



Getting Started with MQTT

Dr. Seno Adi Putra, S.Si., M.T

**Laboratorium *Enterprise Intelligent System*
Kelompok Keilmuan *Cybernetics*
Fakultas Rekayasa Industri – Universitas Telkom**

Agenda

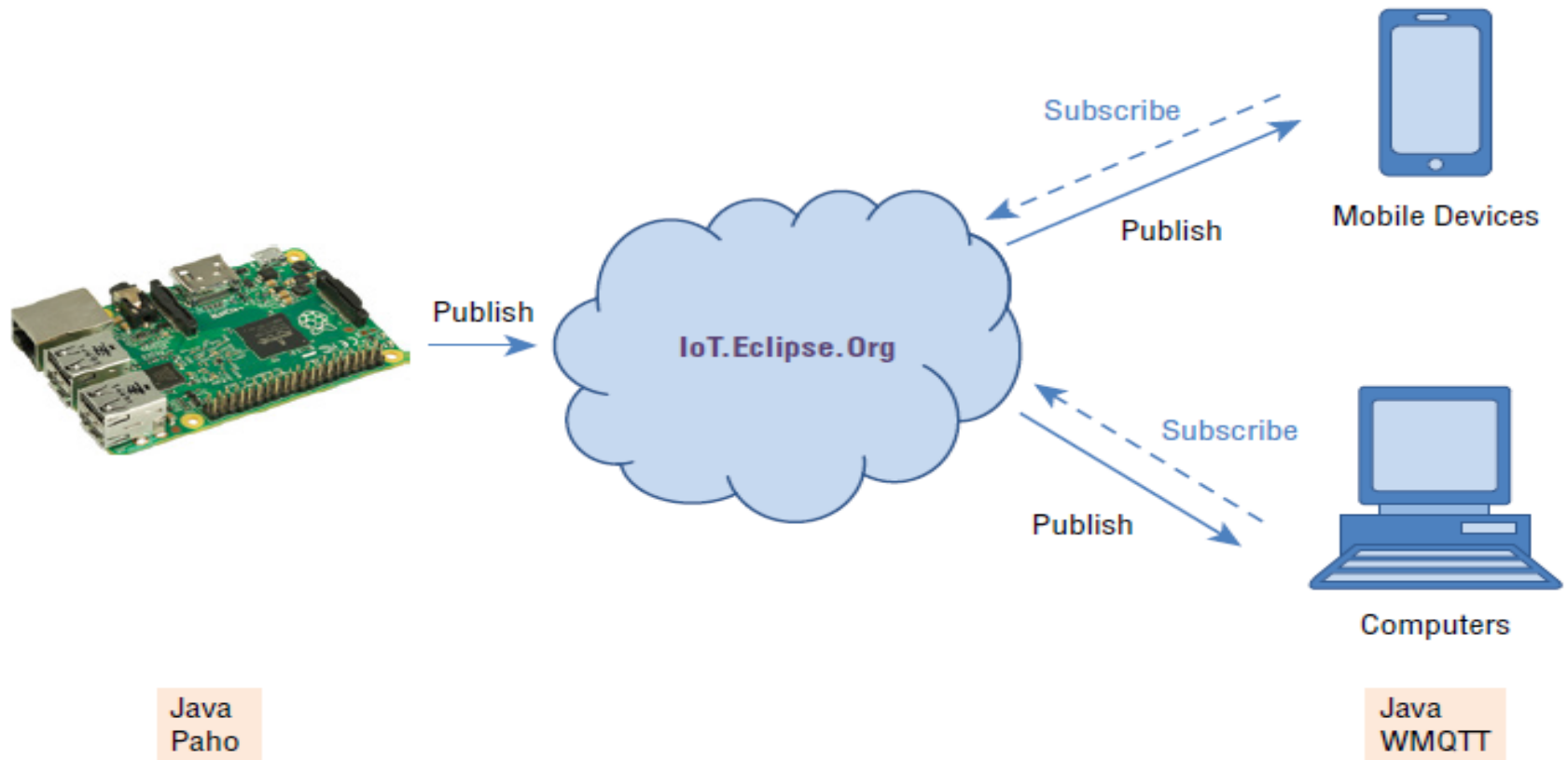
- MQTT Overview
- Software Preparation
- Example Code

Bagian I

MQTT Overview

MQTT

Message Queuing Telemetry Transport (MQTT) is an IoT protocol that allows IoT devices to publish messages on a topic through a broker, which in turn forwards the messages to all subscribers



