GROUP NO 3

Mohit Jajoriya, Lacky Kulhari, Utkarsh Chavan, Ega Vashisht, Ritwik Gupta, Samay Jain

MQTT -
Broker - broker.hivemq.com
Colab Code -
Publisher
https://colab.research.google.com/drive/1SeUK-63tk69ol86KvL_TnkEeLIYSfmCN?usp=sharing
Subscriber -
https://colab.research.google.com/drive/1pAT6941nbMxKxXXrO_J2lbjIK-vj5uyQ?usp=sharing

Publisher Code - Parameters - Velocity, Feed, positions in X, Y and Z axis.
Currently taken as random values.
Code File -

```
import paho.mqtt.client as mqtt
from random import randrange, uniform
import time

mqttBroker = "broker.hivemq.com"
client = mqtt.Client("ParaMeters")
client.connect(mqttBroker)



try:
  while True:
    velocity = uniform(500,520)
    feed     = uniform(0.1,0.9)
    x_axis   = uniform(0,10)
    y_axis   = uniform(0,10)
    z_axis   = uniform(0,1)

    client.publish("VELOCITY", velocity)
    client.publish("FEED", feed)
    client.publish("X_AXIS", x_axis)
    client.publish("Y_AXIS", y_axis)
    client.publish("Z_AXIS", z_axis)

    print("Just published " + str(velocity) + " to Velocity")
    print("Just published " + str(feed) + " to Feed")
    print("Just published " + str(x_axis) + " X - Axis")
```

```
    print("Just published " + str(y_axis) + " Y - Axis")
    print("Just published " + str(z_axis) + " Z - Axis")
    time.sleep(5)
except KeyboardInterrupt:
    print("Stopping the MQTT clients.")
```

Output @ terminal of Publisher

```
Just published 504.66792194806953 to Velocity
Just published 0.395018767157412 to Feed
Just published 8.748362344375005 X - Axis
Just published 1.5196179714882174 Y - Axis
Just published 0.7692995313434461 Z - Axis
Stopping the MQTT clients.
```

Subscriber Code -

```
def on_message(client, userdata, message):
    print(f"Received message on topic {message.topic}:
{str(message.payload.decode('utf-8'))}")

# Define the broker and topics
mqttBroker = "broker.hivemq.com"
topics = ["VELOCITY", "FEED", "X_AXIS","Y_AXIS","Z_AXIS"]

# Create a client
client = mqtt.Client("Subscriber")
client.on_message = on_message
client.connect(mqttBroker)

# Subscribe to multiple topics
for topic in topics:
    client.subscribe(topic)

# Start the background loop
client.loop_start()

try:

    while True:
        time.sleep(1)
```

```
except KeyboardInterrupt:
    print("Stopping the MQTT client.")
    client.disconnect()
    client.loop_stop()
```

