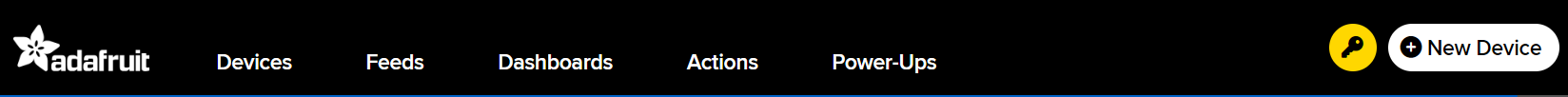
IOT Report for the Data Analytics Part

This report summarizes the details and methods used in the data analytics section of Internet of Things. Data analytics is the process of collecting varying data from different sources of data, and to visualize the data make sense of it by finding hidden patterns, which leads to data-driven decision making. In the case of Internet of Things, data is primarily collected by **sensors** (also called things or devices) from the environment. These sensors can be one or many depending on the size and complexity of the problem. The data collected by these sensors can be large, volatile, and hard to work with when it is not stored in a centralized system. This is where the role of cloud/local storage system comes in. We store the volatile data to prevent data loss, and to make it easier to work with large amounts of data. Data analytics also offers visualization of data through **dashboards**. We can also automize actions that can be taken by the system when certain conditions are met, which satisfies one of the goals of IoT, which is to make all actions taken for a certain problem automated.

To cover the basic concepts of data analytics, we will show the use of three widely used IoT platforms: Adafruit IO, Kaa IoT, ThingsBoard. In this report, we will be simulating data generation with the use of python commands instead of using sensors for the sake of learning.

**Adafruit IO:**

This platform gives us cloud-based service. This means that the data is stored in a centralized system whose location is unknown to us. All actions taken are also cloud-based. Setting up an Adafruit platform is as simple as creating an account in Adafruit and directly using its services.

