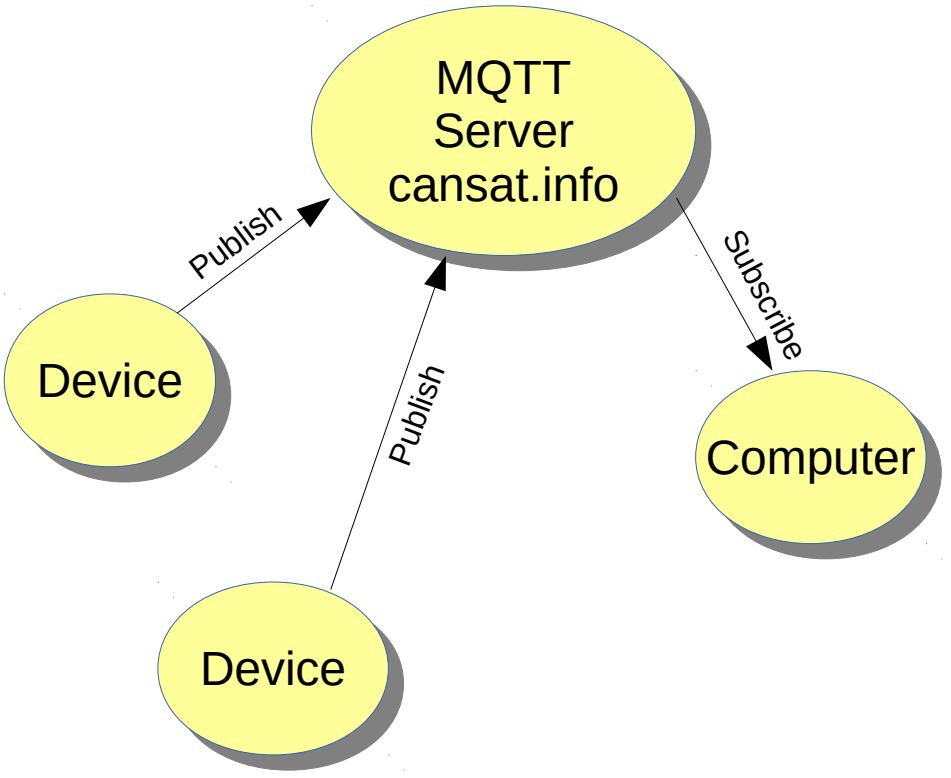

Using a Cloud Service

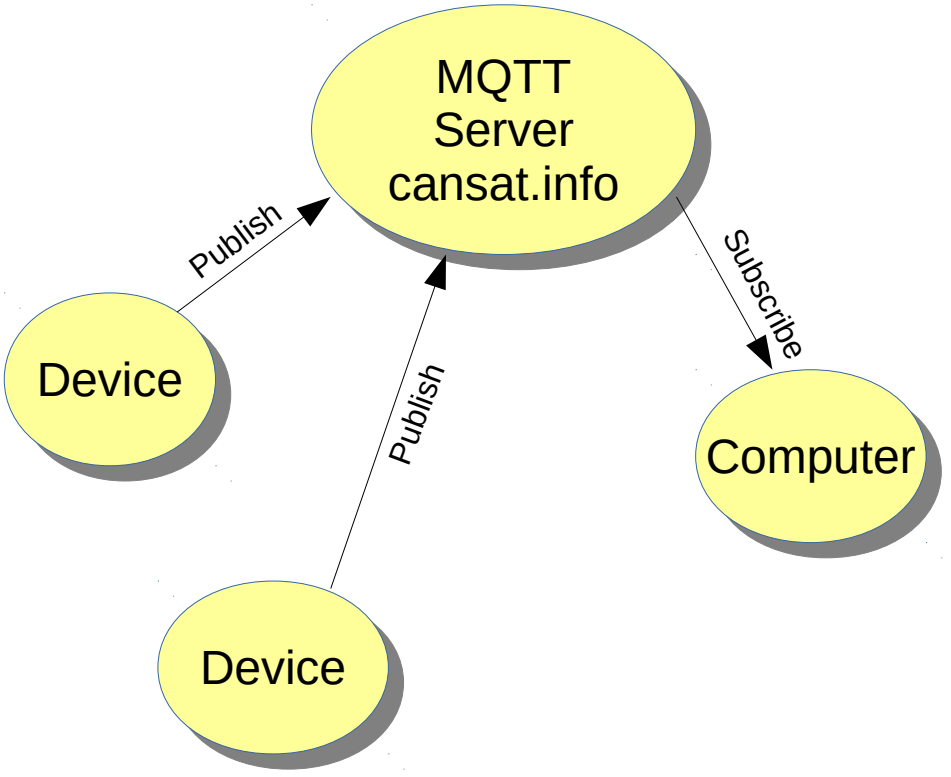
MQTT 2

- In this lesson, all the classes SLATEs will publish sensor data to an mqtt service and run a python program to collect data from all the sensors.
- The thermistor will be used to measure temperature. Each student will set up their slate to measure temperature and publish it once a minute for about 24 hours.