Bagian II

Software Preparation

Downloading Softwares

- In this example, we are going to create a Java MQTT program on the Raspberry Pi. We will use **iop.eclipse.org** as the MQTT broker to publish some temperature and humidity information to a topic called **PX Temperature and Humidity**
- We will use a computer (or other mobile devices) to subscribe to the topic and receive the messages using the **IBM WMQTT A92** program
- To implement this example, you will need to download two pieces of software:
 - Eclipse Paho Java Client (on Raspberry Pi)
<https://www.eclipse.org/paho/clients/java/>
 - IBM's WMQTT IA92 Java utility (on computer)
<https://github.com/mqtt/mqtt.github.io/wiki/ia92>

Downloading Software

- For the Eclipse Paho Java Client there are different versions, and you can choose the version you prefer. Here, you will use MQTT V3, version 1.0.2, which you can download directly from this link:

<https://repo.eclipse.org/content/repositories/paho/org/eclipse/paho/org.eclipse.paho.client.mqttev3/1.0.2/org.eclipse.paho.client.mqttev3-1.0.2.jar>

- For IBM's WMQTT IA92, just go to the GitHub web site and follow the instructions to download and install it. The simplest way is to download it as a zipped file and just unzip it to your computer.

Bagian III

Example Code

Code Description

- In this case, we create a simple Java MQTT program, modified from the `MqttPublishSample` Java example on the Eclipse Paho Java Client web site.
 - The program uses `iot.eclipse.org:1883` as the broker
- The topic is `Temperature and Humidity`, and message content is `T=30C and RH=40%`. The QoS is level 2, which means exactly once.

Compiling The Code

```
import org.eclipse.paho.client.mqttv3.MqttClient;
import org.eclipse.paho.client.mqttv3.MqttConnectOptions;
import org.eclipse.paho.client.mqttv3.MqttException;
import org.eclipse.paho.client.mqttv3.MqttMessage;
import org.eclipse.paho.client.mqttv3.persist.MemoryPersistence;

public class MqttExample {
    public static void main(String[] args) {
        String topic          = "PX Temperature and Humidity";
        String content         = "T=30C and RH=40%";
        int qos                = 2;
        String broker          = "tcp://iot.eclipse.org:1883";
        String clientId        = "JavaMQTTExample";
        MemoryPersistence persistence = new MemoryPersistence();

        try {
            MqttClient sampleClient = new MqttClient(broker, clientId,
persistence);
            MqttConnectOptions connOpts = new MqttConnectOptions();
            connOpts.setCleanSession(true);
            System.out.println("Connecting to broker: "+broker);
```

