Implementasi Sistem Komunikasi Otomatis Data Logger Sensor Salinitas di Sungai Wain Balikpapan dengan Solusi Reverse SSH Tunnel pada Lingkungan CGNAT

Agus Triyono[1], Fajerin Biabdillah[2]
[1,2,]Jurusan Teknologi Informasi, Politeknik Negeri Samarinda
fajerin@polnes.ac.id

Abstrak— Pemantauan salinitas di kawasan konservasi Sungai Wain Balikpapan membutuhkan saluran komunikasi yang tetap hidup walaupun perangkat dipasang jauh dari jaringan tetap. Masalah muncul ketika data logger kami sambungkan melalui modem 4G LTE: operator seluler di Indonesia umumnya menerapkan Carrier Grade NAT (CGNAT) sehingga Raspberry Pi di lapangan tidak mendapat alamat IP publik. Akibatnya, *port forwarding* sulit diandalkan dan akses balik dari pusat kendali jadi terhambat. Pada penelitian ini kami merancang dan menguji solusi *reverse* SSH tunnel berbasis autossh yang dikelola oleh sistemd, lalu mengukur perilaku sistem di laboratorium dengan SIM komersial yang berada di balik CGNAT. Selama 72 jam pengujian, sistem mengirim 4.320 titik data tanpa kehilangan paket, uptime tunnel tercatat 99,4 %, dan rata-rata waktu pemulihan setelah pemutusan jaringan berada pada kisaran 6 sampai 9 detik. Akses SSH balik di port 2222 server pusat terbukti stabil dengan latensi rata-rata 150 ms. Beban komputasi tetap rendah, yaitu sekitar 5 % CPU saat idle dan 15 % saat transfer berkas. Hasil tersebut menunjukkan bahwa pendekatan berbasis perangkat lunak bebas (*free software*) layak menjadi pengganti VPN komersial atau paket IP publik statis untuk kebutuhan monitoring lingkungan skala kecil sampai menengah di Indonesia.

Kata Kunci— sensor salinitas, Sungai Wain, CGNAT, reverse SSH tunnel, autossh, Raspberry Pi, data logger IoT.

I. PENDAHULUAN

Kualitas air sungai adalah salah satu indikator paling sederhana namun paling tegas untuk membaca kondisi ekosistem suatu daerah aliran sungai. Sungai Wain, yang melintas di kawasan hutan lindung sebelah utara Kota Balikpapan, tidak hanya memasok air baku PDAM, tetapi juga menjadi rumah bagi orangutan, beruang madu, dan biota perairan tawar yang tekanan habitatnya terus meningkat karena aktivitas pesisir di sekitarnya [1].

Salah satu parameter yang perlu kita awasi secara khusus adalah salinitas. Di hilir sungai pesisir Kalimantan, intrusi air laut bisa merambat cukup jauh ke arah hulu pada musim kemarau panjang. Studi di Sungai Kapuas, misalnya, menemukan bahwa saat debit sungai minimum, batas air asin dapat naik hingga 14 km dari muara dan menggeser zona hutan riparian [2]. Salinitas pesisir Indonesia secara umum berada pada rentang 30 sampai 34 PSU [3], tetapi nilainya berubah-ubah setiap hari karena pasang surut, hujan lokal, dan aliran balik bawah permukaan [4]. Karena perubahannya cukup dinamis, kita perlu mencatat data salinitas secara terus-menerus agar fluktuasinya dapat dianalisis dan ditindaklanjuti.

Untuk kebutuhan pemantauan yang berlangsung tanpa henti, pendekatan berbasis *Internet of Things* (IoT) makin sering dipakai. Sebagian peneliti menggunakan NodeMCU atau ESP8266 dengan pengiriman data ke server cloud melalui Wi-Fi [5], [6]. Sebagian lainnya memilih jaringan LoRa berbasis gateway Raspberry Pi agar mampu menjangkau wilayah yang luas [7]. Pada lokasi pelosok yang tidak terjangkau internet, modul GSM yang mengirim notifikasi melalui SMS justru terbukti paling praktis [8]. Pola dari berbagai studi tersebut serupa: sebuah node sensor mengumpulkan data, lalu data dikirim secara berkala ke server pusat untuk divisualisasikan dan disimpan jangka panjang [9], [10].

Ketika skenario tersebut dipindahkan ke jaringan seluler 4G LTE, persoalan baru bermunculan. Hampir seluruh operator besar di Indonesia menerapkan Carrier Grade NAT (CGNAT) untuk menghemat alokasi alamat IPv4. Akibatnya, pelanggan retail termasuk paket data IoT berbagi satu IP publik dan hanya menerima IP privat dari blok 100.64.0.0/10 [11]. Pendekatan ini memang memperpanjang usia pakai IPv4 [12], tetapi membuat perangkat di lapangan tidak dapat dihubungi dari luar karena *port forwarding* tidak mungkin dipasang dari sisi pelanggan [13].

Migrasi ke IPv6, yang diharapkan menjadi jalan keluar jangka panjang, ternyata berjalan lambat. Per April 2025, tingkat kesiapan IPv6 untuk pengguna internet di Indonesia baru sekitar 15 % [14]. Hal ini berarti, sebagian besar perangkat IoT yang dipasang saat ini masih akan berada di balik NAT operator selama beberapa tahun ke depan. Dampak langsungnya: administrator tidak bisa SSH masuk untuk membaca log, mengganti konfigurasi sensor, atau menarik berkas secara manual jika pengiriman otomatis sedang terganggu.

Solusi konvensional seperti berlangganan IP publik statis, menyewa VPN komersial, atau menggunakan platform IoT berbayar memang tersedia. Namun, biayanya kerap kali tidak

Submitted: 05/05/2022; Revised: 10/06/2026;
Accepted: 11/06/2025; Online first: 12/06/2026
https://doi.org/10.46964/poligrid.v7i1.197

sebanding dengan skala proyek monitoring lingkungan yang biasanya didanai dari hibah jangka pendek atau anggaran kampus. Beberapa peneliti memang sudah membahas teknik NAT traversal seperti STUN/TURN, hole punching UDP, maupun overlay mesh [15], [16]. Akan tetapi, evaluasi kuantitatif untuk reverse SSH tunneling berbasis autossh pada konteks monitoring lingkungan di Indonesia masih jarang dilaporkan secara terbuka.