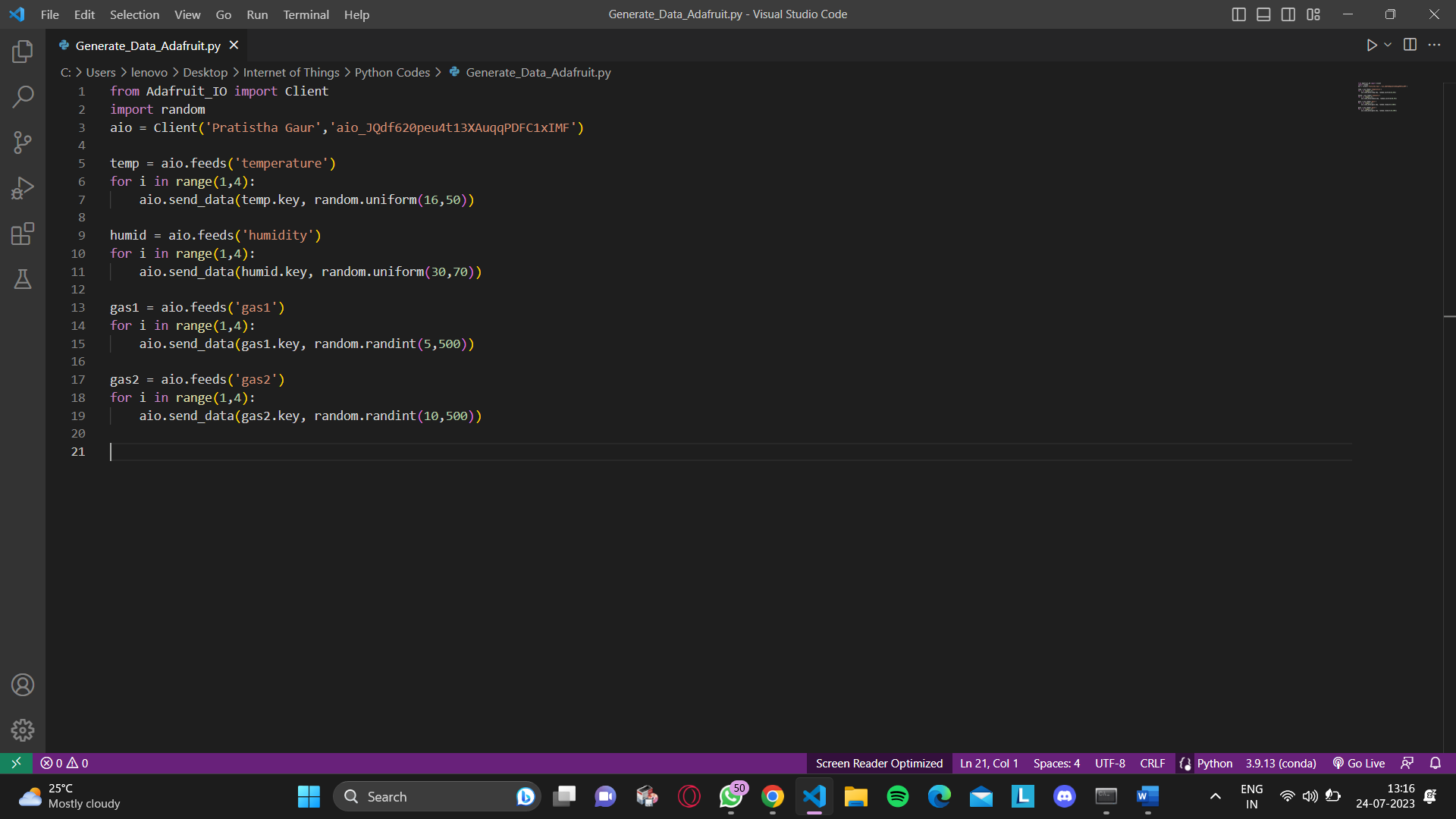
Graphical user interface, application

Description automatically generatedFirstly, we must create data feeds to store the data generated. This is as simple as going to the ‘Feeds’ tab in the Adafruit website and creating a new feed and giving the feed a name. Make sure to note down the feed key, as it is important and used in data generation. An example of some feeds is shown below:

To generate data and send this data to a feed using python, we must first install Adafruit’s libraries. This can be done by running the command prompt and entering the following command:

**pip3 install adafruit-io**

After running this command, we open any python interpreter and run the following code:



Note that the censor used above is to hide the **AIO Key**, which is specific to every Adafruit user, and is not recommended to be shared as data can easily be compromised with this key. Note that the arguments passed to the **aio.feeds()** method is the feed key of each feed. This code block, when executed, generates 3 random data values in mentioned ranges and sends it to each data feed. An example of how the data could look is shown below:

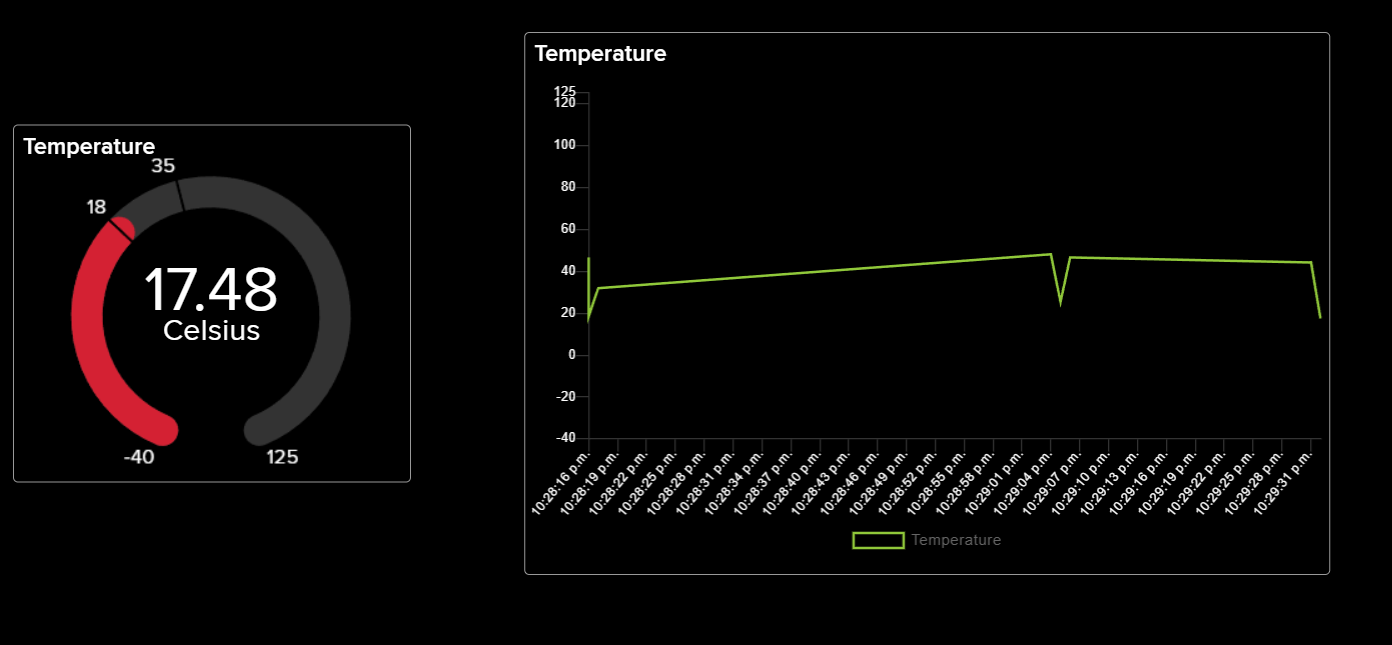
Graphical user interface, text, application, email