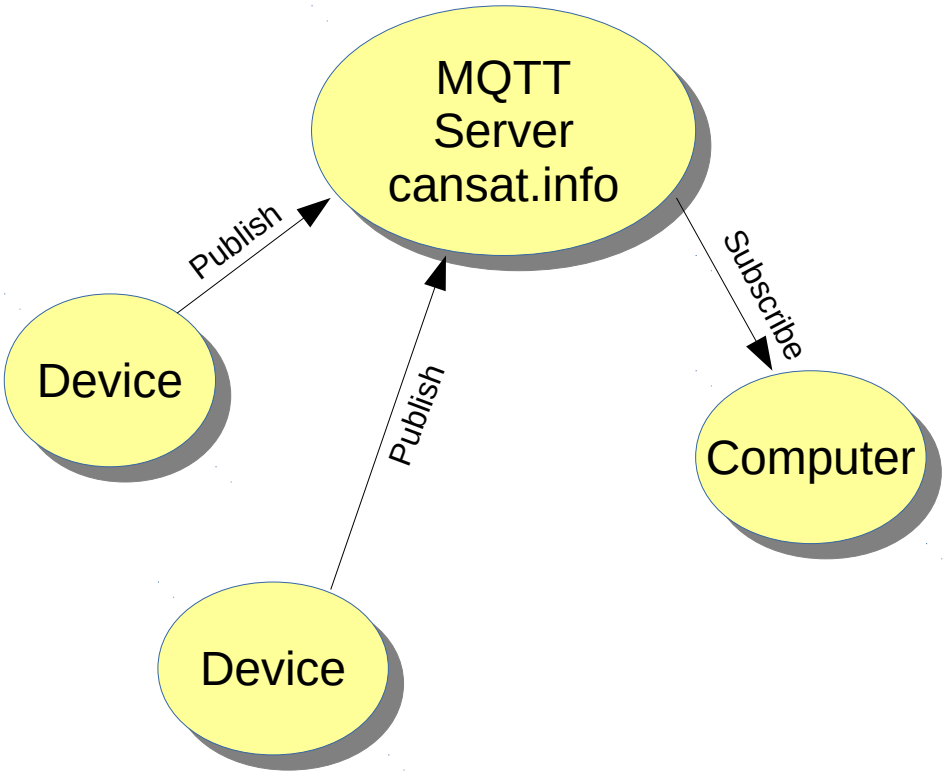
MQTT 2

- Each student shall send their data to the topic gmu/temperature.
- The format of the data shall be
  - last name,temperature
- Each student shall also run a python program to subscribe to the same topic and collect all data published. The data can then be sorted by name and plotted over time.



MQTT 2

- The mqtt service resides on a computer at **cansat.info**. A user name and password is required.
- The user name is **gmu2021**
- The password is **rockets41!**



MQTT SLATE Code

- This program will publish data to the mqtt server. The SLATE **setup()** function is similar to the previous program.
- Variable **ttag** is declared as an unsigned 32-bit integer. This will be used to determine the timing to publish the data.

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

WiFiClient esp;
PubSubClient client(esp);

uint32_t ttag;

void setup() {
  Serial.begin(115200);
  pinMode(13,OUTPUT);
  pinMode(15,OUTPUT);
  WiFi.mode(WIFI_STA);
  WiFi.begin("SSID","PASSWORD");
  while(WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
  randomSeed(micros());
  String clientID = "me" + String(random(0xffff));
  Serial.println();
  Serial.println(WiFi.localIP());
  client.setServer("cansat.info",1883);
  if(client.connect(clientID.c_str(),"gmu2021","rockets41!")) {
      Serial.println("Connected");
  }
}
```

MQTT SLATE Code

- Instead of connecting to the laptop mqtt server, a cloud service will be used called **cansat.info**.
- Next, the connection has two parameters added, the user name and password.
- The **subscribe()** function is not used in this program since it will only publish sensor data.

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

WiFiClient esp;
PubSubClient client(esp);

uint32_t ttag;

void setup() {
  Serial.begin(115200);
  pinMode(13,OUTPUT);
  pinMode(15,OUTPUT);
  WiFi.mode(WIFI_STA);
  WiFi.begin("SSID","PASSWORD");
  while(WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
  randomSeed(micros());
  String clientID = "me" + String(random(0xffff));
  Serial.println();
  Serial.println(WiFi.localIP());
  client.setServer("cansat.info",1883);

if(client.connect(clientID.c_str(),"gmu2021","rockets41!"))
{
      Serial.println("Connected");
  }
}
```

