APPLICATIONS TO SENTIMENT ANALYSIS USING RASPBERRY AND DIFFERENT MICROCONTROLLERS FOR DATA STREAMS CLASSIFICATION IN THE CONTEXT OF COVID-19

Valentin Mogoroase, Ph. D Student
University of Craiova, Department of Statistics and Informatics
Faculty of Economics and Business Administration
Craiova, Romania

Abstract: Using Raspberry Pi in order to perform an advanced Sentiment Analysis task, based on the pandemic context of COVID-19 and the classification process of all related tweets coming from my Twitter developer account as Positive or Negative. In additional a temperature senor connected to an IOT platform will be used to collect all critical values, in real time.

JEL classification: D81, D85, D91, O31, O33, O35.

Key words: Sentiment Analysis, Raspberry Pi, Temperature Sensor, Groove Relay, IOT Platform, COVID-19.

1. INTRODUCTION

Smart devices control using Raspberry Pi in order to perform sentiment analysis on a data stream extracted from my Twitter developer account, was manifested in the form of a desire to get acquainted with an advanced programming concept. This action is implemented through a microcontroller, which can provide a variety of connection types, in terms of synchronization models, provided in the BCM2835 board.

Creating an application using smart devices control requires both advanced knowledge techniques in various programming environments, as well as a creative process, regarding to the implementation of certain functionalities, all of which result in a structuring of ideas and how a diverse functionality can be designed.

The main goal is to perform sentiment analysis on a data stream containing tweets of a public opinion on Twitter about a certain subject, taking as example the actual context of COVID-19. We will analyze a series of positive or negative factors which can lead to a physiological decline or mood increasing/decreasing based on the tweets data stream extracted from the social network Twitter.

By performing this action we can access the available data through our smart device, in our case Raspberry Pi, which is connected to a temperature sensor MCP9808 to indicate body (higher than 37 ℃ - Critical) or environmental temperature and a Groove Relay to detect and warn us about the data stream receiving.

Data will be collected from an administrator account, which may be made public, if it wants to be visible to other users connecting to that channel, or by imposing a security restriction.

Internet protection and privacy required for a user to be in full security, when accessing the control of smart devices, are also ensured through a series of protocols established at the time data transmission is performed.

This application is also based on a process of posting the temperature on an IOT platform, being able to update current status for all users in the list and also call for a status, which can result in a graphical manifestation of the recorded data.

Within the application the user will have at his disposal a series of functionalities, the main one will be represented by data retrieving from the social platform Twitter, being classified as Positive/Negative, a warning trough Groove Relay and an eventual posting of the temperature registered by the temperature sensor (Low/Normal/High) on an IOT platform. All actions have administrative value and are established through a secure connection.

2. RASPBERRY PI - SYNCHRONIZATION MODELS OF A SMART DEVICE

The Raspberry Pi device, provide the user with a series of synchronization models, such as: UART Serial Interface, SPI Serial Interface (Peripheral Serial Interface), as well as Highway I_2C, which uses 2-wire- connection.

These synchronization models allow the control of the devices, connected to Raspberry Pi, through the pins provided by it, but each pin, can be used by different synchronization models [Sean and Mike, 2017].

In the following we will try to understand each feature of these synchronization models, provided by the Raspberry Pi device, every model having its own unique functionality, specialized for a purpose only.

2.1 UART Serial Interface

Because serial ports are asynchronous (no clock data transmission), the device that use them must agree before the certain time with a data rate. The two devices must also have clocks close to the same rate and will remain so excessive that the difference between the clock rates from each end will produce enormous data.

Asynchronous serial ports need hardware at the end of UART, to each end is relatively complex and difficult to implement exactly a software, if is necessary. At least one start and stop bit is a part of each data frame.

This means that 10 bits of transmission time are required for every 8 bits of data sent, which consumes from the data rate. Another basic malfunction in asynchronous serial ports is that they are in an inherently way suitable for communications between two and only two devices [Warren Gay, 2018].

While it is possible connecting multiple devices to a single serial port, bus conflict (if two devices try to transmit on same line at the same time) is always a problem and must be treated with caution to prevent any damage to the device in question, usually by external hardware.

