TUGAS AKHIR – IF184802

RANCANG BANGUN COMPUTATION OFFLOADING FRAMEWORK DENGAN METODE FUZZY-MCDM MENGGUNAKAN PEMBOBOTAN DINAMIS UNTUK PENINGKATAN KINERJA SERTA PENGHEMATAN DAYA PADA PROSES KOMPUTASI DI RASPBERRY PI

ARYA WIRANATA
NRP 05111540000163

Dosen Pembimbing I
Waskitho Wibisono, S.Kom., M.Eng., Ph.D.

Dosen Pembimbing II
Tohari Ahmad, S.Kom., MIT., Ph.D.

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2019

TUGAS AKHIR – IF184802

RANCANG BANGUN COMPUTATION OFFLOADING FRAMEWORK DENGAN METODE FUZZY-MCDM MENGGUNAKAN PEMBOBOTAN DINAMIS UNTUK PENINGKATAN KINERJA SERTA PENGHEMATAN DAYA PADA PROSES KOMPUTASI DI RASPBERRY PI

ARYA WIRANATA
NRP 05111540000163

Dosen Pembimbing I
Waskitho Wibisono, S.Kom., M.Eng., Ph.D.

Dosen Pembimbing II
Tohari Ahmad, S.Kom., MIT., Ph.D.

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2019

(Halaman ini sengaja dikosongkan)

UNDERGRADUATE THESIS – IF4802

DESIGN AND DEVELOPMENT OF COMPUTATION OFFLOADING FRAMEWORK USING FUZZY-MCDM METHOD AND DYNAMIC WEIGHTING FOR IMPROVING PERFORMANCE AND POWER SAVING IN RASPBERRY PI

ARYA WIRANATA
NRP 05111540000163

First Advisor
Waskitho Wibisono, S.Kom., M.Eng., Ph.D.

Second Advisor
Tohari Ahmad, S.Kom., MIT., Ph.D.

DEPARTMENT OF INFORMATICS
Faculty of Information and Communication Technology
Sepuluh Nopember Institute of Technology
Surabaya 2019

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

RANCANG BANGUN COMPUTATION OFFLOADING FRAMEWORK DENGAN METODE FUZZY-MCDM MENGGUNAKAN PEMBOBOTAN DINAMIS UNTUK PENINGKATAN KINERJA SERTA PENGHEMATAN DAYA PADA PROSES KOMPUTASI DI RASPBERRY PI

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Komputasi Berbasis Jaringan
Program Studi S-1 Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh:
ARYA WIRANATA
NRP: 05111540000163

Disetujui oleh Pembimbing Tugas Akhir:

1. Waskitho Wibisono, S.Kom., M.Eng., Ph.D. .....
   (NIP. 197410222000031001) (Pembimbing 1)

2. Tohari Ahmad, S.Kom., MIT., Ph.D. .....
   (NIP. 197505252003121002) (Pembimbing 2)

SURABAYA
JUNI, 2019

(Halaman ini sengaja dikosongkan)

RANCANG BANGUN *COMPUTATION OFFLOADING FRAMEWORK* DENGAN METODE *FUZZY-MCDM* MENGGUNAKAN PEMBOBOTAN DINAMIS UNTUK PENINGKATAN KINERJA SERTA PENGHEMATAN DAYA PADA PROSES KOMPUTASI DI RASPBERRY PI

Nama Mahasiswa	**: ARYA WIRANATA**
NRP	**: 05111540000163**
Jurusan	**: Departemen Informatika FTIK-ITS**
Dosen Pembimbing 1	**: Waskitho Wibisono, S.Kom., M.Eng., Ph.D.**
Dosen Pembimbing 2	**: Tohari Ahmad, S.Kom., MIT., Ph.D.**

Abstrak

Offloading adalah suatu metode pengeksekusian sebuah beban kerja pada sebuah perangkat dengan mengirimkan modul berisi beban kerja tersebut kepada perangkat lain yang memiliki sumber daya dan kemampuan komputasi yang lebih baik. Hasil eksekusi dari beban kerja akan diterima kembali oleh perangkat yang telah mengirim modul beban kerja sebelumnya. Teknik ini dianggap sebagai salah satu cara mengatasi keterbatasan perangkat bergerak yang memiliki sumber daya dan kemampuan komputasi yang terbatas. Oleh karena itu, diperlukan adanya penerapan metode offloading dalam mengeksekusi beban kerja pada perangkat bergerak dengan tujuan dapat melakukan peningkatan kinerja pada perangkat.

Penentuan beban kerja diproses secara lokal atau offloading ditentukan oleh sebuah decision maker dengan menggunakan pembobotan dinamis. Pada tugas akhir ini decision maker menggunakan metode Fuzzy Multi-Criteria Decision Making dengan pembobotan dinamis.

Pada tugas akhir ini, pemanfaatan computation offloading diharapkan dapat meningkatkan kinerja proses komputasi sehingga meminimalisir waktu eksekusi beban kerja serta

penghematan daya. Dari uji coba yang telah dilakukan, metode computation offloading dengan Fuzzy Multi-Criteria Decision Making dengan pembobotan dinamis mendapatkan waktu eksekusi beban kerja yang lebih cepat dibandingkan dengan metode lokal dan offloading saja.

Kata kunci: Offloading, image processing, Perangkat IoT, beban kerja, pembobotan dinamis.

DESIGN AND DEVELOPMENT OF COMPUTATION OFFLOADING FRAMEWORK USING FUZZY-MCDM METHOD AND DYNAMIC WEIGHTING FOR IMPROVING PERFORMANCE AND POWER SAVING IN RASPBERRY PI

Student's Name	**: ARYA WIRANATA**
Student's ID	**: 05111540000163**
Department	**: Informatics FTIK-ITS**
First Advisor	**: Waskitho Wibisono, S.Kom., M.Eng., Ph.D.**
Second Advisor	**: Tohari Ahmad, S.Kom., MIT., Ph.D.**

Abstract

Offloading is a method of executing a workload on a device by sending a module containing workload to another device that has better resources and computing capabilities. This technique is considered as one of the ways to overcome the limitations of devices that have limited resources and computing capabilities. Therefore, this offloading method to executing the workload on a device with the purpose of improving performance on devices.

The decision of task to be done in local or offloading mode is calculated by the decision maker. In this undergraduate thesis, the decision maker using Fuzzy Multi-Criteria Decision Making method with dynamic weighting.

In this undergraduate thesis, the hope to improve the computation process to reduce execution time and increase power saving. From the experiment, the computation offloading method with Fuzzy Multi-Criteria Decision Making with dynamic weighting got the execution time of task less than the only local method and offloading method.

Keywords : Offloading, image processing, workload, dynamic weighting.

(***Halaman ini sengaja dikosongkan***)

KATA PENGANTAR

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

Al-ḥamdu li-llāhi rabbi l-ʿālamīn, segala puji bagi Allah SWT yang telah melimpahkan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul:

"**RANCANG BANGUN *COMPUTATION OFFLOADING FRAMEWORK* DENGAN METODE *FUZZY-MCDM* MENGGUNAKAN PEMBOBOTAN DINAMIS UNTUK PENINGKATAN KINERJA SERTA PENGHEMATAN DAYA PADA PROSES KOMPUTASI DI RASPBERRY PI**"

yang merupakan salah satu syarat dalam menempuh ujian sidang guna memperoleh gelar Sarjana Komputer. Selesainya Tugas Akhir ini tidak terlepas dari bantuan dan dukungan beberapa pihak, sehingga pada kesempatan ini penulis mengucapkan terima kasih kepada:

1. Allah SWT, karena atas izin-Nya lah penulis dapat menyelesaikan Tugas Akhir dengan baik.
2. Kedua orang tua, Bambang Damyasik serta Wa Ode Ndipo Tayb, dan kedua saudara Pandu Wicaksono serta Indrabrata, terima kasih atas doa dan bantuan moral dan material selama penulis belajar di departemen Informatika ITS.
3. Bapak Waskitho Wibisono, S.Kom., M.Eng., Ph.D. selaku dosen pembimbing I yang telah membimbing dan memberikan motivasi, nasehat dan bimbingan dalam menyelesaikan Tugas Akhir ini sekaligus dosen wali penulis yang telah memberikan arahan, masukan dan motivasi kepada penulis.
4. Tohari Ahmad, S.Kom., MIT., Ph.D. selaku dosen pembimbing II yang telah membimbing dan memberikan motivasi, nasehat dan bimbingan dalam menyelesaikan Tugas Akhir ini.

5. Bapak Darlis Herumurti, S.Kom., M.Kom. selaku kepala departemen Informatika ITS.
6. Bapak Dr. Eng. Radityo Anggoro, S.Kom., M.Sc. selaku koordinator Tugas Akhir di departemen Informatika ITS.
7. Seluruh dosen dan karyawan Teknik Informatika ITS yang telah memberikan ilmu dan pengalaman kepada penulis selama menjalani masa studi di ITS.
8. Ibu Eva Mursidah dan Ibu Sri Budiati yang selalu mempermudah penulis dalam peminjaman buku di RBTC.
9. Beberapa *Record Store* lokal yang telah memperjual-belikan album kepada penulis dalam rilisan fisik berupa piringan hitam, kaset pita, dan *Compact Disc* (CD).
10. Seluruh rekan–rekan TC 2015 yang sudah mendukung penulis selama perkuliahan.
11. Teman-teman seperjuangan RMK NCC/KBJ, yang telah menemani dan menyemangati penulis.
12. Rekan-rekan Admin Sekre x Warkop NF Adam, Adib, Affan, Aidil, Aqil, Ariya, Djohan, Faiq, Fatur, Hatta, Hilmi, Huda, Ichsan, Illham, Mail, Naufal, Tamtam, Tegar, dan Yasin yang telah memberi dukungan, hiburan dan motivasi kepada penulis.
13. Eko, Reinhart, dan Rogo yang telah memberi dukungan, hiburan, dan motivasi kepada penulis.
14. Rekan-rekan penulis yang tidak dapat disebutkan satu per satu yang selalu membantu, menghibur, menjadi tempat bertukar ilmu dan berjuang bersama-sama penulis.

Penulis menyadari bahwa Tugas Akhir ini masih memiliki banyak kekurangan sehingga dengan kerendahan hati penulis mengharapkan kritik dan saran dari pembaca untuk perbaikan ke depan.

Surabaya, 25 Juni 2019

DAFTAR ISI

LEMBAR PENGESAHAN ..... **v**
Abstrak ..... **vii**
Abstract ..... **ix**
KATA PENGANTAR ..... **xi**
DAFTAR ISI ..... **xiii**
DAFTAR GAMBAR ..... **xvii**
DAFTAR TABEL ..... **xix**
DAFTAR KODE SUMBER ..... **xxi**
BAB I PENDAHULUAN **1**
1.1 Latar Belakang 1
1.2 Rumusan Masalah 3
1.3 Batasan Permasalahan 3
1.4 Tujuan 4
1.5 Manfaat 4
1.6 Metodologi 4
1.6.1 Penyusunan Proposal 4
1.6.2 Studi Literatur 5
1.6.3 Analisis dan Desain Perangkat Lunak 5
1.6.4 Implementasi Perangkat Lunak 5
1.6.5 Pengujian dan Evaluasi 6
1.6.6 Penyusunan Buku 6
1.7 Sistematika Penulisan Laporan 6
BAB II TINJAUAN PUSTAKA **9**
2.1 Raspberry Pi 9
2.2 Sistem Operasi Raspbian 9
2.3 Sensor Arus INA219 10
2.4 *Internet of Things (IoT)* 10
2.5 *Offloading* 11
2.6 Faktor Penentu Metode *Offloading* 12
2.7 *Fuzzy Multi Criteria Decision Making* 12
2.8 *Entropy* 15
2.9 *Entropy Based Fuzzy* MCDM 16
2.10 OpenCV 18

2.11 MobileNets 19
BAB III PERANCANGAN SISTEM 21
3.1 Deskripsi Umum Sistem 21
3.2 Arsitektur Umum Sistem 21
3.3 Data 24
3.3.1 Data Masukan 25
3.3.2 Data Keluaran 25
3.4 Faktor Penentu Metode *Offloading* 26
3.4.1 Kapasitas Raspberry Pi RAM yang tersedia 26
3.4.2 *Download Speed* 27
3.4.3 *Upload Speed* 28
3.4.4 Jumlah Data Citra yang diproses 28
3.4.5 Daya yang dikonsumsi 29
3.5 *Offloading* dengan Metode *Entropy Based Fuzzy* MCDM 30
BAB IV IMPLEMENTASI 33
4.1 Lingkungan Implementasi 33
4.2 Implementasi 34
4.2.1 Faktor Penentu Metode *Offloading* 34
4.2.2 Perhitungan *Entropy Based Fuzzy* MCDM 37
4.2.3 *Entropy Based Fuzzy* MCDM *Code* 43
4.2.4 Komunikasi Perangkat IoT dan *cloud server* 50
4.2.5 Penghitungan Jumlah Mobil pada Gambar 54
BAB V HASIL UJI COBA DAN EVALUASI 57
5.1 Lingkungan Pengujian 57
5.2 Skenario Uji Coba 63
5.2.1 Skenario Uji Coba 1 65
5.2.2 Skenario Uji Coba 2 69
5.2.3 Skenario Uji Coba 3 73
5.3 Evaluasi Umum Skenario Uji Coba 78
BAB VI KESIMPULAN DAN SARAN 81
6.1 Kesimpulan 81
6.2 Saran 82
DAFTAR PUSTAKA 83
LAMPIRAN 85

BIODATA PENULIS.....87

(***Halaman ini sengaja dikosongkan***)

DAFTAR GAMBAR

Gambar 2.1 Raspberry Pi 1 Model B 9
Gambar 2.2 Sensor Arus INA219 10
Gambar 2.3 Alur eksekusi beban kerja pada perangkat bergerak ke perangkat komputasi awan [7] 11
Gambar 2.4 *Multi Criteria Decision Making* 13
Gambar 2.5 *Fuzzy Number* dan variabel linguistik [8]..... 13
Gambar 2.6 Grafik representasi *fuzzy number* [8]..... 14
Gambar 3.1 Arsitektur umum sistem 22
Gambar 3.2 Diagram alir sistem..... 23
Gambar 3.3 Diagram alir penghitungan jumlah mobil pada gambar 24
Gambar 3.4 Contoh data masukan citra yang digunakan sebagai datatest [3]..... 25
Gambar 3.5 Contoh data citra keluaran proses penghitungan jumlah mobil 26
Gambar 3.6 Fungsi keanggotaan metode lokal dan *offloading* kapasitas Raspberry Pi RAM yang tersedia 27
Gambar 3.7 Fungsi keanggotaan metode lokal dan *offloading download speed* 27
Gambar 3.8 Fungsi keanggotaan metode lokal dan *offloading upload speed*..... 28
Gambar 3.9 Fungsi keanggotaan metode lokal dan *offloading* jumlah data citra yang diproses 29
Gambar 3.10 Fungsi keanggotaan metode lokal dan *offloading* daya yang dikonsumsi 29
Gambar 5.1 Perangkat IoT 58
Gambar 5.2 Contoh data citra 1..... 59
Gambar 5.3 Contoh hasil pemrosesan data citra 1 59
Gambar 5.4 Contoh data citra 2..... 60
Gambar 5.5 Contoh hasil pemrosesan data citra 2 60
Gambar 5.6 Contoh data citra 3..... 61
Gambar 5.7 Contoh hasil pemrosesan data citra 3 61
Gambar 5.8 Contoh data citra 4..... 62

Gambar 5.9 Contoh hasil pemrosesan data citra 4 62
Gambar 5.10 Bagan alur kerja skenario uji coba 64
Gambar 5.11 Grafik waktu eksekusi beban kerja Skenario Uji Coba 1.....66
Gambar 5.12 Grafik konsumsi baterai beban kerja Skenario Uji Coba 1 67
Gambar 5.13 Grafik waktu eksekusi beban kerja Skenario Uji Coba 2.....70
Gambar 5.14 Grafik konsumsi baterai beban kerja Skenario Uji Coba 2 71
Gambar 5.15 Grafik waktu eksekusi beban kerja Skenario Uji Coba 3.....74
Gambar 5.16 Grafik konsumsi baterai beban kerja Skenario Uji Coba 3 76

DAFTAR TABEL

Tabel 4.1 Lingkungan Implementasi Perangkat Lunak.....33
Tabel 4.2 Inisialisasi *Fuzzy Number*.....37
Tabel 4.3 Hasil Konversi Nilai Faktor ke Variabel Linguistik....38
Tabel 4.4 Hasil Defuzzyfikasi Nilai Faktor Penentu Metode *Offloading*.....39
Tabel 4.5 Hasil *weight* setiap fitur untuk setiap alternatif pada setiap kriteria.....40
Tabel 4.6 Hasil *Entropy* dari setiap kriteria.....40
Tabel 4.7 Hasil *Variant Coefficient* dari setiap kriteria.....41
Tabel 4.8 Hasil *Entropy Weight* dari setiap kriteria 41
Tabel 4.9 Hasil normalisasi matriks *decision maker*.....41
Tabel 4.10 Hasil *weight* matriks *decision maker*.....42
Tabel 4.11 Hasil *weight* matriks *decision maker*.....42
Tabel 4.12 Hasil skor akhir 43
Tabel 5.1 Spesifikasi Lingkungan Pengujian 57
Tabel 5.2 Waktu eksekusi beban kerja dengan metode lokal dan *offloading* pada Skenario Uji Coba 1 65
Tabel 5.3 Konsumsi Baterai beban kerja dengan metode lokal dan *offloading* pada Skenario Uji Coba 1 66
Tabel 5.4 Waktu eksekusi beban kerja dengan *decision making* pada Skenario Uji Coba 1.....67
Tabel 5.5 Konsumsi baterai beban kerja dengan *decision making* pada Skenario Uji Coba 1.....68
Tabel 5.6 Waktu eksekusi beban kerja dengan metode lokal dan *offloading* pada Skenario Uji Coba 2 69
Tabel 5.7 Konsumsi Baterai beban kerja dengan metode lokal dan *offloading* pada Skenario Uji Coba 2 70
Tabel 5.8 Waktu eksekusi beban kerja dengan *decision making* pada Skenario Uji Coba 2.....72
Tabel 5.9 Konsumsi baterai beban kerja dengan *decision making* pada Skenario Uji Coba 2.....72
Tabel 5.10 Waktu eksekusi beban kerja dengan metode lokal dan *offloading* pada Skenario Uji Coba 3 74