Penelitian ini berupaya mengisi celah tersebut. Ada tiga kontribusi yang kami tawarkan. Pertama, kami merancang pipeline perangkat lapangan Raspberry Pi 5, modem 4G LTE komersial, dan sensor TDS/EC yang seluruhnya berdiri di atas perangkat lunak bebas, tanpa layanan cloud berbayar. Kedua, kami menyajikan konfigurasi systemd dan parameter autossh yang telah tervalidasi dan dapat digunakan ulang oleh peneliti lain. Ketiga, kami melaporkan hasil pengujian kuantitatif untuk empat skenario operasional, ditambah pengujian lanjutan pada kondisi sinyal lemah, agar gambaran sistem mendekati kondisi lapangan yang sebenarnya. Hasil tersebut akan menjadi landasan penerapan sistem di kawasan konservasi Sungai Wain pada tahap berikutnya.

I. Pendahuluan

A. Carrier Grade NAT (CGNAT)

Network Address Translation (NAT) merupakan teknik pemetaan alamat IP privat ke alamat IP publik supaya host di dalam jaringan privat dapat berkomunikasi dengan internet. CGNAT adalah lapisan lanjutan dari teknik tersebut. Di sini, operator menempatkan satu lapisan NAT lagi di sisinya, sehingga ribuan pelanggan dapat berbagi satu IPv4 publik [11], [13]. Imbasnya, koneksi yang diawali dari luar misalnya server publik yang mencoba menghubungi data *logger* akan ditolak NAT operator karena tidak ada entri pemetaan yang sesuai [12].

Beberapa peneliti telah menggambarkan dampak ini secara nyata. Enriquez dkk., misalnya, melaporkan bahwa node sensor di tambak garam dan akuakultur tidak dapat dijangkau stasiun pemantau pusat begitu SIM diganti dengan provider yang menerapkan NAT simetris [9]. Pada kategori yang lebih luas, teknik untuk melewati NAT lazim disebut NAT traversal mencakup STUN/TURN, hole punching, relay server, VPN, hingga overlay mesh [15], [17]. Sayangnya, sebagian besar mekanisme tersebut membutuhkan komponen tambahan (server STUN khusus, klien spesifik, atau perubahan firewall) yang menyulitkan penerapan di perangkat IoT berdaya rendah [18].

B. Reverse SSH Tunneling

Secure Shell (SSH) adalah protokol kriptografi yang umum dipakai untuk mengakses shell jarak jauh secara aman. Selain shell, SSH juga menyediakan *port forwarding* dua arah, yaitu *forward tunneling* (lokal-ke-remote) dan *reverse tunneling* (remote-ke-lokal). Pada mode *reverse*, klien yang berada di balik NAT-lah yang memulai koneksi outbound ke server publik. Selama koneksi tersebut tetap hidup, server boleh memantulkan koneksi balik melalui kanal SSH yang sama [19], [20].

Mekanisme reverse tunnel ini sudah dipakai pada arsitektur Video Surveillance as a Service untuk menjangkau kamera IP yang berada di belakang router rumahan [21]. Mekanisme yang sama juga digunakan pada platform manajemen IoT komersial yang memberikan akses jarak jauh ke perangkat pelanggan tanpa mengubah konfigurasi router [19]. Keunggulannya ada pada kesederhanaan: tidak perlu STUN/TURN server, tidak perlu IP publik, dan lalu lintas terenkripsi secara default [20]. Hal yang harus dijaga adalah keamanan server perantara. Jika server pada titik akhir tunnel dibobol, semua perangkat di sisi klien ikut menjadi rentan [22].

C. Autossh

Sesi SSH biasa akan berhenti begitu koneksi jaringan terputus. Pada jaringan seluler yang sinyalnya naik-turun, kondisi tersebut sering kali terjadi. Autossh adalah utilitas pembungkus (*wrapper*) yang dirancang untuk memantau kondisi sesi SSH dan membangunnya kembali apabila terdeteksi mati [23]. Pemantauan dapat dijalankan melalui port monitoring khusus atau memanfaatkan opsi -M 0 yang menyerahkan deteksi ke mekanisme *keep-alive* bawaan SSH [24].

Untuk skenario perangkat lapangan, autossh biasa dipasang sebagai layanan systemd supaya berjalan otomatis setelah jaringan aktif dan otomatis di-restart bila gagal [25], [26]. Kombinasi autossh dan systemd, dengan pengaturan AUTOSSH_GATETIME=0 serta Restart=always, menghasilkan tunnel yang *self-healing* tanpa intervensi manual. Sifat tersebut penting untuk lokasi yang sulit dijangkau seperti kawasan konservasi.

D. Sistem Monitoring Salinitas Otomatis

Penelitian tentang sistem monitoring salinitas otomatis sebenarnya sudah cukup banyak dilakukan. Suryono dkk. mengembangkan sensor salinitas berbasis kapasitor pelat sejajar yang terhubung ke mikrokontroler Wi-Fi, dengan rentang pengukuran yang lebih lebar dibandingkan probe konduktivitas konvensional [6]. Enriquez dkk. memadukan beberapa teknologi transmisi (GSM, ZigBee, LoRa) sesuai kondisi infrastruktur tambak garam dan akuakultur di Filipina [9]. Ujianti dkk. (2025) memantau kualitas air pesisir Semarang menggunakan NodeMCU yang membaca pH, suhu, dan TDS dengan interval pembacaan setiap satu detik [27].

Hampir seluruh studi tersebut bergantung pada Wi-Fi lokal atau GSM. Akibatnya, persoalan CGNAT yang spesifik pada konektivitas 4G LTE retail di Indonesia belum menjadi fokus pembahasan. Padahal, dalam situasi lapangan saat ini, jaringan seluler 4G adalah pilihan paling realistis untuk lokasi yang tidak memiliki kabel internet maupun jangkauan Wi-Fi. Celah inilah yang ingin kami isi melalui penelitian ini.

