TUGAS AKHIR – TE141599

RANCANG BANGUN SISTEM PENDETEKSI KESEGARAN DAGING BERDASARKAN SENSOR BAU DAN WARNA

Joshwa Simamora
NRP 2212100192

Dosen Pembimbing
Dr. Muhammad Rivai, ST., MT.
Fajar Budiman, ST., M.Sc.

JURUSAN TEKNIK ELEKTRO
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya 2017

FINAL PROJECT – TE141599

DESIGN OF MEAT FRESHNESS DETECTION SYSTEM BASED ON SMELL AND COLOR SENSORS

Joshwa Simamora
NRP 2212100192

Supervisor
Dr. Muhammad Rivai, ST., MT.
Fajar Budiman, ST., M.Sc.

ELECTRICAL ENGINEERING DEPARTMENT
Faculty of Industrial Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2017

PERNYATAAN KEASLIAN
TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya dengan judul "RANCANG BANGUN SISTEM PENDETEKSI KESEGARAN DAGING BERDASARKAN SENSOR BAU DAN WARNA" adalah benar-benar hasil karya intelektual sendiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diijinkan dan bukan merupakan karya orang lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun durujuk telah ditulis secara lengkap pada daftar pustaka. Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai dengan peraturan yang berlaku.

Surabaya, 19 Januari 2017

Joshwa Simamora
NRP 2212100192

RANCANG BANGUN SISTEM PENDETEKSI KESEGARAN DAGING BERDASARKAN SENSOR BAU DAN WARNA

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik
Pada
Bidang Studi Elektronika
Jurusan Teknik Elektro
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember

Menyetujui

Dosen Pembimbing I	Dosen Pembimbing II
Dr. Muhammad Rivai, ST., MT.	**Fajar Budiman, ST., M.Sc.**
NIP. 196904261994031003	**NIP. 198607072014041001**

SURABAYA
JANUARI, 2017

RANCANG BANGUN SISTEM PENDETEKSI KESEGARAN DAGING BERDASARKAN SENSOR BAU DAN WARNA

Nama : Joshwa Simamora
Pembimbing I : Dr. Muhammad Rivai, ST., MT.
Pembimbing II : Fajar Budiman, ST., M.Sc.

ABSTRAK

Kesegaran daging adalah faktor terpenting dalam menentukan kelayakan dari sebuah daging untuk dikonsumsi. Pada tugas akhir ini dirancang sebuah sistem yang dapat mengidentifikasi tingkat kesegaran daging secara cepat, presisi, dan bersifat *non-destructive*. Sistem ini diimplementasikan ke dalam Raspberry Pi dengan menggunakan sensor gas dan sensor warna sebagai alat pendeteksi kesegaran yang menggantikan indera penciuman dan penglihatan pada manusia dalam menentukan tingkat kesegaran daging.

Pada tugas akhir ini digunakan *neural network* sebagai metode untuk melakukan pengenalan pola pada tingkat kesegaran daging yang diuji. Input yang digunakan pada *neural network* adalah berupa nilai tegangan dari ketiga buah sensor gas yaitu sensor MQ-136, MQ-137 dan TGS 2602 beserta nilai *Red*, *Green* dan *Blue* yang didapatkan dari sensor warna TCS 3200. Terdapat 3 buah kondisi kesegaran daging yang diuji yaitu daging segar, daging agak busuk dan daging busuk.

Penggunaan ketiga buah sensor gas dan sensor warna pada sistem telah berhasil mendapatkan pola yang khusus untuk setiap tingkat kesegaran daging yang diuji. Dari hasil pengujian terhadap tiga buah sampel yang mewakili tingkat kesegaran daging, didapatkan tingkat keberhasilan dalam proses identifikasi mencapai 80 %. Error terjadi pada identifikasi daging agak busuk dan busuk yang mempunyai pola yang tidak terlalu berbeda. Bagaimanapun daging ini tidaklah layak untuk dikomsumsi.

Kata kunci: kesegaran daging, *neural network*, sensor gas, sensor warna

Halaman ini sengaja dikosongkan

DESIGN OF MEAT FRESHNESS DETECTION SYSTEM BASED ON SMELL AND COLOR SENSORS

Name **: Joshwa Simamora**
Supervisor **: Dr. Muhammad Rivai, ST., MT.**
Co-Supervisor **: Fajar Budiman, ST., M.Sc.**

ABSTRACT

The freshness level of meat is the most important factor in determining the quality of meat for comsumption. In this final project, a sensor system has been designed to identify the freshness level of meat in fast, precise and non-destructive manners. The system is implemented into the Raspberry Pi by using gas and color sensors as the freshness identifier tools to replace the human vision and olfaction in determining a fresh meat.

In this final project, a neural network is used as a method to process the pattern recognition of the meat's freshness level sensed by the sensors. The inputs of the neural network are the voltage readings sensed by the gas sensors of MQ-136, MQ-137, TGS 2620 and Red, Green, Blue values sensed by TCS 3200 color sensor. There are three levels of freshness that has been tested, which are fresh meat, half-rotten meat, and rotten meat.

The usage of the three gas sensors and one color sensor of the system is capable to acquire a distinct pattern for the three categories of freshness. The freshness identification of the meat has a high percentage of success up to 80%. The errors are caused by the small different of the pattern sensed by sensors for half-rotten meat and rotten meat. However, these two kinds of meat are not consumable.

Key words: color sensor, freshness of meat, gas sensor, neural network

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Puji syukur kepada Tuhan Yang Maha Esa atas kasih dan rahmat-Nya penulis dapat menyelesaikan Tugas Akhir ini dengan judul :

Rancang Bangun Sistem Pendeteksi Kesegaran Daging Berdasarkan Sensor Bau dan Warna

Tugas Akhir ini merupakan persyaratan dalam menyelesaikan pendidikan program Strata-Satu di Jurusan Teknik Elektro, Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember Surabaya.

Pada kesempatan ini, penulis ingin berterima kasih kepada pihak-pihak yang telah membantu dalam pembuatan tugas akhir ini, khususnya kepada:

1. Bapak, Ibu, adik, serta seluruh keluarga yang memberikan dukungan baik moril maupun materiil.
2. Dr. Muhammad Rivai, ST., MT. selaku dosen pembimbing 1 atas bimbingan dan arahan selama penulis mengerjakan tugas akhir ini.
3. Fajar Budiman ST., M.Sc. selaku dosen pembimbing 2 atas bimbingan dan arahan selama penulis mengerjakan tugas akhir ini.
4. Para dosen penguji pada sidang Tugas Akhir yaitu Bapak Dr. Ir Hendra Kusuma, M.Eng.Sc., Ir. Tasripan, MT., Ir. Haris Pringadi, MT., Astria Nur Irfansyah, ST., M.Eng., Ph.D.
5. Seluruh dosen bidang studi elektronika dan teknik elektro.
6. Teman-teman laboratorium Elektronika yang tidak dapat disebutkan satu-persatu, telah membantu proses pengerjaan tugas akhir ini.

Penulis menyadari bahwa Tugas Akhir ini penuh dengan kekurangan dan masih banyak hal yang harus diperbaiki. Saran, kritik dan masukan dari semua pihak sangat membantu penulis untuk dapat menyempurnakan Tugas Akhir ini.

Surabaya, 19 Januari 2017

Penulis

Halaman ini sengaja dikosongkan

DAFTAR ISI

Halaman ini sengaja dikosongkan

DAFTAR GAMBAR

DAFTAR TABEL

Halaman ini sengaja dikosongkan

BAB I
PENDAHULUAN

1.1 Latar Belakang

Kesegaran daging merupakan faktor utama dalam menentukan kualitas dari sebuah daging. Tingkat kesegaran suatu daging akan menentukan apakah daging tersebut masih layak untuk dikonsumsi [1].

Saat ini masih digunakan cara tradisional untuk menentukan kualitas dan kesegaran sebuah daging yaitu dengan menggunakan kontak langsung manusia melalui inspeksi visual dan juga penciuman. Selain itu juga terdapat metode lain yang lebih modern yaitu dengan menggunakan metode pendeteksian secara kimiawi. Namun umumnya proses ini relatif kompleks, memakan waktu yang lama, serta bersifat destruktif (daging yang diuji akan rusak oleh zat kimia). Oleh karena itu sudah sewajarnya dibangun suatu sistem yang dapat mendeteksi tingkat kesegaran daging dengan cepat, akurat dan bersifat non-destruktif [2].

Dengan memanfaatkan karakteristik dari pembusukan daging, digunakan sebuah *electronic nose* dan sensor warna untuk dapat mendeteksi tingkat kesegaran daging. *Electronic nose* terdiri dari tiga buah sensor gas yang akan mendeteksi bau yang dikeluarkan oleh daging. Kemudian sensor warna akan digunakan untuk mendeteksi perubahan nilai RGB dari warna daging.

Dalam Tugas Akhir ini sistem dibangun dengan menggunakan *neural network*. Nilai tegangan dari sensor gas dan juga nilai RGB dari sensor warna akan menjadi input dari *neural network* yang dirancang. Output dari sistem ini adalah berupa pengenalan tingkat kesegaran daging yang diuji.

1.2 Perumusan Masalah

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

1. Bagaimana sensor gas dan sensor warna dapat mengidentifikasi tingkat kesegaran suatu daging.
2. Bagaimana proses pengolahan data tegangan dari sensor gas dan data berupa nilai RGB dari sensor warna agar tingkat kesegaran daging dapat diketahui secara akurat.
3. Bagaimana mengimplementasikan metode *neural network* untuk identifikasi tingkat kesegaran daging.

4. Bagaimana cara mengimplementasikan sistem yang dibuat ke dalam Raspberry Pi.

1.3 Tujuan Tugas Akhir

Penelitian pada tugas akhir ini memiliki tujuan sebagai berikut :

1. Mampu menggunakan sensor gas semikonduktor jenis MQ-136, MQ-137, TGS 2602 dan sensor warna TCS 3200 sebagai alat untuk mengidentifikasi tingkat kesegaran daging.
2. Mampu menampilkan data dari masing-masing sensor yang digunakan ke dalam Raspberry Pi.
3. Mampu menerapkan metode *neural network* untuk mengenali tingkat kesegaran daging.
4. Mampu melakukan pengenalan terhadap tingkat kesegaran daging secara cepat, *real-time*, dan bersifat non-destruktif yang diimplementasikan dalam Raspberry Pi.

1.4 Batasan Masalah

Batasan masalah pada tugas akhir ini adalah sebagai berikut :

1. Parameter yang akan digunakan untuk menentukan tingkat kesegaran daging adalah gas H_2S dan NH_3 yang dihasilkan oleh daging serta karakteristik dari perubahan warna daging.
2. Daging yang diuji merupakan daging sapi.
3. Proses akuisisi data tegangan sensor gas dan nilai RGB dari sensor warna menggunakan mikrokontroller Arduino dan dikirim menggunakan komunikasi serial menuju Raspberry Pi.

1.5 Metodologi Penelitian

Dalam pengerjaan tugas akhir ini digunakan metodologi sebagai berikut :

1. Studi Literatur

   Pada tahap ini dilakukan pengumpulan dasar teori yang menunjang dalam proses pengerjaan maupun penulisan tugas akhir. Studi literatur yang dilakukan adalah sebagai berikut :

   a. Studi mengenai penggunaan sensor gas dan sensor warna serta penggunaan ADC 10 bit pada Arduino.
   b. Studi mengenai komunikasi serial antara Arduino dan Raspberry Pi.

c. Studi mengenai *electronic nose*, *neural network* dan pemrogramannya pada *software* Lazarus.

Dasar teori ini dapat diambil dari buku, jurnal, *paper* dan tutorial yang terdapat di internet.

2. Perancangan Sistem

Setelah mempelajari dasar teori dan literatur yang ada, langkah selanjutnya adalah melakukan perancangan sistem. Sistem yang akan dirancang meliputi dua buah bagian yaitu perancangan *hardware* dan perancangan *software*.

Perancangan *hardware* meliputi perancangan modul sensor gas, perancangan ruang uji sensor, dan desain mekanik. Sedangkan perancangan *software* meliputi pemrograman mikrokontroller Arduino dan Raspberry Pi menggunakan Lazarus.

3. Pengujian Sistem

Pengujian sistem dilakukan secara bertahap dengan cara menguji sistem satu per satu atau bagian demi bagian. Hal ini dilakukan untuk mengetahui apakah setiap blok dari sistem yang telah dibuat dapat berfungsi secara benar. Setelah semua bagian dipastikan telah bekerja dengan baik, maka alat akan diuji untuk mendeteksi tingkat kesegaran daging. Pengujian ini bertujuan untuk mengambil nilai tegangan sensor gas dan nilai warna dari daging tersebut. Kemudian akan dilakukan evaluasi dan perbaikan terhadap sistem yang telah dibuat.

4. Pengolahan Data

Data berupa karakteristik gas dan warna dari daging yang telah diperoleh digunakan sebagai data untuk proses *learning* pada *neural network* dan juga untuk proses *forward propagation* pada saat mengidentifikasi tingkat kesegaran daging.

5. Penulisan Laporan Tugas Akhir

Tahap penulisan laporan tugas akhir dilakukan pada saat tahap pengujian sistem dimulai sampai selesai.

1.6 Sistematika Penulisan

Laporan tugas akhir ini terdiri dari lima bab dengan sistematika penulisan sebagai berikut :

1. Bab 1 : Pendahuluan

   Bab ini meliputi latar belakang, perumusan masalah, tujuan penelitian, batasan masalah, metodologi, sistematika penulisan, dan relevansi.

2. Bab 2 : Dasar Teori

   Bab ini menjelaskan mengenai dasar-dasar teori yang dibutuhkan dalam pengerjaan tugas akhir ini, yang meliputi teori dasar mengenai karakteristik daging dan fenomena pembusukannya, sensor gas jenis MQ dan juga TGS, sensor warna TCS 3200, mikrokontroller Arduino, *analog to digital converter*, komunikasi serial, Raspberry Pi dan terakhir adalah mengenai teori *neural network.*

3. Bab 3 : Perancangan Alat

   Bab ini menjelaskan tentang perencanaan sistem berupa sistem *hardware* maupun *software* untuk mendeteksi tingkat kesegaran daging yang diuji.

4. Bab 4 : Pengujian Alat

   Bab ini menjelaskan mengenai hasil yang didapat dari tiap blok sistem dan juga subsistem serta hasil evaluasi dari sistem tersebut.

5. Bab 5 : Penutup

   Bab ini menjelaskan mengenai kesimpulan dan juga kekurangan dari alat yang telah dirancang beserta dengan saran untuk pengembangan penelitian di masa yang akan datang.

1.7 Relevansi

Hasil dari tugas akhir ini diharapkan dapat memberikan manfaat sebagai berikut :

1. Dapat mendukung penelitian yang melibatkan dua disiplin ilmu yang berbeda yaitu bidang elektronika dan pertanian (peternakan) untuk menghasilkan alat pendeteksi kesegaran daging yang lebih mutakhir di masa yang akan datang.
2. Dihasilkan alat-alat yang dapat mendeteksi kesegaran jenis makanan lainnya dan tidak berfokus pada daging saja.

BAB II
DASAR TEORI

2.1 Kesegaran Daging

Faktor terpenting yang mempengaruhi kesegaran dan kualitas dari daging adalah aroma, warna, tekstur, dan rasa. Kualitas rasa dari daging itu sendiri ditentukan oleh banyaknya kandungan *volatile organic compound* (VOC) yang terdapat di dalamnya. Daging dapat diklasifikasikan menggunakan sebuah *electronic nose* dengan cara yang sama seperti persepsi manusia dalam menentukan kualitas dan tingkat kesegaran. Aroma atau bau dari daging terbentuk dari gabungan kompleks dari beberapa VOC yang berasal dari beragam reaksi kimia yang terjadi dalam daging. Banyak pendapat yang menyatakan jika sebuah daging segar tidak memiliki bau sama sekali [3].

Pembusukan daging dapat disebabkan oleh aktivitas mikroorganisme dalam daging atau karena terjadi pelepasan enzim intraselular dan ekstraseluler mikrobial pada daging. Parameter dari kebusukan daging antara lain adalah perubahan warna dan aroma, tekstur, terbentuknya lendir, dan terbentuknya gas.

Klasifikasi tingkat kesegaran daging yang ada saat ini didasarkan pada jumlah kandungan *total volatile basic nitrogen* (TVB-N) yang terdapat pada daging. TVB-N adalah jumlah material nitrit yang disuling dari uap atau gas dari daging dalam kondisi alkalisasi. TVB-N ini mengandung seluruh kandungan nitrogen yang dapat membentuk ammonia dalam kondisi tersebut.

Berdasarkan standar nasional RRC GB2722-81, didefinisikan korelasi antara nilai TVB-N yang terkandung dalam daging dengan tingkat kesegarannya yang ditunjukkan pada Tabel 2.1. Daging segar difefinisikan sebagai daging dengan kandungan TVB-N lebih kecil dari 15 mg / 100 g daging. Daging setengah segar bernilai antara 15-30 mg / 100g daging dan daging busuk bernilai diatas 30 mg / 100 g daging.

Tabel 2.1 Definisi tingkat kesegaran daging berdasarkan TVB-N.

Tingkat Kesegaran Daging	Jumlah kandungan TVB-N
Daging Segar	< 15 mg / 100 g
Daging Setengah Segar	15 – 30 mg / 100 g
Daging Busuk	> 30 mg / 100 g

2.2 Electronic Nose

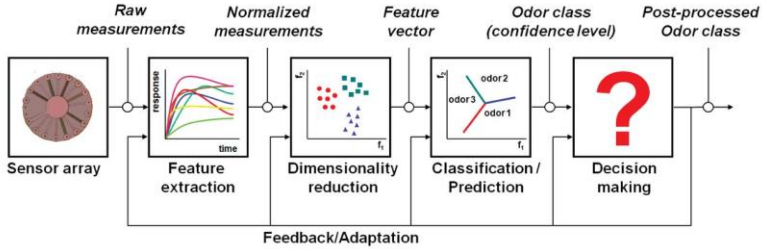
Sebuah *electronic nose* atau yang sering disingkat sebagai *e-nose* adalah instrumen analitik yang dirancang untuk meniru cara manusia dalam merasakan bau. Pada *e-nose*, proses analisanya tidaklah berfokus pada identifikasi dan kuantifikasi dari campuran gas yang menguap namun lebih ke arah deskripsi kuantitatif dari profil aroma secara keseluruhan meliputi hubungan antar komponen.

Dua buah komponen utama dari sebuah *electronic nose* adalah sistem perasa (*sensing system*) dan sistem pengenalan pola (*pattern recognition system). Sensing system* dapat terdiri atas *array* atau deret dari beberapa *sensing element* yang berbeda (misalnya sensor kimiawi), dimana setiap elemen mengukur sifat yang berbeda dari bahan kimia yang diuji. Setiap uap kimiawi atau gas yang dipaparkan terhadap *array* atau deret sensor akan menghasilkan sebuah ciri khas atau karakteristik pola dari gas tersebut.

Dengan memaparkan sebuah deret / *array* dari sensor terhadap jenis gas yang berbeda-beda, maka sebuah *database* berupa pola atau ciri khas dari gas tersebut dapat dibangun. *Database* dari pola atau ciri khas ini kemudian dapat digunakan untuk melatih sistem pengenalan pola. Tujuan dari proses *training* ini adalah untuk mengatur *recognition system* agar menghasilkan klasifikasi yang unik dari setiap gas sehingga proses pengidentifikasian secara otomatis dapat diimplementasikan [4].

Artificial neural networks atau yang sering disingkat sebagai ANN telah digunakan untuk menganalisa data yang kompleks dan dapat digunakan untuk melakukan pengenalan pola (*pattern recognition*), oleh karena itu penggunaan ANN bersamaan dengan *e-nose* dapat melakukan pengenalan pola dari gas kimiawi. ANN yang sudah dilatih untuk tujuan pengenalan gas dapat mengidentifikasi suatu gas secara cepat di lapangan. Hal ini disebabkan karena proses pengenalan hanya melibatkan proses propagasi maju yang pada dasarnya merupakan operasi perkalian dan penjumlahan dalam matriks.

