JISKA: Jurnal Sistem Informasi Dan Informatika
Vol 4 No 1 (2026): Januari 2026 Hal. 39-43
http://jurnal.unidha.ac.id/index.php/jiska

E-ISSN : 2985-9735
P-ISSN :

Implementasi Protokol Redis Pub/Sub Menggunakan Python untuk Sistem Monitoring Suhu IoT Secara Real-Time

Hadadd Sammir[a], Khairil Hamdi[b], Isnardi[c]
[a]Sistem Komputer, STMIK Jayanusa, hsammir@jayanusa.ac.id
[b]Sistem Komputer, STMIK Jayanusa, khairilhamdi@jayanusa.ac.id
[c]Manajemen Informatika, AMIK Jayanusa, Isnardi.is@gmail.com

Abstract
This research implements the Redis Publish/Subscribe (Pub/Sub) mechanism using the Python programming language for real-time transmission of IoT temperature sensor data. The primary focus of this study is to address latency challenges in data distribution for logging and alerting system requirements. The system is designed with an architecture where temperature data is published to a Redis channel and simultaneously received by multiple subscribers. One subscriber unit is responsible for recording data into a database for historical analysis, while another unit validates temperature thresholds to trigger instant alerts upon detecting anomalies. Test results demonstrate that the use of Redis Pub/Sub effectively achieves decoupling between data senders and receivers, thereby enhancing system scalability. This architecture proves capable of distributing information with low latency and high efficiency. This study concludes that Redis Pub/Sub is a reliable solution for IoT monitoring systems that require rapid response and seamless data synchronization between monitoring functions and preventive actions.
Keywords: *IoT, Redis Pub/Sub, Python, Temperature Sensor, Real-Time.*

Abstrak
Penelitian ini mengimplementasikan mekanisme Redis Publish/Subscribe (Pub/Sub) menggunakan bahasa pemrograman Python untuk pengiriman data suhu sensor IoT secara real-time. Fokus utama penelitian adalah mengatasi tantangan latensi dalam distribusi data untuk kebutuhan pencatatan (logging) dan sistem notifikasi (alerting). Sistem dirancang dengan arsitektur di mana data suhu dipublikasikan ke kanal Redis, yang kemudian diterima secara simultan oleh beberapa subscriber. Satu unit subscriber berfungsi mencatat data ke basis data untuk analisis historis, sementara unit lainnya memvalidasi ambang batas suhu guna memicu peringatan instan saat terjadi anomali. Hasil pengujian menunjukkan bahwa penggunaan Redis Pub/Sub secara efektif melakukan dekopling (decoupling) antara pengirim dan penerima data, sehingga meningkatkan skalabilitas sistem. Arsitektur ini terbukti mampu mendistribusikan informasi dengan latensi rendah dan efisiensi tinggi. Penelitian ini menyimpulkan bahwa Redis Pub/Sub merupakan solusi andal untuk sistem monitoring IoT yang membutuhkan respons cepat dan sinkronisasi data antara fungsi pemantauan serta tindakan preventif.
Kata Kunci: IoT, Redis Pub/Sub, Python, Sensor Suhu, Real-Time.

PENDAHULUAN

Pesatnya perkembangan teknologi *Internet of Things* (IoT) telah memungkinkan pemantauan parameter fisik secara otomatis dan berkelanjutan di berbagai sektor. Salah satu parameter yang paling krusial adalah suhu, terutama pada industri yang memerlukan kontrol ketat seperti rantai dingin (*cold chain*) medis dan sistem pendingin pusat data [1]. Tantangan utama dalam ekosistem ini bukan sekadar pengumpulan data, melainkan bagaimana mentransportasikan data tersebut secara *real-time* dari sensor ke berbagai layanan konsumen tanpa menimbulkan latensi yang signifikan atau hambatan pada *throughput* data [2].