Output @ terminal of Publisher

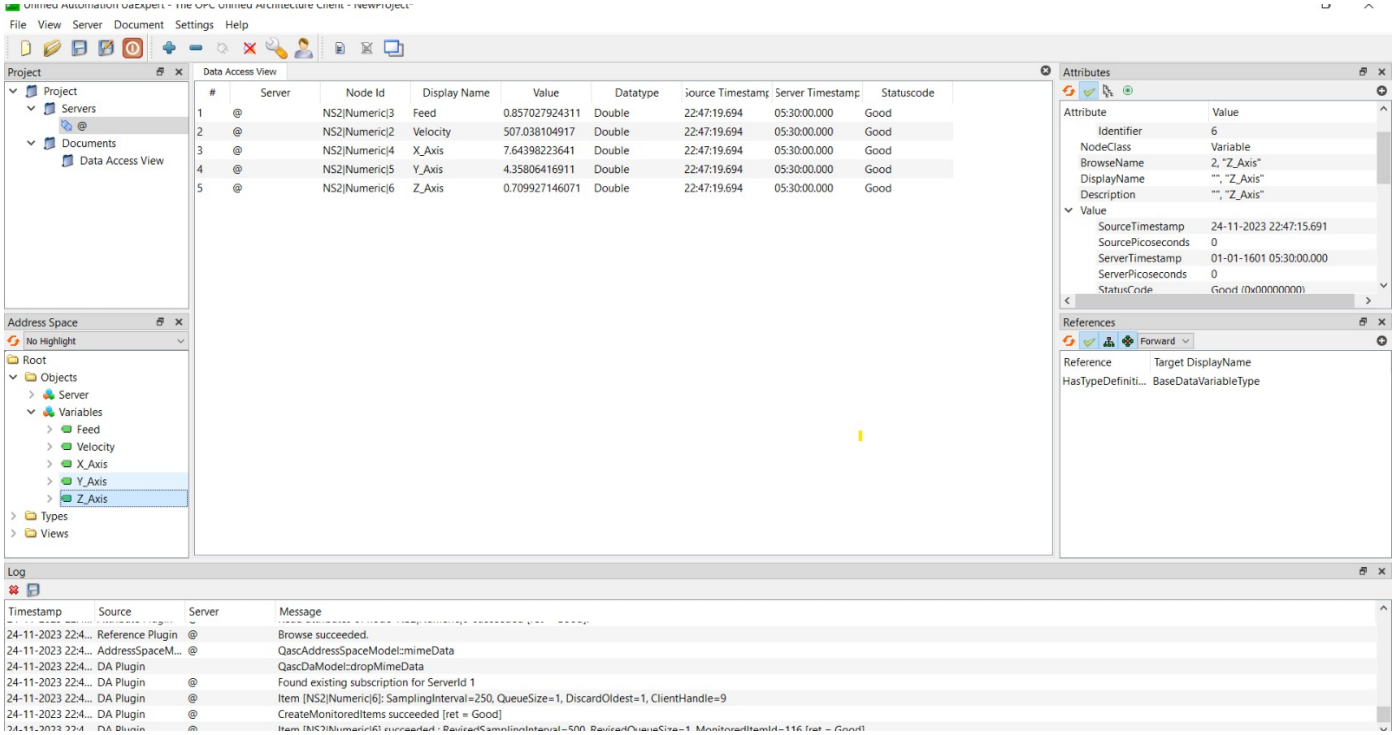
```
Received message on topic FEED: 0.395018767157412
Received message on topic Z_AXIS: 0.7692995313434461
Received message on topic VELOCITY: 504.66792194806953
Received message on topic Y_AXIS: 1.5196179714882174
Received message on topic X_AXIS: 8.748362344375005
Stopping the MQTT client.
```