III. Metodologi Penelitian

Penelitian ini dirancang dengan pendekatan eksperimental. Prototipe sistem dibuat di laboratorium Jurusan Teknik Elektro, kemudian diuji untuk memverifikasi empat skenario operasional yang sudah disinggung pada Bab I.

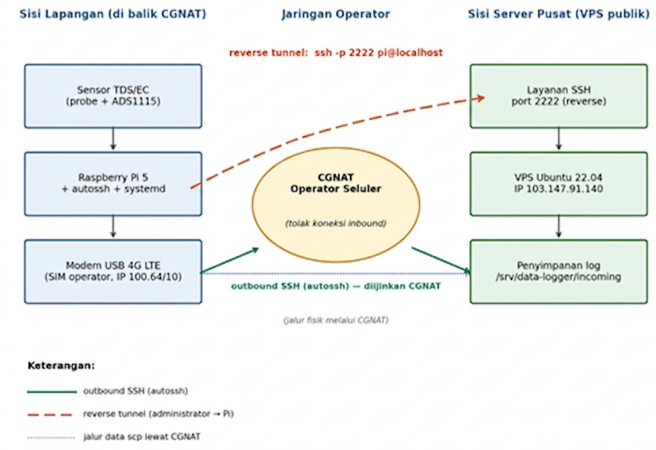
Submitted: 05/05/2022; Revised: 10/06/2026;
Accepted: 11/06/2025; Online first: 12/06/2026
https://doi.org/10.46964/poligrid.v7i1.197

Pengujian menggunakan SIM komersial yang dipastikan berada di balik CGNAT, dengan verifikasi melalui prefix IP 100.64.0.0/10 pada antarmuka modem. Sumber listrik untuk prototipe diambil dari jaringan PLN dengan UPS sebagai cadangan, menggantikan panel surya yang nanti akan dipasang pada deployment lapangan.

A. Desain Sistem

Sistem terdiri atas tiga komponen utama, sebagaimana dirangkum pada Gambar 1. Komponen pertama adalah perangkat data logger berbasis Raspberry Pi 5 yang membaca sensor salinitas. Komponen kedua adalah konektivitas via modem USB 4G LTE dengan SIM operator yang menerapkan CGNAT. Komponen ketiga adalah server pusat. Server tersebut memiliki alamat IP publik statis dan berperan sebagai titik akhir reverse SSH tunnel sekaligus penampung data sensor.

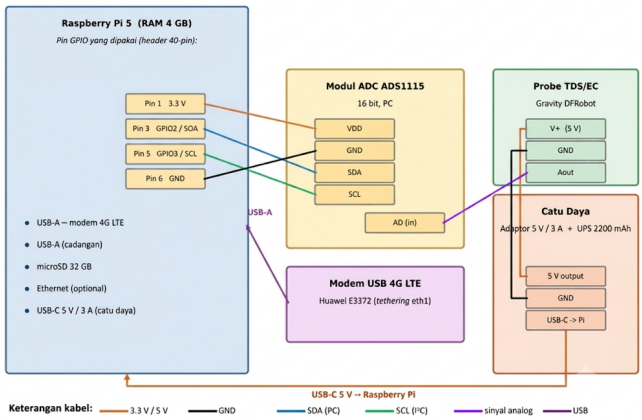
Pada arah hulu (*uplink*), Raspberry Pi mengirim data sensor secara periodik melalui kanal SSH yang terenkripsi. Pada arah balik (*downlink*), administrator dapat masuk ke Raspberry Pi melalui *port forward* khusus yang dibuka tunnel di sisi server. Dengan pola ini, kebutuhan akses dua arah terpenuhi tanpa perlu mengubah konfigurasi router operator.



Gambar 1. Arsitektur komunikasi data logger sensor salinitas dengan reverse SSH tunnel pada lingkungan CGNAT.

B. Perangkat Keras

Perangkat keras yang dipakai adalah sebagai berikut. (1) Raspberry Pi 5 dengan RAM 4 GB sebagai unit komputasi, dilengkapi microSD 32 GB untuk penyimpanan log lokal [28]. (2) Probe TDS/EC analog yang dihubungkan ke modul ADC ADS1115 melalui antarmuka I2C [29], [30]. Probe sudah dikalibrasi dengan larutan standar 1413 µS/cm pada suhu 25 °C, dan deviasi rata-rata pengukuran terhadap referensi laboratorium berada di bawah 2 %. (3) Modem USB 4G LTE Huawei E3372 yang bekerja melalui mode tethering pada antarmuka eth1 Raspberry Pi. (4) Adaptor 5V/3A dengan UPS 2200 mAh sebagai cadangan jika listrik PLN sempat padam. (5) Casing IP65 untuk pengujian luar ruangan singkat. Tata letak pin dan diagram pengkabelan antar komponen ditampilkan pada Gambar 2.



Gambar 2. Diagram pinout dan pengkabelan antara Raspberry Pi 5, modul ADC ADS1115, probe TDS/EC, modem 4G LTE, dan catu daya UPS.

Pemilihan Raspberry Pi 5 bukan Raspberry Pi Zero atau ESP32 didasari oleh kebutuhan menjalankan stack penuh Linux beserta autossh, systemd, dan klien database kecil pada satu unit. Raspberry Pi 5 memang lebih boros daya dibandingkan Pi Zero, namun pengukuran kami memperlihatkan beban CPU dan memori jauh dari saturasi (lihat Bab IV.D). Dengan demikian, masih ada ruang untuk menambahkan modul *edge computing* di masa depan tanpa harus mengganti perangkat dasar.