So the data rate is an issue. Although theoretically there is no limit for asynchronous serial communication, most UART devices support only a certain set of baud data (bits per second), the largest being usually known to be around 230400 bits per second.

2.2 I_2C Bus (2-wire method)

I_2C bus – (Inter-Integrated Circuit) uses a protocol designed to allow digital integrated circuits called Slaves (chips) to communicate with one or more chips Master. As a Serial Peripheral Interface (SPI), it is intended only for short-distance communication in a single device.

Apart from the serial asynchronous interfaces (such as RS-232 or UART), it only requires two signal wires for information exchange. Although I_2C requires only two wires, such as asynchronous serial, the two wires can support up to 1008 Slave devices.

Unlike SPI, I_2C can withstand a multi-master system, allowing more master chips to communicate with all devices on the bus, although the main devices cannot talk to each other in bus and must rotate using bus lines [Derek Molly, 2016].

Data rates decrease between asynchronous serial and SPI. Most I_2C devices can communicate at 100 kHz or 400 kHz. There are also some higher rates with I_2C. For every 8 data bits to be transmitted, an additional bit must be transmitted, known as meta-data.

Each I_2C bus it consists of two signals: SCL and SDA. SCL is the clock and SDA is the data signal. The clock signal is always generated by the current master of the bus. Some Slave devices may force the clock to come down from time to time to delay the Master to send more data, or to take more time to prepare the data before the Master tries to get it clocking. This procedure is called "Clock Stretching".

Unlike UART or SPI connections, bus drivers I_2C are "Open Way", which means it can draw the corresponding signal line, but it cannot lead it to a higher level.

Thus, there can be a contradiction of the bus in which a device is trying to lead the high line, while another tries to pull it down, lowering the potential for damage to the conductor or excessive dissipation of power from the system [Simon Monk, 2012].

Each line of signal has a shooting resistance placed on it, to restore its level when no device claims to be low. The selection of the resistor varies from the devices on the bus, but a good rule is to always start with 4.7k and adjust if necessary.

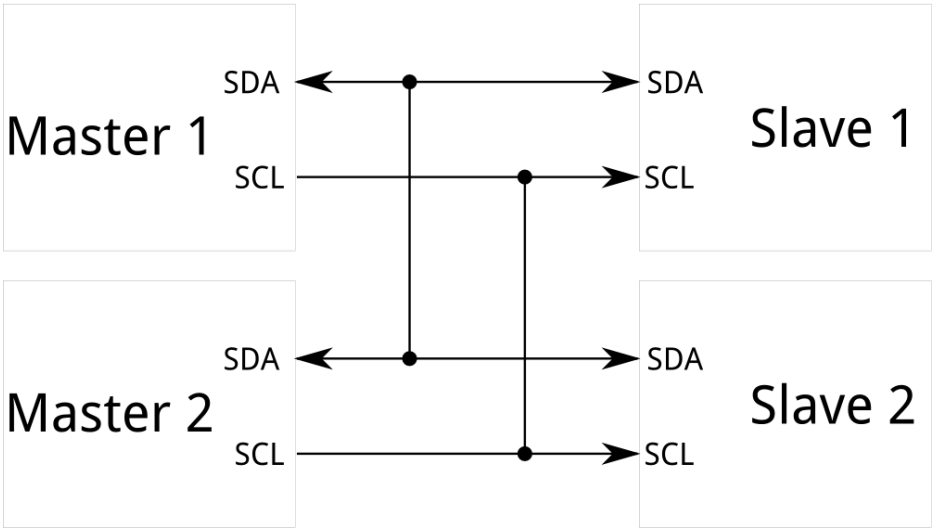

Figure no. 1
Transmission on I_2C level using the clock signal SCL and the data signal SDA.

I_2C is a robust protocol and can be used with short wires (2-3 m). For long circuits or systems with many connected devices, smaller resistors are better. Communication through I_2C is more complex than a UART or SPI. The signal must follow a certain protocol for the device on bus to recognize them as an available I_2C communication.

3. TWITTER COMMUNICATION USING TWYTHON

Twython is a Python library, being used to communicate with the Twitter API, the process of interaction between the smart device (Raspberry Pi) and the special platform created by Twitter for application development **developer.twitter.com** is achieved by the two types of authentication provided, OAuth1 & OAuth2.

