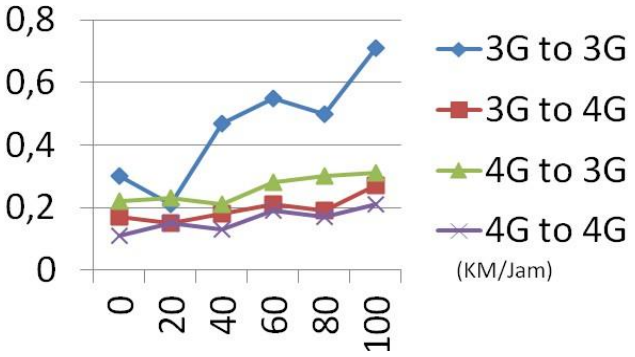
A. Design Manufacture Steps

In the process of making this system needed some special measures so that the system can be in accordance with the need to build a system to secure mobile which can move along with the vehicle. Such steps include:

1. Installing the USB modem on raspberry-pi[4]. This is needed because the system is required to be built mobile Internet connection that is always connected to the raspberry-pi. Seeing this need, the USB modem is the right choice to provide the mobile internet connection.
2. Installation Telegram chat applications[5], because in the system that will be built will use a chat application telegram as a liaison between system and user devices. Installation telegram by way clone source that has been provided in github.com. Then after telegram installed do some steps to make the raspberry-pi telegram can respond to commands automatically use control.lua file.
3. Enable pin GPIO[5] to provide output in response to a command that will be received by telegram on raspberry-pi. In some specific commands will make GPIO provide output voltage of 3.3V which will activate a relay connected in parallel and then decide the current flowing between the vehicle battery and the vehicle engine. [5]
4. Installing the USB webcam on raspberry-pi. This step is only as an additional step that complements the system with something that can display a visual field conditions.
5. This step is a step that is quite important. Ie activate geolocation in raspberry-pi[6]. That needs to be done at this stage is that first we need to get the API KEY from developer google maps as a support to obtain geolocation. Then we also have to have a Private Key of online storage that is sparkfun.com useful for storing location data obtained by raspberry-pi. Then the final step is to create an HTML file that will extract data from sparkfun and displays them in a user's browser page.

B. System Response Based on Internet Connectivity

This section will display the data obtained from the system response to certain conditions different conditions on the side of the transmitter (user) and the receiver (the system) to run the GPIO function.



Grafik 3.1 System Response Data Based on Internet Connection

From the data obtained can be seen clearly that when in one side or on both sides using a 4G connection, the delay will be smaller and will be faster system response. But in table 4.2 and table 4.3 it appears that when the 4G connection is in place on the system side (receiver) will have a faster system response than when a 4G connection in place on the user side (transmitter). This is due to the condition of the download and upload speed at the time of data collection 3G to 4G is higher than on the conditions of download and upload speed at the time of data collection 4G to 3G so that the system was more rapid response during data retrieval 3G to 4G.

C. System Response Based on Distance Between User and System

This section will display the data obtained from the response of the system when the user and the system is at a certain distance. In this section the data taken using a 3G connection to 3G at 8.30am when internet traffic is not too busy.

distance (km)	Respon Sistem (s)
0	0,63
2	0,88
4	0,75
6	0,96
8	0,55
10	0,93

Tabel 3.1 Response Time Data System Based on Distance between User and System

From the data obtained above, show that there is no increase or decrease of the response time of the system significantly. It shows that the distance between the user and the system does not affect the system's response time because the connection is used in this system is a network of the Internet, so it is not affected by distance but by the speed of the internet user and the system, and is also influenced by the condition of the internet network.

D. Response System Based on Network Performance

This section will display the data obtained from the response during different network conditions. The first data will be loaded when network traffic is not dense at around 17.00 pm. And the second is when network traffic was quite dense at around 02.00 am. In this section the data taken using a 3G connection to 3G to the distance between the user and the system is 0 km.

sample	Response system time (s)
1	0,4
2	0,21
3	0,38
4	0,19
5	0,2
average	0,276

Tabel 3.2 Response Time Data System On Not Busy Time

sample	Response System Time (s)
1	0,56
2	0,63
3	0,36
4	0,26
5	0,36
average	0,434

Tabel 3.3 Response Time Data System On Busy Time

From the table above it can be seen that the system response data taken during peak hours, which is about 02.00am, have a system response that is slightly slower than the system response data taken when the clock was not busy at around 17:00pm. It shows that the condition of the internet network traffic that is used influence on the resulting delay that will affect the system response time given.

E. System Response Based Overclocking Mode

This section will display the data obtained from the system response under conditions when overclock the raspberry pi in different settings each data collection for the functioning of geolocation or to get the location in relatime

Tabel 3.4 Response Time Data System Based on Overclock

In Table 3.4 show the results when users ask for the location of the system. In the overclocking mode None earned an average time to obtain the location of the system for 20 minutes 16 seconds. When using modest overclocking mode, medium, high, turbo, to PI2, showed a faster system response. But when seen from the time obtained from mode to mode PI2 none, the system response time is not far adrift. So it is not necessary overclocking mode fast by taking into account the risks posed on the hardware side.

