Robotic arm control using MQTT protocol

V. Mautner[1], D. Olenišák[1], and M. Jurák[1]

[1]Střední odborné učiliště elektrotechnické Plzeň, Czech Republik

E-mail: vojtulam@email.cz, kdansubs@gmail.com, jurak@souepl.cz

Abstract— The aim of this work is to create a simple system that will serve mainly as a teaching tool for demonstrating communication in a simple IoT system. The proposed system should allow easily add different devices to the system and establish communication between them and create individual communication commands between devices The proposed system uses the MQTT protocol, which is widespread in most IoT systems. This protocol is simple and therefore easy to implement even on microcontrollers that can be connected to the system.

Keywords— robotic arm control, MQTT protocol, OpenMV camera,

1. INTRODUCTION

MQTT (Message Queuing Telemetry Transport) is a basic protocol used in IoT systems. The advantages of the protocol lie in its simplicity and simple implementation (it can be easily implemented on microcontrollers). The aim of our work was to create a simple educational IoT system that would allow to understand the basic MQTT communication. We chose a robotic arm supplemented by a computer vision module as the basic device of the whole system. This arm is then remotely controlled by other system devices (microcontroller, mobile phone etc.). All devices communicate with each other via MQTT messages and therefore it does not matter in which programming language the device control application is created. It is possible to easily add additional devices to the system and define new MQTT messages for them. This paper describes a simple IoT system in which a robotic arm can be remotely controlled. The structure of the article is as follows. Chapter two deals with the basic technical implementation of the whole system and a description of individual components, Chapter three describes the MQTT protocol, and the format of messages used in the system, Chapter Four describes the results of our work and topics that we would like to implement in the future. Chapter 5 shows the possibilities of practical use of the proposed system

2. TECHNICAL REALIZATION

The basis of the whole system is a commercially produced robotic arm uArm Swift Pro, which is connected to the control computer (Fig. 1). A computer vision module is attached to the arm to scan the workspace. The robotic arm can be controlled from various devices connected to the system via the MQTT protocol.

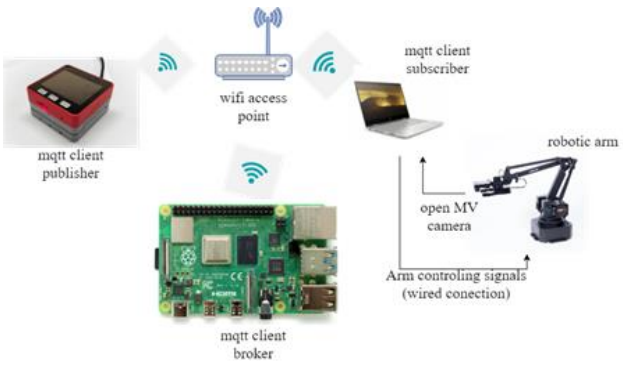

Figure 1: Robotic hand control system wiring diagram

One of the devices that can be used to control the robotic arm is the M5Stack module (Fig. 2a). It is a microcontroller based on ESP32, to which a display is connected. The microcontroller has the option of connecting to WIFI and it is possible to create an MQTT communication protocol on it. The M5stack module is programmed in the micro-Python programming language. In our case, the module serves as an MQTT client (sends commands to control the robotic arm and camera and receives information from camera module).

a) b)

Figure 2: a) M5Stack module b) OpenMVH7 Plus camera

Another module used in the system is the computer vision module based on the OpenMV H7 Plus camera (Figure 2b) [1] (a simple OV5640 image sensor with 2592x1944 pixels connected to a relatively powerful microcontroller STM32H743II ARM Cortex M7 running at 480 MHz). The camera can be connected to the MQTT network via a suitable WiFi module [2] and can process commands from a broker. OpenMV camera is programmable in micro-Python, which implements some advanced computer vision features that allow you to easily create the necessary applications. In our system, OpenMV is involved as an MQTT client (publisher and subscriber).

The Python programming language was used to create the application for controlling the robotic arm.

3. MQTT PROTOCOL

The MQTT protocol is the most used protocol in IoT systems (Fig. 3). The basic device of the whole communication is the MQTT server called the so-called broker, which receives messages from clients of publishers (Publishers), sorts them and passes them to clients to subscribers (Subscribers). Messages are sorted and forwarded based on topic, i.e., the publisher sends a message with a given topic to the intermediary, who receives it, and sends it to subscribers who are subscribed to the given topic.

