СИСТЕМА ЗА СЪБИРАНЕ И СЪХРАНЕНИЕ НА ДАННИ ОТ НЕЗАВИСИМИ ИЗТОЧНИЦИ

Делян Генков[1], Емил Ангелов[1]
[1]*Технически университет - Габрово*

SYSTEM FOR COLLECTION AND STORAGE OF DATA FROM INDEPENDENT SOURCES

Delyan Genkov[1], Emil Angelov[1]
[1] *Technical University - Gabrovo*

Abstract

Internet of Things becomes very popular technology nowadays, not only in home systems, but also in many industrial fields of application. The technology presumes many autonomous systems in form of controllers with sensors attached to measure some data, and to send the measured values for centralized storage and management. There are many possible approaches for this technology – including cloud computing, fog computing and combined approaches.

This paper presents a network setup and software realization for sending data from many IoT devices through MQTT protocol and storing them in a central location.

Keywords: Internet of Things, data, centralized management, MQTT.

ВЪВЕДЕНИЕ

Технологията Интернет на нещата (Internet of Things) представлява свързване на сензори и/или изпълнителни механизми към евтини контролери с ограничени ресурси и възможности, които могат да се свързват към компютърна мрежа/Интернет за да бъдат наблюдавани данните, следени от устройството и подавани команди към изпълнителните механизми ръчно от оператор или автоматично чрез вградена логика в софтуера на контролера.

Често контролерите се намират физически на отделни места и нямат директна връзка помежду си, затова за да се обединят данните от различните контролери е необходимо да се организира предаването им към централно устройство – сървър и съхранението им в база данни за бъдеща обработка.

В рамките на предишни изследвания бяха проведени експерименти с различни готови системи за приемане и съхранение на данни от контролери, но беше установено че работата с готова програмна система не е толкова гъвкава и е трудна за поддръжка.

В настоящия доклад е представена мрежова постановка, осигуряваща предаване на данни чрез протокола MQTT към централен сървър и програмна реализация на система за получаването и съхранението им в база данни.

MQTT (MQ Telemetry Transport или Message Queuing Telemetry Transport) е отворен стандарт на OASIS и ISO (ISO / IEC 20922) на мрежов протокол, който транспортира съобщения между устройства. Протоколът се отличава с много ниско натоварване на каналите за връзка, и е изграден на базата публикуване-абониране. Протоколът обикновено работи през TCP/IP, въпреки това всеки мрежов протокол, който осигурява подредени, без загуби на съобщения, двупосочни канали за връзка, може да поддържа MQTT. Той е предназначен за връзка с отдалечени места, където се изисква генериране на малки съобщения или мрежовата честотна лента е ограничена.

Протоколът дефинира два типа мрежови обекти: брокер на съобщения и множество абонати (клиенти). MQTT брокер е сървър, който получава всички съобщения от абонираните клиенти към дадена тема и след това маршрутизира съобщенията до съответните клиенти които са абонирани и подслушват същата тема. MQTT клиент може да бъде всякакво устройство (от микроконтролер до напълно функциониращ сървър), устройство което посредством помоща на MQTT библиотека се свързва към MQTT брокер през мрежа (най-често безжична).[1]

Информацията се организира в йерархия от теми. Когато клиент получи актуална информация, дали чрез сензор или статично получена информация, той публикува съобщение с данните към брокер изчакващ нови съобщения към определена тема. След това брокерът разпространява информацията на всички клиенти, които са се абонирали за тази тема. Клиентът публикуващ информацията не получава данни за броя или местоположението на абонатите, а абонатите от своя страна не трябва да бъдат конфигурирани с никакви данни за издателите на съобщенията.[1]

ИЗЛОЖЕНИЕ

Избраният модел на комуникация в разработката е клиент-сървър. При този модел на комуникация, IoT контролерите със различни сензори се свързват към сървърна част на която се изпълнява програмната система. Клиентът (контролерът) ще изпраща съобщения към сървъра (брокера), след което клиентът (приложението) ще се абонира към определена тема на сървъра (брокера) и ще получава съобщенията изпращани от клиента (контролера) за тази определена тема.

