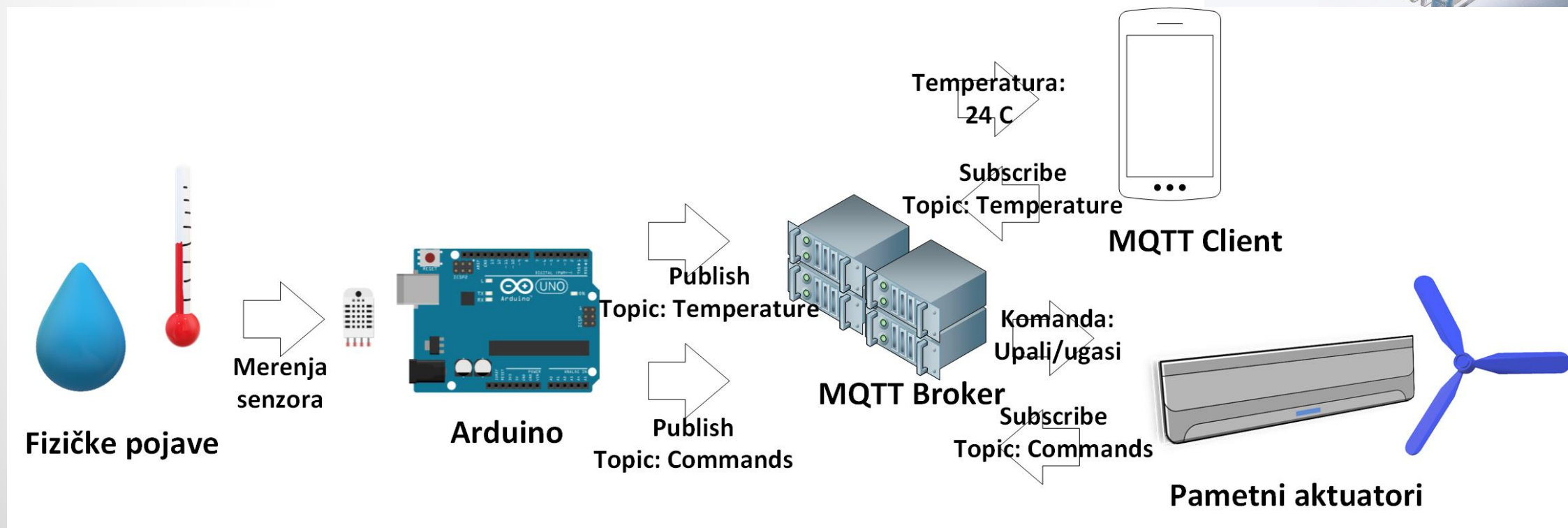


# Arduino – MQTT preko Ethernet Shield-a

## *Internet stvari 2023. - IX termin*

Nenad Petrović

Univerzitet u Nišu, Elektronski fakultet  
[nenad.petrovic@elfak.ni.ac.rs](mailto:nenad.petrovic@elfak.ni.ac.rs), kancelarija 323