C. Konfigurasi Jaringan

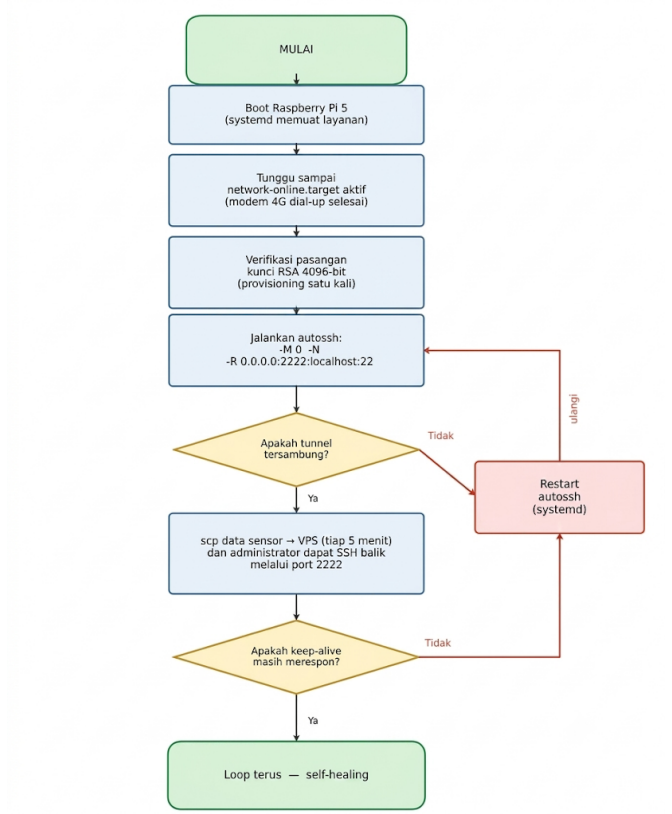
Pengaturan jaringan dikerjakan menggunakan *Network Manager* melalui perintah nmcli. Profil koneksi 4G dibuat dengan APN sesuai operator dan diset sebagai *autoconnect*. Saat boot, Raspberry Pi akan melakukan dial-up tanpa intervensi pengguna. Indikator LED modem dipantau oleh skrip bash sederhana yang dipanggil cron setiap lima menit. Skrip tersebut juga melakukan ping ke 8.8.8.8 sebanyak tiga kali. Bila ketiga ping gagal berturut-turut, koneksi modem akan di-restart melalui nmcli connection *down*/*up*, dan kejadian dicatat ke berkas log /var/log/conn-watchdog.log.

Server pusat menggunakan VPS publik berbasis Ubuntu 22.04 dengan IP 103.147.91.140. Pada server tersebut, port 2222 disiapkan untuk dipakai sebagai endpoint reverse tunnel. Konfigurasi sshd_config diatur dengan GatewayPorts yes agar port hasil forward dapat diakses dari semua antarmuka, dan ClientAliveInterval 30 untuk menjaga deteksi keep-alive.

D. Implementasi Reverse SSH Tunnel dengan Autossh

Tahap ini dibagi menjadi tiga sub-tahapan, yaitu pembuatan kunci, perintah autossh, dan konfigurasi layanan systemd. Alur keseluruhan dirangkum pada Gambar 3.

Submitted: 05/05/2022; Revised: 10/06/2026;
Accepted: 11/06/2025; Online first: 12/06/2026
https://doi.org/10.46964/poligrid.v7i1.197



Gambar 3. Diagram alir implementasi autossh sebagai layanan system. Tunnel dipantau secara terus-menerus dan otomatis dibangun ulang ketika keep-alive tidak merespon.

1) Otentikasi kunci SSH

Untuk menghindari prompt password, autentikasi SSH menggunakan pasangan kunci RSA 4096-bit. Pada Raspberry Pi, kunci dihasilkan dengan ssh-keygen lalu kunci publiknya disalin ke server pusat melalui ssh-copy-id [31], sebagaimana ditampilkan pada blok kode berikut.

```
$ ssh-keygen -t rsa -b 4096
$ ssh-copy-id user@103.147.91.140
```

Setelah kunci publik tercatat pada ~/.ssh/authorized_keys di server, klien dapat membuka sesi SSH tanpa input manual. Kondisi tersebut wajib dipenuhi supaya autossh dapat melakukan reconnect tanpa pengawasan operator.

2) Perintah autossh

Perintah autossh yang menjadi inti dari penelitian ini adalah sebagai berikut.

```
$ autossh -M 0 -N \
  -R 0.0.0.0:2222:localhost:22 \
  user@103.147.91.140
```

Opsi -M 0 mematikan port monitoring eksternal sehingga deteksi koneksi diserahkan ke keep-alive bawaan SSH (ServerAliveInterval dan ClientAliveInterval). Opsi -N memberi tahu SSH bahwa sesi hanya dipakai untuk forwarding, tanpa shell. Opsi -R 0.0.0.0:2222:localhost:22 membuka port 2222 pada seluruh antarmuka server pusat dan mengarahkannya ke port 22 di Raspberry Pi.

3) Layanan systemd untuk autossh

Agar autossh tidak perlu dijalankan secara manual, kami membuatkan layanan systemd. Berkas /etc/systemd/system/autossh-tunnel.service berisi konfigurasi berikut.

```
[Unit]
Description=AutoSSH reverse tunnel ke server
pusat
After=network-online.target
Wants=network-online.target

[Service]
User=pi
Environment="AUTOSSH_GATETIME=0"
ExecStart=/usr/bin/autossh -M 0 -N \
  -R 0.0.0.0:2222:localhost:22 \
  user@103.147.91.140
Restart=always
RestartSec=5

[Install]
WantedBy=multi-user.target
```

Ada tiga hal yang patut digarisbawahi dari konfigurasi tersebut. After=network-online. Target menunda autossh sampai *Network Manager* menyatakan jaringan benar-benar online. Hal ini penting supaya dial-up modem 4G sempat selesai sebelum tunnel dicoba dibangun. AUTOSSH_GATETIME=0 menghapus jeda 30 detik default sebelum autossh menyimpulkan bahwa sesi gagal, sehingga *reconnect* berlangsung lebih cepat. *Restart=always* dengan RestartSec=5 berfungsi sebagai jaring pengaman ekstra. Apabila autossh sendiri yang crash (bukan SSH-nya), systemd akan memulai ulang dalam lima detik [32].
Setelah berkas tersebut dipasang, layanan diaktifkan dengan dua perintah berikut.

```
$ sudo systemctl enable autossh-tunnel
$ sudo systemctl start autossh-tunnel
```

