

ITS
Institut
Teknologi
Sepuluh Nopember

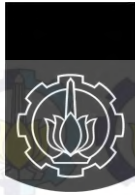
TUGAS AKHIR - TE 145561

**PEMBUATAN APLIKASI BERBASIS *WEB* MENGGUNAKAN
SISTEM *BOARD INTEL GALILEO* UNTUK MEMONITORING
SERTA KONTROL KONDISI LISTRIK DI PANEL DISTRIBUSI
TEGANGAN LISTRIK PLN**

Ifa Fitrianti
NRP 2213039001
Ilham Sudrajat
NRP 2213039006

Dosen Pembimbing
Suwito, S.T.,M.T.
Onie Meiyanto, S.Pd

PROGRAM STUDI D3 TEKNIK ELEKTRO
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya 2016



ITS
Institut
Teknologi
Sepuluh Nopember

FINAL PROJECT - TE 145561

MANUFACTURE OF WEB-BASED APPLICATIONS USING INTEL'S GALILEO SYSTEM FOR MONITORING AND CONTROL OF ELECTRICAL CONDITIONS IN THE POWER SUPPLY VOLTAGE DISTRIBUTION PANELS PLN

Ifa Fitrianti
NRP 2213039001
Ilham Sudrajat
NRP 2213039006

Advisor
Suwito, S.T., MT
Onie Meiyanto, S.Pd

ELECTRICAL ENGINEERING D3 STUDY PROGRAM
Faculty of Industrial Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2016

PERNYATAAN KEASLIAN TUGAS AKHIR

Dengan ini kami menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir kami dengan judul “Pembuatan Aplikasi Berbasis *Web* Menggunakan *Board Intel Galileo* Untuk Memonitoring Serta Kontrol Kondisi Listrik Di Panel Distribusi Tegangan Listrik PLN” adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diijinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka.

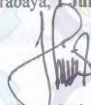
Apabila ternyata pernyataan ini tidak benar, kami bersedia menerima sanksi sesuai peraturan yang berlaku.

Surabaya, 1 Juni 2016



Ifa Fitrianti
NRP 2213039001

Surabaya, 1 Juni 2016



Ilhami Sudrajat
NRP 2213039006

**PEMBUATAN APLIKASI BERBASIS WEB MENGGUNAKAN
BOARD INTEL GALILEO UNTUK MEMONITORING SERTA
KONTROL KONDISI LISTRIK DI PANEL DISTRIBUSI
TEGANGAN PLN**

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Ahli Madya Teknik

Pada
Bidang Studi Elektro Industri
Program Studi D3 Teknik Elektro
Institut Teknologi Sepuluh Nopember

Menyetujui:

Dosen Pembimbing 1

Dosen Pembimbing 2

Suwito, S.T., M.T.

NIP. 19810529 199512 1 001

Onie Melvanto, S.Pd

NIP. 19850501 201101 1 008

**SURABAYA
JUNI, 2016**



PEMBUATAN APLIKASI BERBASIS WEB MENGGUNAKAN BOARD INTEL GALILEO UNTUK MEMONITORING SERTA KONTROL KONDISI LISTRIK DI PANEL DISTRIBUSI TEGANGAN LISTRIK PLN

Nama : Ifa Fitrianti
Nama : Ilham Sudrajat
Pembimbing 1: Suwito, S.T,M.T.
Pembimbing 2: Onie Meiyanto, S.Pd

ABSTRAK

Untuk memenuhi kebutuhan masyarakat terhadap listrik, PLN berusaha untuk memberikan pelayanan yang terbaik untuk para pelanggan dengan meminimalis terjadinya gangguan pada jaringan yang menyebabkan ketidak puasan pelanggan terhadap pelayanan yang diberikan oleh PLN, sedangkan sistem *Monitoring* kondisi panel distribusi yang dilakukan PLN selama ini masih menggunakan sistem konvensional dan belum adanya sistem kontrol untuk mengontrol kondisi panel distribusi PLN. Dalam sistem konvensional memiliki banyak kelemahan, yaitu belum adanya alarm atau indikator indikator yang tertampil dipusat pemantauan distribusi jika terjadi gangguan pada panel distribusi PLN, sehingga penanganan terhadap gangguan tersebut sering tertunda. Berdasarkan kondisi tersebut pada Tugas Akhir ini akan dibuat sistem *monitoring* serta kontrol panel distribusi PLN berbasis *web* menggunakan sistem *board intel galileo*. Sistem ini memanfaatkan layanan jaringan PT. Indonesia *Connets Plus (ICON+)* sebagai media komunikasi antara panel distribusi dengan pusat pemantauan distribusi. Pada alat yang telah dibuat. Alat ini memiliki kemampuan untuk memonitoring jaringan distribusi tegangan rendah, dan sebagai kontrol untuk memutus dan menyambungkan arus listrik jika terjadi gangguan ataupun saat diperlukan. Hasil yang diperoleh dari alat yang telah dibuat. Dihasilakan pengujian sensor arus 1,5 % dan sensor tegangan didapatkan hasil eror 0 % dari hasil tersebut, dapat mengetahui gangguan pada panel distribusi PLN dengan cepat dan dapat langsung diperbaiki oleh petugas PLN demi kepuasan pelanggan terhadap pelayanan PLN yang diberikan.