Description automatically generated

The next step is to create a dashboard. This can also be simply done by going to the ‘Dashboard’ tab and creating a new empty dashboard. We can then add widgets that can be used to visualize the data received. An example of a dashboard is shown below:

Graphical user interface

Description automatically generatedA screenshot of a computer screen

Description automatically generated with low confidence

The last step is to automate a response from the system when certain conditions are met. This could be anything from sending a message or an email, to performing on-site actions through the use of actuators. All of these actions work based on the same principle of satisfying user-defined conditions. In Adafruit, actions can be implemented by going to the ‘Actions’ tab. A new action can be created by creating conditions based on the requirements of the user. An example of an action and its creation is shown below:

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Text

Description automatically generated

**Kaa IoT:**

The KAA cloud is an IoT platform-as-a-service that you can sign up and use it for free for 14 days. KAA offers a set of IoT features that can be easily plugged in and can be used in several IoT use cases.

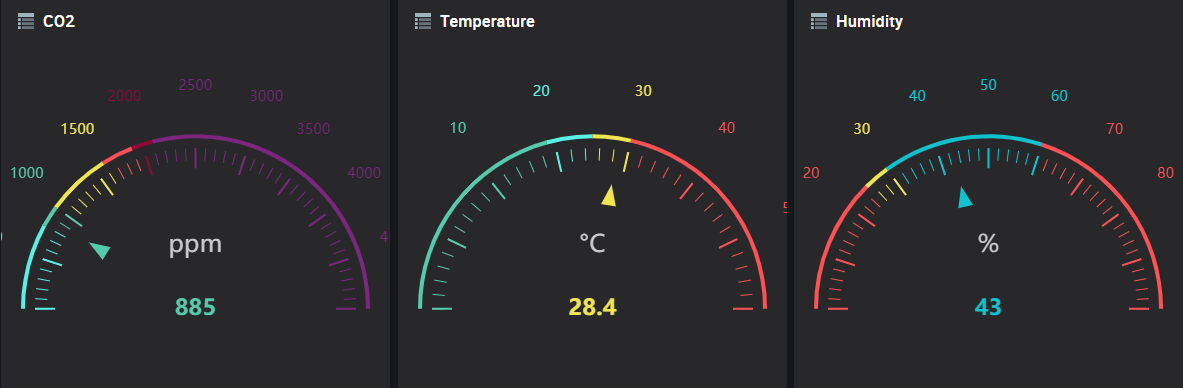
DIY air monitoring template provides with a customizable inbuilt code for automatic simulation of the data, wherein the temperature, humidity, Air quality index, CO2 range is generated using the function random.randint.

Text

Description automatically generated

Chart

Description automatically generatedChart

Description automatically generatedThe code is integrated to the application version of KAA that activates the dashboard tab and creates a dashboard with digital gauge widgets and the graph which again can be edited based on our needs.

Graphical user interface

Description automatically generated

In order to set up an alert when certain conditions are met, we need to integrate to the kibana interface and also create a slack workspace account.

We need to create a monitor for each alert we wish to notify, which can be done by just loading the variables for the observation.

The alert type and message are then created, and the parameters are set for which we need to trigger the action.