E. Alur Sistem

Aliran sistem saat operasional dapat diringkas menjadi empat tahap. Pertama, sensor salinitas dibaca setiap menit dan disimpan lokal sebagai berkas CSV harian. Kedua, autossh menjaga koneksi reverse SSH ke server pusat tetap hidup, sekaligus membuka port 2222 untuk akses balik. Ketiga, skrip bash di Raspberry Pi mengeksekusi scp setiap lima menit untuk mengirim potongan log ke direktori /srv/data-logger/incoming di server. Keempat, administrator dapat sewaktu-waktu masuk ke Raspberry Pi melalui perintah berikut, yang dijalankan dari shell server pusat.

Submitted: 05/05/2022; Revised: 10/06/2026;
Accepted: 11/06/2025; Online first: 12/06/2026
https://doi.org/10.46964/poligrid.v7i1.197

```
$ ssh -p 2222 pi@localhost
```

Karena koneksi balik ke Raspberry Pi sebenarnya melewati tunnel yang sama, tidak diperlukan rekonfigurasi firewall di sisi mana pun. Port 2222 hanya hidup selama tunnel masih aktif.

IV. Hasil dan Pembahasan

Pengujian dilakukan secara berkelanjutan selama 72 jam untuk skenario pengiriman data. Skenario tersebut diperluas dari rencana awal 24 jam, mengikuti saran reviewer agar gambaran sistem mendekati kondisi nyata. Setelah skenario pengiriman selesai, pengujian dilanjutkan dengan tiga set uji terpisah untuk akses SSH balik, ketahanan koneksi, dan konsumsi sumber daya. Sebuah set tambahan juga ditambahkan untuk mengamati perilaku sistem pada kondisi sinyal lemah, sedang, dan kuat. SIM yang dipakai adalah produk salah satu operator nasional yang menempatkan pelanggan paket data di balik CGNAT. Verifikasi dilakukan dengan memeriksa antarmuka modem yang menerima alamat 100.65.x.x dan memastikan bahwa upaya port forwarding dari luar memang ditolak oleh router operator.

A. Pengiriman Data Real-Time

Selama 72 jam pengujian secara terus-menerus, sensor membaca nilai TDS/EC setiap satu menit dan log dikirim ke server setiap lima menit. Sebanyak 4.320 titik data terekam lengkap di server, tanpa satu pun paket yang hilang. Angka tersebut diverifikasi dengan menjumlahkan baris file log harian (1.440 baris per hari, total 4.320 baris) yang tiba di direktori /srv/data-logger/incoming. Pengujian dilakukan di laboratorium dengan air kontrol sehingga nilai salinitas relatif stabil di kisaran 0,5 sampai 0,7 PSU. Cuplikan pembacaan selama satu jam ditampilkan pada Tabel 1.

Ukuran berkas log per pengiriman lima menit hanya sekitar 2 KB. Pada koneksi 4G dengan kecepatan unggah rata-rata 5 Mbps di lokasi pengujian, transfer berkas sekecil itu selesai dalam waktu kurang dari satu detik. Kami juga mencoba metode pemantauan alternatif, yaitu membuka sesi SSH interaktif melalui tunnel (port 2222) lalu menjalankan tail -f pada berkas log sensor di Raspberry Pi. Output muncul secara real-time di terminal server pusat dengan jeda yang nyaris tidak terasa (sekitar 50 sampai 100 ms). Kombinasi metode scp periodik dan SSH interaktif ini memberi fleksibilitas: pengiriman batch dilakukan otomatis, sementara pemantauan langsung tersedia setiap kali dibutuhkan.

TABEL 1
CUPLIKAN PEMBACAAN SALINITAS SELAMA SATU JAM (DATA UJI LABORATORIUM)

Waktu (WITA)	Salinitas (PSU)
10:00	0,52
10:10	0,53
10:20	0,52
10:30	0,54
10:40	0,53
10:50	0,52
11:00	0,55
11:10	0,56
11:20	0,54
11:30	0,53
11:40	0,52
11:50	0,52

B. Akses Remote Administrator

Skenario kedua menguji akses balik untuk administrator. Dari shell server pusat, perintah ssh -p 2222 pi@localhost selalu berhasil membuka prompt shell Raspberry Pi selama tunnel aktif. Waktu rata-rata sejak perintah dijalankan sampai prompt muncul berada pada kisaran 1 sampai 2 detik. Waktu tersebut mencakup proses negosiasi kunci SSH dan resolusi DNS internal. Karena kunci publik sudah dipasang sebelumnya, tidak terjadi interaksi password sama sekali.

Selama sesi remote, perintah-perintah administratif berjalan tanpa kendala. Pemeriksaan log via journalctl, pemantauan beban dengan top, hingga eksekusi skrip pembacaan manual untuk diagnostik sensor, semuanya berjalan responsif. Transfer file melalui scp dari server ke Raspberry Pi mencatat kecepatan rata-rata 1,2 MB/s, sedangkan arah sebaliknya berada di kisaran 0,8 MB/s. Selisih ini wajar karena pada jaringan seluler retail kecepatan unggah biasanya lebih kecil daripada kecepatan unduh. Untuk keperluan pengambilan log harian, yang biasanya hanya beberapa MB, atau konfigurasi remote, *throughput* tersebut sudah lebih dari cukup.

C. Ketahanan Koneksi (Persistence)

Ketahanan koneksi diuji melalui dua mode gangguan. Pada mode pertama, modem 4G dimatikan kemudian dihidupkan kembali setelah jeda satu menit. Pada mode kedua, perangkat dipindahkan ke ruang dengan sinyal lemah selama lima menit, sehingga koneksi seluler putus-nyambung beberapa kali (*flapping*).