3.1 Authentication (OAuth1 & OAuth2)

In the authentication process we will focus on the creation of the application as well as making basic API calls. First we will illustrate the application registration process on my developer account: **https://dev.twitter.com/apps**.

After registering the application we will need two keys generated from authentication process **Consumer Key** and **Consumer Secret**, these unique keys can be obtained from the application details tab.

Twython provides support for both types of authentication, OAuth1 as well as OAuth2, but there is a certain difference between the two types of authentication:

Oauth1 – is used for calls authenticated by the user (tweeting, tracking a tweet, filtering a tweet, etc.).

OAuth2 – is used for calls authenticated by the application (When we do not want to authenticate a user and make read-only calls on Twitter or when we want to search or read public users chronology).

3.2 OAuth1 (User Authentication)

A Twitter instance will be created with the two keys generated by the application registration, Consumer Key and Consumer Secret. If it's a web application it will pass the authorization address **callback_url** to authentication method **get_authentication_tokens**, otherwise if it's a desktop or mobile application do not require a callback_url. A callback_url have the following structure:

```
<< from twython import Twython >>
                    << APP_KEY = 'MY_APP_KEY' >>
                    << APP_SECRET = 'MY_APP_SECRET' >>
               << twitter = Twython (APP_KEY, APP_SECRET) >>
 << auth = twitter.get_authentication_tokens (callback_url = "callback_address") >>
<< OAUTH_TOKEN = auth ['oauth_token'] >>
<< OAUTH_TOKEN_SECRET = auth ['oauth_token_secret'] >>
```

From the authentication variable *auth*, the two tokens are stored, being used in the communication process between Raspberry Pi the platform integration of the application. Once the application is authorized, it will be allowed to access some of the account information and redirect it to a callback_url, as previously mentioned in the authentication method **get_authentication_tokens**.

It will also need to be extracted from the URL a code for application authentication called **oauth_verifier**, after which the creation of a Twython variable and final tokens incorporation.

<< Twitter = Twython (APP_KEY, APP_SECRET, oauth_token, oauth_token_secret) >>

<< end_s = twitter.get_authorized_tokens (oauth_verifier) >>

3.3 OAuth2 (Application Authentication)

Authentication method OAuth2, as a utilization stage, it can be easier to manipulate, compared to the authentication method OAuth1. Supposing to have already created the application and we have the two keys, Consumer_Key and Consumer_Secret, we will move on to the stage of obtaining one token through the authentication process OAuth2:

<< APP_KEY = 'MY_APP_KEY' >>

<< APP_SECRET = 'MY_APP_SECRET' >>

<< twitter = Twython (APP_KEY, APP_SECRET, oauth_version = 2) >>

<< ACCESS_TOKEN = twitter.obtain_access_token () >>

After extracting the access_token it can be implemented a final method of obtaining access to the application development platform and also establishing a stable connection between this platform and the smart device:

<< APP_KEY = 'MY_APP_KEY' >>

<< APP_SECRET = 'MY_APP_SECRET' >>

<< twitter = Twython (APP_KEY, access_token = ACCESS_TOKEN) >>

4. APPLICATION DEVELOPMENT BASED ON API

It is necessary to determinate the process by which the application can send feedback directly, following which a response from the server is expected. The connection that the Raspberry Pi device establishes with the web server is made through an API, following this communication process, we will be able to achieve a series of actions.

An API refers to an Application Programming Interface with additional layers and standardized communications and can also provide options on how to format inputs and outputs so that they can be used frequently in several systems.

Web-API applications usually include instructions for sending data or commands in order to retrieve information from a website or web application. An API can be:

1) A **language dependent** – meaning that is available only to use syntax and elements of a particular language, making the API interface convenient to use.

2) A **language independent** – written so that it can be called trough many programming languages. This feature is desired in an API service-orientated, which is not related to a particular process or system and can be provided from distance, as a procedural call or as a web service.

4.1 Data Exchange between sensor and platform (REST-API & JSON)

This data exchange between the MCP9808 temperature sensor and the platform, help us understand how they communicate internally, when the value of the recorded temperature is posted.

REST (Representational State Transfer), is used as a particular style of creation for APIs, being designed to be easy to use, suitable for a large volume of services. Twitter as well as many other sites, use this REST approach for most application programming interfaces.