Compiling The Code

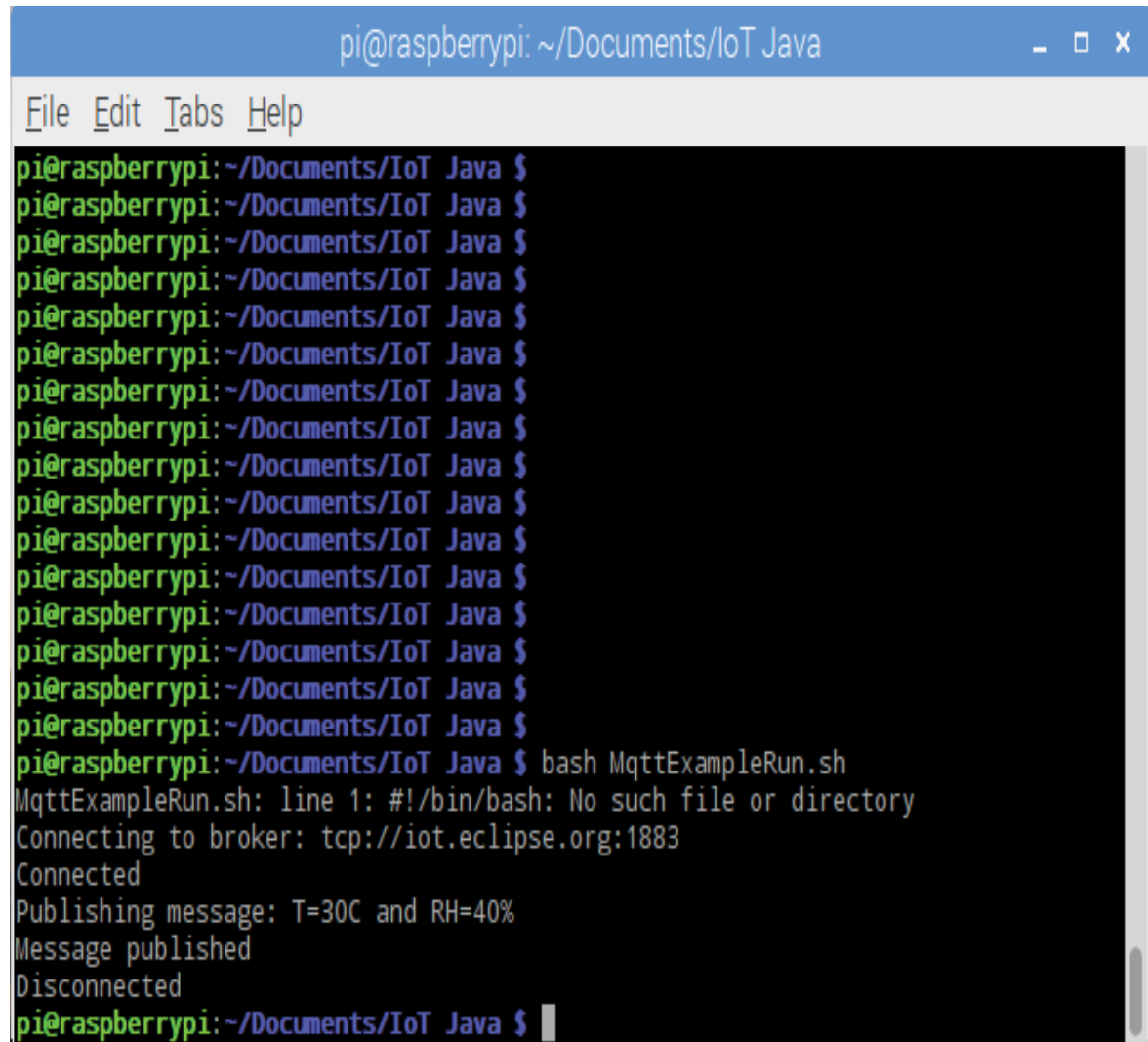
```
        sampleClient.connect(connOpts);
        System.out.println("Connected");
        System.out.println("Publishing message: "+content);
        MqttMessage message = new MqttMessage(content.getBytes());
        message.setQos(qos);
        sampleClient.publish(topic, message);
        System.out.println("Message published");
        sampleClient.disconnect();
        System.out.println("Disconnected");
        System.exit(0);
    } catch (MqttException me) {
        me.printStackTrace();
    }
}
}
```

Compiling The Code

Put the MQTT JAR file and the Java program in the same folder and then compile and execute the program using the following commands. Here it is important to include the MQTT JAR file in the classpath for both compilation and execution.

```
$ javac -classpath  
".:org.eclipse.paho.client.mqttv3-1.0.2.jar"  
MqttExample.java
```

```
$ sudo java -classpath  
".:org.eclipse.paho.client.mqttv3-1.0.2.jar"  
MqttExample
```

