Which Application Layer Protocol Is Best Suited for Networked Monitoring of Beehives? A Comparison of HTTP And MQTT in IoT

Langen, Jan*; Sahler, Kerstin*; Brell, Claus+

*Hochschule Niederrhein University of Applied Science, +clabremo Research Department

Abstract: Today, the Internet of Things (IoT) is here to stay. In addition to various application areas, IoT is useful in the context of beekeeping, especially for digital and networked monitoring of hives. For practical implementation, different application layer protocols can be chosen. But which protocol is more suitable in the context of beekeeping? HTTP (Hypertext Transfer Protocol) or MQTT (Message Queuing Telemetry Transport)? This is one of the questions addressed in the Biene40 project, which focuses on networked sensors. In this paper, we derive the advantages and disadvantages of each protocol based on a literature review and the development of two IoT-based prototypes that communicate via either HTTP or MQTT. Our research show that the message overhead of the MQTT protocol is lower than that of HTTP. However, the connection cost of MQTT is quite high, especially at the beginning of a connection. Therefore, the choice of protocol depends on the specific use case. If only a few messages are transmitted at long intervals, HTTP is more resource efficient. With respect to the developed prototypes and the application context of beehive monitoring, HTTP is the preferred choice.

Keywords: Internet of Things (IoT); Hypertext Transfer Protocol (HTTP); Message Queue Telemetry Transport (MQTT); Microcontroller; ESP8266; wifi plug; honey bee; beekeeping; beehive monitoring

Citation: Langen, Jan; Sahler, Kerstin; Brell, Claus (2024) Which Application Layer Protocol Is Best Suited for Networked Monitoring of Beehives? A Comparison of HTTP And MQTT in IoT. In: Working Reports in the Biene40 Project.

1 Introduction

In addition to cattle and pigs, the honey bee is another important animal in food production. About 80% of flowering plants in Germany are pollinated by insects. A large part of this pollination is done by the honey bee. The value of honeybee pollination is estimated at 2 billion euros in Germany and 70 billion dollars worldwide. (DEUTSCHER IMKERBUND E.V., n.d.-a) The influence of bees on the biodiversity of cultivated and wild plants is considerable and their role in stabilising the natural ecological cycle is of great importance. (DEUTSCHER IMKERBUND E.V., n.d.-b)

To protect honey bees and other pollinating insects, the German Federal Ministry of Food and Agriculture (BMEL) is currently funding 16 collaborative projects, which are grouped together under the networking and transfer measure Beenovation. (BEENOVATION, n.d.) The Biene40 research project is one of these collaborative projects and is being carried out by the Institute for Business Process Management and IT (GEMIT, https://hs-niederrhein.com/gemit) at the Niederrhein University of Applied Sciences.

The aim of the Biene40 project is to develop intelligent sensor technology that will enable beekeepers to remotely monitor honey bee colonies and their environment, without the need for the beekeeper to be physically near the hive or to open it and disturb the bees inside. The sensor technology is intended to increase the vitality of the honey bee colony and positively influence its pollination performance. (BRELL, n.d.-a)

One of the results of the project is the realisation that cables on and in the hive interfere with the beekeeper's way of working. As a result, there are different approaches to wirelessly transfer data from the hive using Internet of Things (IoT) communication technologies. (BRELL, 2022) Currently, the Biene40 project uses HTTP (Hypertext Transfer Protocol) as the application layer network

protocol for data transfer. The MQTT (Message Queuing Telemetry Transport) protocol is an alternative to this approach. It should be investigated whether MQTT can serve as a viable alternative to HTTP within the Biene40 project.

The following research questions (RQ) are derived from this task:

RQ1: What is MQTT and how do the characteristics of MQTT and HTTP differ?

RQ2: Which application domains are more suitable for MQTT and which for HTTP?

RQ3: What are the advantages and disadvantages of the practical implementation?

RQ4: Which of the two implementation approaches is more suitable for a practical application by technophile beekeepers?

This paper aims to compare the advantages and disadvantages of the HTTP and MQTT network protocols for the Biene40 project. To gain a comprehensive understanding of the existing knowledge related to MQTT., we performed a qualitative literature review on the characteristics of MQTT and HTTP. In addition, we developed two prototypes with corresponding manuals for technophile beekeepers in the project. Based on the literature and the prototypes, we derived the advantages and disadvantages of MQTT and HTTP and answered the research questions.

This paper is organized as follows: Section 2 deals with the related work. In Section 3, the proceedings to perform the literature review and the foundation for developing the prototypes are described. Resulting from that, Section 4 presents the findings of the literature review and the development of the prototypes. In Section 5, the results are discussed and the advantages and disadvantages of the protocols for Biene40 are derived. Section 6 serves as the conclusion of this paper.

2 Related Work

WURM AND BRELL (2022) performed a systematic literature review on the digitalization of beekeeping. Based on their work, we identified four publications that already used MQTT as a network protocol for controlling and monitoring beehives:

MAHAMUD ET AL. (2019) use MQTT in their work to monitor the honey and wax production in the hive. For this, a load cell, temperature sensors, humidity sensors, a sensor for detecting gas, and a sound module are connected to an ESP8266 microcontroller. The system uses sound data to evaluate the bees' work performance. It is usable via a mobile application and designed to be user-friendly. Operated with a pair of 2.5V batteries, it has low power consumption as well as it is cost-effective.

YUSOF ET AL. (2019) also gather temperature, humidity, weight, and air quality data through their system for monitoring the health of stingless bees. The collected data is uploaded to a cloud via a wireless connection, forwarded using the MQTT protocol, and then visualized graphically. The results can be accessed through a website.