Tabel 5.11 Konsumsi Baterai beban kerja dengan metode lokal dan *offloading* pada Skenario Uji Coba 3 75
Tabel 5.12 Waktu eksekusi beban kerja dengan *decision making* pada Skenario Uji Coba 3..... 76
Tabel 5.13 Konsumsi baterai beban kerja dengan *decision making* pada Skenario Uji Coba 3..... 77
Tabel 5.14 Hasil waktu eksekusi pada skenario uji coba 78
Tabel 5.15 Hasil konsumsi baterai pada skenario uji coba 79

DAFTAR KODE SUMBER

Pseudocode 4.1 Kode sumber untuk mendapatkan jumlah data citra yang diproses 35
Pseudocode 4.2 Kode sumber untuk mendapatkan kapasitas RAM yang tersedia 35
Pseudocode 4.3 Kode sumber untuk mendapatkan *download speed* dan *upload speed* 36
Pseudocode 4.4 Kode sumber untuk mendapatkan daya yang dikonsumsi dan tegangan baterai 37
Pseudocode 4.5 Kode sumber untuk inisialisasi *fuzzy number* ... 43
Pseudocode 4.6 Kode sumber untuk mengkonversi nilai faktor penentu *offloading* menjadi variabel linguistik 45
Pseudocode 4.7 Kode sumber untuk menghitung nilai defuzzyfikasi untuk metode lokal dan *offloading* 46
Pseudocode 4.8 Kode sumber untuk menghitung *weight* tiap fitur dan menghitung nilai *Entropy* tiap kriteria 47
Pseudocode 4.9 Kode sumber untuk menghitung *Variant Coefficient* dan *Entropy Weight* dari setiap kriteria 47
Pseudocode 4.10 Kode sumber untuk menghitung normalisasi matriks *decision maker* 48
Pseudocode 4.11 Kode sumber untuk menghitung *weight* matriks *decision maker* yang telah dinormalisasi 48
Pseudocode 4.12 Kode sumber untuk menghitung S^+ dan S^- 49
Pseudocode 4.13 Kode sumber untuk menghitung skor akhir metode lokal dan *offloading* 49
Pseudocode 4.14 Kode sumber implementasi komunikasi socket pada Perangkat IoT 51
Pseudocode 4.15 Kode sumber implementasi komunikasi socket pada *cloud server* 53
Pseudocode 4.16 Kode sumber untuk penghitungan jumlah mobil pada gambar 55

(Halaman ini sengaja dikosongkan)

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi komunikasi saat ini telah berkembang dengan pesat guna memenuhi kebutuhan hidup manusia. Perkembangan teknologi ini memunculkan berbagai konsep baru seperti *Internet of Things* (IoT). IoT merupakan konsep dimana perangkat yang terhubung dengan internet dapat bertukar data dan mempengaruhi perangkat atau objek fisik sekitarnya [1]. Contoh perangkat IoT Raspberry Pi. Raspberry Pi merupakan *Single Board Computer* (SBC) yang dapat menjalankan sistem operasi Raspbian dan ukurannya yang kecil dengan harga yang murah [2].

Tujuan awal dibuat Raspberry Pi adalah untuk membuat perangkat mudah dan murah, dari perangkat yang ada saat itu. Dan perangkat tersebut ditujukan untuk para siswa yang akan membuat perangkat desain dan interaksi. Berbagai macam *project* telah dibuat dengan menggunakan Raspberry Pi dengan tujuan mempermudah suatu kegiatan atau menyelesaikan permasalahan masyarakat dunia saat ini. Namun demikian, Raspberry Pi memiliki masalah tersendiri yang tidak bisa dihindari seperti sumber energi baterai yang memiliki daya tahan yang terbatas, memori yang terbatas, dan kemampuan komputasi yang terbatas. Kebanyakan *project* yang dioperasikan pada Raspberry Pi membutuhkan sumber daya yang besar seperti pengolahan citra gambar secara *high-definition*, *image recognition*, dan pengolahan data dari sensor yang dipasang pada Raspberry Pi.

Dalam perkembangannya, teknologi komputasi *cloud* secara *offloading* telah direncanakan sebagai teknologi yang menjanjikan untuk mengatasi keterbatasan pada perangkat. Mekanisme komputasi *cloud* secara *offloading* dianggap sebagai salah satu teknik yang paling baik dan sangat diperlukan yang berpotensi dalam menghemat energi untuk pengguna sebuah perangkat.

Namun, upaya penelitian saat ini untuk mekanisme *offloading* masih terbatas dan memiliki banyak kekurangan. Mekanisme *offloading* tidak dapat dilakukan jika perangkat tidak memiliki koneksi internet dan juga tidak dapat diterapkan pada beberapa proses komputasi yang harus dilakukan secara lokal. Keputusan dalam menentukan sebuah proses komputasi aplikasi dilakukan secara metode *offloading* atau secara lokal dipengaruhi oleh beberapa faktor-faktor yang menjadi tantangan bagaimana mengambil suatu keputusan dalam menentukan proses komputasi tersebut.

Saat membuat keputusan, *decision maker* selalu berusaha untuk memilih proses komputasi yang optimal antara metode lokal maupun metode *offloading.* Sayangnya proses komputasi yang optimal hanya terjadi jika hanya ada satu kriteria tunggal. Dalam situasi pengambilan suatu keputusan yang nyata hampir semua keputusan melibatkan beberapa konflik atau ketidakpuasan. Maka dari itu, diperlukan suatu metode *decision making* yang memilih proses komputasi yang optimal antara metode lokal maupun metode *offloading.* Pada tugas akhir ini diusulkan metode *Multi Criteria Decision Making* (MCDM) sebagai metode pengambilan suatu keputusan proses komputasi yang optimal. Metode MCDM digunakan untuk membantu membuat suatu keputusan yang sesuai dengan faktor-faktor *decision maker*, dalam kasus dimana terdapat lebih dari satu kriteria yang saling bertentangan, menemukan pilihan optimal di antara alternatif [3]. Metode tersebut akan diterapkan sebagai penentu proses komputasi yang optimal antara metode lokal maupun metode *offloading* pada lingkungan Raspberry Pi yang akan diteliti pada tugas akhir ini.

Hasil yang diharapkan adalah aplikasi Raspberry Pi (*client*) dapat menentukan sebuah komputasi dilakukan secara *offloading* pada server atau lokal pada kondisi tertentu secara optimal agar mendapatkan *execution time* paling minimal dan penghematan sumber daya baterai yang maksimal pada perangkat di sisi *client*.

1.2 Rumusan Masalah

Tugas akhir ini mengangkat beberapa rumusan masalah sebagai berikut:

1. Bagaimana menerapkan metode pembobotan dinamis pada penerapan *computation offloading* dengan metode *Fuzzy*-MCDM pada perangkat IoT?
2. Bagaimana menentukan penerapan *computation offloading* dengan metode *Fuzzy*-MCDM pada perangkat IoT dengan menggunakan faktor–faktor pendukung sebagai acuan (seperti daya baterai dan kecepatan *download*)?
3. Bagaimana menerapkan *computation offloading* dengan metode *Fuzzy*-MCDM pada perangkat IoT yang dapat meningkatkan performa proses komputasi?
4. Bagaimana melakukan ujicoba dan analisis kinerja dari *prototype computation offloading* pada perangkat IoT yang dikembangkan dengan studi kasus nyata?
5. Mengapa perlu adanya *Multi Criteria Decision Making* dalam proses *offloading*?

1.3 Batasan Permasalahan

Permasalahan yang dibahas pada tugas akhir ini memiliki batasan sebagai berikut:

1. Perangkat IoT yang digunakan adalah Raspberry Pi.
2. Data gambar yang digunakan untuk *image processing* adalah dataset dari Street parking Dataset [4].
3. Metode yang digunakan untuk meningkatkan kinerja proses komputasi adalah *computation offloading.*
4. Metode penentuan beban kerja diproses secara lokal atau secara *offloading* adalah metode *Fuzzy Multi Criteria Decision Making.*
5. Metode yang digunakan untuk pembobotan dinamis pada *Fuzzy*-MCDM adalah *Entropy*.
6. Library yang digunakan untuk pengolahan citra digital adalah OpenCV.

1.4 Tujuan

Tujuan dari tugas akhir ini adalah sebagai berikut:

1. Membangun sebuah *computation offloading* dengan metode *Fuzzy Multi Criteria Decision Making* pada perangkat IoT yang dapat menentukan secara dinamis dan optimal suatu beban kerja diproses secara *offloading* ke *server* atau secara lokal berdasarkan faktor–faktor pendukung *computation offloading*.
2. Melakukan implementasi komputasi *offloading* pada perangkat IoT untuk peningkatan kinerja proses komputasi suatu beban kerja dan penghematan konsumsi baterai pada perangkat IoT.

1.5 Manfaat

Dengan dibuatnya tugas akhir ini diharapkan dapat memberikan manfaat untuk menghasilkan sebuah *computation offloading* yang dapat menentukan secara dinamis pemrosesan suatu beban kerja yang optimal dengan tujuan untuk memaksimalkan performa komputasi agar memiliki waktu eksekusi seminimal mungkin.

1.6 Metodologi

Pembuatan tugas akhir ini dilakukan dengan menggunakan metodologi sebagai berikut :

1.6.1 Penyusunan Proposal

Tahap awal tugas akhir ini adalah menyusun proposal tugas akhir. Pada proposal, diajukan gagasan untuk menerapkan komputasi *offloading* sebagai solusi dari keterbatasan seperti keterbatasan *processor* yang memproses komputasi pada setiap beban kerja perangkat lunak.

1.6.2 Studi Literatur

Pada tahap ini dilakukan untuk mencari informasi dan studi literatur apa saja yang dapat dijadikan referensi untuk membantu pengerjaan Tugas Akhir ini. Informasi didapatkan dari buku dan literatur yang berhubungan dengan *computation offloading* yang akan dibangun dan metode untuk pengerjaan *clustering*. Informasi yang dicari adalah implementasi *computation offloading* pada perangkat, faktor–faktor yang digunakan untuk menentukan dilakukannya *computation offloading*, dan metode-metode pengerjaan *clustering.*

1.6.3 Analisis dan Desain Perangkat Lunak

Pada tahap ini akan dilakukan analisa, perancangan, dan pendefinisian kebutuhan sistem untuk mengetahui permasalahan yang akan dihadapi pada tahap implementasi. Kemudian akan dijabarkan kebutuhan–kebutuhan tersebut ke dalam perancangan fitur sistem. Berikut langkah yang akan dilakukan dalam perancangan proses perangkat lunak:

1. Instalasi *library* pada Raspberry Pi.
2. Instalasi *library* pada *cloud server*.
3. Perancangan komunikasi antara perangkat IoT dan *cloud server*.
4. Perancangan metode *Fuzzy Multi Criteria Decision Making* pada Raspberry Pi.
5. Perancangan metode pembobotan dinamis pada Raspberry Pi.

1.6.4 Implementasi Perangkat Lunak

Implementasi merupakan tahap untuk membangun metode-metode yang sudah diajukan pada proposal Tugas Akhir. Implementasi dilakukan dengan menggunakan suatu perangkat lunak yaitu Visual Studio Code sebagai *Text Editor* dan *debugger*.

1.6.5 Pengujian dan Evaluasi

Pada tahap ini perangkat lunak yang telah dibangun diuji coba dengan menggunakan data uji coba yang ada. Data uji coba tersebut diuji coba pada perangkat lunak Raspberry Pi sebagai client dan cloud server dengan tujuan mengetahui kemampuan perangkat lunak dalam menentukan penggunaan metode *offloading* atau tidaknya (lokal) pada suatu beban kerja berdasarkan faktor–faktor pendukung yang terjadi saat itu pada perangkat *client* dan mengevaluasi hasil tugas akhir dengan jurnal pendukung yang ada. Hasil evaluasi mencakup waktu eksekusi dari kemampuan perangkat lunak tersebut dalam menentukan *computation offloading* suatu beban kerja pada Raspberry Pi.

1.6.6 Penyusunan Buku

Pada tahap ini disusun buku sebagai dokumentasi dari pelaksanaan tugas akhir yang mencangkup seluruh konsep, teori, implementasi, serta hasil yang telah dikerjakan.

1.7 Sistematika Penulisan Laporan

Sistematika penulisan laporan tugas akhir adalah sebagai berikut:

1. Bab I. Pendahuluan
   Bab ini berisikan penjelasan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika penulisan dari pembuatan tugas akhir.
2. Bab II. Tinjauan Pustaka
   Bab ini berisi kajian teori dari metode dan algoritma yang digunakan dalam penyusunan Tugas Akhir ini. Secara garis besar, bab ini berisi tentang Raspberry Pi, sistem operasi Raspbian, Sensor Arus INA219, metode *offloading*, faktor – faktor penentu dilakukannya metode *offloading*, pembobotan dinamis *Entropy*, socket sebagai metode pengiriman data, serta

MobileNets dan OpenCV digunakan dalam penerapan *image processing* pada citra.

3. Bab III. Perancangan Sistem
   Bab ini berisi pembahasan mengenai perancangan dari *computation offloading* dengan pembobotan dinamis yang diterapkan pada Raspberry Pi untuk menentukan dilakukannya metode *offloading* pada suatu beban kerja saat melakukan *image processing* berdasarkan faktor – faktor tertentu.
4. Bab IV. Implementasi
   Bab ini menjelaskan implementasi yang berbentuk *Pseudocode* yang berupa *Pseudocode* dari *offloading computation* dengan pembobotan dinamis dari sistem perangkat lunak *image recognition* beserta penerapan komunikasi antar server dan client.

5. Bab V. Hasil Uji Coba dan Evaluasi
   Bab ini berisikan hasil uji coba dari *computation offloading* yang digunakan untuk menentukan dilakukannya metode *offloading* pada suatu beban kerja dengan pembobotan dinamis dari sistem perangkat lunak *image processing* yang sudah diimplementasikan pada *Pseudocode*. Uji coba dilakukan dengan menggunakan datatest dan dataset citra yang memiliki kualitas rendah. Hasil evaluasi mencakup kedinamisan dari kemampuan *computation offloading* dalam menentukan dilakukannya metode *offloading* pada beban kerja berdasarkan faktor – faktor penentu yang di alami perangkat client dan server saat itu dengan pembobotan dinamis.
6. Bab VI. Kesimpulan dan Saran
   Bab ini merupakan bab yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan, masalah-masalah yang dialami pada proses pengerjaan Tugas Akhir, dan saran untuk pengembangan solusi ke depannya.
7. Daftar Pustaka
   Bab ini berisi daftar pustaka yang dijadikan literatur dalam tugas akhir.
8. Lampiran
   Dalam lampiran terdapat tabel-tabel data hasil uji coba dan *Pseudocode* program secara keseluruhan.

BAB II
TINJAUAN PUSTAKA

Bab ini berisi pembahasan mengenai teori-teori dasar yang digunakan dalam tugas akhir. Teori-teori tersebut diantaranya adalah Raspberry Pi, sistem operasi Raspbian, sensor arus INA219, metode *offloading*, faktor–faktor penentu dilakukannya metode *offloading*, pembobotan dinamis *Entropy*, socket dan beberapa teori lain yang mendukung pembuatan tugas akhir.

2.1 Raspberry Pi

Raspberry Pi adalah komputer *single-board circuit* yang dapat digunakan untuk menjalankan program perkantoran, permainan, dan pemutar media beresolusi tinggi [5]. Raspberry Pi memiliki dua model yaitu model A dan model B [5]. Perbedaan model A dan model B terletak pada penyimpanan yang digunakan, model A sebesar 256 MB dan model B sebesar 512 MB [5].

Gambar 2.1 Raspberry Pi 1 Model B

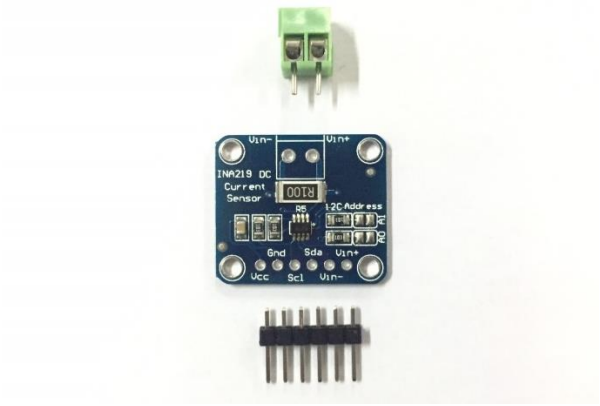
2.2 Sistem Operasi Raspbian

Sistem operasi Raspbian adalah sistem operasi berbasis Debian yang dikhususkan untuk perangkat Raspberry Pi [6].

Raspbiian memiliki beberapa seri diantaranya Raspbian Stretch dan Raspbian Jessie [6].

2.3 Sensor Arus INA219

INA219 adalah *current shunt* dan *power monitor* dengan antarmuka yang kompatibel dengan I2C atau SMBUS. Alat ini memonitor penurunan *shunt voltage* dan *bus supply voltage*, dengan waktu konversi dan penyaringan yang dapat diprogram. Nilai kalibrasi yang dapat diprogram, dikombinasikan dengan pengali internal, memungkinkan mendapatkan nilai arus dalam bentuk ampere. Register pengali tambahan menghitung daya dalam bentuk watt. Antarmuka yang kompatibel dengan I2C - atau SMBUS menampilkan 16 *addresses* yang dapat diprogram [7].



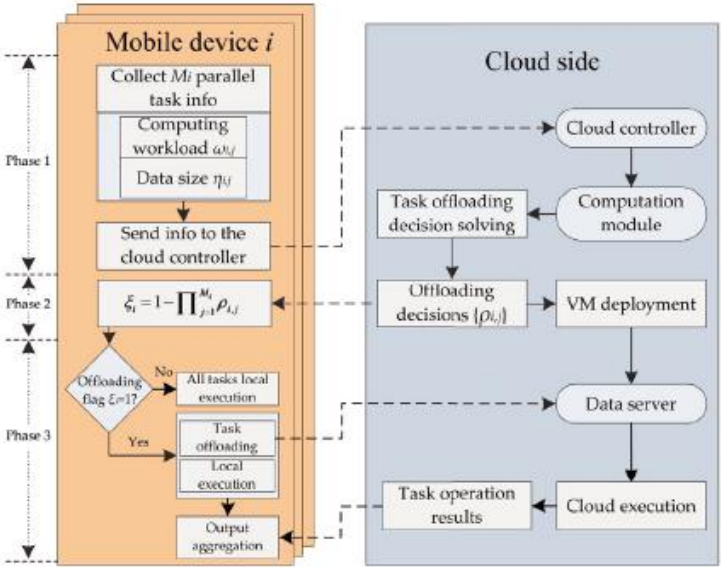
Gambar 2.2 Sensor Arus INA219

2.4 *Internet of Things (IoT)*

IoT merupakan konsep dimana perangkat yang terhubung dengan internet dapat bertukar data dan mempengaruhi perangkat atau objek fisik sekitarnya [1]. IoT adalah internet dari 3 hal diantaranya manusia dengan manusia, manusia dengan mesin, dan

mesin dengan mesin [1]. Contoh perangkat IoT adalah Raspberry Pi.