One of the main reasons is that REST is based on a protocol that represents the source of all connections made on the internet, HTTP. This protocol uses the most important methods like POST, GET, PUT and DELETE. This REST is extremely popular and used since any program that connects to the internet uses this type of API.

JSON (Java Script Object Notation) describes a data exchange format based on JavaScript programming language. What is so special about this format is that can be easily understood by both machine language and programmer [Rushi Gajjar, 2015].

Its main advantage is that most programming language already has specific encoders and decoders for converting their data structure to JSON and vice versa. This means that a JSON interface is possible to behave like an interpreter between two applications that use different programming languages, interpreter without which this communication would not have been possible.

In order for the device to be able to communicate with Twitter server, it must use a certain protocol, in our case HTTP, connection that is made through the **httplib**, implemented within the application.

The principle behind Twitter is to provide a single method of accessing the API, which represent the process of communication between the device and its application development platform.

This request method can be called trough any end point of reference, points that can be found on the developer's Twitter site.

4.2 Raspberry Pi synchronization with MCP9808 via Groove Relay

The application consists in control trough the Twitter platform of a temperature sensor MCP9808 via a Groove Relay and sending the value recorded from the user or environment to an IOT statistical chart, being real-time recorded.

After recording the current sensor temperature and posting it on an IOT platform, the Groove Relay filter is triggered every time a post containing a tweet with of a specific category took place in users account.

We can type a specific command which will not only bring the desired result but it will also signal its status reported to a specified filter through the Groove Relay, setting it to HIGH (Open), in the moment of receiving the status, after which to LOW (Closed), after the transmission has been done.

Control of the Groove Relay through the Twitter platform, as well as its displayed status on runtime, HIGH or LOW, is given by the following streaming function:

```
<< BlinkyStreamer class (Twython Streamer): >>
                    << def on_success (self, data): >>
                        << if 'text' in data: >>
                    << print data ['text'].encode ('utf-8') >>
                        << print 'Check' >>
```

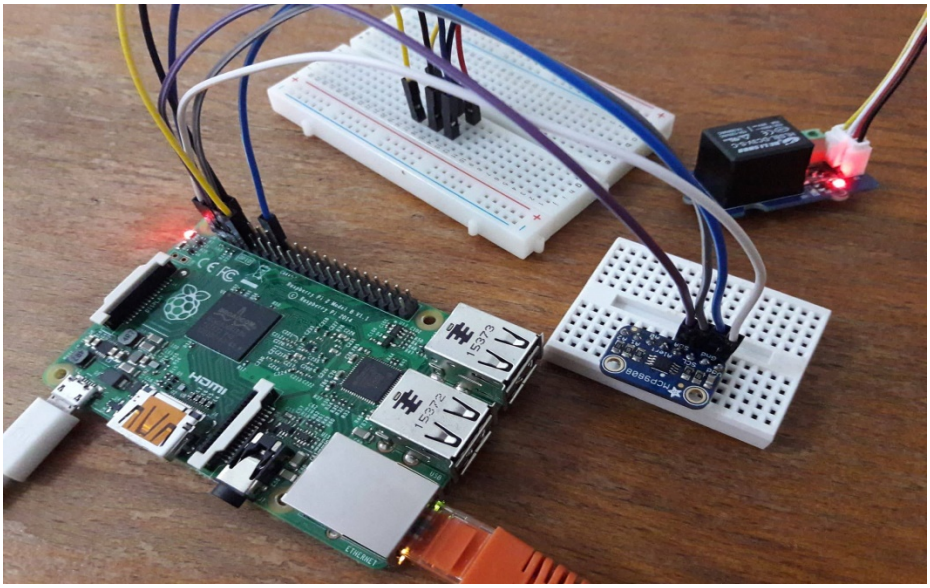

Figure no. 2

Connect the temperature sensor MCP9808 with a Groove Relay on the motherboard, by 2-wire method, to determinate environment or body temperature.

When the message filtered meets the parameter of the imposed condition, Groove Relay, set to one of the GPIO pins, will set the let from the plate, on which is incorporated in one of the defined statuses: HIGH (ON), if the message was successfully filtered and LOW (OFF), if the message transmission was finished or if the corresponding message was not filtered successfully.