Mode	Time to Get Location (minute)					
	1	2	3	4	5	Average
None	25:01	13:27	19:52	23:17	19:44	20:16
Modest	27:43	31:09	16:29	19:03	12:56	19:03
Medium	19:11	22:03	20:51	12:18	11:39	17:12
High	20:28	18:52	18:09	14:37	10:16	16:28
Turbo	11:57	17:32	11:19	13:02	12:03	13:10
Pi2	13:01	23:35	17:23	11:03	12:14	15:27

IV. CONCLUSION

1. When viewed from the overall system response data table based internet connectivity, it can be concluded that the faster download speeds than upload speeds. This is caused by the average user needs more using a download service that Internet service providers or operator provides services which have an average download speed is faster than the upload. It can also be influenced by the multiplexing techniques used at specific frequencies that can affect the download and upload speeds generated.
2. From the data system response is taken based on the distance indicates that the distance is not greatly affect the resulting system response, it is because the interface used in this system is a form of internet that is not limited by distance. During the system and the user is connected to the Internet network, the system can work well.
3. However, the response data captured by the system clock dense network traffic shows that the dense network traffic affect the response of a given system. That is because when a dense network used, then the commands sent by the user to the system will have a greater delay, it is what

then causes the system response will be slower when network traffic was congested internet use.

4. If the system response based overclock of raspberry pi, it can be concluded that when using the higher overclock mode will provide faster system response. That is because if the overclock mode in the settings at high speed on a raspberry pi, the processor can work up to speed 1000MHz which of course will provide faster system response.
5. In a system that is made in this study, the researchers only use overclock mode to none, it is because when seen from the table the data obtained, it can be concluded that the difference in average response time is not far adrift system. And taking into account the risk that when using overclock mode that is too high can cause rapid raspberry pi heat, which can damage hardware, then the overclock mode that is too high is not needed in this system. So the researchers only use the default overclock mode is none.

REFERENCES

[1] http://www.raspberrypi.org/ [accessed on 13 April 2015]

[2] https://Telegram.org/ [accessed on 21 April 2015]

[3] http://www.geolocation.com/ [accessed on 22 December 2015]

[4] Abadi, Iskandar. 2014. Pengertian Fungsi dan Jenis Jenis Modem. [Online] Available at: http://www.indoza.com/2014/04/penertian-modem-fungsi-dan-jenis.html/ [accessed on 13 April 2015]

[5] Max. 2014. Telegram on Raspberry Pi. [Online] Available at: http://www.emmeshop.eu/blog/node/44 [accessed on 27 December 2015]

[6] Gupta, Vikash. 2015. Raspberry Pi Based Location Tracker. [Online] Available at: https://theguywith3thumbs.wordpress.com/tag/raspberry-pi/ [accessed on 27 December 2015]

ATTACHMENT

A. *File Simlocator.py*

```
import requests,re
import xml.etree.ElementTree as ET

def getCellTowerInfo():
    print 'Logging in to router home page'
    baseurl = 'http://192.168.0.1/' #sesuaikan dengan yang kita gunakan
    url = baseurl + 'login.cgi'
    payload = {'ID':'admin','PASSWORD':'admin','REDIRECT':'index.asp','REDIRECT_ERR':'login.asp'}
    response = requests.post(url, data = payload)
    sessionPattern = re.compile('BID\d*')
    sessionId = sessionPattern.findall(response.text)[0]

    #mengambil data posisi dari data seluler
    payload ={'COUNT':'3','WWW_SID':sessionId,'ACTION_1':'function','NAME_1':'get_lac','VALUE_1':'','ACTION_2':'function','NAME_2':'get_cell_id','VALUE_2':'','ACTION_3':'function','NAME_3':'get_op_info','VALUE_3':''}
    rpcUrl = baseurl + 'rpc.cgi'
    response = requests.post(rpcUrl, data = payload)

    #parse data
    results = ET.fromstring(response.text).findall('rpc_result')
    lac = results[0].text
    cellid = results[1].text
    operator = results[2].text
    print 'Got tower location, logging out'
    print lac, cellid, operator

    logoutUrl = baseurl + 'logout.cgi'
    payload ={'WWW_SID':sessionId}
    requests.post(logoutUrl, data = payload)
    return (lac, cellid, operator)
```

B. *File Geolocation.py*

```
import requests,simlocator,json

def getLocation():
    (lac, cellTower,operator) = simlocator.getCellTowerInfo()
    #print lac, cellTower, operator
    codesDict =  {'Telkomsel':[510,10],'Smartfren':[510,9]}
# sesuaikan dengan daerah percobaan dan operator yang digunakan
    mcc = codesDict[operator][0]
    mnc = codesDict[operator][1]
    payload = {'considerIp':'false','cellTowers':[{'cellId':cellTower,'locationAreaCode':lac,'mobileCountryCode':mcc,'mobileNetworkCode':mnc}]}
    jsonPayload = json.dumps(payload)
    #print jsonPayload
    headers = {'content-type': 'application/json'}
    privateKey = "<your_key>" #dapatkan API KEY dari googlemaps developer
    url = "https://www.googleapis.com/geolocation/v1/geolocate?key=" + privateKey
    r = requests.post(url,data=jsonPayload,headers = headers)
    response = json.loads(r.text)
    #print response
    return (response['location']['lat'],response['location']['lng'],response['accuracy'])#mendapatkan Longitude dan Latitude
```