MQTT SLATE Code

- In the loop function, the **client.loop()** function is called to service the mqtt operations.
- Time is checked using **millis()** function which increments every millisecond. ttag is used to keep tack of when 60 seconds passes.
- If **millis()** is greater than ttag, the temperature is published to the mqtt server.

```
void loop() {
  char msg[64];
  client.loop();
  if(millis() > ttag) {
    ttag = millis() + 60000;
    Serial.println("Sending message");
    int a = analogRead(0);
    float v = (float)a / 1024;
    float t = (22.64 * v * v) - (119.89 * v) + 79.34;
    snprintf(msg,64,"lastname,%f",t);
    client.publish("gmu/temperature",msg);
  }
}
```

MQTT SLATE Code

- The reason a simple **delay()** function is not used to wait 60 seconds is that **client.loop()** has to be executed frequently. The function services and maintains the connection to the mqtt server. Also when completing the **loop()** function, the WiFi connection is serviced and maintain before **loop()** is executed again.

```
void loop() {
  char msg[64];
  client.loop();
  if(millis() > ttag) {
    ttag = millis() + 60000;
    Serial.println("Sending message");
    int a = analogRead(0);
    float v = (float)a / 1024;
    float t = (22.64 * v * v) - (119.89 * v) + 79.34;
    snprintf(msg,64,"lastname,%f",t);
    client.publish("gmu/temperature",msg);
  }
```

MQTT SLATE Code

- The time tag set the current **millis()** time plus 60000 for 60 seconds in the future.
- The analog voltage is measured and converted to a voltage.
- A text message is created with **snprintf()** that contains the last name and temperature.
- Lastly, the text message is published to the mqtt server.

```
void loop() {
  char msg[64];
  client.loop();
  if(millis() > ttag) {
    ttag = millis() + 60000;
    Serial.println("Sending message");
    int a = analogRead(0);
    float v = (float)a / 1024;
    float t = (22.64 * v * v) - (119.89 * v) + 79.34;
    snprintf(msg,64,"lastname,%f",t);
    client.publish("gmu/temperature",msg);
  }
```

MQTT Python Code

- The python program will subscribe and collect the data received. There are three functions called callback functions. These execute based on the event that occurred.
- The first one is **on_connect()**. It executes when the program successfully connects to the mqtt server. This function just displays the text **connected**.
- The second callback function is **on_message()**. It executes anytime a message from a subscribed topic is received. The topic is also included in case the python program subscribes to more than one topic.

```
import paho.mqtt.client as mqtt

def on_connect(client, userdata, flags, rc):
    print("Connected”)

def on_message(client, obj, msg):
    print(msg.topic + " " + str(msg.payload))

def on_subscribe(client, obj, mid, granted_qos):
    print("Subscribed: " + str(mid

mqttc = mqtt.Client()
mqttc.on_message = on_message
mqttc.on_connect = on_connect
mqttc.on_subscribe = on_subscribe

mqttc.username_pw_set("gmu2021", "rockets41!")

mqttc.connect("cansat.info",1883)
mqttc.subscribe("gmu/temperature")
rc = 0
while rc == 0:
    rc = mqttc.loop()
print("rc: " + str(rc))
```

MQTT Python Code

- The **on_subscribe()** function executes when the python program successfully subscribes to a topic.
- The client object is created and named **mqttc**.
- The callback functions are declared and assigned to the functions.

```
import paho.mqtt.client as mqtt

def on_connect(client, userdata, flags, rc):
    print("Connected”)

def on_message(client, obj, msg):
    print(msg.topic + " " + str(msg.payload))

def on_subscribe(client, obj, mid, granted_qos):
    print("Subscribed: " + str(mid

mqttc = mqtt.Client()
mqttc.on_message = on_message
mqttc.on_connect = on_connect
mqttc.on_subscribe = on_subscribe

mqttc.username_pw_set("gmu2021", "rockets41!")

mqttc.connect("cansat.info",1883)
mqttc.subscribe("gmu/temperature")
rc = 0
while rc == 0:
    rc = mqttc.loop()
print("rc: " + str(rc))
```