MACHHAMER ET AL. (2020) describe the setup of a monitoring system for beehives. In their work, temperature, humidity, air pressure, and weight data are used to detect anomalies and vandalism of the beehive. In addition, the flight activity of the bees is monitored using a motion sensor. The system is based on an ESP8266 and uses Node-RED's MQTT interface to visualize the data.

In the fourth publication, ZABASTA ET AL. (2019) use temperature data collected from both inside and outside the beehive, along with humidity and weight measurements, to monitor the hive's health status. In addition, video recordings and meteorological data are used for analysis. Like MACHHAMER ET AL. (2020), the authors leverage MQTT and Node-RED for data visualization. The data transmission is executed by an Arduino microcontroller, which transmits the data to a web server via GSM and 4G connections, resulting in a cost-saving monitoring system.

While the described publications use MQTT in the beekeeping context, none of them neither describe the exact development of the artifacts using MQTT nor guide to enable technophile beekeepers to replicate the developed solutions. Additionally, none of the mentioned publications describe why MQTT was chosen as the application layer protocol. Our work intends to show not only the general advantages and disadvantages of MQTT and HTTP, but also displays which advantages and disadvantages can be derived from a practical implementation. Furthermore, we describe how we proceeded to develop the prototypes as a starting point for technophile beekeepers.

The interim results of Biene40 project, as well as a more detailed description of this paper including the source code, will appear as work reports (in German, called Arbeitsberichte) on the *http://bieneviernull.de* website.

3 Methodology

To address the research questions outlined in Section 1, the following measures were undertaken.

3.1 Literature Review

To perform a literature review on the characteristics of MQTT and HTTP, we employed an unsystematic approach following the snowball method.

To initiate our research, a first search was conducted within the online library of the Hochschule Niederrhein (Niederrhein University of Applied Sciences). The search string "mqtt" generated eleven hits spanning a publication period from 2017 to 2021. These results were refined by applying a filter for monographs, thus excluding non-relevant materials such as bachelor theses, for example. Additionally, inaccessible sources and redundant entries were excluded from our consideration. Based on the title and the table of contents, two relevant titles were identified from the list of results, serving as the starting point for our snowball system. Following this procedure, we identified five sources. Three of the sources are websites associated with organizations that offer the advantage of regularly updated information.

One of the mentioned organizations is OASIS, which offers a standardization for the MQTT protocol. In addition to the standardization from OASIS, an ISO standard (ISO/IEC 20922:2016) and several RFCs are available. For our work, we only considered the OASIS standard as it has the same content as the ISO standard but is free to use.

Abts, 2015, was used as the starting point for research on HTTP's characteristics. Based on that, RFC 2616 (HTTP 1.1 specification) was identified as a further source.

During the evaluation of the identified literature, it was found that performance in terms of overhead and latency is not adequately represented in the selected sources. As these two characteristics are important criteria for applications in the Biene40 project, an additional search was carried out in the databases ScienceDirect and IEEE Xplore.

The search string "comparison AND http AND mqtt" was used. The period was limited to the years 2019 to 2023. Nine articles were found in IEEE Xplore. The results from ScienceDirect had to be narrowed down due to the high number of articles. Therefore, we filtered for publications with open access. With the restriction, 263 publications were found. Only publications that experimentally investigated the differences between the protocols in terms of performance were considered relevant. Based on this restraint, we identified another two articles from the IEEE Xplore database as relevant for our review. (Refer to *Appendix 1* for the identified literature.)

3.2 Prototypes

To develop the prototypes, we followed Hevner's Design Science in Information Systems Research "design-orientated research approach". (HEVNER ET AL., 2004)

For each prototype, a motion detector is connected to a microcontroller. Whenever the motion sensor is triggered,

the microcontroller transmits a message to a wireless plug, that turns on or off based on the transmitted message. The plug needs to support both the MQTT and the HTTP protocol.

A possible use case for the prototype is the application as a detector for theft or movement of the hive by wild animals. In case the sensor is triggered, a surveillance camera is switched on by the microcontroller.

The prototypes need to meet the goals of the Biene40 project. Therefore, the prototypes need to be simple, cost-effective, and minimally invasive to support the beekeeper's operation. (BRELL, n.d.-a)

The prototypes were created and tested in stable and secure WIFI networks.

4 Results

The results of the literature review and the prototypes are listed separately.

4.1 Literature Review

Table 1 displays a comparison of the identified characteristics of HTTP and MQTT based on the reviewed literature. Additionally, we performed two experiments to measure the packet sizes of MQTT and HTTP messages by using the network sniffer WireShark. We determined the packet size of the network traffic of the sent messages as well as their responses. In the MQTT transmission, we sent messages from one PC (publisher) to another PC (subscriber) via the Mosquitto test broker "test.mosquitto.org". The Mosquitto test broker uses port 1883 and processes messages without encryption. The used topic was named "mfp" and the transmitted message field was empty. The HTTP setup was also performed on a PC. The PC acts as a client that sends the requests to a web server. The URL of the server is nine characters long. A URI with eight characters is used to address the subdirectory of the server in which the script is located. Either "on" or "off" is transmitted as the payload of the message to change the status on the server.

In our setup, for a complete message transmission from publisher to subscriber, we measured a total packet size of 67 bytes for MQTT (see Appendix 3). In contrast to that, the total header size of an HTTP transmission measured 319 bytes (see Appendix 4).

