TESIS

STUDI FUNGSIONALITAS DAN SKALABILITAS PERANGKAT ELECTRONIC NOSE PADA JARINGAN SENSOR NIRKABEL

MUHAMMAD ASEP SUBANDRI
05111650010038

DOSEN PEMBIMBING
Prof. Drs.Ec. Ir. Riyanarto Sarno, M.Sc., Ph.D.

PROGRAM MAGISTER
BIDANG KEAHLIAN REKAYASA PERANGKAT LUNAK
DEPARTEMEN INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN KOMUNIKASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2019

LEMBAR PENGESAHAN TESIS

Telah disusun untuk memenuhi salah satu syarat memperoleh gelar
Magister Komputer (M.Kom.)
di
Institut Teknologi Sepuluh Nopember
Oleh:
MUHAMMAD ASEP SUBANDRI
NRP: 05111650010038

Tanggal Ujian: 8 Juli 2019
Periode Wisuda: September 2019

Disetujui oleh:
Pembimbing:

1. Prof. Drs. Ec. Ir. Riyanarto Sarno, M. Sc., Ph. D
   NIP. 195908031986011001 .....

Penguji:

1. Dr. Ir. Siti Rochimah, M.T
   NIP. 196810021994032001 .....
2. Dr. Ir. Raden Venantius Hari Ginardi, M.Sc
   NIP. 196505181992031003 .....
3. Hadziq Fabroyir, S.Kom., Ph.D
   NIP. 198602272019031006 .....

Kepala Departemen Teknik Informatika
Fakultas Teknologi Informasi dan Komunikasi

Dr. Eng. Darlis Herumurti, S.Kom., M.Kom
NIP. 197712172003121001

[Halaman ini sengaja dikosongkan]

STUDI FUNGSIONALITAS DAN SKALABILITAS PERANGKAT ELECTRONIC NOSE PADA JARINGAN SENSOR NIRKABEL

Nama mahasiswa : Muhammad Asep Subandri

NRP : 05111650010038

Pembimbing : Prof. Ir. Drs. Ec. Riyanarto Sarno, M.Sc., Ph.D.

ABSTRAK

Electronic nose (e-nose) adalah instrumen elektronik yang diciptakan untuk dapat mengenali bau dengan meniru prinsip kerja sistem penciuman manusia. Pada umumnya e-nose dirancang dan dibangun untuk aplikasi yang berdiri sendiri. Penerapan teknologi *wireless sensor network* (WSN) dan *internet of things* (IoT) pada sistem *e-nose* dapat membuatnya menjadi terskalakan dan membuatnya dapat diaplikasikan secara luas mulai dari skala rumahan hingga industri. Pada penelitian sebelumnya, dikembangkan sistem *e-nose* berbasis IoT, dimana dengan sistem tersebut sebuah jaringan yang terdiri dari beberapa perangkat *e-nose* dapat melakukan pengukuran secara terdistribusi. Pada penelitian tersebut digunakan arsitektur *cloud* sentris, seluruh data dikumpulkan, diproses, dan dianalisa di *cloud.* Dari studi literatur yang penulis temukan, sistem seperti ini menghasilkan latensi yang tinggi dengan skalabilitas yang rendah. Pada penelitian ini digunakan arsitektur *fog.* Dimana, sebagian data diproses terlebih dahulu sebelum dikirimkan ke *cloud.* Dari pengujian klasifikasi, sistem mampu memprediksi sampel dengan tingkat akurasi 78%. Sementara dari pengujian skalabilitas yang dilakukan, 300 *sensor node* mampu dihubungkan ke sistem secara bersamaan apabila menggunakan 8 menit data, dan mencapai 500 *sensor node* apabila menggunakan 6 menit data.

Kata kunci: *electronic nose, internet of things, fungsionalitas, skalabilitas, pengujian, wireless sensor network*.

[Halaman ini sengaja dikosongkan]

FUNCTIONALITY AND SKALABILITY STUDY OF ELECTRONIC NOSE DEVICES IN WIRELESS SENSOR NETWORK

Student name	: Muhammad Asep Subandri
NRP	: 05111650010038
Supervisor	: Prof. Ir. Drs. Ec. Riyanarto Sarno, M.Sc., Ph.D.

ABSTRACT

Electronic nose (e-nose) is an electronic instrument created to be able to recognize odors by imitating the working principle of the human olfactory system. In general, e-nose is designed and built for stand-alone applications. The implementation of wireless sensor network (WSN) technology and internet of things (IoT) on the e-nose system can make it scalable and make it applicable from home to industrial scale. In the previous studies, an IoT-based e-nose system was developed, where with this system, a network consisting of several e-nose devices can conduct distributed measurements. In their study, cloud centric architecture was used. All data is collected, processed, and analyzed in the cloud. From the literature study that we found, this system produces high latency with low scalability. In this study fog architecture was used. Where, some data is aggregated and processed before being sent to the cloud. From classification testing, the system is able to predict samples with an accuracy rate of 78%. While the scalability testing is carried out, 300 sensor nodes are able to be connected to the system simultaneously when using 8 minutes of data, and reach 500 sensor nodes when using 6 minutes of data.

Keywords: *electronic nose, internet of things, functionality, scalability, testing, wireless sensor network.*

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Segala puji dan syukur atas kehadirat Allah SWT yang telah memberikan rahmat, hidayah dan karunia-Nya sehingga penulis dapat menyelesaikan tesis yang berjudul "Studi Fungsionalitas dan Skalabilitas Perangkat *Electronic Nose* pada Jaringan Sensor Nirkabel". Selama pengerjaan tesis ini penulis banyak mendapat bantuan dan bimbingan dari berbagai pihak, baik berupa arahan maupun motivasi yang membangun sehingga dengan kerja keras, do'a dan kesabaran tesis ini dapat diselesaikan. Pada kesempatan ini juga penulis ingin menyampaikan ucapan terima kasih dan penghormatan yang sebesar-besarnya kepada :

1. Bapak, Ibu, dan Adikku tercinta yang selalu memberikan do'a dan dukungan.
2. Bapak Prof. Ir. Drs. Ec. Riyanarto Sarno, M.Sc., Ph.D selaku dosen pembimbing yang telah bersedia meluangkan waktu untuk memberikan ilmu, arahan, petunjuk dan motivasi selama proses pengerjaan tesis ini.
3. Ibu Dr. Ir. Siti Rochimah, M.T, Bapak Dr. Ir. Raden Venantius Hari Ginardi, M.Sc, dan Bapak Hadziq Fabroyir, S.Kom., Ph.D selaku dosen penguji yang telah memberikan saran, arahan dan koreksi dalam tesis ini.
4. Bapak dan Ibu dosen Jurusan Teknik Informatika ITS yang telah banyak memberikan ilmu dan bimbingan yang tak ternilai harganya.
5. Seluruh staf dan karyawan Teknik Informatika ITS yang telah banyak memberikan kelancaran administrasi akademik
6. Seluruh teman-teman seperjuangan, sahabat dan kakak tingkat di lab Pasca Sarjana Teknik Informatika ITS yang juga selalu memberikan dukungan dan motivasi selama pengerjaan tesis ini.

Surabaya, Juli 2019

Muhammad Asep Subandri

[Halaman ini sengaja dikosongkan]

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Kualitas merupakan faktor paling penting yang menentukan diterimanya suatu produk di pasaran. Saat ini ada dua metode yang digunakan untuk mengevaluasi kualitas makanan, yakni metode sensorik manusia dan metode analitis seperti *gas chromatography* (GC-MS). Kedua metode tersebut cenderung destruktif dan dilakukan secara manual. Kemudian tidak dapat mengevaluasi dalam jumlah banyak secara bersamaan. Lebih jauh lagi, ia hanya bisa dilakukan secara *off-line*, dalam artian baru bisa dilakukan ketika produk telah selesai diproduksi. Oleh karena itu, industri makanan membutuhkan alat ukur yang dapat mereproduksi fungsi sensorik manusia, dan dapat mengevaluasi kualitas makanan secara *on-line* (Garcia-Esteban et al., 2018).

Pada tahun 1982, Persaud dan Dodd dari Universitas Warwick memperkenalkan *electronic nose (e-nose)* (Persaud & Dodd, 1982). Sebagaimana yang dapat dijelaskan dari namanya, *e-nose* merupakan instrumen elektronik yang diciptakan untuk dapat mengenali bau seperti indra penciuman manusia. *E-nose* berpotensi untuk mengisi celah kekurangan metode analisis aroma klasik, hal ini dikarenakan *e-nose* portabel, cepat, akurat, dapat diandalkan, dan tidak memerlukan pereaksi kimia. Selain itu, pengoperasian *e-nose* jauh lebih murah dan mudah (Patel, Austin, Barber, & Patel, 2014).

Pada umumnya, *e-nose* merupakan yang berdiri sendiri *(stand-alone)*. Hal ini membuatnya tidak dapat melakukan penginderaan secara terdistribusi (Saad et al., 2012). Baru-baru ini, muncul penelitian yang mengusulkan penggunaan teknologi jaringan sensor nirkabel (*wireless sensor network;* WSN) pada perangkat *e-nose*. Diantaranya pada penelitian (Choi, Park, Chang, Lee, & Lee, 2015; Deshmukh, Kasbe, Mujawar, Mule, & Shaligram, 2016; Hassan & Bermak, 2012). Dengan menjadikannya *wireless electronic nose* (WEN), setiap perangkat yang terhubung dapat berkolaborasi dan dimonitoring dari jarak jauh.

Penelitian ini mengusulkan penerapan *internet of things* (IoT) pada *e-nose* agar sistem menjadi lebih fleksibel dan terskalakan. IoT merupakan istilah yang digunakan pada skenario dimana suatu perangkat dihubungkan ke internet, diberikan identitas unik, dan diberikan kemampuan untuk berinteraksi tanpa campur tangan manusia (da Cruz, Rodrigues, Al-Muhtadi, Korotaev, & Albuquerque, 2018). IoT telah menarik perhatian berbagai pihak dalam beberapa tahun terakhir ini. Ukuran sensor yang semakin kecil, konsumsi energi yang semakin sedikit, harga yang semakin murah, teknologi komunikasi baru dan konektivitas yang semakin mudah, mendorong penerapan teknologi ini. Beberapa sektor seperti transportasi, kesehatan, logistik, manufaktur telah merasakan manfaat dari penerapan IoT (Ray, 2016).

Ide pemanfaatan teknologi IoT pada *e-nose* pernah diusulkan oleh (Herrero, Lozano, Santos, & Suárez, 2016; Wijaya, Sarno, Zulaika, & Sabila, 2017). Hanya saja, masih terdapat beberapa kesulitan ketika solusi *cloud* terpusat yang mereka usulkan benar-benar diimplementasi. Diantaranya: karena menghabiskan banyak *bandwidth*, tinggi *latency,* redudansi data dan potensi kegagalan jaringan yang tinggi (Giang, Blackstock, Lea, & Leung, 2015).

Kebaruan dari penelitian ini dibandingkan penelitian sebelumnya ialah mengusulkan arsitektur *fog* untuk sistem *e-nose* berbasis IoT. Pada arsitektur ini, dilakukan pemrosesan pendahuluan *(preliminary)* seperti menyaring, mengumpulkan dan menganalisis data mentah, sebelum dikirimkan ke *cloud*. Dengan cara seperti ini, biaya sistem dapat ditekan, dan respon sistem secara keseluruhan meningkat. Selain itu, menurut (Dastjerdi & Buyya, 2016; Yigitoglu, Mohamed, Liu, & Ludwig, 2017) dapat meningkatkan skalabilitas sistem.

Sebagai studi kasus, dibuat sebuah sistem untuk proses kontrol kualitas makanan, dalam hal ini pisang *(Musa spp.)*. Sistem dievealuasi dengan melakukan pengujian fungsionalitas, klasifikasi, dan pengujian skalabilitas untuk memastikan bahwa sistem dapat diterapkan oleh industri.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dikemukakan sebelumnya maka dapat dirumuskan beberapa masalah antara lain:

1. Bagaimana mengembangkan perangkat *e-nose* dengan larik sensor kimia non-spesifik.
2. Bagaimana proses analisis sinyal yang dihasilkan larik sensor sehingga dapat mengenali aroma.
3. Bagaimana melakukan optimisasi larik sensor untuk meminimalisir penggunaan sensor.
4. Bagaimana membangun sistem *e-nose* yang terskalakan dan bagaimana mengujinya.

1.3 Batasan Masalah

1. Sampel yang digunakan adalah pisang kepok *(Musa x Acuminata)*.
2. Sistem mengklasifikasi kualitas sampel pisang berdasarkan tingkat kematangannya (mentah atau matang).
3. Pengujian fungsionalitas dan klasifikasi sistem menggunakan satu *sensor node (e-nose)*.
4. Pengujian skalabilitas sistem dilakukan dengan mensimulasikan sejumlah *sensor node*. Parameter yang digunakan untuk mengukur skalabilitas adalah kecepatan waktu pemrosesan, jumlah data yang dikirim dalam selang waktu, jumlah data terkirim dan yang diterima, dan jumlah data yang benar-benar tersimpan di basis data

1.4 Tujuan Penelitian

Tujuan yang ingin dicapai dalam penelitian ini adalah menghasilkan sebuah sistem *e-nose* yang terskalakan. Dimana, ia mampu memonitor dan memprediksi kualitas sampel pisang dengan benar secara *on-line* dan otomatis.

1.5 Manfaat Penelitian

Adapun manfaat ini ialah dapat digunakan secara luas oleh industri sebagai teknologi alternatif untuk mengevaluasi kualitas makanan sebagai upaya menjaga kepuasan konsumen.

1.6 Kontribusi

Adapun kontribusi yang dihasilkan dari penelitian ini adalah:

1. Mengusulkan *fog computing* pada arsitektur sistem *e-nose* berbasis IoT.
2. Menguji skalabilitas sistem *e-nose* yang menerapkan teknologi IoT.

BAB 2
KAJIAN PUSTAKA

2.1 *Electronic Nose (E-Nose)*

Electronic nose yang selanjutnya disingkat *e-nose* adalah instrumen elektronik yang terinspirasi dari hidung manusia, ia dirancang untuk dapat mendeteksi dan mengenali gas, bau, atau aroma. Dari segi kinerja, kemampuan *e-nose* dapat disetarakan dengan hidung manusia. Bahkan, dalam kasus tertentu justru lebih unggul. Seperti saat mendeteksi zat-zat yang tidak dapat tercium oleh hidung manusia, atau saat mendeteksi tingkat konsentrasi aroma (Patel et al., 2014).

E-nose tidak mencari suatu molekul atau senyawa gas tertentu, tetapi lebih kepada pencarian pola unik semacam "sidik jari" dari udara yang dianalisa. Pola unik tersebut diperoleh menggunakan larik sensor *(sensor array)* kimia. Larik sensor terdiri dari kombinasi beberapa sensor kimia berbeda, tujuannya agar setiap sensor pada larik memiliki respon khusus ketika terpapar suatu aroma. Misalnya pada aroma A, sensor 1 menghasilkan respon yang tinggi, sementara sensor lain responnya rendah. Pada aroma B, sensor lain menghasilkan respon yang tinggi, sensor 1 responnya rendah. Perbedaan respon ini disimpan pada *database* yang diperlukan untuk melatih algoritma pengenalan pola dalam identifikasi dan klasifikasi aroma.

Penciuman, menurut penelitian modern, belum bisa diklasifikasikan berdasarkan sifat yang dikenal. Tidak seperti indra perasa dan penglihatan, indera penciuman tidak memiliki dimensi kategori yang diketahui (Heredia, Cruz, Balbin, & Chung, 2016). Oleh karena itu banyak penelitian *e-nose* yang menggunakan larik sensor kimia non-spesifik.

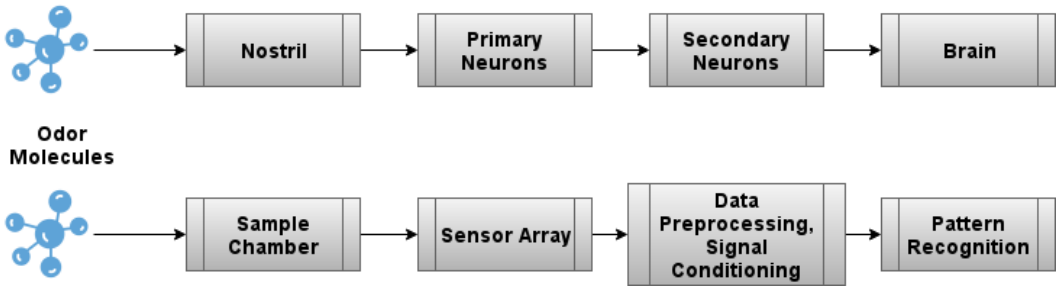
Wojnowski (Wojnowski, Majchrzak, Dymerski, Gębicki, & Namieśnik, 2017) pada penelitiannya yang berjudul *Electronic noses: Powerful tools in meat quality assessment,* mengatakan beberapa senyawa volatil dapat digunakan sebagai indikator pembusukan daging, sehingga perangkat *e-nose* yang dilengkapi

dengan sensor sensitif terhadap senyawa tersebut akan sangat berguna. Sayangnya, belum ada senyawa tunggal yang diidentifikasi sebagai penyebab utama pembusukan daging. Hal ini karena perubahan yang terjadi pada sampel daging selama penyimpanan bergantung pada banyak faktor. Akhirnya digunakan "sidik jari" unik dari kombinasi beberapa senyawa volatil untuk mengidentifikasi tingkat kebusukan daging.

Penggunaan sensor gas non-spesifik seperti cara diatas juga dilakukan oleh Valdez (Valdez & Gutiérrez, 2016). Larik sensor yang pilihannya mampu memberikan data yang cukup untuk dapat mengelompokkan sampel coklat. Ia juga menggunakan "sidik jari" unik dari karakteristik campuran senyawa volatil yang terdapat di setiap sampel.

2.2 Komponen *E-Nose*

Perangkat *e-nose* modern umumnya terdiri dari tiga komponen utama yakni ruang sampel, larik sensor, dan sistem pemrosesan sinyal. Sistem pemrosesan sinyal terdiri atas dua subsistem yakni sistem prapemrosesan sinyal dan sistem pengenalan pola (Zheng & Lin, 2012). Karena meniru sistem penciuman manusia, komponen pada *e-nose* mirip dengan komponen yang ada pada pada sistem penciuman manusia, begitu juga dengan prosesnya, Gambar 2.1 menunjukkan kesamaan tersebut (Heredia et al., 2016; Patel et al., 2014).



Gambar 2.1 Komponen sistem penciuman manusia dan *e-nose*

a) Ruang Sampel

Merupakan ruang tertutup untuk meletakkan sampel, ruang ini dibuat untuk mengisolasi senyawa volatil yang dihasilkan sampel dari pengaruh

lingkungan sekitarnya. Kondisi di luar ruang sampel seperti angin, suhu, dan kelembaban dapat mempengaruhi pembacaan sensor gas. Untuk itu, diperlukan suatu kondisi lingkungan yang kondusif untuk meminimalisir noise tersebut. Hasil eksperimen yang dilakukan oleh (Chansongkram & Nimsuk, 2016) menunjukkan bahwa respon sensor *e-nose* lebih stabil jika menggunakan tempat sampel yang tertutup dibandingkan dengan tempat sampel yang terbuka.

b) Larik Sensor

Merupakan komponen inti dari perangkat *e-nose*, yang dapat menyerap dan mengubah senyawa volatil menjadi sinyal listrik. Ada berbagai jenis sensor yang tersedia untuk mengembangkan perangkat *e-nose*, diantaranya *Metal Oxide Semiconductors* (MOS), *Conducting Polymers* (CP), *Chemocapacitors*; *MOS Field Effect Transistors* (MOSFET), *Quartz Crystal Microbalance* (QCM), *Surface Acoustic Wave* (SAW), dan SPR. Sensor dengan jenis tersebut dapat mendeteksi senyawa gas yang menguap dengan mengukur aliran listrik, panas, optik, dan perubahan massa (Patel et al., 2014).