Graphical user interface, application

Description automatically generatedText

Description automatically generatedWhenever the trigger conditions are met, we receive an alert message in the slack workspace and also in the dashboard the graph shows an indication of the alert sent.

**ThingsBoard:**

ThingsBoard offers both cloud-based and local-based platforms. We will be using the cloud version of ThingsBoard as it is easier and does not require installation of the ThingsBoard software itself. Setting up a ThingsBoard Cloud platform is done by creating a ThingsBoard account with an email.

We first create a device that collects data. This is done by clicking on Device groups > All, and then clicking the + button.

Graphical user interface, text, application, email

Description automatically generated

We specify the details of the device and then click on Add. Clicking on a device shows the following details:

Table

Description automatically generated with medium confidence

Note that the **access token** is an important alphanumeric string(works like API key) which is used to bind a hardware(device) with the ThingsBoard platform. The recorded data is stored in the “**Latest Telemetry**” tab shown above. As the device is just created, there isn’t any recorded data stored. We can send data to the platform either through the hardware, or manually sending the data values through python or scripting commands. We will be using python commands to send time series values.

Data can be transferred through MQTT protocol or HTTP protocol. We will be using the MQTT protocol to send data. Before moving onto the data generation code, we must first install paho-mqtt library for using MQTT data transfer. This can be done by typing the following in the command prompt:

**pip3 install paho-mqtt**

After installing the library, we must execute the following code block in a python interpreter:

Text

Description automatically generated

The access token entered must match with the device’s access token for this code to work correctly. This code generates temperature, humidity, and gas data every 5 seconds until it is stopped manually. The generated data is then sent to the device in ThingsBoard, where the values are recorded in the “Latest Telemetry” section of the device’s data.

The next step is to visualize this data in the form of understandable format. This is done with the help of dashboards with widgets. We can create a new dashboard by clicking Dashboard Groups > All and then clicking the + button.

Graphical user interface, text, application

Description automatically generated

Enter a suitable dashboard title with description and then click on Add. Open the empty dashboard to add widgets. When adding a new widget, we must choose an appropriate entity that is connected to a device, and also choose appropriate data sources. An example is shown below:

Graphical user interface, text, application

Description automatically generated

Multiple such widgets can be created with different tools like charts, cards, digital meters, etc. Data can be visualized both in real time or as a history of past data. This can be done by clicking the following button:



Examples of real-time and history type visualization is shown below:

Chart, line chart

Description automatically generated

Chart

Description automatically generated with medium confidence

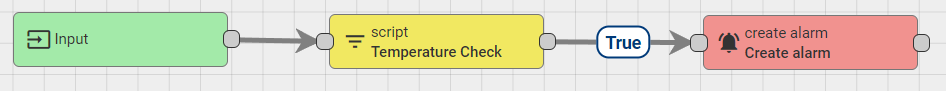
Other examples for widgets include:

Chart

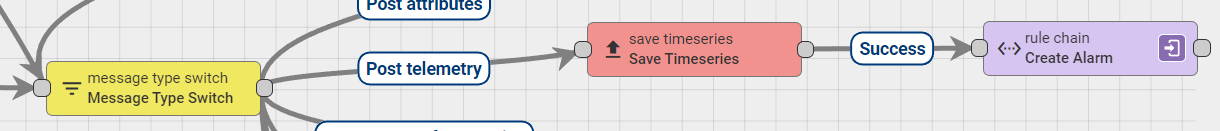
Description automatically generated

Notification alerts and other appropriate actions that can be taken when certain conditions are met, can be defined using **Rule Chains**. To create a rule chain, we need to click on Rule chain and then click the + button.

We open the rule chain and add user defined rules. When these rules are satisfied, user defined actions are performed. One such example of rule chain is:



We then add this newly created rule chain to the root rule chain and connect the rule chain appropriately to the ends of the root rule chain.



To implement this rule chain we just created, we must go back to our dashboard and add the Alarms Table widget. By adding the appropriate entity and choosing required options, the alarm widget can send alarms when certain threshold values are passed during data generation.

Graphical user interface, text, application, chat or text message

Description automatically generated

Graphical user interface, text, application, chat or text message

Description automatically generated

Using rule chains, we can perform many other actions such as sending messages or emails to the customer/user, push data to other devices, generate reports, etc.