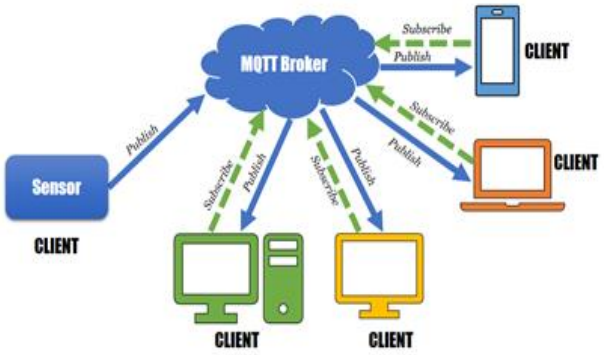

Figure 3: MQTT communication network scheme

In our system, the Mosquitto message server running at the Raspberry Pi microcomputer was used. M5Stack module, a computer vision module and an Android mobile phone were used as clients. Messages in the MQTT protocol consist of two parts – topic (it can be text string) and own message (it can be a text string, byte array, image, etc.) Clients - subscribers join the subscription of the given topic

In our case, we have used 3 topics so far:

- **RAC** (Robotic Arm Control) - a control computer (PC) is connected to the subscription of this topic, to which a robotic arm is connected, the messages control the movement of the arm.

  The following text strings are used as the messages, to inform the control computer that it should perform the elementary movement of the arm in individual directions (see Figure 4) or that it should lift or place the object:

  "**left**", "**right**", "**up**", "**down**", "**fwd**", "**bwd**", "**set_pos r fi z**", "**pump on**", "**pump off**"

  The strings "pump on", "pump off" are used to hold the object and to store it (we used an air vacuum compressor, located at the end of the arm)

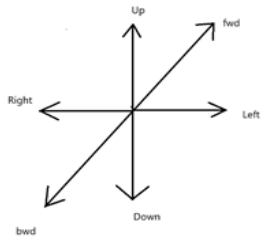

Figure 4: basic directions of robotic arm movement

- **CAM** (Camera) - the camera is attached to the subscription of this topic; the messages contain a question whether the object of the given color is in the working field of the arm. The following strings are used:

  "**red**", "**green**", "**yellow**"

- **ESP** - the M5Stack microcontroller is connected to this topic, the messages are used to inform the microcontroller whether the object was found or not and in what position. The following strings are used

  "**find_obj x y** ", "**not found**"

It is possible to add other messages and topics to the system as needed (e.g., to find other objects with the camera, etc.)

4. RESULTS AN FUTURE WORK

Figure 5 shows a system for remote control of a robotic arm. So far, we have implemented the following functions:

- Creating MQTT communication between devices.
- A control application in Python language was created for the robotic arm.
- An application in micro-Python language that finds an object of a given color has been created for the OpenMV camera.
- An application in micro-Python which allows you to build a three-color tower from cubes was created on the M5stack module,
- An application (in Java) was created on an Android mobile phone, which enables control of the robotic arm in basic directions and enables the transfer of objects.

In our future work, we would like to implement one of the following points:

- Image transfer of the captured scene to the LCD display of M5stack, PC or on a mobile phone so that it is possible to monitor what objects are in the camera workspace,
- use of brainwaves to control the robotic arm - you need to use a suitable EEG headband, which was not yet available,
- using the Leap motion controller (a device used in VR) to control robotic arm by more complex gestures.

Figure 5: Robotic Arm control system

5. CONCLUSION

A simple educational system showing the possibilities of the MQTT protocol, was created, However, the resulting system could find other practical uses in the following areas:

- Neuro-rehabilitation - instead of a robotic arm, a rehabilitation robot would be used, in which a rehabilitated arm would be hung.
- Industrial applications such as search for an object and transfer / transport it to a specified location - in this case, a mobile robotic arm would be used, which would allow searching for a specified object (e.g., a package of a given color, with a corresponding barcode, etc.) and transfer it to the destination position.

REFERENCES

[1] OpenMV camera H7 description. Accessed March 03, 2022. [Online]. Available: https://openmv.io/products/openmv-cam-h7

[2] WiFi shield for OpenMV camera description. Accessed March 03, 2022. [Online]. Available: https://openmv.io/products/wifi-shield-1

[3] OpenMV MicroPython documentation 1.15. Accessed March 03, 2022. [Online]. Available: https://docs.openmv.io/

[4] uArm-Python-SDK Manual, Accessed March 03, 2022. [Online]. Available: https://github.com/uArm-Developer/uArm-Python-SDK

[5] uArm-SwiftPro-User-Manual Accessed March 03, 2022. [Online]. Available: https://cdn.shopify.com/s/files/1/0012/6979/2886/files/uArm_Swift_Pro_User_Manual-V1.1.23.pdf?v=1615280407

[6] M5stack Fire IoT development Kit Manual Accessed March 03, 2022. [Online]. Available: https://m5stack.hackster.io/products/m5stack-fire-iot-development-kit-psram-2-0