# SPRINT 1

**Data Collection and Data Pre-processing**

| | |
|---|---|
| Date | 03 November 2022 |
| Team ID | PNT2022TMID27071 |
| Project Name | Project - Gas Leakage Monitoring and Alerting System for Industries. |

## Data Collection:

- Data Collection is a process of gathering information from all the relevant sources to find a solution to the research problem.
- Most leak detectors are primarily responsible for locating the leak, determining the extend or rate of leakage, and keeping track of increase or decrease in leakage.

## Pre-processing:

- Data pre-processing, a component of data preparation, describes any type of processing performed on raw data to prepare it for another data processing procedure.
- Data can be cleaned by dividing it into equal size segment that are thus smoothed (binning), by fitting it to a linear or multiple regression function (regression), or by grouping it into cluster of similar data (clustering).

## Code:

```
#include <ESP8266WiFi.h>

#include <PubSubClient.h>

WiFiClient wifiClient;


//Enter your network credentials below in ssid and password

const char* ssid = " ";

const char* password = " ";


//Provide your IBM IOT Platform credentials

#define ORG ""

#define DEVICE_TYPE ""

#define DEVICE_ID ""

#define TOKEN ""
```

```
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
char publishTopic[] = "iot-2/evt/Data/fmt/json";
char topic[] = "iot-2/cmd/home/fmt/String";  // cmd  REPRESENT command type AND COMMAND IS TEST OF FORMAT STRING
char authMethod[] = "use-token-auth";
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;

void callback(char* topic, byte* payload, unsigned int payloadLength);
PubSubClient client(server, 1883, callback, wifiClient);

int publishInterval = 5000; // 30 seconds
long lastPublishMillis;
String data;

void setup()
{
  Serial.begin(9600);
  pinMode(D0, OUTPUT);
  wifiConnect();
  mqttConnect();
}

void loop() {
  if (millis() - lastPublishMillis > publishInterval)
  {
    publishData();
    lastPublishMillis = millis();
  }

  if (!client.loop()) {
    mqttConnect();
```

```
  }
}

void wifiConnect() {
  Serial.print("Connecting to "); Serial.print(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.print("nWiFi connected, IP address: "); Serial.println(WiFi.localIP());
}

void mqttConnect() {
  if (!client.connected()) {
    Serial.print("Reconnecting MQTT client to "); Serial.println(server);
    while (!client.connect(clientId, authMethod, token)) {
      Serial.print(".");
      delay(500);
    }
    initManagedDevice();
    Serial.println();
  }
}

void initManagedDevice() {
  if (client.subscribe(topic)) {
    // Serial.println(client.subscribe(topic));
    Serial.println("subscribe to cmd OK");
  } else {
    Serial.println("subscribe to cmd FAILED");
```

```
  }
}

void callback(char* topic, byte* payload, unsigned int payloadLength) {

  Serial.print("callback invoked for topic: ");
  Serial.println(topic);

  for (int i = 0; i < payloadLength; i++) {
    //Serial.print((char)payload[i]);
    data += (char)payload[i];
  }

  Serial.println("Data: " + data );
  if (data == "lon") {
    digitalWrite(D0, HIGH);
  }
  else if (data == "loff") {
    digitalWrite(D0, LOW);
  }
  data = "";
}
void publishData()
{
  int a = 10;
  Serial.print("Sample Value: ");
  Serial.println(a);

  String payload = "{\"d\":{\"data\":";
  payload += a;
  payload += "}}";
```

```
  Serial.print("\n");
  Serial.print("Sending payload: ");
  Serial.println(payload);

  if (client.publish(publishTopic, (char*) payload.c_str())) {
    Serial.println("Publish OK");
  } else {
    Serial.println("Publish FAILED");
  }
}
```