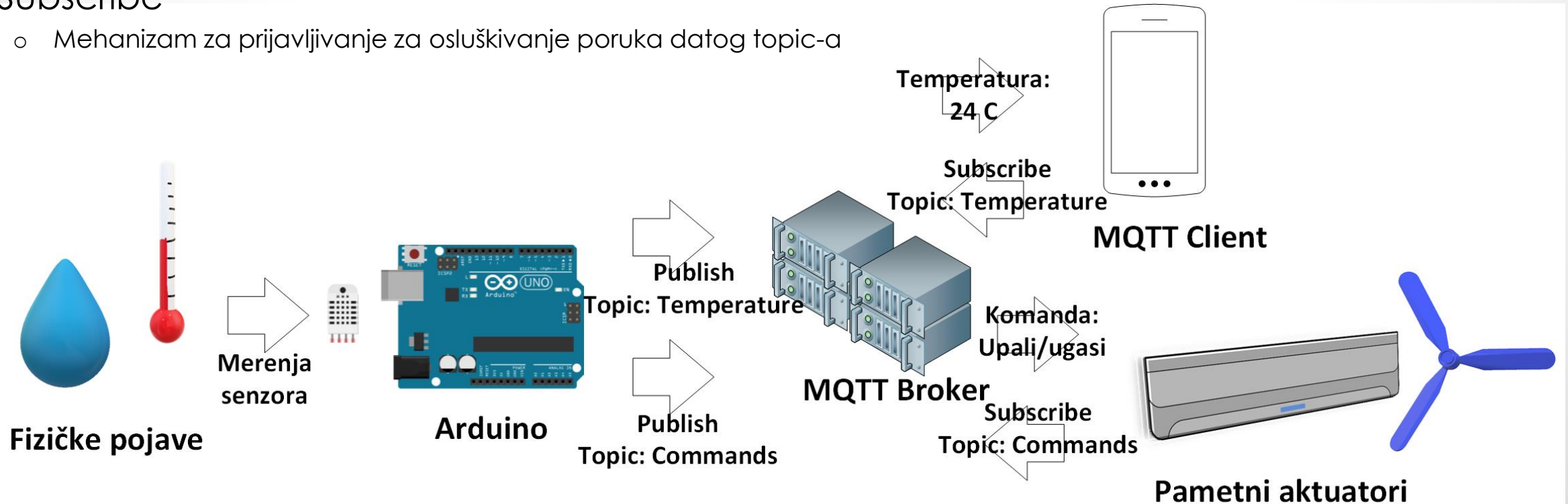


# Uvod

- Message Queuing Telemetry Transport (MQTT) je publish-subscribe protokol za prenos podataka namenjen manjim IoT uređajima
  - Male dimenzije
  - Mala potrošnja
  - Ograničena procesorska moć
- Mali zahtev resursa i nizak overhead komunikacije
  - Zaglavlja malih dimenzija sa ciljem povećanja protoka
- Zahteva mrežnu konekciju i povezanost na internet
- Arduino nema ugrađen mrežni čip, pa zahteva dodatke
  - Ethernet Shield
  - WiFi čip – ESP32 ili ESP8266
- U nastavku ćemo videti primer kako Arduino može vršiti razmenu poruka u oba smera preko MQTT protokola korišćenjem Ethernet Shield-a i MQTT biblioteke