2.5 *Offloading*



Gambar 2.3 Alur eksekusi beban kerja pada perangkat bergerak ke perangkat komputasi awan [8]

Metode *Offloading* telah menjadi teknik yang menjanjikan untuk memecahkan masalah yang dihadapi perangkat *smartphone*, yaitu dengan memungkinkan *smartphone* melakukan metode *offload* atau mengirimkan suatu beban kerja komputasi yang intensif ke server [8]. Berbagai upaya telah dilakukan untuk menerapkan metode *offload* ke *server* untuk memperoleh keuntungan dari kapabilitas *crossplatform bytecode* [8]. Contoh alur eksekusi beban kerja pada penerapan metode *offloading* yang

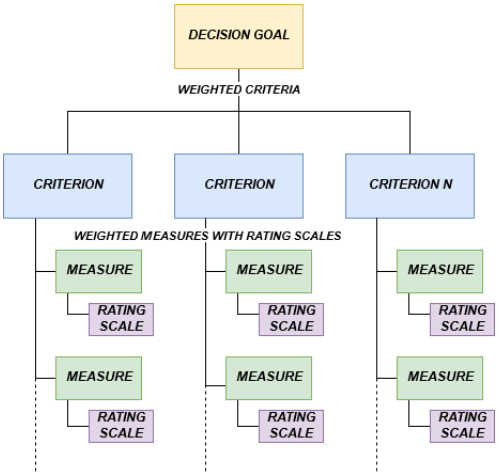
diimplementasikan pada perangkat bergerak dapat dilihat pada Gambar 2.3.

2.6 Faktor Penentu Metode *Offloading*

Keputusan dalam menentukan dilakukannya metode *Offloading* adalah tantangan. Hal ini bergantung pada *volume* data dan transmisi *bandwidth*, konteks yang di eksekusi dari beban kerja komputasi, perbandingan dalam kemampuan eksekusi antara SoC (*system on chip*) *smartphone* dan *processor server* serta seberapa efek keuntungan jika melakukan metode *Offloading* [8]. Membuat data rinci sebuah beban kerja komputasi dan pemantauan jaringan *quality-of-service* (QoS) sangat penting dalam *framework*. Selain itu, karena komputasi dan karakteristik komunikasi adalah dinamis, sebuah *daemon* yang secara periodik mengukur kapabilitas komputasi, performa komunikasi, konsumsi daya dan secara dinamis membuat keputusan dilakukannya metode *offloading* untuk setiap *task* yang memenuhi syarat untuk dilakukannya *offloading*, hal ini dibutuhkan dalam *framework* [8].

2.7 *Fuzzy Multi Criteria Decision Making*

Multi Criteria Decision Making (MCDM) adalah suatu metode pengambilan keputusan untuk menetapkan alternatif terbaik dari sejumlah alternatif berdasarkan beberapa kriteria tertentu. Salah satu jenis MCDM yaitu *Fuzzy Multi Criteria Decision Making*. Metode ini dikembangkan untuk membantu pengambil keputusan dalam melakukan pengambilan keputusan terhadap beberapa alternatif keputusan untuk mendapatkan suatu keputusan yang akurat dan optimal [3]. *Fuzzy Multi Criteria Decision Making* adalah salah satu metode yang bisa membantu pengambil keputusan dalam melakukan pengambilan keputusan terhadap beberapa alternatif keputusan yang harus diambil dengan beberapa kriteria yang akan menjadi bahan pertimbangan [3]. Secara umum MCDM digambarkan pada Gambar 2.4



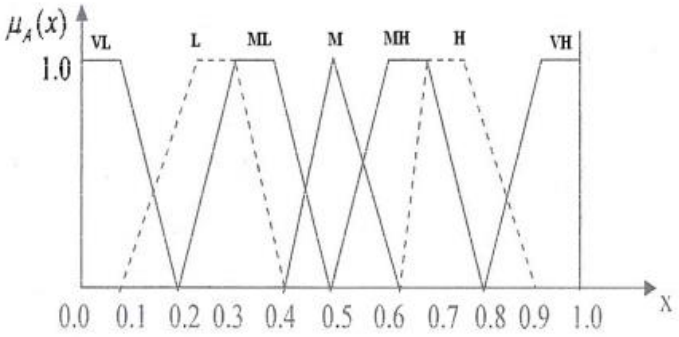
Gambar 2.4 *Multi Criteria Decision Making* [3]

Fuzzy Multi Criteria Decision Making diawali dengan inisialisasi *fuzzy number* yang ditunjukkan pada Gambar 2.5.

Linguistic variable	Fuzzy number
Very high (VH)	(0.8,0.9,1,1)
High	(0.6,0.7,0.8,0.9)
Moderate high	(0.5,0.6,0.7,0.8)
Moderate	(0.4,0.5,0.5,0.6)
Moderate low	(0.2,0.3,0.4,0.5)
Low	(0.1,0.2,0.3,0.4)
Very low	(0.0,0.0,0.1,0.2)

Gambar 2.5 *Fuzzy Number* dan variabel linguistik [3]

Fuzzy Number direpresentasikan pada grafik yang ditunjukkan pada Gambar 2.6.



Gambar 2.6 Grafik representasi *fuzzy number* [3]

Langkah 2: Setiap kriteria akan diberikan variabel linguistik dan *fuzzy number* dimasukkan ke masing-masing kriteria. Lalu dihitung rata-ratanya dengan persamaan berikut [3].

$$\boldsymbol{A_{ij}} = \frac{\mathbf{1}}{\boldsymbol{p}} \oplus \left(\boldsymbol{a_{i1}^{j}} \oplus \boldsymbol{a_{i2}^{j}} + \ldots + \boldsymbol{a_{ik}^{j}}\right); \quad K = 1, 2, \ldots, P \tag{2.1}$$

Langkah 3: Menghitung nilai defuzzyfikasi dari setiap kriteria dengan rumus yang ditunjukkan pada persamaan berikut [3].

$$e = \frac{(a + b + c + d)}{4} \tag{2.2}$$

Langkah 4: Menghitung normalisasi *weight* dari masing-masing kriteria dengan membagi nilai defuzzyfikasi kriteria dengan total jumlah kriteria yang ada.

Langkah 5: Mengkonversi nilai-nilai faktor penentu menjadi variabel linguistik untuk masing-masing alternatif.

Langkah 6: Menghitung skor akhir dari masing-masing alternatif dengan mengalikan nilai defuzzyfikasi dengan nilai *weight* dari masing-masing kriteria yang ditunjukkan pada persamaan berikut [3].

$$TS = [Xij]\,[Wj] \tag{2.3}$$

Langkah 7: Membandingkan nilai skor akhir dari masing-masing alternatif. Alternatif yang memiliki nilai skor akhir yang paling besar adalah alternatif yang dipilih.

2.8 *Entropy*

Entropy adalah sebuah *decision tree* yang dibangun dengan skema *top-down* dari *root node* dan melibatkan partisi data menjadi Himpunan bagian yang berisi variabel-variabel dengan nilai yang sama (homogen) [9]. Algoritma ID3 (*Iterative Dichotomiser 3*) menggunakan *Entropy* untuk menghitung homoginetas dari sampel yang ada. Jika sampel benar-benar homogen, *Entropy* akan mengeluarkan nilai nol dan jika sampel dibagi rata maka *Entropy* akan mengeluarkan nilai satu. [9]. *Entropy* memiliki rumus untuk menentukan nilai *Entropy* jika menggunakan satu atribut yang ditunjukkan pada persamaan 2.4 dan rumus untuk menentukan nilai *Entropy* jika menggunakan dua atribut yang ditunjukkan pada persamaan 2.5.

$$E(S) = \sum_{i=1}^{c} -p_i \log_2 p_i \tag{2.4}$$

$$E(T,X) = \sum_{c \in X} P(c)E(c) \tag{2.5}$$

2.9 *Entropy Based Fuzzy* MCDM

MCDM adalah metode yang sangat praktisi di dunia nyata, dan memiliki efek yang sangat signifikan pada teori dan praktisi. Tujuan MCDM ini tidak lain adalah untuk menemukan alternatif terbaik di antara kriteria – kriteria yang telah diberikan [9]. Penghitungan *weight* dengan metode *Entropy* dilakukan sebagai berikut.

Langkah 1: Menghitung nilai defuzzyfikasi dari setiap kriteria dengan rumus yang ditunjukkan pada persamaan (2.2) [3].

Langkah 2: Menghitung *weight* setiap fitur (P_{ij}) untuk setiap alternatif ke- i dan setiap kriteria ke- j dengan rumus yang ditunjukkan pada persamaan berikut [9].

$$P_{ij} = \frac{a_{ij}}{\sum_{i=1}^{m} a_{ij}^2}, (1 \leq i \leq m,\ 1 \leq j \leq n) \tag{2.6}$$

Nilai m adalah total *alternatives* dan nilai n adalah total kriteria.

Langkah 3: Menghitung nilai *Entropy* dari setiap kriteria ke-j dengan rumus yang ditunjukkan pada persamaan berikut [9].

$$e_j = -k \sum_{i=1}^{m} (P_{ij} \ln P_{ij}), (1 \leq j \leq n)$$
$$k = {}^{1}/_{(\ln m)} \tag{2.7}$$

Langkah 4: Menghitung *variant coefficient* dari setiap kriteria ke- j dengan rumus yang ditunjukkan pada persamaan berikut [9].

$$g_j = \left|1 - e_j\right|, (1 \leq j \leq n) \quad (2.8)$$

Langkah 5: Menghitung *weight* dari nilai *Entropy* yang sudah masuk ke langkah 4 dengan rumus yang ditunjukkan pada persamaan berikut [9].

$$W_j = \frac{g_j}{\sum_{i=1}^{m} g_j}, (1 \leq j \leq n) \quad (2.9)$$

Setelah mendapatkan nilai *weight* dengan metode *Entropy* maka langkah selanjutnya adalah mendapatkan nilai akhir dengan metode *Fuzzy MCDM* yang dapat dilakukan sebagai berikut.

Langkah 1: Menghitung normalisasi matriks *decision maker* dengan rumus yang ditunjukkan pada persamaan berikut [9].

$$X_{ij} = \frac{a_{ij}}{\sum_{i=1}^{m} a_{ij}} \quad (2.10)$$

Langkah 2: Menghitung *weight* matriks *decision maker* yang telah dinormalisasi dengan rumus yang ditunjukkan pada persamaan berikut [9].

$$W_{ij} = w_j * X_{ij} \quad (2.11)$$

Langkah 3: Menghitung S^+ dan S^- dengan rumus yang ditunjukkan pada persamaan 2.12 [9] dan persamaan 2.13 [9].

$$S_i^+ = \sum_{j=1}^{n} W_{ij}, (1 = 1,2,3 \ldots m) \qquad (2.12)$$

Dimana W_{ij} adalah *weight* matriks *decision maker* yang bersifat menguntungkan pada kriteria ke- j

$$S_i^- = \sum_{j=1}^{n} W_{ij}, (1 = 1,2,3 \ldots m) \qquad (2.13)$$

Dimana W_{ij} adalah *weight* matriks *decision maker* yang bersifat merugikan pada kriteria ke- j.

Langkah 4: Menghitung nilai skor akhir *Entropy Based Fuzzy* MCDM dengan rumus yang ditunjukkan pada persamaan berikut [9].

$$Q_i = (S_i^+) + \frac{\sum_{i=1}^{m} S_i^-}{S_i^- \sum_{i=1}^{m} \frac{1}{S_i^-}} \qquad (2.14)$$

Langkah 5: Membandingkan skor akhir dari metode lokal dan metode *offloading*, jika skor akhir metode lokal lebih besar maka metode tersebut yang digunakan dan begitu juga sebaliknya.

2.10 OpenCV

*Open*CV (*Open Source Computer Vision*) adalah *library* yang utamanya digunakan untuk pemrosesan visi komputer. *Open*CV adalah *library* gratis yang dapat digunakan di berbagai *platform*, seperti GNU/Linux maupun Windows. *Open*CV mulanya ditulis dalam bahasa pemrograman C++, namun saat ini

*Open*CV dapat digunakan pada berbagai bahasa seperti Python, Java, atau MATLAB [10].

2.11 MobileNets

MobileNets adalah *class* model yang efisien digunakan untuk perangkat *mobile* dan perangkat *embedded*. MobileNets digunakan untuk *object detection*. MobileNets menggunakan *width multiplier* dan *resolution multiplier* untuk mengurangi *size* dan latency [11].

(*Halaman ini sengaja dikosongkan*)

BAB III
PERANCANGAN SISTEM

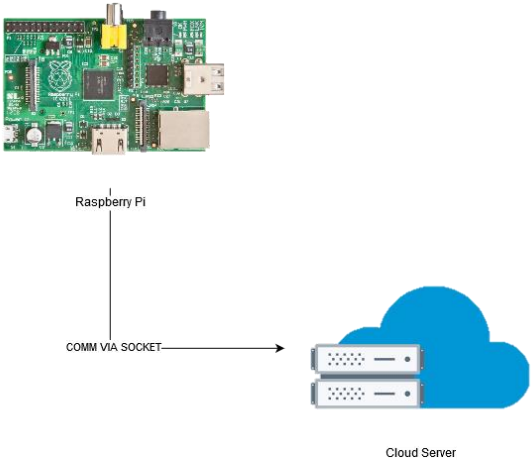
Bab ini membahas mengenai perancangan dan pembuatan sistem. Perangkat lunak pada perangkat IoT yang dibuat pada tugas akhir ini adalah menentukan keputusan dilakukannya *computation offloading* dengan metode *Entropy Based Fuzzy* MCDM berdasarkan faktor-faktor penentu yang mendukung metode *offloading*.

3.1 Deskripsi Umum Sistem

Pada sub bab ini dijelaskan mengenai deskripsi umum sistem yang dibangun. Sistem yang dibangun adalah sistem yang menerapkan *computation offloading* dengan metode *Entropy Based Fuzzy* MCDM pada perangkat IoT yang terdiri dari Raspberry Pi. Studi kasus yang digunakan adalah penghitungan jumlah mobil pada gambar dengan menggunakan data citra dari Street Parking Dataset [4]. Raspberry Pi menentukan apakah beban kerja diproses secara lokal atau *offloading* dengan metode *Entropy Based Fuzzy* MCDM. *Fuzzy* MCDM menggunakan nilai RAM (*Random Access Memory*) Raspberry Pi yang tersedia, *download speed*, *upload speed*, jumlah data citra yang diproses, dan daya yang dikonsumsi untuk menentukan apakah beban kerja diproses secara lokal atau *offloading*. Jika diproses secara lokal maka data citra akan diproses di Raspberry Pi, jika diproses secara *offloading* maka data citra akan dikirimkan ke *cloud server* lalu diproses.

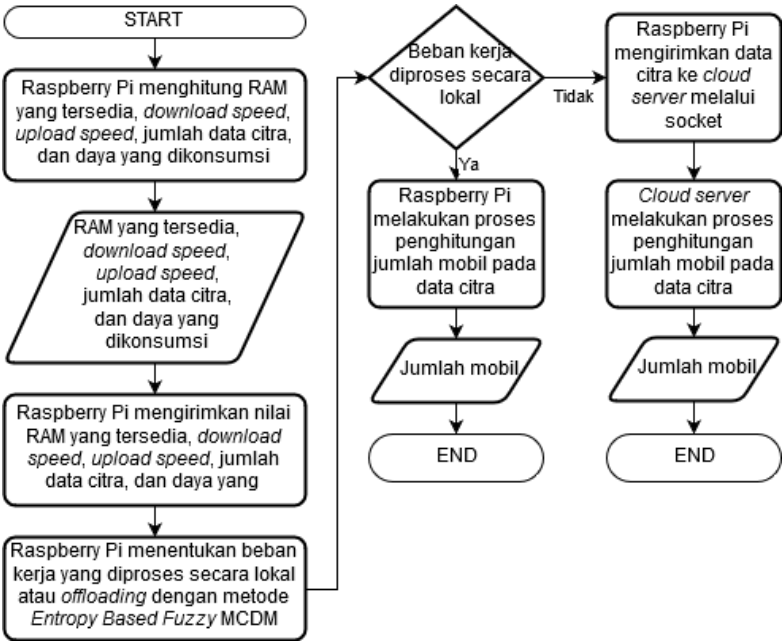
3.2 Arsitektur Umum Sistem

Pada sub bab ini dijelaskan mengenai arsitektur umum sistem yang dibangun, sistem terdiri dari perangkat IoT (Raspberry Pi) dan *cloud server*. Komunikasi antara perangkat IoT dan *cloud server* menggunakan socket. Arsitektur umum sistem ditunjukkan pada Gambar 3.1.



