Introduction

MQTT is a publishing and subscription based messaging system. MQTT stands for Messaging Queuing Telemetry Transport. For an explanation of how it works, go to https://randomnerdtutorials.com/what-is-mqtt-and-how-it-works/

The competition judges will use MQTT to view team data received by their ground station in real time. An MQTT broker has been set up on a server using the mosquitto software. Teams will need to add code to their ground station software to connect to the mqtt broker and publish their data in the CSV format. Each team will receive a username and password to connect to the broker. The broker software is located here: https://mosquitto.org/

Implementation

The ground station software will operate as an MQTT client. The ground station software shall publish data in the CSV format in a single text line to the MQTT broker. The topic shall be **teams/xxxx** where the team inserts their team number in place of **xxxx**. When the ground station software saves a line of data to a file, add code there to publish to the broker.

Teams can test if their ground station is properly submitting their data by going to cansat.info/plot.html. At the bottom of the web page, enter your team number and click the button. This will connect the plotting page to the team topic.

MQTT is supported by many programming languages such as Python, C, C++, Go, Java and others. It is also supported by Matlab, Labview and Processing.

The container data and payload data are to be published as separate messages. Below is an example from a data file used to test the plotter web page.

1000,23:52:00,80,SP1,293,29.4,0,,,,,,,,,,
1000,23:52:01,81,C,F,R,N,422,25.6,8.8,23:52:01,37.2314,80.4264,422.6,7,RELEASE_1,73,0,CX-ON
1000,23:52:01,82,SP1,272.6,29.6,0,,,,,,,,,,
1000,23:52:02,83,C,F,R,N,415,25.7,8.8,23:52:02,37.2315,80.4265,418.2,7,RELEASE_1,74,0,CX-ON
1000,23:52:02,84,SP1,252.2,29.7,0,,,,,,,,,,
1000,23:52:03,85,C,F,R,N,408,25.7,8.8,23:52:03,37.2315,80.4265,410.4,7,RELEASE_1,75,0,CX-ON
1000,23:52:03,86,SP1,231.8,29.8,0,,,,,,,,,,
1000,23:52:04,87,C,F,R,N,399,25.8,8.8,23:52:04,37.2315,80.4265,402.2,7,RELEASE_1,76,0,CX-ON
1000,23:52:04,88,SP1,211.4,30,0,,,,,,,,,,
1000,23:52:05,89,C,F,R,R,395,25.8,8.8,23:52:05,37.2316,80.4266,395.3,7,RELEASE_2,77,0,CX-ON
1000,23:52:05,90,SP1,191,30.1,0,,,,,,,,,,
1000,23:52:05,91,SP2,395,28.1,0,,,,,,,,,,
1000,23:52:06,92,C,F,R,R,389,25.8,8.8,23:52:06,37.2316,80.4266,389.3,7,RELEASE_2,78,78,CX-ON
1000,23:52:06,93,SP1,170.6,30.2,0,,,,,,,,,,

Connection Information

The mqtt broker is located at cansat.info
The port number is 1883
Each team will receive a unique username and password.
The topic is teams/xxxx where xxxx is the team number.
Do not share usernames and passwords. The username is tied to the team topic and can only be used to access the team topic.

Python MQTT

For Python, it is suggested to use paho-mqtt. https://pypi.org/project/paho-mqtt/
An example code is shown in the appendix. This code was used to test the plot program by sending sample CSV data from a file.

Matlab MQTT Client

This is a link to matlab mqtt:
https://www.mathworks.com/help/thingspeak/use-desktop-mqtt-client-to-subscribe-to-channel-updates.html

They show the connection to thinkspeak. Change the connection profile to cansat.info. The broker port is 1883. For Client ID, click on the Generate button for a unique ID.

This MQTT interface for Matlab may work better:
https://github.com/gnotomista/mqtt_matlab_interface

Labview MQTT Client

https://github.com/LabVIEW-Open-Source/MQTT-Client
https://www.vipm.io/package/labview_open_source_project_lib_mqtt_client/

Processing MQTT Client

Go to the Sketch Menu, select Import Library and then select Add Library. A window will open with a list of available libraries. Scroll down and locate the MQTT library. Once installed, open the Examples by selecting the File menu and and selecting Examples. Go toward the bottom and select PublishSubscribe example. Change the function **Client.connect()** to :
Client.connect("mqtt://username:password@cansat.info");

A copy of the example is in Appendix B.

C++ MQTT Client

The source code for the mqtt client can be found here: https://github.com/eclipse/paho.mqtt.cpp

Summary

The competition staff cannot provide any technical support for implementing mqtt. The staff can verify team username and password. When a username and password is created, the staff will verify that the system has been updated properly by testing each team account.

Appendix A

Sample Python Code

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

# Define event callbacks
def on_connect(client, userdata, flags, rc):
        print("rc: " + str(rc))

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

def on_publish(client, obj, mid):
        print("mid: " + str(mid))

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

def on_log(client, obj, level, string):
        print(string)

# setup mqtt client call backs
mqttc = mqtt.Client()
# Assign event callbacks
mqttc.on_message = on_message
mqttc.on_connect = on_connect
mqttc.on_publish = on_publish
mqttc.on_subscribe = on_subscribe

# Uncomment to enable debug messages
mqttc.on_log = on_log


topic = 'teams/1010'                    # team number
# Connect
mqttc.username_pw_set("t1010", "t1010pass")             # made up username and password
for mqtt
#mqttc.connect(url.hostname, url.port)              # establish connection
mqttc.connect("cansat.info",1883)
```

```
# Publish a message
fd = open("./cansat1.csv","r")                          # open csv file
dat = fd.read()                 # read in whole file
fd.close()

dat = dat.split('\n')           # split lines
while 1:
   for i in dat:                # go through all the lines in the file
         b = i.split(',')       # split line to locate element 3
         if len(b) > 1:
               if b[3] == 'C':          # check if container data
                     time.sleep(1)      # insert 1 second interval unless payload adata
               mqttc.publish(topic, i)  # send the line of data
```

Appendix B

Processing Example

```
// by Joël Gähwiler
// https://github.com/256dpi/processing-mqtt

import mqtt.*;

MQTTClient client;

void setup() {
  client = new MQTTClient(this);
  client.connect("mqtt://t1000:t1000pass@cansat.info");
}

void draw() {}

void keyPressed() {
  client.publish("teams/1000", "world");
}

void clientConnected() {
  println("client connected");

//  client.subscribe("/hello");
}

void messageReceived(String topic, byte[] payload) {
  println("new message: " + topic + " - " + new String(payload));
}

void connectionLost() {
  println("connection lost");
}
```