За комуникация между приложението и контролера се използва мрежовия протокол MQTT и брокер на съобщения.

Моделът на комуникация и връзките между отделните компоненти са представени на фигура 1.

Фиг. 1. *Модел на комуникация*

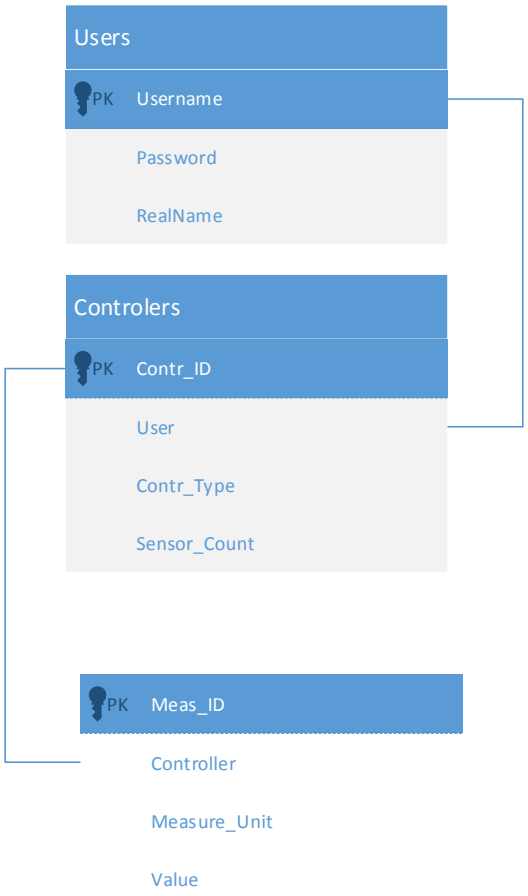
Стъпките в комуникацията са:

1. IoT контролерът прави опит да се свърже към локалния маршрутизатор посредством WiFi.
2. След успешно свързване към него, контролерът се опитва да се свърже с Mosquitto брокера.
3. След успешно свързване контролерът се абонира към определена MQTT тема.
4. След абонирането, контролерът започва да изпраща съобщения, съдържащи информацията която отчита посредством сензора свързан към него, използвайки протокола MQTT.
5. Брокерът получава съобщенията, но не ги съхранява.
6. Сървърът се свързва към Mosquitto брокера.
7. След успешно свързване, сървърът се абонира към същата тема в която се получават съобщенията от IoT контролера.
8. Сървърът започва да получава съобщения от брокера.
9. Сървърът се свързва към базата данни, където съхранява информацията съдържаща се в MQTT съобщенията.
10. Уеб приложението се свързва към базата данни, след което извлича последното получено съобщение.

11. При свързване на потребител към приложението, той трябва да въведе потребителско име и парола, съхранявани в базата данни, за да успее да визуализира съхраняваните данни.

12. При успешно въвеждане на валидни потребителско име и парола, приложението визуализира актуална информация получавана от темата към която е абониран сървърът.

Базата данни на приложението е показана на фигура 2.



***Фиг. 2**. База данни на приложението*

В разработката се използва MySQL Community Server 8.0.21 64-bit [2], инсталиран върху операционна система Microsoft Windows 10x64. Базата данни дава възможност да се регистрират различни потребители, всеки потребител да регистрира свои контролери и да има достъп до данните само от тях. Всеки контролер може да има определен брой сензори, чиито показания се предават през MQTT брокера към web приложението и се записват в базата данни.

Разработеното приложение е написано на програмен език Java и работи върху Web сървър Apache Tomcat, версия 8.5.59. [3]

Ролята на MQTT брокер се изпълнява от Eclipse Mosquitto версия 1.6.12a - брокер на съобщения с отворен код, имплементиращ протокола MQTT, инсталиран като услуга под операционна система Microsoft Windows 10x64. Брокерът се грижи за управлението на съобщенията, клиентите се абонират към определена тема, когато брокерът получи съобщение предназначено за тази тема, той изпраща копие на съобщението към всички активни абонати на тази тема. Брокерът не изпраща информация за подателя на съобщението или за получателите.