Proses kerja dari *e-nose* dapat diilustrasikan pada Gambar 2.1. Bagian pertama dari blok diagram menunjukkan sebuah deret sensor yang merupakan *hardware* dari sebuah *e-nose*. Setelah sinyal dari deret sensor didapatkan dan disimpan dalam komputer, maka proses perhitungan (pemrosesan sinyal) pertama akan dimulai yang bertujuan untuk mengekstrak parameter deskriptif dari respon deret sensor dan mempersiapkan *feature vector* untuk proses selanjutnya [5].



Gambar 2.1 Proses kerja sebuah *e-nose* [6].

Tahap *dimensionality reduction* memproyeksikan *feature vector* awal menuju ke dimensi yang lebih rendah untuk menghindari masalah yang berhubungan dengan himpunan data yang tersebar pada dimensi yang lebih tinggi. *Feature vector* pada dimensi rendah yang telah dihasilkan akan digunakan untuk melakukan proses klasifikasi atau prediksi. Proses klasifikasi adalah identifikasi sebuah sampel gas yang tidak diketahui dengan menggunakan himpunan data yang telah melalui proses *training*.

2.3 Sensor Gas Semikonduktor

Hingga saat ini telah terdapat beberapa jenis sensor gas yang telah dikembangkan. Sensor gas dibedakan berdasarkan bahan atau material pembentuknya diantaranya adalah jenis *metal oxide semiconductor* (MOS), *conducting polymer* (CP), dan sensor piezoelektrik seperti *quartz crystal microbalances* (QCM). Sensor gas berjenis MOS merupakan salah satu jenis sensor yang paling banyak digunakan untuk membangun sistem *electronic nose*. Hal ini disebabkan oleh sensitivitas sensor yang tinggi dan harganya yang relatif murah [7].

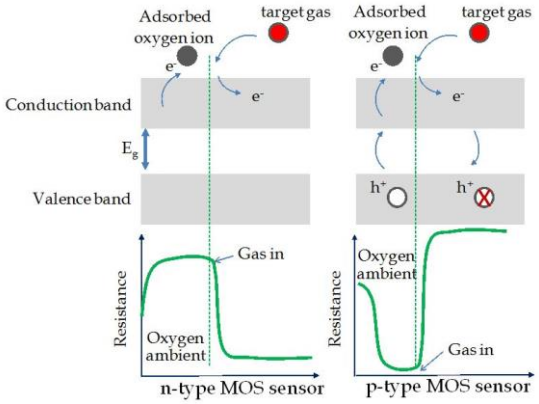
Generasi pertama dari sensor gas tipe MOS dibuat dari bahan berupa lapisan tebal dari SnO_2 pada tahun 1960 yang pertama kali diamati oleh Taguchi. Sensor gas tipe MOS memiliki beberapa keuntungan antara lain adalah ukurannya yang kecil, konsumsi daya yang rendah, konstruksi yang sederhana, dan kompabilitas yang tinggi dengan pemrosesan mikroelektronika [8]. Hal ini menyebabkan teknologi sensor gas tipe MOS berkembang secara pesat beberapa tahun belakangan ini.

Prinsip kerja dari sensor gas tipe MOS dapat dirangkum menjadi dua tahap seperti pada Gambar 2.2. Tahap pertama adalah pada saat sensor berada dalam udara bersih, elektron donor yang berada di dalam

SnO_2 akan tertarik menuju ke arah oksigen yang diserap pada permukaan dari *sensing material* yang akan mencegah adanya aliran arus listrik. Tahap kedua adalah saat sensor berada dalam paparan gas yang terdeteksi. Hal ini menyebabkan kerapatan permukaan dari oksigen yang diserap akan berkurang seiring dengan reaksi yang terjadi terhadap gas yang terdeteksi. Elektron kemudian akan dilepaskan menuju ke dalam SnO_2 yang menyebabkan arus listrik mengalir secara bebas pada sensor.

Pada kasus ekstrim kebanyakan, dimana konsentrasi oksigen adalah sebesar 0%, saat material sensor berupa *metal oxide* (umumnya adalah *tin oxide* / SnO_2) dipanaskan pada suhu tinggi misalnya pada 400°C, elektron bebas akan mengalir melalui *grain boundary* dari kristal *tin dioxide*. Dalam udara bersih (sekitar 21% O_2), oksigen diserap pada permukaan *metal oxide*. Dengan tingkat affinitas elektron yang tinggi, oksigen yang diserap akan menarik elektron bebas ke dalam *metal oxide*, dan membentuk sebuah *potential barrier* (eVs di udara) pada *grain boundaries*. *Potential barrier* yang terbentuk akan mencegah aliran elektron dan mengakibatkan sensor memiliki resistansi yang tinggi di udara bersih.

Ketika sensor terpapar terhadap gas yang terdeteksi (seperti karbon dioksida), reaksi oksidasi antara gas tersebut dengan oksigen yang diserap akan terjadi pada permukaan dari *tin dioxide*. Sebagai akibatnya kerapatan dari oksigen yang diserap pada permukaan *tin dioxide* akan



Gambar 2.2 Prinsip kerja sensor gas tipe MOS [8].

berkurang, dan ketinggian dari *potential barrier* akan berkurang. Elektron dapat mengalir dengan mudah melalui *potential barrier* yang telah mengalami pengurangan ketinggian dan resistansi sensor akan berkurang.

Konsentrasi gas di udara dapat dideteksi dengan mengukur perubahan resistansi dari sensor gas tipe MOS. Reaksi kimia dari gas yang terdeteksi dan oksigen yang diserap pada permukaan *tin dioxide* bervariasi bergantung pada reaktivitas dari *sensing materials* dan suhu kerja dari sensor tersebut [9].

2.3.1 Sensor Gas Figaro

Elemen sensor terdiri dari lapisan oksida logam semikonduktor dibentuk pada substrat alumina dari *chip* sensor bersama-sama dengan pemanas yang terintegrasi. Dengan adanya gas yang terdeteksi, konduktivitas sensor meningkat bergantung pada konsentrasi gas di udara. Sebuah rangkaian listrik sederhana dapat mengkonversi perubahan konduktivitas untuk sinyal output yang sesuai dengan konsentrasi gas. Karena chip sensor yang kecil, sensor gas Figaro hanya membutuhkan arus sebesar 42 mA. Fitur-fitur yang terdapat pada sensor gas Figaro :

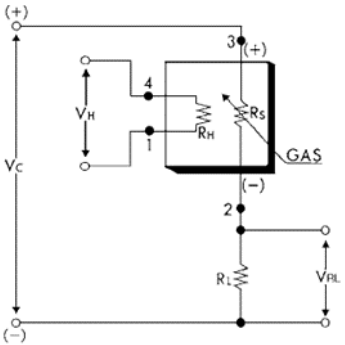
1. Konsumsi daya yang rendah
2. Tahan lama dan harga yang murah
3. Menggunakan rangkaian listrik yang sederhana
4. Ukuran chip yang kecil

2.3.2 Sensor Gas TGS 2602

Sensing element dari sensor ini terdiri atas sebuah lapisan *metal oxide semiconductor* yang terbentuk pada substrat alumina dari *sensing chip* bersamaan dengan sebuah *heater* yang terintegrasi.

Apabila sensor terpapar gas yang terdeteksi maka konduktivitas sensor akan bertambah bergantung pada konsentrasi gas tersebut di udara. Sebuah rangkaian listrik yang sederhana dapat mengubah perubahan konduktivitas sensor menjadi sebuah sinyal output yang berkorelasi dengan konsentrasi gas yang terdeteksi.

Gambar 2.3 Bentuk fisik TGS 2602 [10].



Gambar 2.4 Skematik rangkaian sensor TGS 2602 [11].

Sensor TGS 2602 memiliki sensitivitas yang tinggi terhadap gas berbau dengan konsentrasi rendah seperti ammonia dan H_2S yang dihasilkan dari limbah rumah tangga dan lingkungan perkantoran. Sensor ini juga memiliki sensitivitas yang tinggi terhadap gas VOC berkonsentrasi rendah seperti toluene yang dihasilkan dari pemolesan kayu dan produk bangunan.

Berkat miniaturisasi terhadap *sensing chip*, TGS 2602 hanya membutuhkan arus *heater* sebesar 42 mA dan divais ini dibangun dalam bentuk TO-5 *package*.

Skematik rangkaian sensor TGS 2602 ditunjukkan pada Gambar 2.4. Dari skematik rangkaian sensor terdapat beberapa parameter seperti R_s dan R_L. R_s merupakan resistansi sensor yang nilainya akan berubah seiring dengan jenis dan konsentrasi gas yang diberikan kepada sensor. Sedangkan R_L sendiri merupakan resistansi beban yang terhubung secara seri terhadap resistansi sensor.

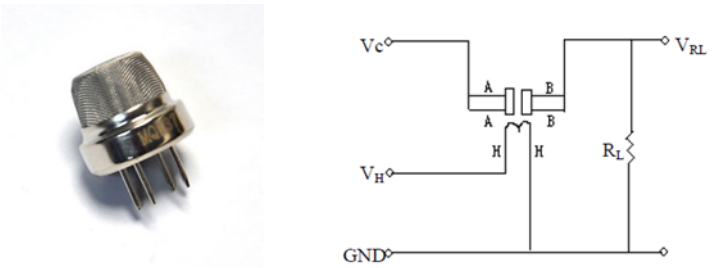
Dengan menerapkan prinsip pembagian tegangan maka konsentrasi gas yang terdeteksi oleh sensor dapat dikorelasikan dengan nilai tegangan pada resistansi beban R_L.

2.3.3 Sensor Gas MQ

Material sensitif dari sensor jenis MQ adalah berupa SnO_2, yang memiliki tingkat konduktivitas yang rendah pada udara bersih. Saat sensor terpapar oleh gas yang terdeteksi maka nilai konduktivitas sensor akan meningkat sebanding dengan konsentrasi gas di udara. Dengan menggunakan rangkaian listrik yang sederhana, perubahan konduktivitas sensor dapat dijadikan sebagai sinyal output yang berhubungan dengan konsentrasi gas yang terdeteksi di udara.

2.3.4 Sensor MQ-136 dan MQ-137

Kedua buah sensor jenis MQ ini memiliki prinsip kerja dan karakteristik yang sama. Yang membedakan keduanya hanyalah jenis gas yang dideteksi. Sensor MQ-136 dikhususkan untuk mendeteksi konsentrasi gas H_2S di udara sedangkan sensor MQ-137 dikhususkan untuk mendeteksi keberadaan gas NH_3 di udara.



Gambar 2.5 Bentuk fisik sensor MQ dan skematik rangkaian [12].

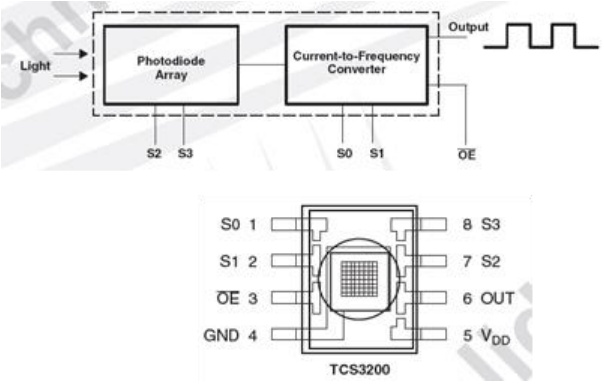
2.4 Sensor Warna TCS 3200

Sensor warna TCS 3200 merupakan sebuah *programmable color light-to-frequency converters* yang menggabungkan photodioda silikon dan sebuah *current-to-frequency converter* ke dalam sebuah IC CMOS monolitik. Blok diagram dari sensor ini ditunjukkan pada Gambar 2.6, dimana output dari sensor ini adalah berupa sebuah gelombang kotak dengan *duty cycle* sebesar 50 % dimana frekuensinya berbanding lurus terhadap intensitas cahaya.

Skala frekuensi output dari sensor dapat diatur ke dalam tiga pilihan skala yang tersedia melalui dua buah pin kontrol input. Dalam sensor TCS 3200, *light to frequency converter* membaca sebuah *array* dari photodioda berukuran 8 x 8. Enam belas buah photodioda memiliki filter warna hijau, 16 photodioda memiliki filter warna biru, 16 photodioda memiliki filter warna merah, dan 16 photodioda lainnya tanpa filter warna.

Frekuensi output dari TCS 3200 ini umumnya berkisar antara 2 Hz hingga 500 KHz. Pengguna dapat mengontrol nilai frekuensi ke dalam tiga nilai yaitu 100%, 20% dan 2% melalui kedua buah output yang dapat diprogram yaitu pin S0 dan S1.

Semua photodioda dengan warna yang sama terhubung secara parallel. Pin S2 dan S3 digunakan untuk memilih grup photodioda (*red, green, blue, clear*) yang akan diaktifkan. TCS 3200 juga memiliki sensitivitas yang berbeda terhadap warna merah, hijau dan biru. Akibatnya nilai output RGB dari warna putih tidaklah selalu bernilai 255.



Gambar 2.6 Blok diagram sensor warna TCS 3200 [13].

Tabel 2.2 Hubungan pin S0 dan S1 terhadap *frequency scaling*.

S0	S1	Output Frequency Scaling
L	L	Power down
L	H	2%
H	L	20%
H	H	100%

Tabel 2.3 Hubungan pin S2 dan S3 terhadap tipe filter.

S2	S3	Tipe filter
L	L	Merah
L	H	Biru
H	L	*Clear* (Tanpa filter)
H	H	Hijau

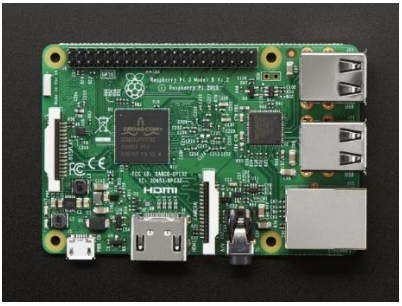
2.5 Raspberry Pi

Raspberry Pi adalah sebuah *single board computer* berukuran sebesar kartu kredit yang dikembangkan di Inggris oleh Raspberry Pi Foundation untuk mempromosikan pengajaran mengenai dasar ilmu komputer di sekolah dan negara-negara berkembang [14].

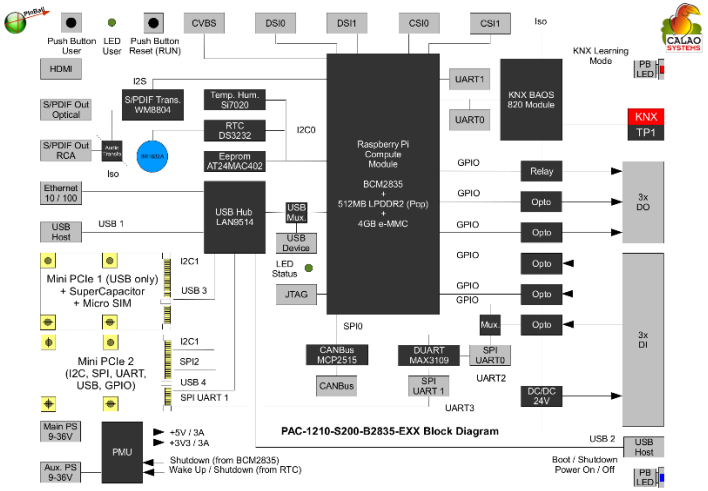
Raspberry Pi berbentuk seperti sebuah modul sehingga dapat disebut sebagai komputer mini. Agar dapat digunakan, Raspberry Pi harus dihubungkan ke layar monitor (via HDMI) dan perangkat input/output seperti *keyboard* dan juga *mouse*. Raspberry Pi dapat bekerja seperti sebuah komputer desktop pada umumnya dimana ia mampu digunakan untuk menjalankan aplikasi *spreadsheet*, pengolah kata (*word processing*), permainan, aplikasi pemrograman dan aplikasi multimedia seperti pemutar musik dan video.

Yang menjadikan Raspberry Pi sangatlah unik adalah kompabilitasnya terhadap peralatan elektronika seperti sensor, komponen elektronika dan bahasa pemrograman. Hal ini disebabkan oleh karena Raspberry Pi dilengkapi dengan pin GPIO yang juga dapat kita temukan pada beberapa tipe mikrokontroller (terutama jenis ARM). Selain itu ia juga dilengkapi dengan beberapa protokol komunikasi seperti I^2C dan SPI.

Desain Raspberry Pi didasarkan pada SoC (*System on chip*) Broadcom BCM2836, yang telah menanamkan processor ARM Cortex-A7 dengan kecepatan 900 MHz, VideoCore IV GPU, dan 256 Megabyte RAM (model B). Penyimpanan data dirancang tidak untuk menggunakan *hard disk* atau *solid-state drive*, melainkan mengandalkan kartu SD (SD *memory card*) untuk proses *booting* dan penyimpanan jangka panjang. Raspberry Pi utamanya menjalankan sistem operasi berbasis kernel Linux.



Gambar 2.7 Bentuk fisik dari Raspberry Pi [15].



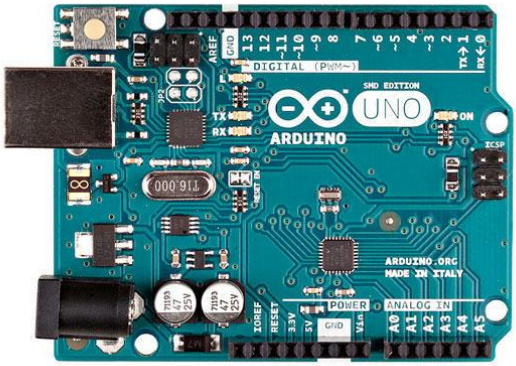
Gambar 2.8 Blok diagram dari Raspberry Pi.

2.6 Arduino Uno

Arduino Uno adalah sebuah mikrokontroller yang berbasis pada ATmega328P. Arduino Uno memiliki 14 buah pin digital input/output dimana 6 buah pin dapat digunakan sebagai output PWM, kemudian 6 buah pin input analog, sebuah kristal kuarsa dengan frekuensi 16 MHz, memiliki *port* USB, sebuah *power jack*, *header* ICSP dan sebuah tombol reset [16].

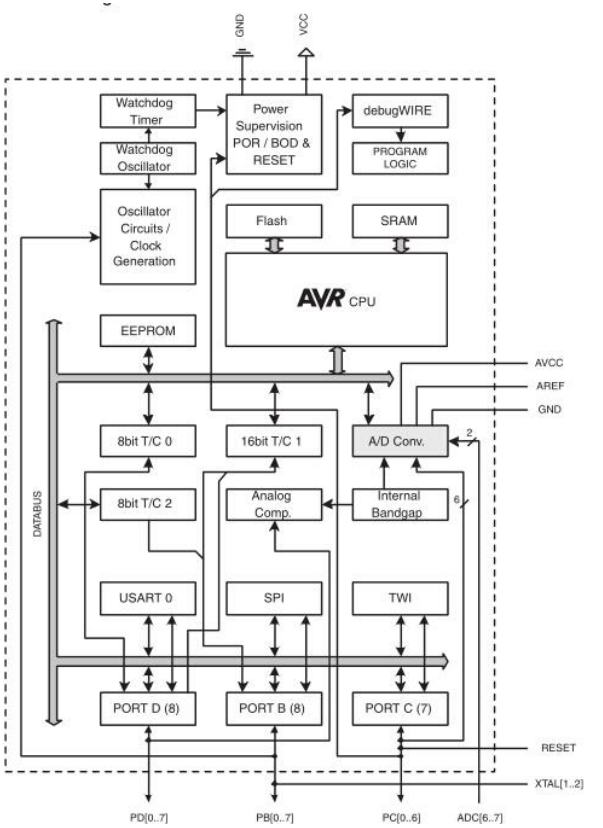
Arduino Uno memiliki semua hal yang dibutuhkan untuk mendukung sebuah mikrokontroller. Pengguna hanya perlu menghubungkan Arduino ke komputer melalui kabel USB atau menyalakannya dengan baterai maupun sebuah *adaptor*.