Transportasi data dalam IoT sering kali menghadapi kendala ketika sistem harus menjalankan tugas ganda secara simultan: pencatatan data ke basis data untuk analisis historis (*logging*) dan pemrosesan logika instan untuk peringatan dini (*alerting system*) [3]. Jika data dikirimkan menggunakan protokol komunikasi sinkron tradisional, keterlambatan pada satu proses pengolahan dapat menghambat keseluruhan alur kerja. Sebagai contoh, dalam logistik farmasi, keterlambatan informasi anomali suhu selama beberapa detik dapat memicu kerusakan pada komoditas yang sensitif terhadap termal [4].

Untuk mengatasi tantangan tersebut, diperlukan arsitektur komunikasi berbasis pesan yang mampu melakukan dekopling (*decoupling*) antara pengirim dan penerima data. Redis Publish/Subscribe (Pub/Sub) merupakan solusi yang sangat efisien karena beroperasi secara *in-memory*, yang memungkinkan distribusi pesan dengan latensi mendekati nol [5]. Dengan mekanisme ini, data suhu yang dipublikasikan oleh sensor melalui bahasa pemrograman Python dapat segera didistribusikan secara paralel ke berbagai modul *subscriber* tanpa saling menunggu.

Beberapa studi terbaru menunjukkan bahwa Python tetap menjadi pilihan utama dalam pengembangan IoT karena fleksibilitasnya dalam integrasi pustaka *asynchronous* yang mendukung skalabilitas sistem [6]. Namun, optimasi penggunaan Redis Pub/Sub untuk menjaga konsistensi data antara fungsi *logging* dan *alerting* pada

JISKA: Jurnal Sistem Informasi Dan Informatika
Vol 4 No 1 (2026): Januari 2026 Hal. 39-43
http://jurnal.unidha.ac.id/index.php/jiska
E-ISSN : 2985-9735
P-ISSN :

perangkat IoT dengan sumber daya terbatas masih menjadi topik yang relevan untuk dikaji. Oleh karena itu, penelitian ini bertujuan untuk mendemonstrasikan efektivitas pemanfaatan Redis Pub/Sub dalam mempercepat transportasi data suhu pada sistem IoT berbasis Python.

METODE PENELITIAN

Sistem yang dirancang berbasis *prototype* yang berfokus kepada pengujian penggunaan Redis pub/sub untuk mentransmisikan data dari sensor suhu ke perangkat pemroses data. *Prototype* edge adalah aplikasi python yang mengirimkan data suhu yang dimbil dari sensor suhu DHT22. *Edge* node menggunakan Esp32 dengan dukungan Micropython.

2.1. Metode

Persiapan sistem IoT ditampilkan pada daftar di bawah ini:

1. Microcontroller ESP32.
2. Sensor DHT11.
3. Sistem micropython-redis yang diungggah ke esp32.
4. Laptop sebagai *server* redis dan modul *subscriber*.

Langkah-langkah edge sensor (esp32) dalam mengirimkan data suhu adalah sebagai berikut.

1. Konfigurasi Wifi.
2. Konfigurasi sensor dan redis (menentukan port sensor dan koneksi ke server redis).
3. Melakukan koneksi micropython ke Redis.
4. Perulangan utama membaca sensor dan melakukan publikasi.

Edge pemroses menggunakan aplikasi Python dengan modul redis yang berlangganan pada kanal "sensor_suhu". *Edge* pemroses bekerja dengan langkah-langkah berikut ini.

1. Menginisialisasi modul redis dan melakukan koneksi.
2. Berlangganan pada kanal "sensor_suhu".
3. Perulangan utama (*main loop*) untuk menerima dan menampilkan data suhu.

HASIL DAN PEMBAHASAN

Implemetasi redis pub/sub pada IoT berbasis microcontroller esp32 melibatkan penggunakan micropython dan modul micropython-redis yang diunggah ke modul esp32. Micropython menyediakan antar muka pemrograman dengan dialek python sekaligus menyediakan dukungan pustaka pemrograman yang luas. Penelitian ini memanfaatkan fungsionalitas internet (wifi) dari esp32 dan dukungan library micropython-redis.