Sensor gas yang paling banyak digunakan pada penelitian terkait *e-nose* saat ini adalah sensor gas dengan jenis MOS. Jenis ini banyak dipilih karena sudah tersedia secara bebas di pasaran, terdapat banyak variasi yang dapat digunakan untuk mendeteksi berbagai jenis gas, memiliki harga yang relatif murah, dan mempunyai sensitivitas yang cukup tinggi (Jiang, Wang, Wang, & Cheng, 2017).

Sensor MOS bekerja berdasarkan tingkat penyerapan oksigen yang terdapat di udara oleh lapisan metal oksida. Pada udara bersih yang mengandung banyak oksigen, elektron tidak dapat mengalir bebas menuju lapisan metal oksida karena tertahan oleh oksigen, sehingga mencegah adanya aliran arus listrik. Saat udara terpapar gas, terjadi oksidasi antara gas tersebut dengan oksigen, hal ini mengakibatkan elektron terlepas dan listrik dapat mengalir (Szulczyński & Gębicki, 2017).

c) Sistem Prapemrosesan Sinyal

Data keluaran sensor gas berupa tegangan listrik (V) atau hambatan listrik (R) yang dihasilkan perangkat *e-nose*, diolah terlebih dahulu untuk menghasilkan hanya informasi yang berguna bagi sistem pengenalan pola. Prosedur ini biasanya terdiri dari proses: (1) Normalisasi, untuk menghasilkan sinyal dengan besaran yang sama; (2) Ekstraksi, untuk mengurangi dimensi fitur; dan (3) Seleksi fitur, untuk menemukan fitur paling signifikan dan relevan terhadap karakteristik aroma.

d) Sistem Pengenalan Pola

Komponen ini berfungsi untuk melakukan prediksi, identifikasi, atau klasifikasi aroma dengan membandingkan data aroma baru dengan data-data aroma yang telah dikenali sebelumnya (Zheng & Lin, 2012). Ada beberapa teknik yang dapat digunakan. Jika terdapat perbedaan respon gas sensor yang signifikan, maka sampel dapat diklasifikasi dengan teknik menyandingkan data pada grafik. Namun, pada kenyataannya data respon sensor gas begitu kompleks, sulit dilihat dengan kasat mata, sehingga perlu menggunakan teknik statistik seperti *Principal Component Analysis* (PCA), *Cluster Analysis* (CLA), *Linear Discriminant Analysis* (LDA), *Partial Least Squares* (PLS), atau yang lainnya. Sedangkan teknik lain yang terbaru, menggunakan jaringan saraf tiruan *(Artificial Neural Networks [ANN])* (Berna, 2010; Wojnowski et al., 2017).

2.3 Aplikasi *E-Nose*

Perangkat *e-nose* telah menarik perhatian yang sangat besar dalam tiga dekade terakhir, banyak penemuan terkait aplikasi *e-nose* yang berasal dari berbagai bidang ilmu terapan. Penemuan ini memberikan banyak sekali manfaat bagi manusia (Patel et al., 2014), diantaranya dapat digunakan untuk mendeteksi penyakit (McWilliams, Beigi, Srinidhi, Lam, & MacAulay, 2015), mendeteksi bahan peledak (López, Triviño, Calderón, Arcentales, & Guamán, 2017), mendeteksi gas beracun tidak berbau (Ravindra & Rajbhoj, 2017), mengetahui

kualitas suatu produk (Somasagar & Kusagur, 2017), mengidentifikasi spesies tanaman (Yu, Meng, & Yin, 2013), dan lain sebagainya. Tabel 2.1 merangkum beberapa studi pemanfaatan *e-nose* di sektor industri.

Tabel 2.1 Beberapa contoh aplikasi *electronic nose* (Wilson & Baietto, 2011)

Industri	**Bidang Aplikasi**	**Contoh Penggunaan Spesifik**
Agrikultur	Perlindungan tanaman pangan Waktu panen & Penyimpanan Daging, Perikanan Produksi tanaman Penyakit pra- & pascapanen	Menjamin suplai makanan Mengetahui tingkat kematangan Kesegaran, kontaminasi Klasifikasi varietas Deteksi hama tanaman
Transportasi udara	Keselamatan & kesejahteraan publik Penumpang & personil keamanan	Deteksi bahan peledak dan bahan mudah terbakar
Kosmetik	Produk untuk aplikasi pribadi Zat aditif pewangi	Pengembangan parfum
Lingkungan	Pemantauan kualitas air dan udara Kontrol kualitas udara dalam ruangan Peraturan pengurangan polusi	Deteksi polusi dan gas berbahaya
Makanan & minuman	Pencegahan penipuan konsumen Penilaian kendali mutu Kematangan, kontaminasi makanan Rasa, karakteristik bau	Mengetahui standar kandungan pada makanan, pengenalan merk
Manufaktur	Kontrol pemrosesan Keseragaman produk Keselamatan, keamanan, kondisi kerja	Karakteristik aroma dan rasa Alarm kebakaran, deteksi kebocoran gas berbahaya
Medis & klinis	Identifikasi patogen Deteksi patogen atau penyakit Kondisi fisiologis	Diagnosa penyakit, kegagalan organ
Militer	Keamanan personil & penduduk Keamanan sipil dan militer	Senjata biologis dan kimia Deteksi bahan peledak

Industri	Bidang Aplikasi	Contoh Penggunaan Spesifik
Farmasi	Kontaminasi, kemurnian produk Variasi dalam campuran produk	Kontrol kualitas kemurnian obat
Regulasi	Perlindungan konsumen Perlindungan lingkungan	Pengujian kontaminasi udara, air, dan tanah

Sebagaimana yang dapat dilihat pada Tabel 2.1, aplikasi *e-nose* sangat beragam, penggunaannya mulai dari lahan kebun sampai medan perang. Sejak tahun 1993, terdapat lebih dari 12.000 artikel terkait *e-nose* yang dipublikasikan. Sebagian besar aplikasinya ialah di sektor industri makanan, seperti: buah, daging, susu, anggur (minuman), teh, dan kopi. Diperkirakan ada 5000 publikasi yang terbit sejak saat itu, jumlahnya hampir separuh dari total keseluruhan publikasi *e-nose* (Loutfi, Coradeschi, Mani, Shankar, & Rayappan, 2015).

2.4 Wireless Electronic Nose (WEN)

Ada tiga hal yang menjadi fokus penelitian *e-nose* saat ini, yaitu bagaimana membuat perangkat yang lebih kecil, murah, dan dapat menghasilkan kinerja yang lebih baik. Walaupun begitu, ada kesamaan diantara kebanyakan penelitian tersebut, dimana pemrosesan data dilakukan secara lokal, baik di perangkat *e-nose* sendiri maupun di komputer personal dengan perangkat lunak pengenalan pola khusus. Cara ini tidak menyediakan akses untuk melihat hasil prediksi sampel dari jarak jauh.

Sedikit sekali publikasi yang membahas tentang upaya meningkatkan kemampuan sistem *e-nose* untuk dapat memantau sampel dari tempat lain. Beberapa artikel yang berhasil ditemukan diantaranya ialah (Deshmukh et al., 2016), ia memberikan kemampuan komunikasi pada perangkat *e-nose*-nya dengan mengadopsi teknologi *Wireless Sensor Network* (WSN). Menggunakan metode PCA untuk analisis data respon sensor (voltase), sampel mangga muda, matang, dan busuk dapat diklasifikasi dengan melihat grafik berbentuk radar. Kemudian

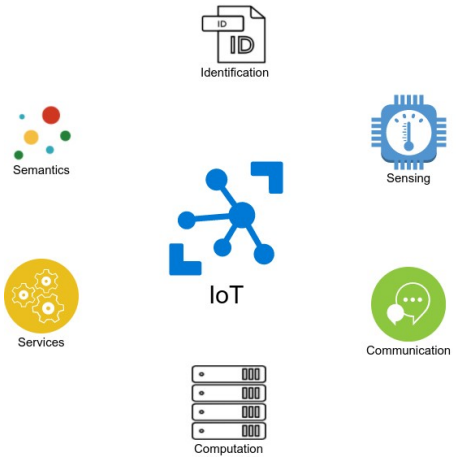
ada (Herrero et al., 2016) menggunakan WSN dan aplikasi web perangkat lunak pengenalan pola yang dipasang di server lokal, penelitiannya dapat memprediksi 12 kelas sampel zat polutan yang terkandung di air dengan tingkat akurasi 91% dan 94% menggunakan algoritma *Backpropagation learning algorithm* (BP) dan *Radial-Basis Functions based neural network* (RBF). Lalu, artikel penelitian yang ditulis oleh (Wijaya & Sarno, 2015), penelitian tersebut mengusulkan arsitektur sistem *Mobile Electronic Nose* (Molen) untuk memprediksi level pembusukan daging. Sistem yang diusulkan berbasis teknologi *Internet of Things* (IoT), terdiri dari beberapa jaringan WSN yang terintegrasi dengan infrastruktur *Cloud Computing.* Namun, usulan tersebut baru sekedar konsep. Ujicoba yang dilakukan masih lokal. Adapun algoritma *K-Nearest Neighbors* yang digunakannya mampu membedakan dua, tiga, dan empat kelas kualitas daging dengan akurasi sebesar 93.64%, 86.00%, dan 85.50%.

2.5 Internet of Things (IoT)

Saat ini tren internet bergerak dari menghubungkan manusia ke menghubungkan perangkat, konsep baru ini disebut *Internet of Things* (IoT). Diperkirakan ada 26 miliar perangkat yang akan terhubung ke internet pada tahun 2020. Perangkat tersebut mulai dari sensor dalam ruangan, sensor yang dikenakan, hingga sensor luar ruangan. Perangkat-perangkat tersebut diberikan kemampuan untuk saling berkomunikasi secara mandiri tanpa campur tangan manusia, saling menerima dan mengirimkan data melalui koneksi jaringan. Hal ini memberikan revolusi bisnis dan aplikasi baru di berbagai bidang, seperti transportasi, perawatan kesehatan, otomatisasi rumah/industri/pertanian, deteksi/pencegahan bencana, dan sebagainya, yang secara signifikan dapat meningkatkan kualitas hidup manusia menjadi lebih baik (Kuo, Li, Jhang, & Lin, 2018).

IoT merupakan konsep yang lebih global dari WSN (Morillo, Orduña, Fernández, & García-Pereira, 2018), WSN cakupannya lokal, sedangkan IoT memiliki lingkup yang lebih luas. Ada enam elemen utama yang diperlukan untuk

membangun sistem yang berbasis teknologi IoT menurut (Al-Fuqaha, Guizani, Mohammadi, Aledhari, & Ayyash, 2015), elemen-elemen tersebut dapat dilihat pada Gambar. 2.2.



Gambar 2.2 Elemen-elemen pada IoT (Al-Fuqaha et al., 2015)

a) Identifikasi

Setiap perangkat IoT perlu diberikan identitas unik agar senantiasa dapat diakses. Selain itu, ia juga diperlukan agar tidak salah dalam mengakses perangkat yang diinginkan.

b) Pengindraan *(sensing)*

Merupakan kemampuan yang harus dimiliki oleh perangkat IoT, yaitu mendapatkan data dari lingkungan sekitar dan mengirimkannya ke tempat penyimpanan data, basis data, atau awan *(cloud)*. Data ini dikumpulkan untuk nantinya dianalisa sesuai kebutuhan.

c) Komunikasi

Sistem IoT memerlukan teknologi komunikasi yang dapat menghubungkan berbagai macam objek yang beragam sehingga dapat menjadi lebih pintar. Perangkat IoT harus dapat beroperasi dengan minim konsumsi daya dan komunikasi yang kurang bagus. Contoh protokol komunikasi yang digunakan pada IoT adalah *WiFi, Bluetooth,* IEEE 802.15.4, *Z-wave,* dan *LTE-Advanced*.

d) Komputasi

Unit pemrosesan seperti misalnya mikrokontroler dan perangkat lunak mewakili otak dari sistem IoT untuk melakukan segala macam perhitungan. Beberapa perangkat keras telah dikembangkan agar dapat menerapkan aplikasi IoT seperti Arduino, UDOO, FriendlyARM, Intel Galileo, Raspberry PI, Gadgeteer, BeagleBone, Cubieboard, Z1, WiSense, Mulle, dan T-Mote Sky. Awan *(cloud)* juga dimanfaatkan sebagai bagian dari IoT untuk melakukan komputasi. Awan memberikan fasilitas tempat penyimpanan data perangkat IoT, untuk kemudian dapat memberikan pengetahuan pengguna dari *Big Data* yang dikumpulkan.

e) Layanan *(services)*

Ada empat jenis layanan yang dapat diberikan oleh sistem IoT, diantaranya: *Identity-related Services, Information Aggregation Services, Collaborative-Aware Services and Ubiquitous Services*. Layanan *Identity-related* adalah yang paling dasar dan digunakan oleh jenis layanan lain, digunakan untuk mengidentifikasi *smart object.* Layanan *Information Aggregation* mengumpulkan dan hasil pengukuran sensor yang diproses dan dikirim ke aplikasi IoT. *Collaborative-Aware,* layanan yang dapat mengambil keputusan dan bereaksi terhadap data yang dikumpulkan oleh layanan *Information Aggregation.* Layanan *Ubiquitous*, bertujuan untuk menyediakan layanan *Collaborative-Aware* kapanpun, siapapun, dan dimanapun.

f) Semantik

Semantik pada IoT mengacu pada kemampuan mengekstrak pengetahuan *(knowledge)* dari berbagai informasi yang dihasilkan objek IoT untuk menyediakan layanan yang dibutuhkan.

2.6 Karakteristik dan Kebutuhan Aplikasi IoT

Terdapat beberapa karakteristik pada infrastruktur IoT (Morin, Harrand, & Fleurey, 2017; Razzaque, Milojevic-Jevric, Palade, & Clarke, 2016), diantaranya:

a) Kendala sumber daya

Perangkat sensor, aktuator dan perangkat *embedded* pada infrastruktur IoT sebisa mungkin memiliki bentuk yang kecil untuk menghemat penggunaan energi, konsekuensinya membuat kemampuan pemrosesan, memori, dan komunikasi yang dimiliki menjadi terbatas.

b) Perangkat yang heterogen

IoT tidak hanya terdiri dari perangkat sensor, aktuator dan perangkat *embedded* saja, namun juga terdapat perangkat komputasi dengan kemampuan komputasi berat untuk menyelesaikan tugas seperti pemrosesan data dan *routing* jaringan. Perangkat-perangkat tersebut sangat beragam, baik dari sisi kapasitas, fitur, kebutuhan, maupun vendor yang memproduksinya.

c) Interaksi spontan

Dalam aplikasi IoT, interaksi yang tiba-tiba dapat terjadi ketika sebuah objek bergerak, dan masuk ke jangkauan komunikasi objek lain, kemudian menghasilkan peristiwa spontan. Biasanya, dalam IoT, interaksi dan terjadinya suatu peristiwa dilakukan tanpa campur tangan manusia.

d) Jaringan berskala besar

Pada lingkungan IoT, terdapat ribuan perangkat yang saling berinteraksi satu sama lain di satu tempat lokal saja (misalnya, di gedung, supermarket, dan universitas), jumlah ini lebih besar dari kebanyakan sistem jaringan konvensional. Secara global, IoT akan menjadi jaringan berskala sangat besar, menghubungkan perangkat dalam skala miliaran dan bahkan dalam triliunan.

e) Jaringan dinamis

IoT akan mengintegrasikan perangkat-perangkat yang banyak diantaranya mudah dipindah *(mobile),* menggunakan jaringan nirkabel, dan memiliki sumber daya yang terbatas. Perangkat tersebut bisa meninggalkan ataupun bergabung ke suatu jaringan. Selain itu, perangkat dapat terputus karena

koneksi nirkabel yang buruk atau kehabisan baterai. Faktor-faktor inilah yang membuat jaringan IoT sangat dinamis.

f) Sadar konteks

Sadar akan konteks adalah kunci dari IoT dan aplikasinya. Sejumlah besar sensor akan menghasilkan sejumlah besar data yang tidak akan memiliki nilai apa-apa jika tidak dianalisis, ditafsirkan, dan dipahami.

g) Terdistribusi

Internet tradisional itu sendiri merupakan jaringan yang terdistribusi secara global, begitu juga dengan IoT. Dalam jaringan IoT, perangkat terdistribusi di berbagai skala, baik skala global seperti Internet, maupun skala lokal di dalam area aplikasi.

Adapun kebutuhan yang harus dipenuhi pada aplikasi IoT adalah sebagai berikut (Razzaque et al., 2016; Yigitoglu et al., 2017):

a) Real-time

Kebanyakan aplikasi IoT membutuhkan respon yang cepat, terjadinya *delay* sebisa mungkin dihindari. Pada aplikasi sistem transportasi pintar, kebutuhan ini sangat krusial, keterlambatan kendaraan menerima status lampu lalu lintas bisa berujung kecelakaan.

b) Konsumsi *bandwidth* rendah

Jumlah sensor IoT dengan skala besar menghasilkan data dengan jumlah yang besar pula setiap detiknya, mengirimkan data tersebut langsung ke awan *(cloud)* untuk dianalisis dan diproses jelas bukan pilihan yang bijak, diperlukan tindak lanjut untuk meminimalisir data yang dikirimkan.

c) Kemampuan mobilitas

Perangkat IoT terkadang melekat pada objek yang dapat berpindah-pindah posisi. Oleh karena itu, diperlukan kemampuan untuk tetap terintegrasi dan tetap dapat mengirimkan data ke objek lain melalui jaringan terdekat.

d) Kemampuan interoperabilitas

   Setiap perangkat/teknologi/aplikasi heterogen yang terdapat pada infrastruktur IoT harus dapat bekerja satu sama lain, saling bertukar data dan layanan tanpa ada upaya tambahan dari pengembang.

e) Keamanan dan privasi terjaga

   Aplikasi IoT mengumpulkan informasi aktivitas sehari-hari penggunanya. Informasi seperti misalnya rute perjalanan, kebiasaan belanja, penggunaan energi harian, dan lain-lain seharusnya tetap rahasia. Penggunaan *cloud computing* memperbesar peluang bocornya informasi pribadi tersebut.

f) Keandalan *(reliability)* dan ketersediaan *(availability)*

   Aplikasi IoT harus andal, tetap dapat beroperasi meskipun terjadi gangguan di beberapa perangkat. Selain itu, aplikasi IoT harus dapat memulihkan diri dalam waktu sesingkat mungkin untuk menjamin ketersediaan.

2.7 Fog Computing

Sistem IoT tradisional umumnya bersifat *cloud-centric*, dimana semua data dikumpulkan, diproses dan dikelola di sisi awan *(cloud)* (Taivalsaari & Mikkonen, 2017). Kelemahan sistem ini adalah tidak didesain untuk aplikasi yang *latency-sensitive* (membutuhkan respon cepat), mengirimkan data secara langsung ke awan akan menyebabkan *delay* yang cukup lama, terlebih jika data yang dikirimkan jumlahnya besar (Dastjerdi & Buyya, 2016). Untuk dapat menghasilkan sistem IoT yang *low-latency*, konsumsi *bandwidth* yang sedikit, dan skalabilitas yang tinggi, tren baru dalam pembangunan sistem IoT saat ini adalah melakukan pemrosesan data dekat dengan dimana data tersebut dihasilkan *(edge)* (Yigitoglu et al., 2017), tren ini dikenal dengan istilah *fog computing* atau *edge computing*.

Fog computing ini tidak dimaksudkan untuk mengganti infrastruktur berbasis *cloud* secara keseluruhan, melainkan untuk meningkatkan kapasitasnya dengan memberdayakan sumber daya komputasi dan penyimpanan yang tersedia

di *edge* dengan mengadopsi *platform* yang menyediakan lapisan komputasi, jaringan, dan penyimpanan sementara.