After synchronization, the Groove Relay connected with the temperature sensor MCP9808, will receive a warning, to highlight or not the incorporated LED by turning it ON or OFF, depending of the message status at that point.

5. MESSAGE FILTERING USING REST-API

In additional for sending calls, for example to determinate if a tweet coming from the Twitter platform contains a certain filter 'f = COVID-19', REST API iterators and streaming APIs can return errors or more messages related to that filter.

However, the application decides the type of the returned object. REST APIs can return even more message types compared to Streaming APIs.

REST APIs provide programmatic access for reading and writing data from Twitter, such as filtering a new message (tweet), reading a user's profile, identifying data flow received by a follower. To filter a word we'll use Natural Language Processing (NLTK).

In our case we will filter a set of tweets coming from the Twitter platform having a specific filter 'COVID-19', after which a strong analysis of this collected data will be made in order to define its social impact. We will illustrate a message filtering containing a predefined filter using Anaconda environment:

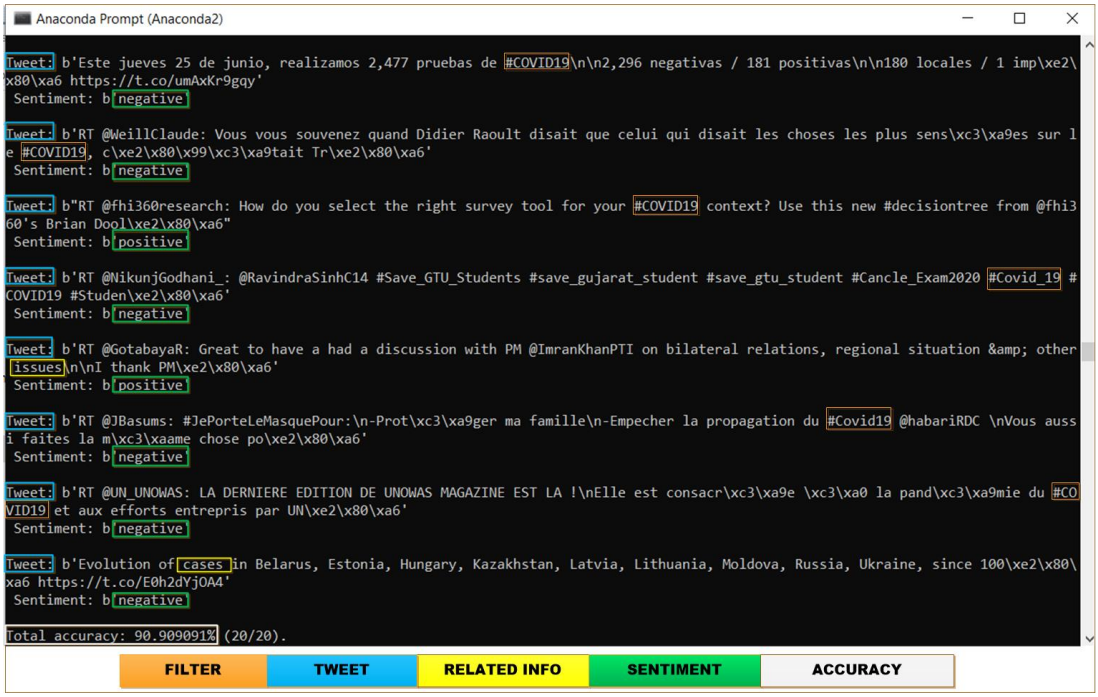

Figure no. 3

Sentiment analysis result by filtering the tweets coming from my Twitter developer account, containing the key word 'COVID-19', using Anaconda2 Prompt.

Every tweet containing the key word 'COVID-19' or a Related Information is analyzed in order to determinate its Positive or Negative value, defining user sentiment related to a certain topic, in our case a pandemic situation.