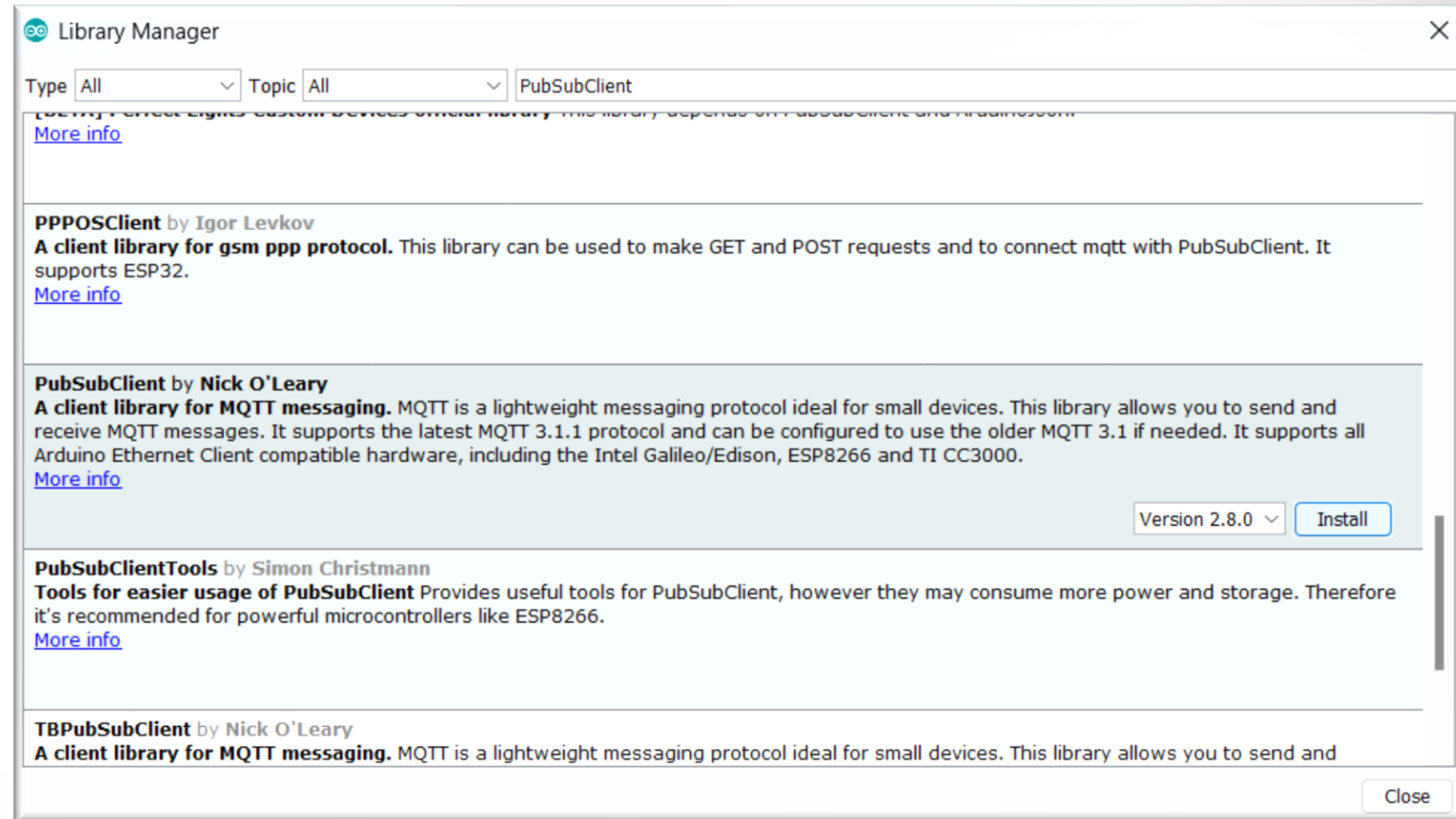
# Pregled MQTT arhitekture – Publish/Subscribe

- Broker
  - Server koji služi kao posrednik u razmeni poruka
  - Ne čuva trajno podatke, samo ih prosleđuje pretplaćenim klijentima
- Topic
  - Predstavlja string na osnovu koga razlikujemo značenje i ulogu poruke
- Publish
  - Slanje poruke svima koji osluškiju dati topic
- Subscribe
  - Mehanizam za prijavljivanje za osluškivanje poruka datog topic-a



# Instalacija biblioteke PubSubClient u Arduino IDE

- Potrebno je prvo instalirati MQTT biblioteku
  - ***Sketch->Include Library->Manage Libraries...***
- Ukucati naziv biblioteke - *PubSubClient*
- Klik na Install
- Na početku programa
  - `#include <PubSubClient.h>`
- MQTT + Ethernet Shield
  - `#include <SPI.h>`
  - `#include <Ethernet.h>`



# Inicijalizacija neophodnih promenljivih

- Podesiti MAC i IP adresu Ethernet Shielda
  - `byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };`
  - `IPAddress ip(192, 168, 1, 160);`
- Podesiti adresu brokera, bez http prefix-a i slash simbola
  - `const char* server = "test.mosquitto.org";`
- Kreirate Ethernet i MQTT objekte
  - `EthernetClient ethClient;`
  - `PubSubClient mqttClient(ethClient);`
- Deklaracija prototipa metoda za callback
  - `void primiPoruka(char* topic, byte* sadrzaj, unsigned int duzina);`