Potongan kode pada gambar 1 di bawah ini memperlihatkan inisialisasi modul dan konfigurasi awal koneksi internet. Kode melakukan *importing* modul yang dibutuhkan, menetapkan setelan SSID wifi dan *password*, serta melakukan koneksi ke jaringan wifi tersebut.

```
import network
import time
import dht
from machine import Pin
import uredis

# 1. Konfigurasi Wi-Fi
SSID = 'hsammir'
PASSWORD = 'xxxxxxxx'

def connect_wifi():
    wlan = network.WLAN(network.STA_IF)
    wlan.active(True)
    if not wlan.isconnected():
        print('Connecting to network...')
        wlan.connect(SSID, PASSWORD)
        while not wlan.isconnected():
            pass
    print('Network config:', wlan.ifconfig())
```

Gambar 1. Inisialisasi dan Konfigurasi Awal

JISKA: Jurnal Sistem Informasi Dan Informatika
Vol 4 No 1 (2026): Januari 2026 Hal. 39-43
http://jurnal.unidha.ac.id/index.php/jiska
E-ISSN : 2985-9735
P-ISSN :

Langkah berikutnya pengaturan pin sensor yang akan dibaca serta konfigurasi alamat dan port server redis beserta nama kanal yang digunakan untuk publikasi seperti yang diperlihatkan pada gambar 2 berikut ini.

```
sensor = dht.DHT22(Pin(15))
REDIS_HOST = '192.168.1.10'
REDIS_PORT = 6379
CHANNEL = 'sensor_suhu'
```

Gambar 2. Pengaturan Pin Sensor dan Pengaturan Redis

Server redis berada pada alamat 192.168.1.10 dengan port standar redis yaitu 6379. Sementara data sensor yang akan ditangkap terhubung pada GPIO pin nomor 15. Konfigurasi selanjutnya adalah publikasi data sensor suhu akan menggunakan kanal pub/sub redis dengan nama "sensor_suhu".

Aktifitas selanjutnya adalah aktifitas utama, yaitu mempublikasikan data suhu. Program mengaktifkan sensor, membaca data suhu, dan mempublikasikan data suhu tersebut pada kanal "sensor_suhu" dalam interval waktu 5 detik. Gambar 3 di bawah ini menampilkan potongan program python yang melakukan aktifitas tersebut.

```
while True:
    try:
        sensor.measure()
        temp = sensor.temperature()

        data = "Temp: {:.1f}C".format(temp)
        print("Publishing:", data)

        r.publish(CHANNEL, data)

    except OSError as e:
        print('Gagal membaca sensor.')

    time.sleep(5)
```

Gambar 3. Alur Pembacaan Sensor dan Publikasi Data Sensor

Edge pemroses dapat dibangun menggunakan ekosistem pemrograman yang mendukung komunikasi dengan *server* redis, namun penelitian ini menggunakan pemrograman python karena kemudahan penggunaanya. *Edge* pemroses pada penelitian ini hanya menguji komnikasi pub/sub sehingga hanya akan menampilkan data suhu yang dikirimkan oleh *edge* sensor. Gambar 4 di bawah ini menampilkan kode yang digunakan oleh *edge* pemroses.

JISKA: Jurnal Sistem Informasi Dan Informatika
Vol 4 No 1 (2026): Januari 2026 Hal. 39-43
http://jurnal.unidha.ac.id/index.php/jiska
E-ISSN : 2985-9735
P-ISSN :

```python
r = redis.Redis(host='localhost', port=6379, db=0, decode_responses=True)

def subscribe_sensor_data():
    channel = "sensor_suhu"

    pubsub = r.pubsub()
    pubsub.subscribe(channel)

    print(f"Berlangganan ke {channel}. Menunggu data...")

    for message in pubsub.listen():
        if message['type'] == 'message':
            print(f"Data Diterima -> {message['data']}, ")

if __name__ == "__main__":
    try:
        subscribe_sensor_data()
    except KeyboardInterrupt:
        print("Subscriber dihentikan.")
```