Selain dapat memberikan respon yang lebih cepat dan dapat mengirimkan data secara *real-time*, penggunaan *fog computing* juga memberikan keleluasan dalam memilih perangkat keras, menawarkan kemampuan interoperabilitas (perangkat, aplikasi, protokol), dan memberikan fitur keamanan dan proteksi data (Farahani et al., 2018). Selain itu, menurut (Al-Fuqaha et al., 2015) ada beberapa alasan yang dapat dijadikan pertimbangan mengapa pengembang aplikasi IoT harus menggunakan *fog computing*, yaitu:

a) Lokasinya yang berada diantara fisik sensor dengan *cloud* memberikan kinerja kecepatan yang lebih baik.
b) Kemampuan penyimpanan, pemrosesan, dan komunikasi *fog* memang terbatas jika dibandingkan dengan *cloud,* namun pengembang dapat menggunakan distribusi *fog* untuk mengatasinya. Cara ini menimbulkan lebih sedikit gesekan apabila menggunakan beberapa *cloud*.
c) Skalabilitas yang lebih baik. Apabila terdapat penambahan jumlah perangkat IoT, pengembang tinggal menambahkan jumlah *fog*. Pada layanan *cloud* terdapat batasan jumlah perangkat IoT yang terhubung, jika ingin menambah jumlah perangkat IoT maka pengembang diharuskan membayar lebih.
d) Mudah dimobilitas.
e) *Fog* memberikan kemampuan beroperasi dengan banyak penyedia layanan *cloud.*
f) *On the fly analysis. Fog* dapat mengumpulkan dan melakukan sebagian pemrosesan data sebelum dikirim ke *cloud* untuk diproses lebih lanjut.

2.8 Skalabilitas

Skalabilitas mengacu pada kapasitas sebuah sistem, jaringan, atau proses untuk beradaptasi dengan perubahan mendadak. Sistem tersebut harus mampu menangani peningkatan sejumlah besar pengguna, volume data, dan beban kerja

lainnya (Pattabhiraman, Bai, & Tsai, 2011). Pengujian skalabilitas mensimulasikan berbagai skenario sebagai cara untuk memastikan bahwa sistem siap dalam situasi apa pun.

Pengujian skalabilitas sangatlah diperlukan (Arsenault, n.d.), hal ini karena terkait dengan masalah mempertahankan kinerja. Kinerja merupakan kesan pertama yang dinilai pengguna dari suatu sistem. Sistem dengan kinerja buruk dapat merusak citra *brand* sistem tersebut. Misalnya, ketika sebuah toko online mengiklankan penjualan besar-besaran, mereka harus bersiap untuk peningkatan lalu lintas data. Jika situs web yang dimiliki tidak terskalakan dengan baik, maka akan terjadi *crash.* Kondisi tersebut akan membuat calon pelanggan yang sebelumnya ingin berbelanja pergi.

Tujuan dari pengujian skalabilitas sederhananya ialah untuk menentukan pada titik mana aplikasi berhenti *scaling,* menentukan batas pengguna aplikasi, menilai degradasi dari sisi klien dan pengalaman pengguna akhir, serta ketahanan dan degradasi dari sisi server di bawah kondisi beban berat. Adapun indikator yang dapat digunakan untuk menguji skalabilitas sistem adalah sebagai berikut (Barzu, Carabas, & Tapus, 2017):

a) Indikator sumberdaya komputasi; seperti alokasi penggunaan CPU, memori, jaringan, *disk,* dan lain-lain.
b) Indikator beban kerja; seperti jumlah pengguna, *request* per detik, transaksi per detik, *hits* per second, jumlah data terkirim dan yang diterima.
c) Indikator performa; seperti kecepatan (waktu) pemrosesan, keandalan dan ketersediaan sistem.

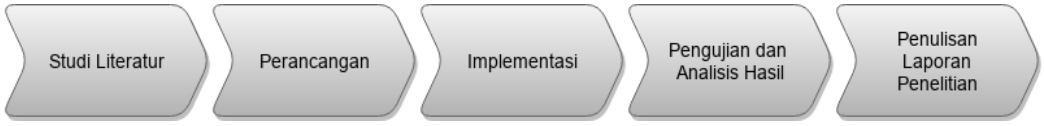
Terdapat beberapa kakas bantu (tools) baik *open source* maupun *close source* yang dapat digunakan untuk menguji skalabilitas suatu sistem, diantaranya Apache JMeter, HP Performance Tester (Load Runner), Web Load, Neo Load, Load View, dan lain-lain. Setelah melakukan pengujian, maka dapat dilakukan peningkatan kapasitas untuk meningkatkan performa sistem dalam hal skalabilitas dengan menambah sumber daya suatu sistem, baik secara horizontal (misalnya

dengan menambah jumlah *server*) maupun vertikal (misalnya dengan mengganti prosesor dengan kemampuan lebih baik).

[Halaman ini sengaja dikosongkan]

BAB 3
METODOLOGI PENELITIAN

Bab ini menjelaskan tentang metodologi yang akan dilalui pada penelitian ini guna mencapai tujuan yang diharapkan. Adapun alur penelitiannya terdiri dari lima tahap, meliputi: (1) studi literatur, (2) perancangan, (3) implementasi sistem, (4) pengujian dan analisis hasil, dan (5) penulisan laporan penelitian. Ilustrasi alur metodologi penelitian dapat dilihat pada Gambar 3.1. Penjelasan dari tahapan metodologi penelitian akan diterangkan secara terperinci pada sub-bab berikutnya.



Gambar 3.1 Alur metodologi penelitian

3.1 Studi Literatur

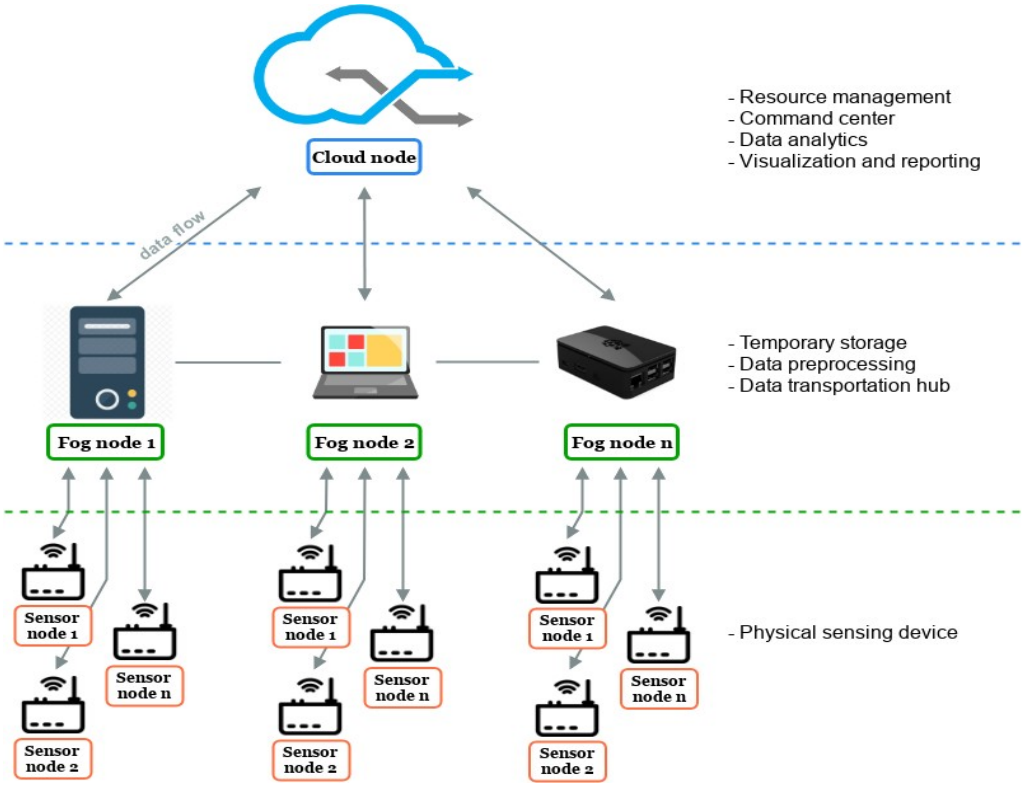
Tahap yang pertama kali dilakukan adalah mencari referensi yang kira-kira diperlukan untuk dapat menyelesaikan penelitian ini. Referensi diperoleh dari sejumlah literatur seperti buku, jurnal atau konferensi internasional terindeks. Literatur yang dipilih merupakan literatur yang terkini dan terkait dengan penelitian yang dilakukan. Dalam hal ini mencakup tentang: *electronic nose,* sensor gas, mikrokontroler Arduino, pemrosesan sinyal, pengenalan pola, normalisasi, seleksi fitur, klasifikasi, *wireless sensor network*, *internet of things*, *cloud computing*, *fog computing*, dan skalabilitas.

3.2 Perancangan

3.2.1. Arsitektur Sistem

Untuk menghadirkan sistem yang fleksibel dan terskalakan sesuai tujuan dari penelitian ini, penulis mengusulkan arsitektur yang berbasis *fog*. Gambar 3.2 menampilkan struktur dari sistem secara keseluruhan, dimana ia terdiri dari tiga

komponen utama, yakni: *sensor node, fog node,* dan *cloud node.* Masing-masing memiliki fungsinya sendiri.

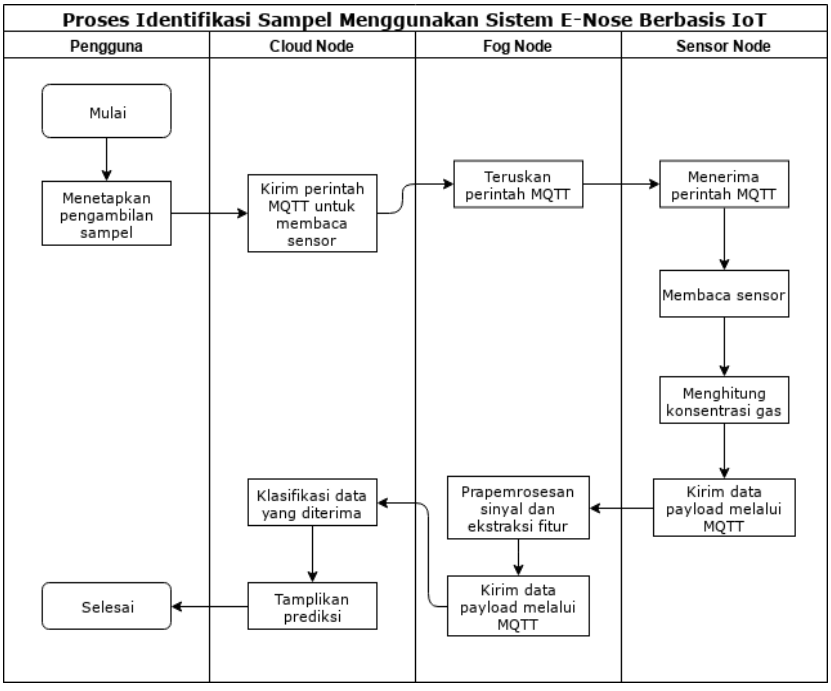


Gambar 3.2 Rancangan arsitektur sistem e-nose berbasis IoT

Komponen *sensor node* merujuk pada perangkat *e-nose,* bertanggung jawab membaca dan mengirimkan sampel aroma berupa data sinyal hasil respon sensor. Data tersebut dikirimkan ke *fog node* melalui jaringan sensor nirkabel. *Fog node*, atau yang bisa juga disebut perangkat gerbang, merupakan komponen bertanggung jawab dalam mengumpulkan, menyimpan, melakukan prapemrosesan data sampel aroma dan mengirimkan hasilnya ke *cloud node* melalui sambungan internet. Adapun prapemrosesan yang dilakukan diantaranya: pengkondisian sinyal, ekstraksi fitur, normalisasi fitur, dan seleksi fitur. Selain itu, *fog node* juga bertanggung jawab meneruskan perintah pembacaan sampel dari *cloud node* ke *sensor node* tujuan. Terakhir, *cloud node.* Komponen ini berfungsi sebagai *platform* untuk meletakkan perangkat lunak pengenalan pola sinyal yang sudah dipraproses sebelumnya dan perangkat lunak antarmuka sistem. Melalui

antarmuka ini pengguna dapat melatih algoritma pembelajaran mesin, mendaftarkan perangkat *sensor node*, memerintahkan *sensor node* untuk membaca sampel, dan melihat hasil prediksi sampel aroma dari *sensor node* tersebut.

Adapun rancangan proses sistem dalam memprediksi sampel, dapat dilihat pada Gambar 3.3. Alur pertama ialah pengguna memerintahkan pengambilan sampel. Selanjutnya, *cloud node* akan mengirimkan perintah pembacaan sensor gas ke *sensor node* yang diinginkan pengguna, perintah ini diteruskan oleh *fog node*. Saat menerima perintah, *sensor node* mulai membaca nilai tegangan sensor, respon dari paparan aroma sampel, lalu menghitung konsentrasi gas, dan mengirimkan data tersebut ke *fog node* dalam periode waktu yang sudah ditentukan. Ketika datanya sudah lengkap, *fog node* kemudian melakukan prapemrosesan data, hasil ini kemudian dikirimkan ke *cloud node.* Terakhir, *cloud node* melakukan proses klasifikasi berdasarkan model pembelajaran mesin yang telah dibangun sebelumnya dan menampilkan hasil prediksinya kepada pengguna dalam bentuk tabel atau grafis.



Gambar 3.3 Rancangan diagram alur proses identifikasi sampel

Pada sistem usulan, digunakan protokol pesan MQTT *(Message Queue Telemetry Transport)* untuk komunikasi data. Protokol MQTT dinilai cocok untuk berbagai proyek IoT, karena ia dirancang khusus untuk perangkat yang memiliki sumber daya terbatas. Protokol ini menggunakan energi dan *bandwith* yang sangat sedikit jika dibandingkan dengan protokol lainnya (seperti HTTP dan REST). Protokol ini juga dapat menjadi jawaban untuk kebutuhan aplikasi yang membutuhkan *latency* rendah.

Berbeda dengan protokol seperti HTTP yang menggunakan paradigma komunikasi request / response, protokol MQTT menggunakan paradigma *publish / subscribe*. Dalam paradigma *publish / subscribe*, terdapat dua jenis klien, yaitu klien yang mengirimkan pesan dan klien yang menerima pesan. Suatu klien tidak perlu mengetahui keberadaan klien yang lain, klien hanya mempublikasikan sebuah pesan dengan tipe data tertentu (yang kemudian disebut topik) hanya kepada klien yang berminat menerima publikasi topik tersebut. Paradigma tersebut dapat dianalogikan seperti proses pengiriman suatu surat kabar / majalah, yang hanya ditujukan kepada pelanggan berminat berlangganan surat kabar / majalah tersebut.

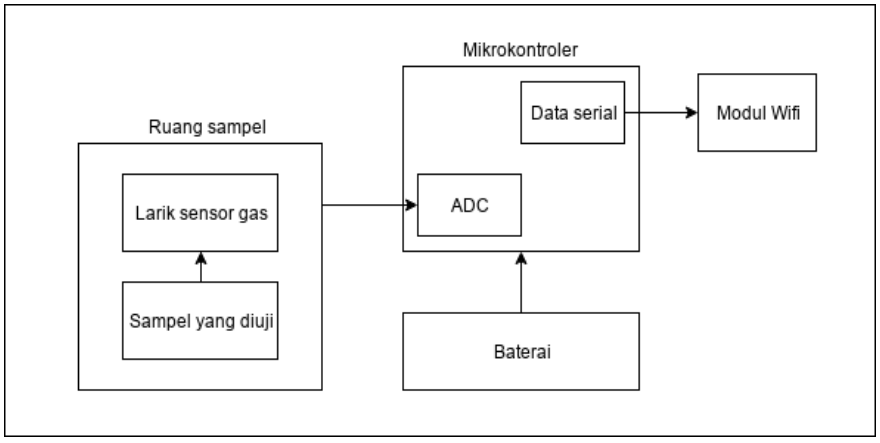
Sistem komunikasi dengan paradigma *publish / subscribe* ini memerlukan semacam agen distribusi, di dalam MQTT ia disebut *broker*. Semua klien harus memiliki koneksi dengan *broker*. Klien yang mengirim pesan kepada *broker*, disebut sebagai *publisher*. *Broker* akan menyaring pesan-pesan yang masuk dan mendistribusikannya kepada klien yang tertarik menerima pesan-pesan tersebut. Klien yang tertarik menerima pesan tersebut dan sebelumnya telah mendaftarkan diri ke *broker* disebut sebagai *subscriber*.

3.2.2. Perangkat *E-Nose*

Secara umum, perangkat *e-nose* yang dirancang terbagi atas dua bagian, yakni perangkat keras dan perangkat lunak. Perangkat keras dari *e-nose* terdiri dari larik sensor gas, ruang sampel mikrokontroler, modul komunikasi nirkabel,

dan baterai. Sedangkan untuk perangkat lunak, terdiri atas program menghitung konsentrasi gas dan program komunikasi data menggunakan protokol MQTT.

Diagram blok dari prototipe e-nose secara keseluruhan ditunjukkan pada Gambar 3.4. Dari diagram blok sistem, dapat dilihat bahwa cara kerja dari perangkat *e-nose* adalah dengan mendeteksi sampel menggunakan larik sensor yang berada pada ruang sampel. Larik sensor yang terdiri dari beberapa sensor semikonduktor, dipaparkan aroma sampel yang diuji. Setiap sensor kemudian akan merespon aroma dengan menghasilkan nilai tegangan yang berbeda-beda bergantung terhadap tingkat kesegaran sampel. Nilai tegangan dari sensor-sensor yang digunakan akan dibaca oleh ADC (*analog to digital converter*) Arduino. Nilai tegangan tersebut selanjutnya diproses untuk mendapatkan nilai konsentrasi gas yang berada di dalam ruang sampel. Data nilai konsentrasi gas inilah yang kemudian dikirimkan ke komputer *(fog node)* melalui modul komunikasi nirkabel untuk selanjutnya dianalisa.



Gambar 3.4 Rancangan *sensor node* (tampak atas)

Untuk menghitung konsentrasi gas, dibutuhkan beberapa langkah. Yang pertama menghitung nilai resistensi sensor saat terpapar gas (*Rs*), dengan persamaan (3.1) dan (3.2) berikut:

$$Rs = \frac{Vc - VRL}{VRL} \times RL \tag{3.1}$$

$$VRL = \frac{ADC \times Vc}{1023} \tag{3.2}$$

Dimana *ADC* adalah nilai hasil dari *analog to digital conversion* yang diperoleh dari Arduino, *Vc* adalah nilai tegangan pada Arduino yang diukur dengan alat multimeter, *VRL* adalah tegangan sensor saat terpapar gas, dan *RL* adalah resistensi sensor yang didapat dari *datasheet*.

Setelah mendapatkan nilai resistensi sensor, baru kemudian dapat digunakan persamaan (3.3) untuk menghitung konsentrasi gas dalam satuan *parts per million* (ppm).

$$C = 10^{(\log(\frac{\frac{Rs}{Ro} - \beta}{\gamma}) + \alpha)} \tag{3.3}$$

Ro adalah nilai resistensi sensor pada udara bersih tanpa terpapar gas, nilai ini dapat dilihat pada setiap *datasheet* sensor gas MQ. Sementara, nilai *α, β,* dan *γ* didapatkan dari proses *curve fitting* menggunakan perangkat lunak LibreOffice Calc.

Perangkat *e-nose* biasanya terdiri dari 8-32 larik sensor, bahkan lebih (Climent, Pelegri-Sebastia, Sogorb, Talens, & Chilo, 2017). Penggunaan sensor dengan jumlah banyak pada perangkat e-nose membuat beban komputasi menjadi lebih tinggi, meningkatkan jumlah data yang harus dikirim, dan menghabiskan daya yang cukup besar. Hal ini menjadikan *e-nose* sulit untuk diterapkan menjadi *sensor node*, karena pada umumnya sistem IoT terdiri dari perangkat dengan sumberdaya terbatas. Selain itu, menurut (Wijaya et al., 2017) penggunaan sensor yang banyak pada *e-nose* juga berpotensi menurunkan kinerja pembelajaran mesin. Untuk itulah penggunaan jumlah sensor pada *e-nose* mesti ditekan.

Untuk mencari sensor paling relevan, dalam rangka mengurangi jumlah penggunaan sensor, kami melakukan langkah-langkah optimisasi sebagai berikut:

1) Impor data

   Mengimpor data respon sensor gas yang tersimpan di basis data.