Pada mode pertama, autossh mendeteksi tunnel mati segera setelah keep-alive SSH tidak menerima respon. Begitu modem aktif kembali dan IP baru diperoleh, sesi autossh dibangun ulang. Berdasarkan stempel waktu di journalctl -u autossh-tunnel, rata-rata jeda dari pemulihan internet sampai port 2222 di server siap menerima koneksi adalah 6,4 detik (n = 12, deviasi standar 1,8 detik). Data sensor yang tertahan selama satu menit pemutusan langsung tersinkronisasi pada siklus scp berikutnya. Tidak ada data yang hilang karena berkas log lokal tetap tersimpan utuh di Raspberry Pi.

Pada mode kedua, log autossh menunjukkan upaya reconnect berkali-kali, dengan rata-rata delapan percobaan dalam lima menit. Beberapa percobaan terhenti di tengah negosiasi karena sinyal turun lagi, namun tidak ada yang berakhir gagal permanen. Setelah perangkat kembali ke area dengan sinyal normal, tunnel kembali stabil dalam waktu kurang dari sepuluh detik. Selama periode flapping, mekanisme

Submitted: 05/05/2022; Revised: 10/06/2026;
Accepted: 11/06/2025; Online first: 12/06/2026
https://doi.org/10.46964/poligrid.v7i1.197

buffer lokal di Raspberry Pi tetap menyimpan log. Pada saat tunnel pulih, scp berikutnya memuat semua berkas yang tertunda secara sekaligus.

D. Konsumsi Sumber Daya dan Kinerja pada Berbagai Kondisi Sinyal

Skenario tambahan dilakukan untuk mengamati perilaku sistem pada tiga tingkat kuat sinyal 4G LTE: kuat (RSRP > -90 dBm), sedang (RSRP -90 hingga -110 dBm), dan lemah (RSRP < -110 dBm). Tujuan pengamatan ini adalah memperlihatkan bagaimana waktu reconnect, latensi, dan rasio keberhasilan pengiriman berubah ketika kualitas sinyal menurun. Ringkasan hasilnya disajikan pada Tabel 2.

TABEL 2
KINERJA SISTEM PADA TIGA TINGKAT KUAT SINYAL 4G LTE (SETIAP TINGKAT DIUJI 6 JAM)

Parameter	Kuat	Sedang	Lemah
Rasio paket tiba (%)	100	100	98,3
Latensi SSH balik (ms)	120	175	260
Waktu reconnect (detik)	6,1	7,8	11,2
Throughput scp (MB/s)	1,3	0,9	0,4

Hasil tersebut menunjukkan tiga hal penting. Pertama, sistem tetap mampu mengirim sebagian besar data bahkan pada sinyal lemah, meskipun rasio paket tiba turun ke 98,3 %. Kedua, waktu reconnect bertambah hampir dua kali lipat ketika sinyal lemah, namun tetap berada di bawah 15 detik. Ketiga, throughput scp memang menurun cukup tajam, sehingga untuk lokasi dengan sinyal lemah disarankan menambah interval pengiriman atau mengompres log sebelum scp. Pengaturan tersebut akan kami terapkan ketika sistem di-deploy di Sungai Wain, mengingat tutupan vegetasi di sana dapat melemahkan sinyal seluler.

TABEL 3
RINGKASAN METRIK KINERJA SISTEM SELAMA 72 JAM PENGUJIAN UTAMA

Parameter Uji	Hasil Pengukuran
Laju pengiriman data	1 data/menit; 4.320 data/72 jam; 0 % paket hilang
Latensi akses SSH balik	~150 ms ping; 1–2 detik waktu login
Waktu reconnect autossh	rata-rata 6,4 detik (n=12; σ=1,8 detik)
Uptime tunnel selama 72 jam	99,4 % (≈ 26 menit downtime saat simulasi)
Throughput scp via tunnel	~1,2 MB/s unggah ke Pi; ~0,8 MB/s unduh dari Pi
Pemakaian CPU Raspberry Pi	≈ 5 % saat idle; ≈ 15 % saat transfer berkas
Pemakaian RAM Raspberry Pi	≈ 220 MB dari 4 GB
Overhead bandwidth keep-alive	≈ 250 byte/menit

Tabel 3 merangkum metrik utama dari empat skenario di atas. Empat hal yang menurut kami patut digarisbawahi dari hasil pengujian akan diuraikan pada bagian berikut.

E. Implikasi dan Keuntungan

(1) CGNAT bisa diatasi tanpa biaya layanan. Tanpa berlangganan IP statis maupun VPN komersial, perangkat di balik CGNAT tetap dapat diakses balik dari pusat kendali. Pada beberapa proyek monitoring lingkungan yang sempat kami konsultasikan, biaya berlangganan layanan IoT cloud dapat mencapai Rp 250.000 sampai Rp 500.000 per perangkat setiap bulan. Pendekatan kami memangkas pos biaya tersebut menjadi sekadar sewa VPS standar yang dapat dibagi untuk banyak perangkat sekaligus.

(2) Stabilitas koneksi memuaskan untuk skenario lapangan. Uptime tunnel 99,4 % selama 72 jam pengujian utama menunjukkan total downtime sekitar 26 menit, yang seluruhnya muncul saat simulasi pemutusan jaringan. Waktu pemulihan rata-rata di bawah sepuluh detik membuat sistem ini cukup tangguh untuk lokasi konservasi yang sinyalnya tidak prima.

(3) Keamanan dasar terpenuhi secara default. Karena seluruh trafik (baik data sensor maupun perintah administratif) melewati kanal SSH terenkripsi, risiko penyadapan dan manipulasi data di tengah jalan menjadi sangat kecil. Akun yang dipakai bukan akun root, sehingga eskalasi hak akses tetap dapat dibatasi jika kunci sempat bocor. Untuk skenario produksi, kami merekomendasikan penambahan blok Match pada sshd agar akun pi hanya boleh login dari server pusat, serta menonaktifkan otentikasi password secara global.