MQTT Python Code

- Next, the user name and password are set.
- The connection to the mqtt server is made.
- The topic is **gmu/temperature** is subscribed.
- Everything is now set up to collect temperature data.

```
import paho.mqtt.client as mqtt

def on_connect(client, userdata, flags, rc):
    print("Connected”)

def on_message(client, obj, msg):
    print(msg.topic + " " + str(msg.payload))

def on_subscribe(client, obj, mid, granted_qos):
    print("Subscribed: " + str(mid

mqttc = mqtt.Client()
mqttc.on_message = on_message
mqttc.on_connect = on_connect
mqttc.on_subscribe = on_subscribe

mqttc.username_pw_set("gmu2021", "rockets41!")

mqttc.connect("cansat.info",1883)
mqttc.subscribe("gmu/temperature")
rc = 0
while rc == 0:
    rc = mqttc.loop()
print("rc: " + str(rc))
```

MQTT Python Code

- Now, the python program goes into an infinite loop. In the infinite loop, **mqttc.loop()** is repeatedly called. This is to service any mqtt operation such as maintaining a connection to the mqtt server and checking for messages received. **mqttc.loop()** waits for any event to occur.
- Run the python program.
- Run the SLATE program.
- The python program will display the temperature measurement when it is received.

```
import paho.mqtt.client as mqtt

def on_connect(client, userdata, flags, rc):
    print("Connected”)

def on_message(client, obj, msg):
    print(msg.topic + " " + str(msg.payload))

def on_subscribe(client, obj, mid, granted_qos):
    print("Subscribed: " + str(mid))

mqttc = mqtt.Client()
mqttc.on_message = on_message
mqttc.on_connect = on_connect
mqttc.on_subscribe = on_subscribe

mqttc.username_pw_set("gmu2021", "rockets41!")

mqttc.connect("cansat.info",1883)
mqttc.subscribe("gmu/temperature")
rc = 0
while rc == 0:
    rc = mqttc.loop()
print("rc: " + str(rc))
```

MQTT Python Code

- To make the python more useful, the **on_message()** callback function will be modified to save the message to a file. Since more than one student may send data, the program will use the last name as the file name and store the temperature data from each student in their own file. It would also be helpful to timetag the data so all the data from all the students can be correlated.
- To get the current time, the python time module needs to be imported as shown.
- Only the top portion of the code is shown including **on_message()**. The rest of the code is still needed.

```
import paho.mqtt.client as mqtt
import time

def on_connect(client, userdata, flags, rc):
    print("Connected”)

def on_message(client, obj, msg):
    print(msg.topic + " " + str(msg.payload))
    message = msg.payload.decode('utf-8')
    message = message.split(',')
    now = time.time()
    fd = open(message[0] + ".csv","a")
    b = str(now) + "," + message[1] + "\n"
    fd.write(b)
    fd.close()
```

MQTT Python Code

- The message received is converted into a String.
- The message is split into a list with the contents separated by a comma.
- The current time is saved to variable **now**.
- Next, a file is opened for appending. The file name is the last name from the message with **.csv** added to the end. The **“a”** tells the open function to add to the file if it exists or create the file if it does not exist.

```
import paho.mqtt.client as mqtt
import time

def on_connect(client, userdata, flags, rc):
    print("Connected”)

def on_message(client, obj, msg):
    print(msg.topic + " " + str(msg.payload))
    message = msg.payload.decode('utf-8')
    message = message.split(',')
    now = time.time()
    fd = open(message[0] + ".csv","a")
    b = str(now) + "," + message[1] + "\n"
    fd.write(b)
    fd.close()
```

MQTT Python Code

- Now, a string is created combining the time and the temperature data separated by a comma and with a line feed added to the end.
- The data is saved to the file with **fd.write()**.
- The file is closed. This is done so that the next time a message is received, the correct file will be opened. Leaving a file open will cause an error.
- Run the python program and verify the file has been created and data is being added. The file will reside where the python program is located.

```
import paho.mqtt.client as mqtt
import time

def on_connect(client, userdata, flags, rc):
    print("Connected”)

def on_message(client, obj, msg):
    print(msg.topic + " " + str(msg.payload))
    message = msg.payload.decode('utf-8')
    message = message.split(',')
    now = time.time()
    fd = open(message[0] + ".csv","a")
    b = str(now) + "," + message[1] + "\n"
    fd.write(b)
    fd.close()
```