Unity and Arduino - MQTT connection

Description of how to connect Unity and an Arduino to an MQTT broker.

- MQTT-Arduino-and-Unity

MQTT-Arduino-and-Unity

LINK TO GITLAB REPOSITORY

Table of Contents

Project description

This repository contains information on how to connect an Arduino to Unity via a MQTT broker. Both the Arduino and the Unity project are able to Subscribe and Publish to multiple MQTT topics. It is important to note that the provided code will need to be changed according to your project specifications. Additionally, the Unity project also has graph visualisation using Xcharts.

Unity overview

This section provides comprehensive information on using the MQTT client in Unity, along with instructions on how to display values on a graph within Unity. The MQTT client allows you to subscribe to MQTT topics, receive and publish messages, and integrate them into your Unity project.

The Unity project has 2 dependencies. These dependencies are already present within the project. When using the prefabs from this project these dependencies automatically get imported into your new project.

☐ MQTTnet 3.1.0 library; responsible for creating a MQTT client within Unity.

☐ Xcharts; responsible for the visualisation of the data with Unity.

As mentioned above, this repository also contains 2 prefabs that can be directly imported into your own Unity project. These prefabs can than be adjusted to your own needs.

☐ MQTT; contains the MQTTSubscriber and MQTTPublisher script. The MQTTSubscriber cannot be deleted from the prefab, as this contains the contains all relevant code that is needed for a connection with the MQTT broker. The MQTTPublisher script is only there for demonstation purposes (as will be shown below) and can be deleted from the prefab.

☐ ChartUpdater; This contains code that reads the data provided by the MQTT prefab and publishes it to the chart. This is not neccesary for a MQTT connection, but is used here for the demonstration.

Features

- MQTT subscription and publication within Unity.
- Debugging options to view messages in the Unity console.
- Support for MQTT brokers on specified TCP servers and ports.
- Optional use of credentials and encryption.
- Easy integration with Unity game objects and C# scripts.

Unity project walktrough

This Unity MQTT client allows you to easily connect to MQTT brokers, subscribe to topics, and publish messages. The MQTT publisher and subscriber can be attached to the same GameObject or separated into two GameObjects, offering flexibility and customization based on your project's requirements.

Getting Started

Configuring the MQTT subscriber

To set up the MQTT subscriber, follow these steps:

1. Attach the MQTT subscriber script to a GameObject in your Unity project.
2. Toggle the "Enable connection" option to establish a connection.
3. Optionally, enable "Toggle Debugging" to print received and published messages to the Unity console.
4. Configure the MQTT broker address (TCP server) and port according to your MQTT host settings.
5. If necessary, provide credentials and enable encryption (e.g., for port 8883).

6. Specify the "Topic Path" as the common path to access the topics you want to subscribe to.
7. Populate the "Topics" list with the variable names of the topics you wish to subscribe to.

Configuring the MQTT Publisher

The MQTT publisher settings are similar to those of the subscriber:

1. Attach the MQTT publisher script to a GameObject in your Unity project.
2. Drag and drop the MQTT subscriber script to MQTT subscriber field.
3. Specify the "Topic Path" as needed. This can be different from the subscriber's topic path.
4. Specify the "Topic" with the variable name of the topic you wish to publish to.
5. The "Data Reading Delay" is an example value that will be published to the MQTT broker on startup. Change this as you please.

MQTT example

Let's consider an example where we connect to the public mosquitto server on port 1883, which doesn't require credentials or encryption. We will subscribe to the topic "DTlab/Subscribe/Humidity" and publish to "DTlab/Publish/DataInterval". See the iamge below for all settings. To begin, disable the "ChartUpdater" object in the Unity hierarchy and enable debugging to verify the subscription and publishing process.

MQTTsetup Image not found or type unknown

You can confirm successful publishing in Unity's console immediatly. To verify subscription, you can use MQTT Explorer, a simplified MQTT client. Publish a value (e.g., 54) to the same topic you are subscribed to ("DTlab/Subscribe/Humidity"). As shown in the console, Unity receives the message correctly.

MQTTConsole Image not found or type unknown

The MQTT Subscriber script automatically creates a dictionary for accessing received messages from other Unity GameObjects and C# scripts. For an example, refer to the "ChartUpdater" object and the "AddToChart" script.

Chart example

We won't delve into full script details here, but let's briefly explain the functionality of the "AddToChart" script.

In this example, we use the same MQTT setup as before and configure the "AddToChart" script as follows:

Chart-Updater-Settings Image not found or type unknown

- The "Humidity" line chart is a child of the canvas GameObject.

- "Max Chart Size" indicates the number of data points the chart can display.
- "Data Interval" is automatically obtained from the MQTT Publisher GameObject.
- The "ChartUpdater" reads values from the MQTT topic list and adds them to the chart in the specified order. This occurs at intervals defined by the "Data Interval" variable.

The result is a dynamic chart displaying MQTT data received within Unity. the example below displays the received value 54 in teh chart every half a second.

Humidity-chart Image not found or type unknown

Arduino overview