C. *File Publisher.py*

```
import urllib2, time

def publishtoInternet(temp):
    url = "http://data.sparkfun.com/input/dZa421pxLOhAVJKE1gYQ?private_key=<your_key>&temp=" + temp #ganti kode unik dengan KEY yang kita dapat dari akun sparkfun
    try:
        print url
```

```
        result = urllib2.urlopen(url).read()
        print result+ str(temp)
    except:
        print 'exception uploading, logging to file'
        log(temp)

def log(temp):
    f = open('log','a')
    t = time.localtime(time.time())
    msg =str(t.tm_hour)+' ' + str(t.tm_min)
    f.write(msg)
    f.write('temp: ')
    f.write(temp)

    f.write('\n')
    f.close()
```

D. File Main.py

```
import sys,time,geolocation,publisher
from subprocess import call

SleepTime = 10 # seconds
_lat = 0.00
_lon = 0.00

def maintain():
    global _lat
    global _lon
    x=y
    (lat,lon,accuracy) = geolocation.getLocation()
    if(lat != _lat or lon !=_lon):
        data = str(lat) + "," + str(lon) + "," + str(accuracy)
        print "publishing ", data
        publisher.publishtoInternet(data)
        _lat = lat
        _lon = lon
    else:
        print "no change in coordinates"

print "program begins"
while True:
    try:
        maintain()
    except Exception as inst:
        print type(inst), ' exception captured'
        print inst
        sys.stdout.flush()
        #file = open('/tmp/loctracker.error.log','a')
        #file.write('exception occured, trying to reboot')
        #file.close()
        #call(["sudo","reboot"])
    #break
    for i in range(0,SleepTime):
        sys.stdout.write("\restarting in %d seconds " %
(SleepTime-i))
        sys.stdout.flush()
        time.sleep(1)
```

E. File map.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Show Location on Google Maps</title>
    <meta name="viewport" content="initial-scale=1.0">
    <meta charset="utf-8">
    <style>
        html, body {
            height: 100%;
            margin: 0;
            padding: 0;
        }
        #map {
            height: 100%;
        }
    </style>
</head>
<body>
    <div id="map">Finding your location...</div>
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquer
y.min.js"></script>
    <script type="text/javascript"
src="http://maps.google.com/maps/api/js?sensor=false"></scri
pt>
    <script>
        function successHadler(lastPositions) {
            var homeLatLng = new google.maps.LatLng("-
6.885998, 107.625912);
            var mapOptions = {
                zoom: 12
                ,center: homeLatLng
            }
            var map = new
google.maps.Map(document.getElementById("map"),
mapOptions);
            var marker = new google.maps.Marker({
                position: homeLatLng,
                title: "Home"
            });
            var markers = [];
            markers.push(marker);
            var numberOfPositions = 3;
            for (var i = 0; i < numberOfPositions; i++)
            {
                lat = parseFloat(lastPositions[i]['temp'].split(',')[0]);
                lng = parseFloat(lastPositions[i]['temp'].split(',')[1]);
                time = new Date(lastPositions[i]['timestamp']);
                title = getTitle(time, lat, lng);
                if (lat === NaN || lng === NaN)
                    continue;
                latLng = new google.maps.LatLng(lat, lng);
                marker = new google.maps.Marker({
                    position: latLng,
                    title: title
                });
```

```
            markers.push(marker);
          }
          for (var i = 0; i < markers.length; i++)
          {
            markers[i].setMap(map);
          }
        }
        function getDistanceFromLatLonInKm(lat1, lon1, lat2,
lon2) {
          var R = 6371; // Radius of the earth in km
          var dLat = deg2rad(lat2 - lat1);  // deg2rad below
          var dLon = deg2rad(lon2 - lon1);
          var a =
            Math.sin(dLat / 2) * Math.sin(dLat / 2) +
            Math.cos(deg2rad(lat1)) * Math.cos(deg2rad(lat2)) *
            Math.sin(dLon / 2) * Math.sin(dLon / 2)
          ;
          var c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));
          var d = R * c; // Distance in km
          return d;
        }
        function getTitle(time, lat, lng) {
          formattedTime = time.getHours() + ": " +
time.getMinutes() + ": " + time.getSeconds();
          return formattedTime + " apart " +
getDistanceFromLatLonInKm(lat, lng, 21.2317941,
81.611056).toFixed(2) + " km";
        }
        function deg2rad(deg) {
          return deg * (Math.PI / 180)
        }
        function errorHandler(msg) {
          console.log(msg);
        }
        $.ajax({
          url:
"https://data.sparkfun.com/output/NJGQoZpO3LHjD2ZaJGG
4.json",
          dataType: 'jsonp',
          success: successHadler,
          error: errorHandler
        });
    </script>
</body>
</html>
```