To be able to compare both protocols, the variable parts of the messages have been subtracted from the total byte count. Therefore, for MQTT we subtracted the topic from the package sizes which resulted in 61 bytes per message transmission. For HTTP we subtracted the bytes for the host, URI, and message content which resulted in 252 bytes. Resulting from that, in our setup, messages sent with the MQTT protocol were only a quarter as big as the messages of the HTTP protocol. Generally, the overhead of MQTT is smaller than the overhead of HTTP as MQTT is designed for the efficient transfer of binary data which enables compact data transfer and a lower latency. (ŠIKIĆ ET AL., 2020: 85 f.) In contrast, HTTP was originally developed for the transfer of documents and therefore lacks in performance concerning the generated traffic load. (ŠIKIĆ ET AL., 2020: 85)

Nevertheless, HTTP is preferable for one-off message transmissions or for regular transmissions with long periods of inactivity, where each new message would require a new connection to be established. This results from the high network load of MQTT when establishing a connection. In such use cases, HTTP generates a lower network load than MQTT. (ŠIKIĆ ET AL., 2020: 85)

In contrast to HTTP, MQTT is a lightweight network protocol, but due to its persistent message and connection structure, it consumes a considerable amount of memory on the IoT gateway. (KAKAKHEL ET AL., 2019: 214)

	MQTT	HTTP
Origin	IBM, 1999 (TROJAN, 2017: 11)	1990 (RFC, 1999)
Technical Specifications	OASIS (TROJAN, 2017: 20)	RFC
Layer	Application layer	Application layer
Based on	TCP/IP (FRITZ ET AL., 2021: 70)	TCP/IP (ABTS, 2015: 161)
Architecture	Publish/Subscribe-principle, event-based-architecture (TROJAN, 2017: 12; HERRERO, 2022: 137)	Request/Response-principle, Client-Server-architecture (RFC, 1999)
Addressing	Topics (FRITZ ET AL., 2021: 70)	URI (URL) (RFC, 1999; ABTS, 2015: 161)
Header	Two Byte fixed header size, variable header size depends on the packet type (OASIS, 2014)	Request: HTTP-method, name of the resource and version of the protocol, (ABTS, 2015: 164) Response: version of the protocol, status code and status message, (ABTS, 2015: 168) min. eight Byte in total (CRAGGS, 2022)
Message size	max. 256 MB (CRAGGS, 2022)	No limit (CRAGGS, 2022)
Functionality	The publisher transmits a message on a specific topic to the broker, which manages the messages and relays them to all subscribers, who have subscribed to that topic. (FRITZ ET AL., 2021: 70; HERRERO, 2022: 137) A subscriber can subscribe to different topics and one topic can have multiple subscribers (HIVEMQ, 2023-a) " One-to-Many (CRAGGS, 2022)	A client initiates an HTTP request to the server to request a resource. The server processes this request and responds by providing the requested resource or an error message (HTTP response) (ABTS, 2015: 162) " One-to-One (CRAGGS, 2022)
Communication	Asynchronous (HERRERO, 2022: 137)	Synchronous (HERRERO, 2022: 111)
Persistent connections	Persistent sessions (HIVEMQ, n.d.-a)	Persistent connection since HTTP 1.1 (ABTS, 2015: 163)
State	See persistent sessions (broker saves the state of the subscriber depending on the QoS stage) (HIVEMQ, 2023-b)	Stateless (RFC, 1999), but Cookies can be used to save a state if necessary (HERRERO, 2022: 118)
Datatypes	Data independent (TROJAN, 2017: 13)	Text based (TROJAN, 2017: 20), binary data is transmitted using Base64 encoding (TROJAN, 2017: 20)
Quality of transmission	In addition to the TCP/IP quality, three QoS stages (TROJAN, 2017: 16f.)	Quality of transmission is based on TCP/IP (TROJAN, 2017: 16)
Application areas	IoT (HERRERO, 2022: 138), especially for instable networks (CRAGGS, 2022)	WWW; IoT, especially use cases with a low number of data transmissions (CRAGGS, 2022)
Distribution	Gaining relevance (CRAGGS, 2022)	Widely distributed because of the WWW (CRAGGS, 2022)
Performance	Smaller overhead (ŠIKIĆ ET AL., 2020: 85), lower latency, lower energy consumption (ŠIKIĆ ET AL., 2020: 86), higher memory consumption (KAKAKHEL ET AL., 2019: 214)	Bigger overhead (ŠIKIĆ ET AL., 2020: 85), higher latency (ŠIKIĆ ET AL., 2020: 86), higher energy consumption (NICHOLAS, 2012), lower memory consumption (KAKAKHEL ET AL., 2019: 214)
Prototype message size	61 Byte (excluding topic and message payload)	252 Byte (excluding host, URI, and message payload)
Prototype Line of Code (LoC)	265 (one script)	119+24+24 = 167 (three scripts)

Table 1: Comparison of MQTT and HTTP

4.2 Prototypes

Within the scope of this work, two prototypes were developed that switch a wifi operated plug Shelly Plus Plug S (SHELLY n.d.) on or off depending on the trigger of a motion sensor. An ESP8266 D1 Mini was employed as the microcontroller, due to its successful utilization in two related works. (MAHAMUD ET AL., 2019; MACHHAMER ET AL., 2020) The list of components is detailed in *Appendix 2*, with each component required once. It is assumed that small components such as pin headers, jumper cables, LEDs, resistors, a micro-USB cable, and a breadboard are already available and were not considered for procurement. To ensure comparability, the same hardware types are employed for both prototypes.