Брокерът използва следните портове: порт номер 1883, използващ се за некриптирани съобщения; порт номер 8883 използващ се за криптирани съобщения; порт номер 8884 изпозлващ се за криптирани съобщения придружавани от клиентски сертификат за достоверност; порт номер 8080, използващ се за некриптирани MQTT съобщения през уеб сокет; порт номер 8081 използващ се за криптирани MQTT съобщения през уеб сокет. Криптираните портове поддържат TLS v1.3, v1.2 и v1.1.

За тестовете на системата са използвани контролери Sonoff TH16 със сензори за температура и влажност. Принципно може да се използват произволни контролери с поддръжка на MQTT протокол и произволни сензори, тъй като в JSON описанието, предавано от контролера се съдържа информация за типа на измерваната величина и измерената стойност, както и датата и часа на измерването. Програмният код на контролера е показан на фигура 3.

```
  const char* ssid = "";
  const char* password = "";
  const char* mqttServer = "";
  const int mqttPort = 1883;

  void setup() {
    Serial.begin(115200);
    dht.begin();
    WiFi.begin(ssid, password);
    while (WiFi.status() !=
WL_CONNECTED) {
```

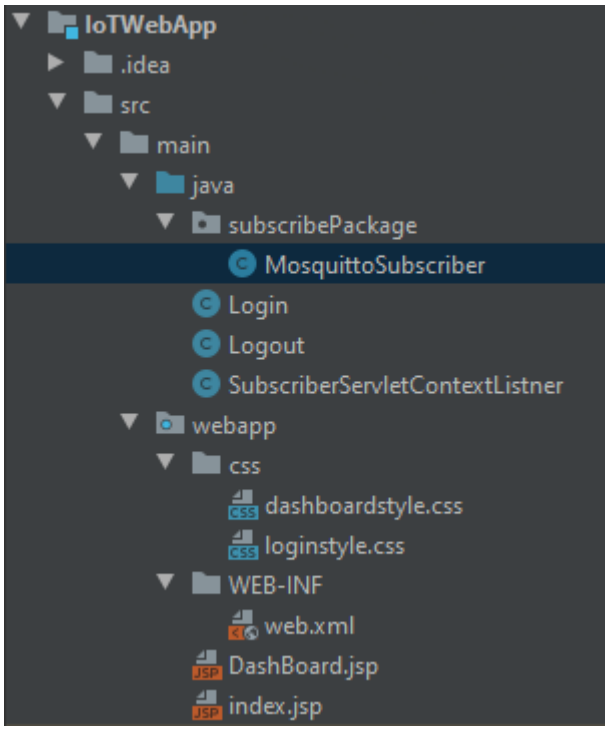
```
      delay(500); }
          client.setServer(
mqttServer, mqttPort );
  }
  void loop() {
  if ( !client.connected() )
  {
      reconnect();
  }

  getAndSendData();
    delay(30000);
  }

  void getAndSendData()
  {
    float humidity =
dht.readHumidity();
    float temperature =
dht.readTemperature();

    if (isnan(humidity) ||
isnan(temperature)) {
          return;
    }
    client.publish("esp/temperature",
String(temperature).c_str());
    client.publish("esp/humidity",
String(humidity).c_str());
  }

  void reconnect() {
    while (!client.connected()) {
      status = WiFi.status();
      if ( status != WL_CONNECTED) {
        WiFi.begin(ssid, password);
        while (WiFi.status() !=
WL_CONNECTED) {
          delay(500);
        }
           }
          if (
client.connect("Esp8266", NULL, NULL)
) {
      } else {
        delay(500);
      }
    }
```

***Фиг. 3**. Програмен код на контролера*

Web приложението се състои от следните компоненти, графично показани на фигура 4:

Maven файл pom.xml - съдържа всички зависимости на проекта. Файлът управлява и организира необходимите библиотеки за изпълнението на проекта.