OPCUA -

Server Code -

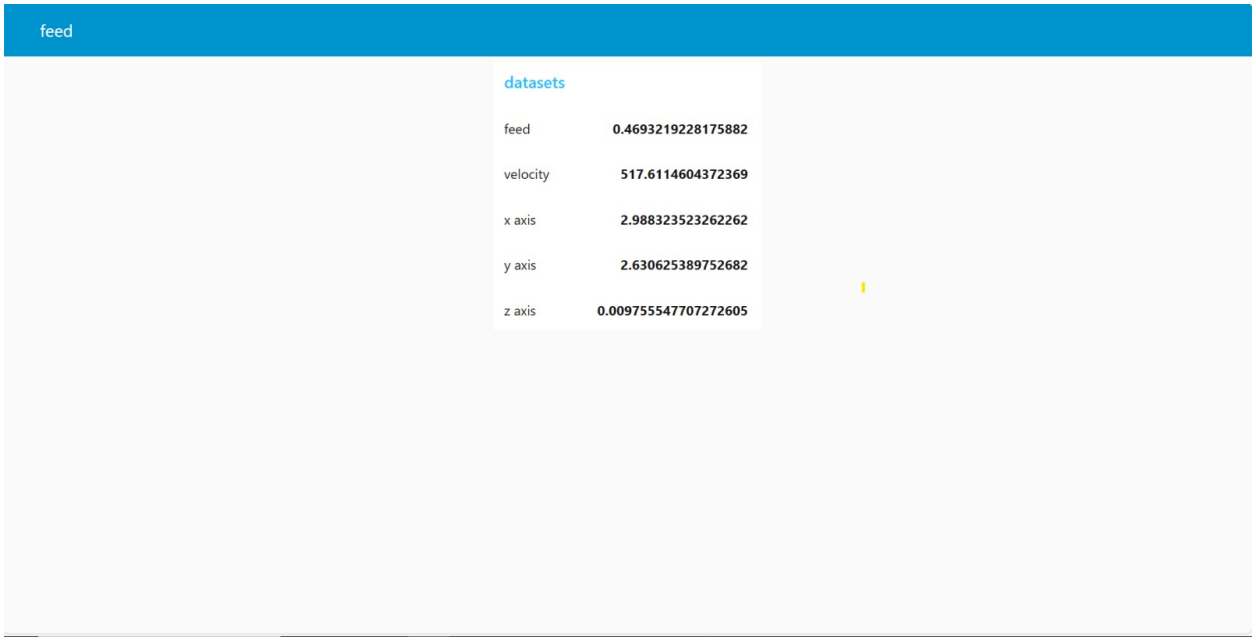
```
from opcua import Server
import random
import time

# Define server endpoint
server_endpoint = "opc.tcp://192.168.0.107:4840/freeopcua/server/"

# Create a server
server = Server()
server.set_endpoint(server_endpoint)

# Setup server namespace
uri = "http://example.org"
idx = server.register_namespace(uri)

# Create a new object 'Variables' under the root folder
variables_obj = server.nodes.objects.add_object(idx, "Variables")

# Add variables: velocity, feed, x_axis, y_axis, z_axis
velocity_var = variables_obj.add_variable(idx, "Velocity", 0.0)
feed_var = variables_obj.add_variable(idx, "Feed", 0.0)
x_axis_var = variables_obj.add_variable(idx, "X_Axis", 0.0)
y_axis_var = variables_obj.add_variable(idx, "Y_Axis", 0.0)
z_axis_var = variables_obj.add_variable(idx, "Z_Axis", 0.0)

# Set the variables as writable
velocity_var.set_writable()
feed_var.set_writable()
x_axis_var.set_writable()
y_axis_var.set_writable()
z_axis_var.set_writable()

# Start the server
server.start()
print(f"Server started at {server_endpoint}")

try:
    while True:
        # Update variable values with random values between 0 and 50
        velocity_var.set_value(random.uniform(500, 520))
        feed_var.set_value(random.uniform(0.1, 0.9))
        x_axis_var.set_value(random.uniform(0, 10))
        y_axis_var.set_value(random.uniform(0, 10))
        z_axis_var.set_value(random.uniform(0, 1))

        # Update values every 1 second (adjust as needed)
        time.sleep(1)

finally:
    # Stop the server on program termination
    server.stop()
    print("Server stopped")
```