It is possible to construct an MQTT broker independently. A Raspberry Pi, for example, can be used as a platform for this purpose. Numerous guides are available on this subject. However, within the scope of our work, the employment and operation of a broker were not extensively examined, as it necessitates port forwarding on the router. This procedure carries a risk of unauthorized access if the network is inadequately secured, and maintenance is only carried out sporadically.

There is a variety of MQTT cloud brokers available on the market. A tabular overview of various MQTT brokers is provided by the GitHub repository *https://github.com/mqtt/mqtt.org/wiki/server-support#capabilitie*s. We chose the HiveMQ Cloud broker as it offers sufficient capacity for the MQTT prototype with a connection of up to 100 Clients and 10 GB data transmission for free.

For the implementation of the prototypes, we followed the guides from MAKESMART (2020; 2021) to set up the ESP8266 D1 Mini for the Arduino IDE, SIMAC ELECTRONICS GMBH (n.d.) to calibrate the sensor, HIVEMQ (n.d.-b) to set up the HiveMQ Cloud and the ESP8266D1 Mini for MQTT, and SHELLY (n.d.-a; n.d.-b; n.d.-c) to set up the Shelly Plug for MQTT.

Costs

To determine the price of the prototypes, we examined the top-selling online stores in Germany, that sell microcontrollers individually. The major providers identified are conrad.de, voelkner.de, and reichelt.de. (EHI RETAIL INSTITUTE GMBH, 2022)

The ESP8266 D1 Mini microcontroller and the Joy-it motion sensor can be ordered together from all online stores with a price range between 10,29 € and 13,48 €. Depending on the application scenario, the switchable socket Shelly Plus Plug S can also be ordered at a price of approx. 30 €.

For the setup of the reference system via HTTP, a server is additionally required, which can be rented from providers such as Strato or Domainfactory for a few euros per month.

Topology

The topology of the MQTT prototype is illustrated in Figure 1.

The hierarchy of the topics is specified by Shelly. The MQTT prefix of the topic can be amended manually in the Shelly dashboard. The Shelly Plug Plus S automatically subscribes to the topics and processes the messages according to its specifications.

The topology of the HTTP prototype is illustrated in Figure 2. Besides the ESP8266 D1 Mini and the Shelly Plug Plus S, a web server is used. Within the server infrastructure, there exists a PHP script named "Shelly.php," responsible for storing the desired status in the text file "status.txt". This is necessary because otherwise the Shelly Plus Plug S can only be operated in the same network or port forwarding must be enabled in the router. The microcontroller sends a message to the web server when a movement is detected, which changes its stored status. This approach, named the "concept of the hollow log", enables a high level of security as the external server is used as a mediating entity (BRELL, n.d.-b). The Shelly Plus Plug S requests status changes from the

web server every 5 seconds. In the code, 5000 milliseconds (5 seconds) are implemented with the line "/* number of miliseconds */ 5000". The time interval of the queries can be customized to align with the beekeeper's needs. With each response, the Shelly Plug adjusts its status to the status of the web server.

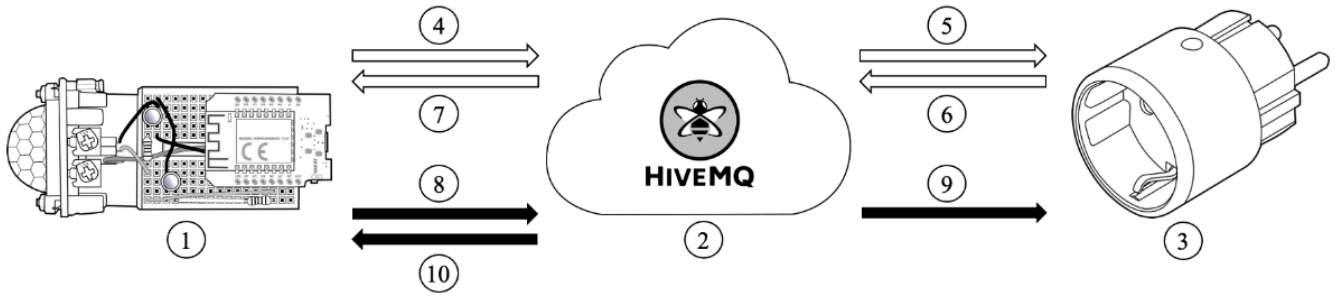

Figure 1: Topology of MQTT prototype; Source: Own representation It means: In this scenario, (1) the microcontroller ESP8266 D1 Mini and (3) the Shelly Plus Plug S send and receive messages as publisher and subscriber respectively. The ESP8266 D1 Mini (4) publishes a "status update" message on the topic "MQTT-prefix/status/switch:0". This message is transmitted to the (2) HiveMQ Cloud Broker. (5) The broker forwards the message to the Shelly plug, that subscribed to the topic. (6) The Shelly plug publishes the status in JSON format on the same topic. (7) The broker forwards the status message to the ESP8266 D1 Mini. (8) Depending on the triggering of the sensor on the ESP8266 D1 Mini, it publishes a message with either "on" or "off" on the topic "MQTT-prefix/command/switch:0". The broker forwards the message to both (9) the Shelly plug and (10) the ESP8266 D1 Mini as both devices have subscribed to the topic.