2) Pengkondisian sinyal

   Menghilangkan *noise* yang terdapat pada data respon sensor gas menggunakan metode *discrete wavelet transform* (DWT) *daubechies*

family. Terdapat pilihan *daubechies* yang bisa digunakan, mulai dari db1 sampai db38. Namun, pada penelitian ini dibatasi sampai db15.

3) Ekstraksi fitur statistik

Data respon sensor gas yang telah bersih dari *noise* diekstraksi menjadi fitur dengan memanfaatkan lima parameter statistik, yaitu: nilai maksimum, rata-rata, *variance*, standar deviasi, dan nilai *skewness*. Perhitungan untuk mendapatkan empat fitur statistik terakhir dapat dilihat pada persamaan (3.4), (3.5), (3.6), (3.7).

$$mean(\mu) = \frac{1}{N}\sum_{n=1}^{N} x_n \tag{3.4}$$

$$var = \frac{1}{N-1}\sum_{n=1}^{N} (x_n - \mu)^2 \tag{3.5}$$

$$std(\sigma) = \sqrt{var} \tag{3.6}$$

$$skew = \frac{\frac{1}{N}\sum_{n=1}^{N} (x_n - \mu)^3}{\sigma^3} \tag{3.7}$$

Dimana X_n adalah nilai sinyal (respon gas sensor dalam ppm) dan N adalah banyaknya data.

4) Penyamaan skala fitur (normalisasi)

Langkah ini bertujuan menyamakan rentang data sehingga tidak ada satu atribut yang terlalu dominan atas atribut yang lain. Normalisasi data dilakukan dengan menggunakan metode *min-max scaling*. Menggunakan persamaan (3.8) dihasilkan nilai standar yang berada di kisaran angka 0 sampai 1.

$$z = \frac{x - min(x)}{max(x) - min(x)} \tag{3.8}$$

Dimana z adalah nilai standar yang sudah dinormalisasi, sedangkan x nilai awal, *min* nilai minimum pada fitur, dan *max* adalah nilai maksimum pada fitur.

5) Seleksi fitur

Tujuan utama teknik seleksi fitur adalah mencari kombinasi optimal dari fitur statistik setiap sensor atau atribut optimal dari fitur statistik yang didapatkan sebelumnya. Pada penelitian ini, metode seleksi fitur yang digunakan adalah *chi-squared statistic*, ia digunakan untuk menghitung 25, 20, 15, 10, dan 5 fitur terbaik. Untuk menghitung nilai *chi-squared statistic* digunakan persamaan (3.9) berikut.

$$X^2 = \sum_{n=1}^{N} \frac{(O_i - E_i)^2}{E_i} \tag{3.9}$$

Dimana X^2 adalah nilai *chi-squared statistic,* O_i adalah frekuensi hasil observasi kelas *i*, dan E_i adalah frekuensi kelas *i* yang diharapkan.

6) Evaluasi performa

Pada langkah optimasi larik sensor ini, penulis menerapkan dan membandingkan tujuh pendekatan algoritma klasifikasi, yakni: *Logistic Regression, KNN, Decission Tree, Naive Bayes, Support Vector Machine, Linear SVC,* dan *Random Forest*. Pendekatan tersebut dievaluasi menggunakan data *ground-truth* sampel yang telah diambil sebelumnya dengan metode validasi silang *k-fold cross validation.*

7) *Tuning* parameter dan waktu pengambilan sampel

Performa model *machine learning* masih bisa ditingkatkan lagi dengan menyesuaikan parameternya, misal merubah jumlah *neighbors* pada algoritma *K-Nearest Neighbors* (KNN) atau merubah jumlah *tree* pada algoritma *Random Forest*. Selain itu, pada penelitian ini juga melihat pengaruh penggunaan 10, 8, 6, 4, dan 2 menit data yang masuk terhadap performa.

3.2.3. Pengambilan Sampel

Pada penelitian ini, penulis menggunakan pisang sebagai sampel. Pisang merupakan buah yang paling banyak diproduksi di Indonesia. Berdasarkan data yang dirilis Badan Pusat Statistik (BPS), produksi pisang pada 2015 mencapai

angka 7,29 juta ton, urutan kedua ditempati mangga dengan jumlah produksi mencapai 2,1 juta ton, disusul jeruk di urutan ketiga dengan produksi 1,74 juta ton. Sementara itu, menurut FAO *(Food and Agriculture Organization)*, Indonesia berada pada posisi keenam sebagai produsen pisang di dunia dengan kontribusi 5,67% dari total produksi pisang dunia.

Tidak tersedianya dataset respon sensor gas terhadap aroma pisang, mau tidak mau mengharuskan penulis mengambil data *ground-truth* menggunakan perangkat *e-nose*. Adapun jenis pisang yang digunakan sebagai sampel pada penelitian ini adalah pisang kepok. Kelas data *ground-truth* sampel pisang kepok ditentukan sesuai warna dan kondisinya. Pisang dengan warna kuning, terdapat memar, dan bertekstur lembek dikategorikan matang. Sedangkan pisang dengan warna hijau, memiliki getah dan bertekstur keras dikategorikan mentah. Data *ground-truth* yang dikumpulkan berjumlah 100, dengan distribusi 50 pisang kepok matang dan 50 pisang kepok mentah.

3.2.4. Pengujian

Terdapat 2 jenis pengujian dan evaluasi sistem yang akan dilakukan, yakni pengujian fungsionalitas, dan pengujian skalabilitas. Berikut penjelasan mengenai kedua pengujian tersebut dan bagaimana skenario melakukannya.

1. Pengujian fungsionalitas

Pada jenis uji coba ini, sistem diverifikasi apakah bisa berfungsi sesuai yang diharapkan. Uji coba ini menggunakan metode *blackbox*, melihat keluaran yang dihasilkan setiap melakukan pola masukan pada antarmuka sistem. Selain itu, pengujian ini juga mengukur akurasi klasifikasi sampel. Kemampuan sistem menggunakan satu unit produk final *e-nose* dilihat, apakah data sampel yang dihasilkan *e-nose* berhasil diproses dan diprediksi dengan benar.

2. Pengujian skalabilitas

Uji coba ini bertujuan untuk menilai seberapa baik dan seberapa mampu sistem usulan terskalakan. Pengujian dilakukan dengan simulasi penambahan beban. 25, 50, 100, 150, 200, 250, 300, 350, 400, 450, dan 500 *sensor node* secara

bertahap dihubungkan ke sistem. Selain itu beban dari segi jumlah data yang dikirim juga akan dimasukkan ke skenario pengujian. Jumlah sampel terkirim, rata-rata waktu repon, dan nilai *throughput* diamati untuk menilai skalabilitas sistem. Ketiganya dapat dihitung dengan persamaan (3.10), (3.11), dan persamaan (3.12). Selain itu, jumlah data yang benar-benar tersimpan di basis data juga turut diamati.

$$Package\,lost = sent\ package - arrived\ package \tag{3.10}$$

$$Avg.response\,time = \frac{Total\,time\,taken\,by\,all\,packages}{number\,of\ packages} \tag{3.11}$$

$$Throughput = \frac{number\,of\ packages}{total\,time} \tag{3.12}$$

BAB 4
HASIL DAN PEMBAHASAN

Pada pembahasan ini diberikan pemaparan mengenai implementasi sistem serta pengujian dari sistem berdasarkan skenario yang telah dirancang pada Bab Tiga. Proses implementasi dilakukan berdasarkan tahapan yang telah diberikan pada pembahasan sebelumnya. Selanjutnya pengujian sistem dilakukan dengan beberapa kondisi yang disesuaikan dengan skenario pengujian. Dari hasil pengujian yang telah didapatkan, selanjutnya diberikan pembahasan dan analisa dari setiap pengujian yang dilakukan dengan tujuan untuk mendapatkan hasil dari penelitian, sehingga mendapatkan kesimpulan yang diberikan pada pembahasan selanjutnya.

4.1 Implementasi

4.1.1. Prototipe *E-Nose*

Berdasarkan rancangan, dihasilkan sebuah prototipe *e-nose*. Prototipe ini menggunakan Arduino MEGA sebagai mikrokontroller, dan sensor MQ sebagai pendeteksi molekul bau (gas). Daftar lengkapnya dapat dilihat pada tabel berikut.

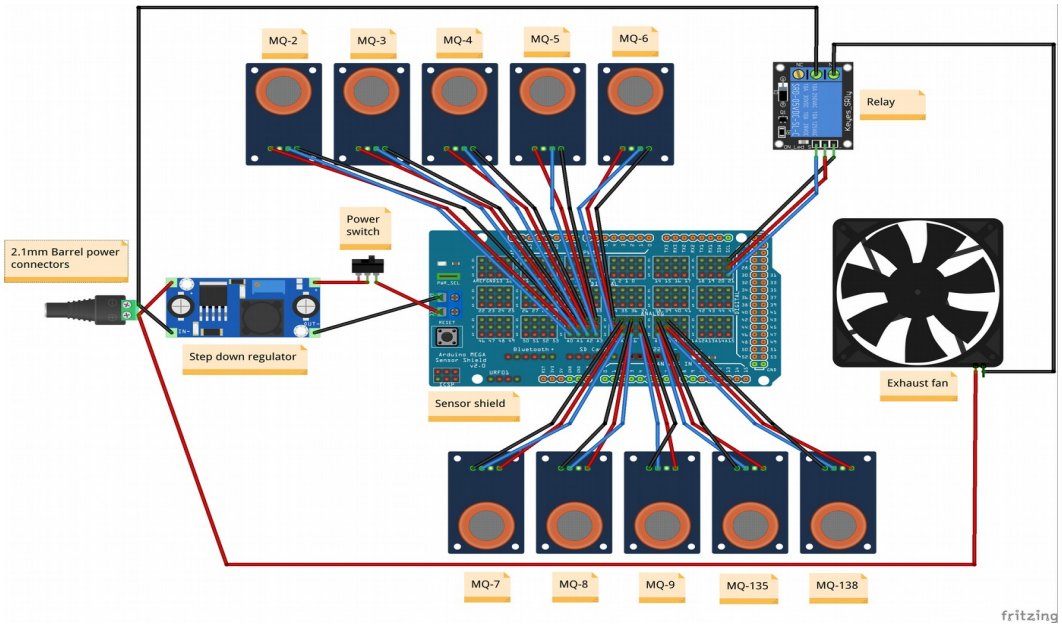
Tabel 4.1 Daftar alat dan bahan untuk pembuatan prototipe *e-nose*

No	Nama Alat/Bahan	Jumlah	Fungsi
1	Kotak sampel	1 buah	Sebagai tempat meletakkan sampel dan mengisolasi gas dari pengaruh suhu dan kelembaban lingkungan sekitarnya
2	Arduino MEGA 2560	1 buah	Mikrokontroller untuk memberi daya dan membaca tegangan sensor gas
3	MEGA sensor shield	1 buah	Membagi tegangan dari Arduino ke sensor
4	Kabel jumper female to female	33 buah	Menghubungkan sensor dan relay ke sensor shield
5	Sensor MQ-2, MQ-3, MQ-4, MQ-5, MQ-6, MQ-7, MQ-8, MQ-9,	Masing-masing 1	Mendeteksi kandungan gas di udara

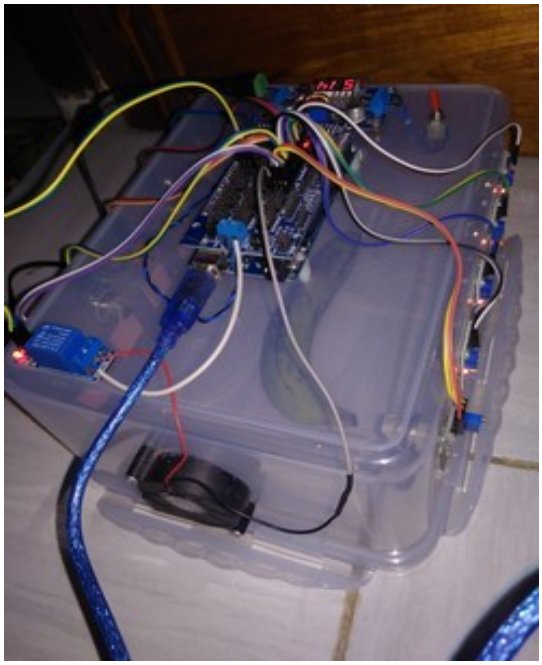
	MQ-135, MQ-138		
6	Relay	1 buah	Menyalakan kipas buang
7	Kipas buang	1 buah	Membersihkan kotak sampel dari sisa gas sampel sebelumnya
8	Step down regulator	1 buah	Mengurangi dan menstabilkan tegangan yang dihasilkan adaptor
9	Saklar	1 buah	Menyalakan dan mematikan peralatan
10	Adaptor power supply	1 buah	Memberikan daya untuk seluruh peralatan
11	2.1mm barrel power connectors	1 buah	Menghubungkan adaptor dengan step down regulator
12	Kabel serat ukuran 1 mm	1 meter	Menghubungkan kipas buang ke sumber listrik (stepdown regulator)

Prototipe *e-nose* ini menggunakan sepuluh jenis sensor MQ mulai dari MQ-2 sampai MQ-138. Sensor mq-2 dapat mendeteksi gas *methane, butane, lpg,* dan asap. Sensor MQ-3 dapat mendeteksi gas *alcohol, ethanol,* dan asap. Sensor mq-4 dapat mendeteksi gas *methane*, dan *CNG*. Sensor mq-5 dapat mendeteksi gas alam dan *LPG*. Sensor mq-6 dapat mendeteksi gas *LPG* dan *butane*. Sensor MQ-7 dapat mendeteksi gas karbon monoksida. Sensor MQ-8 dapat mendeteksi gas hidrogen. Sensor MQ-9 dapat mendeteksi gas karbon monoksida dan gas mudah terbakar. Sensor MQ-135 dapat mendeteksi *benzene,* alkohol dan asap. Sementara sensor MQ-138 dapat mendeteksi gas *benzene, toluene*, alkohol, *acetone, propane, formaldehyde* dan hidrogen.

Adapun skematik rangkaian *e-nose* ditunjukkan pada gambar 4.1, sementara hasil dari implementasi prototipe e-nose dapat dilihat pada Gambar 4.2 berikut.



Gambar 4.1 Skematik rangkaian alat



Gambar 4.2 Perangkat *e-nose* dengan 10 sensor MQ

Prototipe e-nose selanjutnya dihubungkan ke sumber listrik melalui *adaptor power supply*, dan dihubungkan ke laptop/PC melalui kabel USB untuk

mengumpulkan data *ground-thruth*. Adapun serangkaian proses yang dilakukan setiap pengambilan satu sampel ialah sebagai berikut: (1) Inisialisasi program akuisisi data; (2) Mengirimkan perintah pembersihan ruang sampel; (3) Meletakkan sampel baru ke dalam ruang sampel; (4) Memerintahkan mikrokontroler untuk melakukan pembacaan sampel; (5) Mengeluarkan sampel setelah selesai.

Proses diatas dilakukan berulang kali sebanyak jumlah sampel yang telah ditetapkan sebelumnya, yakni 50. Dari setiap sampel yang terkumpul di basis data, diperoleh sebanyak 54 kolom data yakni konsentrasi senyawa volatil dan 300 baris data dari pembacaan sampel selama 10 menit dengan jeda 2 detik. Jika ditotal seluruhnya, terdapat 810.000 data yang tersimpan di mongoDB.

4.1.2. Optimisasi Larik Sensor

Data yang dikumpulkan melalui prototipe *e-nose* kemudian dianalisis. Sebagaimana yang dijelaskan sebelumnya pada bab 3, langkah pertama yang dilakukan ialah pengkondisian sinyal. Langkah ini bertujuan untuk menghilangkan *noise* dari data *ground-truth*. *Noise* dapat terjadi ketika pada saat pengambilan sampel terdapat pengaruh seperti angin, sisa residu sampel sebelumnya, senyawa volatil lain, arus listrik yang tidak stabil, atau kemampuan sensor yang sudah melemah. Biasanya, jika terdapat *noise* pada sinyal, ia akan membentuk sudut yang tajam. Teknik ini dapat membuat sinyal yang dihasilkan menjadi lebih halus.

Di Python sendiri terdapat pustaka untuk pemrosesan sinyal *discrete wavelet transform* (DWT). Pada pustaka ini, tersedia 38 pilihan *daubechies family* yang berbeda (db1-db38). Dimana, semakin tinggi nilainya, semakin halus pula sinyal yang dihasilkan. Akan tetapi hal ini justru berdampak pada semakin hilangnya informasi. Untuk itu perlu dilakukan percobaan untuk menemukan pilihan *daubechies* mana yang paling baik dalam menghilangkan *noise.* Dimulai dengan menetapkan pilihan *daubechies,* ektraksi fitur statistik, normalisasi fitur, membangun model pembelajaran mesin, kemudian melakukan evaluasi akurasi

yang dihasilkan. Dari 38 *daubechies familiy* yang tersedia, percobaan dibatasi hanya sampai db15 saja.

Ekstraksi fitur statistik menghasilkan 270 fitur untuk masing-masing sampel, sementara normalisasi fitur menghasilkan rentang data hasil ekstraksi fitur antara 0-1. Dari percobaan ditemukan bahwa *daubechies* yang menghasilkan kinerja paling baik ialah db6. Selain itu, dari proses evaluasi performa dapat diketahui bahwa kinerja sebagian besar model pembelajaran mesin mengalami peningkatan ketika menggunakan data yang sudah dilakukan pengkondisian sinyal dan normalisasi fitur sebelumnya, sebagaimana yang dapat dilihat pada Tabel 4.2.

Tabel 4.2 menunjukkan perbandingan akurasi algoritma antara data mentah dengan data yang sudah dipraproses (pengkondisian sinyal; PS, dan normalisasi fitur; NF). Sebagaimana yang terlihat, ada empat algoritma pembelajaran mesin yang mengalami peningkatan kinerja, dua menurun, dan satu tidak berubah. Tapi secara garis besar, kinerjanya mengalami peningkatan yang cukup berarti.

Tabel 4.2 Akurasi *raw data* dengan *preprocessed data*

Algoritma	Akurasi Data Mentah	Akurasi Data PS+NF
LR	0.67	0.6
KNN	0.5	0.6
CART	0.53	0.59
NB	0.57	0.44
SVM	0	0
Linear SVC	0.63	0.71
Random Forest	0.56	0.59

Langkah kedua dari analisis data yang dilakukan adalah seleksi fitur. Bertujuan untuk meningkatkan efisiensi penggunaan sensor. Dari hasil kegiatan ini, diketahui bahwa kinerja model pembelajaran mesin bervariatif. Saat kembali dilakukan evaluasi performa model pembelajaran mesin menggunakan 25, 20, 15, 10, dan 5 fitur, ada yang menghasilkan kinerja lebih baik ketika menggunakan 270 fitur, ada pula yang menurun. Tabel 4.3 memperlihatkan perbandingan tujuh algoritma ketika dilakukan pengurangan fitur. Algoritma KNN dan Random

Forest memperoleh kinerja yang paling tinggi dibanding algoritma lainnya, dua algoritma ini mendapatkan kinerja terbaiknya ketika menggunakan 15 fitur. 15 fitur ini dihasilkan oleh 4 sensor MQ, yakni MQ2, MQ3, MQ6, dan MQ7. Detailnya dapat dilihat pada Tabel 4.4.