Kata Kunci : *Intel Galileo, Sensor Arus ACS712 , Monitoring , Kontrol*

MANUFACTURE OF WEB-BASED APPLICATIONS USING INTEL'S GALILEO SYSTEM FOR MONITORING AND CONTROL OF ELECTRICAL CONDITIONS IN THE POWER SUPPLY VOLTAGE DISTRIBUTION PANELS PLN

Name : Ifa Fitrianti

Name : Ilham Sudrajat

Advisor 1 : Suwito, S.T.,M.T.

Advisor 2 : Onie Meiyanto, S.Pd.

ABSTRACT

To meet the public demand for electricity, PLN strive to provide the best service to its customers by minimizing the disruption of the tissue which causes dissatisfaction of customers with the services provided by PLN, meanwhile the condition of monitoring system of distribution panel that conducted by PLN still use the conventional system and there is no control system to control the condition of PLN's distribution panel. conventional systems have many shortcomings such as the absence of an alarm or indicators which is displayed in monitoring center if there a disruption in the distribution panel of PLN, so that the handling of the disorder is often delayed. Under these conditions, in this final project will be made monitoring system and PLN's panel controller using the web-based distribution of Galileo intel board system. This system utilizes a network service by PT. Indonesia Comnets Plus (ICON +) as a medium communication between the central distribution panel with the monitoring distribution. On tools that have been created. This tool has the ability to monitor low voltage distribution network, and as a control to disconnect and connect the electrical current if an interruption occurs or when necessary. The results that obtained from the tools that have been made is a current sensor test of 1.5% and a voltage sensor test showed the error of 0%, an ability to initialize the PLN disruption of distribution panels quickly and immediately be rectified by PLN officer for customer satisfaction with the services provided by PLN.

Keywords : Intel Galileo, Flow Sensor ACS712, Monitoring, Control



KATA PENGANTAR

Puji syukur penulis panjatkan kehadiran Allah SWT yang selalu memberikan rahmat dan hidayah-Nya sehingga Tugas Akhir ini dapat terselesaikan dengan baik. Shalawat serta salam semoga selalu dilimpahkan kepada Rasulullah Muhammad SAW, keluarga, sahabat, dan umat muslim yang senantiasa meneladani beliau.

Tugas Akhir ini disusun untuk memenuhi sebagian persyaratan guna menyelesaikan pendidikan Diploma-3 pada Bidang Studi Elektro Industri, Jurusan Teknik Elektro, Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember Surabaya dengan judul:

PEMBUATAN APLIKASI BERBASIS WEB MENGGUNAKAN BOARD INTEL GALILEO UNTUK MEMONITORING SERTA KONTROL KONDISI LISTRIK DI PANEL DISTRIBUSI TEGANGAN LISTRIK PLN

Dalam Tugas Akhir ini dirancang alat penghitung penggunaan daya dan biaya penggunaan listrik pada kamar kost menggunakan ethernet sebagai media komunikasinya yang dapat diakses melalui website.

Penulis mengucapkan terima kasih kepada Ibu dan Bapak penulis yang memberikan berbagai bentuk doa serta dukungan tulus tiada henti, Bapak Suwito, S.T., M.T. dan Bapak Onie Meiyanto, S.Pd. atas segala bimbingan ilmu, moral, dan spiritual dari awal hingga terselesaikannya Tugas Akhir ini. Penulis juga mengucapkan banyak terima kasih kepada teman-teman yang telah membantu baik secara langsung maupun tidak langsung dalam proses penyelesaian Tugas Akhir ini.