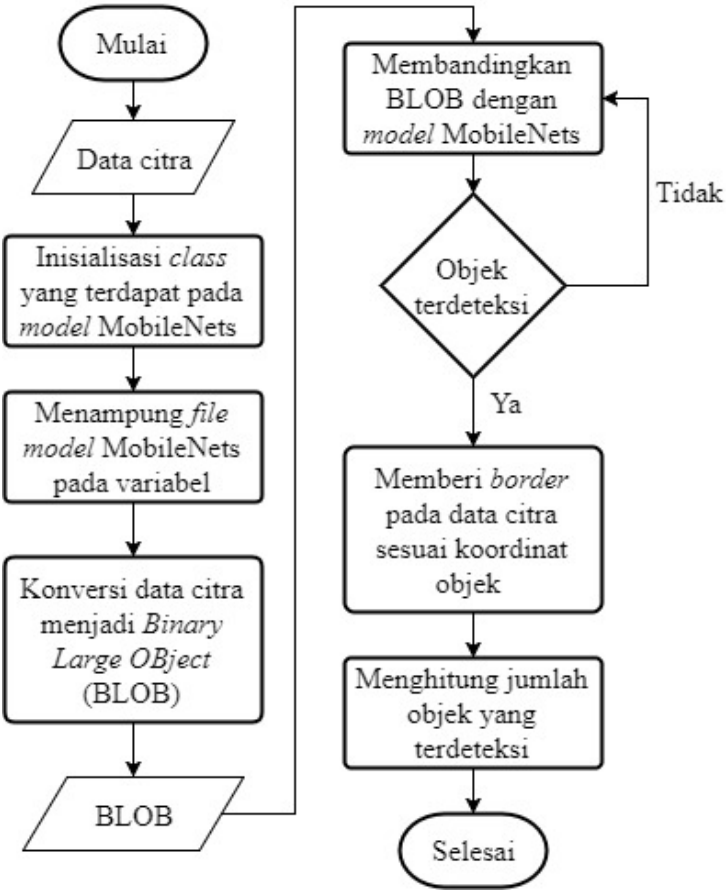
Gambar 3.1 Arsitektur umum sistem

Sistem dimulai dengan Raspberry Pi menghitung nilai RAM yang tersedia, *download speed*, dan *upload speed*, jumlah data citra yang akan diproses, dan daya yang dikonsumsi. Setelah itu, Raspberry Pi menentukan apakah beban kerja akan diproses secara lokal atau secara *offloading* dengan metode *Entropy Based Fuzzy* MCDM yang menggunakan faktor-faktor diantaranya RAM Raspberry Pi yang tersedia, *download speed*, *upload speed*, jumlah data citra yang akan diproses, dan daya yang dikonsumsi Raspberry Pi. Jika beban kerja diproses secara lokal maka Raspbery Pi melakukan proses penghitungan jumlah mobil pada gambar. Sedangkan jika beban kerja diproses secara *offloading* maka Raspberry Pi akan mengirimkan data citra ke *cloud server* lalu *cloud server* akan melakukan proses penghitungan jumlah mobil pada gambar dan mengirimkan hasilnya ke Raspberry Pi melalui *socket*. Alur kerja sistem ditunjukkan pada Gambar 3.2.



Gambar 3.2 Diagram alir sistem

Proses penghitungan jumlah mobil pada gambar dimulai dengan mengakses *file model* MobileNets, lalu data citra dikonversi menjadi *Binary Large OBject* (BLOB). Hasil konversi dibandingkan dengan *model* MobileNets, jika objek sudah terdeteksi maka data citra akan diberi penanda sesuai dengan lokasi objek yang terdeteksi. Proses penghitungan jumlah mobil ditunjukkan pada Gambar 3.3.



Gambar 3.3 Diagram alir penghitungan jumlah mobil pada gambar

3.3 Data

Pada sub bab ini dijelaskan mengenai data yang digunakan sebagai masukan pada perangkat IoT untuk selanjutnya diolah secara lokal atau secara *offloading* sehingga menghasilkan data

keluaran yang diharapkan dengan waktu pemrosesan seminimal mungkin.

3.3.1 Data Masukan

Data masukan adalah data yang digunakan sebagai masukan awal dari sistem. Data masukan berupa citra yang akan digunakan sebagai datatest untuk uji coba perhitungan jumlah mobil.

Contoh citra sebagai data masukan datatest ditunjukkan pada Gambar 3.4.

Gambar 3.4 Contoh data masukan citra yang digunakan sebagai datatest [4]

3.3.2 Data Keluaran

Data masukan akan dilakukan *image processing* dengan menggunakan model MobileNets dan memulai proses deteksi

objek dengan mengubah data masukan menjadi *Binary Large OBject* (BLOB) lalu membandingkannya dengan model yang sudah dimuat sebelumnya. Hasil dari proses *image processing* adalah jumlah mobil dan citra yang sudah diperbaharui dengan posisi mobil yang terdeteksi. Contoh data keluaran ditunjukkan pada Gambar 3.5.

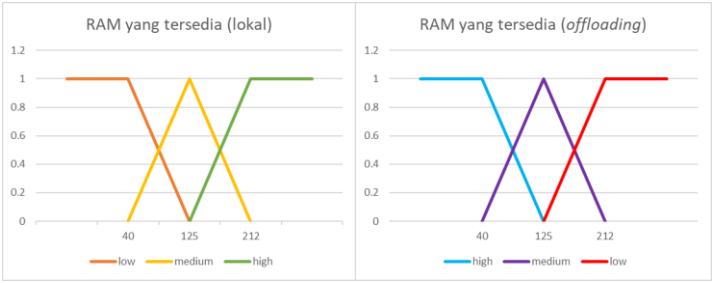
Gambar 3.5 Contoh data citra keluaran proses penghitungan jumlah mobil

3.4 Faktor Penentu Metode *Offloading*

3.4.1 Kapasitas Raspberry Pi RAM yang tersedia

Data yang pertama adalah kapasitas Raspberry Pi RAM yang tersedia yang terbagi menjadi 3 kelas urut dari yang buruk ke baik yaitu *Low*, *Medium*, dan *High*. Data ini diperoleh dengan mengambil RAM Raspberry Pi yang tersedia melalui *bash*

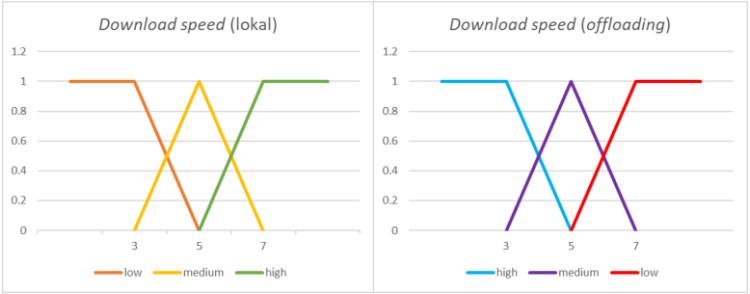
command. Penentuan kelas terbagi menjadi 2 macam. Penentuan kelas untuk metode lokal dan *offloading* menggunakan fungsi keanggotaan yang ditunjukkan pada Gambar 3.6.



Gambar 3.6 Fungsi keanggotaan metode lokal dan *offloading* kapasitas Raspberry Pi RAM yang tersedia

3.4.2 *Download Speed*

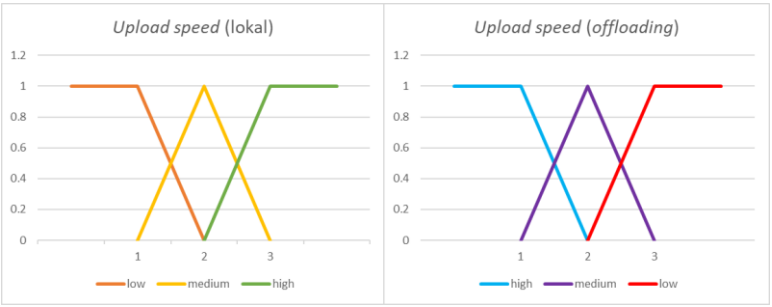
Data yang kedua adalah *download speed* dari Perangkat IoT yang terbagi menjadi 3 kelas urut dari yang buruk ke baik yaitu *Low*, *Medium*, dan *High*. Data ini diperoleh dengan *library speedtest* agar kecepatan *download* diperoleh. Penentuan kelas terbagi menjadi 2 macam. Penentuan kelas untuk metode lokal dan *offloading* menggunakan fungsi keanggotaan yang ditunjukkan pada Gambar 3.7.



Gambar 3.7 Fungsi keanggotaan metode lokal dan *offloading download speed*

3.4.3 *Upload Speed*

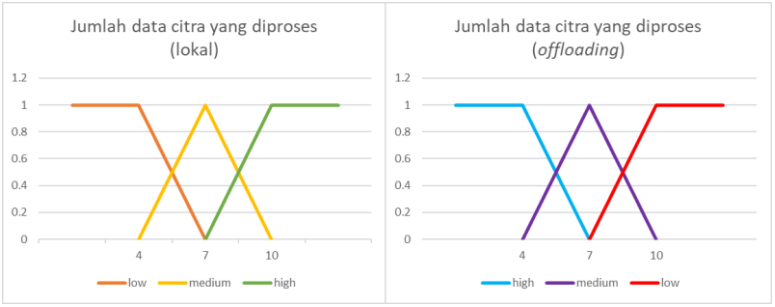
Data yang ketiga adalah *upload speed* dari Perangkat IoT yang terbagi menjadi 3 kelas urut dari yang buruk ke baik yaitu *Low*, *Medium*, dan *High*. Data ini diperoleh dengan *library speedtest* agar kecepatan *upload* diperoleh. Penentuan kelas terbagi menjadi 2 macam. Penentuan kelas untuk metode lokal dan *offloading* menggunakan fungsi keanggotaan yang ditunjukkan pada Gambar 3.8.



Gambar 3.8 Fungsi keanggotaan metode lokal dan *offloading* *upload speed*

3.4.4 Jumlah Data Citra yang diproses

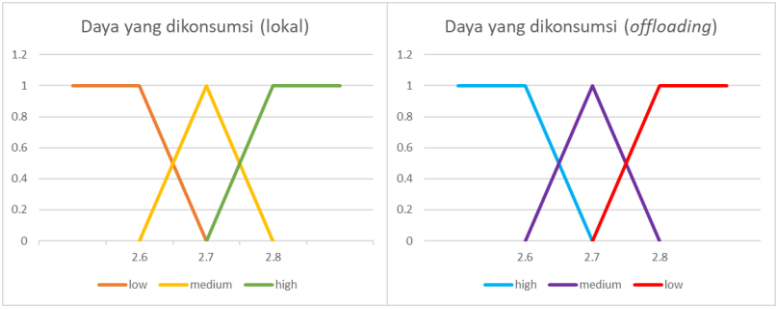
Data yang keempat adalah jumlah data citra yang diproses yang terbagi menjadi 3 kelas urut dari yang buruk ke baik yaitu *Low*, *Medium*, dan *High*. Penentuan kelas terbagi menjadi 2 macam. Penentuan kelas untuk metode lokal menggunakan fungsi keanggotaan yang ditunjukkan pada Gambar 3.9.



Gambar 3.9 Fungsi keanggotaan metode lokal dan *offloading* jumlah data citra yang diproses

3.4.5 Daya yang dikonsumsi

Data yang kelima adalah daya yang dikonsumsi dari Perangkat IoT yang terbagi menjadi 3 kelas urut dari yang buruk ke baik yaitu *Low*, *Medium*, dan *High*. Penentuan kelas terbagi menjadi 2 macam. Penentuan kelas untuk metode lokal menggunakan fungsi keanggotaan yang ditunjukkan pada Gambar 3.10.



Gambar 3.10 Fungsi keanggotaan metode lokal dan *offloading* daya yang dikonsumsi

3.5 *Offloading* dengan Metode *Entropy Based Fuzzy* MCDM

Offloading merupakan salah satu metode untuk peningkatan kinerja proses komputasi pada perangkat dengan cara melakukan *offload* beban kerja dari perangkat ke *cloud server*. Metode ini sangat mendukung untuk diimplementasikan sebab konektivitas internet yang bertindak sebagai *virtual bus* dapat diperoleh atau didapatkan dimana saja dan kapan saja saat ini. Studi kasus pada tugas akhir ini, *cloud server* tidak dibatasi oleh sumber daya baterai dan memiliki *processor* dengan performa lebih cepat dibandingkan dengan perangkat IoT sebagai *client* sehingga apabila melakukan *offloading* beban kerja ke *cloud server*, diharapkan eksekusi operasi terhadap beban kerja lebih cepat dan meminimalisir waktu eksekusi.

Pada tugas akhir ini, beban kerja yang akan di *offload* ke *cloud server* sudah ditentukan yaitu *image processing* dengan studi kasus penghitungan jumlah mobil pada gambar. Dalam memulai eksekusi *offload* beban kerja pada *cloud server*, ada beberapa hal yang harus dilakukan sebelumnya agar hasil dari eksekusi *offload* beban kerja ke *cloud server* maksimal dan sesuai yang diharapkan. Identifikasi beban kerja, kondisi Perangkat IoT *client* dan pemantauan kualitas jaringan internet sangat penting dalam metode ini. Kondisi Perangkat IoT seperti kapasitas RAM yang tersedia, *download speed*, *upload speed*, jumlah data citra yang akan diproses, dan daya yang dikonsumsi. Selanjutnya, hal–hal yang dapat mempengaruhi pengambilan keputusan metode eksekusi kita sebut sebagai faktor–faktor penentu metode *offloading* dan fungsi yang digunakan untuk menentukan keputusan metode eksekusi disebut dengan *decision maker*.

Decision maker adalah fungsi yang dipanggil setiap beban kerja yang telah ditentukan pada Perangkat IoT akan dieksekusi. Data faktor–faktor penentu metode *offloading* akan digunakan sebagai catatan histori performa oleh *decision maker* untuk memperkirakan dan membandingkan biaya dari kedua metode eksekusi tersebut sebelum memutuskan metode eksekusi yang

akan dipilih. Pada perancangan tugas akhir ini, *decision maker* melakukan pengecekan data terhadap kapasitas RAM yang tersedia pada Perangkat IoT, *download speed, upload speed*, jumlah data citra yang akan diproses, dan daya yang dikonsumsi.

Decision maker pada tugas akhir ini menggunakan metode *Entropy Based Fuzzy* MCDM. Langkah-langkah *Entropy Based Fuzzy* MCDM pada tugas akhir ini adalah sebagai berikut:

1. Inisialisasi *fuzzy number* dari masing-masing variabel linguistik (*Low, Medium,* High)
2. Mengkonversi nilai-nilai faktor penentu menjadi variabel linguistik (*Low, Medium, High*) untuk metode lokal dan metode *offloading* (kapasitas RAM yang tersedia, *download speed, upload speed*, jumlah data citra yang diproses, dan daya yang dikonsumsi) dengan menggunakan fungsi keanggotaan dari masing-masing faktor.
3. Menghitung nilai defuzzyfikasi dari tiap faktor penentu untuk metode lokal dan metode *offloading*
4. Menghitung *weight* setiap fitur pada setiap alternatif dan setiap kriteria
5. Menghitung nilai *Entropy* dari setiap kriteria
6. Menghitung *variant coefficient* dari setiap kriteria
7. Menghitung *weight* dari nilai *Entropy* berdasarkan *variant coefficient*
8. Menghitung normalisasi matriks *decision maker*
9. Menghitung *weight* matriks *decision maker* yang telah dinormalisasi
10. Menghitung S^+ dan S^-
11. Menghitung nilai skor akhir *Entropy Based Fuzzy* MCDM
12. Membandingkan skor akhir dari metode lokal dan metode *offloading*, jika skor akhir metode lokal lebih besar maka metode tersebut yang digunakan dan begitu juga sebaliknya

(*Halaman ini sengaja dikosongkan*)

BAB IV
IMPLEMENTASI

Bab ini berisi penjelasan mengenai implementasi dari perancangan yang sudah dilakukan pada bab sebelumnya. Implementasi berupa *Pseudocode* untuk membangun program.

4.1 Lingkungan Implementasi

Implementasi penerapan *computation offloading* dengan metode *Fuzzy Multi Criteria Decision Making* pada sistem perangkat lunak *image processing* menggunakan spesifikasi perangkat keras dan perangkat lunak seperti yang ditunjukkan pada Tabel 4.1.

Tabel 4.1 Lingkungan Implementasi Perangkat Lunak

Privilege	**Perangkat**	**Jenis Perangkat**	**Spesifikasi**
***Perangkat IoT* (Raspberry Pi 1 Model B)**	**Perangkat Keras**	Prosesor	ARM1176JZF-S 700 MHz
		Memori	512 MB
		Koneksi	USB wifi dongle
	Perangkat Lunak	Sistem Operasi	Raspbian OS
Cloud Server	**Penyedia Layanan**	Digital Ocean	
	IP *Address*	128.199.246.173	
	Perangkat Keras	Prosesor	Intel(R) Xeon(R) CPU E5-2630 @ 2.30GHz
		Memori	512 MB

	Perangkat Lunak	Sistem Operasi	Ubuntu 16.04
	Perangkat Lunak	Perangkat Pengembang	Visual Studio Code sebagai *text editor*, dan Wondershaper sebagai pengatur kecepatan internet

4.2 Implementasi

Pada sub bab implementasi ini menjelaskan mengenai pembangunan perangkat lunak secara detail dan menampilkan *Pseudocode* yang digunakan untuk pengambilan nilai-nilai faktor penentu metode *offloading*, penghitungan jumlah mobil pada gambar, metode *Entrpoy Based Fuzzy* MCDM untuk menentukan apakah beban kerja diproses secara lokal atau secara *offloading*, dan komunikasi antara Perangkat IoT dan *cloud server*.

4.2.1 Faktor Penentu Metode *Offloading*

Pada awal sistem berjalan, setelah inisialisasi serial antar Perangkat IoT maka sistem akan mengambil nilai dari faktor penentu metode *offloading*. Faktor penentu metode *offloading* terdiri dari 5 macam diantaranya kapasitas RAM Raspberry Pi yang tersedia, *download speed*, *upload speed*, jumlah data citra yang diproses, dan daya yang dikonsumsi. Tiap nilai faktor metode *offloading* diperoleh pada dan Raspberry Pi. Raspberry Pi mengambil nilai jumlah data citra yang diproses menggunakan file `countfile.py`. File `countfile.py` pada *Pseudocode* 4.1.

```
1.  import os
2.
3.  path, dir, files = next(os.walk("/home/pi/penelit
    ian/data"))
4.  file_count = len(files)
5.  print file_count
```

***Pseudocode* 4.1 Kode sumber untuk mendapatkan jumlah data citra yang diproses**

Sedangkan untuk mendapatkan kapasitas RAM yang tersedia menggunakan *bash command* "/proc/meminfo". *Bash command* dijalankan melalui bahasa Python pada fungsi `getFreeRAM`. Satuan keluaran dari fungsi `getFreeRAM` adalah *Kilo Bytes*. Fungsi `getFreeRAM` ditunjukkan pada *Pseudocode* 4.2.

```
1.  def getFreeRam():
2.      command = "grep 'MemFree' /proc/meminfo | awk
        '{print $2}'"
3.      result = subprocess.check_output(command, shel
        l=True).strip()
4.      return result
```

***Pseudocode* 4.2 Kode sumber untuk mendapatkan kapasitas RAM yang tersedia**

Faktor penentu metode *offloading* yang lainnya diantaranya *download speed* dan *upload speed*. *Download speed* dan *upload speed* diperoleh dengan menggunakan *library speedtest*. *Library speedtest* diterapkan pada fungsi `getInternetSpeed` yang ditunjukkan pada *Pseudocode* 4.3 dengan parameter *category* yang digunakan sebagai pembeda *download speed* dan *upload speed*.

```
1.  def getInternetSpeed():
2.      command = “speedtest --simple --no-pre-
        allocate | grep 'Download\|Upload' | awk '{pri
        nt $2}'”
3.      result = subprocess.check_output(command, shel
        l=True).strip().splitlines()
4.      return result
```

Pseudocode* 4.3 Kode sumber untuk mendapatkan *download speed* dan *upload speed

Faktor penentu metode *offloading* yang terakhir adalah daya yang dikeluarkan. Nilai daya dan tegangan baterai diperoleh dengan menggunakan *library ina219*. *Library ina219* diterapkan pada fungsi `getPower` dan `getVoltage` yang ditunjukkan pada *Pseudocode* 4.4.

```
1.  SHUNT_OHMS = 0.1
2.  ina = INA219(SHUNT_OHMS)
3.  ina.configure()
4.
5.  def run_command(command):
6.      result =  subprocess.check_output(command, sh
    ell=True).strip()
7.      return result
8.
9.  def get_power(stop, power):
10.     while True:
11.         power_now = ina.power()
12.         if power_now > power:
13.             power = power_now
14.
15.         if stop():
16.             global max_power
17.             max_power = power
18.             break
19.
20. def get_voltage(stop, flag):
21.     avg_voltage = []
22.
```

```
    while True:
        avg_voltage.append(ina.voltage())

        if stop():
            voltage = mean(avg_voltage)
            if flag == 0:
                global initial_voltage
                initial_voltage = voltage

            else:
                global final_voltage
                final_voltage = voltage

            break
```

***Pseudocode* 4.4 Kode sumber untuk mendapatkan daya yang dikonsumsi dan tegangan baterai**

4.2.2 Perhitungan *Entropy Based Fuzzy* MCDM

Perhitungan *Entropy Based Fuzzy* MCDM dilakukan dengan langkah-langkah sebagai berikut :

1. Inisialisasi *fuzzy number* dari masing-masing variabel linguistik (*Low, Medium, High*). Inisialisasi ditunjukkan pada Tabel 4.2.