Tabel 4.3 Akurasi algoritma menggunakan fitur terpilih

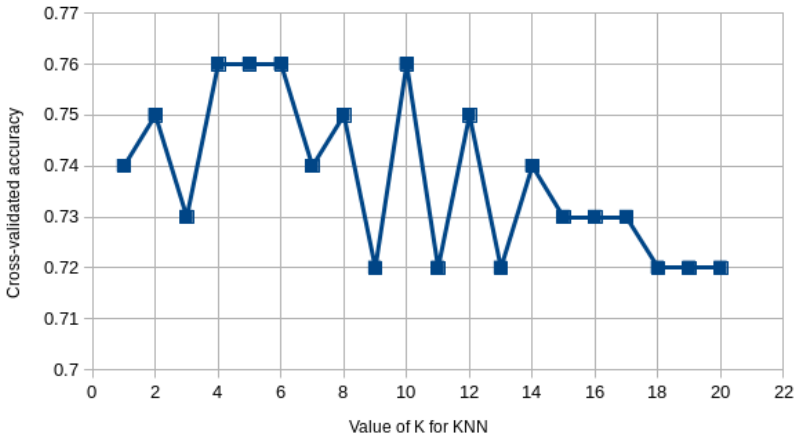
Algoritma	25 Fitur	20 Fitur	15 Fitur	10 Fitur	5 Fitur
LR	0.62	0.62	0.61	0.61	0.54
KNN	0.74	0.74	0.76	0.75	0.65
CART	0.65	0.66	0.65	0.67	0.64
NB	0.55	0.58	0.6	0.65	0.63
SVM	0.48	0.49	0.51	0.54	0.45
Linear SVC	0.64	0.63	0.62	0.62	0.63
Random Forest	0.7	0.69	0.79	0.7	0.71

Tabel 4.4 Daftar fitur paling berpengaruh

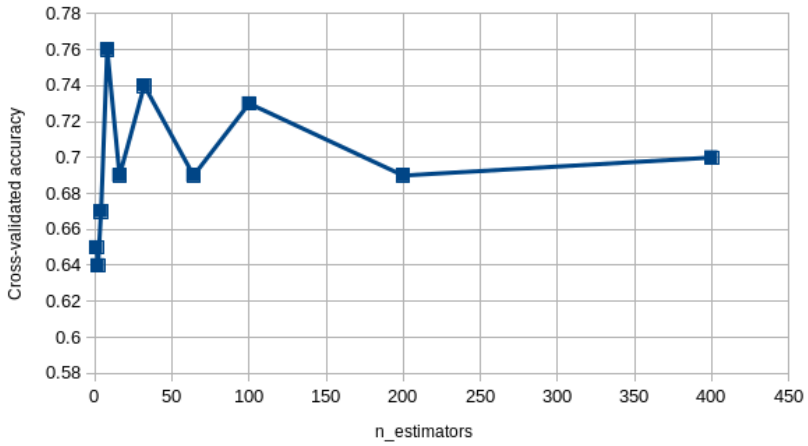
No	25 Fitur	20 Fitur	15 Fitur	10 Fitur	5 Fitur
1	MQ2_LPG_skew	MQ2_LPG_skew	MQ2_LPG_skew	MQ2_LPG_skew	MQ3_CO_skew
2	MQ2_CO_skew	MQ2_SMOKE_skew	MQ2_H2_skew	MQ2_H2_skew	MQ6_CH4_skew
3	MQ2_SMOKE_skew	MQ2_H2_skew	MQ2_PROPANE_skew	MQ3_CO_skew	MQ6_H2_skew
4	MQ2_ALCOHOL_skew	MQ2_PROPANE_skew	MQ3_CO_skew	MQ3_LPG_skew	MQ6_LPG_skew
5	MQ2_CH4_skew	MQ3_CH4_var	MQ3_LPG_skew	MQ6_CH4_skew	MQ7_H2_skew
6	MQ2_H2_skew	MQ3_CO_skew	MQ6_CH4_std	MQ6_H2_std	
7	MQ2_PROPANE_skew	MQ3_LPG_skew	MQ6_CH4_skew	MQ6_H2_skew	
8	MQ3_CH4_mean	MQ4_CO_var	MQ6_H2_std	MQ6_LPG_skew	
9	MQ3_CH4_var	MQ6_CH4_std	MQ6_H2_skew	MQ7_CO_skew	
10	MQ3_CO_skew	MQ6_CH4_skew	MQ6_LPG_skew	MQ7_H2_skew	
11	MQ3_HEXANE_var	MQ6_H2_std	MQ7_CO_skew		
12	MQ3_LPG_skew	MQ6_H2_skew	MQ7_H2_skew		
13	MQ4_CO_var	MQ6_LPG_std	MQ7_LPG_mean		
14	MQ6_CH4_std	MQ6_LPG_skew	MQ7_LPG_std		
15	MQ6_CH4_skew	MQ7_CO_skew	MQ7_LPG_var		
16	MQ6_H2_std	MQ7_H2_std			

17	MQ6_H2_skew	MQ7_H2_skew			
18	MQ6_LPG_std	MQ7_LPG_mean			
19	MQ6_LPG_skew	MQ7_LPG_std			
20	MQ7_CO_skew	MQ7_LPG_var			
21	MQ7_H2_std				
22	MQ7_H2_skew				
23	MQ7_LPG_mean				
24	MQ7_LPG_std				
25	MQ7_LPG_var				

Langkah ketiga, yang merupakan langkah terakhir analisis data yang diperoleh dari prototipe *e-nose,* ialah melakukan *tuning* parameter dua algoritma pembelajaran mesin yang terbaik, dan *tuning* waktu pembacaan sampel (dalam hal ini jumlah data). Dari hasil tuning parameter, dapat diketahui bahwa jumlah tetangga pada algoritma KNN yang menghasilkan kinerja terbaik adalah 4, 5, 6, 10 (Gambar 4.3), sedangkan jumlah tree pada RF yang menghasilkan kinerja terbaik adalah 8 (Gambar 4.4). Kedua algoritma ini sama-sama memperoleh 76% akurasi.



Gambar 4.3 Hasil tuning KNN



Gambar 4.4 Hasil tuning RF

Sementara itu, dari hasil tuning jumlah data berdasarkan lama waktu pembacaan sampel, diperoleh akurasi tertinggi pada 8 menit (240) data jika menggunakan KNN, dan 6 menit (180) data jika menggunakan Random Forest. Tabel 4.5 menunjukkan perbandingan akurasi keduanya.

Tabel 4.5 Tabel hasil tuning waktu

	2 Menit Data	4 Menit Data	6 Menit Data	8 Menit Data	10 Menit Data
KNN	0.43	0.62	0.67	0.8	0.76
RF	0.53	0.63	0.79	0.74	0.75

Setelah mendapatkan informasi sensor mana saja yang paling relevan dan berapa lama waktu pembacaan sampel yang ideal, selanjutnya dihasilkan produk final *e-nose*. Hasil implementasinya dapat dilihat pada Gambar 4.5 berikut ini.

Gambar 4.5 Perangkat *e-nose* dengan 4 sensor MQ

Bentuk, posisi, maupun peralatan yang digunakan kurang lebih sama dengan prototipe *e-nose*, hanya saja perangkat *e-nose* akhir dibuat dengan ukuran yang lebih kecil. Selain itu, ia diberi sumber energi sendiri, dan diberikan modul untuk berkomunikasi secara nirkabel, sehingga dapat diterapkan menjadi *sensor node*. Adapun daftar perangkat yang digunakan selengkapnya dapat dilihat pada Tabel 4.6 berikut.

Tabel 4.6 Daftar alat dan bahan untuk pembuatan perangkat *e-nose* akhir

No	Nama Alat/Bahan	Jumlah	Fungsi
1	Kotak sampel	1 buah	Sebagai tempat meletakkan sampel dan mengisolasi gas dari pengaruh suhu dan kelembaban lingkungan sekitar
2	Arduino Uno	1 buah	Mikrokontroller untuk memberi daya dan membaca tegangan sensor gas
3	Uno sensor shield	1 buah	Membagi tegangan dari Arduino ke sensor
4	ESP8266 + shield	1 buah	Menghubungkan perangkat *e-nose* ke jaringan nirkabel
5	Kabel jumper female to female	15 buah	Menghubungkan sensor dan modul wifi
6	Sensor MQ-2, MQ-3, MQ-6, MQ-7	Masing-masing 1	Mendeteksi kandungan gas di udara
7	Baterai / Power bank	1 buah	Memberikan daya untuk Arduino,

			sensor, dan modul wifi
8	Kabel USB	1 buah	Menghubungkan Arduino ke baterai

4.1.3. *Fog* dan *Cloud Node*

Metode prapemrosesan dan metode klasifikasi sampel yang terpilih dari langkah optimisasi larik sensor, diimplementasikan ke dalam perangkat lunak yang ditulis dalam bahasa pemrograman Python. Perangkat lunak prapemrosesan yang terdiri dari modul penyimpanan sementara, pengkondisian sinyal, ekstraksi fitur, dan normalisasi fitur diluncurkan di *fog node*, sementara perangkat lunak untuk klasifikasi yang terdiri dari model pembelajaran mesin dan antarmuka untuk identifikasi sampel diluncurkan di *cloud node*.

Untuk perangkat kerasnya, pada penelitian ini digunakan Raspberry Pi 3 Model B sebagai *fog node.* Dengan kemampuan seperti komputer desktop, Raspberry Pi sering digunakan sebagai perangkat prototipe oleh berbagai penelitian terkait IoT sebelum beralih ke perangkat dengan kemampuan yang lebih mumpuni. Adapun spesifikasi yang dimiliki oleh Raspberry Pi 3 Model B adalah sebagai berikut:

- SoC: Broadcom BCM2837
- CPU: 4x ARM Cortex-A53, 1.2GHz
- GPU: Broadcom VideoCore IV @ 400 MHz
- RAM: 1GB LPDDR2 (900 MHz)
- Networking: 10/100 Ethernet, 2.4GHz 802.11n wireless
- Bluetooth: Bluetooth 4.1 Classic, Bluetooth Low Energy

Konfigurasi yang dilakukan pada perangkat Raspberry Pi 3 Model B untuk menjadikannya sebagai *fog node* diantaranya:

- Menginstal sistem operasi Raspbian
- Menghubungkan ke jaringan
- Menginstal Mosquitto MQTT Broker
- Mengatur MQTT Bridge
- Menginstal MongoDB

- Menginstal pustaka yang dibutuhkan
- Men-*deploy* perangkat lunak penyimpanan data, pemrosesan, dan transportasi data

Berikut hasil dokumen MongoDB keluaran dari perangkat lunak penyimpanan sementara yang terdapat di *fog node*:

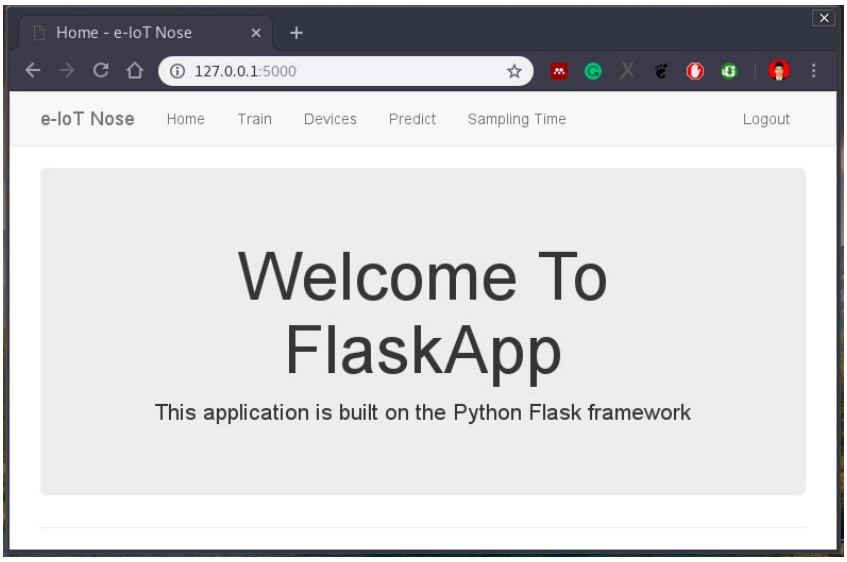
```
{
    "_id" : ObjectId("5ba72d87e2e49a23aae1806b"),
        "enose_id" : "node1",
        "sampling_time" : ISODate("2018-08-18T13:07:03.915Z"),
        "value" : [
            {
                "MQ2_LPG" : 1.52,
                "MQ2_H2" : 2.63,
                ...
                "MQ7_LPG" : 0
            },
            ...
            {
                "MQ2_LPG" : 1.52,
                "MQ2_H2" : 2.63,
                ...
                "MQ7_LPG" : 0
            }
        ]
}
```

Adapun untuk perangkat keras *cloud node,* saat ini penulis baru bisa menggunakan komputer lokal, yang berada di jaringan yang sama dengan *fog node*. Spesifikasi komputer yang mensimulasikan *cloud node* dapat dilihat pada daftar dibawah ini:

- Jenis: Laptop
- Tipe: Acer E5-475G
- Prosesor: Intel Core i5-7200U 2.5Ghz
- Memori: 8GB DDR4
- Sistem Operasi: Debian 9

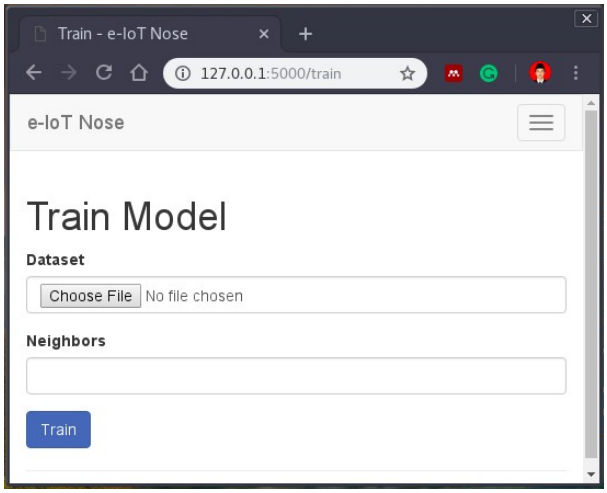
Konfigurasi yang dilakukan ialah: Menginstal Mosquitto MQTT Broker, MariaDB, pustaka yang dibutuhkan, dan meluncurkan perangkat lunak klasifikasi beserta antarmuka. Hasil implementasi antarmuka untuk identifikasi sampel, dapat dilihat pada tangkapan layar berikut. Perangkat lunak / aplikasi ini dibangun berbasis web, menggunakan Flask; sebuah kerangka kerja aplikasi web mikro yang ditulis dalam bahasa pemrograman Python. Terdapat beberapa pilihan menu

yang dapat digunakan oleh pengguna untuk mengelola sistem. Menu pertama ialah menu utama (Home), Gambar 4.6 menampilkan halaman yang muncul ketika pengguna mengakses menu ini atau ketika pengguna pertama kali membuka aplikasi. Halaman ini menyajikan informasi seputar aplikasi.



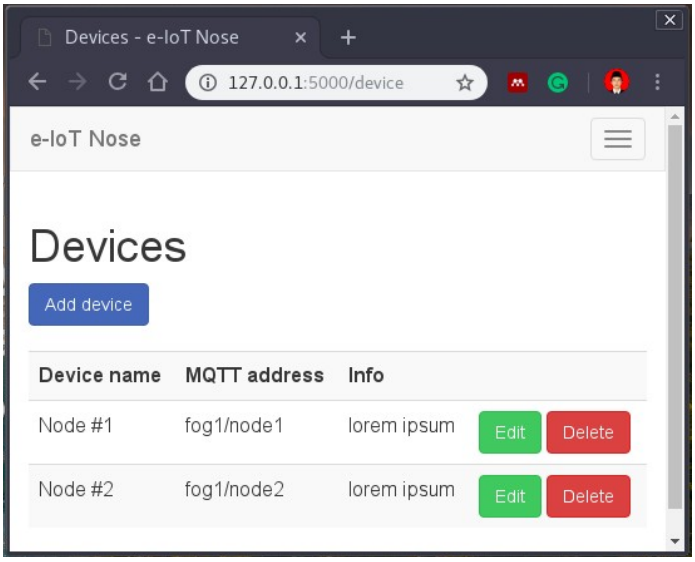
Gambar 4.6 Tampilan menu utama

Menu yang kedua ialah menu “Train”, menu ini digunakan untuk membagun model pembelajaran mesin. Model dibangun menggunakan algoritma KNN, yang sebelumnya keluar sebagai algoritma pembelajaran mesin dengan kinerja terbaik pada tahap optimisasi larik sensor. Ketika menu ini diakses, pengguna disuguhkan dengan formulir isian untuk mengunggah file set data dan mengatur jumlah parameter tetangga algoritma KNN, serta tombol untuk memulai pembangunan model (lihat Gambar 4.7).



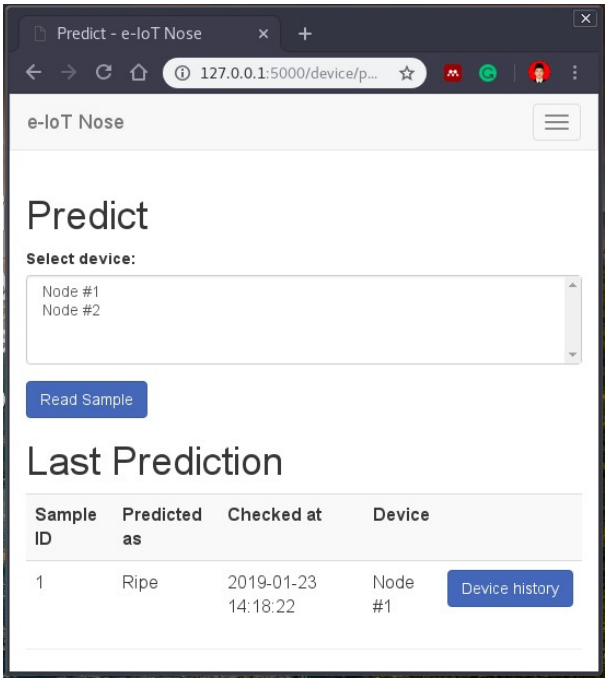
Gambar 4.7 Tampilan halaman menu “train”

Gambar 4.8 menampilkan halaman hasil eksekusi menu yang ketiga, yakni menu “Devices”. Pada halaman ini terdapat hal-hal terkait pengelolaan perangkat *e-nose*. Terdapat tombol untuk mendaftarkan *sensor node* (perangkat *e-nose*) baru ke dalam sistem; daftar tabel berisi informasi perangkat yang sudah didaftarkan; tombol untuk mengubah informasi perangkat; dan tombol untuk menghapus perangkat dari sistem.



Gambar 4.8 Tampilan halaman menu “Devices”

Keempat, menu “Predict” (Gambar 4.9). Ketika pengguna mengklik menu ini, muncul halaman yang menyediakan antarmuka untuk mengirim perintah pembacaan sampel ke berbagai *sensor node*, mengetahui prediksi sampel terakhirnya, dan tombol untuk melihat data historis lengkap hasil prediksi sampel sebuah *sensor node.*



Gambar 4.9 Tampilan halaman menu “predict”

4.2 Hasil Pengujian dan Analisis

4.2.1. Pengujian Fungsionalitas

Pada pengujian ini, sistem usulan diverifikasi apakah berfungsi sesuai harapan. Pengujian difokuskan pada pola masukan dan keluaran yang dihasilkan sistem melalui antarmuka. Terdapat 9 skenario uji coba yang dilakukan, diantaranya: (1) Melatih algoritma pembelajaran mesin; (2) Melihat perangkat e-nose yang sudah terdaftar; (3) Mendaftarkan perangkat e-nose baru; (4) Mengubah data perangkat e-nose; (5) Menghapus data perangkat e-nose; (6) Melihat prediksi sampel terakhir setiap perangkat e-nose; (7) Mengirimkan

perintah pembacaan sampel; (8) Melihat historis prediksi sampel perangkat e-nose; (9) Mengatur ulang data prediksi sampel.

Pada skenario yang pertama, uji coba yang dilakukan ialah melatih algoritma pembelajaran mesin. Proses ini, menghasilkan sebuah model yang nantinya akan digunakan sistem sebagai basis untuk melakukan prediksi sampel. Langkah dan hasil uji coba dari skenario ini dapat dilihat pada Tabel 4.7.