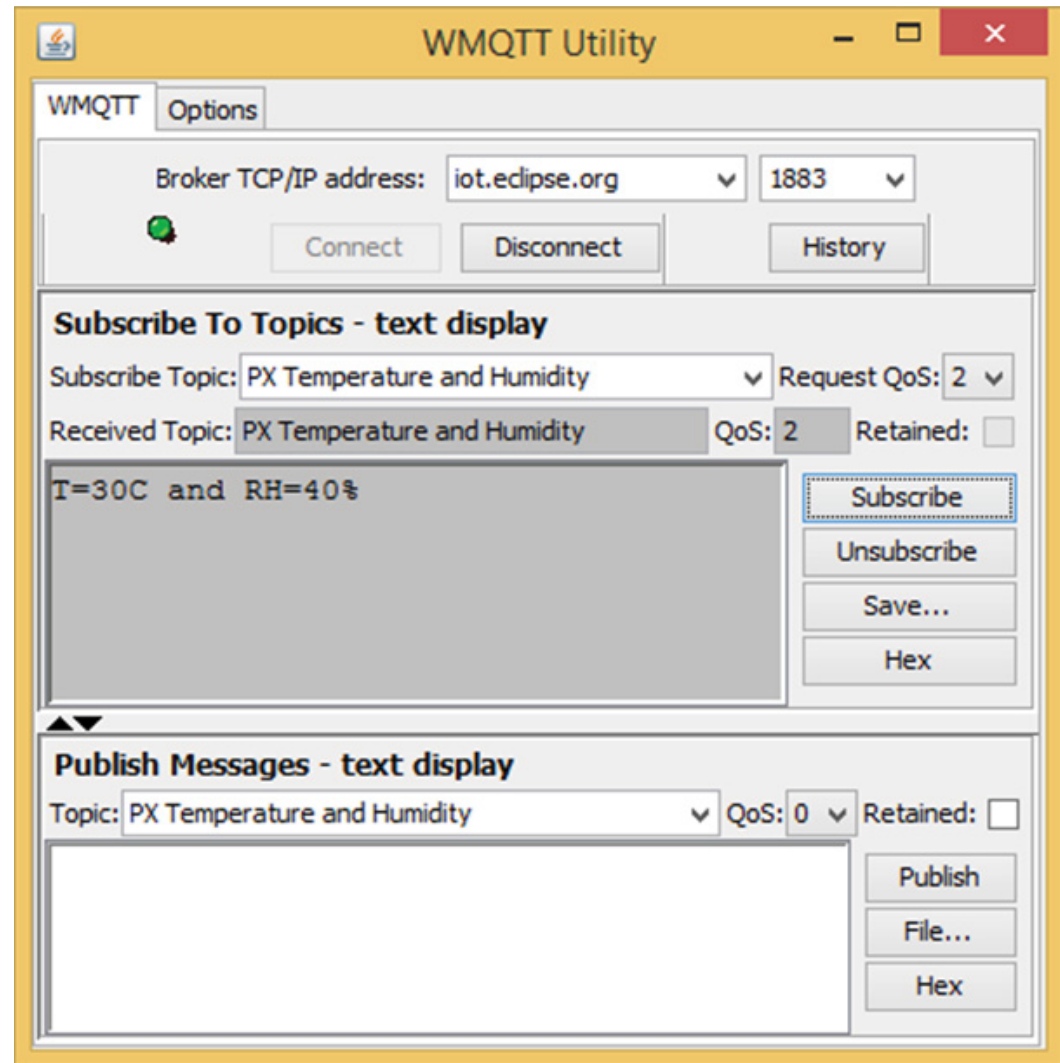


```
pi@raspberrypi: ~/Documents/IoT Java
File Edit Tabs Help
pi@raspberrypi:~/Documents/IoT Java $
pi@raspberrypi:~/Documents/IoT Java $
pi@raspberrypi:~/Documents/IoT Java $
pi@raspberrypi:~/Documents/IoT Java $
pi@raspberrypi:~/Documents/IoT Java $
pi@raspberrypi:~/Documents/IoT Java $
pi@raspberrypi:~/Documents/IoT Java $
pi@raspberrypi:~/Documents/IoT Java $
pi@raspberrypi:~/Documents/IoT Java $
pi@raspberrypi:~/Documents/IoT Java $
pi@raspberrypi:~/Documents/IoT Java $
pi@raspberrypi:~/Documents/IoT Java $
pi@raspberrypi:~/Documents/IoT Java $
pi@raspberrypi:~/Documents/IoT Java $
pi@raspberrypi:~/Documents/IoT Java $
pi@raspberrypi:~/Documents/IoT Java $
pi@raspberrypi:~/Documents/IoT Java $
pi@raspberrypi:~/Documents/IoT Java $ bash MqttExampleRun.sh
MqttExampleRun.sh: line 1: #!/bin/bash: No such file or directory
Connecting to broker: tcp://iot.eclipse.org:1883
Connected
Publishing message: T=30C and RH=40%
Message published
Disconnected
pi@raspberrypi:~/Documents/IoT Java $
```

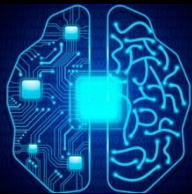
Viewing MQTT Message

To view the MQTT messages, go into IBM's WMQTT IA92 unzipped folder called `ia92`, and from the *J2SE subdirectory* find the `wmqttSample.jar` file and double-click it to run it.

- In case it does not run, you can also run it by typing `java -jar wmqttSample.jar` in the *Windows command window*.
- Make sure you connect to the `iot.eclipse.org` at port `1883` first, and subscribe to the `PX Temperature and Humidity` topic.
- You will be able to receive the message every time the Java program sends output from Raspberry Pi.



Terima Kasih



Laboratorium *Enterprise Intelligent System*

Kelompok Keilmuan *Cybernetics*

Fakultas Rekayasa Industri – Universitas Telkom