The REST API identifies Twitter applications and users who use the process of authentication (OAuth). Depending on the endpoint we can handle a certain type of information. We will consider the following pattern to classify tweets which meets the specified filter condition (q) or its identification code:

```
<< r = api.request ('search / tweet', {'q': 'tweet'}) >>
        << for item in r.get_iterator (): >>
            << if 'text' in item >>
            << print item ['text'] >>
        << elif 'message' in item: >>
<< print '%s (%d)'% (item ['message'], item ['code'])
```

5.1 Social Impact Analysis based on Collected Data

The command performed to return a specified filter containing the key word 'COVID-19', can be achieved within our message or tweet through the process of streaming.

For example a message "limit" contains the number of missing tweets from the stream. This happens when the number of tweets that match the filter search, exceed a certain threshold set by Twitter (tweet ≤ 140).

Every tweet collected from my Twitter developer account is analyzed based on psychological effects of a pandemic situation, targeting users experience and the type of sentiment (Positive or Negative) transmitted in a social environment, having a huge influence to a wide public.

We will also analyze a significant part of the collected data through our application, in order to determinate the evolution of a certain topic extracted by the filter. In our case to determinate if a pandemic situation like COVID-19 can psychologically affect people in a significant percent (HIGH = Negative) or an insignificant percent (LOW = Positive).

Table no. 1

Performing sentiment analysis on the data collected from my Twitter account, establishing the impact of the COVID-19, based on user's interaction.

TWEETS EXTRACTED (REST-API & STREAMING)	SENTIMENT (POSITIVE/NEGATIVE)	EFFECT (LOW/HIGH)
@*Twitter*: Evolution of cases in Belarus, Estonia, Hungary, Kazakhstan, Latvia, Lithuania, Moldova, Russia, Ukraine, since 100\xe2\x80\xa6 https://t.co/E0h2dYjOA4.	Negative	HIGH
@*galileocheng:* First in 17yrs, Police officially objects the July 1st assembly & rally upon #COVID19 & citing violence after @chrf_hk ra\xe2\x80\xa6.	Negative	HIGH
@*PoliticalIbrah*: Eii you people can talk oo…eii \n Are done assessing this gov. \n What's the Dollar rate now? \n How far wi\xe2\x80\xa6 https://t.co/9LxsA7VShi.	Negative	HIGH
@*WIONews*: The Indian film industry is hit by #COVID19 & suffering huge financial losses. We discuss the future of filmmaking \xe2\x80\xa6.	Negative	HIGH
@*CGTNOfficial*: Spanish virologists have found traces of the novel #coronavirus in a sample of #Barcelona waste water collected in March\xe2\x80\xa6	Negative	HIGH
@*jocelyn90028*: Tell @VP that because of him and @realdonaldtrump, my grandpa will be buried on Tuesday. #COVID19 https://t.co/RNRqDifNMA.	Negative	HIGH
@*aubrey_huff*: #COVID19 was never about a virus. It is a plan to usher in communism.	Negative	HIGH
@*Twitter*: It has now been proven that exams are also an act of spread of the virus as a 10th class student who attempted Kar\xe2\x80\xa6 https://t.co/qpseIrZQdk.	Negative	HIGH
@*jamewils:* The reports seemed to take doctors by surprise: The \xe2\x80\x9crespiratory\xe2\x80\x9d virus that causes Covid-19 made some patients nauseous. \n\nI\xe2\x80\xa6.	Negative	HIGH
@*Umar_Khan10:* Government paying utility bills for small business in an attempt to shoulder some of their expenses in times of Covid19.\xe2\x80\xa6.	Negative	HIGH
@*QuickTake:* Young Americans are having trouble finding jobs and internships this summer due to the #Covid19 pandemic https://t.co/RCpZKg\xe2\x80\xa6.	Negative	HIGH
@*ZackBornstein:* Finally some good news, scientists discovered a treatment that can reduce COVID19 transmission by 70%, and it's just a pi\xe2\x80\xa6.	Positive	LOW
@*eupatientsforum:* #COVID19 has shone a light on the		