Клас MosquittoSubscriber.java - главният клас на приложението, който имплементира интерфейс MqttCallback. Чрез него се свързва към Mosquitto брокера, след което се абонира за определени теми и данните получавани от тези теми се въвеждат в база данни.



***Фиг. 4**. Компоненти на приложението*

Клас MosquittoSubscriber - класът който имплементира интерфейс MqttCallback, има за цел да установи връзката с брокера и да получава и съхранява съобщения от него.

Методът doSubscribe() има за цел да създаде връзка към брокера. Задава се нова чиста сесия, след което се свързва с брокера.

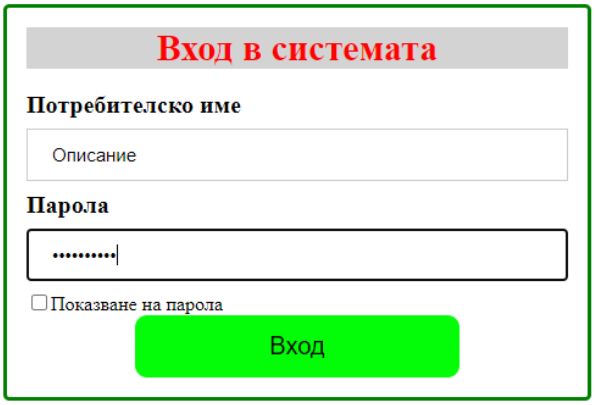
Методът messageArrived() обработва съобщенията получавани от брокера. Той приема за аргументи темата обвързана със съобщенията които трябва да се получават и съобщението което е получено. При получаване на съобщение се остановява връзка с базата данни, след което се създава запис в таблицата съответстваща на темата към която е пред-назначено съобщението.

Клас SubscriberServletContextListner.java - класът имплементира интерфейс ServletContextListener. При стартиране на уеб сървъра метод contextInitialized() извиква клас MosquittoSubscriber и изпълнява метода за абониране към темите.

Сървлет Login.java - целта му е управлението на JSP страниците. При вход към системата, потребителите трябва да въведат валидни потребителско име и парола. Сървлетът се свързва към база данни, за да провери валидността на данните.

Сървлет Logout.java - целта му е да обяви сесията за невалидна след нейния край.

Входната форма на приложението е показана на фигура 5.



***Фиг. 5**. Входна форма*

В текущата реализация данните се записват в базата данни и е създадена проста визуализация, показана на фигура 6.

Система за визуализация на данни от контролери

Температура	Влажност
24.4 C	75%
Дата и час на получаване на информацията	Дата и час на получаване на информацията
2020-06-17 23:17:55	2020-06-17 23:17:55

***Фиг. 6**. Визуализация на данните*

ЗАКЛЮЧЕНИЕ

В настоящия документ е представена собствена разработка за мрежова постановка, осигуряваща предаване на данни чрез протокол MQTT към централен сървър и програмна реализация на система за получаването и съхранението им в база данни. Системата може да се използва за събиране и обработка на различни данни от сензори на различни географски места Подходът може да намери приложение в разнообразни приложения на „Интернет на нещата“.

Бъдещите планове включват подобряване на потребителския интерфейс, добавяне на възможности за визуализация и справки, графично представяне на данните и добавяне на аналитични функции за обработка на данните.

БЛАГОДАРНОСТИ

Настоящият документ е изготвен с финансовата помощ на договор № 2004Е за провеждане на научни изследвания по проект на тема: „Съвременни методи за окачествяване на храни и материали“ към Технически университет – Габрово.

REFERENCE

[1] MQTT, https://en.wikipedia.org/wiki/MQTT, дата на използване: юни, 2020.

[2] Oracle, MySQL Community Edition, https://www.mysql.com/products/community/, дата на използване: септември, 2020.

[3] Apache Software Foundation, Apache Tomcat, http://tomcat.apache.org/, дата на използване октомври 2020.

[4] Eclipse Foundation, Eclipse Mosquitto™ An open source MQTT broker, https://mosquitto.org/, дата на използване 10.2020.