Tabel 4.2 Inisialisasi *Fuzzy Number*

Variabel Linguistik	*Fuzzy Number*
Low (L)	(0.0, 0.3, 0.3, 0.5)
Medium (M)	(0.3, 0.5, 0.5, 0.7)
High (H)	(0.5, 0.7, 0.7, 1.0)

2. Konversi nilai faktor penentu metode *offloading* menjadi variabel linguistik dengan fungsi keanggotaan pada Sub Bab 3.4. Sebagai contoh, nilai RAM yang tersedia yaitu 183 MB, *download speed* yaitu 10.99 Mbit/s, *upload speed* yaitu 1.89 Mbit/s, jumlah data citra yaitu 1, dan daya yang

dikonsumsi yaitu 2.7 W. Untuk metode lokal, nilai RAM yang tersedia dikonversi dengan rumus *high* = $\frac{\text{nilai}-125}{212-125}$ sedangkan untuk *medium* = $1-(\frac{\text{nilai}-125}{212-125})$. Maka diperoleh nilai *high* = $\frac{183-125}{212-125}$ = 0.667, sedangkan *medium* = $1-(\frac{183-125}{212-125})$ = 0.333. Jika dibandingkan maka nilai *high* lebih besar dari *medium* sehingga variabel linguistik metode lokal menjadi *high*. Sedangkan metode *offloading*, nilai RAM yang tersedia dikonversi dengan rumus *low* = $\frac{\text{nilai}-125}{212-125}$ sedangkan untuk *medium* = $1-(\frac{\text{nilai}-125}{212-125})$. Maka diperoleh nilai *low* = $\frac{183-125}{212-125}$ = 0.667, sedangkan *medium* = $1-(\frac{183-125}{212-125})$ = 0.333. Jika dibandingkan maka nilai *low* lebih besar dari *medium* sehingga variabel linguistik metode *offloading* menjadi *low*. Hasil konversi nilai faktor ke variabel linguistik ditunjukkan pada Tabel 4.3.

Tabel 4.3 Hasil Konversi Nilai Faktor ke Variabel Linguistik

Faktor	Metode	Variabel Linguistik
RAM yang tersedia	Lokal	*High*
	Offloading	*Low*
Download Speed	Lokal	*High*
	Offloading	*Low*
Upload Speed	Lokal	*Medium*
	Offloading	*Medium*
Jumlah Data Citra	Lokal	*Low*
	Offloading	*High*
Total Daya yang dikonsumsi	Lokal	*High*
	Offloading	*Low*

3. Pengubahan variabel linguistik menjadi *fuzzy number* lalu menghitung nilai defuzzyfikasi dari tiap faktor penentu untuk metode lokal dan metode *offloading* yang ditunjukkan pada Tabel 4.4.

Tabel 4.4 Hasil Defuzzyfikasi Nilai Faktor Penentu Metode *Offloading*

Faktor	Metode	Variabel Linguistik	*Fuzzy Number*	Defuzzy-fikasi
RAM yang tersedia	Lokal	*High*	(0.5, 0.7, 0.7, 1.0)	0.725
	Offloading	*Low*	(0.0, 0.3, 0.3, 0.5)	0.275
Download Speed	Lokal	*High*	(0.5, 0.7, 0.7, 1.0)	0.725
	Offloading	*Low*	(0.0, 0.3, 0.3, 0.5)	0.275
Upload Speed	Lokal	*Medium*	(0.3, 0.5, 0.5, 0.7)	0.5
	Offloading	*Medium*	(0.3, 0.5, 0.5, 0.7)	0.5
Jumlah Data Citra	Lokal	*Low*	(0.0, 0.3, 0.3, 0.5)	0.275
	Offloading	*High*	(0.5, 0.7, 0.7, 1.0)	0.725
Total Daya yang dikonsumsi	Lokal	*High*	(0.5, 0.7, 0.7, 1.0)	0.725
	Offloading	*Low*	(0.0, 0.3, 0.3, 0.5)	0.275

4. Menghitung *weight* setiap fitur pada setiap alternatif dan setiap kriteria dengan persamaan (2.6) [9]. Hasil penghitungan *weight* dapat dilihat pada Tabel 4.5.

Tabel 4.5 Hasil *weight* setiap fitur untuk setiap alternatif pada setiap kriteria

Faktor	Metode	*Feature weight*
RAM yang tersedia	Lokal	1.507
	Offloading	0.572
Download Speed	Lokal	1.507
	Offloading	0.572
Upload Speed	Lokal	1.250
	Offloading	1.250
Jumlah Data Citra	Lokal	0.572
	Offloading	1.507
Total Daya yang dikonsumsi	Lokal	1.507
	Offloading	0.572

5. Menghitung nilai *Entropy* dari setiap kriteria dengan persamaan (2.7) [9]. Hasil penghitungan *Entropy* dapat dilihat pada Tabel 4.6.

Tabel 4.6 Hasil *Entropy* dari setiap kriteria

Faktor	*Entropy*
Available RAM	-0.431
Download Speed	-0.431
Upload Speed	-0.805
Jumlah Data Citra	-0.431
Total Daya yang dikonsumsi	-0.431

6. Menghitung nilai *Variant Coefficient* dari setiap kriteria dengan persamaan (2.8) [9]. Hasil penghitungan *Variant Coefficient* dapat dilihat pada Tabel 4.7.

Tabel 4.7 Hasil *Variant Coefficient* dari setiap kriteria

Faktor	*Variant Coefficient*
RAM yang tersedia	1.431
Download Speed	1.431
Upload Speed	1.805
Jumlah Data Citra	1.431
Total Daya yang dikonsumsi	1.431

7. Menghitung *weight* dari nilai *Entropy* berdasarkan *variant coefficient* dengan persamaan (2.9) [9]. Hasil penghitungan *weight* dapat dilihat pada Tabel 4.8.

Tabel 4.8 Hasil *Entropy Weight* dari setiap kriteria

Faktor	*Weight*
RAM yang tersedia	0.190
Download Speed	0.190
Upload Speed	0.240
Jumlah Data Citra	0.190
Total Daya yang dikonsumsi	0.190

8. Menghitung normalisasi matriks *decision maker* dengan persamaan (2.10) [9]. Hasil penghitungan normalisasi matriks dapat dilihat pada Tabel 4.9.

Tabel 4.9 Hasil normalisasi matriks *decision maker*

Faktor	Metode	Normalisasi
RAM yang tersedia	Lokal	0.725
	Offloading	0.275
Download Speed	Lokal	0.725
	Offloading	0.275
Upload Speed	Lokal	0.500
	Offloading	0.500
Jumlah Data Citra	Lokal	0.275
	Offloading	0.725
Total Daya yang dikonsumsi	Lokal	0.725
	Offloading	0.275

9. Menghitung *weight* matriks *decision maker* yang telah dinormalisasi dengan persamaan (2.11) [9]. Hasil penghitungan *weight* matriks *decision maker* dapat dilihat pada Tabel 4.10.

Tabel 4.10 Hasil *weight* matriks *decision maker*

Faktor	Metode	*Weight*
RAM yang tersedia	Lokal	0.138
	Offloading	0.052
Download Speed	Lokal	0.138
	Offloading	0.052
Upload Speed	Lokal	0.120
	Offloading	0.120
Jumlah Data Citra	Lokal	0.052
	Offloading	0.138
Total Daya yang dikonsumsi	Lokal	0.138
	Offloading	0.052

10. Menghitung Menghitung S^+ dan S^- dengan persamaan (2.12) [9] dimana W_{ij} adalah *weight* matriks *decision maker* yang bersifat menguntungkan pada kriteria ke- j, dan persamaan (2.13) [9] dimana W_{ij} adalah *weight* matriks *decision maker* yang bersifat merugikan pada kriteria ke- j. Hasil penghitungan *weight* matriks *decision maker* dapat dilihat pada Tabel 4.11.

Tabel 4.11 Hasil *weight* matriks *decision maker*

Metode	S^+/S^-	Nilai
Lokal	S^+	0.190
	S^-	0.395
Offloading	S^+	0.190
	S^-	0.224

11. Menghitung skor akhir dengan persamaan (2.14) [9]. Hasil penghitungan skor akhir dapat dilihat pada Tabel 4.12.

Tabel 4.12 Hasil skor akhir

Metode	Skor akhir
Lokal	0.414
Offloading	0.586

12. Bandingkan skor akhir metode lokal dan metode *offloading*, sesuai dengan hasil perhitungan diatas maka metode *offloading* memiliki nilai lebih besar dari metode lokal sehingga akan dilakukan metode *offloading*.

4.2.3 *Entropy Based Fuzzy* MCDM *Code*

Entropy Based Fuzzy MCDM pada tugas akhir ini digunakan sebagai *decision maker* dalam menentukan apakah beban kerja akan diproses secara lokal atau secara *offloading*. *Entropy Based Fuzzy* MCDM menggunakan nilai faktor penentu metode *offloading* yang sudah didapatkan sebelumnya untuk menentukan beban kerja diproses secara lokal atau secara *offloading*. Langkah pertama yaitu inisialisasi *fuzzy number* dari masing-masing variabel linguistik (*Low*, *Medium*, dan *High*) pada fungsi `getFuzzyNumber` yang ditunjukkan pada *Pseudocode* 4.5.

```
1.  def getFuzzyNumber(category, pos):
2.      fnLow = [0.0, 0.3, 0.3, 0.5]
3.      fnMedium = [0.3, 0.5, 0.5, 0.7]
4.      fnHigh = [0.5, 0.7, 0.7, 1.0]
5.
6.      if category == 'L':
7.          return fnLow[pos]
8.      elif category == 'M':
9.          return fnMedium[pos]
10.     elif category == 'H':
11.         return fnHigh[pos]
```

Pseudocode 4.5 Kode sumber untuk inisialisasi *fuzzy number*

Langkah kedua yaitu mengkonversi nilai-nilai faktor penentu menjadi variabel linguistik (*Low*, *Medium*, *High*) untuk metode lokal dan metode *offloading* dengan menggunakan fungsi keanggotaan dari masing-masing faktor. Pada tugas akhir ini pemberi bobot terdiri dari lima data. Langkah ketiga yaitu menghitung rata-rata *fuzzy number* tiap faktor penentu metode *offloading* sesuai dengan jumlah pemberi bobot. Langkah ini diterapkan pada fungsi `convertToLinguistic` yang ditunjukkan pada *Pseudocode* 4.6.

```
1.  def convertToLinguistic(criteria, value):
2.      batas1 = 0
3.      batas2 = 0
4.      batas3 = 0
5.
6.      if criteria == 1:
7.          batas1 = 40.0
8.          batas2 = 125.0
9.          batas3 = 212.0
10.     elif criteria == 2:
11.         batas1 = 3.0
12.         batas2 = 5.0
13.         batas3 = 7.0
14.     elif criteria == 3:
15.         batas1 = 1.0
16.         batas2 = 2.0
17.         batas3 = 3.0
18.     elif criteria == 4:
19.         batas1 = 4
20.         batas2 = 7
21.         batas3 = 10
22.     elif criteria == 5:
23.         batas1 = 2600.0
24.         batas2 = 2650.0
25.         batas3 = 2700.0
26.
27.     if value <= batas1:
28.
29.         houseLokal.append('L')
30.         houseOffloading.append('H')
```

```
31.
32.     elif value >= batas3:
33.
34.         houseLokal.append('H')
35.         houseOffloading.append('L')
36.
37.     elif value == batas2:
38.         houseLokal.append('M')
39.         houseOffloading.append('M')
40.
41.     elif value > batas1 and value < batas2:
42.         low = 1-((value-batas1)/(batas2-
    batas1))
43.         medium = (value-batas1)/(batas2-batas1)
44.         if low > medium:
45.
46.             houseLokal.append('L')
47.             houseOffloading.append('H')
48.
49.         else:
50.             houseLokal.append('M')
51.             houseOffloading.append('M')
52.
53.     elif value > batas2 and value < batas3:
54.         high = (value-batas2)/(batas3-batas2)
55.         medium = 1-((value-batas2)/(batas3-
    batas2))
56.         if high > medium:
57.
58.             houseLokal.append('H')
59.             houseOffloading.append('L')
60.
61.         else:
62.             houseLokal.append('M')
63.             houseOffloading.append('M')
```

***Pseudocode* 4.6 Kode sumber untuk mengkonversi nilai faktor penentu *offloading* menjadi variabel linguistik**

Langkah ketiga yaitu menghitung nilai defuzzyfikasi dari tiap faktor penentu untuk metode lokal dan metode *offloading*. Penerapan langkah ini ditunjukkan pada *Pseudocode* 4.7.

```
defuzzyLokal = []
defuzzyOffloading = []

for x in range(5):
    totalFuzzyLokal = 0
    totalFuzzyOffloading = 0
    for y in range(4):
        totalFuzzyLokal += getFuzzyNumber(houseLokal[x], y)
        totalFuzzyOffloading += getFuzzyNumber(houseOffloading[x], y)

    defuzzyLokal.append(totalFuzzyLokal/5)
    defuzzyOffloading.append(totalFuzzyOffloading/5)
```

Pseudocode* 4.7 Kode sumber untuk menghitung nilai defuzzyfikasi untuk metode lokal dan *offloading

Langkah keempat yaitu menghitung *weight* setiap fitur pada setiap alternatif dan setiap kriteria dan langkah kelima yaitu menghitung nilai *Entropy* dari setiap kriteria. Kedua langkah ini diterapkan pada fungsi `countEntropy` yang ditunjukkan pada *Pseudocode* 4.8.

```
def countEntropy(valueLokal, valueOffloading):
    #calculating feature weight
    featureWeightLocal = valueLokal/(pow(valueLokal,2) + pow(valueOffloading, 2))
    featureWeightOffloading = valueOffloading/(pow(valueLokal,2) + pow(valueOffloading, 2))

    #calculating entropy
    entropy = -(1/log(2, 2))*((featureWeightLocal * log(featureWeightLocal, 2) + (featureWeightOffloading * log(featureWeightOffloading, 2))))
    return entropy
```

***Pseudocode* 4.8 Kode sumber untuk menghitung *weight* tiap fitur dan menghitung nilai *Entropy* tiap kriteria**

Langkah keenam yaitu menghitung *variant coefficient* dati setiap kriteria dan langkah ketujuh yaitu menghitung *weight* dari nilai *Entropy* berdasarkan *variant coefficient*. Penerapan kedua langkah ini ditunjukkan pada *Pseudocode* 4.9.

```
#calculate entropy and variant coefficient
entropy = []
for x in range(5):
    entropy.append(abs(1-countEntropy(defuzzyLokal[x],defuzzyOffloading[x])))

#calculate weight
weight = []
for x in range(5):
    weight.append(entropy[x]/sum(entropy))
```

***Pseudocode* 4.9 Kode sumber untuk menghitung *Variant Coefficient* dan *Entropy Weight* dari setiap kriteria**

Langkah kedelapan yaitu menghitung normalisasi matriks *decision maker*. Penerapan langkah ini ditunjukkan pada *Pseudocode* 4.10.

```
1.  #calculate normalized decision
2.  normalizedLokal = []
3.  normalizedOffloading = []
4.
5.  for x in range(5):
6.      normalizedLokal.append(defuzzyLokal[x]/(defuz
    zyLokal[x]+defuzzyOffloading[x]))
7.      normalizedOffloading.append(defuzzyOffloading
    [x]/(defuzzyLokal[x]+defuzzyOffloading[x]))
```

Pseudocode* 4.10 Kode sumber untuk menghitung normalisasi matriks *decision maker

Langkah kesembilan yaitu menghitung *weight* matriks *decision maker* yang telah dinormalisasi. Penerapan langkah ini ditunjukkan pada *Pseudocode* 4.11.

```
1.  #calculate weighted normalized
2.  weightedLokal = []
3.  weightedOffloading = []
4.
5.  for x in range(5):
6.      weightedLokal.append(normalizedLokal[x] * wei
    ght[x])
7.      weightedOffloading.append(normalizedOffloadin
    g[x] * weight[x])
```

***Pseudocode* 4.11 Kode sumber untuk menghitung *weight* matriks *decision maker* yang telah dinormalisasi**

Langkah kesepuluh yaitu menghitung S^+ dan S^-. Penerapan langkah ini ditunjukkan pada *Pseudocode* 4.12.

```
#calculate S+ and S-
#Si = [S+, S-]
sLokal = [weightedLokal[0] + weightedLokal[3], weightedLokal[1] + weightedLokal[2] + weightedLokal[4]]
sOffloading = [weightedOffloading[0] + weightedOffloading[3], weightedOffloading[1] + weightedOffloading[2] + weightedOffloading[4]]
```