Arduino Uno dilengkapi dengan sebuah *polyfuse* yang dapat di*reset* untuk melindungi *port* USB dari komputer pengguna terhadap hubungan singkat maupun arus berlebih. Meskipun kebanyakan komputer telah dilengkapi dengan proteksi internal, namun *fuse* internal pada Arduino telah menambahkan lapisan perlindungan ekstra terhadap komputer. Jika Arduino menarik arus lebih besar dari 500 mA dari *port* USB, maka *fuse* ini akan putus secara otomatis sampai hubungan singkat atau kelebihan beban ini berhenti.



Gambar 2.9 Bentuk fisik dan diagram pin dari Arduino Uno [16].

Sama seperti mikrokontroller lainnya, Arduino mempunyai aplikasi pemrograman tersendiri yaitu Arduino Software (IDE). Bahasa pemrograman yang digunakan dalam software ini adalah Bahasa C++. Dikarenakan Bahasa C sudah begitu populer di masyarakat, maka tidak heran jika Arduino menjadi mikrokontroller yang paling disukai oleh kalangan mahasiswa, insinyur, maupun pencinta dunia elektronika. Bahasa pemrograman dalam Arduino bersifat *user-friendly* karena sintaks yang ada di dalamnya telah disederhanakan menjadi bahasa yang mudah dimengerti bagi orang awam yang belum mahir dalam pemrograman sekalipun.



Gambar 2.10 Blok diagram dari Arduino Uno.

2.7 Analog To Digital Converter

Sebuah rangkaian yang disebut sebagai *Analog to Digital Converter* dapat berupa sebuah *integrated circuit* atau sekumpulan op-amp dan perangkat lainnya. Sebuah ADC menerima sinyal analog sebagai sinyal input dan menghasilkan sebuah output digital yang berkorelasi dengan sinyal input analog tersebut. Output digital yang dihasilkan terdiri atas beberapa deret angka, yang bergantung terhadap besarnya resolusi dari ADC tersebut. Resolusi mendeskripsikan persentase dari perubahan tegangan input yang dibutuhkan untuk menyebabkan perubahan *step* pada output.

Misalkan sebuah ADC 8-bit dirancang untuk menerima sinyal input yang berkisar antara 0 V – 10 V. Selisih tegangan sebesar 10 V ini akan dibagi ke dalam 256 buah *step*, yaitu dengan 10/256 atau sekitar 39 miliVolt untuk setiap *step*. Sebaliknya sebuah ADC 4-bit akan memiliki nilai resolusi yang lebih rendah sehingga ke 16 buah step output yang ada bernilai sebesar 6,25% dari *full-scale input* atau sebesar 625 milivolt. Oleh karena itu semakin besar resolusi dari ADC, maka semakin kecil perubahan input yang dibutuhkan untuk berpindah ke *step* output selanjutnya. Umumnya besar resolusi yang sering digunakan dalam ADC adalah 8 bit, 10 bit, 12 bit dan 20 bit [17].

ADC pada Arduino Uno didasarkan pada ATmega 328P yang memiliki enam buah input analog yaitu pin A0-A5 pada Arduino Uno. Namun sebenarnya ATmega 328P hanya dilengkapi dengan sebuah ADC 10-bit tunggal. Hal ini dapat terjadi karena penggunaan *multiplexer* pada Arduino. Ke enam buah input analog disambungkan menuju ke ADC melalui *multiplexer* yang akan memilih jalur input secara otomatis sesuai dengan perintah yang diberikan pada program. Oleh karena itu Arduino Uno tidak dapat melakukan pengukuran nilai analog dalam waktu yang bersamaan.

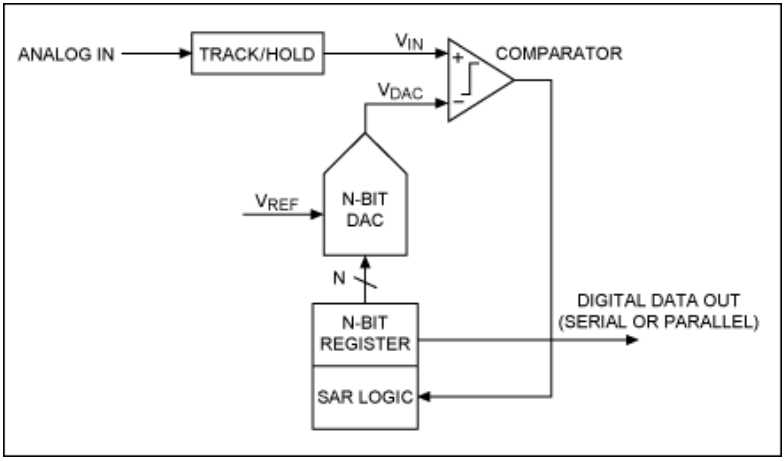
Tabel 2.4 Hubungan antara jumlah bit dan resolusi.

Resolusi dalam Bit (n)	Jumlah Step (2^n)	Resolusi sebagai Persentase dari Full Scale (%)
1	2	50
2	4	25
3	8	12.5
4	16	6.25

Resolusi dalam Bit (n)	Jumlah Step (2^n)	Resolusi sebagai Persentase dari Full Scale (%)
5	32	3.125
6	64	1.5625
7	128	0.78125
8	256	0.390625

Saat ini terdapat beberapa jenis ADC yang digunakan antara lain *flash* ADC, *tracking* ADC, *dual-slope* ADC, dan *successive approximation* ADC. Jenis ADC yang paling banyak digunakan saat ini adalah jenis *successive approximation* ADC. Keuntungan dari ADC jenis ini adalah kecepatan operasi yang tinggi dan menggunakan rangkaian yang cukup sederhana.

Sebuah blok diagram dari *successive approximation* ADC ditunjukkan pada Gambar 2.11. Terlihat bahwa sinyal input analog akan menjadi salah satu input bagi komparator. Sedangkan input komparator yang kedua berasal dari output sebuah DAC. Input yang menuju ke DAC disediakan oleh sebuah *latch* yang disebut dengan *successive approximation register* (SAR). Setiap bit dari *register* ini dapat diatur untuk *set* atau *clear* oleh unit kontrol.



Gambar 2.11 Blok diagram dari SAR ADC [18].

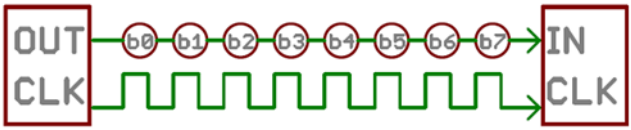
2.8 Komunikasi Serial

Tujuan utama dari sebuah sistem elektronika tertanam atau *embedded electronic* adalah untuk menghubungkan rangkaian satu ke rangkaian lainnya (*processor* atau *integrated circuit*) dengan tujuan membentuk sebuah sistem yang saling berhubungan. Agar masing-masing rangkaian dalam sistem ini dapat bertukar informasi maka mereka harus memiliki protokol komunikasi yang sama. Saat ini terdapat begitu banyak protokol komunikasi yang telah ditemukan untuk mencapai proses pertukaran data, dan secara umum dapat dikategorikan ke dalam salah satu dari dua kategori yaitu komunikasi parallel atau serial [19].

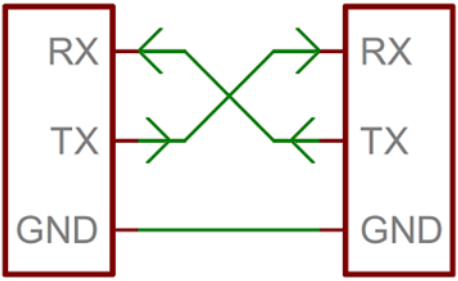
Komunikasi parallel mengirimkan beberapa bit data dalam waktu yang bersamaan. Jenis komunikasi ini umumnya memerlukan sebuah *data bus*, yang ditransmisikan melalui 8 atau 16 buah kabel, bahkan lebih. Komunikasi serial di lain pihak mentransmisikan data sebanyak satu bit untuk satu waktu. Komunikasi jenis ini dapat beroperasi dengan menggunakan minimal satu buah kabel dan biasanya tidak pernah melebihi empat buah kabel. Ilustrasi dari proses komunikasi serial ditunjukkan pada gambar di bawah ini.

Komunikasi parallel memiliki beberapa keuntungan antara lain komunikasi yang cepat, langsung, dan tergolong sederhana untuk diimplementasikan. Namun jenis komunikasi ini membutuhkan lebih banyak jalur input/output. Hal ini menjadikannya tidak praktis untuk diimplementasikan pada *embedded system* seperti mikroprosesor dikarenakan jumlah jalur input/output yang terbatas. Oleh karena itu saat ini komunikasi parallel perlahan-lahan mulai ditinggalkan dan digantikan oleh komunikasi serial.

Selama beberapa tahun ini sekumpulan protokol komunikasi serial telah dikembangkan untuk memenuhi kebutuhan dari *embedded system*. Beberapa *serial interface* yang umum kita temui saat ini adalah I^2C, SPI, dan *Universal Serial Bus*. Masing-masing protokol komunikasi serial ini kemudian dapat dibagi lagi ke dalam dua buah kategori yaitu *synchronous* dan *asynchronous*.



Gambar 2.12 Ilustrasi dari komunikasi serial.



Gambar 2.13 Hubungan dalam komunikasi serial.

Synchronous serial interface selalu memasangkan datanya dengan sebuah sinyal *clock*, sehingga semua divais pada *bus* dari *synchronous serial* akan memiliki *clock* yang sama. Hal ini mengakibatkan transfer data yang lebih mudah dan seringkali lebih cepat. Namun hal ini juga membutuhkan paling tidak satu buah kabel lainnya antara divais yang saling berkomunikasi. Contoh dari *synchronous interface* adalah SPI dan I^2C.

Asynchronous serial interface melakukan transfer data tanpa bantuan dari sinyal *clock* eksternal. Metode transmisi seperti ini sangatlah cocok untuk meminimalkan kabel dan jumlah pin input/output yang diperlukan. Namun memang dibutuhkan sejumlah usaha tambahan untuk membentuk proses transfer dan penerimaan data yang dapat diandalkan. Hubungan atau *wiring* dalam komunikasi serial ditunjukkan pada Gambar 2.13.

2.9 Neural Network

Neural network merupakan sebuah paradigma pemrosesan informasi yang terinspirasi oleh cara sistem saraf biologis seperti otak dalam memproses informasi. Bagian penting dari paradigma ini adalah struktur baru dari sistem pemrosesan informasi. *Neural network* terdiri atas sejumlah besar *neuron* yang saling terhubung yang bekerja sebagai satu kesatuan untuk memecahkan suatu masalah. Sama seperti manusia, *neural network* belajar berdasarkan contoh atau pengalaman. *Neural network* dapat diatur untuk sebuah aplikasi yang spesifik seperti *pattern recognition* atau *data classification* melalui proses *learning*.

2.9.1 Struktur dasar Neural Network

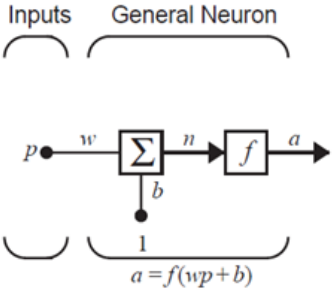
Gambar 2.14 menunjukkan sebuah *neuron* dengan input tunggal dimana struktur ini merupakan bagian paling sederhana dari sebuah *neural network*. Input *p* akan dikalikan dengan sebuah *weight* yang dilambangkan dengan *w* yang akan menghasilkan *wp*, yang merupakan salah satu input yang akan dimasukkan menuju ke sebuah *summer*. Input lainnya dari *summer* adalah angka 1 yang dikalikan dengan sebuah *bias* yang dilambangkan dengan *b* dan diinputkan menuju ke *summer*. *Summer* akan menjumlahkan seluruh input yang masuk kepadanya dan menghasilkan output yang dilambangkan sebagai *n*. Selanjutnya output dari summer yaitu *n* akan masuk menuju ke sebuah *transfer function* yang dilambangkan dengan *f*, yang akan menghasilkan output akhir dari *neuron* yang dilambangkan sebagai *a*.

Diagram diatas dapat ditulis menjadi persamaan matematis sebagai

$$a = f(wp + b) \tag{2.1}$$

dimana *a* merupakan output dari *neuron*, *w* adalah *weight*, *p* adalah input dari *neuron*, *b* adalah *bias* dan *f* adalah *transfer function*.

Sebuah *weight* adalah bobot yang akan mempengaruhi nilai input yang akan masuk menuju ke *neuron*. Sedangkan *bias* hampir sama seperti *weight*, hanya saja ia selalu mempunyai input konstan yang bernilai 1. Dalam beberapa kasus, *bias* dapat dihilangkan, namun sebuah *neural network* dengan *bias* terbukti jauh lebih baik dengan *neural network* tanpa *bias* [20].

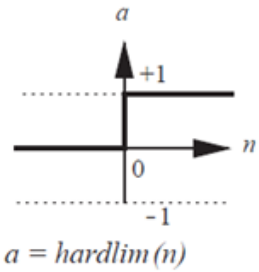


Gambar 2.14 Struktur dasar dari *neural network*.

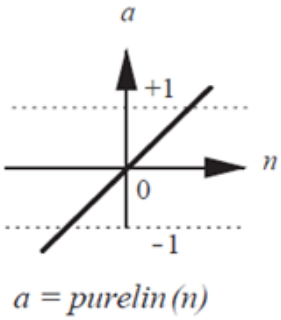
2.9.2 Transfer Function

Terdapat beberapa *transfer function* yang umum digunakan dalam *neural network* diantaranya adalah *hardlimit* TF, *linear* TF, *Log-sigmoid* TF dan *Tan-sigmoid* TF. Pemilihan jenis *transfer function* bergantung pada jenis permasalahan yang akan *neuron* coba untuk selesaikan.

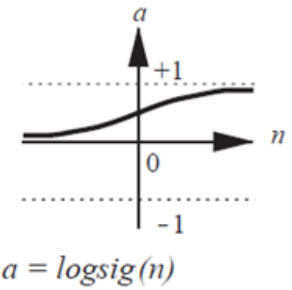
Hard-limit transfer function mempunyai karakteristik yaitu hanya mempunyai dua buah nilai output yaitu 0 dan 1. Jika input yang masuk ke *transfer function* bernilai kurang dari 0 maka output akan bernilai 0 sedangkan jika input bernilai lebih dari atau sama dengan nol maka output akan bernilai 1. *Transfer function* ini digunakan untuk membuat sebuah neuron yang akan mengklasifikasikan input ke dalam dua kategori yang berbeda.



Gambar 2.15 Grafik dari *Hard-limit transfer function*.



Gambar 2.16 Grafik dari *Linear transfer function*.



Gambar 2.17 Grafik dari *Log-sigmoid transfer function*.

Linear transfer function memiliki karakteristik yaitu input yang masuk ke *transfer function* adalah sama dengan output dari *transfer function.*

Log-sigmoid transfer function memiliki karakteristik yaitu dapat menerima input yang dapat bernilai dari -∞ hingga +∞ dan menyusutkan nilai tersebut sebagai output dengan rentang nilai antara 0 dan 1. Rumus matematika dari *Log-sigmoid transfer function* adalah sebagai berikut :

$$a = \frac{1}{1+ e^{n}} \qquad (2.2)$$

Log-sigmoid transfer function umumnya digunakan dalam *multilayer networks* yang dilatih dengan menggunakan algoritma *backpropagation*, dikarenakan fungsi mempunyai turunan yang unik.

Tabel 2.5 Rangkuman beberapa jenis *Transfer Function.*

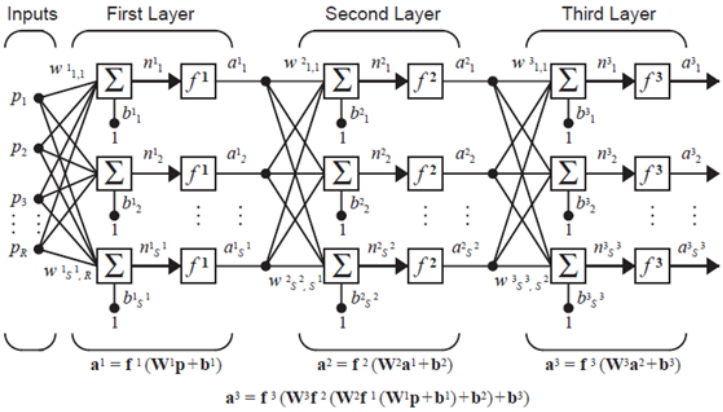
Jenis Transfer Function	Hubungan Input(n) terhadap Output(a)
1. Hard Limit	$a = \begin{cases} 0, & n < 0 \\ 1, & n \geq 0 \end{cases}$
2. Linear	$a = n$
3. Log-simoid	$a = \frac{1}{1 + e^{-n}}$
4. Tangent-sigmoid	$a = \frac{e^{n} - e^{-n}}{e^{n} + e^{-n}}$

2.9.3 Multilayer Perceptron

Sebuah *multilayer perceptron* (MLP) adalah sebuah model *neural network* yang bersifat *feedforward* yang memetakan sekumpulan data input menuju ke sekumpulan output yang sesuai. *Multilayer perceptron* terdiri atas beberapa lapisan atau *layers* dari *neuron* dimana setiap *neuron* dari *layer* sebelumnya akan saling berhubungan dengan *neuron* pada *layer* selanjutnya. *Multilayer perceptron* menggunakan teknik *supervised learning* berupa *backpropagation* untuk melakukan *training* atau pelatihan. MLP merupakan modifikasi dari *perceptron* yang tidak dapat mengenali data yang tidak dapat dipisahkan secara linear.

Sebuah contoh dari *multilayer perceptron* ditunjukkan pada Gambar 2.18 dimana terdapat 3 *layer perceptron* yang disusun secara seri. Output dari *layer* pertama menjadi input bagi *layer* kedua, dan output dari *layer* kedua menjadi *input* bagi layer ketiga. Dalam *multilayer perceptron* setiap *layer* dapat memiliki jumlah *neuron* yang berbeda, dan juga memiliki *transfer function* yang berbeda.

Untuk mengidentifikasi struktrur dari sebuah *multilayer perceptron* dapat digunakan sebuah notasi yang sederhana dimana jumlah input diikuti oleh jumlah *neuron* pada setiap *layer* :



Gambar 2.18 Struktur dari *Multilayer Perceptron*.

$$R - S^1 - S^2 - S^3$$

dimana R adalah jumlah input dari *neural network*, S^1 adalah jumlah *neuron* pada *layer* pertama, S^2 adalah jumlah *neuron* pada *layer* kedua dan S^3 adalah jumlah *neuron* pada *layer* ketiga.

2.9.4 Algoritma Backpropagation

Tujuan utama dalam pengembangan model *neural network* adalah untuk mencari sebuah himpunan optimal dari parameter *weight* dan *bias* sehingga fungsi *dari neural network* dapat mendekati atau mewakili perilaku dari permasalahan yang aslinya. Hal ini dapat dilakukan melalui proses yang disebut sebagai *training*. Pada algoritma *backpropagation* sekumpulan data *training* disediakan untuk *neural network*. Data *training* adalah berupa pasangan dari

$$\{p_1, t_1\}, \{p_2, t_2\}, \dots, \{p_Q, t_Q\}$$

dimana p_Q adalah input dari *neural network* dan t_Q merupakan target atau output yang diinginkan dari *neural network*.

Selama *training*, performa dari *neural network* dievaluasi dengan menghitung selisih antara output yang dihasilkan oleh *neural network* dengan output yang diinginkan untuk semua sampel dari data *training*. Nilai selisih ini dinyatakan sebagai error dan dapat dirumuskan sebagai

$$J = \frac{1}{2} \sum_{j=1}^{i} (t_j - a_j)^2 \qquad (2.3)$$

atau yang sering dikenal sebagai *Mean Squared Error* (MSE).