(4) Beban komputasi rendah, masih ada ruang untuk modul tambahan. Pemakaian CPU Raspberry Pi tertahan di sekitar 5 % saat idle dan naik ke kisaran 15 % saat scp berjalan. RAM yang terpakai sekitar 220 MB dari total 4 GB. Kondisi ini membuka peluang untuk menambahkan komponen edge computing, misalnya pra-pemrosesan deteksi anomali salinitas berbasis aturan ambang, tanpa perlu mengganti perangkat dasar.

Hasil pengujian ini menegaskan bahwa rekayasa berbasis perangkat lunak bebas, ketika dikonfigurasi dengan benar, dapat menggantikan layanan komersial untuk kebutuhan monitoring lingkungan. Dalam konteks Sungai Wain, hasil tersebut memberi dasar bagi rencana pemasangan empat hingga enam node sensor di sepanjang aliran sungai pada tahap berikutnya, dengan satu VPS terpusat sebagai concentrator data.

Submitted: 05/05/2022; Revised: 10/06/2026;
Accepted: 11/06/2025; Online first: 12/06/2026
https://doi.org/10.46964/poligrid.v7i1.197

F. Perbandingan dengan Pendekatan Lain

Agar kontribusi kami dapat ditempatkan pada konteks yang lebih luas, Tabel 4 membandingkan beberapa pendekatan komunikasi pada penelitian monitoring lingkungan yang telah dilaporkan. Pendekatan reverse SSH tunnel memang bukan satu-satunya jalan, tetapi cukup menonjol dari sisi kombinasi: bekerja di bawah CGNAT, gratis, self-healing, sekaligus menyediakan akses dua arah.

TABEL 4
PERBANDINGAN PENDEKATAN KOMUNIKASI PADA STUDI MONITORING LINGKUNGAN

Pendekatan	Tahan CGNAT	Akses Balik	Biaya Layanan	Self-healing
Wi-Fi + Cloud Public [5], [27]	Tidak relevan	Sebagian	Sedang–tinggi	Tergantung penyedia
GSM/SMS [8]	Ya	Tidak	Rendah	Tidak
LoRa + Gateway lokal [7]	Tidak relevan*	Ya	Rendah	Sebagian
VPN komersial	Ya	Ya	Tinggi	Ya
IP publik statis	—	Ya	Tinggi	Tergantung perangkat
Reverse SSH tunnel (studi ini)	Ya	Ya	Sangat rendah	Ya

*Apabila gateway berada di lokasi yang sama, CGNAT bukan isu; akses balik tetap perlu dirancang ulang bila gateway juga di belakang NAT.

G. Keterbatasan Penelitian

Beberapa keterbatasan perlu kami catat. Pertama, pengujian masih dilakukan di lingkungan laboratorium dengan air kontrol, belum di Sungai Wain langsung. Variasi konduktivitas alami, gelombang, dan biofouling pada probe akan menjadi tantangan tambahan saat sistem ditempatkan di lapangan. Kedua, jumlah node yang diuji baru satu unit. Perilaku jaringan ketika beberapa node berlomba membuka port forward pada satu VPS yang sama belum kami evaluasi. Ketiga, sumber listrik masih menggunakan PLN. Pengujian jangka panjang dengan panel surya dan baterai masih perlu dikerjakan untuk mengukur dampak duty cycle modem 4G terhadap konsumsi daya total.

V. KESIMPULAN

Penelitian ini menunjukkan bahwa kombinasi *reverse* SSH tunnel berbasis autossh dengan manajemen layanan systemd dapat dijadikan solusi yang murah dan efektif untuk mengatasi hambatan akses balik pada perangkat data logger 4G LTE yang berada di balik CGNAT. Selama 72 jam pengujian, sistem mengirim 4.320 titik data tanpa kehilangan paket, mempertahankan uptime tunnel sebesar 99,4 %, serta memulihkan koneksi rata-rata dalam waktu 6 sampai 9 detik setelah simulasi pemutusan jaringan. Akses SSH balik dari server pusat tetap responsif dengan latensi rata-rata 150 ms, dan beban komputasi pada Raspberry Pi 5 berada jauh di bawah saturasi. Pada pengujian dengan tiga tingkat kuat sinyal, sistem tetap menunjukkan kinerja yang dapat diterima, walaupun throughput menurun ketika sinyal lemah. Berdasarkan hasil-hasil tersebut, pendekatan ini layak diimplementasikan pada sistem monitoring salinitas Sungai Wain. Selain itu, konfigurasi yang kami susun dapat dipakai ulang untuk proyek IoT lingkungan lain di Indonesia. Pada tahap berikutnya, tim akan menempatkan empat node sensor di sepanjang aliran sungai, mengintegrasikan panel surya, menambahkan mekanisme failover ganda (kombinasi 4G dengan LoRa atau satelit murah), dan memasang modul deteksi anomali salinitas berbasis aturan ambang.

REFERENSI

[1] Yayasan Konservasi RASI, "Profil Kawasan Hutan Lindung Sungai Wain Balikpapan," Laporan Konservasi Kalimantan Timur, Balikpapan, 2023.

[2] M. Ridwan, S. Hartono, dan A. Yusuf, "Analisis intrusi air laut pada Sungai Kapuas Kalimantan Barat saat musim kemarau," *Jurnal Manajemen Hutan dan Manajemen Sumberdaya Alam*, vol. 9, no. 2, pp. 145–157, 2022.

[3] T. Prasetyo dan R. Susanto, "Karakteristik salinitas di Perairan Teluk Jakarta berdasarkan 25 tahun data," *Jurnal Riset Daerah*, vol. 14, no. 1, pp. 21–34, 2021.

[4] B. Wijaya, A. Ramadhan, dan I. Putra, "Variasi salinitas pesisir akibat pasang surut dan curah hujan: studi kasus pesisir Kalimantan," *Jurnal Kelautan Tropis*, vol. 24, no. 3, pp. 311–322, 2021.

[5] F. Ujianti, A. Setiawan, dan R. Hidayat, "Internet of things-based water quality monitoring: a case study in the coastal areas of Semarang and Kendal," *Asian J. of Water, Environment and Pollution*, vol. 22, no. 4, 2025.