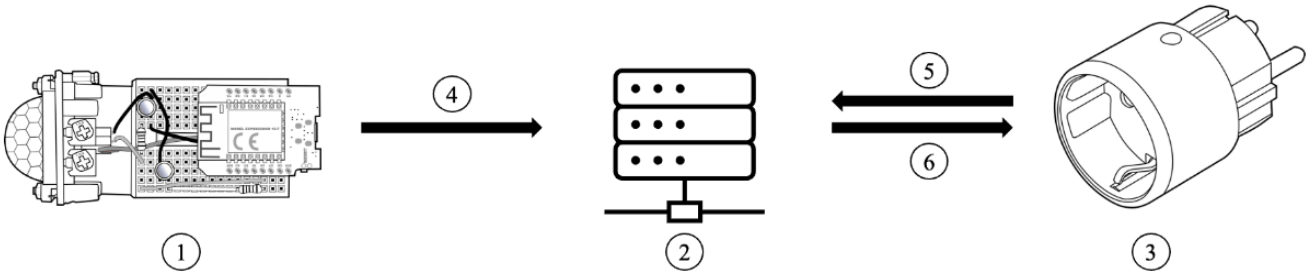

Figure 2: Topology of the HTTP prototype; Source: Own representation. It means: (1) The microcontroller ESP8266 D1 Mini sends (4) a message with the content "on" or "off" to the (2) web server. This changes the file entry in the file "status.txt". The (3) Shelly Plus Plug S requests (5) the status cyclically from the web server and receives (6) a response. The Shelly Plus Plug S then switches to the desired status.

5 Discussion

The advantages and disadvantages of MQTT and HTTP can be derived from the literature review and the development of the two prototypes.

5.1 Literature Review

MQTT offers one-to-many communication through the publish/subscribe principle, which facilitates scalability in distributed systems. (FRITZ ET AL., 2021: 70) With HTTP, on the other hand, scalability is aggravated by one-to-one communication.

In addition, the publish/subscribe principle and the use of a broker improve latency and network utilization. In contrast, the entire system is dependent on the accessibility and reliability of the broker. Thus, if the broker fails or stops working, the entire communication in the network may be disrupted or interrupted. (HERRERO, 2022: 111)

Although the overhead of MQTT is generally smaller than the overhead of HTTP, HTTP is preferable for a one-time message transmission or for regular transmissions with long periods of inactivity. In these cases, HTTP generates a lower network load than MQTT.

In the context of beekeeping, hives are often located in places with a poor WIFI connection and where sometimes even a poor mobile network can be expected. In this context, a protocol that ensures transmission is favorable.

Compared to HTTP, MQTT is particularly advantageous for unstable networks due to the asynchronous communication, the availability of three QoS levels, and persistent sessions. HTTP has offered persistent connections since HTTP 1.1, where multiple messages can be exchanged over the same TCP connection, but it remains stateless. Therefore, previous communication does not affect current requests. This poses the risk of packet loss if the TCP connection is interrupted. An add-on mechanism called cookies can be implemented to store the state in HTTP. However, the implementation of cookies requires additional effort in comparison to MQTT.

Furthermore, there might be no access to electricity at the hive locations, which should be considered when selecting an application layer protocol as well. MQTT offers more advantages for transmission in such cases as energy consumption and latency are lower with MQTT. In contrast, MQTT also requires more memory capacity due to its messaging structure, which also needs to be considered.

With MQTT, the message size is limited to 256 MB, while HTTP allows a larger message size if the post-method is used. This makes it possible to transmit relevant data about bees such as hive weight, internal and external temperatures, as well as other small data packages such as air quality data. MQTT might be not suitable for transmitting real-time audio, image, or video material due to the limited message size.

An advantage of MQTT lies in its data format independence, whereas HTTP is text-based. Consequently, HTTP messages are in a format easily readable by humans without requiring additional effort, thereby facilitating the troubleshooting processes. (HERRERO, 2022: 116)

5.2 Prototypes

During the implementation of the prototypes, the availability of resources and widespread expertise also stood out as a disadvantage of MQTT in comparison to HTTP. In particular, the Shelly plugs' documentation for setting up the MQTT protocol was often unclear. Blog posts or other resources that normally can be relied upon for troubleshooting were hard to find.

The code for controlling the ESP8266 D1 Mini in the Arduino IDE has more LoC for MQTT in comparison and therefore could be interpreted as more elaborate. However, in comparison to the HTTP prototype, for the MQTT prototype, an additional status query has been implemented. On top of that, SSL encryption of the data transmission was programmed for the MQTT prototype. Accordingly, the security level of the transmission is higher for the MQTT prototype than for the HTTP prototype. Furthermore, for the HTTP prototype, additional scripts had to be written for the server, as well as for the Shelly plug, which puts the number of LoCs into perspective.

Advantages and disadvantages also emerged for the use of the HiveMQ Cloud Broker. On the one hand, the use of the broker is free of charge with a connection of up to 100 devices and 10 GB data transfers per month. On the other hand, a user profile with personal data must be created to be able to use the broker. Furthermore, the HiveMQ Cloud Broker is like a black box. The only overview in the browser dashboard shows the current usage, i.e. how many endpoints are currently connected and how much traffic was transmitted in total. It is neither possible to monitor which devices are connected to the broker, nor what the messages being transmitted look like, which was especially problematic for troubleshooting. Moreover, the individual message sizes cannot be traced either.

Another aspect is the standard use of the 8883 port of the HiveMQ Cloud Broker. On the one hand, this guarantees secure transmission using SSL encryption. On the other hand, messages transmitted via the 8883 port cannot be easily read out using network sniffers such as "WireShark", which makes both troubleshooting and measuring parameters such as the header more difficult.