Tahap-tahap dalam algoritma *backpropagation* adalah sebagai berikut :

1 Forward Propagation

Forward propagation adalah tahap dimana sinyal yang terdapat pada *layer* input diteruskan sampai menuju ke *layer* output. Proses ini diawali dengan menginisialisasi nilai *weight* dan *bias* pada setiap *neuron*. Umumnya pemilihan nilai *weight* dan *bias* ini adalah berupa nilai acak antara 0 sampai 1.

Proses yang terjadi pada setiap *neuron* adalah sebagai berikut

$$n_i = \sum_{j=1}^{k} w_{ij}\, p_j + b_j \qquad (2.4)$$

Kemudian nilai n akan dimasukkan menuju ke *transfer function* yang menghasilkan output dari *neuron*

$$a = \frac{1}{1+ e^{-n_i}} \quad (2.5)$$

Setelah *forward propagation* selesai maka didapatkan output dari *neural network*. Langkah selanjutnya adalah menghitung *local gradient* pada masing-masing *neuron*. *Log-sigmoid transfer function* mempunyai turunan sebagai berikut

$$\dot{f}(n) = f(n)[1 - f(n)] \quad (2.6)$$

sehingga nilai *gradient* dari *neuron* dapat dihitung dengan rumus

$$\delta_j = \delta_{inj} * \dot{f}(n) \quad (2.7)$$

Tahap terakhir dalam algoritma *backpropagation* adalah melakukan *update* nilai *weight* dan *bias*.

$$w(n+1) = w(n) + \varphi * \delta(n) * y \quad (2.8)$$

$$b(n+1) = b(n) + \varphi * \delta(n) * 1 \quad (2.9)$$

BAB III
PERANCANGAN SISTEM

Pada bab perancangan sistem akan dijelaskan mengenai perancangan sistem secara menyeluruh. Perancangan sistem dapat dibagi ke dalam dua bagian yaitu perancangan *hardware* dan juga *software*. Perancangan *hardware* dimulai dengan perancangan modul *electronic nose* dimana ketiga buah sensor gas yang digunakan akan diubah ke dalam konfigurasi *array* dalam satu buah papan sirkuit. Selanjutnya perancangan *hardware* dilanjutkan dengan mengatur koneksi dari sensor warna menuju ke mikrokontroller. Perancangan *hardware* juga meliputi desain ruang sensor atau ruang uji dan desain kelistrikan yang meliputi sumber tegangan yang dibutuhkan oleh sistem.

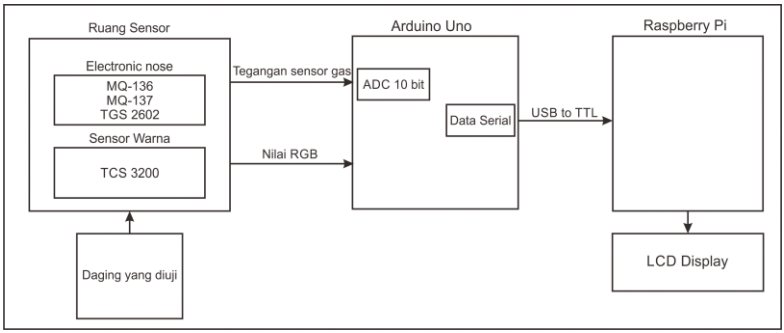
Perancangan *software* meliputi pemrograman mikrokontroller Arduino dan perancangan GUI dan *neural network* pada Raspberry Pi. Perancangan GUI dan *neural network* dilakukan menggunakan *software* Lazarus pada sistem operasi (OS) Raspbian.

3.1 Diagram Blok Sistem

Secara umum sistem yang dirancang dapat dibagi ke dalam dua buah bagian yaitu *hardware* dan juga *software*. Sistem *hardware* terdiri atas modul *electronic nose*, mikrokontroller Arduino Uno, Raspberry Pi, USB to TTL, sensor warna, dan konstruksi ruang sensor. Untuk sistem *software* terdiri atas program pembacaan ADC dan sensor warna pada Arduino dan program *neural network* menggunakan Lazarus pada Raspberry Pi.

Diagram blok dari sistem secara keseluruhan ditunjukkan pada Gambar 3.1. Dari diagram blok sistem, dapat dilihat bahwa cara kerja dari sistem ini adalah dengan mendeteksi tingkat kesegaran daging dengan menggunakan *electronic nose* dan sensor warna yang berada pada ruang sensor. *Electronic nose* terdiri atas *array* dari tiga buah sensor gas yang berbeda yaitu MQ-136, MQ-137 dan TGS 2602.

Ketiga buah sensor gas akan dipaparkan terhadap aroma yang dikeluarkan oleh daging yang diuji. Sensor gas kemudian akan merespon aroma dari daging dengan menghasilkan nilai tegangan yang berbeda-beda bergantung terhadap tingkat kesegaran daging yang diuji. Nilai tegangan dari ketiga buah sensor dalam *electronic nose* akan dibaca melalui ADC dari mikrokontroller Arduino.



Gambar 3.1 Diagram blok dari sistem.

Sedangkan sensor warna akan merespon warna dari daging dengan mengambil nilai RGB dari warna daging yang diuji.

Data berupa tegangan ketiga buah sensor dan nilai RGB dari sensor warna selanjutnya akan dikirim ke Raspberry Pi melalui komunikasi serial. Data yang diterima akan dipecah pada *software* Lazarus menjadi input dari *neural network*. Setelah data masuk maka data akan diolah terlebih dahulu sebelum proses pendeteksian secara *online* dilakukan.

3.2 Perancangan *Electronic Nose*

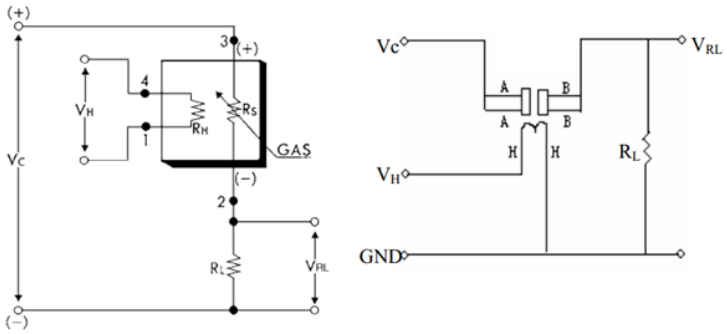
Pada perancangan *electronic nose* digunakan 3 buah *array* dari sensor gas tipe MOS yaitu MQ-136, MQ-137 dan TGS 2602. Pemilihan ketiga buah sensor ini dilandaskan pada studi literatur yang telah dilakukan. Berdasarkan *paper* yang berjudul "Detection of Meat Fresh Degree Based on Neural Network" disebutkan bahwa saat daging membusuk akan dihasilkan gas seperti NH_3 dan H_2S. Selanjutnya disebutkan bahwa agar dapat mengukur konsentrasi gas yang dihasilkan pada saat pembusukan daging, kita dapat menggunakan sensor gas yang juga berguna untuk mengidentifikasi dan menganalisa tingkat kesegaran daging.

Pada *paper* tersebut dilakukan penggunaan sensor berupa sensor gas MQ-136 dan MQ-137 yang masing-masing dapat mendeteksi jenis gas H_2S dan NH_3. Atas dasar ini maka *electronic nose* yang dirancang pada tugas akhir ini juga menggunakan sensor MQ-136 dan MQ-137 dengan tambahan sensor gas TGS 2602.

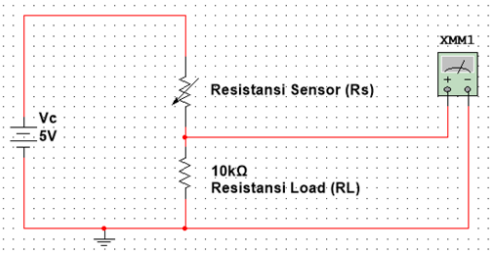
Seperti yang telah dijelaskan sebelumnya, sensor gas memiliki nilai resistansi yang berbeda-beda apabila dipaparkan terhadap jenis gas yang berbeda pula. Nilai resistansi sensor gas ini juga bergantung pada besarnya konsentrasi gas tersebut di udara.

Pada udara bersih, sensor akan memiliki nilai resistansi yang cukup tinggi jika dibandingkan dengan resistansi beban. Sedangkan jika sensor terpapar gas yang terdeteksi maka nilai resistansi sensor akan turun bergantung pada besarnya konsentrasi gas tersebut.

Korelasi antara nilai resistansi sensor terhadap jenis dan konsentrasi gas dapat dimanfaatkan untuk mendapatkan informasi mengenai tingkat kesegaran daging yang akan diuji. Secara teori daging yang masih segar akan menghasilkan respon sensor yang berbeda dengan daging yang sudah mulai membusuk. Dengan menggunakan prinsip pembagi tegangan pada setiap sensor gas maka tingkat kesegaran daging dapat diketahui.



Gambar 3.2 Rangkaian dasar sensor TGS dan MQ.



Gambar 3.3 Prinsip pembagi tegangan pada sensor gas.

Nilai resistansi beban (R_L) bernilai tetap sedangkan nilai resistansi sensor akan berubah-ubah bergantung pada jenis dan konsentrasi gas. Besarnya nilai tegangan pada resistansi beban adalah

$$V_L = \frac{R_L}{R_S + R_L} \, x \, V_C \qquad (3.1)$$

dimana V_L adalah tegangan dari resistor beban yang diambil oleh ADC, R_L adalah resistansi *load*, R_S adalah resistansi dari sensor dan V_C adalah tegangan input sensor yang bernilai 5 volt.

Pemilihan nilai resistansi beban R_L akan mempengaruhi nilai tegangan masing-masing sensor pada saat udara bersih. Pada perancangan *electronic nose* dalam tugas akhir ini dibutuhkan nilai tegangan sensor yang kecil pada udara bersih. Hal ini dilakukan untuk mencegah sensor mengalami saturasi yang terlalu cepat saat mendeteksi gas yang dihasilkan dari pembusukan daging. Tabel 3.1 menyajikan rentang nilai resistansi beban yang diperbolehkan untuk masing-masing sensor gas.

Pada Tabel 3.2 terlihat bahwa untuk nilai R_L yang sedemikian rupa, nilai tegangan untuk masing-masing sensor relatif kecil sehingga dapat digunakan dalam perancangan ini.

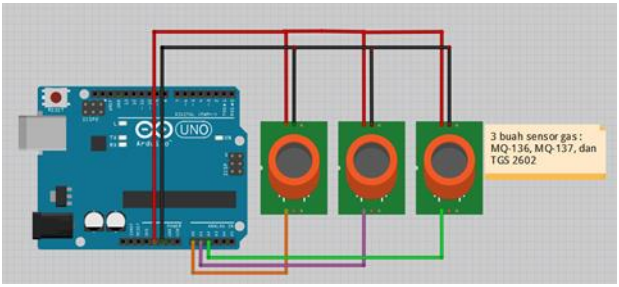
Tabel 3.1 Karakteristik sensor gas berdasarkan *datasheet*.

Jenis Sensor	R_L	R_S	V_C	V_H
MQ-136	10 KΩ - 47 KΩ	30 KΩ - 200 KΩ	5 V	5 V
MQ-137	10 KΩ - 100 KΩ	900 KΩ - 4900 KΩ	5 V	5 V
TGS 2602	> 0.45 KΩ	10 KΩ - 100 KΩ	5 V	5 V

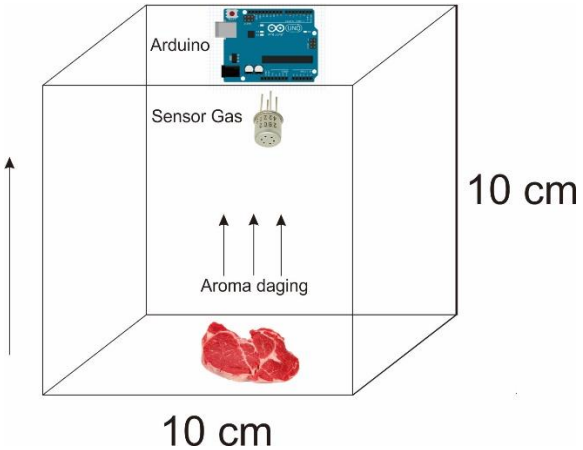
Tabel 3.2 Hubungan antara R_L dan tegangan sensor.

Jenis Sensor	R_L	**Tegangan Pada Udara Bersih**
MQ-136	22 KΩ	2.30 V
MQ-137	10 KΩ	1.95 V
TGS 2602	10 KΩ	0.96 V

Gambar 3.4 Modul *electronic nose* dalam satu papan sikuit.



Gambar 3.5 Skematik rangkaian sensor gas menuju Arduino.



Gambar 3.6 Perancangan konstruksi ruang sensor.

Perancangan *electronic nose* tidak hanya berfokus pada perancangan *hardware* saja. *Electronic nose* yang dirancang juga akan dihubungkan dengan Arduino untuk mengambil informasi berupa tegangan dari ketiga buah sensor gas. Proses ini melibatkan pemrograman mikrokontroller pada Arduino. Hal yang pertama dilakukan adalah mengakses ADC dari Arduino, karena terdapat tiga buah sensor gas, maka diperlukan tiga buah *port* ADC untuk membaca nilai tegangan dari ketiga buah sensor gas.

Skematik rangkaian *electronic nose* menuju ke Arduino ditunjukkan pada Gambar 3.5, dimana digunakan 3 buah *port* ADC yaitu *port* A0, A1 dan A2 untuk masing-masing sensor MQ-136 , MQ-137 dan TGS 2602. Pada program Arduino hal ini dapat dilakukan sebagai berikut

```
int rMQ_136 = analogRead(A0);
int rMQ_137 = analogRead(A1);
int rTGS_2602 = analogRead(A2);
```

Program diatas akan meginisialisasi *setting* ADC pada Arduino dengan nama variabel yang diinginkan. ADC membutuhkan tegangan referensi, pada Arduino nilai tegangan referensi ADC umumnya bernilai sebesar 5 v. Untuk mengubah nilai digital kembali ke nilai analog berupa tegangan maka dapat digunakan rumus sebagai berikut

$$ADC = \frac{V_{input}\, x\, 1023}{V_{referensi}} \tag{3.2}$$

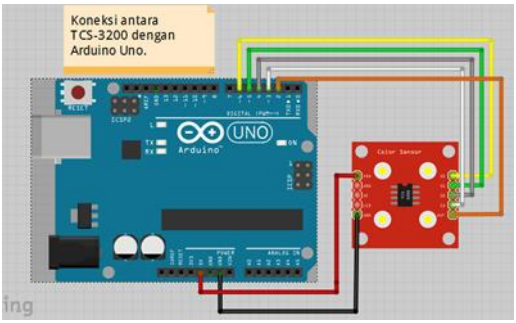
Dalam program, proses ini dilakukan melalui sintaks berikut

```
float MQ_136 = rMQ_136 * (5.00/1023.00);
float MQ_137 = rMQ_137* (5.00/1023.00);
float TGS_2602 = rTGS_2602* (5.00/1023.00);
```

3.3 Perancangan Sensor Warna

Sensor warna TCS 3200 digunakan untuk mengambil data berupa karakteristik warna dari daging yang sedang diuji. Terdapat empat buah filter warna photodioda yaitu *red*, *green*, *blue*, dan *clear*. Output dari sensor ini berupa gelombang kotak yang frekuensinya akan bervariasi terhadap warna yang terdeteksi oleh photodioda.

Proses pengambilan data warna dari daging dilakukan dengan mendekatkan sensor warna menuju ke permukaan daging yang akan diuji sehingga warna dari permukaan daging akan ditangkap oleh photodioda dari sensor warna.



Gambar 3.7 Hubungan antara Arduino dengan sensor warna.

Untuk mendapatkan nilai RGB, sensor warna harus dihubungkan menuju ke mikrokontroller Arduino. Hal ini membutuhkan *wiring* antara sensor warna menuju ke Arduino. Hubungan *wiring* ditunjukkan pada Gambar 3.7.

Hubungan pin antara sensor warna dengan Arduino dilakukan dengan mengaturnya pada program

```
// Init TSC230 and setting Frequency.
void TSC_Init()
{
  pinMode(S0, OUTPUT);
  pinMode(S1, OUTPUT);
  pinMode(S2, OUTPUT);
  pinMode(S3, OUTPUT);
  pinMode(OUT, INPUT);
  digitalWrite(S0, LOW);
  digitalWrite(S1, HIGH);
}
```

Setelah itu untuk menentukan filter warna yang akan digunakan kita harus mengatur nilai pin S2 dan S3 pada sensor warna sebagai berikut

```
// Select the filter color
void TSC_FilterColor(int Level01, int Level02)
{
  if(Level01 != 0)
    Level01 = HIGH;
  if(Level02 != 0)
    Level02 = HIGH;
  digitalWrite(S2, Level01);
```

```
  digitalWrite(S3, Level02);
}
```

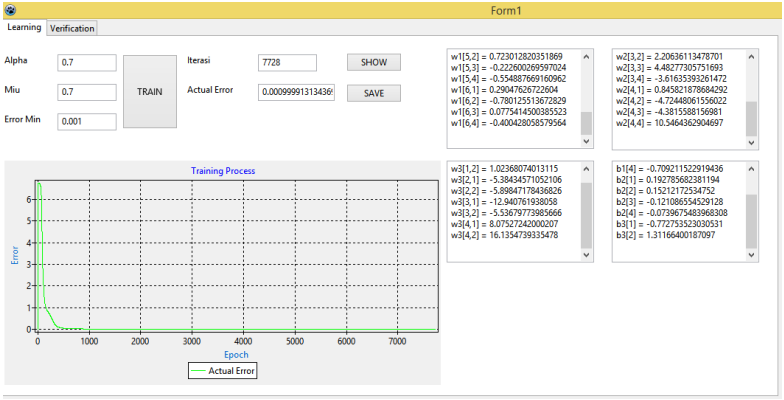
3.4 Perancangan Graphical User Interface

Perancangan GUI dilakukan pada *software* Lazarus, dimana tampilan GUI ini merupakan bantuan untuk menjalankan proses identifikasi kesegaran daging. Tampilan GUI terdiri atas dua buah bagian. Bagian pertama merupakan tampilan GUI untuk proses *training* dari *neural network* yang ditunjukkan pada Gambar 3.8.

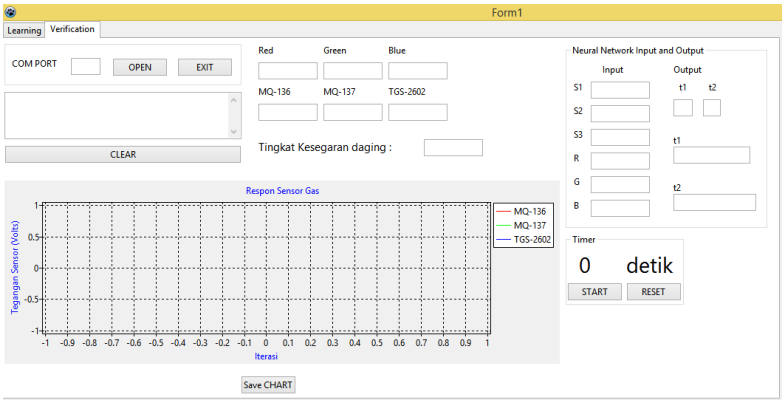
Pada tampilan GUI bagian pertama terlihat beberapa tombol yang dapat diklik untuk menjalankan program. Langkah pertama dalam proses *training* adalah memasukkan nilai Alpha, Miu dan Error Minimum pada masing-masing *box* yang telah disediakan. Setelah parameter dari proses *training* dilengkapi maka proses *training* dapat dimulai dengan mengklik tombol TRAIN pada GUI.

Setelah tombol TRAIN diklik maka proses *training* akan berjalan. Grafik yang terdapat pada tampilan ini berfungsi untuk menampilkan nilai error saat ini terhadap jumlah iterasi yang telah dilalui.

Setelah proses *training* selesai maka nilai *weight* dan *bias* yang terakhir dapat ditampilkan dengan mengklik tombol SHOW. Nilai *weight* dan *bias* hasil *training* ini juga dapat disimpan dalam data berbentuk format teks yaitu .txt dalam folder yang telah ditentukan.