weaknesses within our healthcare systems & we hope that the impact on vulnerable po\xe2\x80\xa6.	Positive	LOW
@revathitweets: Famous games show producer/anchor who also directed a hit horror film earlier, tested positive #COVID19. He was shooting\xe2\x80\xa6.	Negative	HIGH
@DrGJackBrown: Houston, the 4th largest city in the United States, has no ICU beds available. They are full. \n\n#COVID19.	Negative	HIGH
@ARYNEWSOFFICIAL: Coronavirus traces found in March 2019 sewage sample: Spanish study\n\n #ARYNews #COVID19.	Negative	HIGH
@krishgm: Very strong warnings from Prof John Edmunds tonight that we could already be heading into a second wave of Covid19 and about t\xe2\x80\xa6.	Negative	HIGH
@USAmbNepal: As #COVID19 cases rise, my team is working hard to support the GoN with funds & equipment to #beatthevirus! Proud to join m\xe2\x80\xa6.	Positive	LOW
@MikeLawlor: Big news! Connecticut\xe2\x80\x99s total prison & jail population has dropped below 10,000 for the first time in 30 years!! This morn\xe2\x80\xa6.	Positive	LOW
@XHNews: Latest count of confirmed #COVID19 cases worldwide at 1000 GMT, June 27:\n World: 9,821,596\n U.S.: 2,467,837\n Brazil: 1,274,974\n Rus\xe2\x80\xa6.	Negative	HIGH
@AAI_Official: Ever since the spread of #COVID19, our lives have undergone vast changes; be it our lifestyle, job, health or financial s\xe2\x80\xa6.	Negative	HIGH
@Twitter: Director of Africa CDC John Nkengasong said Africa must be careful and prepare for a rise in the number of #COVID19 \xe2\x80\xa6 https://t.co/jPhbFEo9pJ.	Negative	HIGH

An analysis of a small part of the collected data have shown me that even on a large set of data containing the key word 'COVID -19', the NEGATIVE sentiment predominates, compared with the POSITIVE sentiment. This is due to the pandemic context of COVID-19, which seems to have a Negative reputation among Twitter users around the world.

6. STATISTICAL DATA REGISTRATION USING AN IOT PLATFORM

In additional we will use an IOT (Internet of Things) platform called *ThingSpeak* to collect data from the Raspberry Pi device. Using the temperature sensor MCP9808 we can record the temperature of the environment or our body temperature, in order to trace Critical Values.

Data collection is made via private channels. The channel has a unique key identification that is used to identify that exact channel when reading or uploading data. Each channel has up to eight fields in which the data, both numerical and alphanumerical, can be stored.

It also contains four additional fields for location details. All entries are stored with a unique identifier and a date and time printout. The data from this channels it's replaced with real time information collected from the temperature sensor.

The temperature sensor read its values from the environment in which it is placed or from our body, creating a graphic chart representation of the recorded temperature values, being displayed in Celsius degrees as well as Fahrenheit degrees.

This procedure is achieved by using the POST method from HTTP protocol, in combination with the unique key of the writing channel.