Furthermore, errors arose during the generation of the SSL certificates in Microsoft, which were difficult to fix because of the limited documentation.

The costs for the HTTP prototype as well as for the MQTT prototype are similar. For 35-40 € the prototypes can be reproduced. In the context of the HTTP prototype, additional costs arise when opting for an external server, particularly if port forwarding within one's own network is not deemed a secure option.

MQTT unfolds its advantages, especially in distributed networks with a high number of devices and data transmissions in weak networks. These criteria were not met in our given use case. Since both devices, the ESP8266 D1 Mini and the Shelly Plus Plug S were in the same stable home network, it would have been possible to use the Shelly plug as a server via HTTP and to address it directly. Consequently, the inclusion of an additional server would be redundant. However, it is important to note that the ability to control the Shelly plug would be compromised if the ESP8266 D1 Mini had been situated outside the network.

6 Conclusion

Honey bees play an important role in pollination and need to be protected. Communication technologies can help in this endeavour. Our work compares the HTTP and MQTT protocols and shows their differences and similarities under the conditions of beehive monitoring.

MQTT and HTTP are both application layer protocols, but they differ in their architecture and communication principles. MQTT offers more flexibility and reliability through QoS levels. HTTP is good for one-off message transmissions or periodic transmissions with long periods of inactivity. MQTT is more efficient on limited and unstable networks. HTTP is more widely used and better documented than MQTT. The evaluation of the developed prototypes suggests that MQTT provides a higher level of security, but requires more code and has limited documentation. The costs are about the same for the HTTP and MQTT prototypes. The advantages and disadvantages need to be weighed up depending on the purpose of the application. In terms of the prototypes developed, HTTP is the preferred choice.

Future research may be devoted to other IoT protocols and the transmission of other relevant data in the beekeeping context.

7 Acknowledgement

This paper is motivated by a specific problem in the Biene40 project - An apiary often has neither a power grid nor a powerfull network. The Biene40 project is scheduled to run from March 1, 2021, to February 29, 2024, and is financially supported by the German Federal Ministry of Food and Agriculture (BMEL). Results in the project will be made available promptly at *http://bieneviernull.de*, so that beekeepers can immediately benefit from the results immediately.

For this paper, Jan Langen and Kerstin Sahler. contributed equally.

8 References

ABTS, D. (2015): Masterkurs Client/Server-Programmierung mit Java: *Anwendungen entwickeln mit Standard-Technologien, 4. Ed., Wiesbaden:* Springer Fachmedien Wiesbaden, 2015.

Beenovation (n.d.), https://www.beenovation.de, Accessed on 04.07.2023.

BRELL, C. (n.d.-a): Biene40 – Entwicklung digitaler vernetzter Sensoren für vitalere Bienen, http://bieneviernull.de, Accessed on 04.07.2023.

BRELL, C.-(n.d.-b): Was ist IoT (Internet of Things)?, https://cbrell.de/blog/was-ist-iot-internet-of-things/, Accessed on 23.09.2023.

BRELL, C. (n.d.-b): Konzept des hohlen Baumstamms – was IoT und Drogendealer gemein haben. https://cbrell.de/blog/konzept-des-hohlen-baumstamms-was-iot-und-drogendealer-gemein-haben/, Accessed on 05.10.2023.

BRELL, C. (2022): Technischer Bericht – Sensoren in Bienenstöcken (Zustand), http://bieneviernull.de/technischer-bericht-sensoren-in-bienenstoecken-zustand/, Accessed on 04.07.2023.

CRAGGS, I. (2022): MQTT Vs. HTTP for IoT, https://www.hivemq.com/blog/mqtt-vs-http-protocols-in-iot-iiot/, Accessed on 25.04.2023.

DEUTSCHER IMKERBUND E.V. (n.d.-a): Bienen als Bestäuber, https://deutscherimkerbund.de/163-Bienen_Bestaeubung_Zahlen_die_zaehlen, Accessed on 04.07.2023.

DEUTSCHER IMKERBUND E.V. (n.d.-b): Bienen als Bestäuber helfen beim Erhalt der Artenvielfalt, https://deutscherimkerbund.de/235-Echter_Deutscher_Honig_Honig_und_Natur_Schutz, Accessed on 04.07.2023.

EHI RETAIL INSTITUTE GMBH (2022): Top 100 Onlineshops in Deutschland, https://www.ehi.org/news/top-100-onlineshops-in-deutschland/, Accessed on 30.06.2023.

Fritz, K.P./Strauß, H./Rathfelder, C./Bülau, A./Gaida, D./Girdvainis, D./Marki, G. (2021): Digitaler Retrofit: von Maschinen und Produktionsanlagen, 1. Ed., Würzburg: Vogel Communications Group, 2021.

Herrero, R. (2022): Fundamentals of IoT Communication Technologies, 1. Ed., Cham: Springer Nature Switzerland, 2022.

Hevner, A./ March, S/ Park, J./Ram, S. (2004): Design Science in Information SystemsResearch, in: MIS Quarterly 28 -1, S.75-100.

HiveMQ (n.d.-a): Key Features, https://www.hivemq.com/mqtt/mqtt-protocol/, Accessed on 26.04.2023.

HiveMQ (n.d.-b): Getting started with Arduino ESP8266, https://console.hivemq.cloud/clients/arduino-esp8266?uuid=42ef8d8b455248368b9538e9282ad896, Accessed on 15.06.2023.