Gambar 3.8 Tampilan GUI pada proses *training*.



Gambar 3.9 Tampilan GUI pada identifikasi *online*.

Pada tampilan GUI bagian kedua, ditampilkan proses pembacaan data dari sensor gas dan sensor warna serta proses identifikasi kesegaran daging secara *online*. Tampilan GUI ditampilkan pada Gambar 3.9.

Untuk dapat melakukan komunikasi serial dengan benar maka langkah pertama yang dilakukan adalah mengisi kotak pilihan dari com port. Hal ini dilakukan untuk memilih jalur mana yang akan digunakan untuk komunikasi serial. Setelah mengisi jalur komunikasi serial, langkah selanjutnya adalah membuka jalur komunikasi yang telah dibuat dengan mengklik tombol OPEN. Untuk menutup jalur komunikasi serial klik tombol CLOSE.

Apabila parameter komunikasi serial yang dibutuhkan telah berlangsung dengan benar, maka data serial yang dikirim oleh Arduino akan langsung diterima oleh GUI setelah tombol OPEN diklik. Informasi mengenai data dari masing-masing sensor akan ditampilkan pada masing-masing *box*.

Groupbox *timer* yang terdapat pada GUI akan memulai penghitungan selama 60 detik terhitung saat tombol START diklik. Saat detik ke 60 proses identifikasi akan dilaksanakan dan tingkat kesegaran daging yang diuji akan ditunjukkan oleh tampilan GUI.

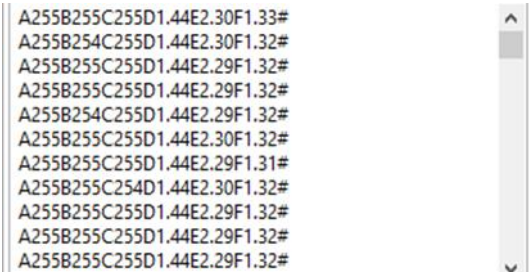
3.5 Perancangan Komunikasi Serial

Proses komunikasi serial antara Arduino dan Raspberry Pi didasarkan pada proses perancangan program. Pada tugas akhir ini, data

dari ketiga buah sensor gas dan data berupa nilai RGB dari sensor warna akan dikirim menuju Raspbery Pi dalam satu buah baris data. Oleh karena itu dibutuhkan suatu aturan pengiriman data yang dapat memisahkan dan menandai proses pengambilan data. Pada Arduino hal ini dilakukan sebagai berikut

```
Serial.print('A');
Serial.print(rgb[0]);
Serial.print('B');
Serial.print(rgb[1]);
Serial.print('C');
Serial.print(rgb[2]);
Serial.print('D');
Serial.print(MQ_136);
Serial.print('E');
Serial.print(MQ_137);
Serial.print('F');
Serial.print(TGS_2602);
Serial.println('#');
```

Data yang diterima pada Raspberry Pi akan memiliki format seperti yang ditampilkan pada Gambar 3.10. Nantinya data ini akan dipecah dengan menggunakan program pada *software* Lazarus sehingga menghasilkan data tunggal untuk masing-masing informasi dari sensor gas dan juga sensor warna.



Gambar 3.10 Format dari data serial.

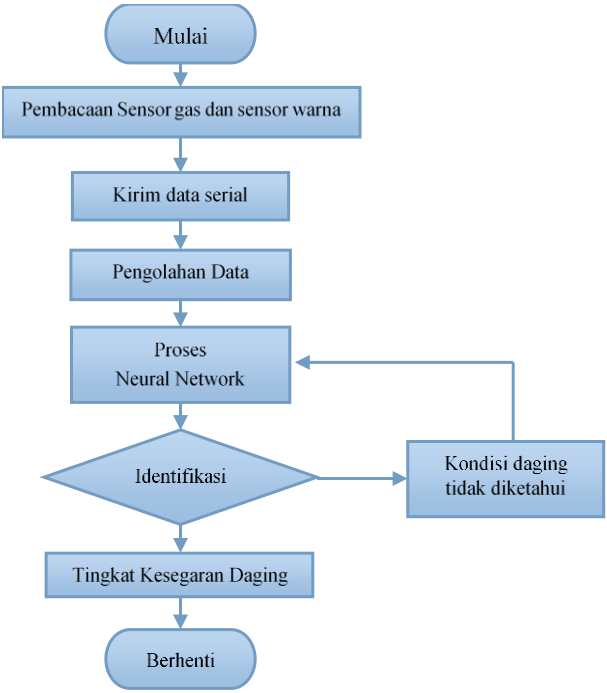
3.6 Perancangan Dan Realisasi Neural Network

Proses pengenalan tingkat kesegaran daging dilakukan dengan metode *neural network*. Parameter input dari *neural network* yang dirancang meliputi 6 buah data input yaitu nilai tegangan sensor gas MQ-136, MQ-137, TGS 2602 dan karakteristik warna daging berupa nilai *Red*, *Green* dan *Blue*. Keenam buah data ini berasal dari akuisisi data yang dilakukan oleh Arduino dan dikirimkan menuju ke program pada Raspberry Pi melalui komunikasi serial.

Struktur dari *neural network* yang dibangun dapat dilihat pada Gambar 3.12. *Neural network* memiliki 6 buah input yang berasal dari 3 buah sensor gas, dan nilai RGB. Terdapat 3 buah *layer* yang tersusun atas 2 buah *hidden layer* dan 1 buah *output layer*. Kedua buah *hidden layer* terdiri atas 4 buah *neuron* dan *output layer* terdiri atas 2 buah *neuron*.

Kombinasi dua buah nilai dari *neuron* pada output *layer* akan menjadi target dari pelatihan *neural network*. Dimana daging segar didefinisikan dengan nilai 00. Daging agak busuk didefinisikan dengan nilai 01. Dan daging busuk didefinisikan dengan nilai output yaitu 11.

Algoritma pemrogramaman *neural network* secara umum ditunjukkan oleh diagram alur sebagai berikut



Gambar 3.11 Diagram alur dari proses identifikasi pada ANN.

Langkah pertama dalam perancangan program *neural network* adalah inisialisasi *weight* dan *bias* seperti pada program berikut

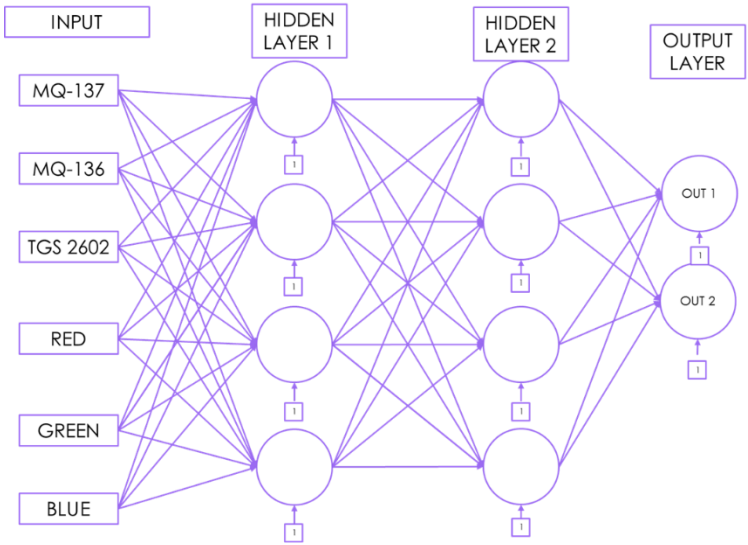
```
procedure TForm1.Weight_init;
var
  i, j : Integer;
begin
  Randomize;
  for i := 1 to 3 do
  begin
    for j := 1 to 4 do
    begin
      w1[i,j] := (Random * 2) - 1;    // Weight Layer 1
    end;
  end;
```

```
  for i := 1 to 4 do
  begin
    for j := 1 to 4 do
    begin
      w2[i,j] := (Random * 2) - 1;   // Weight Layer 2
    end;
  end;

  for i := 1 to 4 do
  begin
    for j := 1 to 2 do
    begin
      w3[i,j] := (Random * 2) - 1;   // Weight Layer 3
    end;
  end;

  for i := 1 to 4 do
  begin
    b1[i] := (Random * 2) - 1;       // Bias Layer 1
    b2[i] := (Random * 2) - 1;       // Bias Layer 2
  end;

  for i := 1 to 2 do
  begin
    b3[i] := (Random * 2) - 1;       // Bias Layer 3
  end;
end;
```

Pada baris program ini ini nilai *weight* dan *bias* ditentukan secara acak menggunakan pembangkit angka acak (*random generator*). Sintaks *random* pada program Lazarus akan menghasilkan angka acak antara 0 sampai dengan 1. Untuk mendapatkan nilai acak antara -1 dan 1 maka nilai acak tersebut dikalikan 2 dan kemudian dikurangkan lagi sebesar 1.

Untuk proses *forward propagation* dilakukan dengan mengalikan input dari ketiga buah sensor gas dan ketiga buah karaktreristik warna dari sensor warna terhadap masing-masing *weight* yang bersangkutan dengan *neuron* tersebut. Kemudian *bias* akan ditambahkan pada masing-masing *neuron*. Dalam program hal ini dapat dilakukan dengan cara berikut.

```
for i := 1 to 4 do   begin
    node1[i] := 0;
    for j := 1 to 6 do
    begin
      node1[i] := node1[i] + (w1[j,i] * input[z,j]);
    end;
    node1[i] := node1[i] + (b1[i] * 1);
    out1[i]  := 1 / (1 + exp(-1 * alfa * node1[i]));
    turunan1[i] := alfa * (out1[i]) * (1 - out1[i]);
  end;
```



Gambar 3.12 Struktur dari *neural network* yang dirancang.

BAB IV
PENGUJIAN DAN ANALISA

Bab ini akan membahas mengenai perancangan sistem yang telah dibuat dalam pengerjaan tugas akhir ini. Hal ini dilakukan untuk mengetahui apakah tujuan dari perancangan sistem telah terlaksana secara baik atau tidak. Pengujian dan pembahasan sistem ini dilakukan secara bertahap yaitu dengan menguji setiap blok dari sistem secara terpisah dan di akhir pengujian akan diuji sistem secara keseluruhan. Pengujian dan analisa sistem disertai dengan data berupa gambar, tabel dan grafik yang mendukung.

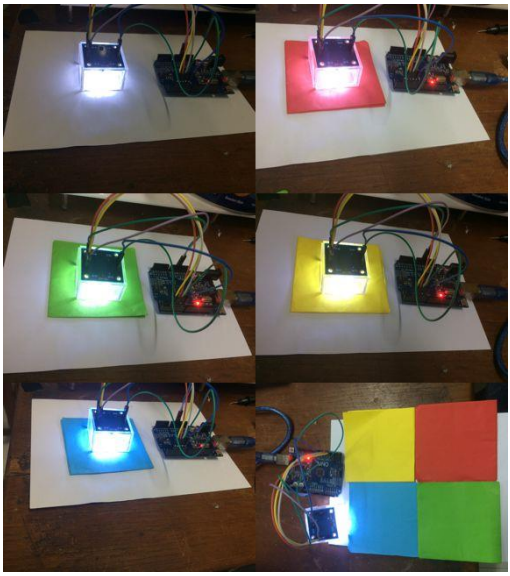
4.1 Pengujian Sensor Warna TCS 3200

Pengujian sensor warna dilakukan untuk mengetahui apakah sensor ini dapat mengambil data berupa karakteristik warna suatu objek dengan benar dan akurat. Untuk mengetahui respon sensor terhadap suatu warna, digunakan objek berupa kertas berwarna. Sensor warna TCS 3200 akan didekatkan pada permukaan kertas berwarna untuk mendapatkan karakteristik warna dari masing-masing kertas berwarna seperti yang ditunjukkan pada Gambar 4.1.

Sensor TCS 3200 akan mengeluarkan output berupa nilai RGB dari objek yang diarahkan pada daerah pendeteksiannya (*array photodioda*). Kertas berwarna yang digunakan untuk pengujian memiliki warna putih, merah, biru, kuning, dan hijau. Untuk dapat melakukan pengujian warna, sensor harus terlebih dahulu dikalibrasi dengan cara meletakkan sensor pada objek berwarna putih (dalam hal ini kertas putih).

Tabel 4.1 Nilai RGB yang didapatkan dari kertas warna.

Warna	***Red***	***Green***	***Blue***
Putih	255	255	255
Biru	78	143	175
Merah	179	49	49
Hijau	92	132	60
Kuning	255	199	80



Gambar 4.1 Pengujian TCS 3200 terhadap kertas warna.

Data yang didapatkan dari hasil pengujian ini diberikan pada Tabel 4.1. Nilai RGB yang didapatkan oleh sensor warna untuk masing-masing kertas berwarna kemudian dapat dibandingkan dengan sebuah aplikasi pemilih warna (*color picker*) pada komputer untuk dijadikan sebagai referensi dalam perbandingan ini.

Pada Tabel 4.2 dapat dilihat bahwa nilai RGB yang dihasilkan oleh sensor warna apabila dibandingkan dengan warna kertas aslinya adalah hampir sama. Perbandingan antara warna visual yang disajikan pada Tabel 4.2 dengan warna dari objek yaitu kertas berwarna pada Gambar 4.1 mengindikasikan bahwa sensor warna sangat akurat dalam menentukan nilai RGB dari suatu objek.

Tabel 4.2 Perbandingan nilai RGB dengan warna asli.

Warna	Nilai RGB	Visual
Biru	(78, 143, 175)	
Merah	(179, 49, 49)	
Hijau	(92, 132, 60)	
Kuning	(255, 199, 80)	

Hal ini tentunya akan sangat berguna dalam penerapannya pada pendeteksian tingkat kesegaran daging. Seperti yang telah dijelaskan pada bab sebelumnya, warna daging akan mengalami perubahan seiring dengan terjadinya proses pembusukan. Warna daging segar yang umumnya ditemukan di pasaran adalah berwarna merah cerah. Daging yang kurang segar akan memiliki warna merah yang agak gelap.

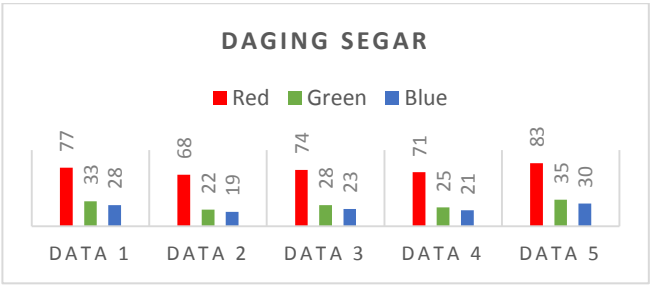
Pada tugas akhir ini definisi mengenai tingkat kesegaran daging ditentukan melalui waktu penyimpanan daging di luar ruangan pada suhu kamar. Definisi mengenai tingkat kesegaran daging dan hasil pengujian sensor warna terhadap ketiga buah sampel daging dengan tingkat kesegaran yang berbeda akan ditunjukkan oleh tabel berikut ini

Tabel 4.3 Definisi tingkat kesegaran daging yang diuji.

Tingkat Kesegaran Daging	Definisi Kesegaran
Segar	Daging yang baru saja disembelih / keluar daru freezer
Agak Busuk	Daging yang berada di luar ruangan pada suhu kamar selama ±12 jam.
Busuk	Daging yang berada di luar ruangan pada suhu kamar selama satu hari atau lebih.

Tabel 4.4 Pengujian sensor warna terhadap daging segar.

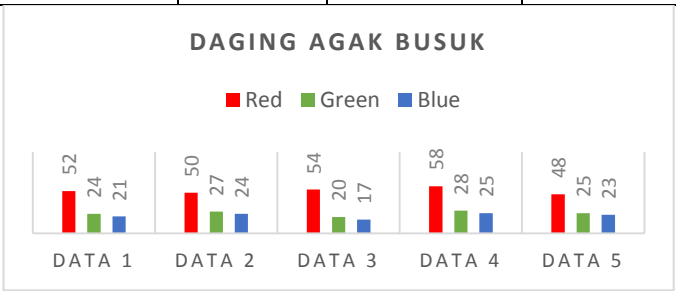
Daging segar	Red	Green	Blue
Data 1	77	33	28
Data 2	68	22	19
Data 3	74	28	23
Data 4	71	25	21
Data 5	83	35	30



Gambar 4.2 Pengujian sensor warna pada daging segar.

Tabel 4.5 Pengujian sensor warna terhadap daging agak busuk.

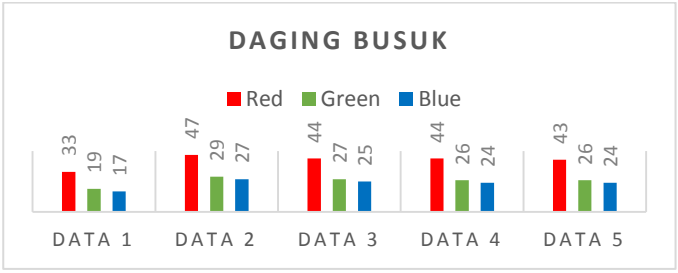
Daging agak busuk	Red	Green	Blue
Data 1	52	24	21
Data 2	50	27	24
Data 3	54	20	17
Data 4	58	28	25
Data 5	48	25	23



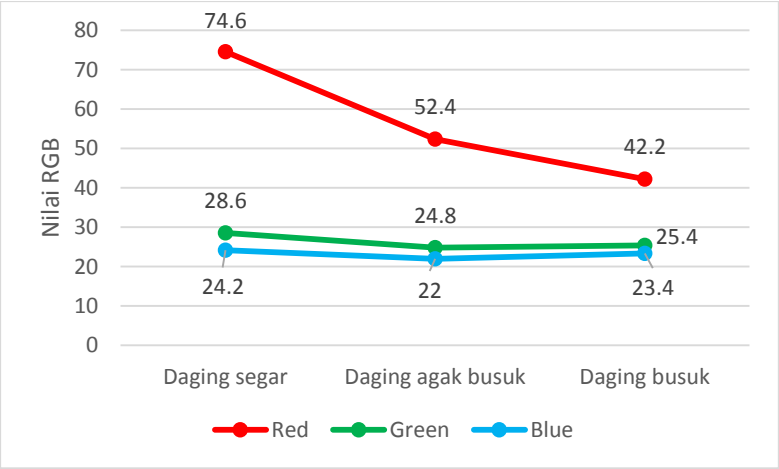
Gambar 4.3 Pengujian sensor warna pada daging agak busuk.

Tabel 4.6 Pengujian sensor warna terhadap daging busuk.

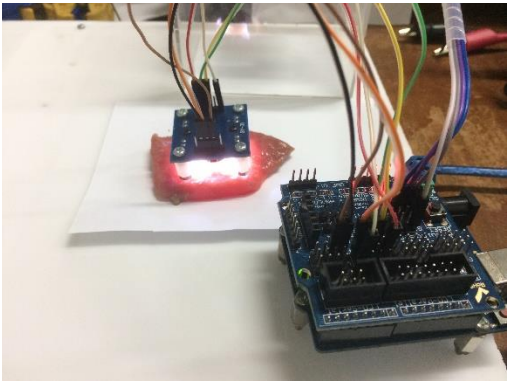
Daging busuk	Red	Green	Blue
Data 1	33	19	17
Data 2	47	29	27
Data 3	44	27	25
Data 4	44	26	24
Data 5	43	26	24



Gambar 4.4 Pengujian sensor warna pada daging busuk.



Gambar 4.5 Grafik warna daging terhadap tingkat kesegaran.



Gambar 4.6 Pengujian daging segar dengan sensor warna.

Dari data yang didapatkan, dapat diperoleh suatu pola dan hubungan antara tingkat kesegaran daging terhadap karakteristik warnanya. Daging segar memiliki nilai RGB tertinggi dibandingkan dengan dua buah sampel daging lainnya. Apabila nilai RGB yang didapatkan oleh sensor warna ini ditampilkan pada aplikasi pemilih warna maka akan terlihat jelas perbedaan tingkat kemerahan dari warna daging.