***Pseudocode* 4.12 Kode sumber untuk menghitung S^+ dan S^-**

Langkah terakhir yaitu menghitung skor akhir dari metode lokal dan *offloading*. Setelah itu membandingkan skor akhir dari metode lokal dan *offloading*. Langkah ini diterapkan pada *Pseudocode* 4.13.

```
#calculate relative weight
finalScoreLokal = sLokal[0] + ((sLokal[1] + sOffloading[1]) / (sLokal[1] * ((1/sLokal[1]) + (1/sOffloading[1]))))
finalScoreOffloading = sOffloading[0] + ((sLokal[1] + sOffloading[1]) / (sOffloading[1] * ((1/sLokal[1]) + (1/sOffloading[1]))))

print '[INFO] Final score Local ---- : {}'.format(finalScoreLokal)
print '[INFO] Final score Offloading : {}'.format(finalScoreOffloading)

if finalScoreLokal > finalScoreOffloading:
    print '[INFO] Decision ------------ : local'
    return 'local'
else:
    print '[INFO] Decision ------------ : offloading'
    return 'offloading'
```

Pseudocode* 4.13 Kode sumber untuk menghitung skor akhir metode lokal dan *offloading

4.2.4 Komunikasi Perangkat IoT dan *cloud server*

Komunikasi perangkat IoT dan *cloud server* dengan menggunakan *socket*. Penerapan komunikasi via *socket* pada Perangkat IoT ditunjukkan pada *Pseudocode* 4.14.

```
1.  s = socket.socket()
2.  port = 6666
3.  s.connect(('128.199.246.173', port))
4.
5.  print '[INFO] Sending {} to server'.format(fileNa
    me[x])
6.  image = open('data/{}'.format(fileName[x]), 'rb')

7.  image_read = image.read()
8.  image_encode = base64.encodestring(image_read)
9.  image_size = len(image_encode)
10.
11. # Send Filename
12. s.send(fileName[x])
13. s.recv(1024)
14.
15. # Send image length
16. s.send(str(image_size))
17. s.recv(1024)
18.
19. # Send image
20. s.send(image_encode)
21. s.recv(1024)
22.
23. print '[INFO] Sending {} succesfully'.format(file
    Name[x])
24. jumlah_mobil = s.recv(1024)
25. s.send('ACK!')
26. print '[INFO] Number of car : {} at {}'.format(ju
    mlah_mobil, fileName[x])
27.
28. image_string = ''
29. image_size = int(s.recv(1024))
30. s.send('ACK!')
31. while image_size > 0:
```

```
    if image_size > 1024:
        image_data = s.recv(1024)
    else:
        image_data = s.recv(image_size)
    if not image_data:
        break
    image_size -= len(image_data)
    image_string += image_data
s.send('ACK!')
image_result = open('hasil/{}'.format(fileName[x]
+'o'),'wb')
image_decode = base64.decodestring(image_string)

image_result.write(image_decode)

s.close()
```

***Pseudocode* 4.14 Kode sumber implementasi komunikasi socket pada Perangkat IoT**

Sedangkan penerapan komunikasi via *socket* pada *cloud server* ditunjukkan pada *Pseudocode* 4.15.

```
s = socket.socket()
print "[INFO] Socket successfully created"
port = 6666
s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEAD
DR, 1)
s.bind(('', port))
print "[INFO] Socket binded to {}".format(port)
s.listen(99)
print '[INFO] Socket is listening'

while True:
    c, addr = s.accept()
    print '[INFO] Got connection from {}'.format(
addr)
    # Receive Image from Client
    image_name = c.recv(1024)
    c.send('ACK!')
    image_size = int(c.recv(1024))
```

```
17.     print image_size
18.     c.send('ACK!')
19.     image_string = ''
20.     while image_size > 0:
21.         if image_size > 1024:
22.             image_data = c.recv(1024)
23.         else:
24.             image_data = c.recv(image_size)
25.         # if not image_data:
26.         #     break
27.         image_size -= len(image_data)
28.         image_string += image_data
29.     c.send('ACK!')
30.     image_result = open('data/{}'.format(image_na
    me),'wb')
31.     dummy = open('log','wb')
32.     image_decode = base64.decodestring(image_stri
    ng)
33.     image_result.write(image_decode)
34.     dummy.write(image_string)
35.     # END OF Receive Image from Client
36.
37.     time.sleep(12)
38.     start_time = time.time()
39.     # Counting car from image
40.     image = cv2.imread('data/{}'.format(image_nam
    e))
41.     car_count = 0
42.     (h, w) = image.shape[:2]
43.     blob = cv2.dnn.blobFromImage(cv2.resize(image
    , (300, 300)), 0.007843, (300, 300), 127.5)
44.     print("[INFO] computing object detections..."
    )
45.     net.setInput(blob)
46.     detections = net.forward()
47.     for i in np.arange(0, detections.shape[2]):
48.         confidence = detections[0, 0, i, 2]
49.         if confidence > 0.2:
50.             idx = int(detections[0, 0, i, 1])
51.             box = detections[0, 0, i, 3:7] * np.a
    rray([w, h, w, h])
```

```
52.             (startX, startY, endX, endY) = box.as
    type("int")
53.             if CLASSES[idx] == "car":
54.                 # display the prediction
55.                 car_count = car_count + 1
56.                 label = "{}: {:.2f}%".format(CLAS
    SES[idx], confidence * 100)
57.                 print("[INFO] {}".format(label))

58.                 cv2.rectangle(image, (startX, sta
    rtY), (endX, endY), COLORS[idx], 2)
59.                 y = startY - 15 if startY - 15 >
    15 else startY + 15
60.                 cv2.putText(image, label, (startX
    , y), cv2.FONT_HERSHEY_SIMPLEX, 0.5, COLORS[idx],
     2)
61.     cv2.imwrite('hasil/{}'.format(image_name), im
    age)
62.     waktunya = time.time() - start_time
63.     print '[INFO] Number of car : {} processed in
     {} seconds'.format(car_count,waktunya)
64.     c.send(str(car_count))
65.     c.recv(1024)
66.     # END OF Counting car from image
67.
68.     image = open('hasil/{}'.format(image_name),'r
    b')
69.     image_read = image.read()
70.     image_encode = base64.encodestring(image_read
    )
71.     image_size =  len(image_encode)
72.
73.     print '[INFO] Send image to client'
74.     c.send(str(image_size))
75.     c.recv(1024)
76.     c.send(image_encode)
77.     c.recv(1024)
78.     c.close()
```

Pseudocode* 4.15 Kode sumber implementasi komunikasi socket pada *cloud server

4.2.5 Penghitungan Jumlah Mobil pada Gambar

Penghitungan jumlah mobil pada gambar dimulai dengan mengakses *file model* MobileNets, lalu data citra dikonversi menjadi *Binary Large OBject* (BLOB). Hasil konversi dibandingkan dengan *model* MobileNets, jika objek mobil sudah terdeteksi maka data citra akan diberi penanda sesuai dengan lokasi objek yang terdeteksi. Objek mobil yang terdeteksi akan dihitung dan dijumlahkan. Penerapan ini ditunjukkan pada *Pseudocode* 4.16.

```
1.  for x in range(0, int(totalFile)):
2.      print "[INFO] Counting car at {}".format(file
    Name[x])
3.      image = cv2.imread('data/{}'.format(fileName[
    x]))
4.      (h, w) = image.shape[:2]
5.      blob = cv2.dnn.blobFromImage(cv2.resize(image
    , (300, 300)), 0.007843, (300, 300), 127.5)
6.      net.setInput(blob)
7.      detections = net.forward()
8.      car = 0
9.      for i in np.arange(0, detections.shape[2]):
10.         confidence = detections[0, 0, i, 2]
11.         if confidence > 0.2:
12.             idx = int(detections[0, 0, i, 1])
13.             box = detections[0, 0, i, 3:7] * np.a
    rray([w, h, w, h])
14.             (startX, startY, endX, endY) = box.as
    type("int")
15.
16.             if CLASSES[idx] == "car":
17.                 car += 1
18.                 label = "{}: {:.2f}%".format(CLAS
    SES[idx], confidence * 100)
19.                 print("[INFO] {}".format(label))

20.                 cv2.rectangle(image, (startX, sta
    rtY), (endX, endY), COLORS[idx], 2)
```

```
                y = startY - 15 if startY - 15 > 15 else startY + 15
                cv2.putText(image, label, (startX, y), cv2.FONT_HERSHEY_SIMPLEX, 0.5, COLORS[idx], 2)
    cv2.imwrite('hasil/{}'.format(fileName[x]), image)
```

***Pseudocode* 4.16 Kode sumber untuk penghitungan jumlah mobil pada gambar**

(*Halaman ini sengaja dikosongkan*)

BAB V
HASIL UJI COBA DAN EVALUASI

Bab ini berisi penjelasan mengenai skenario uji coba dan evaluasi penentuan keputusan eksekusi beban kerja pada *computation offloading*. Hasil uji coba didapatkan dari implementasi pada Bab 4 dengan skenario yang berbeda. Bab ini berisikan pembahasan mengenai lingkungan pengujian, data pengujian, dan uji kinerja.

5.1 Lingkungan Pengujian

Lingkungan pengujian pada uji coba permasalahan penentuan keputusan eksekusi beban kerja oleh sistem melalui faktor – faktor penentu metode *offloading* menggunakan spesifikasi keras dan perangkat lunak seperti yang ditunjukkan pada Tabel 5.1.

Tabel 5.1 Spesifikasi Lingkungan Pengujian

Privilege	**Perangkat**	**Jenis Perangkat**	**Spesifikasi**
***Perangkat IoT* (Raspberry Pi 1 Model B)**	**Perangkat Keras**	Prosesor	ARM1176JZF-S 700 MHz
		Memori	512 MB
		Koneksi	USB wifi dongle
	Perangkat Lunak	Sistem Operasi	Raspbian OS
Cloud Server	**Penyedia Layanan**	Digital Ocean	
	IP *Address*	128.199.246.173	

	Perangkat Keras	Prosesor	Intel(R) Xeon(R) CPU E5-2630 @ 2.30GHz
		Memori	512 MB
	Perangkat Lunak	Sistem Operasi	Ubuntu 16.04
	Perangkat Lunak	Perangkat Pengembang	Visual Studio Code sebagai *text editor*, dan Wondershaper sebagai pengatur kecepatan internet

Perangkat pengujian terdiri dari Raspberry Pi, sensor arus INA219, *Power Bank* dan *Cloud Server*. Raspberry Pi dan sensor arus INA219 merupakan satu Perangkat IoT dan saling terhubung dengan kabel *micro* USB yang ditunjukkan pada Gambar 5.1.

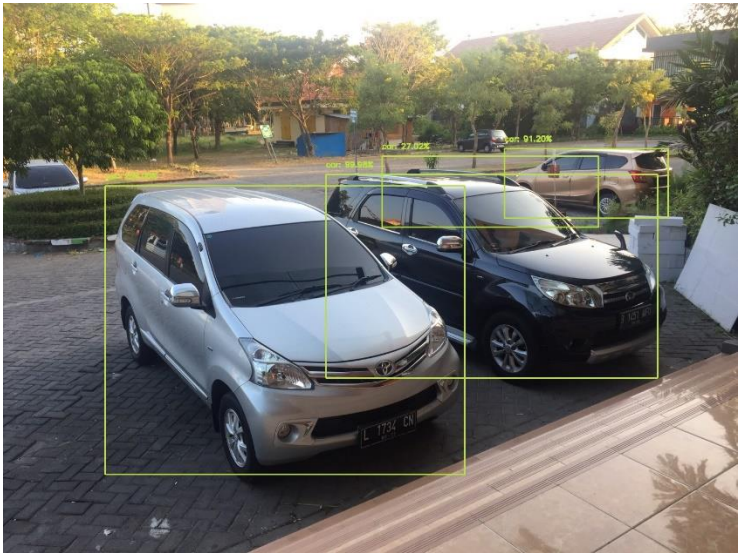
Gambar 5.1 Perangkat IoT

Data citra yang digunakan pada uji coba adalah gambar mobil. Berikut adalah beberapa data citra yang digunakan pada uji coba dan hasil pemrosesannya.

Gambar 5.2 Contoh data citra 1

Gambar 5.3 Contoh hasil pemrosesan data citra 1

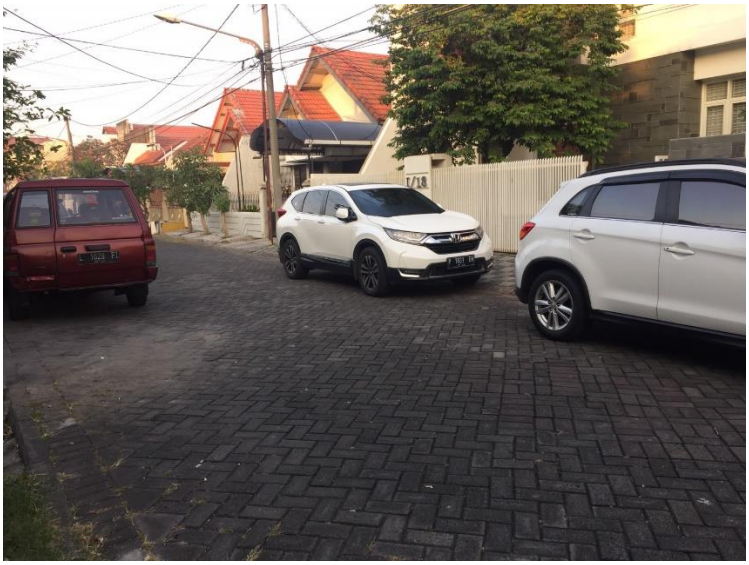
Gambar 5.4 Contoh data citra 2



Gambar 5.5 Contoh hasil pemrosesan data citra 2

Gambar 5.6 Contoh data citra 3

Gambar 5.7 Contoh hasil pemrosesan data citra 3



Gambar 5.8 Contoh data citra 4

Gambar 5.9 Contoh hasil pemrosesan data citra 4

5.2 Skenario Uji Coba

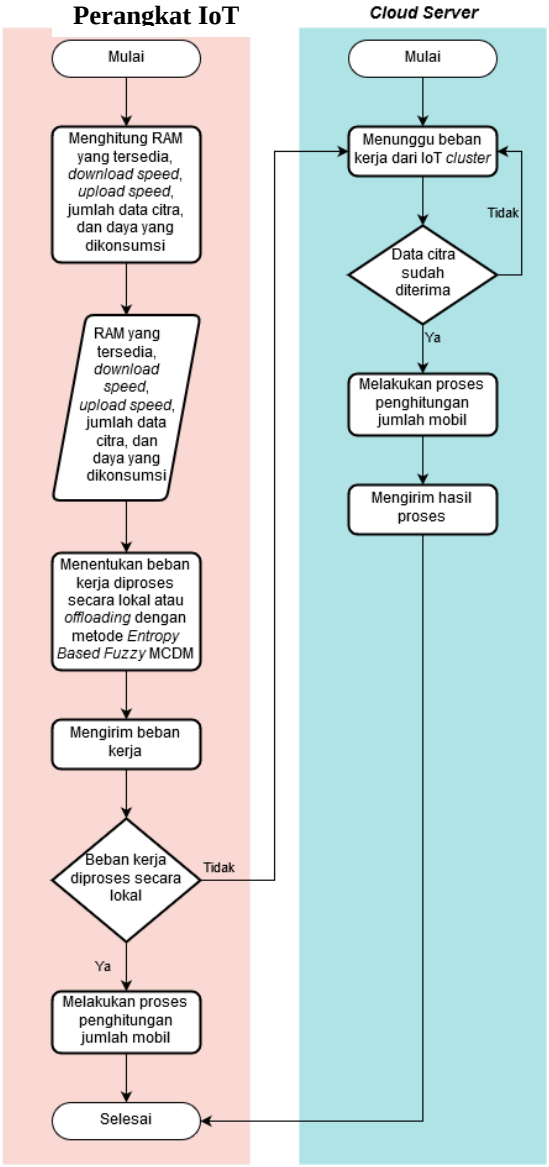
Sebelum melakukan uji coba, perlu ditentukan skenario yang akan digunakan dalam uji coba. Melalui skenario ini, perangkat akan diuji apakah sudah berjalan dengan benar dan bagaimana performa pada masing-masing skenario. Dan membandingkan skenario manakah yang memiliki hasil lebih baik.

Di dalam skenario uji coba, terdapat kondisi faktor yaitu RAM yang tersedia pada Raspberry Pi, *download speed*, *upload speed*, jumlah data citra yang diproses, dan daya yang dikonsumsi dalam menentukan metode pengeksekusian beban kerja. Kondisi faktor tersebut akan dikategorikan ke dalam syarat tertentu untuk keperluan uji coba.

Masing-masing kondisi faktor dikategorikan menjadi tiga kategori yaitu *Low*, *Medium*, dan *High*. Pembagian kategori tiap kondisi faktor disesuaikan dengan fungsi keanggotaan pada Sub Bab 3.4. pada saat perangkat *client* melakukan proses *image processing*.

Dalam menjalankan eksekusi beban kerja proses *image processing* akan melakukan urutan langkah–langkah kerja yang sudah dijelaskan pada Sub Bab 3.3. Pada Gambar 5.10 akan dijelaskan bagaimana langkah–langkah alur kerja skenario uji coba sistem. Terdapat 3 macam skenario uji coba, yaitu:

1. Perhitungan waktu eksekusi yang dihasilkan oleh eksekusi beban kerja proses penghitungan jumlah mobil pada gambar oleh *computation offloading* dengan jumlah 4 gambar sebanyak 15 kali pengujian.
2. Perhitungan waktu eksekusi yang dihasilkan oleh eksekusi beban kerja proses penghitungan jumlah mobil pada gambar oleh *computation offloading* dengan jumlah 7 gambar sebanyak 15 kali pengujian.
3. Perhitungan waktu eksekusi yang dihasilkan oleh eksekusi beban kerja proses penghitungan jumlah mobil pada gambar oleh *computation offloading* dengan jumlah 10 gambar sebanyak 15 kali pengujian.