HiveMQ (2023-a): Introducing the MQTT Protocol – MQTT Essentials: Part 1, https://www.hivemq.com/blog/mqtt-essentials-part-1-introducing-mqtt/, Accessed on 23.09.2023.

HiveMQ (2023-b): Understanding Persistent Sessions and Clean Sessions – MQTT Essentials: Part 7, Accessed on 23.09.2023.

Jara Ochoa, H.J.; Peña, R.; Ledo Mezquita, Y.; Gonzalez, E.; Camacho-Leon, S (2023).: Comparative Analysis of Power Consumption between MQTT and HTTP Protocols in an IoT Platform Designed and Implemented for Remote Real-Time Monitoring of Long-Term Cold Chain Transport Operations. Sensors 2023, 23,4896. https://doi.org/10.3390/ s23104896, Accessed on 04.01.2024.

Kakakhel, S. R. U./Westerlund, T./Daneshtalab, M./Zou, Z./Plosila, J./Tenhunen, H. (2019): A Qualitative Comparison Model for Application Layer IoT Protocols, 2019 Fourth International Conference on Fog and Mobile Edge Computing (FMEC), Rome, Italy, 2019, S.210-215, Verfügbar unter: 10.1109/FMEC.2019.8795324.

Machhamer, R. / Altenhofer, J. / Ueding, K. / Czenkusch, L. / Stolz, F. / Harth, M. / Mattern, M. / Latif, A. / Haab, S. / Herrmann, J. / Schmeink, A. / Gollmer, K.-U. / Dartmann, G. (2020): Visual Programmed IoT Beehive Monitoring for Decision Aid by Machine Learning based Anomaly Detection, https://ieeexplore.ieee.org/document/9134323, Accessed on 06.07.2023.

Mahamud, S. / Rakib, A. / Faruqi, T. / Haque, M. / Rukaia, S. / Nazmi, S. (2019): Mouchak - An IoT Basted Smart Beekeeping System Using MQTT, https://ieeexplore.ieee.org/document/9043815, Accessed on 06.07.2023.

makesmart (2020): ESP8266 D1 Mini programmieren, https://makesmart.net/esp8266-d1-mini-programmieren/, Accessed on 11.05.2023.

makesmart (2021): Arduino IDE – Arbeiten mit JSON Objekten für Einsteiger, https://makesmart.net/arduino-ide-arbeiten-mit-json-objekten-fur-einsteiger/, Accessed on 20.06.2023.

Nicholas, S.D. (2012): Power Profiling: HTTPS Long Polling vs. MQTT with SSL, on Android, http://stephendnicholas.com/posts/power-profiling-mqtt-vs-https, Accessed on 30.09.2023.

OASIS (2014): MQTT Version 3.1.1, http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html#_Toc398718018, Accessed on 06.07.2023.

RFC (1999): Hypertext Transfer Protocol – HTTP/1.1, https://www.rfc-editor.org/rfc/rfc2616, Accessed on 23.09.2023.

Shelly (n.d.-a): Shelly Plus Plug S User and Safety Guide, https://kb.shelly.cloud/knowledge-base/shelly-plus-plug-s-user-and-safety-guide, Accessed on 15.06.2023.

Shelly(n.d.-b):MQTT,https://shelly-api-docs.shelly.cloud/gen2/0.14/ComponentsAndServices/Mqtt/, Accessed on 15.06.2023.

Shelly (n.d.-c): Unsere App Shelly Smart Control, https://www.shelly.cloud/de/shelly-smart-control, Accessed on 21.07.2023.

Šikić, L./ Janković, J./ Afrić, P./Šilić, M./Ilić, Ž./Pandžić, H./Živić, M./Džanko, M. (2020): A comparison of Application Layer Communication Protocols in IoT-enabled Smart Grid, 2020 International Symposium ELMAR, Zadar, Croatia, 2020, S.83-86, Verfügbar unter: 10.1109/ELMAR49956.2020.9219030.

SIMAC Electronics GmbH (n.d.): Passiver Infrarot-Bewegungssensor: SEN-HC-SR501, https://asset.conrad.com/media10/add/160267/c1/-/gl/002361790ML01/bedienungsanleitung-2361790-joy-it-sen-hc-sr501-bewegungssensor-1-st.pdf, Accessed on: 21.05.2023.

Trojan, W. (2017): Das MQTT-Praxisbuch, 1. Ed., Aachen: Elektor Verlag GmbH, 2017.

Wurm, J./Brell, C. (2022): Imkerei und Digitalisierung – Stand und Perspektiven aus technischer Sicht. Arbeitsbericht Nr. 3 / Biene40, http://bieneviernull.de/wp-content/uploads/2022/08/Biene40_Arbeitsbericht_Nr_03_Literaturanalyse_Informatik_final.pdf, Accessed on 06.07.2023.

Yokotani, Tetsuya & Sasaki, Yuya (2016) Comparison with HTTP and MQTT on Required Network Resources for IoT. The 2016 International Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC). https://dl.icdst.org/pdfs/files3/ca1b4950d958c329cecac645b95101f9.pdf . Accessed on 04.01.2024.

Yusof, Z. M. / Billah, M. / Kadir, K. / Mohd Ali, A. M. / Ahmad, I. (2019) Improvement of Honey Production: A Smart Honey honey bee Health Monitoring System,https://ieeexplore.ieee.org/document/9057336, Accessed on 06.07.2023.

Zabasta, A. / Kunicina, N. / Kondratjevs, K. / Ribickis, L. (2019) IoT Approach Application for Development of Autonomous Beekeeping System, https://ieeexplore.ieee.org/document/8883460, Accessed on 06.07.2023.