Dimana warna merah yang didapatkan akan semakin gelap apabila daging semakin tidak segar.

4.2 Pengujian ADC dan Komunikasi Serial

Pengujian ADC dan komunikasi serial dilakukan secara *real time* dengan menghubungkan ADC dari Arduino menuju ke sumber tegangan. Nilai tegangan yang masuk ke ADC akan dikirim melalui komunikasi serial menggunakan USB to TTL menuju ke Raspberry Pi. Pada Raspberry Pi, GUI akan menampilkan nilai tegangan yang dibaca oleh ADC dari Arduino. Sumber tegangan yang digunakan memiliki indikator tegangan tersendiri sehingga akan dijadikan referensi dalam pengujian ADC dan komunikasi serial.

Pengujian ini dilakukan dengan cara membandingkan nilai tegangan yang ditampilkan oleh sumber tegangan terhadap nilai pembacaan tegangan multimeter dan pembacaan ADC.

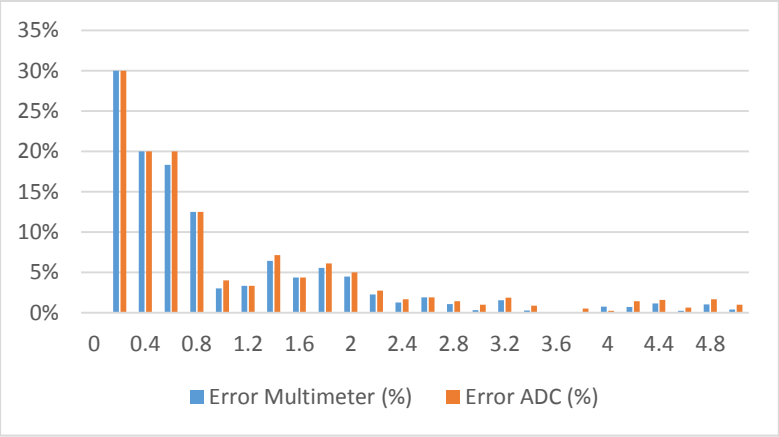
Tabel 4.7 Perbandingan pembacaan ADC dan komunikasi serial.

Input ke-	Sumber Tegangan DC (volt)	Pembacaan Multimeter (volt)	Pembacaan ADC (volt)	Error Multimeter (%)	Error ADC (%)
1	0.0 v	0 v	0 v	0 %	0 %
2	0.2 v	0.14 v	0.14 v	30 %	30 %
3	0.4 v	0.32 v	0.32 v	20 %	20 %
4	0.6 v	0.49 v	0.48 v	18.33 %	20 %
5	0.8 v	0.7 v	0.7 v	12.5 %	12.5 %
6	1.0 v	0.97 v	0.96 v	3 %	4 %
7	1.2 v	1.16 v	1.16 v	3.33 %	3.33 %
8	1.4 v	1.31 v	1.30 v	6.42 %	7.14 %
9	1.6 v	1.53 v	1.53 v	4.375 %	4.375 %
10	1.8 v	1.70 v	1.69 v	5.55 %	6.11 %
11	2.0 v	1.91 v	1.90 v	4.5 %	5 %
12	2.2 v	2.15 v	2.14 v	2.27 %	2.72 %
13	2.4 v	2.37 v	2.36 v	1.25 %	1.66 %
14	2.6	2.55 v	2.55 v	1.92 %	1.92 %
15	2.8 v	2.77 v	2.76 v	1.07 %	1.42 %
16	3.0 v	2.99 v	2.97 v	0.333 %	1 %
17	3.2 v	3.15 v	3.14 v	1.56 %	1.875 %
18	3.4 v	3.39 v	3.37 v	0.294 %	0.882 %
19	3.6 v	3.60 v	3.58 v	0 %	0.055 %

Input ke-	Sumber Tegangan DC (volt)	Pembacaan Multimeter (volt)	Pembacaan ADC (volt)	Error Multimeter (%)	Error ADC (%)
20	3.8 v	3.80 v	3.78 v	0 %	0.526 %
21	4.0 v	4.03 v	4.01 v	0.75 %	0.25 %
22	4.2 v	4.17 v	4.14 v	0.714 %	1.42 %
23	4.4 v	4.35 v	4.33 v	1.136 %	1.59 %
24	4.6 v	4.59 v	4.57 v	0.217 %	0.652 %
25	4.8 v	4.75 v	4.72 v	1.04 %	1.667 %
26	5.0 v	4.98 v	4.95 v	0.4 %	1 %

Data hasil pengujian disajikan pada Tabel 4.6 dimana error rata-rata dari hasil pembacaan ADC adalah sebesar 5%. Error ini masih dapat dianggap dalam batas toleransi mengingat tegangan yang diberikan hanya berkisar antara 0 hingga 5 volt. Error pada pembacaan tegangan dengan nilai dua angka dibelakang koma akan memberikan error yang cukup signifikan apabila tegangan tersebut bernilai di bawah 1 volt.

Namun dikarenakan penggunaan ADC adalah untuk melakukan pembacaan nilai tegangan sensor gas yang biasanya berkisar antara 2 volt – 5 volt maka error pembacaan tidak akan berpengaruh begitu besar. Dengan demikian dapat disimpulkan bahwa pembacaan nilai ADC dari Arduino sudah cukup akurat dan proses komunikasi serial juga dapat berjalan dengan baik tanpa terjadinya error komunikasi.



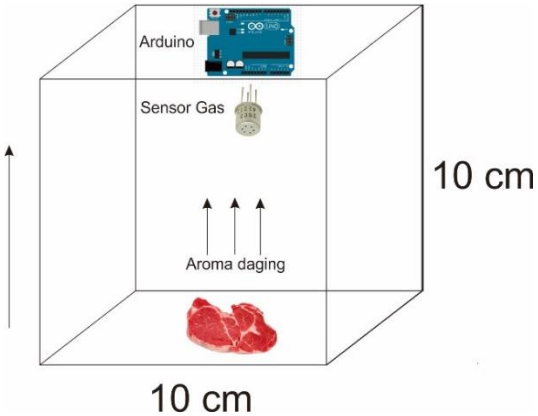
Gambar 4.7 Error hasil pembacaan nilai tegangan oleh ADC.

4.3 Pengujian Sensor Gas

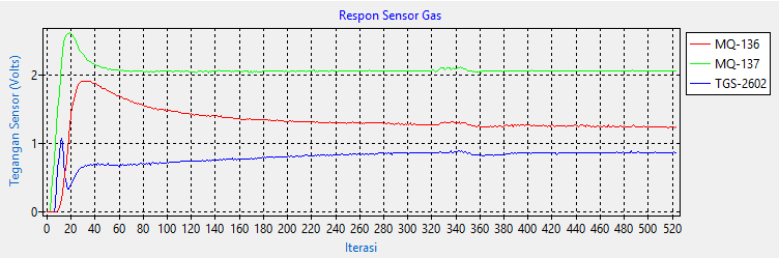
Pengujian sensor gas dilakukan dengan tujuan untuk mengetahui bagaimana sensor ini akan merespon terhadap perubahan tingkat kesegaran yang terjadi pada daging. Untuk menguji respon sensor gas terhadap sampel daging yang diuji diterapkan beberapa aturan yang akan secara konsisten dilakukan dalam setiap pengujian sebagai berikut :

1. Ruang uji adalah sebuah kubus berukuran 10 cm x 10 cm x 10 cm yang tertutup rapat dimana di dalamnya terdapat modul sensor gas.
2. Sensor gas berada di bagian atas dari ruang uji dan sampel daging berada di bawah saat pengujian berlangsung.
3. Pengambilan data tegangan sensor gas berlangsung selama 60 detik mulai dari saat daging memasuki ruang uji sensor.
4. Nilai yang diambil sebagai input dari *neural network* adalah nilai tegangan dari ketiga buah sensor saat detik ke 60.
5. Daging segar didefinisikan sebagai daging yang baru dikeluarkan dari kulkas. Daging agak busuk didefinisikan sebagai daging yang diletakkan diluar ruangan selama ±12 jam. Dan daging busuk adalah daging yang berada di ruang terbuka selama satu hari atau lebih.

Pengujian sensor gas yang pertama kali dilakukan adalah pengujian sensor gas saat tidak terdapat sampel daging pada ruang uji sensor atau dengan kata lain ruang sensor berisi udara bersih. Grafik dari respon sensor gas terhadap udara bersih ditunjukkan oleh Gambar 4.9.



Gambar 4.8 Konstruksi dari *sensor chamber* (ruang sensor).



Gambar 4.9 Grafik respon sensor gas dalam udara bersih.

Terlihat respon sensor ketika pertama kali dinyalakan adalah memiliki nilai tegangan yang naik secara tajam kemudian setelah beberapa saat ketiga buah sensor gas akan mengalami penurunan tegangan dan kemudian akan mencapai nilai *steady state* dalam kondisi udara bersih. Nilai tegangan sensor gas pada udara bersih ini dapat dijadikan sebagai sebuah referensi dari sistem yang dibuat.

Adapun nilai tegangan ketiga buah sensor gas dalam udara bersih disajikan pada Tabel 4.8. Berdasarkan pengujian sensor gas dalam udara bersih, nilai tegangan ini akan berubah-ubah dalam waktu yang cukup lama. Untuk membuat nilai tegangan sensor gas yang stabil pada udara bersih dibutuhkan waktu pemanasan sensor gas yang cukup lama. Hal ini diperlukan untuk meningkatkan kestabilan dalam pembacaan nilai tegangan sensor gas.

Pengujian sensor gas terhadap sampel daging dimulai dengan sampel berupa daging segar. Proses pengambilan data dilakukan dengan mengikuti aturan pengambilan data yang telah dijelaskan sebelumnya. Dari hasil pengujian didapatkan data yang ditunjukkan pada Tabel 4.9 – Tabel 4.11.

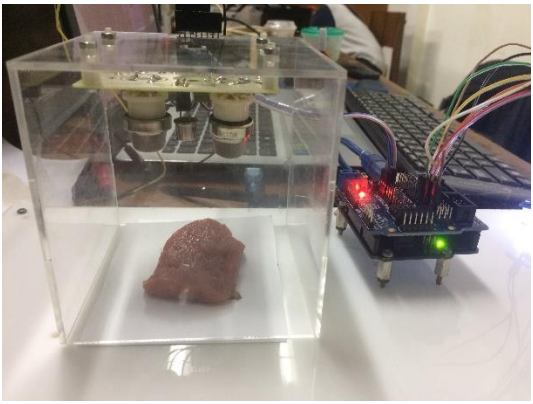
Tabel 4.8 Pengujian sensor gas pada udara bersih.

Udara Bersih	MQ-136	MQ-137	TGS 2602
Data 1	1.81 v	1.86 v	1.72 v
Data 2	1.72 v	1.89 v	1.75 v
Data 3	1.76 v	1.88 v	1.77 v
Data 4	1.86 v	2.06 v	1.74 v
Data 5	1.75 v	1.88 v	1.77 v

Tabel 4.9 Pengujian sensor gas pada daging segar.

Daging segar	MQ-136	MQ-137	TGS 2602
Data 1	2.18 v	2.06 v	2.09 v
Data 2	1.88 v	2.03 v	1.86 v
Data 3	1.94 v	2.06 v	1.95 v
Data 4	1.84 v	2.01 v	1.80 v
Data 5	1.83 v	2.01 v	1.82 v
Data 6	2.03 v	1.96 v	1.88 v
Data 7	1.96 v	2.0 v	1.93 v
Data 8	1.87 v	2.0 v	1.81 v
Data 9	1.85 v	2.02 v	1.74 v
Data 10	1.89 v	2.02 v	1.83 v

Nilai tegangan dari ketiga buah sensor gas ini membentuk sebuah pola yang merepresentasikan keadaan dari sebuah daging yang masih segar. Apabila dibandingkan dengan nilai tegangan sensor gas pada udara bersih, maka dapat dilihat bahwa tidak terdapat perbedaan yang signifikan. Hal ini disebabkan oleh kondisi daging segar yang tidak mengeluarkan bau yang menyengat.



Gambar 4.10 Pengujian sensor gas dengan sampel daging segar.

Tabel 4.10 Pengujian sensor gas pada daging agak busuk.

Daging agak busuk	MQ-136	MQ-137	TGS 2602
Data 1	2.96 v	2.43 v	2.30 v
Data 2	2.70 v	2.30 v	2.11 v
Data 3	2.67 v	2.42 v	2.09 v
Data 4	2.66 v	2.35 v	1.98 v
Data 5	2.63 v	2.26 v	2.10 v
Data 6	2.64 v	2.38 v	2.08 v
Data 7	2.59 v	2.41 v	2.22 v
Data 8	2.70 v	2.44 v	3.16 v
Data 9	2.58 v	2.43 v	2.27 v
Data 10	2.61 v	2.39 v	2.34 v

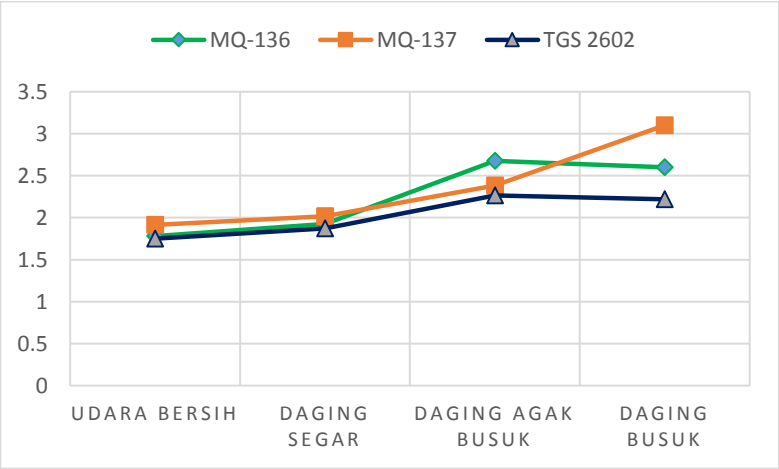
Selanjutnya pengujian sensor gas dilakukan dengan sampel berupa daging agak busuk. Kondisi daging pada keadaan ini adalah memiliki bau yang tidak sedap namun dengan aroma yang tidak terlalu pekat. Hasil pengujian sensor gas terhadap sampel daging agak busuk ditunjukkan pada Tabel 4.10.

Apabila dibandingkan dengan hasil pengujian sensor gas pada sampel daging segar, terdapat kenaikan nilai tegangan pada ketiga buah sensor. Hal ini mengindikasikan jika ketiga buah sensor tersebut merespon terhadap aroma yang dihasilkan oleh daging selama proses pembusukan daging.

Dari dua buah data berupa tegangan sensor gas pada dua buah kondisi daging yang berbeda dapat diambil kesimpulan awal jika penggunaan sensor gas sebagai pendeteksi tingkat kesegaran daging dapat diterapkan dengan menggunakan metode *neural network*. Hal ini disebabkan karena kondisi mengenai tingkat kesegaran daging memiliki pola yang berbeda yang diwakili oleh nilai tegangan dari ketiga buah sensor gas.

Tabel 4.11 Pengujian sensor gas pada daging busuk.

Daging busuk	MQ-136	MQ-137	TGS 2602
Data 1	2.67 v	3.04 v	2.17 v
Data 2	2.8 v	3.09 v	2.3 v
Data 3	2.6 v	3.14 v	2.22 v
Data 4	2.56 v	3.06 v	2.19 v
Data 5	2.52 v	3.14 v	2.23 v
Data 6	2.57 v	3.13 v	2.21 v
Data 7	2.56 v	3.1 v	2.22 v
Data 8	2.64 v	3.12 v	2.27 v
Data 9	2.47 v	3.06 v	2.17 v
Data 10	2.61 v	3.1 v	2.22 v



Gambar 4.11 Grafik respon sensor gas terhadap kesegaran daging.

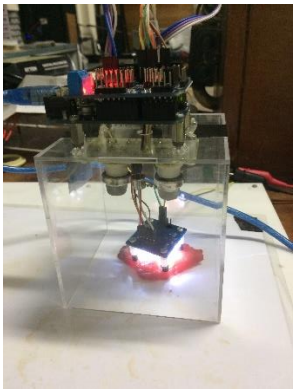
4.4 Pengujian Sensor Secara Keseluruhan

Pengujian sensor warna dan juga sensor gas pada sub bab sebelumnya merupakan pengujian secara terpisah terhadap ketiga buah sampel daging yang diuji. Pada sub bab ini pengujian sensor gas dan sensor warna terhadap sampel daging akan dilakukan secara bersamaan. Hal ini dilakukan untuk mencari tahu respon kedua sensor secara keseluruhan terhadap tingkat kesegaran daging yang diuji.

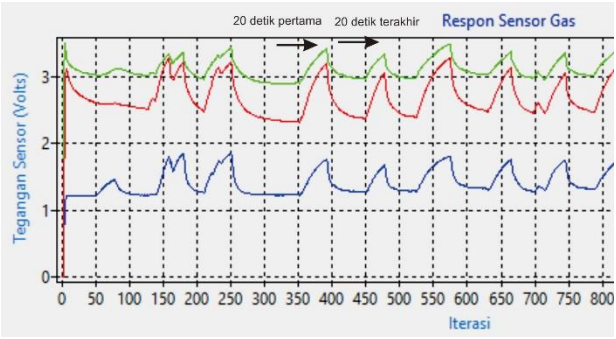
Pada pengujian sensor gas di sub bab sebelumnya digunakan nilai akhir dari tegangan sensor gas sebagai parameter input untuk *neural network*. Namun berdasarkan hasil pengujian, nilai tegangan sensor untuk setiap kondisi udara bersih adalah berbeda-beda. Hal ini menyebabkan tidak menentunya nilai tegangan sensor sebagai respon dari sampel daging yang diuji.

Sebagai contoh pada Tabel 4.10 dan 4.11, nilai tegangan dari sensor MQ-136 dan TGS 2602 memiliki rentang nilai yang hampir sama. Namun ternyata nilai tegangan pada kondisi udara bersih untuk kedua buah pengujian memiliki nilai yang berbeda. Hal ini tentunya akan mengakibatkan kesalahan pengenalan pola apabila data ini digunakan sebagai input dari *neural network*.

Untuk itu pada pengujian ini digunakan nilai *baseline* untuk pengolahan data pada sensor gas. Proses pengambilan data berlangsung selama 60 detik terhitung saat daging yang diuji memasuki ruang sensor. Untuk mendapatkan pola tegangan sensor gas untuk setiap sampel daging dengan tingkat kesegaran yang berbeda, digunakan nilai *baseline* dari nilai tegangan sensora gas.



Gambar 4.12 Desain sistem sensor secara keseluruhan.



Gambar 4.13 Metode *baseline* nilai tegangan dari respon sensor gas.

Algoritma dari pengambilan nilai tegangan sensor gas adalah sebagai berikut. Nilai tegangan ketiga buah sensor gas pada detik ke 1 hingga detik ke 20 akan dijumlahkan kemudian dicari nilai rata-ratanya. Nilai ini dapat disebut sebagai nilai rata-rata bawah V_{LOW}. Kemudian nilai tegangan sensor pada detik ke 41 hingga ke 60 juga akan diambil dan kemudian dilakukan proses rata-rata tegangan. Nilai ini dapat disebut sebagai nilai rata-rata atas V_{UP}. Selanjutnya nilai rata-rata atas akan dikurangkan dengan nilai rata-rata bawah yang akan menghasilkan selisih nilai rata-rata tegangan yang sebanding dengan tingkat kesegaran dari daging yang diuji.

$$V_{LOW} = \frac{\sum_{t=1}^{t=20} V_{Sensor}}{20} \quad (4.1)$$

$$V_{UP} = \frac{\sum_{t=41}^{t=60} V_{Sensor}}{20} \quad (4.2)$$

$$V_S = V_{UP} - V_{LOW} \quad (4.3)$$

Dari hasil selisih nilai tegangan ini didapatkan data untuk ketiga buah sampel daging dengan kondisi kesegaran yang berbeda.

Tabel 4.12 Pengujian selisih tegangan pada daging segar.