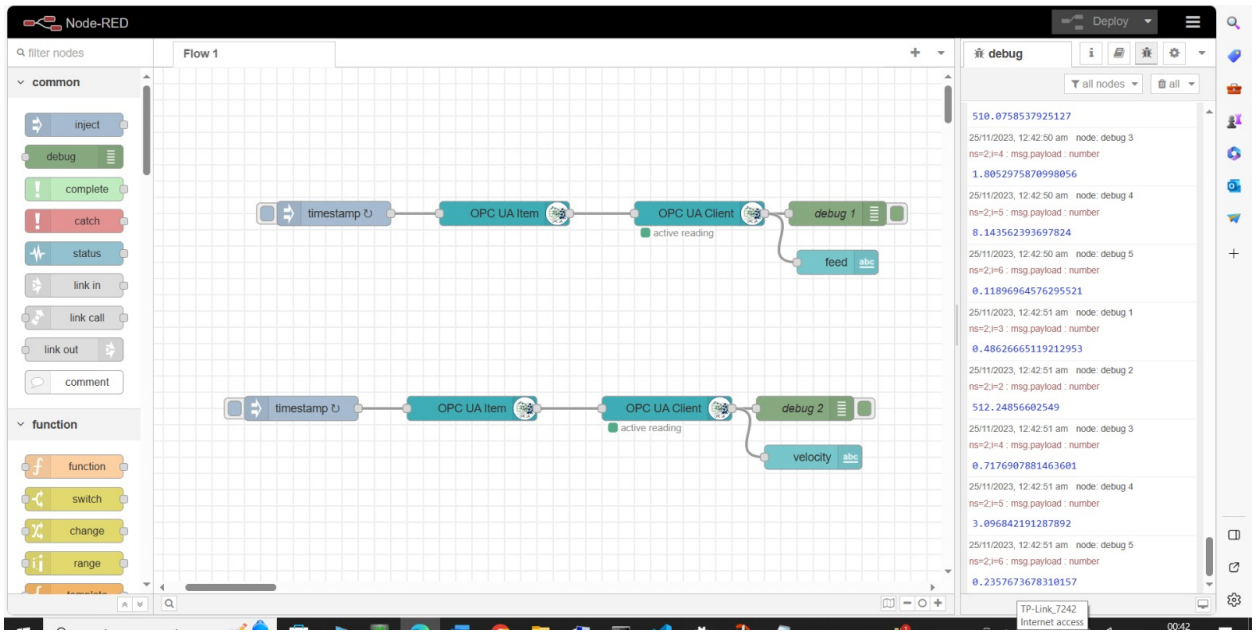
UAExpert - Client

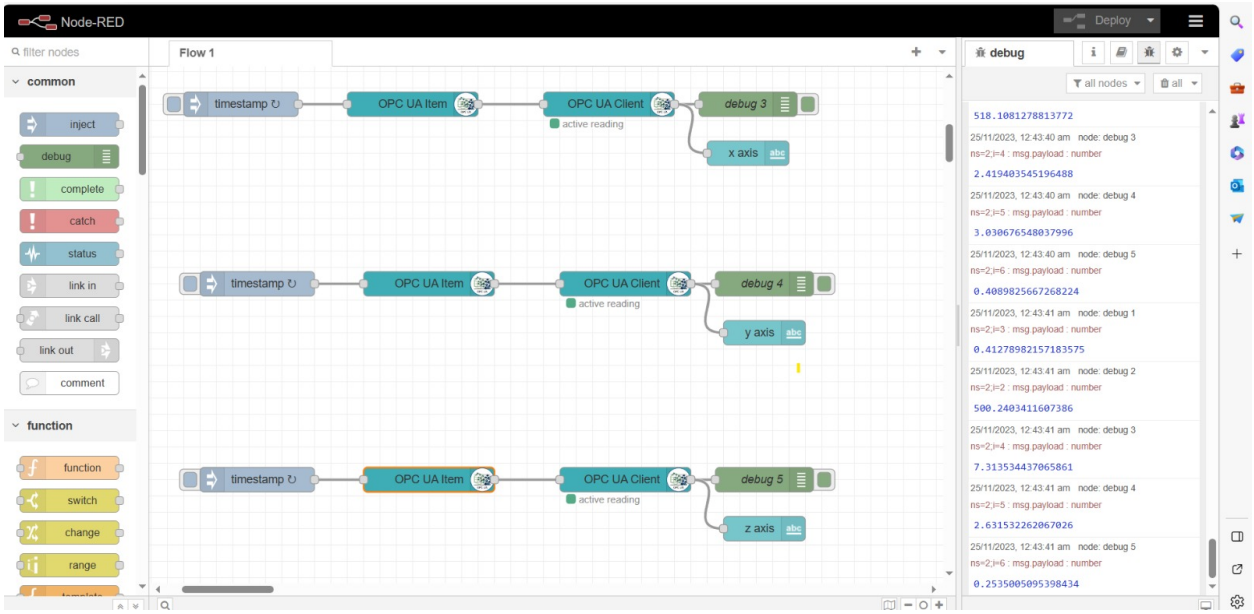


Python Code -
https://drive.google.com/drive/folders/1og8viuVw-emiAn54ZnmhC48kU1wdGZW4

Integration with NodeRed for OPCUA Client - Dashboard







NodeRED JSON FILES are uploaded in the OPCUA drive link.