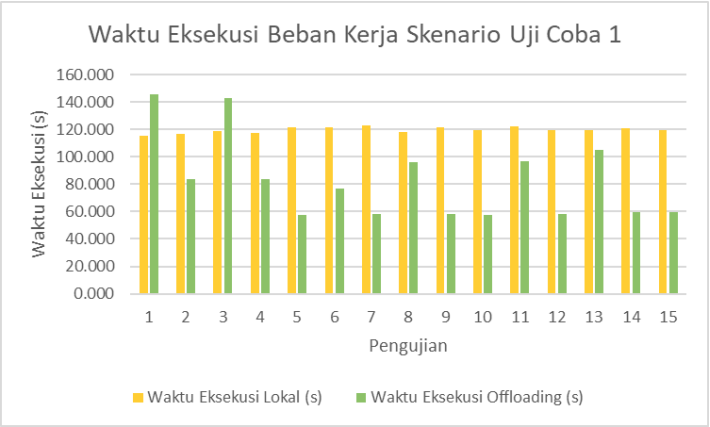
Gambar 5.10 Bagan alur kerja skenario uji coba

5.2.1 Skenario Uji Coba 1

Skenario uji coba 1 adalah pengujian yang dilakukan sebanyak 15 kali dengan kondisi RAM yang tersedia (7 *High*, 3 *Medium*, 5 *Low*), kondisi *download speed* (2 *High*, 6 *Medium*, 7 *Low*), kondisi *upload speed* (5 *High*, 3 *Medium*, 7 *Low*), dan jumlah gambar dari masing-masing pengujian sebanyak 4. Perhitungan waktu eksekusi yang dihasilkan oleh eksekusi beban kerja proses penghitungan jumlah mobil pada gambar dengan metode lokal dan *offloading* ditunjukkan pada Tabel 5.2.

Tabel 5.2 Waktu eksekusi beban kerja dengan metode lokal dan *offloading* pada Skenario Uji Coba 1

Pengujian	RAM yang tersedia	*Download Speed*	*Upload Speed*	Daya Konsumsi	Waktu Eksekusi	
	MB	(Mbit/s)	(Mbit/s)	(W)	Lokal (s)	Offloading (s)
1	51.472	0.070	0.150	2.642	115.166	145.758
2	49.520	4.230	0.250	2.728	116.894	83.734
3	245.052	0.070	0.160	2.631	118.804	142.646
4	82.740	4.100	0.220	2.660	117.547	83.820
5	212.484	10.840	7.830	3.088	121.385	57.774
6	220.288	3.350	0.300	2.658	121.270	76.926
7	105.224	3.920	7.350	2.950	123.046	57.867
8	32.333	3.400	0.210	2.717	117.819	96.164
9	245.200	12.460	8.010	2.947	121.297	57.872
10	114.232	4.930	7.440	2.903	119.658	57.782
11	50.000	2.700	0.190	2.656	122.194	96.456
12	250.712	4.810	8.310	2.873	119.458	58.433
13	34.052	0.070	2.240	2.815	119.148	104.722
14	227.900	2.830	2.350	2.829	120.779	59.822
15	228.096	4.620	2.390	2.795	119.414	59.477
Jumlah					1793.878	1239.251



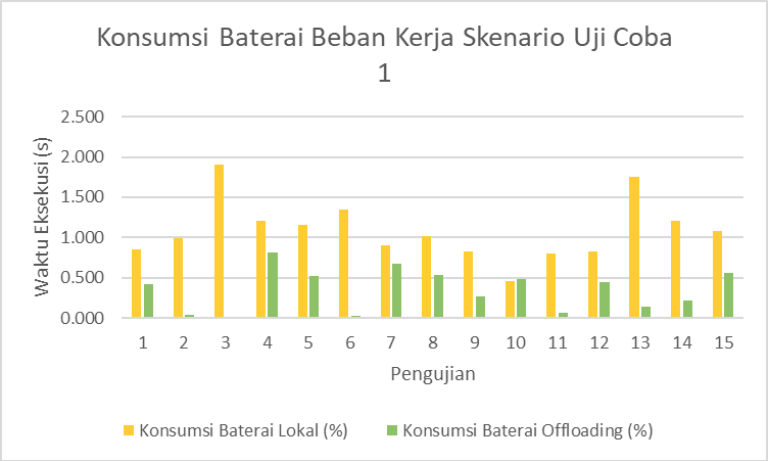
Gambar 5.11 Grafik waktu eksekusi beban kerja Skenario Uji Coba 1

Untuk perhitungan konsumsi baterai yang dihasilkan oleh eksekusi beban kerja proses penghitungan jumlah mobil pada gambar dengan metode lokal dan *offloading* ditunjukkan pada Tabel 5.3.

Tabel 5.3 Konsumsi Baterai beban kerja dengan metode lokal dan *offloading* pada Skenario Uji Coba 1

Pengujian	RAM yang tersedia	*Download Speed*	*Upload Speed*	Daya Konsumsi	Konsumsi Baterai	
	MB	(Mbit/s)	(Mbit/s)	(W)	Lokal (%)	*Offloading* (%)
1	51.472	0.070	0.150	2.642	0.850	0.420
2	49.520	4.230	0.250	2.728	0.987	0.034
3	245.052	0.070	0.160	2.631	1.911	0.014
4	82.740	4.100	0.220	2.660	1.210	0.808
5	212.484	10.840	7.830	3.088	1.155	0.521
6	220.288	3.350	0.300	2.658	1.349	0.024
7	105.224	3.920	7.350	2.950	0.903	0.674
8	32.333	3.400	0.210	2.717	1.016	0.539
9	245.200	12.460	8.010	2.947	0.820	0.270
10	114.232	4.930	7.440	2.903	0.462	0.484
11	50.000	2.700	0.190	2.656	0.807	0.070
12	250.712	4.810	8.310	2.873	0.832	0.440

13	34.052	0.070	2.240	2.815	1.756	0.143
14	227.900	2.830	2.350	2.829	1.205	0.213
15	228.096	4.620	2.390	2.795	1.081	0.564
Jumlah					16.344	5.218



Gambar 5.12 Grafik konsumsi baterai beban kerja Skenario Uji Coba 1

Sedangkan hasil waktu eksekusi dengan *decision making* pada uji coba 1 ditunjukkan pada Tabel 5.4.

Tabel 5.4 Waktu eksekusi beban kerja dengan *decision making* pada Skenario Uji Coba 1

Pengujian	***Entropy Based Fuzzy* MCDM Skor**		***Decision Making***	**Waktu Eksekusi (s)**
	Lokal	***Offloading***		
1	0.500	0.500	*Offloading*	145.758
2	0.414	0.586	*Offloading*	83.734
3	0.586	0.414	Lokal	118.804
4	0.500	0.500	*Offloading*	83.820
5	0.365	0.635	*Offloading*	57.774
6	0.586	0.414	Lokal	121.270
7	0.414	0.586	*Offloading*	57.867

8	0.455	0.545	*Offloading*	96.164
9	0.365	0.635	*Offloading*	57.872
10	0.378	0.622	*Offloading*	57.782
11	0.500	0.500	*Offloading*	96.456
12	0.414	0.586	*Offloading*	58.433
13	0.414	0.586	*Offloading*	104.722
14	0.500	0.500	*Offloading*	59.822
15	0.459	0.541	*Offloading*	59.477
Jumlah				1259.754

Untuk hasil konsumsi baterai dengan *decision making* pada uji coba 1 ditunjukkan pada Tabel 5.5.

Tabel 5.5 Konsumsi baterai beban kerja dengan *decision making* pada Skenario Uji Coba 1

Pengujian	***Entropy Based Fuzzy* MCDM Skor**		***Decision Making***	**Konsumsi baterai (%)**
	Lokal	***Offloading***		
1	0.500	0.500	*Offloading*	0.420
2	0.414	0.586	*Offloading*	0.034
3	0.586	0.414	Lokal	1.911
4	0.500	0.500	*Offloading*	0.808
5	0.365	0.635	*Offloading*	0.521
6	0.586	0.414	Lokal	1.349
7	0.414	0.586	*Offloading*	0.674
8	0.455	0.545	*Offloading*	0.539
9	0.365	0.635	*Offloading*	0.270
10	0.378	0.622	*Offloading*	0.484
11	0.500	0.500	*Offloading*	0.070
12	0.414	0.586	*Offloading*	0.440
13	0.414	0.586	*Offloading*	0.143
14	0.500	0.500	*Offloading*	0.213
15	0.459	0.541	*Offloading*	0.564
Jumlah				8.440

Berdasarkan hasil waktu eksekusi yang ditunjukkan pada Tabel 5.2 dan Tabel 5.4, diperoleh jumlah waktu eksekusi dengan metode lokal sebesar 1793.878 detik, metode *offloading* sebesar 1239.251 detik, dan dengan *decision making* sebesar 1259.754

detik. Hal ini menunjukkan hasil waktu eksekusi dengan *decision making* 534.125 detik lebih cepat dari metode lokal dan 20.502 detik lebih lambat dari metode *offloading*. Sedanghkan berdasarkan hasil konsumsi baterai yang ditunjukkan pada Tabel 5.3 dan Tabel 5.5, diperoleh jumlah konsumsi baterai dengan metode lokal sebesar 16.344%, metode *offloading* sebesar 5.218%, dan dengan *decision making* sebesar 8.440%. Hal ini menunjukkan hasil konsumsi baterai dengan *decision making* 7.904% lebih hemat dari metode lokal dan 3.222% lebih boros dari metode *offloading*.

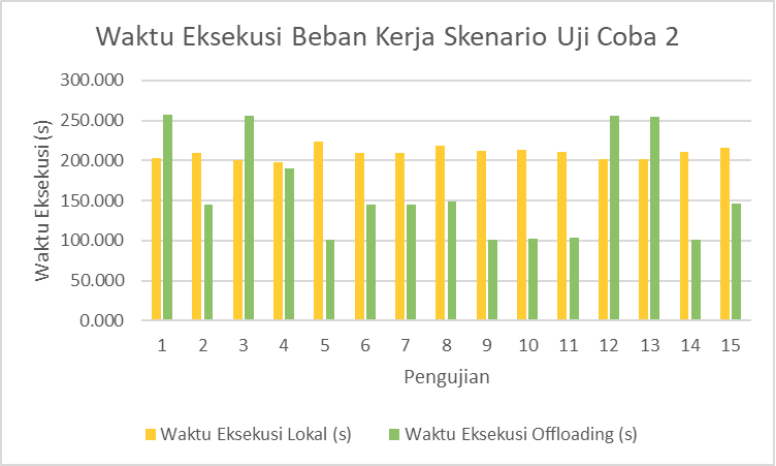
5.2.2 Skenario Uji Coba 2

Skenario uji coba 2 adalah pengujian yang dilakukan sebanyak 15 kali dengan kondisi RAM yang tersedia (5 *High*, 2 *Medium*, 8 *Low*), kondisi *download speed* (4 *High*, 3 *Medium*, 8 *Low*), kondisi *upload speed* (7 *High*, 2 *Medium*, 6 *Low*), dan jumlah gambar dari masing-masing pengujian sebanyak 7. Perhitungan waktu eksekusi yang dihasilkan oleh eksekusi beban kerja proses penghitungan jumlah mobil pada gambar dengan metode lokal dan *offloading* ditunjukkan pada Tabel 5.6.

Tabel 5.6 Waktu eksekusi beban kerja dengan metode lokal dan *offloading* pada Skenario Uji Coba 2

Pengujian	RAM yang tersedia	*Download Speed*	*Upload Speed*	Daya Konsumsi	Waktu Eksekusi	
	MB	(Mbit/s)	(Mbit/s)	(W)	Lokal (s)	*Offloading* (s)
1	55.112	0.070	0.160	2.612	203.523	256.701
2	245.448	6.080	0.260	2.569	209.671	144.866
3	224.720	0.070	0.140	2.508	200.395	255.799
4	53.284	0.060	2.180	2.672	198.175	190.086
5	118.104	13.340	5.750	2.773	223.825	100.993
6	76.280	6.640	0.260	2.535	209.170	144.869
7	32.740	0.130	7.280	2.717	209.863	144.686
8	248.480	0.120	7.800	2.678	219.035	148.796
9	142.692	5.210	8.010	2.794	211.659	101.270
10	247.016	5.160	6.090	2.708	213.387	102.739

11	139.000	5.130	2.340	2.643	210.073	103.575
12	29.768	0.070	0.140	2.676	201.605	256.128
13	47.184	0.060	0.160	2.523	201.183	255.131
14	180.244	13.480	8.200	2.746	210.603	101.357
15	48.728	0.130	7.580	2.705	215.222	145.805
Jumlah					3137.390	2452.801



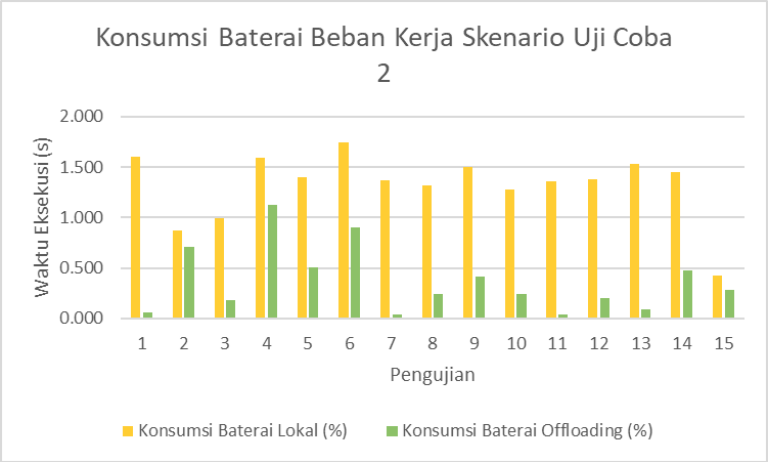
Gambar 5.13 Grafik waktu eksekusi beban kerja Skenario Uji Coba 2

Untuk perhitungan konsumsi baterai yang dihasilkan oleh eksekusi beban kerja proses penghitungan jumlah mobil pada gambar dengan metode lokal dan *offloading* ditunjukkan pada Tabel 5.7.

Tabel 5.7 Konsumsi Baterai beban kerja dengan metode lokal dan *offloading* pada Skenario Uji Coba 2

Pengujian	**RAM yang tersedia**	***Download Speed***	***Upload Speed***	**Daya Konsumsi**	**Konsumsi Baterai**	
	MB	**(Mbit/s)**	**(Mbit/s)**	**(W)**	**Lokal (%)**	***Offloading* (%)**
1	55.112	0.070	0.160	2.612	1.601	0.060
2	245.448	6.080	0.260	2.569	0.869	0.708

3	224.720	0.070	0.140	2.508	0.998	0.177
4	53.284	0.060	2.180	2.672	1.593	1.124
5	118.104	13.340	5.750	2.773	1.396	0.503
6	76.280	6.640	0.260	2.535	1.745	0.899
7	32.740	0.130	7.280	2.717	1.368	0.034
8	248.480	0.120	7.800	2.678	1.318	0.242
9	142.692	5.210	8.010	2.794	1.499	0.418
10	247.016	5.160	6.090	2.708	1.280	0.245
11	139.000	5.130	2.340	2.643	1.358	0.034
12	29.768	0.070	0.140	2.676	1.376	0.203
13	47.184	0.060	0.160	2.523	1.534	0.087
14	180.244	13.480	8.200	2.746	1.453	0.471
15	48.728	0.130	7.580	2.705	0.426	0.286
Jumlah					19.814	5.491



Gambar 5.14 Grafik konsumsi baterai beban kerja Skenario Uji Coba 2

Sedangkan hasil waktu eksekusi dengan *decision making* pada uji coba 2 ditunjukkan pada Tabel 5.8.

Tabel 5.8 Waktu eksekusi beban kerja dengan *decision making* pada Skenario Uji Coba 2

Pengujian	*Entropy Based Fuzzy* MCDM Skor		*Decision Making*	Waktu Eksekusi (s)
	Lokal	*Offloading*		
1	0.586	0.414	Lokal	203.523
2	0.586	0.414	Lokal	209.671
3	0.671	0.329	Lokal	200.395
4	0.500	0.500	*Offloading*	190.086
5	0.378	0.622	*Offloading*	100.993
6	0.500	0.500	*Offloading*	144.869
7	0.414	0.586	*Offloading*	144.686
8	0.500	0.500	*Offloading*	148.796
9	0.422	0.578	*Offloading*	101.270
10	0.459	0.541	*Offloading*	102.739
11	0.500	0.500	*Offloading*	103.575
12	0.500	0.500	*Offloading*	256.128
13	0.586	0.414	Lokal	201.183
14	0.414	0.586	*Offloading*	101.357
15	0.414	0.586	*Offloading*	145.805
Jumlah				2355.076

Untuk hasil konsumsi baterai dengan *decision making* pada uji coba 2 ditunjukkan pada Tabel 5.9.

Tabel 5.9 Konsumsi baterai beban kerja dengan *decision making* pada Skenario Uji Coba 2

Pengujian	*Entropy Based Fuzzy* MCDM Skor		*Decision Making*	Konsumsi baterai (%)
	Lokal	*Offloading*		
1	0.586	0.414	Lokal	1.601
2	0.586	0.414	Lokal	0.869
3	0.671	0.329	Lokal	0.998
4	0.500	0.500	*Offloading*	1.124
5	0.378	0.622	*Offloading*	0.503
6	0.500	0.500	*Offloading*	0.899
7	0.414	0.586	*Offloading*	0.034
8	0.500	0.500	*Offloading*	0.242
9	0.422	0.578	*Offloading*	0.418

10	0.459	0.541	*Offloading*	0.245
11	0.500	0.500	*Offloading*	0.034
12	0.500	0.500	*Offloading*	0.203
13	0.586	0.414	Lokal	1.534
14	0.414	0.586	*Offloading*	0.471
15	0.414	0.586	*Offloading*	0.286
Jumlah				9.461

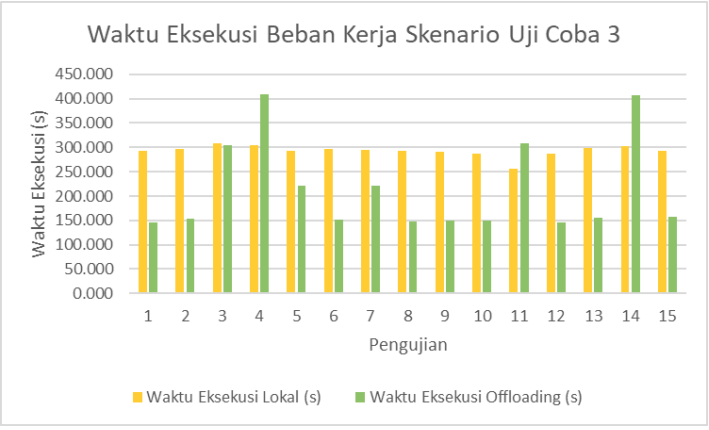
Berdasarkan hasil waktu eksekusi yang ditunjukkan pada Tabel 5.6 dan Tabel 5.8, diperoleh jumlah waktu eksekusi dengan metode lokal sebesar 3137.390 detik, metode *offloading* sebesar 2452.801 detik, dan dengan *decision making* sebesar 2355.076 detik. Hal ini menunjukkan hasil waktu eksekusi dengan *decision making* 782.313 detik lebih cepat dari metode lokal dan 97.725 detik lebih cepat dari metode *offloading*. Berdasarkan hasil konsumsi baterai yang ditunjukkan pada Tabel 5.7 dan Tabel 5.9, diperoleh jumlah konsumsi baterai dengan metode lokal sebesar 19.814%, metode *offloading* sebesar 5.491%, dan dengan *decision making* sebesar 9.461%. Hal ini menunjukkan hasil konsumsi baterai dengan *decision making* 10.353% lebih hemat dari metode lokal dan 3.970% lebih boros dari metode *offloading*.