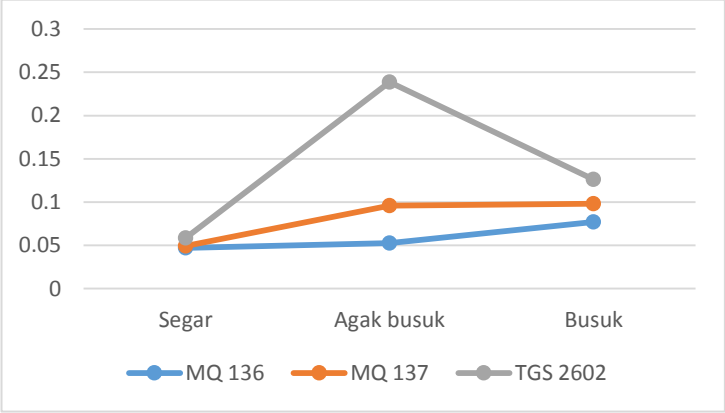
Segar						
No	**MQ 136**	**MQ 137**	**TGS 2602**	**Red**	**Green**	**Blue**
1	0.0295 v	0.0665 v	0.0395 v	82	27	24
2	0.045 v	0.053 v	5.60E-02 v	74	22	18
3	0.0465 v	0.053 v	0.033 v	78	26	22
4	0.053 v	0.044 v	7.25E-02 v	80	26	22
5	0.0355 v	0.0195 v	0.0685 v	80	25	21
6	0.0575 v	0.059 v	0.0755 v	83	26	22
7	0.057 v	0.058 v	0.057 v	79	25	22
8	0.0635 v	0.0435 v	0.0505 v	74	30	26
9	0.044 v	0.042 v	0.06 v	75	32	29
10	0.04 v	0.0555 v	0.074 v	81	32	28

Tabel 4.13 Pengujian selisih tegangan pada daging agak busuk.

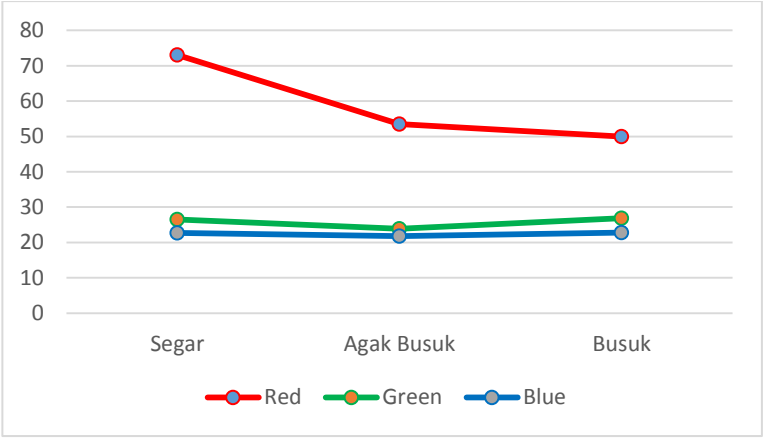
Agak Busuk						
No	**MQ 136**	**MQ 137**	**TGS 2602**	**Red**	**Green**	**Blue**
1	0.026 v	0.057 v	0.092 v	55	27	24
2	0.026 v	0.052 v	0.072 v	55	28	24
3	0.062 v	0.124 v	0.302 v	55	29	25
4	0.048 v	0.0255 v	0.214 v	54	27	23
5	0.0445 v	0.063 v	0.278 v	55	29	25
6	0.078 v	0.188 v	0.286 v	50	28	24
7	0.0535 v	0.1445 v	0.1855 v	54	28	24
8	0.054 v	0.098 v	0.292 v	51	27	24
9	0.082 v	0.128 v	0.392 v	48	25	21
10	0.0515 v	0.0805 v	0.2745 v	54	28	25

Tabel 4.14 Pengujian selisih tegangan pada daging busuk.

Busuk						
No	**MQ 136**	**MQ 137**	**TGS 2602**	**Red**	**Green**	**Blue**
1	0.069 v	0.1285 v	0.1105 v	30	16	16
2	0.0445 v	0.1225 v	1.02E-01 v	29	16	15
3	0.0785 v	0.0915 v	0.0715 v	29	16	15
4	0.08 v	0.102 v	1.32E-01 v	30	16	15
5	0.0705 v	0.1575 v	0.1285 v	29	16	15
6	0.098 v	0.108 v	0.11 v	31	17	16
7	0.088 v	0.116 v	0.204 v	31	17	17
8	0.092 v	0.082 v	0.148 v	25	14	13
9	0.078 v	0.043 v	0.1135 v	31	17	17
10	0.0715 v	0.031 v	0.1445 v	27	15	14



Gambar 4.14 Grafik selisih tegangan pada kesegaran daging.



Gambar 4.15 Grafik respon sensor warna terhadap kesegaran daging.

Dari data yang telah didapatkan dapat diamati jika nilai rata-rata tegangan terkecil berada pada pengujian daging segar. Kecilnya nilai rata-rata tegangan ini mengindikasikan jika sensor gas tidak terlalu merespon terhadap daging segar. Seperti yang telah dijelaskan sebelumnya bahwa daging segar tidak memiliki aroma menyengat sehingga sensor merespon sama seperti pada kondisi udara bersih.

Nilai rata-rata tegangan terbesar berada pada antara kondisi daging agak busuk menuju ke daging busuk. Hal ini disebabkan karena daging busuk memiliki aroma yang sangat menyengat sehingga sensor gas akan merespon dengan laju kenaikan tegangan yang cukup signifikan.

Dari semua pengujian ini dapat disimpulkan bahwa masing-masing tingkat kesegaran dari daging memiliki pola tertentu, meskipun pada kasus tertentu pola ini tidaklah berbeda begitu jauh antara satu dengan yang lainnya.

4.5 Pengujian Software Neural Network

Pengujian *software neural network* dilakukan sebagai persiapan untuk mengolah seluruh data yang telah didapatkan dari pegujian sensor warna dan juga sensor gas. Setelah semua data terkumpul, hal pertama yang dilakukan adalah melakukan proses normalisasi data terhadap masing-masing data dari sensor warna dan juga sensor gas.

Proses normalisasi data ini dilakukan dengan mengikuti rumus sebagai berikut

$$Y_{norm} = \frac{Y}{Y_{max}} \quad (4.4)$$

dimana Y_{norm} adalah data hasil normalisasi, Y adalah data sebenarnya yang akan dinormalisasi dan Y_{max} adalah nilai data sejenis yang terbesar. Setelah melalui proses normalisasi ini semua data akan memiliki rentang antara nilai nol dan satu.

Data hasil akuisisi data dari sensor gas dan sensor warna disajikan pada tabel berikut ini

Tabel 4.15 Data hasil normalisasi untuk daging segar.

Segar						
No	**MQ 136**	**MQ 137**	**TGS 2602**	**Red**	**Green**	**Blue**
1	0.44360	1	0.59398	1	0.32926	0.29268
2	0.80357	0.94642	1	1	0.2972	0.24324
3	0.87735	1	0.622641	1	0.3333	0.2820
4	0.7310	0.6068	1	1	0.325	0.275
5	0.51824	0.2846	1	1	0.3125	0.2625
6	0.761	0.781	1	1	0.31325	0.2650
7	0.98275	1	0.98275	1	0.31645	0.27848
8	1	0.68503	0.79527	1	0.40540	0.35135
9	0.73333	0.7	1	1	0.42666	0.38666
10	0.54054	0.75	1	1	0.3950	0.3456

Tabel 4.16 Data hasil normalisasi untuk daging agak busuk.

Agak Busuk						
No	**MQ 136**	**MQ 137**	**TGS 2602**	**Red**	**Green**	**Blue**
1	0.282608	0.619565	1	1	0.4909090	0.436363
2	0.361111	0.722222	1	1	0.509090	0.436363
3	0.205298	0.41059	1	1	0.52727	0.454545
4	0.224299	0.119158	1	1	0.5	0.425925
5	0.160071	0.226618	1	1	0.527272	0.454545
6	0.272727	0.657342	1	1	0.56	0.48
7	0.288409	0.778975	1	1	0.51851	0.444444
8	0.1849315	0.335616	1	1	0.529411	0.470588
9	0.20918	0.32653	1	1	0.52083	0.4375
10	0.18761	0.29326	1	1	0.51851	0.462962

Tabel 4.17 Data hasil normalisasi untuk daging busuk.

Busuk						
No	**MQ 136**	**MQ 137**	**TGS 2602**	**Red**	**Green**	**Blue**
1	0.536964	1	0.85992	1	0.53333	0.53333
2	0.36326	1	0.82857	1	0.551724	0.517241
3	0.85792	1	0.78142	1	0.55172	0.5172
4	0.60606	0.7727	1	1	0.53333	0.5
5	0.44761	1	0.8158	1	0.551724	0.517241
6	0.890909	0.981818	1	1	0.548387	0.51612
7	0.43137	0.56862	1	1	0.54838	0.548387
8	0.621621	0.55405	1	1	0.56	0.52
9	0.6872	0.37885	1	1	0.54838	0.548387
10	0.4948096	0.214532	1	1	0.55555	0.51851

Gambar 4. 16 Grafik error dari proses training *neural network*.

Terdapat total sebanyak 30 buah data input dimana terdapat 10 pasang input/output untuk 3 buah kondisi. Data ini akan dimasukkan menuju ke *neural network*. Setelah itu sistem akan melalui proses *training*.

Dari grafik yang ditunjukkan pada Gambar 4.16, terlihat bahwa nilai error dari proses *training* semakin kecil seiring dengan bertambahnya jumlah iterasi. Pada program, proses *training* akan berhenti saat nilai error aktual bernilai sama dengan atau lebih kecil dari nilai error minimum yang ditentukan pada program.

Setelah proses *training* selesai, maka nilai *weight* dan *bias* akan didapatkan untuk melakukan proses identifikasi secara *online*. Dari hasil pengujian didapatkan hasil yang ditunjukkan pada Tabel 4.18.

Dari total 10 kali pengujian terhadap 3 buah sampel daging dengan tingkat kesegaran yang berbeda, didapatkan persentase kesuksesan dalam identifikasi sebesar 80%. Pada kenyataannya sistem yang dibangun dapat membedakan secara pasti antara daging segar dengan daging busuk. Namun untuk proses identifikasi daging agak busuk terkadang dikenali sebagai daging busuk. Hal ini disebabkan karena pola yang dimiliki oleh daging agak busuk dengan daging busuk tidaklah jauh berbeda.

Kesalahan dalam identifikasi ini juga disebabkan karena kondisi ataupun lingkungan disekitar daging yang tidak sama pada saat proses pembusukan terjadi. Sehingga dengan waktu pembusukan yang sama sebuah daging membutuhkan waktu yang lebih cepat ataupun lebih lama dibandingkan data *training* yang didapatkan sebelumnya.

Tabel 4.18 Hasil pengujian secara *online*.

No	Target 1	Target 2	Daging Yang Diuji	Hasil
1	0	0	Segar	Segar
2	0	0	Segar	Segar
3	0	1	Agak Busuk	Agak Busuk
4	1	1	Agak Busuk	Busuk
5	1	1	Busuk	Busuk
6	1	1	Busuk	Busuk
7	0	0	Segar	Segar
8	0	0	Segar	Segar
9	1	1	Busuk	Busuk
10	1	1	Agak Busuk	Busuk

Halaman ini sengaja dikosongkan

BAB V
PENUTUP

5.1 Kesimpulan

Berdasarkan hasil pengujian dan analisa dari sistem yang telah dibuat dapat ditarik beberapa buah kesimpulan. Berdasarkan pengujian pada blok sensor warna, didapatkan kesimpulan bahwa sensor warna dapat merespon perbedaan tingkat kemerahan yang menjadi parameter kesegaran daging. Pada pengujian sensor gas didapatkan pola tegangan yang berbeda untuk ketiga buah sensor terhadap 3 buah sampel daging dengan tingkat kesegaran yang berbeda. Pola yang berbeda inilah yang memungkinkan penggunaan *neural network* sebagai metode pengenalan pola atau *pattern recognition*. Kesimpulan pertama dari sistem secara keseluruhan adalah penggunaan sensor warna dan tiga buah sensor gas dapat diimplementasikan untuk mengklasifikasikan tingkat kesegaran dengan baik. Selain itu penggunaan indera penciuman dan penglihatan manusia dalam menentukan tingkat kesegaran daging dapat digantikan oleh divais elektronik berupa sensor gas dan sensor warna. Dan yang terakhir adalah melalui penggunaan metode *neural network*, sistem yang dibangun dapat melakukan pengenalan pola terhadap tingkat kesegaran daging dengan tingkat keberhasilan mencapai 80%.

5.2 Saran

Beberapa saran yang penulis bisa berikan untuk pengembangan tugas akhir ini di masa yang akan datang adalah keakuratan pengujian pada tingkat kesegaran dagigng dapat ditingkatkan dengan menambahkan jumlah data *training* dari *neural network*. Selain itu proses identifikasi atau pengenalan pola dari tingkat kesegaran daging dapat menggunakan metode kecerdasan buatan lainnya seperti PCA, SVM dan lain-lain.

Halaman ini sengaja dikosongkan

DAFTAR PUSTAKA

[1] P. Guo and M. Bao, "Research and realization of hand-held mobile bacon detection based on Neural Network Pattern recognition," in *2016 Chinese Control and Decision Conference (CCDC)*, 2016, pp. 2018–2021.

[2] G. Peiyuan, B. Man, Q. Shiha, and C. Tianhua, "Detection of Meat Fresh Degree Based on Neural Network," in *2007 International Conference on Mechatronics and Automation*, 2007, pp. 2726–2730.

[3] E. Górska-Horczyczak *et al.*, "Applications of electronic noses in meat analysis," *Food Sci. Technol. Camp.*, vol. 36, no. 3, pp. 389–395, Sep. 2016.

[4] P. E. Keller, L. J. Kangas, L. H. Liden, S. Hashem, and R. T. Kouzes, "Electronic Noses And Their Applications," *ResearchGate*, Dec. 1995.

[5] T. Aguilera, J. Lozano, J. A. Paredes, F. J. Álvarez, and J. I. Suárez, "Electronic Nose Based on Independent Component Analysis Combined with Partial Least Squares and Artificial Neural Networks for Wine Prediction," *Sensors*, vol. 12, no. 6, pp. 8055–8072, Jun. 2012.

[6] J. Lozano, J. P. Santos, J. I. Suárez, M. Cabellos, T. Arroyo, and C. Horrillo, "Automatic Sensor System for the Continuous Analysis of the Evolution of Wine," *Am. J. Enol. Vitic.*, vol. 66, no. 2, pp. 148–155, May 2015.

[7] S. Bedoui, R. Faleh, H. Samet, and A. Kachouri, "Electronic nose system and principal component analysis technique for gases identification," in *10th International Multi-Conferences on Systems, Signals Devices 2013 (SSD13)*, 2013, pp. 1–6.

[8] S. Choopun, N. Hongsith, and E. Wongrat, "Metal-Oxide Nanowires for Gas Sensors," 2012.

[9] "Operating principle -MOS-type gas sensor." [Online]. Available: http://www.figaro.co.jp/en/technicalinfo/principle/mos-type.html. [Accessed: 01-Dec-2016].

[10] "TGS2602 : Gas Sensors and Modules - Products - Figaro Engineering Inc." [Online]. Available: http://www.figarosensor.com/products/entry/tgs2602.html. [Accessed: 09-Jan-2017].

[11] D. oleh khoi roni, “Contoh Penggunaan Dasar Sensor Gas TGS2602 dgn Microcontroller AVR Atmega8535.” .
[12] “Arduino Playground - MQGasSensors.” [Online]. Available: http://playground.arduino.cc/Main/MQGasSensors. [Accessed: 09-Jan-2017].
[13] manish, “Get Better With TCS3200 Color Sensor Module.” .
[14] “Raspberry Pi,” *Wikipedia*. 06-Dec-2016.
[15] “Raspberry Pi 3 - Model B - ARMv8 with 1G RAM ID: 3055 - $39.95 : Adafruit Industries, Unique & fun DIY electronics and kits.” [Online]. Available: https://www.adafruit.com/product/3055. [Accessed: 09-Jan-2017].
[16] “Arduino - ArduinoBoardUno.” [Online]. Available: https://www.arduino.cc/en/Main/ArduinoBoardUno. [Accessed: 09-Dec-2016].
[17] D. Terrell, *Op Amps: Design, Application, and Troubleshooting, Second Edition*, 2 edition. Boston: Newnes, 1996.
[18] “Understanding SAR ADCs: Their Architecture and Comparison with Other ADCs - Tutorial - Maxim.” [Online]. Available: https://www.maximintegrated.com/en/app-notes/index.mvp/id/1080. [Accessed: 09-Jan-2017].
[19] “Serial Communication - learn.sparkfun.com.” [Online]. Available: https://learn.sparkfun.com/tutorials/serial-communication. [Accessed: 10-Dec-2016].
[20] M. T. Hagan, H. B. Demuth, M. H. Beale, and O. D. Jesús, *Neural Network Design*, 2 edition. Place of publication not identified: Martin Hagan, 2014.