Penulis menyadari dan memohon maaf atas segala kekurangan pada Tugas Akhir ini. Akhir kata, semoga Tugas Akhir ini dapat bermanfaat dalam pengembangan keilmuan di kemudian hari.

Surabaya, 1 Juni 2016

Penulis

DAFTAR ISI

	HALAMAN
HALAMAN JUDUL.....	i
HALAMAN JUDUL.....	iii
PERNYATAAN KEASLIAN TUGAS AKHIR.....	v
HALAMAN PENGESAHAN.....	vii
DAFTAR ISI.....	xv
DAFTAR GAMBAR.....	xvii
DAFTAR TABEL.....	xix
BAB I PENDAHULUAN.....	21
1.1 Latar Belakang.....	21
1.2 Permasalahan.....	2
1.3 Batasan Masalah.....	2
1.4 Tujuan.....	2
1.5 Metodologi.....	2
1.6 Sistematika Laporan.....	4
1.7 Relevansi.....	4
BAB II TEORI DASAR.....	5
2.1 Tinjauan Pustaka.....	5
2.2 Gardu Distribusi.....	6
2.3 Sensor Arus.....	8
2.4 Voltage Divider.....	11
2.5 Intel Galileo.....	13
2.6 Javascript.....	16
2.7 Bootstrap.....	17
2.8 Pubnub.....	17
2.9 Jade (java agent development framemwork).....	18
2.10Peak Detector.....	19
2.11Optocoupler.....	20
2.12MQTT (Message Queuing Telemetry Transport).....	22
BAB III PERANCANGAN SISTEM KONTROL.....	25
3.1 Blok Fungsional Sistem.....	25
3.2 Perancangan Elektronik.....	27
3.2.1 Perancangan Pengondisi Sinyal Sensor Arus ACS712.....	28
3.2.2 Perancangan Sensor Tegangan.....	30

3.2.3	Perancangan Power Supply	31
3.2.4	Perancangan Breaker.....	32
3.2.5	Perancangan Panel	34
3.2.6	Perancangan Buzzer.....	35
3.3	Perancangan Mekanik.....	36
3.3.1	Panel.....	37
3.4	Perancangan Perangkat Lunak (Software)	38
3.4.1	Percobaan Koneksi Intel Galileo dengan PC	40
3.4.2	Pemrograman Blink Input-Output	40
3.4.3	Pemrograman Pembacaan ADC	41
3.4.4	Pemrograman Tampilan HMI	41
BAB IV	HASIL SIMULASI DAN IMPLEMENTASI	45
4.1	Pengujian Sensor Arus ACS172	45
4.2	Kalibrasi Pengujian Sensor Arus ACS172.....	46
4.3	Pengujian Sensor Tegangan.....	52
4.4	Kalibrasi Pengujian Tegangan	52
4.5	Pengujian Supply Rangkaian Pengondisi Sinyal ACS712.....	59
4.6	Pengujian Driver Relay	59
4.7	Pengujian Sistem Intel Galileo.....	59
4.8	Pengujian Keseluruhan	63
BAB V	PENUTUP	69
5.1	Kesimpulan	69
5.2	Saran.....	69
DAFTAR	PUSTAKA	71
LAMPIRAN A	GAMBAR	A-1
LAMPIRAN B	PROGRAM	B-1
LAMPIRAN C	<i>DATASHEET</i>	C-1
DAFTAR	RIWAYAT HIDUP	D-1