Tabel 4.7 Melatih algoritma pembelajaran mesin

Prasyarat	• Sudah *login* ke sistem • Menyiapkan *file dataset* dengan ekstensi .csv
Langkah Uji Coba	1. Membuka halaman "Train" 2. Mengunggah *file dataset* dan mengisi jumlah parameter tetangga KNN di formulir isian 3. Menekan tombol "Train"
Keluaran yang Diharapkan	Sistem menampilkan notifikasi pembangunan model telah selesai dan informasi besaran akurasi yang diperoleh
Keluaran yang Didapatkan	Sistem menampilkan notifikasi pembangunan model telah selesai dan informasi besaran akurasi yang diperoleh
Hasil Uji Coba	Berhasil

Pada skenario yang kedua, dilakukan uji coba untuk melihat *sensor node* yang terdaftar di basis data sistem. Langkah dan hasil uji coba dari skenario ini dapat dilihat pada Tabel 4.8.

Tabel 4.8 Melihat *sensor node* yang sudah terdaftar

Prasyarat	• Sudah *login* ke sistem
Langkah Uji Coba	1. Membuka halaman "Devices"
Keluaran yang Diharapkan	Sistem menampilkan seluruh data *sensor node* yang terdaftar
Keluaran yang Didapatkan	Sistem menampilkan seluruh data *sensor node* yang terdaftar
Hasil Uji Coba	Berhasil

Pada skenario yang ketiga, uji coba yang dilakukan ialah menambahkan *sensor node* baru kedalam basis data sistem. Langkah dan hasil uji coba dari skenario ini dapat dilihat pada Tabel 4.9.

Tabel 4.9 Mendaftarkan *sensor node* baru

Prasyarat	• Sudah *login* ke sistem
Langkah Uji Coba	1. Membuka halaman "Devices" 2. Menekan tombol "Add device" 3. Mengisi data nama, alamat MQTT, dan informasi singkat *sensor node* 4. Menekan tombol "Submit"
Keluaran yang Diharapkan	Data *sensor node* berhasil didaftarkan, muncul notifikasi pemberitahuan dan tampil dalam tabel *sensor node*
Keluaran yang Didapatkan	Data *sensor node* berhasil didaftarkan, muncul notifikasi pemberitahuan dan tampil dalam tabel *sensor node*
Hasil Uji Coba	Berhasil

Pada skenario yang keempat, dilakukan uji coba untuk mengubah data *sensor node* yang terdaftar. Langkah dan hasil uji coba dari skenario ini dapat dilihat pada Tabel 4.10.

Tabel 4.10 Mengubah data *sensor node*

Prasyarat	• Sudah *login* ke sistem
Langkah Uji Coba	1. Membuka halaman "Devices" 2. Memilih *sensor node* yang ingin diubah 3. Menekan tombol "Edit" di sebelah daftar *sensor node* 4. Melakukan perubahan sebagian data 5. Menekan tombol "Submit"
Keluaran yang Diharapkan	Sistem mencatat perubahan data *sensor node* dan muncul notifikasi pemberitahuan sukses
Keluaran yang Didapatkan	Sistem mencatat perubahan data *sensor node* dan muncul notifikasi pemberitahuan sukses
Hasil Uji Coba	Berhasil

Pada skenario yang kelima, uji coba yang dilakukan ialah menghapus data *sensor node* dari basis data sistem. Langkah dan hasil uji coba dari skenario ini dapat dilihat pada Tabel 4.11.

Tabel 4.11 Menghapus data *sensor node*

Prasyarat	• Sudah *login* ke sistem
Langkah Uji Coba	1. Membuka halaman "Devices" 2. Memilih *sensor node* yang ingin diubah 3. Menekan tombol "Delete" di sebelah daftar *sensor node*
Keluaran yang Diharapkan	Data *sensor node* hilang dari tabel dan muncul notifikasi pemberitahuan data sukses dihapus
Keluaran yang Didapatkan	Data *sensor node* hilang dari tabel dan muncul notifikasi pemberitahuan data sukses dihapus
Hasil Uji Coba	Berhasil

Pada skenario yang keenam, dilakukan uji coba untuk melihat prediksi sampel terakhir dari *sensor node*. Langkah dan hasil uji coba dari skenario ini dapat dilihat pada Tabel 4.12.

Tabel 4.12 Melihat prediksi sampel terakhir perangkat *e-nose*

Prasyarat	• Sudah *login* ke sistem
Langkah Uji Coba	1. Membuka halaman "Predict"
Keluaran yang Diharapkan	Sistem menampilkan seluruh data *sensor node* dan hasil prediksi sampel yang terakhir
Keluaran yang Didapatkan	Sistem menampilkan seluruh data *sensor node* dan hasil prediksi sampel yang terakhir
Hasil Uji Coba	Berhasil

Pada skenario yang ketujuh, uji coba yang dilakukan ialah mengirimkan perintah pembacaan sampel ke *sensor node*. Langkah dan hasil uji coba dari skenario ini dapat dilihat pada Tabel 4.13.

Tabel 4.13 Mengirimkan perintah pembacaan sampel

Prasyarat	• Sudah *login* ke sistem
Langkah Uji Coba	1. Membuka halaman "Predict"

	2. Memilih *sensor node* mana saja yang ingin dilakukan pembacaan sampel 3. Menekan tombol “Read sample”
Keluaran yang Diharapkan	Sistem menampilkan data hasil prediksi sampel *sensor node* baru
Keluaran yang Didapatkan	Sistem menampilkan data hasil prediksi sampel *sensor node* baru
Hasil Uji Coba	Berhasil

Pada skenario yang kedelapan, dilakukan uji coba untuk melihat data historis prediksi sampel *sensor node*. Langkah dan hasil uji coba dari skenario ini dapat dilihat pada Tabel 4.14.

Tabel 4.14 melihat historis prediksi sampel perangkat *e-nose*

Prasyarat	• Sudah *login* ke sistem
Langkah Uji Coba	1. Membuka halaman “Predict” 2. Memilih *sensor node* mana yang ingin dilihat data historisnya 3. Menekan tombol “Device history”
Keluaran yang Diharapkan	Sistem menampilkan data historis lengkap hasil prediksi sampel *sensor node* terpilih
Keluaran yang Didapatkan	Sistem menampilkan data historis lengkap hasil prediksi sampel *sensor node* terpilih
Hasil Uji Coba	Berhasil

Pada skenario yang terakhir, uji coba yang dilakukan ialah mengatur ulang data prediksi sampel *sensor node*. Langkah dan hasil uji coba dari skenario ini dapat dilihat pada Tabel 4.15.

Tabel 4.15 Mengatur ulang data prediksi sampel

Prasyarat	• Sudah *login* ke sistem
Langkah Uji Coba	1. Membuka halaman “Predict” 2. Memilih *sensor node* yang ingin dihapus data prediksinya 3. Menekan tombol “Device history” 4. Menekan tombol “Reset”
Keluaran yang	Sistem menghapus seluruh data historis prediksi

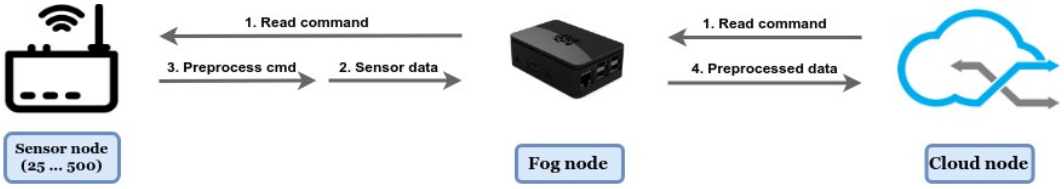
Diharapkan	sampel *sensor node* yang dipilih
Keluaran yang Didapatkan	Sistem menghapus seluruh data historis prediksi sampel *sensor node* yang dipilih
Hasil Uji Coba	Berhasil

Menggunakan model pembelajaran mesin yang dibangun dari hasil optimisasi larik sensor, sistem mampu memprediksi 50 sampel pisang kepok baru yang dikirimkan *sensor node* melalui *fog node,* dengan tingkat akurasi sebesar 78%. Berdasarkan hasil uji coba yang telah dilakukan, semua skenario pengujian yang ditetapkan berhasil dilakukan oleh sistem, sehingga dapat dikatakan bahwa sistem bekerja sesuai dengan yang diharapkan.

4.2.2. Pengujian Skalabilitas

Terdiri dari dua tahap pengujian. Tahap pertama melakukan pengujian menggunakan 8 menit data dengan algoritma KNN, tahap kedua melakukan pengujian menggunakan dengan 6 menit data dengan algoritma RF. Hal ini dikarenakan pada tahap optimisasi larik sensor, kedua algoritma memperoleh hasil yang tidak jauh berbeda, hanya 1%. KNN memperoleh akurasi sebesar 80% ketika menggunakan 8 menit data, sementara RF memperoleh akurasi 79% ketika menggunakan 6 menit data. Dengan membandingkan jumlah data dan algoritma yang digunakan, ada tidaknya pengaruh yang signifikan terhadap skalabilitas sistem dapat diketahui.

Kedua tahap pengujian yang telah dijelaskan, terdiri dari 4 skenario pengujian dengan 11 kali penambahan beban *sensor node*. Adapun skenario yang dimaksud ialah: (1) Mengirimkan perintah pembacaan sampel ke *sensor node;* (2) Mengirimkan data pembacaan sensor ke *fog node*; (3) Mengirimkan perintah ke *fog node* untuk melakukan prapemrosesan data; dan (4) Mengirimkan data hasil prapemrosesan ke *cloud node* untuk dilakukan identifikasi. Sementara 11 kali penambahan beban *sensor node* dimulai dari 25, 50, 100, seterusnya kelipatan 50 hingga mencapai 500*,* Skema pengujian dapat dilihat pada Gambar 4.10 berikut.



Gambar 4.10 Pengujian skalabilitas

Adapun data yang digunakan untuk pengujian ini dapat dilihat pada gambar berikut ini. Data yang digunakan merupakan data asli yang diperoleh dari *sensor node* dan *fog node*. Gambar 4.11 Menampilkan salah satu data pembacaan sampel yang dikirim dari *sensor node,* sedangkan gambar 4.12 menampilkan data hasil prapemrosesan yang dilakukan *fog node*:


```
[{"sample_id":xx-001,"status":read,"MQ2_LPG":0.03,"MQ2_H2":3.12,"MQ2_PROPANE":5.85,"MQ3_CO":0.03,"MQ3_LPG":3.44,"MQ6_CH4":
,"MQ6_H2":11.69,"MQ6_LPG":3.39,"MQ7_CO":0.8,"MQ7_H2":0.69,"MQ7_LPG":4.92}]
```


Gambar 4.11 Data pembacaan yang dikirimkan *sensor node*


```
[{"MQ2_LPG_skew":0.5066223154410646,"MQ2_H2_skew":0.5068572111514019,"MQ2_PROPANE_skew":0.5096378462491764,"MQ3_CO_skew":
0.27395187319321646,"MQ3_LPG_skew":0.9999999999999988,"MQ6_CH4_std":0.4260948174879171,"MQ6_CH4_skew":0.35119223688086865,
"MQ6_H2_std":0.4377232908404123,"MQ6_H2_skew":0.4034751315781631,"MQ6_LPG_skew":0.3475999645521177,"MQ7_CO_skew":
0.8210939784788753,"MQ7_H2_skew":0.742183747163169,"MQ7_LPG_mean":0.0,"MQ7_LPG_std":0.0,"MQ7_LPG_var":0.0}]
```


Gambar 4.12 Data praproses yang dikirimkan *fog node*

Semua pengujian diatas dilakukan menggunakan kakas bantu simulasi Apache Jmeter. Kakas bantu ini merupakan aplikasi yang sudah dikenal luas dan sering digunakan untuk melakukan tes beban pada aplikasi web terdistribusi. JMeter dapat menghasilkan sejumlah klien virtual dan permintaan (*request*) secara bersamaan, terlepas dari *platform* yang digunakan. Memang, pada awalnya JMeter dirancang untuk menguji fungsionalitas dan kinerja aplikasi web, tetapi sekarang penggunaannya diperluas ke berbagai fungsi pengujian lainnya. Seperti yang dilakukan oleh (Ali et al., 2017), dimana ia menggunakan JMeter untuk menguji tingkat skalabilitas sistem IoT yang diusulkannya.

1. Tahap pertama: 8 menit data dengan algoritma KNN

Pada skenario yang pertama, JMeter mensimulasikan perintah pembacaan sampel ke seluruh *sensor node* yang terdaftar di sistem layaknya menggunakan

antarmuka sistem. Pengiriman dilakukan secara serentak dalam satu waktu. Tabel 4.16 menampilkan hasil laporan JMeter terkait skenario pengujian ini.

Tabel 4.16 Mengirimkan perintah pembacaan sampel

Jumlah sampel	**Terkirim**	**Rata-rata waktu respon**	**Throughput**
25	25	0.16	24.98
50	50	1.18	49.95
100	100	0.22	99.9
150	150	0.53	149.85
200	200	0.31	199.6
250	250	0.13	249.75
300	300	0.56	299.7
350	350	0.11	349.65
400	400	0.35	398.8
450	450	0.2	449.55
500	500	0.25	498.01

Pada skenario yang selanjutnya, JMeter mensimulasikan sejumlah *sensor node* yang mengirim sejumlah data hasil pembacaan sensor ke *fog node.* Data dikirim setiap 2 detik selama 8 menit. Jadi, total sampel yang dikirim satu *sensor node* ialah 240 data. Tabel 4.17 menampilkan hasil laporan JMeter untuk skenario pengujian ini.

Tabel 4.17 Mengirimkan data pembacaan sensor

Jumlah sampel	**Terkirim**	**Rata-rata waktu respon**	**Throughput**
6000	6000	0.19	12.53
12000	12000	0.16	25
24000	24000	0.11	50.1
36000	36000	0.1	75.16
48000	48000	0.09	100.21
60000	60000	0.09	125.26
72000	72000	0.08	150.31

84000	84000	0.09	175.36
96000	96000	0.09	200.42
108000	108000	0.09	225.47
120000	120000	0.08	250

Pada skenario yang ketiga, JMeter mensimulasikan *sensor node* yang telah selesai melakukan pembacaan sensor, dimana ia akan mengirimkan perintah ke *fog node* untuk mem-praproses serangkaian data yang telah dikirimkan. Tabel 4.18 menampilkan hasil laporan JMeter terkait skenario pengujian ini.

Tabel 4.18 Mengirimkan perintah prapemrosesan data

Jumlah sampel	**Terkirim**	**Rata-rata waktu respon**	**Throughput**
25	25	0.32	24.98
50	50	0.3	49.95
100	100	0.52	99.9
150	150	1.65	149.85
200	200	0.1	199.8
250	250	0.2	249
300	300	0.39	299.7
350	350	0.11	348.95
400	400	0.11	397.22
450	450	0.26	449.55
500	500	0.18	498.5

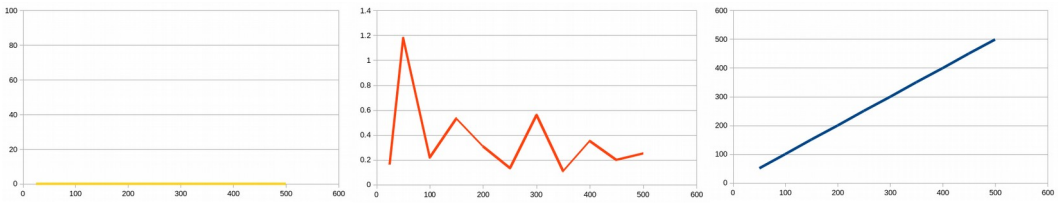
Pada skenario yang terakhir, JMeter mensimulasikan *fog node* yang mengirimkan data hasil prapemrosesan ke *cloud node.* Tabel 4.19 menampilkan hasil laporan JMeter untuk skenario pengujian ini.

Tabel 4.19 mengirimkan data hasil prapemrosesan ke *cloud node*

Jumlah sampel	**Terkirim**	**Rata-rata waktu respon**	**Throughput**
25	25	5.6	24.98
50	50	6.42	49.95

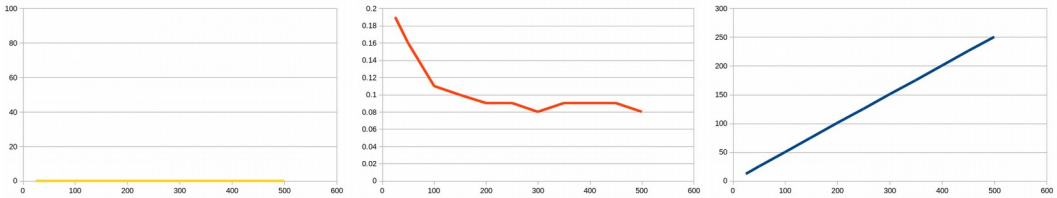
100	100	7.8	99.8
150	150	3.37	149.55
200	200	3.85	199.6
250	250	2.12	249.25
300	300	2.2	298.21
350	350	1.43	318.76
400	400	0.58	398.41
450	450	8.01	224.89
500	500	0.58	249.75

Tabel hasil laporan JMeter menunjukkan beberapa hal. Pertama, pada keempat skenario, sampel yang diterima jumlahnya sesuai dengan yang dikirimkan, tidak terjadi *package lost*. Kedua, waktu respon rata-rata pengiriman data ke sejumlah sensor node bervariasi, tidak dipengaruhi oleh besar kecilnya jumlah data yang dikirim (Gambar 4.13b, 4.14b, 4.14b, 4.14b). Ketiga, nilai *throughput* pada semua skenario linear (Gambar 4.13c, 4.14c, 4.15c) seiring bertambahnya jumlah data yang harus dikirim, kecuali skenario yang ke-4 (Gambar 4.16c), dimana terjadi penurunan nilai *throughput* ketika mulai mensimulasikan 350 *sensor node.*



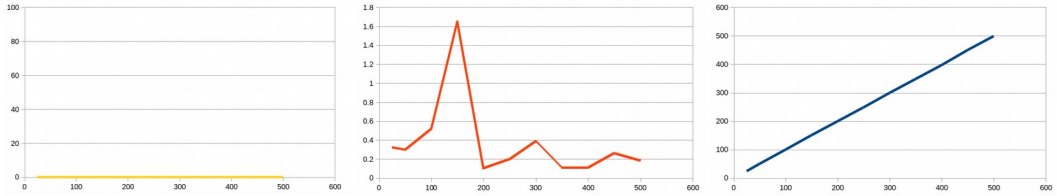
(a) (b) (c)

Gambar 4.13 Hasil pengujian skalabilitas skenario m*engirimkan perintah pembacaan sampel ke sensor node. (a) jumlah simulasi sensor node vs package lost. (b) jumlah simulasi sensor node vs waktu respon. (c) jumlah simulasi sensor node vs throughput.*



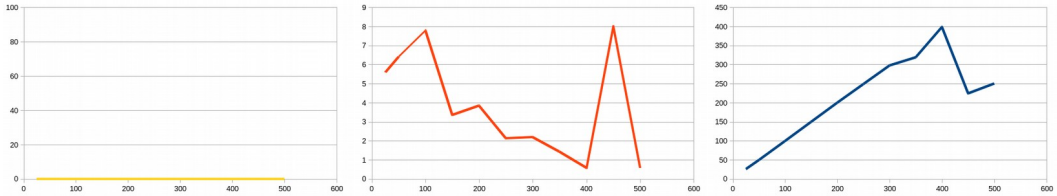
(a) (b) (c)

Gambar 4.14 Hasil pengujian skalabilitas skenario *mengirimkan data pembacaan sensor ke fog node. (a) jumlah simulasi sensor node vs package lost. (b) jumlah simulasi sensor node vs waktu respon. (c) jumlah simulasi sensor node vs throughput.*



(a) (b) (c)

Gambar 4.15 Hasil pengujian skalabilitas skenario *mengirimkan perintah ke fog node untuk melakukan prapemrosesan data. (a) jumlah simulasi sensor node vs package lost. (b) jumlah simulasi sensor node vs waktu respon. (c) jumlah simulasi sensor node vs throughput.*



(a) (b) (c)

Gambar 4.16 Hasil pengujian skalabilitas skenario *mengirimkan data hasil prapemrosesan ke cloud node untuk dilakukan identifikasi*. *(a) jumlah simulasi sensor node vs package lost. (b) jumlah simulasi sensor node vs waktu respon. (c) jumlah simulasi sensor node vs throughput.*