LAMPIRAN

1. Program pada Arduino

```
#include <TimerOne.h>

#define S0    6
#define S1    5
#define S2    4
#define S3    3
#define OUT   2

int   g_count = 0;    // count the frequecy
int   g_array[3];     // store the RGB value
int   g_flag = 0;     // filter of RGB queue
float g_SF[3];        // save the RGB Scale factor
int rgb[3];

// Init TSC230 and setting Frequency.
void TSC_Init()
{
  pinMode(S0, OUTPUT);
  pinMode(S1, OUTPUT);
  pinMode(S2, OUTPUT);
  pinMode(S3, OUTPUT);
  pinMode(OUT, INPUT);

  digitalWrite(S0, LOW);  // OUTPUT FREQUENCY SCALING 2%
  digitalWrite(S1, HIGH);
}

// Select the filter color
void TSC_FilterColor(int Level01, int Level02)
{
  if(Level01 != 0)
    Level01 = HIGH;

  if(Level02 != 0)
    Level02 = HIGH;

  digitalWrite(S2, Level01);
```

```
  digitalWrite(S3, Level02);
}

void TSC_Count()
{
  g_count ++;
}

void TSC_Callback()
{
  switch(g_flag)
  {
    case 0:
        TSC_WB(LOW, LOW);             //Filter without Red
        break;
    case 1:
        g_array[0] = g_count;
        TSC_WB(HIGH, HIGH);           //Filter without Green
        break;
    case 2:
        g_array[1] = g_count;
        TSC_WB(LOW, HIGH);            //Filter without Blue
        break;

    case 3:
        g_array[2] = g_count;
        TSC_WB(HIGH, LOW);            //Clear(no filter)
        break;
   default:
        g_count = 0;
        break;
  }
}

void TSC_WB(int Level0, int Level1)      //White Balance
{
  g_count = 0;
  g_flag ++;
  TSC_FilterColor(Level0, Level1);
  Timer1.setPeriod(1000000);            // set 1s period
```

```
}

void setup()
{
  TSC_Init();
  Serial.begin(9600);
  Timer1.initialize();            // defaulte is 1s
  Timer1.attachInterrupt(TSC_Callback);
  attachInterrupt(0, TSC_Count, RISING);

  delay(4000);

  for(int i=0; i<3; i++)

  g_SF[0] = 255.0/ g_array[0];     //R Scale factor
  g_SF[1] = 255.0/ g_array[1] ;    //G Scale factor
  g_SF[2] = 255.0/ g_array[2] ;    //B Scale factor

}

void loop()
{
  int rMQ_136 = analogRead(A0);
  int rMQ_137 = analogRead(A1);
  int rTGS_2602 = analogRead(A2);
  int rTGS_2611 = analogRead(A3);

  float MQ_136 = rMQ_136 * (5.00/1023.00);
  float MQ_137 = rMQ_137* (5.00/1023.00);
  float TGS_2602 = rTGS_2602* (5.00/1023.00);

   g_flag = 0;
   rgb[0] = (g_array[0] * g_SF[0]);
   if (rgb[0] >= 255)
   {
    rgb[0] = 255;
   }
   else
   {
    rgb[0] = rgb[0];
```

```
  }
  rgb[1] = (g_array[1] * g_SF[1]);
  if (rgb[1] >= 255)
  {
   rgb[1] = 255;
  }
  else
  {
   rgb[1] = rgb[1];
  }
  rgb[2] = (g_array[2] * g_SF[2]);
  if (rgb[2] >= 255)
  {
   rgb[2] = 255;
  }
  else
  {
   rgb[2] = rgb[2];
  }

  Serial.print('A');
  Serial.print(rgb[0]);
  Serial.print('B');
  Serial.print(rgb[1]);
  Serial.print('C');
  Serial.print(rgb[2]);
  Serial.print('D');
  Serial.print(MQ_136);
  Serial.print('E');
  Serial.print(MQ_137);
  Serial.print('F');
  Serial.print(TGS_2602);
  Serial.println('#');

  delay(4000);
}
```

2. Program pada Lazarus

```
const
  // 00 = SEGAR, 01 = AGAK BUSUK, 11 = BUSUK
  target : array [1..30,1..2] of Integer = ((0,0),(0,0),(0,0),(0,0), (0,0), (0,0), (0,0), (0,0), (0,0),
(0,0), (0,1), (0,1), (0,1), (0,1), (0,1), (0,1), (0,1), (0,1), (0,1), (1,1), (1,1), (1,1), (1,1),
(1,1), (1,1), (1,1), (1,1), (1,1), (1,1));
input : array [1..30,1..6] of Extended =
                                ((0.443609023, 1, 0.593984962, 1, 0.329268293, 0.292682927),
                                (0.803571429, 0.946428571, 1, 1, 0.297297297, 0.243243243),
                                (0.877358491, 1, 0.622641509, 1, 0.333333333, 0.282051282),
                                (0.731034483, 0.606896552, 1, 1, 0.325, 0.275),
                                (0.518248175, 0.284671533, 1, 1, 0.3125, 0.2625),
                                (0.761589404, 0.781456954, 1, 1, 0.313253012, 0.265060241),
                                (0.982758621, 1, 0.982758621, 1, 0.316455696, 0.278481013),
                                (1, 0.68503937, 0.795275591, 1, 0.405405405, 0.351351351),
                                (0.733333333, 0.7, 1, 1, 0.426666667, 0.386666667),
                                (0.540540541, 0.75, 1, 1, 0.395061728, 0.345679012),

                                (0.282608696, 0.619565217, 1, 1, 0.490909091, 0.436363636),
                                (0.361111111, 0.722222222, 1, 1, 0.509090909, 0.436363636),
                                (0.205298013, 0.410596026, 1, 1, 0.527272727, 0.454545455),
                                (0.224299065, 0.119158879, 1, 1, 0.5, 0.425925926) ,
                                (0.160071942, 0.226618705, 1, 1, 0.527272727, 0.454545455),
                                (0.272727273, 0.657342657, 1, 1, 0.56, 0.48),
                                (0.288409704, 0.778975741, 1, 1, 0.518518519, 0.444444444) ,
                                (0.184931507, 0.335616438, 1, 1, 0.529411765, 0.470588235) ,
                                (0.209183673, 0.326530612, 1, 1, 0.520833333, 0.4375) ,
                                (0.187613843, 0.293260474, 1, 1, 0.518518519, 0.462962963),

                                (0.536964981, 1, 0.859922179, 1, 0.533333333, 0.533333333),
                                (0.363265306, 1, 0.828571429, 1, 0.551724138, 0.517241379),
                                (0.857923497, 1, 0.781420765, 1, 0.551724138, 0.517241379),
                                (0.606060606, 0.772727273, 1.00E+00, 1, 0.533333333, 0.5),
                                (0.447619048, 1, 0.815873016, 1, 0.551724138, 0.517241379),
                                (0.890909091, 0.981818182, 1, 1, 0.548387097, 0.516129032),
                                (0.431372549, 0.568627451, 1, 1, 0.548387097, 0.548387097),
```

```
                    (0.621621622, 0.554054054, 1, 1, 0.56, 0.52),
                    (0.68722467, 0.378854626, 1, 1, 0.548387097, 0.548387097),
                    (0.494809689, 0.214532872, 1, 1, 0.555555556, 0.518518519));

procedure TForm1.TRAIN_BUTTONClick(Sender: TObject);
begin
  Chart1LineSeries1.Clear;
  alfa := StrToFloat(ALPHA_EditBox.Text);
  miu := StrToFloat(MIU_EditBox.Text);
  epsilon := StrToFloat(EPSILON_EditBox.Text);
  eror := 1;
  iterasi := 1;
  Weight_init;
  repeat
    Application.ProcessMessages;
    Chart1LineSeries1.AddXY(iterasi, errortotal);
    for z := 1 to 30 do
    begin
      Forward_propagation;
      Back_Propagation;
      UpdateNilai;
    end;
    errortotal := (errorkum[1] + errorkum[2] + errorkum[3] + errorkum[4] + errorkum[5] +
errorkum[6] + errorkum[7] + errorkum[8] + errorkum[9] + errorkum[10] + errorkum[11] +
errorkum[12] + errorkum[13] + errorkum[14] + errorkum[15] + errorkum[16] +
errorkum[17] + errorkum[18] + errorkum[19] + errorkum[20] + errorkum[21] +
errorkum[22] + errorkum[23] + errorkum[24] + errorkum[25] + errorkum[26] +
errorkum[27] + errorkum[28] + errorkum[29] + errorkum[30]);
    iterasi := iterasi + 1;
    ERROR_EditBox.Text := FloatToStr(errortotal);
    ITERASI_EditBox.Text := FloatToStr(iterasi);
    delay(3);
  until errortotal <= epsilon;
end;
procedure TForm1.Timer1Timer(Sender: TObject);
var
  x, y, t: Integer;
  max_gas, max_color : Extended;
begin
```

```
x := StrToInt(SECOND_TIME_Label.Caption);
SECOND_TIME_Label.Caption := IntToStr(x + 1);
y := StrToInt(SECOND_TIME_Label.Caption);

if y = 1 then
begin
  MQ_136_avg[1] := MQ_136f;
  MQ_137_avg[1] := MQ_137f;
  TGS_2602_avg[1] := TGS_2602f;
end;

if y = 2 then
begin
  MQ_136_avg[2] := MQ_136f;
  MQ_137_avg[2] := MQ_137f;
  TGS_2602_avg[2] := TGS_2602f;
end;

if y = 3 then
begin
  MQ_136_avg[3] := MQ_136f;
  MQ_137_avg[3] := MQ_137f;
  TGS_2602_avg[3] := TGS_2602f;
end;

if y = 4 then
begin
  MQ_136_avg[4] := MQ_136f;
  MQ_137_avg[4] := MQ_137f;
  TGS_2602_avg[4] := TGS_2602f;
end;

if y = 5 then
begin
  MQ_136_avg[5] := MQ_136f;
  MQ_137_avg[5] := MQ_137f;
  TGS_2602_avg[5] := TGS_2602f;
end;
```

```
if y = 6 then
begin
  MQ_136_avg[6] := MQ_136f;
  MQ_137_avg[6] := MQ_137f;
  TGS_2602_avg[6] := TGS_2602f;
end;

if y = 7 then
begin
  MQ_136_avg[7] := MQ_136f;
  MQ_137_avg[7] := MQ_137f;
  TGS_2602_avg[7] := TGS_2602f;
end;

if y = 8 then
begin
  MQ_136_avg[8] := MQ_136f;
  MQ_137_avg[8] := MQ_137f;
  TGS_2602_avg[8] := TGS_2602f;
end;

if y = 9 then
begin
  MQ_136_avg[9] := MQ_136f;
  MQ_137_avg[9] := MQ_137f;
  TGS_2602_avg[9] := TGS_2602f;
end;

if y = 10 then
begin
  MQ_136_avg[10] := MQ_136f;
  MQ_137_avg[10] := MQ_137f;
  TGS_2602_avg[10] := TGS_2602f;
end;

if y = 11 then
begin
```

```
  MQ_136_avg[11] := MQ_136f;
  MQ_137_avg[11] := MQ_137f;
  TGS_2602_avg[11] := TGS_2602f;
end;

if y = 12 then
begin
  MQ_136_avg[12] := MQ_136f;
  MQ_137_avg[12] := MQ_137f;
  TGS_2602_avg[12] := TGS_2602f;
end;

if y = 13 then
begin
  MQ_136_avg[13] := MQ_136f;
  MQ_137_avg[13] := MQ_137f;
  TGS_2602_avg[13] := TGS_2602f;
end;

if y = 14 then
begin
  MQ_136_avg[14] := MQ_136f;
  MQ_137_avg[14] := MQ_137f;
  TGS_2602_avg[14] := TGS_2602f;
end;

if y = 15then
begin
  MQ_136_avg[15] := MQ_136f;
  MQ_137_avg[15] := MQ_137f;
  TGS_2602_avg[15] := TGS_2602f;
end;

if y = 16 then
begin
  MQ_136_avg[16] := MQ_136f;
  MQ_137_avg[16] := MQ_137f;
  TGS_2602_avg[16] := TGS_2602f;
```

```
end;

if y = 17 then
begin
  MQ_136_avg[17] := MQ_136f;
  MQ_137_avg[17] := MQ_137f;
  TGS_2602_avg[17] := TGS_2602f;
end;

if y = 18 then
begin
  MQ_136_avg[18] := MQ_136f;
  MQ_137_avg[18] := MQ_137f;
  TGS_2602_avg[18] := TGS_2602f;
end;

if y = 19 then
begin
  MQ_136_avg[19] := MQ_136f;
  MQ_137_avg[19] := MQ_137f;
  TGS_2602_avg[19] := TGS_2602f;
end;

if y = 20 then
begin
  MQ_136_avg[20] := MQ_136f;
  MQ_137_avg[20] := MQ_137f;
  TGS_2602_avg[20] := TGS_2602f;
end;

if y = 41 then
begin
  MQ_136_avg[41] := MQ_136f;
  MQ_137_avg[41] := MQ_137f;
  TGS_2602_avg[41] := TGS_2602f;
end;

if y = 42 then
```

```
begin
  MQ_136_avg[42] := MQ_136f;
  MQ_137_avg[42] := MQ_137f;
  TGS_2602_avg[42] := TGS_2602f;
end;

if y = 43 then
begin
  MQ_136_avg[43] := MQ_136f;
  MQ_137_avg[43] := MQ_137f;
  TGS_2602_avg[43] := TGS_2602f;
end;

if y = 44 then
begin
  MQ_136_avg[44] := MQ_136f;
  MQ_137_avg[44] := MQ_137f;
  TGS_2602_avg[44] := TGS_2602f;
end;

if y = 45 then
begin
  MQ_136_avg[45] := MQ_136f;
  MQ_137_avg[45] := MQ_137f;
  TGS_2602_avg[45] := TGS_2602f;
end;

if y = 46 then
begin
  MQ_136_avg[46] := MQ_136f;
  MQ_137_avg[46] := MQ_137f;
  TGS_2602_avg[46] := TGS_2602f;
end;

if y = 47 then
begin
  MQ_136_avg[47] := MQ_136f;
  MQ_137_avg[47] := MQ_137f;
```

```
  TGS_2602_avg[47] := TGS_2602f;
end;

if y = 48 then
begin
  MQ_136_avg[48] := MQ_136f;
  MQ_137_avg[48] := MQ_137f;
  TGS_2602_avg[48] := TGS_2602f;
end;

if y = 49 then
begin
  MQ_136_avg[49] := MQ_136f;
  MQ_137_avg[49] := MQ_137f;
  TGS_2602_avg[49] := TGS_2602f;
end;

if y = 50 then
begin
  MQ_136_avg[50] := MQ_136f;
  MQ_137_avg[50] := MQ_137f;
  TGS_2602_avg[50] := TGS_2602f;
end;

if y = 51 then
begin
  MQ_136_avg[51] := MQ_136f;
  MQ_137_avg[51] := MQ_137f;
  TGS_2602_avg[51] := TGS_2602f;
end;

if y = 52 then
begin
  MQ_136_avg[52] := MQ_136f;
  MQ_137_avg[52] := MQ_137f;
  TGS_2602_avg[52] := TGS_2602f;
end;
```

```
if y = 53 then
begin
  MQ_136_avg[53] := MQ_136f;
  MQ_137_avg[53] := MQ_137f;
  TGS_2602_avg[53] := TGS_2602f;
end;

if y = 54 then
begin
  MQ_136_avg[54] := MQ_136f;
  MQ_137_avg[54] := MQ_137f;
  TGS_2602_avg[54] := TGS_2602f;
end;

if y = 55 then
begin
  MQ_136_avg[55] := MQ_136f;
  MQ_137_avg[55] := MQ_137f;
  TGS_2602_avg[55] := TGS_2602f;
end;

if y = 56 then
begin
  MQ_136_avg[56] := MQ_136f;
  MQ_137_avg[56] := MQ_137f;
  TGS_2602_avg[56] := TGS_2602f;
end;

if y = 57 then
begin
  MQ_136_avg[57] := MQ_136f;
  MQ_137_avg[57] := MQ_137f;
  TGS_2602_avg[57] := TGS_2602f;
end;

if y = 58 then
begin
  MQ_136_avg[58] := MQ_136f;
```

```
  MQ_137_avg[58] := MQ_137f;
  TGS_2602_avg[58] := TGS_2602f;
end;

if y = 59 then
begin
  MQ_136_avg[59] := MQ_136f;
  MQ_137_avg[59] := MQ_137f;
  TGS_2602_avg[59] := TGS_2602f;
end;

if y = 60 then
begin
  MQ_136_avg[60] := MQ_136f;
  MQ_137_avg[60] := MQ_137f;
  TGS_2602_avg[60] := TGS_2602f;

  MQ_136_total_avg := 0;
  MQ_137_total_avg := 0;
  TGS_2602_total_avg := 0;

  MQ_136_total_avg_lower := 0;
  MQ_137_total_avg_lower := 0;
  TGS_2602_total_avg_lower := 0;

  MQ_136_total_avg_upper := 0;
  MQ_137_total_avg_upper := 0;
  TGS_2602_total_avg_upper := 0;

  for t := 1 to 20 do
  begin
    MQ_136_total_avg_lower := MQ_136_total_avg_lower  + MQ_136_avg[t];
    MQ_137_total_avg_lower := MQ_137_total_avg_lower + MQ_137_avg[t];
    TGS_2602_total_avg_lower := TGS_2602_total_avg_lower + TGS_2602_avg[t];
  end;

  MQ_136_total_avg_lower := MQ_136_total_avg_lower / 20;
  MQ_137_total_avg_lower := MQ_137_total_avg_lower / 20;
```

```
TGS_2602_total_avg_lower := TGS_2602_total_avg_lower / 20;

for t := 41 to 60 do
begin
  MQ_136_total_avg_upper := MQ_136_total_avg_upper  + MQ_136_avg[t];
  MQ_137_total_avg_upper := MQ_137_total_avg_upper + MQ_137_avg[t];
  TGS_2602_total_avg_upper := TGS_2602_total_avg_upper + TGS_2602_avg[t];
end;

MQ_136_total_avg_upper := MQ_136_total_avg_upper / 20;
MQ_137_total_avg_upper := MQ_137_total_avg_upper / 20;
TGS_2602_total_avg_upper := TGS_2602_total_avg_upper / 20;

MQ_136_total_avg := abs(MQ_136_total_avg_upper - MQ_136_total_avg_lower) ;    //
baseline (selisih)
MQ_137_total_avg := abs(MQ_137_total_avg_upper - MQ_137_total_avg_lower);    //
baseline (selisih)
TGS_2602_total_avg          :=          abs(TGS_2602_total_avg_upper          -
TGS_2602_total_avg_lower);// baseline (selisih)

if (MQ_136_total_avg >= MQ_137_total_avg) and (MQ_136_total_avg >=
TGS_2602_total_avg) then
begin
  max_gas := MQ_136_total_avg ;
end;

if (MQ_137_total_avg >= MQ_136_total_avg) and (MQ_137_total_avg >=
TGS_2602_total_avg) then
begin
  max_gas := MQ_137_total_avg ;
end;

if (TGS_2602_total_avg >= MQ_136_total_avg) and (TGS_2602_total_avg >=
MQ_137_total_avg) then
begin
  max_gas := TGS_2602_total_avg;
end;

if (Redf >= Greenf) and (Redf >= Bluef) then
```

```
  begin
    max_color := Redf;
  end;

  if (Greenf >= Redf) and (Greenf >= Bluef) then
  begin
    max_color := Greenf;
  end;

  if (Bluef >= Redf) and (Bluef >= Greenf) then
  begin
    max_color := Bluef;
  end;

  AssignFile(data100, 'nilai.txt');
  Rewrite(data100);
  Append(data100);

  WriteLn(data100, MQ_136f);
  WriteLn(data100, MQ_137f);
  WriteLn(data100, TGS_2602f);
  WriteLn(data100, Redf);
  WriteLn(data100, Greenf);
  WriteLn(data100, Bluef);
  CloseFile(data100);

  Edit8.Text := FloatToStr(MQ_136_total_avg);
  Edit9.Text := FloatToStr(MQ_137_total_avg);
  Edit10.Text := FloatToStr(TGS_2602_total_avg);
  Edit1.Text := FloatToStr(Redf);
  Edit2.Text := FloatToStr(Greenf);
  Edit3.Text := FloatToStr(Bluef);


  Sensor[1] := abs(StrToFloat(Edit8.Text)/max_gas);       // Ganti dengan cara copy nilai
stlah 60 detik ke editboxnya
  Sensor[2] := abs(StrToFloat(Edit9.Text)/max_gas);      // Ada 6 buah editbox untuk R, G,
B, MQ-136, MQ-137, TGS 2602
```

```
Sensor[3] := abs(StrToFloat(Edit10.Text)/max_gas);
Sensor[4] := abs(StrToFloat(Edit1.Text)/max_color);
Sensor[5] := abs(StrToFloat(Edit2.Text)/max_color);
Sensor[6] := abs(StrToFloat(Edit3.Text)/max_color);

Edit7.Text := FloatToStr(Sensor[1]);
Edit13.Text := FloatToStr(Sensor[2]);
Edit14.Text := FloatToStr(Sensor[3]);
Edit15.Text := FloatToStr(Sensor[4]);
Edit16.Text := FloatToStr(Sensor[5]);
Edit17.Text := FloatToStr(Sensor[6]);

Verification;

Edit11.Text := FloatToStr(round(out3_2[1]));
Edit12.Text := FloatToStr(round(out3_2[2]));
Edit5.Text:= FloatToStr(out3_2[1]);
Edit6.Text:= FloatToStr(out3_2[2]);

if (round(out3_2[1]) = 0) then
begin
  if (round(out3_2[2]) = 0) then
  begin
    Edit4.Text:= 'Segar';
  end;
end;

if (round(out3_2[1]) = 0) then
begin
  if (round(out3_2[2]) = 1) then
  begin
    Edit4.Text:= 'Agak Busuk';
  end;
end;

if (round(out3_2[1]) = 1) then
begin
  if (round(out3_2[2]) = 1) then
```

```
      begin
        Edit4.Text:= 'Busuk';
      end;
    end;

  end;
end;
```

BIODATA PENULIS

Joshwa simamora lahir di Balikpapan pada 23 Agustus 1994, yang merupakan anak pertama dari tiga bersaudara dari pasangan Milton Simamora dan Sarmina Sitinjak. Penulis menyelesaikan pendidikan dasar di SD YBBSU Balikpapan dan dilanjutkan dengan pendidikan menengah di SMPN 1 Balikpapan dan SMAN 1 Balikpapan. Pada tahun 2012, penulis memulai pendidikan di jurusan Teknik Elektro, Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember (ITS) Surabaya. Selama kuliah, penulis aktif membantu penyelenggaraan kegiatan dan aktif sebagai asisten laboratorium Elektronika Dasar.

Email:
joshwasimamora@gmail.com