[6] S. Suryono, B. Surarso, dan E. A. Sarwoko, "Capacitive parallel-plate salinity sensor with Wi-Fi microcontroller for real-time water monitoring," *IEEE Access*, vol. 6, pp. 55418–55426, 2018.

[7] K. Pratama dan H. Cahyono, "LoRa-based water quality monitoring with Raspberry Pi gateway in remote areas," *J. Electrical Eng. and Computer Innovations*, vol. 11, no. 2, pp. 233–242, 2023.

[8] L. Enriquez, M. Reyes, dan J. Cruz, "GSM-based salinity monitoring for solar salt farms in remote islands," *Philippine J. of Science*, vol. 152, no. 4, pp. 845–856, 2023.

[9] L. Enriquez dan J. Cruz, "Hybrid wireless monitoring framework for aquaculture and salt farming," *Aquacultural Engineering*, vol. 105, art. 102441, 2023.

[10] P. Gunawan, "Penerapan sistem otomasi hemat energi untuk pemantauan tambak," *Jurnal Otomasi dan Energi*, vol. 7, no. 1, pp. 12–24, 2022.

[11] M. Boucadair, "IPv4-IPv6 Coexistence: Mitigating IPv4 Exhaustion via CGN Deployment," RFC 6888 / IETF Internet Standard, 2013.

[12] C. Donley, L. Howard, V. Kuarsingh, J. Berg, dan J. Doshi, "Assessing the impact of carrier-grade NAT on network applications," *RFC 7021*, IETF, Sep. 2013.

[13] S. Perreault, R. Penno, dan I. Yamagata, "Common requirements for carrier-grade NATs," *RFC 6888*, IETF, Apr. 2013.

[14] G. Huston, "IPv6 capability reaches 50% in the Asia Pacific region," APNIC Blog, Apr. 23, 2025. [Online]. Tersedia: https://blog.apnic.net/2025/04/23/ipv6-capability-reaches-50-in-the-asia-pacific-region/

[15] M. R. Chowdhury dan S. Ahmed, "NAT traversal techniques: a survey," *Int. J. of Computer Applications*, vol. 175, no. 32, pp. 11–19, 2020.

[16] J. Rosenberg, R. Mahy, P. Matthews, dan D. Wing, "Session traversal utilities for NAT (STUN)," *RFC 5389*, IETF, Oct. 2008.

Submitted: 05/05/2022; Revised: 10/06/2026;
Accepted: 11/06/2025; Online first: 12/06/2026
https://doi.org/10.46964/poligrid.v7i1.197

[17] B. Ford, P. Srisuresh, dan D. Kegel, “Peer-to-peer communication across network address translators,” *in Proc. USENIX Annual Tech. Conf*., 2005, pp. 179–192.

[18] F. Dressler, F. Klingler, M. Segata, dan R. L. Cigno, “Network address translation considerations for constrained networks,” *IEEE Internet Computing*, vol. 25, no. 3, pp. 64–72, 2021.

[19] qbee.io, “Reverse SSH tunneling: the ultimate guide,” qbee.io documentation, 2024. [Online]. Tersedia: https://qbee.io/misc/reverse-ssh-tunneling-the-ultimate-guide/

[20] T. Ylonen dan C. Lonvick, “The Secure Shell (SSH) connection protocol,” *RFC 4254*, IETF, Jan. 2006.

[21] R. P. Setiawan, A. Pratama, dan H. Wibowo, “Video Surveillance as a service architecture using HTTP polling and reverse SSH tunneling,” *Jurnal Teknik Informatika dan Sistem Informasi*, vol. 9, no. 2, pp. 156–168, 2023.

[22] OpenSSH Project, “OpenSSH security: best practices for jump hosts and reverse tunneling,” OpenSSH Manual, 2024. [Online]. Tersedia: https://www.openssh.com/manual.html

[23] C. C. Stewart, “autossh(1) – Linux man page,” Harding Software, 2023. [Online]. Tersedia: https://www.harding.motd.ca/autossh/

[24] C. C. Stewart, “autossh README,” GitHub repository, 2024. [Online]. Tersedia: https://github.com/Phenome/autossh

[25] L. Poettering, “systemd.service – service unit configuration,” systemd documentation, freedesktop.org, 2024. [Online]. Tersedia: https://www.freedesktop.org/software/systemd/man/systemd.service.html

[26] J. Geerling, “SSH and HTTP to a Raspberry Pi behind CG-NAT,” Jeff Geerling Blog, Mar. 2022. [Online]. Tersedia: https://www.jeffgeerling.com/blog/2022/ssh-and-http-raspberry-pi-behind-cg-nat

[27] F. Ujianti, A. Setiawan, dan R. Hidayat, “Real-time pH, temperature, and TDS monitoring with NodeMCU for coastal water quality,” in *Proc. Int. Conf. Smart Sensors*, pp. 78–86, 2025.

[28] Raspberry Pi Foundation, “Raspberry Pi 5 product brief,” Raspberry Pi Ltd., 2024. [Online]. Tersedia: https://www.raspberrypi.com/products/raspberry-pi-5/

[29] Texas Instruments, “ADS1115 16-bit ADC datasheet,” Texas Instruments, Rev. SBAS444E, 2018.

[30] DFRobot, “Gravity: analog TDS sensor/meter for Arduino — datasheet,” DFRobot Wiki, 2023. [Online]. Tersedia: https://wiki.dfrobot.com/Gravity__Analog_TDS_Sensor___Meter_For_Arduino_SKU__SEN0244

[31] OpenSSH Project, “ssh-keygen(1) – authentication key generation,” OpenSSH Manual, 2024. [Online]. Tersedia: https://man.openbsd.org/ssh-keygen

[32] Red Hat, “Managing services with systemctl,” Red Hat Enterprise Linux Documentation, 2024. [Online]. Tersedia: https://access.redhat.com/documentation

Submitted: 05/05/2022; Revised: 10/06/2026;
Accepted: 11/06/2025; Online first: 12/06/2026
https://doi.org/10.46964/poligrid.v7i1.197