Pada pengamatan hasil *query* basis data, dapat diketahui bahwa tidak semua data yang dikirimkan berhasil diproses. Ketika mensimulasikan 25 sampai 300 *sensor node* untuk mengirimkan data hasil pembacaan sensor ke *fog node,* data tersimpan seluruhnya (Gambar 4.17 – Gambar 4.23). Namun, ketika mensimulasikan 350 *sensor node* untuk mengirimkan data hasil pembacaan sensor ke *fog node*, data yang tersimpan mulai tidak lengkap. Data yang seharusnya berjumlah 84.000 ((240 [jumlah data selama pembacaan sampel] x 350 [jumlah sensor node]), di basis data hanya ada 71.448 (Gambar 4.24). Begitupun kasusnya

ketika mensimulasikan pengiriman data 400, 450, dan 500 *sensor node* (Gambar 4.25 – Gambar 4.27). Hal yang berbeda ditunjukkan ketika mensimulasikan pengiriman data hasil prapemrosesan. Dimana, seluruh data yang dikirimkan *fog node* berhasil diproses oleh *cloud node* (Gambar 4.28 – Gambar 4.37).

```
> use kepok_25
switched to db kepok_25
> db.sensors_data.aggregate({$project: {value: 1}}, {$unwind: "$value"}, {$group
: {_id: "result", count: {$sum: 1}}});
{ "result" : [ { "_id" : "result", "count" : 6000 } ], "ok" : 1 }
> 
```

Gambar 4.17 Hasil *query* jumlah data yang diterima dari 25 *sensor node*

```
> use kepok_50
switched to db kepok_50
> db.sensors_data.aggregate({$project: {value: 1}}, {$unwind: "$value"}, {$group
: {_id: "result", count: {$sum: 1}}});
{ "result" : [ { "_id" : "result", "count" : 12000 } ], "ok" : 1 }
> 
```

Gambar 4.18 hasil *query* jumlah data yang diterima dari 50 *sensor node*

```
> use kepok_100
switched to db kepok_100
> db.sensors_data.aggregate({$project: {value: 1}}, {$unwind: "$value"}, {$group
: {_id: "result", count: {$sum: 1}}});
{ "result" : [ { "_id" : "result", "count" : 24000 } ], "ok" : 1 }
> 
```

Gambar 4.19 Hasil *query* jumlah data yang diterima dari 100 *sensor node*

```
> use kepok_150
switched to db kepok_150
> db.sensors_data.aggregate({$project: {value: 1}}, {$unwind: "$value"}, {$group
: {_id: "result", count: {$sum: 1}}});
{ "result" : [ { "_id" : "result", "count" : 36000 } ], "ok" : 1 }
> 
```

Gambar 4.20 Hasil *query* jumlah data yang diterima dari 150 *sensor node*

```
> use kepok_200
switched to db kepok_200
> db.sensors_data.aggregate({$project: {value: 1}}, {$unwind: "$value"}, {$group
: {_id: "result", count: {$sum: 1}}});
{ "result" : [ { "_id" : "result", "count" : 48000 } ], "ok" : 1 }
> 
```

Gambar 4.21 Hasil *query* jumlah data yang diterima dari 200 *sensor node*

```
> use kepok_250
switched to db kepok_250
> db.sensors_data.aggregate({$project: {value: 1}}, {$unwind: "$value"}, {$group
: {_id: "result", count: {$sum: 1}}});
{ "result" : [ { "_id" : "result", "count" : 60000 } ], "ok" : 1 }
>
```

Gambar 4.22 Hasil *query* jumlah data yang diterima dari 250 *sensor node*

```
> use kepok_300
switched to db kepok_300
> db.sensors_data.aggregate({$project: {value: 1}}, {$unwind: "$value"}, {$group
: {_id: "result", count: {$sum: 1}}});
{ "result" : [ { "_id" : "result", "count" : 72000 } ], "ok" : 1 }
>
```

Gambar 4.23 Hasil *query* jumlah data yang diterima dari 300 *sensor node*

```
> use kepok_350
switched to db kepok_350
> db.sensors_data.aggregate({$project: {value: 1}}, {$unwind: "$value"}, {$group
: {_id: "result", count: {$sum: 1}}});
{ "result" : [ { "_id" : "result", "count" : 71448 } ], "ok" : 1 }
>
```

Gambar 4.24 Hasil *query* jumlah data yang diterima dari 350 sensor node

```
> use kepok_400
switched to db kepok_400
> db.sensors_data.aggregate({$project: {value: 1}}, {$unwind: "$value"}, {$group
: {_id: "result", count: {$sum: 1}}});
{ "result" : [ { "_id" : "result", "count" : 68591 } ], "ok" : 1 }
>
```

Gambar 4.25 Hasil *query* jumlah data yang diterima dari 400 *sensor node*

```
> use kepok_450
switched to db kepok_450
> db.sensors_data.aggregate({$project: {value: 1}}, {$unwind: "$value"}, {$group
: {_id: "result", count: {$sum: 1}}});
{ "result" : [ { "_id" : "result", "count" : 76855 } ], "ok" : 1 }
>
```

Gambar 4.26 Hasil *query* jumlah data yang diterima dari 450 sensor node

```
> use kepok_500
switched to db kepok_500
> db.sensors_data.aggregate({$project: {value: 1}}, {$unwind: "$value"}, {$group
: {_id: "result", count: {$sum: 1}}});
{ "result" : [ { "_id" : "result", "count" : 77899 } ], "ok" : 1 }
>
```

Gambar 4.27 Hasil *query* jumlah data yang diterima dari 500 *sensor node*

```
MariaDB [enose]> SELECT COUNT(*) FROM sample;
+----------+
| COUNT(*) |
+----------+
|       25 |
+----------+
1 row in set (0.00 sec)
```

Gambar 4.28 Jumlah data prediksi sampel dari 25 data *sensor node*

```
MariaDB [enose]> SELECT COUNT(*) FROM sample;
+----------+
| COUNT(*) |
+----------+
|       50 |
+----------+
1 row in set (0.00 sec)
```

Gambar 4.29 Jumlah data prediksi sampel dari 50 data *sensor node*

```
MariaDB [enose]> SELECT COUNT(*) FROM sample;
+----------+
| COUNT(*) |
+----------+
|      100 |
+----------+
1 row in set (0.00 sec)
```

Gambar 4.30 Jumlah data prediksi sampel dari 100 data *sensor node*

```
MariaDB [enose]> SELECT COUNT(*) FROM sample;
+----------+
| COUNT(*) |
+----------+
|      150 |
+----------+
1 row in set (0.01 sec)
```

Gambar 4.31 Jumlah data prediksi sampel dari 150 data *sensor node*

```
MariaDB [enose]> SELECT COUNT(*) FROM sample;
+----------+
| COUNT(*) |
+----------+
|      200 |
+----------+
1 row in set (0.00 sec)
```

Gambar 4.32 Jumlah data prediksi sampel dari 200 data *sensor node*

```
MariaDB [enose]> SELECT COUNT(*) FROM sample;
+----------+
| COUNT(*) |
+----------+
|      250 |
+----------+
1 row in set (0.00 sec)
```

Gambar 4.33 Jumlah data prediksi sampel dari 250 data *sensor node*

```
MariaDB [enose]> SELECT COUNT(*) FROM sample;
+----------+
| COUNT(*) |
+----------+
|      300 |
+----------+
1 row in set (0.00 sec)
```

Gambar 4.34 Jumlah data prediksi sampel dari 300 data *sensor node*

```
MariaDB [enose]> SELECT COUNT(*) FROM sample;
+----------+
| COUNT(*) |
+----------+
|      350 |
+----------+
1 row in set (0.00 sec)
```

Gambar 4.35 Jumlah data prediksi sampel dari 350 data *sensor node*

```
MariaDB [enose]> SELECT COUNT(*) FROM sample;
+----------+
| COUNT(*) |
+----------+
|      400 |
+----------+
1 row in set (0.01 sec)
```

Gambar 4.36 Jumlah data prediksi sampel dari 400 data *sensor node*

```
MariaDB [enose]> SELECT COUNT(*) FROM sample;
+----------+
| COUNT(*) |
+----------+
|      450 |
+----------+
1 row in set (0.00 sec)
```

Gambar 4.37 Jumlah data prediksi sampel dari 450 data *sensor node*

2. Tahap kedua: 6 menit data dengan algoritma RF

Pada skenario yang pertama, JMeter mensimulasikan perintah pembacaan sampel ke seluruh *sensor node* yang terdaftar di sistem layaknya menggunakan antarmuka sistem. Pengiriman dilakukan secara serentak dalam satu waktu. Tabel 4.20 menampilkan hasil laporan JMeter terkait skenario pengujian ini.

Tabel 4.20 Mengirimkan perintah pembacaan sampel

Jumlah sampel	**Terkirim**	**Rata-rata waktu respon**	**Throughput**
25	25	0.44	25
50	50	0.3	49.95
100	100	0.41	99.9
150	150	0.44	149.85
200	200	1.36	199.8
250	250	1.69	249.75
300	300	0.47	299.7
350	350	0.2	349.65
400	400	0.46	199.8
450	450	2.86	224.89
500	500	0.38	249.88

Pada skenario yang selanjutnya, JMeter mensimulasikan sejumlah *sensor node* yang mengirim sejumlah data hasil pembacaan sensor ke *fog node.* Data dikirim setiap 2 detik selama 6 menit. Jadi, total sampel yang dikirim satu *sensor node* ialah 180 data. Tabel 4.21 menampilkan hasil laporan JMeter untuk skenario pengujian ini.

Tabel 4.21 Mengirimkan data pembacaan sensor

Jumlah sampel	Terkirim	Rata-rata waktu respon	Throughput
4500	4500	0.11	12.53
9000	9000	0.09	25.07
18000	18000	0.08	50.14
27000	27000	0.07	75.21
36000	36000	0.07	100.28
45000	45000	0.07	125.35
54000	54000	0.07	150.42
63000	63000	0.07	175.49
72000	72000	0.07	200.56
81000	81000	0.07	225.63
90000	90000	0.07	250

Pada skenario yang ketiga, JMeter mensimulasikan *sensor node* yang telah selesai melakukan pembacaan sensor, dimana ia akan mengirimkan perintah ke *fog node* untuk mem-praproses serangkaian data yang telah dikirimkan. Tabel 4.22 menampilkan hasil laporan JMeter terkait skenario pengujian ini.

Tabel 4.22 Mengirimkan perintah prapemrosesan data

Jumlah sampel	Terkirim	Rata-rata waktu respon	Throughput
25	25	0.28	24.9
50	50	0.36	49.95
100	100	0.34	100
150	150	0.25	149.85
200	200	0.31	199.8
250	250	0.2	248
300	300	0.1	297.91
350	350	0.07	349.65
400	400	0.1	199.9
450	450	0.11	442.91

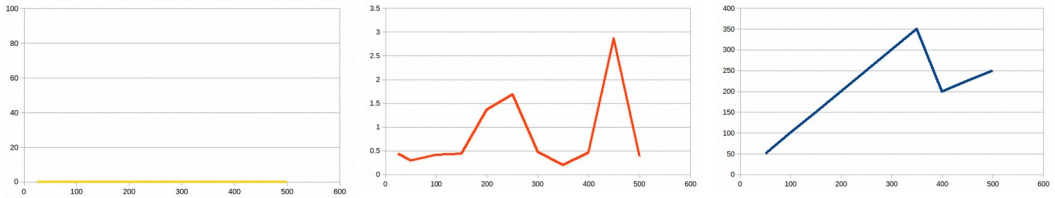
500	500	0.07	496.52

Pada skenario yang terakhir, JMeter mensimulasikan *fog node* yang mengirimkan data hasil prapemrosesan ke *cloud node.* Tabel 4.23 menampilkan hasil laporan JMeter untuk skenario pengujian ini.

Tabel 4.23 Mengirimkan data hasil prapemrosesan ke *cloud node*

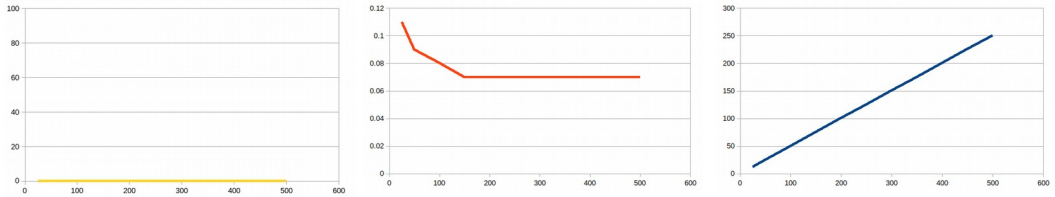
Jumlah sampel	**Terkirim**	**Rata-rata waktu respon**	**Throughput**
25	25	6.16	24.98
50	50	6.78	49.95
100	100	3.58	99.8
150	150	7.21	149.7
200	200	5.05	199.2
250	250	3.65	248.26
300	300	3.13	295.86
350	350	3.77	346.88
400	400	1.58	199.8
450	450	3.18	224.78
500	500	2.49	249.63

Tabel hasil laporan JMeter menunjukkan beberapa hal. Pertama, pada keempat skenario, sampel yang diterima jumlahnya sesuai dengan yang dikirimkan, tidak terjadi *package lost*. Kedua, waktu respon rata-rata pengiriman data ke sejumlah sensor node bervariasi, tidak dipengaruhi oleh besar kecilnya jumlah data yang dikirim (Gambar 4.38b, 4.39b, 4.40b, 4.41b). Ketiga, nilai *throughput* pada skenario kedua linear seiring bertambahnya jumlah data yang harus dikirim (Gambar 4.39c), sementara pada skenario yang lain terjadi penurunan nilai *throughput* ketika mulai mensimulasikan 400 *sensor node* (Gambar 4.38c, 4.40c, 4.41c).



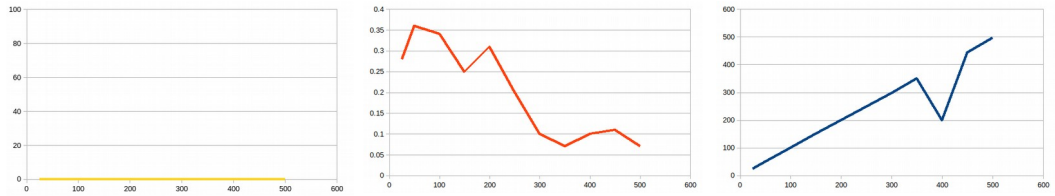
(a) (b) (c)

Gambar 4.38 Hasil pengujian skalabilitas skenario m*engirimkan perintah pembacaan sampel ke sensor node. (a) jumlah simulasi sensor node vs package lost. (b) jumlah simulasi sensor node vs waktu respon. (c) jumlah simulasi sensor node vs throughput.*



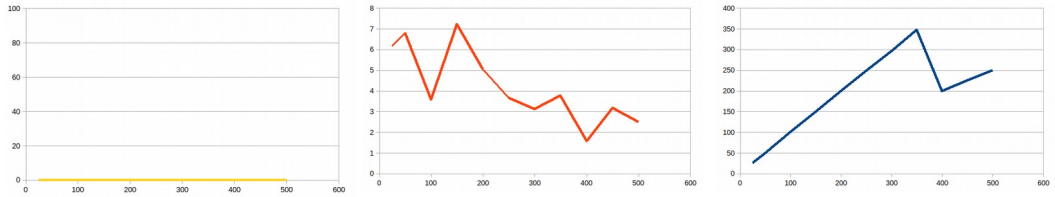
(a) (b) (c)

Gambar 4.39 Hasil pengujian skalabilitas skenario *mengirimkan data pembacaan sensor ke fog node. (a) jumlah simulasi sensor node vs package lost. (b) jumlah simulasi sensor node vs waktu respon. (c) jumlah simulasi sensor node vs throughput.*



(a) (b) (c)

Gambar 4.40 Hasil pengujian skalabilitas skenario *mengirimkan perintah ke fog node untuk melakukan prapemrosesan data. (a) jumlah simulasi sensor node vs package lost. (b) jumlah simulasi sensor node vs waktu respon. (c) jumlah simulasi sensor node vs throughput.*



(a) (b) (c)

Gambar 4.41 Hasil pengujian skalabilitas skenario *mengirimkan data hasil prapemrosesan ke cloud node untuk dilakukan identifikasi. (a) jumlah simulasi sensor node vs package lost. (b) jumlah simulasi sensor node vs waktu respon. (c) jumlah simulasi sensor node vs throughput.*

Berdasarkan pengamatan hasil *query* basis data, dapat diketahui bahwa semua data yang dikirimkan berhasil diproses. Ketika mensimulasikan 25 sampai 500 *sensor node* untuk mengirimkan data hasil pembacaan sensor ke *fog node,* data tersimpan seluruhnya (Gambar 4.42 – Gambar 4.52). Begitupun kasusnya ketika mensimulasikan pengiriman data hasil prapemrosesan, seluruh data yang dikirimkan *fog node* berhasil diproses oleh *cloud node* (Gambar 4.53 – Gambar 4.62).

```
> use kepok_25
switched to db kepok_25
> db.sensors_data.aggregate({$project: {value: 1}}, {$unwind: "$value"}, {$group
: {_id: "result", count: {$sum: 1}}});
{ "result" : [ { "_id" : "result", "count" : 4500 } ], "ok" : 1 }
>
```

Gambar 4.42 Hasil *query* jumlah data yang diterima dari 25 *sensor node*

```
> use kepok_50
switched to db kepok_50
> db.sensors_data.aggregate({$project: {value: 1}}, {$unwind: "$value"}, {$group
: {_id: "result", count: {$sum: 1}}});
{ "result" : [ { "_id" : "result", "count" : 9000 } ], "ok" : 1 }
>
```

Gambar 4.43 Hasil *query* jumlah data yang diterima dari 50 *sensor node*

```
> use kepok_100
switched to db kepok_100
> db.sensors_data.aggregate({$project: {value: 1}}, {$unwind: "$value"}, {$group
: {_id: "result", count: {$sum: 1}}});
{ "result" : [ { "_id" : "result", "count" : 18000 } ], "ok" : 1 }
>
```

Gambar 4.44 Hasil *query* jumlah data yang diterima dari 100 *sensor node*

```
> use kepok_150
switched to db kepok_150
> db.sensors_data.aggregate({$project: {value: 1}}, {$unwind: "$value"}, {$group
: {_id: "result", count: {$sum: 1}}});
{ "result" : [ { "_id" : "result", "count" : 27000 } ], "ok" : 1 }
>
```

Gambar 4.45 Hasil *query* jumlah data yang diterima dari 150 *sensor node*

```
> use kepok_200
switched to db kepok_200
> db.sensors_data.aggregate({$project: {value: 1}}, {$unwind: "$value"}, {$group
: {_id: "result", count: {$sum: 1}}});
{ "result" : [ { "_id" : "result", "count" : 36000 } ], "ok" : 1 }
>
```

Gambar 4.46 Hasil *query* jumlah data yang diterima dari 200 *sensor node*

```
> use kepok_250
switched to db kepok_250
> db.sensors_data.aggregate({$project: {value: 1}}, {$unwind: "$value"}, {$group
: {_id: "result", count: {$sum: 1}}});
{ "result" : [ { "_id" : "result", "count" : 45000 } ], "ok" : 1 }
>
```

Gambar 4.47 Hasil *query* jumlah data yang diterima dari 250 *sensor node*

```
> use kepok_300
switched to db kepok_300
> db.sensors_data.aggregate({$project: {value: 1}}, {$unwind: "$value"}, {$group
: {_id: "result", count: {$sum: 1}}});
{ "result" : [ { "_id" : "result", "count" : 54000 } ], "ok" : 1 }
>
```

Gambar 4.48 Hasil *query* jumlah data yang diterima dari 300 *sensor node*

```
> use kepok_350
switched to db kepok_350
> db.sensors_data.aggregate({$project: {value: 1}}, {$unwind: "$value"}, {$group
: {_id: "result", count: {$sum: 1}}});
{ "result" : [ { "_id" : "result", "count" : 63000 } ], "ok" : 1 }
>
```