Gambar 4. Alur Algoritma *Edge* Pemroses

Edge pemroses adalah program yang ditulis dalam bahasa Python. Program tersebut melakukan koneksi ke *server* redis dan berlangganan pada kanal "sensor_suhu", kanal yang sama yang digunakan oleh *edge* sensor untuk mempublikasikan data suhu. Program mengeksekusi fungsi listen() yang bertugas untuk menyimak semua data yang dipublikasikan. Perulangan dilakukan untuk menampilkan data suhu yang dikirimkan.

Edge pemroses akan menampilkan informasi suhu setiap kali data suhu tersebut dipublikasikan. Tampilan *edge* pemroses ditampilkan pada gambar 5 di bawah ini.

```
Berlangganan ke sensor_suhu. Menunggu data...
Data Diterima -> Temp: 20.6C,
Data Diterima -> Temp: 20.2C,
Data Diterima -> Temp: 20.5C,
```

Gambar 5. Tampilan *Edge* Pemroses

SIMPULAN

Penelitian ini berhasil membuktikan metoda komunikasi antara perangkat IoT dengan *server* redis dan *edge* pemroses. *Edge* pemroses dapat berupa PC atau aplikasi berbasis web yang mampu memproses data dari sensor untuk visualisasi, sistem pemberitahuan (*alerting*) serta pengolahan data. Mekanisme pub/sub redis sebagai perantara memungkinkan banyak *edge* pemroses dapat mengakses data sensor tanpa membebani kinerja jaringan dan komputasi *edge* sensor. Setiap *edge* pemroses dapat memproses data secara independen sesuai dengan fungsinya masing-masing.

Sifat alami dari pub/sub adalah komunikasi satu arah, di mana pengiriman data hanya dimungkinkan dari *publisher* ke *subscriber*. Penelitian selanjutnya bisa memanfaatkan websocket untuk komunikasi dua arah antara *edge* sensor dan *edge* pemroses.

UCAPAN TERIMAKASIH

Teima kasih yang sebesarnya kami berikan kepada berbagai pihak yang mendukung kami melakukan penelitian dan penulisan artikel ini. Kepada pihak Yayasan dan rekan sejawat untuk dukungan moral dan diskusi teknis serta kepada keluarga yang selalu memberikan semangat bagi kami dalam setiap berkegiatan.

DAFTAR PUSTAKA

[1] M. A. Rahaman, C. Chakraborty, and M. R. Islam, "Real-time Patient Monitoring System using IoT and Cloud Computing for Smart Healthcare," J. Med. Syst., vol. 48, no. 1, pp. 12–25, 2024.

[2] S. K. Sood and S. Rani, "A Secure and Scalable IoT Architecture for Smart Temperature Monitoring in Cold Chain Logistics," IEEE Trans. Ind. Inform., vol. 19, no. 3, pp. 2845–2854, 2023.

E-ISSN : 2985-9735
P-ISSN :

[3] R. Kumar and A. Singh, "Comparative Analysis of Messaging Protocols for Real-Time IoT Applications," Int. J. Commun. Syst., vol. 35, no. 14, p. e5281, 2022.
[4] J. Wang and et al., "Low-Latency Data Transmission Mechanism for IoT-Based Cold Chain Monitoring Systems," IEEE Internet Things J., vol. 11, no. 2, pp. 1540–1552, 2024.
[5] L. Zhang and Y. Chen, "Optimization of In-Memory Data Grids for Real-Time IoT Data Processing using Redis," Future Gener. Comput. Syst., vol. 141, pp. 310–322, 2023.
[6] H. Nguyen and T. Pham, "Scalability of Python-Based Microservices in IoT Environments: A Performance Study," Softw. Pract. Exp., vol. 55, no. 4, pp. 889–910, 2025.