9 Appendix

Appendix 1:

#	Author (year)	Titel	Short Abstract
1	**ABTS, D.** (2015)	Masterkurs Client/Server-Programmierung mit Java	Technologies for the development of client/server applications including presentation of various network protocols
2	**FRITZ, K.P./STRAUß, H./RATHFELDER, C./BÜLAU, A./GAIDA, D./GIRDVAINIS, D./MARKI, G.** (2021)	Digitaler Retrofit: von Maschinen und Produktionsanlagen	Overview of standards, interfaces, and protocols
3	**HERRERO, R.** (2022)	Fundamentals of IoT Communication Technologies	Various communication technologies of the IoT, including different network protocols
4	**HIVEMQ** (n.d.)	MQTT Essentials	Provider for broker and member of OASIS; information on various elements of MQTT as well
5	**JARA OCHOA, H.J.; PEÑA, R.; LEDO MEZQUITA, Y.; GONZALEZ, E.; CAMACHO-LEON, S** (2023)	Comparative Analysis of Power Consumption between MQTT and HTTP Protocols in an IoT Platform Designed and Implemented for Remote Real-Time Monitoring of Long-Term Cold Chain Transport Operations.	Analysing Power Consumption. Experimentation is carried out for HTTP and MQTT with different QoS levels to make a comparison and demonstrate the differences in power consumption. The results in power savings of MQTT are 6.03% and 8.33% compared with HTTP
6	**MQTT.ORG** (n.d.)	FAQ	General information on various elements of MQTT
7	**NICHOLAS, S.D.** (2012)	Power Profiling: HTTPS Long Polling vs. MQTT with SSL, on Android	Measurement of battery consumption when using MQTT and HTTPS
8	**OASIS** (n.d.)	MQTT Specifications	Documentation of the MQTT standards according to OASIS
9	**RFC 2616** (1999)	Hypertext Transfer Protocol – HTTP/1.1	Specification for HTTP 1.1
10	**TROJAN, W.** (2017)	Das MQTT-Praxisbuch: Mit ESP8266 und Node-RED	Introduction to the functionality, architecture, and practical implementation of MQTT
11	**YOKOTANI, TETSUYA & SASAKI, YUYA** (2016)	Comparison with HTTP and MQTT on Required Network Resources for IoT	HTTP has been widely applied for data transfer but causes a large overhead. To solve this problem, named based transfer protocols have been discussed. This paper compares the performance of HTTP with that of MQTT, a type of named based transfer protocol.

Table 2: Overview for found literature with the snowball method for HTTP and MQTT

Appendix 2:

Name of the component	conrad.de	voelkner.de	reichelt.de	combined
ESP 8266 D1 Mini (for soldering)	9,49 €	6,49 €	7,99 €	6,49 €
Joy-it SEN-HC-SR501 Sensor	3,99 €	3,99 €	2,30 €	2,30 €
Shelly Plus Plug S	23,68 €	23,48 €	30,95 €	23,48 €
Total	**37,16 €**	**33,96 €**	**41,24 €**	**32,27 €**

Table 3: Prices of the various components

Appendix 3:

Source	Type of Packet	Size of Packet
Subscriber to Broker	CONNECT	14 Byte
Broker to Subscriber	CONNACK	4 Byte
Subscriber to Broker	SUBSCRIBE	10 Byte
Broker to Subscriber	SUBACK	5 Byte
Publisher to Broker	CONNECT	14 Byte
Broker to Publisher	CONNACK	4 Byte
Publisher to Broker	PUBLISH	7 Byte
Publisher to Broker	DISCONNECT	2 Byte
Broker to Subscriber	PUBLISH	7 Byte
Subscriber		**24 Byte**
Publisher		**23 Byte**
Broker		**20 Byte**
Total		**67 Byte**

Table 4: MQTT packet size for topic "mfp" and payload "" (no payload).

Appendix 4:

Source	Typ of packet	Content	Size of Packet
Client to Server	**REQUEST**		**95 Byte**
	Request Method	"GET "	4 Byte
	Request URI	"<HTTP URI> /shelly.php?x=on " (for 8 characters)	25 Byte (if x=on) *26* Byte *(if x=off)*
	Request Version	"HTTP/1.1 "	10 Byte
	Host	"Host: HTTP-SERVER-URL .de " (for 6 characters)	17 Byte
	User Agent	"User-Agent: curl/8.0.1 "	24 Byte
	Accept	"Accept: */* "	15 Byte
Server to Client	**RESPONSE**		**224 Byte**
	Response version	"HTTP/1.1 "	9 Byte
	Status Code	"200 "	4 Byte
	Response Phrase	"OK "	4 Byte
	Date	"Date: Thu, 05 Oct 2023 07:34:37 GMT "	37 Byte
	Server	"Server: Apache/2.4.57 (Unix) "	30 Byte
	Response Line	"X-Powered-By: PHP/8.2.10 "	26 Byte
	Content Type	"Content-Type: text/html;charset=UTF-8 "	40 Byte
	Transfer Encoding	"Transfer-Encoding: chunked "	28 Byte
	Chunk Size	" 15 "	6 Byte
	Chunk Data	"Der Modus wurde auf 'on' gesetzt."	33 Byte *34 Byte (Der Modus wurde auf 'off' gesetzt.)*
	Chunk Size	" 0 "	5 Byte
	Text Item	" "	2 Byte
Total			**319 Byte**

Table 5: Packet sizes in HTTP. The contents of the packages are also listed for a better understanding.