Gambar 4.49 Hasil *query* jumlah data yang diterima dari 350 *sensor node*

```
> use kepok_400
switched to db kepok_400
> db.sensors_data.aggregate({$project: {value: 1}}, {$unwind: "$value"}, {$group
: {_id: "result", count: {$sum: 1}}});
{ "result" : [ { "_id" : "result", "count" : 72000 } ], "ok" : 1 }
>
```

Gambar 4.50 Hasil *query* jumlah data yang diterima dari 400 *sensor node*

```
> use kepok_450
switched to db kepok_450
> db.sensors_data.aggregate({$project: {value: 1}}, {$unwind: "$value"}, {$group
: {_id: "result", count: {$sum: 1}}});
{ "result" : [ { "_id" : "result", "count" : 81000 } ], "ok" : 1 }
>
```

Gambar 4.51 Hasil query jumlah data yang diterima dari 450 sensor node

```
> use kepok_500
switched to db kepok_500
> db.sensors_data.aggregate({$project: {value: 1}}, {$unwind: "$value"}, {$group
: {_id: "result", count: {$sum: 1}}});
{ "result" : [ { "_id" : "result", "count" : 90000 } ], "ok" : 1 }
>
```

Gambar 4.52 Hasil *query* jumlah data yang diterima dari 500 *sensor node*

```
MariaDB [enose]> SELECT COUNT(*) FROM sample;
+----------+
| COUNT(*) |
+----------+
|       50 |
+----------+
1 row in set (0.00 sec)
```

Gambar 4.53 Jumlah data prediksi sampel dari 50 data *sensor node*

```
MariaDB [enose]> SELECT COUNT(*) FROM sample;
+----------+
| COUNT(*) |
+----------+
|      100 |
+----------+
1 row in set (0.00 sec)
```

Gambar 4.54 Jumlah data prediksi sampel dari 100 data *sensor node*

```
MariaDB [enose]> SELECT COUNT(*) FROM sample;
+----------+
| COUNT(*) |
+----------+
|      150 |
+----------+
1 row in set (0.01 sec)
```

Gambar 4.55 Jumlah data prediksi sampel dari 150 data *sensor node*

```
MariaDB [enose]> SELECT COUNT(*) FROM sample;
+----------+
| COUNT(*) |
+----------+
|      200 |
+----------+
1 row in set (0.00 sec)
```

Gambar 4.56 Jumlah data prediksi sampel dari 200 data *sensor node*

```
MariaDB [enose]> SELECT COUNT(*) FROM sample;
+----------+
| COUNT(*) |
+----------+
|      250 |
+----------+
1 row in set (0.00 sec)
```

Gambar 4.57 Jumlah data prediksi sampel dari 250 data *sensor node*

```
MariaDB [enose]> SELECT COUNT(*) FROM sample;
+----------+
| COUNT(*) |
+----------+
|      300 |
+----------+
1 row in set (0.00 sec)
```

Gambar 4.58 Jumlah data prediksi sampel dari 300 data *sensor node*

```
MariaDB [enose]> SELECT COUNT(*) FROM sample;
+----------+
| COUNT(*) |
+----------+
|      350 |
+----------+
1 row in set (0.00 sec)
```

Gambar 4.59 Jumlah data prediksi sampel dari 350 data *sensor node*

```
MariaDB [enose]> SELECT COUNT(*) FROM sample;
+----------+
| COUNT(*) |
+----------+
|      400 |
+----------+
1 row in set (0.01 sec)
```

Gambar 4.60 Jumlah data prediksi sampel dari 400 data *sensor node*

```
MariaDB [enose]> SELECT COUNT(*) FROM sample;
+----------+
| COUNT(*) |
+----------+
|      450 |
+----------+
1 row in set (0.00 sec)
```

Gambar 4.61 Jumlah data prediksi sampel dari 450 data *sensor node*

```
MariaDB [enose]> SELECT COUNT(*) FROM sample;
+----------+
| COUNT(*) |
+----------+
|      500 |
+----------+
1 row in set (0.00 sec)
```

Gambar 4.62 Jumlah data prediksi sampel dari 500 data *sensor node*

Berdasarkan hasil uji coba skalabilitas, dapat disimpulan bahwa sistem dapat terskalakan dengan baik, namun tentunya terbatas. Sesuai hasil simulasi menggunakan kakas bantu JMeter, pengujian tahap pertama menunjukkan bahwa sistem mulai mengalami degradasi performa ketika lebih dari 300 *sensor node* terhubung ke sistem. Hal ini dapat diketahui dari nilai *throughput* hasil laporan JMeter dan dari jumlah data pembacaan sensor yang tidak tersimpan secara utuh di basis data. Sementara itu, pada pengujian tahap kedua, sistem mulai tergredasi performanya ketika menghubungkan 400 *sensor node,* namun seluruh data yang dikirimkan berhasil disimpan di basis data dengan kondisi utuh.

[Halaman ini sengaja dikosongkan]

BAB 5
PENUTUP

Pada bab terakhir ini, akan dijelaskan beberapa kesimpulan yang diperoleh selama pengerjaan tugas akhir dan saran-saran yang dapat digunakan sebagai bahan pertimbangan untuk pengembangan / penelitian selanjutnya.

5.1 Kesimpulan

Pada penelitian sebelumnya, *e-nose* sekedar dibangun untuk aplikasi yang berdiri sendiri (*stand alone*). Penelitian ini mengusulkan sebuah sistem *e-nose* yang berbasis *internet of things* (IoT). Sistem usulan menerapkan arsitektur *fog* yang terdiri dari tiga komponen, yakni *sensor node, fog node,* dan *cloud node.* Dengan membagi tugas ke tiga perangkat tersebut, pemrosesan yang biasanya dibebankan seluruhnya ke *cloud node* menjadi berkurang, sehingga dapat dihasilkan sebuah sistem yang memberikan respon lebih cepat dan lebih terskalakan.

Untuk menjadikan perangkat *e-nose* sebagai *sensor node* yang cocok dengan karakteristik IoT yang kebanyakannya memiliki sumber daya terbatas, diperlukan langkah optimasi larik sensor, agar sensor yang digunakan dapat dikurangi tanpa menurunkan kinerja. Dari proses optimisasi larik sensor, sensor awal yang digunakan berjumlah 10, berkurang menjadi 4 sensor. Dari hasil evaluasi kinerja proses tersebut, algoritma KNN menjadi algoritma dengan tingkat akurasi terbaik, ia memperoleh akurasi 80%. Hasil akurasi tersebut diperoleh setelah melakukan *tuning* parameter tetangga menjadi 5 dan *tuning* waktu pembacaan menjadi 8 menit data. Sementara itu, peringkat kedua, diperoleh algoritma RF dengan akurasi sebesar 79%. Dengan melakukan *tuning* jumlah *tree* menjadi 100 dan menggunakan 6 menit data.

Sementara itu, dari hasil pengujian fungsionalitas dengan metode *black-box,* disimpulkan bahwa sistem yang diusulkan secara keseluruhan dapat berfungsi dengan baik. Selain itu, sistem yang diusulkan juga mampu mengklasifikasi sampel aroma pisang dengan akurasi sebesar 78% (algoritma

KNN). Dari hasil pengujian skalabilitas, sistem dengan konfigurasi satu unit *fog node* yang dihubungkan ke *cloud node*, mampu menangani hingga 300 *sensor node* secara bersamaan ketika menggunakan 8 menit data, dan 500 *sensor node* ketika menggunakan 6 menit data, dengan catatan terjadi penurunan performa komunikasi data *(throughput)* ketika mulai menambahkan 400 lebih *sensor node*.

5.2 Saran

1. Melakukan pengujian menggunakan beberapa *sensor node* fisik, bukan cuma simulasi.
2. Menggunakan beberapa *fog node,* dengan spesifikasi perangkat keras yang lebih mumpuni.
3. Meluncurkan perangkat lunak klasifikasi dan antarmuka sistem di awan.
4. Menerapkan teknologi *containerization* dengan *load balancing* seperti *docker* pada sistem untuk mempermudah peluncuran perangkat lunak ke beberapa *fog node* dan untuk membagi beban antar *fog node* agar merata.
5. Pengujian skalabilitas dari segai lain, misalnya dengan melihat performa perangkat keras.

DAFTAR PUSTAKA

Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., & Ayyash, M. (2015). Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Communications Surveys & Tutorials*, *17*(4), 2347–2376. https://doi.org/10.1109/COMST.2015.2444095

Ali, M. I., Ono, N., Kaysar, M., Shamszaman, Z. U., Pham, T. Le, Gao, F., … Mileo, A. (2017). Real-time data analytics and event detection for IoT-enabled communication systems. *Journal of Web Semantics*, *42*, 19–37. https://doi.org/10.1016/j.websem.2016.07.001

Arsenault, C. (n.d.). Scalability Testing - a How-To Guide and Checklist. Retrieved December 26, 2018, from https://www.keycdn.com/blog/scalability-testing

Barzu, A.-P., Carabas, M., & Tapus, N. (2017). Scalability of a Web Server: How Does Vertical Scalability Improve the Performance of a Server? *2017 21st International Conference on Control Systems and Computer Science (CSCS)*, 115–122. https://doi.org/10.1109/CSCS.2017.22

Berna, A. (2010). Metal oxide sensors for electronic noses and their application to food analysis. *Sensors*, *10*(4), 3882–3910. https://doi.org/10.3390/s100403882

Chansongkram, W., & Nimsuk, N. (2016). Development of a Wireless Electronic Nose Capable of Measuring Odors Both in Open and Closed Systems. *Procedia Computer Science*, *86*, 192–195. https://doi.org/10.1016/j.procs.2016.05.060

Choi, J., Park, J. S., Chang, S., Lee, H. R., & Lee, H. (2015). *The Wireless Electronic Noses and Mobile Devices Interoperation Based on Internet of Things Technology*. *120*(Gst), 149–152.

Climent, E., Pelegri-Sebastia, J., Sogorb, T., Talens, J., & Chilo, J. (2017). Development of the MOOSY4 eNose IoT for Sulphur-Based VOC Water Pollution Detection. *Sensors*, *17*(8), 1917. https://doi.org/10.3390/s17081917

da Cruz, M. A. A., Rodrigues, J. J. P. C., Al-Muhtadi, J., Korotaev, V., & Albuquerque, V. H. C. (2018). A Reference Model for Internet of Things Middleware. *IEEE Internet of Things Journal*, *4662*(c), 1–1. https://doi.org/10.1109/JIOT.2018.2796561

Dastjerdi, A. V., & Buyya, R. (2016). Fog Computing: Helping the Internet of Things Realize Its Potential. *Computer*, *49*(8), 112–116. https://doi.org/10.1109/MC.2016.245

Deshmukh, L. P., Kasbe, M. S., Mujawar, T. H., Mule, S. S., & Shaligram, A. D. (2016). A wireless electronic nose (WEN) for the detection and classification of fruits: A case study. *2016 International Symposium on Electronics and Smart Devices (ISESD)*, 174–178. https://doi.org/10.1109/ISESD.2016.7886714

Farahani, B., Firouzi, F., Chang, V., Badaroglu, M., Constant, N., & Mankodiya, K. (2018). Towards fog-driven IoT eHealth: Promises and challenges of IoT in medicine and healthcare. *Future Generation Computer Systems*, *78*, 659–676. https://doi.org/10.1016/j.future.2017.04.036

Garcia-Esteban, J. A., Curto, B., Moreno, V., Gonzalez-Martin, I., Revilla, I., & Vivar-Quintana, A. (2018). A digitalization strategy for quality control in food industry based on Artificial Intelligence techniques. *2018 IEEE 16th International Conference on Industrial Informatics (INDIN)*, 221–226. https://doi.org/10.1109/INDIN.2018.8471994

Giang, N. K., Blackstock, M., Lea, R., & Leung, V. C. M. (2015). Developing IoT applications in the Fog: A Distributed Dataflow approach. *Proceedings - 2015 5th International Conference on the Internet of Things, IoT 2015*, 155–162. https://doi.org/10.1109/IOT.2015.7356560

Hassan, M., & Bermak, A. (2012). Wireless microelectronic nose network for gas classification. *Proceedings of the International Conference on Microelectronics, ICM*, (Icm), 2–5. https://doi.org/10.1109/ICM.2012.6471410

Heredia, A. P. D., Cruz, F. R., Balbin, J. R., & Chung, W.-Y. (2016). Olfactory classification using electronic nose system via artificial neural network. *2016 IEEE Region 10 Conference (TENCON)*, 3569–3574. https://doi.org/10.1109/TENCON.2016.7848722

Herrero, J. L., Lozano, J., Santos, J. P., & Suárez, J. I. (2016). On-line classification of pollutants in water using wireless portable electronic noses. *Chemosphere*, *152*, 107–116. https://doi.org/10.1016/j.chemosphere.2016.02.106

Jiang, S., Wang, J., Wang, Y., & Cheng, S. (2017). A novel framework for analyzing MOS E-nose data based on voting theory: Application to evaluate the internal quality of Chinese pecans. *Sensors and Actuators, B: Chemical*, *242*, 511–521. https://doi.org/10.1016/j.snb.2016.11.074

Kuo, Y. W., Li, C. L., Jhang, J. H., & Lin, S. (2018). Design of a Wireless Sensor Network-Based IoT Platform for Wide Area and Heterogeneous Applications. *IEEE Sensors Journal*, *18*(12), 5187–5197. https://doi.org/10.1109/JSEN.2018.2832664

López, P., Triviño, R., Calderón, D., Arcentales, A., & Guamán, A. V. (2017). Electronic nose prototype for explosive detection. *2017 CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies, CHILECON 2017 - Proceedings*, *2017-Janua*, 1–4. https://doi.org/10.1109/CHILECON.2017.8229657

Loutfi, A., Coradeschi, S., Mani, G. K., Shankar, P., & Rayappan, J. B. B. (2015). Electronic noses for food quality: A review. *Journal of Food Engineering*, *144*, 103–111. https://doi.org/10.1016/j.jfoodeng.2014.07.019

McWilliams, A., Beigi, P., Srinidhi, A., Lam, S., & MacAulay, C. E. (2015). Sex and smoking status effects on the early detection of early lung cancer in high-risk smokers using an electronic nose. *IEEE Transactions on Biomedical Engineering*, *62*(8), 2044–2054. https://doi.org/10.1109/TBME.2015.2409092

Morillo, P., Orduña, J. M., Fernández, M., & García-Pereira, I. (2018). Comparison of WSN and IoT approaches for a real-time monitoring system of meal distribution trolleys: A case study. *Future Generation Computer Systems*, *87*, 242–250. https://doi.org/10.1016/j.future.2018.01.032

Morin, B., Harrand, N., & Fleurey, F. (2017). Model-Based Software Engineering to Tame the IoT Jungle. *IEEE Software*, *34*(1), 30–36. https://doi.org/10.1109/MS.2017.11

Patel, H. K., Austin, R. H., Barber, J., & Patel, H. K. (2014). The Electronic Nose: Artificial Olfaction Technology. In *Biological and Medical Physics, Biomedical Engineering*. https://doi.org/10.1007/978-81-322-1548-6

Pattabhiraman, P., Bai, X., & Tsai, T. (2011). *SaaS Performance and Scalability in Clouds*. (Sose), 61–71.

Persaud, K., & Dodd, G. (1982). Analysis of discrimination mechanisms in the mammalian olfactory system using a model nose. *Nature*, *299*(5881), 352–355. https://doi.org/10.1038/299352a0

Ravindra, V., & Rajbhoj, S. M. (2017). Identification of Toxic Gases using Electronic Nose. *2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA)*, 1–5. https://doi.org/10.1109/ICCUBEA.2017.8463908

Ray, P. P. (2016). A survey on Internet of Things architectures. *Journal of King Saud University - Computer and Information Sciences*. https://doi.org/10.1016/j.jksuci.2016.10.003

Razzaque, M. A., Milojevic-Jevric, M., Palade, A., & Clarke, S. (2016). Middleware for Internet of Things: A Survey. *IEEE Internet of Things Journal*, *3*(1), 70–95. https://doi.org/10.1109/JIOT.2015.2498900

Saad, N. H., Jaffar, A., Kuncoro, C. B. D., Kasolang, S., Low, C. Y., & Armansyah. (2012). Wireless e-Nose Sensor Node: State of the Art. *Procedia Engineering*, *41*(Iris), 1405–1411. https://doi.org/10.1016/j.proeng.2012.07.328

Somasagar, R. B., & Kusagur, A. (2017). Flavor Determination for Milk Quality Assessment using Embedded Electronic Noses. *2017 2nd International Conference On Emerging Computation and Information Technologies (ICECIT)*, 1–4. https://doi.org/10.1109/ICECIT.2017.8453375

Szulczyński, B., & Gębicki, J. (2017). Currently Commercially Available Chemical Sensors Employed for Detection of Volatile Organic Compounds in Outdoor and Indoor Air. *Environments*, *4*(1), 21. https://doi.org/10.3390/environments4010021

Taivalsaari, A., & Mikkonen, T. (2017). A Roadmap to the Programmable World: Software Challenges in the IoT Era. *IEEE Software*, *34*(1), 72–80. https://doi.org/10.1109/MS.2017.26

Valdez, L. F., & Gutiérrez, J. M. (2016). Chocolate classification by an electronic nose with pressure controlled generated stimulation. *Sensors (Switzerland)*, *16*(10). https://doi.org/10.3390/s16101745

Wijaya, D. R., & Sarno, R. (2015). Mobile Electronic Nose Architecture for Beef Quality Detection Based on Internet of Things Technology. *SECOND INTERNATIONAL CONFERENCE ON Global Trends in Academic Research*

GTAR, At Bandung, Indonesia, 2, 655–663. Retrieved from http://www.globalilluminators.org/wp-content/uploads/2015/05/GTAR-15-331.pdf

Wijaya, D. R., Sarno, R., Zulaika, E., & Sabila, S. I. (2017). Development of mobile electronic nose for beef quality monitoring. *Procedia Computer Science*, *124*, 728–735. https://doi.org/10.1016/j.procs.2017.12.211

Wilson, A. D., & Baietto, M. (2011). Advances in electronic-nose technologies developed for biomedical applications. *Sensors*, *11*(1), 1105–1176. https://doi.org/10.3390/s110101105

Wojnowski, W., Majchrzak, T., Dymerski, T., Gębicki, J., & Namieśnik, J. (2017). Electronic noses: Powerful tools in meat quality assessment. *Meat Science*, *131*, 119–131. https://doi.org/10.1016/j.meatsci.2017.04.240

Yigitoglu, E., Mohamed, M., Liu, L., & Ludwig, H. (2017). Foggy: A Framework for Continuous Automated IoT Application Deployment in Fog Computing. *Proceedings - 2017 IEEE 6th International Conference on AI and Mobile Services, AIMS 2017*, 38–45. https://doi.org/10.1109/AIMS.2017.14

Yu, H., Meng, M., & Yin, Y. (2013). The research on the application of electronic nose in discriminate the rice varieties. *International Conference on Advanced Mechatronic Systems, ICAMechS*, 449–454. https://doi.org/10.1109/ICAMechS.2013.6681826

Zheng, Z., & Lin, X. (2012). Study on Application of Medical Diagnosis by Electronic Nose. *World Science and Technology*, *14*(6), 2115–2119. https://doi.org/10.1016/S1876-3553(13)60016-2

[Halaman ini sengaja dikosongkan]

BIOGRAFI PENULIS

Muhammad Asep Subandri lahir pada tanggal 9 Desember 1992 di Bengkalis, Riau. Merupakan anak pertama dari dua bersaudara pasangan Bapak Muhammad Yusuf dan Ibu Suryana. Penulis memulai pendidikan di SD Negeri 003 Siak, lalu melanjutkan di SMP Negeri 2 Siak. Penulis menempuh jenjang sekolah menengah di SMKN 1 Siak. Pada tahun 2010 penulis diterima di Universitas Islam Negeri Sultan Syarif Kasim Riau Jurusan Sistem Informasi. Setelah menyelesaikan studi di tahap sarjana, pada tahun 2016, penulis melanjutkan pendidikan di Program Magister Jurusan Teknik Informatika Institut Teknologi Sepuluh Nopember (ITS) dengan memilih bidang keahlian Rekayasa Perangkat Lunak (RPL).

E-mail: madasepandri@gmail.com