# Setup funkcija

- Započeti Ethernet konekciju
  - `Ethernet.begin(mac, ip);`
- Poželjno sačekati par sekundi da se Ethernet Shield pokrene
  - `delay(3000);`
- Podesiti koji MQTT broker se koristi u razmeni poruka
  - `mqttClient.setServer(server, 1883);`
    - server – adresa brokera
    - port – default za MQTT je 1883
- Konektovati se na broker
  - `mqttClient.connect("klijentId1")`
  - U ovom primeru se konektujemo na javni broker, pa nisu neophodni username i password
  - Za naš slučaj je dovoljno samo identifikator klijenta da se definiše
  - Ova funkcija vraća Boolean vrednost zavisno od uspešnosti konekcije
    - True: ako smo se uspešno konektovali
    - False: nije uspešna konekcija
- Ukoliko je prethodno vraćen true, podesiti callback funkciju za rukovanje događajem prijema poruke
  - `mqttClient.setCallback(primiPoruku);`

# Glavna petlja

- **mqttClient.loop() ;**
  - Poziv .loop() funkcije je neophodan
  - Keep-alive signal
  - Rukovanje primljenim porukama omogućava
- **mqttClient.subscribe("topic1") ;**
  - Prijavljuje se naš Arduino da osluškujemo poruke na temu topic1
  - Tema/topic predstavlja string promenljivu koje je broker svestan
  - Klijenti mogu da publikuju nove poruke i osluškuju tuđe poruke za datu temu
  - Primeri
    - Recimo, za „temperature“ topic možemo slati merenja temperaturnog senzora, pri čemu sam sadržaj poruke predstavlja temperaturu u celzijusima
    - Za topic „relay“ možemo osluškiivati komande da li da upalimo ili ugasimo potrošački uređaj, zavisno od sadržaja poruke – 1 ili 0
- **mqttClient.publish("temperatura", tempC)**
  - Slanje izmerene temperature na topic temperatura

```
void loop()
{
    // Neophodno na početku petlje
    mqttClient.loop();

    // Pretplatiti se na temu "topic1"
    mqttClient.subscribe("topic1");

    // Pokušati slanje sadržaja na temu "topic1"
    if(mqttClient.publish("topic1", "Hello World"))
    {
        Serial.println("Uspešno slanje");
    }
    else
    {
        Serial.println("Neuspešno :(");
    }

    // Da ne preopteretimo server!
    delay(4000);
}
```

# Rukovanje primljenim porukama

- Defnišemo callback funkciju pod nazivom primiPoruku
- Poziva se svaki put kada dođe poruka sa MQTT brokera
- Tri promenljive prosleđujemo
  - Topic
    - Niz karaktera koji označava temu poruke
  - Message payload – sadržaj poruke
    - Sam sadržaj poruke
    - Niz bajtova
  - Dužina poruke
    - Dužina sadržaja poruke u bajtovima

```
void primiPoruku(char* topic, byte* sadrzaj, unsigned int duzina)
{
    //Štampaj topic
    Serial.print("Topic: ");
    Serial.println(topic);

    // Štampaj poruku
    Serial.print("Poruka: ");
    for(int i = 0; i < duzina; i ++)
    {
        Serial.print(char(sadrzaj[i]));
    }


    // Novi red po poruci
    Serial.println("");
}
```



# Javni test broker

- HiveMQ
  - <https://www.hivemq.com/public-mqtt-broker/>
- Demo
  - <https://www.hivemq.com/demos/websocket-client/>
- Skup funkcionalnosti
  - Publikovanje poruka - Publish
  - Prijem poruka - Messages
- Dostupan javno online
  - Broker: broker.hivemq.com
  - TCP Port: 1883
  - Websocket Port: 8000

## Osluškivanje poruka

Color 

QoS

Topic

**Subscribe**

## Konektovanje klijenta

**Connection**

Host  Port  ClientID  **Connect**

Username  Password  Keep Alive  SSL ☐ Clean Session ☐

Last-Will Topic  Last-Will QoS  Last-Will Retain ☐

Last-Will Message

**Publish** **Subscriptions**

**Messages**

## Slanje poruke

**Publish**

Topic  QoS  Retain ☐ **Publish**

Message