5.2.3 Skenario Uji Coba 3

Skenario uji coba 3 adalah pengujian yang dilakukan sebanyak 15 kali dengan kondisi RAM yang tersedia (5 *High*, 4 *Medium*, 6 *Low*), kondisi *download speed* (2 *High*, 7 *Medium*, 6 *Low*), kondisi *upload speed* (6 *High*, 6 *Medium*, 3 *Low*), dan jumlah gambar dari masing-masing pengujian sebanyak 10. Perhitungan waktu eksekusi yang dihasilkan oleh eksekusi beban kerja proses penghitungan jumlah mobil pada gambar dengan metode lokal dan *offloading* ditunjukkan pada Tabel 5.10.

Tabel 5.10 Waktu eksekusi beban kerja dengan metode lokal dan *offloading* pada Skenario Uji Coba 3

Pengujian	RAM yang tersedia	*Download Speed*	*Upload Speed*	Daya Konsumsi	Waktu Eksekusi	
	MB	(Mbit/s)	(Mbit/s)	(W)	Lokal (s)	*Offloading* (s)
1	62.204	14.460	7.940	2.712	293.436	146.057
2	132.448	4.830	2.360	2.632	297.491	153.563
3	236.888	0.070	2.280	2.699	308.627	304.919
4	130.588	0.060	0.150	2.586	305.124	408.767
5	224.096	0.110	7.880	2.700	292.009	220.679
6	28.772	4.700	2.360	2.672	296.719	151.820
7	113.524	0.130	5.140	2.663	294.310	220.451
8	28.948	4.890	6.850	2.668	293.288	147.556
9	53.560	4.250	2.350	2.656	291.347	150.178
10	248.300	4.210	7.570	2.709	286.507	150.080
11	141.512	4.270	0.220	2.542	255.927	308.992
12	247.680	4.730	6.670	2.733	287.000	146.169
13	54.808	10.890	2.310	2.642	297.744	156.227
14	234.416	0.070	0.140	2.575	302.302	406.652
15	35.496	3.990	2.370	2.638	291.983	157.790
Jumlah					4393.814	3229.899

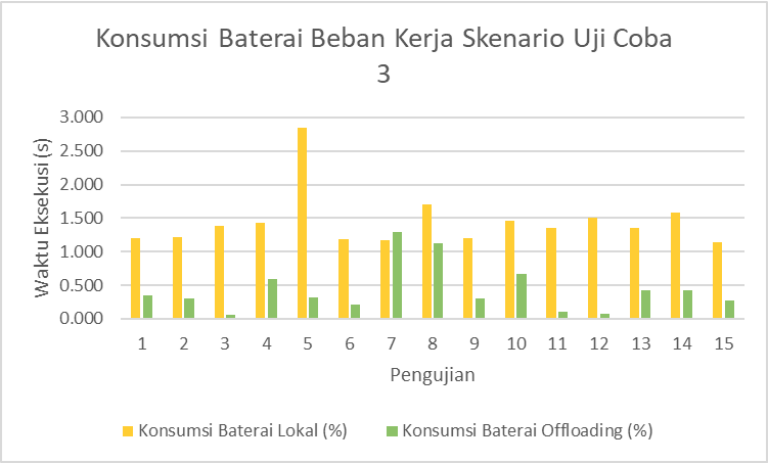


Gambar 5.15 Grafik waktu eksekusi beban kerja Skenario Uji Coba 3

Untuk perhitungan konsumsi baterai yang dihasilkan oleh eksekusi beban kerja proses penghitungan jumlah mobil pada gambar dengan metode lokal dan *offloading* ditunjukkan pada Tabel 5.11.

Tabel 5.11 Konsumsi Baterai beban kerja dengan metode lokal dan *offloading* pada Skenario Uji Coba 3

Pengujian	RAM yang tersedia	*Download Speed*	*Upload Speed*	Daya Konsumsi	Konsumsi Baterai	
	MB	(Mbit/s)	(Mbit/s)	(W)	Lokal (%)	*Offloading* (%)
1	62.204	14.460	7.940	2.712	1.203	0.344
2	132.448	4.830	2.360	2.632	1.209	0.303
3	236.888	0.070	2.280	2.699	1.383	0.057
4	130.588	0.060	0.150	2.586	1.424	0.598
5	224.096	0.110	7.880	2.700	2.847	0.316
6	28.772	4.700	2.360	2.672	1.182	0.209
7	113.524	0.130	5.140	2.663	1.168	1.296
8	28.948	4.890	6.850	2.668	1.698	1.129
9	53.560	4.250	2.350	2.656	1.194	0.300
10	248.300	4.210	7.570	2.709	1.455	0.670
11	141.512	4.270	0.220	2.542	1.359	0.104
12	247.680	4.730	6.670	2.733	1.509	0.070
13	54.808	10.890	2.310	2.642	1.347	0.417
14	234.416	0.070	0.140	2.575	1.584	0.419
15	35.496	3.990	2.370	2.638	1.137	0.271
Jumlah					21.699	6.503



Gambar 5.16 Grafik konsumsi baterai beban kerja Skenario Uji Coba 3

Sedangkan hasil waktu eksekusi dengan *decision making* pada uji coba 3 ditunjukkan pada Tabel 5.12.

Tabel 5.12 Waktu eksekusi beban kerja dengan *decision making* pada Skenario Uji Coba 3

Pengujian	*Entropy Based Fuzzy* MCDM Skor		*Decision Making*	Waktu Eksekusi (s)
	Lokal	*Offloading*		
1	0.365	0.635	*Offloading*	146.057
2	0.537	0.463	Lokal	297.491
3	0.586	0.414	Lokal	308.627
4	0.671	0.329	Lokal	305.124
5	0.545	0.455	Lokal	292.009
6	0.500	0.500	*Offloading*	151.820
7	0.541	0.459	Lokal	294.310
8	0.459	0.541	*Offloading*	147.556
9	0.500	0.500	*Offloading*	150.178
10	0.500	0.500	*Offloading*	150.080
11	0.586	0.414	Lokal	255.927
12	0.500	0.500	*Offloading*	146.169

13	0.459	0.541	*Offloading*	156.227
14	0.725	0.275	Lokal	302.302
15	0.500	0.500	*Offloading*	157.790
Jumlah				3261.666

Untuk hasil konsumsi baterai dengan *decision making* pada uji coba 3 ditunjukkan pada Tabel 5.13.

Tabel 5.13 Konsumsi baterai beban kerja dengan *decision making* pada Skenario Uji Coba 3

Pengujian	***Entropy Based Fuzzy* MCDM Skor**		***Decision Making***	**Konsumsi baterai (%)**
	Lokal	***Offloading***		
1	0.365	0.635	*Offloading*	0.344
2	0.537	0.463	Lokal	1.209
3	0.586	0.414	Lokal	1.383
4	0.671	0.329	Lokal	1.424
5	0.545	0.455	Lokal	2.847
6	0.500	0.500	*Offloading*	0.209
7	0.541	0.459	Lokal	1.168
8	0.459	0.541	*Offloading*	1.129
9	0.500	0.500	*Offloading*	0.300
10	0.500	0.500	*Offloading*	0.670
11	0.586	0.414	Lokal	1.359
12	0.500	0.500	*Offloading*	0.070
13	0.459	0.541	*Offloading*	0.417
14	0.725	0.275	Lokal	1.584
15	0.500	0.500	*Offloading*	0.271
Jumlah				14.384

Berdasarkan hasil waktu eksekusi yang ditunjukkan pada Tabel 5.10 dan Tabel 5.12, diperoleh jumlah waktu eksekusi dengan metode lokal sebesar 4393.814 detik, metode *offloading* sebesar 3229.899 detik, dan dengan *decision making* sebesar 3261.666 detik. Hal ini menunjukkan hasil waktu eksekusi dengan *decision making* 1132.148 detik lebih cepat dari metode lokal dan 31.767 detik lebih lambat dari metode *offloading*. Berdasarkan hasil konsumsi baterai yang ditunjukkan pada Tabel 5.11 dan Tabel 5.13, diperoleh jumlah konsumsi baterai dengan metode lokal

sebesar 21.699%, metode *offloading* sebesar 6.503%, dan dengan *decision making* sebesar 14.384%. Hal ini menunjukkan hasil konsumsi baterai dengan *decision making* 7.315% lebih hemat dari metode lokal dan 7.881% lebih boros dari metode *offloading*.

5.3 Evaluasi Umum Skenario Uji Coba

Hasil waktu eksekusi dari ketiga skenario dapat dilihat pada Tabel 5.14.

Tabel 5.14 Hasil waktu eksekusi pada skenario uji coba

Waktu Eksekusi (s)	**Skenario Uji Coba**		
	1	**2**	**3**
Lokal	1793.878	3137.390	4393.814
Offloading	1239.251	2452.801	3229.899
Decision Making	1259.754	2355.076	3261.666

Berdasarkan skenario uji coba 1 yang telah dilakukan, dapat diketahui waktu eksekusi dengan metode lokal sebesar 1793.878 detik, metode *offloading* sebesar 1239.251 detik, dan dengan *decision making* sebesar 1259.754 detik. Hal ini menunjukkan hasil waktu eksekusi dengan *decision making* 534.125 detik lebih cepat dari metode lokal dan 20.502 detik lebih lambat dari metode *offloading*.

Sedangkan pada skenario uji coba 2 yang telah dilakukan, waktu eksekusi dengan metode lokal sebesar 3137.390 detik, metode *offloading* sebesar 2452.801 detik, dan dengan *decision making* sebesar 2355.076 detik. Hal ini menunjukkan hasil waktu eksekusi dengan *decision making* 782.313 detik lebih cepat dari metode lokal dan 97.725 detik lebih cepat dari metode *offloading*.

Sedangkan pada skenario uji coba 3 yang telah dilakukan, waktu eksekusi dengan metode lokal sebesar 4393.814 detik, metode *offloading* sebesar 3229.899 detik, dan dengan *decision*

making sebesar 3261.666 detik. Hal ini menunjukkan hasil waktu eksekusi dengan *decision making* 1132.148 detik lebih cepat dari metode lokal dan 31.767 detik lebih lambat dari metode *offloading*. Dari setiap skenario uji coba, dengan *decision making* waktu eksekusi beban kerja lebih singkat dibandingkan dengan metode lokal saja namun dua dari tiga skenario ujicoba, dengan *decision making* waktu eksekusi beban kerja lebih lama dibandingkan dengan metode *offloading* saja.

Sedangkan untuk hasil konsumsi baterai dari ketiga skenario dapat dilihat pada Tabel 5.15.

Tabel 5.15 Hasil konsumsi baterai pada skenario uji coba

Konsumsi Baterai (%)	**Skenario Uji Coba**		
	1	**2**	**3**
Lokal	16.344	19.814	21.699
Offloading	5.218	5.491	6.503
Decision Making	8.440	9.461	14.384

Berdasarkan skenario uji coba 1 yang telah dilakukan, dapat diketahui jumlah konsumsi baterai dengan metode lokal sebesar 16.344%, metode *offloading* sebesar 5.218%, dan dengan *decision making* sebesar 8.440%. Hal ini menunjukkan hasil konsumsi baterai dengan *decision making* 7.904% lebih hemat dari metode lokal dan 3.222% lebih boros dari metode *offloading*.

Sedangkan pada skenario uji coba 2 yang telah dilakukan, dapat diketahui jumlah konsumsi baterai dengan metode lokal sebesar 19.814%, metode *offloading* sebesar 5.491%, dan dengan *decision making* sebesar 9.461%. Hal ini menunjukkan hasil konsumsi baterai dengan *decision making* 10.353% lebih hemat dari metode lokal dan 3.970% lebih boros dari metode *offloading*.

Sedangkan pada skenario uji coba 3 yang telah dilakukan, dapat diketahui jumlah konsumsi baterai dengan metode lokal

sebesar 21.699%, metode *offloading* sebesar 6.503%, dan dengan *decision making* sebesar 14.384%. Hal ini menunjukkan hasil konsumsi baterai dengan *decision making* 7.315% lebih hemat dari metode lokal dan 7.881% lebih boros dari metode *offloading*. Dari setiap skenario uji coba, dengan *decision making* konsumsi baterai beban kerja lebih hemat dibandingkan dengan metode lokal namun lebih boros dibandingkan dengan metode *offloading*.

BAB VI
KESIMPULAN DAN SARAN

Bab ini berisikan kesimpulan yang dapat dapat diambil dari hasil uji coba yang telah dilakukan. Selain kesimpulan, terdapat juga saran yang ditujukan untuk pengembangan perangkat lunak selanjutnya.

6.1 Kesimpulan

Kesimpulan yang didapatkan berdasarkan hasil uji coba peningkatan kinerja serta penghematan daya pada proses komputasi di raspberry pi menggunakan *computation offloading* yang diterapkan pada sistem adalah sebagai berikut:

1. Implementasi *computation offloading* dengan metode *Entropy Based Fuzzy* MCDM dapat menentukan beban kerja diproses secara lokal atau secara *offloading*, hal ini dapat meningkatkan performa proses komputasi beban kerja dibandingkan dengan performa lokal
2. Pada kasus Tugas Akhir ini, *computation offloading* meningkatkan kinerja proses komputasi dengan hasil waktu eksekusi lebih cepat dari metode lokal saja namun lebih lambat dari metode *offloading* saja pada uji coba 1 dan 3 serta lebih cepat dari metode lokal saja dan lebih cepat dari metode *offloading* saja pada uji coba 2. Akan tetapi, *computation offloading* tidak menghematkan daya dengan hasil konsumsi baterai dengan selisih konsumsi baterai lebih hemat dari metode lokal saja namun lebih boros dari metode *offloading* saja pada ketiga uji coba.
3. *Entropy Based Fuzzy* MCDM sebagai *Decision Maker* pada sistem ini dapat menentukan beban kerja diproses secara lokal atau secara *offloading* secara dinamis dengan memperhitungkan dari nilai faktor-faktor penentu metode *offloading*.

6.2 Saran

Saran yang diberikan terkait pengembangan pada Tugas Akhir ini adalah:

1. Menambah faktor-faktor penentu metode *offloading* pada *Entropy Based Fuzzy* MCDM agar penentuan pengeksekusian beban kerja lebih optimal.
2. Pengecekan nilai faktor-faktor penentu metode *offloading* agar lebih akurat.
3.

DAFTAR PUSTAKA

[1] K. K. Patel dan S. M. Patel, “Internet of Things: Definition, Characteristics, Architecture, Enabling Technologies, Application & Future Challenge,” *IJESQ*, p. 10, 2016.

[2] S. J. Johnson dan S. J. Cox, “The Raspberry Pi: A Technology Disrupter, and the,” *Electronics 2017*, vol. 6, no. 51, p. 7, 2017.

[3] A. Nagar, "Development of Fuzzy Multi Criteria Decision Making Method for Selection of Optimum Maintenance Alternative," *International Journal of Applied Research In Mechanical Engineering*, vol. 1, no. 2, pp. 87-92, 2011.

[4] A. Geiger, P. Lenz and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Computer Vision and Pattern Recognition*, Providence, RI, USA, 2012.

[5] Raspberry Pi, “Raspberry Pi FAQs - Frequently Asked Questions,” 8 April 2017. [Online]. Available: https://www.raspberrypi.org/documentation/faqs/. [Diakses 25 Desember 2018].

[6] Raspbian, “About Raspbian,” [Online]. Available: https://www.raspbian.org/RaspbianAbout. [Diakses 25 Desember 2018].

[7] Texas Instruments, “INA219 Zero-Drift, Bidirectional Current/Power Monitor With I2C Interface datasheet (Rev. G) - ina219,” Desember 2015. [Online]. Available: http://www.ti.com/lit/ds/symlink/ina219.pdf. [Diakses 4 Juni 2019].

[8] A. u. R. Khan, M. Othman, A. N. Khan, J. Shuja and S. Mustafa, "Computation Offloading Cost Estimation in Mobile Cloud Application Models," *Wireless*

Personal Communications, vol. 97, no. 3, pp. 4897-4920, 2017.

[9] H. Garg, N. Agarwal dan A. Choubey, "Entropy Based Multi-criteria Decision Making Method under Fuzzy," *Global Journal of Technology & Optimization*, vol. 6, no. 3, pp. 182-186, 2015.

[10] OpenCV, "About - OpenCV Library," 2018. [Online]. Available: https://opencv.org/about.html. [Diakses 25 Desember 2018].

[11] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto dan H. Adam, "[1704.04861] MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," 17 April 2017. [Online]. Available: https://arxiv.org/pdf/1704.04861.pdf. [Diakses 5 Juni 2019].

LAMPIRAN

(*Halaman ini sengaja dikosongkan*)

BIODATA PENULIS

Arya Wiranata merupakan anak dari pasangan Bapak Bambang Damyasik dan Ibu Wa Ode Ndipo Tayb. Lahir di Gresik pada tanggal 21 Februari 1998. Penulis menempuh pendidikan formal dimulai dari TK Petrokimia Gresik (2002-2004), SDN Petrokimia Gresik (2004-2010), SMPN 1 Gresik (2010-2013), SMAN 1 Gresik (2013-2015) dan S1 Departemen Informatika ITS (2015-2019). Bidang studi yang diambil oleh penulis pada saat berkuliah di Departemen Informatika ITS adalah Komputasi Berbasis Jaringan (KBJ). Penulis aktif dalam organisasi seperti Himpunan Mahasiswa Teknik Computer-Informatika (2016-2018). Penulis juga aktif dalam berbagai kegiatan kepanitiaan yaitu SCHEMATICS 2016, SCHEMATICS 2017, dan ITS EXPO 2017. Penulis memiliki hobi mendengarkan musik, menjelajahi internet, bermain game, dan menonton film. Penulis dapat dihubungi melalui email: wiranata.arya.wiranata@gmail.com.