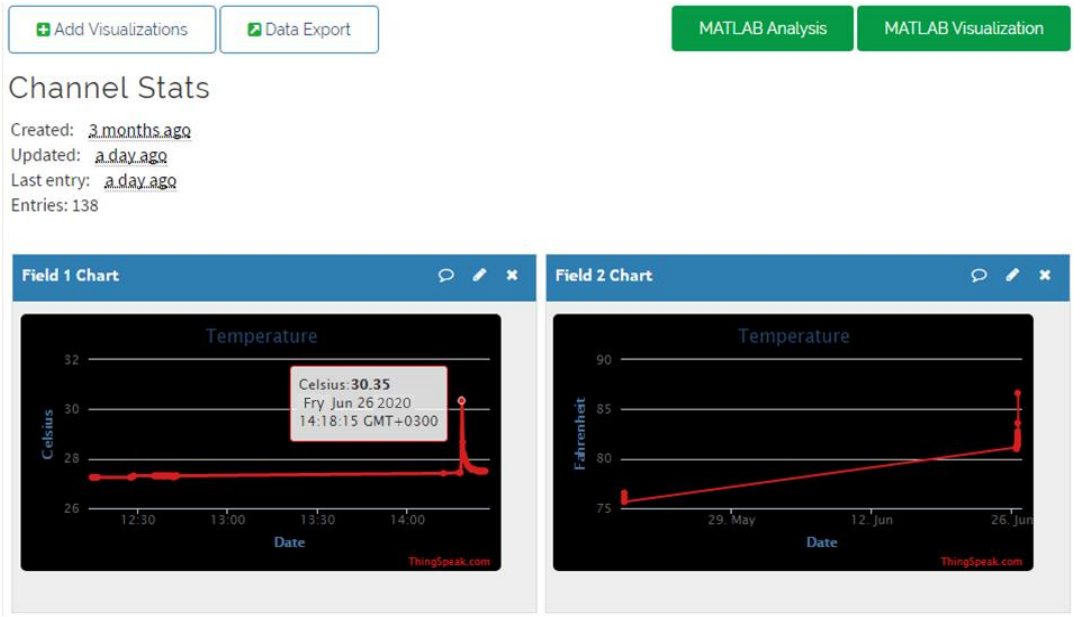

Figure no. 4

Graphic representation of a Critical Value using the temperature sensor MCP9808 on an IOT platform, using as main source the Raspberry Pi device.

Using the IOT platform we can monitor real time data collected from the temperature sensor MCP9808, based on the temperature of the environment or the body temperature, being able to identify Critical Value, as well as the location in time of that value extracted by the private channel.

7. CONCLUSIONS

Using Raspberry Pi in order to perform a sentiment analysis on the data collected from the social platform Twitter, have been brought a number of functionalities, which can offer a wide range of applicability.

Apart from the complex hardware connection of its components, the application uses advanced strategies of NLTK and methods like TWYTHON, in order to analyze a series of tweets extracted through a streaming process from my Twitter developer account and to classify them, related to a certain topic, in our case the pandemic context of COVID-19, as Positive or Negative.

In additional, the application offer the possibility of data temperature recoding from the temperature sensor MCP9808, based on IOT platform ThingSpeak, being able to display a graphical representation of all Critical Values, in real time, using private channels.

Temperature data recorded from the sensor can be the temperature of the environment or if it rise above 37 ℃ and it's a value recorded from the human body, can be classified as a Critical Value and should offer some vital information about an infection with the COVID-19.

Being designed on a complex level, the application has also its weaknesses. When we load data through the API, there is a limit to the channel update for every fifteen seconds. I have found that the reason for this limit on loading is due to the excessive bandwidth that could be used and therefore additional funds would be needed, being a Non-Profit Service.

A solution for this limit problem would be to place the entire API to another web host provider to run the API. Although it is not considered a weakness comparing to the competing APIs introduced, it would offer a more complex functionality if the chart types could contain a wide variety to choose from, in additional to the current ones.

In order to use ThingSpeak some additional knowledge are required. For other paid competitors such as Carriots, APIs offers code fragmentation to make integration easier. This should not represent a major problem since it is a Non-Profit Organization and the ThingSpeak platform progresses every day, having a growing community.

REFERENCES

1. Simon Monk. Programming the Raspberry Pi: Getting Started with Python, McGraw Hill Professional, p192, 2012.
2. Sean M, Mike C. Raspberry Pi For Dummies, 3rd ed., John Wiley & Sons, p512, 2017.
3. Derek Molloy. Exploring Raspberry Pi: Interfacing to the Real World with Embedded Linux, John Wiley & Sons, p720, 2016.
4. Rushi Gajjar. Raspberry Pi Sensors. Community experience distilled, Packt Publishing Ltd, p192, 2015.
5. Warren Gay. Advanced Raspberry Pi: Raspbian Linux and GPIO Integration, Apress, 2nd ed., p512, 2018.
6. * * * https://learn.sparkfun.com/tutorials/i2c/i2c-at-the-hardware-level
7. * * * http://telelinea.free.fr/a10/module_i2c/ch_2/sec_1/page1.htm
8. * * * http://dlnware.com/dll/Clock-Synchronization
9. * * * http://www.nxp.com/documents/user_manual/UM10204.pdf
10. * * * http://ww1.michrochip.com/downloads/en/DeviceDoc/25095A.pdf
11. * * * https://learn.adafruit.com/mcp9808-temperature-sensor-python-library/overview
12. * * * http://wiki.seed.cc/Grove-Relay/
13. * * * http://mediatechnology.leiden.edu/images/uploads/docs/wt2014_thingspeak.pdf
14. * * * https://dev.twitter.com/docs
15. * * * http://twitterapi.pbworks.com/w/page/22554679/Twitter%20API%20Documentation
16. * * * https://pinout.xyz