DAFTAR GAMBAR

HALAMAN

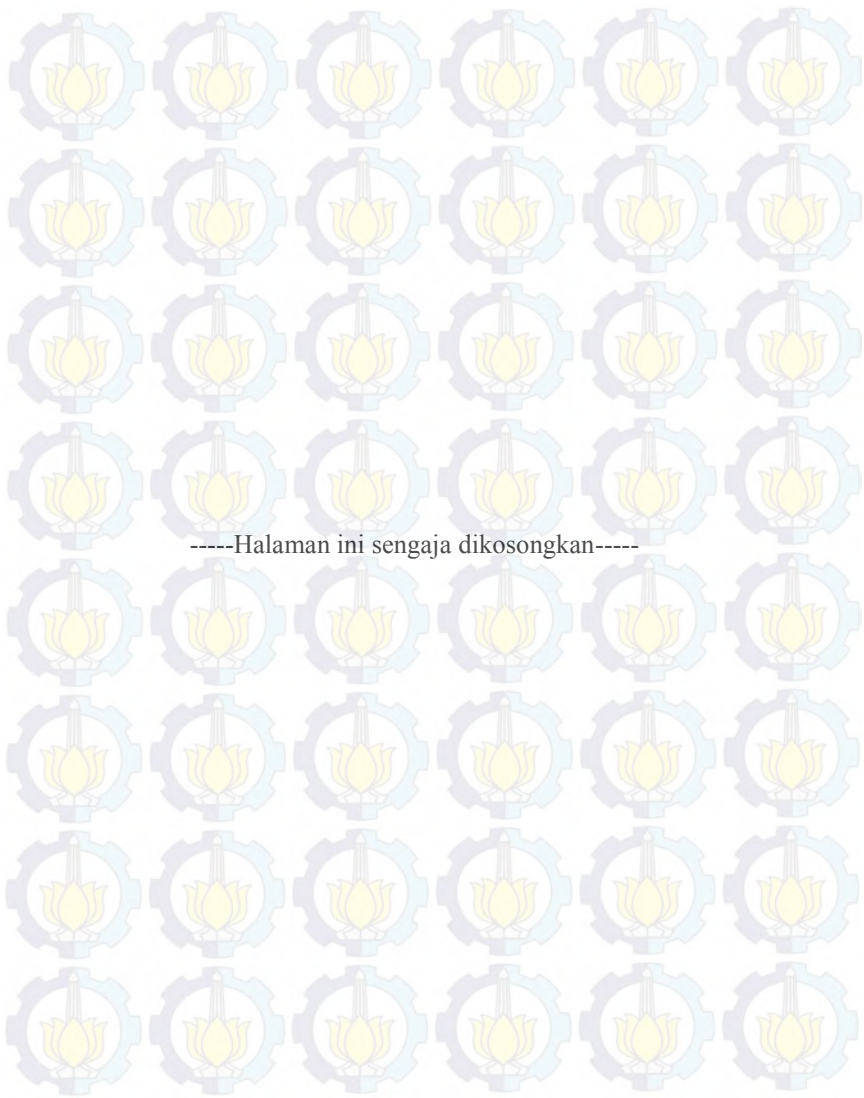
Gambar 2.1	Trafo Distribusi	7
Gambar 2.2	Panel Distribusi PLN	7
Gambar 2.3	Sensor Arus ACS712	9
Gambar 2.4	Contoh Skema ACS712	10
Gambar 2.5	Blok Diagram Sensor Arus ACS712	11
Gambar 2.6	Rangkaian Voltage Divider	12
Gambar 2.7	Intel Galileo	13
Gambar 2.8	Skema Intel Galileo	15
Gambar 2.9	Rangkaian Peak Detector	19
Gambar 2.10	Peak Detector	20
Gambar 2.11	Gelombang Peak Detector	20
Gambar 2.12	Simbol Dan Bentuk Optocoupler	21
Gambar 2.13	Rangkaian Optocoupler	21
Gambar 2.14	Sistem Umum <i>IoT</i> Memakai <i>MQTT</i>	23
Gambar 3.1	Blok Fungsional	25
Gambar 3.2	Flowchart Pada Hardware	27
Gambar 3.3	Diagram Koneksi Sensor Arus ACS712	28
Gambar 3.4	Skematik Pengondisi Sinyal Output ACS712	29
Gambar 3.5	Diagram Analog Sensor Tegangan	30
Gambar 3.6	Skematik Sensor Tegangan	30
Gambar 3.7	Diagram Power Supply	31
Gambar 3.8	Skematik Rangkaian Power Supply	32
Gambar 3.9	Rangkaian Breaker	32
Gambar 3.10	Skematik Driver Breaker	33
Gambar 3.11	Driver Relay	34
Gambar 3.12	Rangkaian Panel	35
Gambar 3.13	Rangkaian Buzzer	35
Gambar 3.14	Box Panel Tampak Depan	36
Gambar 3.15	Box Panel Tampak Samping	37
Gambar 3.16	Panel	38
Gambar 3.17	Diagram Alir program keseluruhan pada Intel Galileo ...	39
Gambar 3.18	Flowchart Visual	42

Gambar 3.19 Tampilan HMI	43
Gambar 4.1 Hasil Gelombang Pengujian ACS712	45
Gambar 4.2 Grafik Sensor Arus Fasa R	48
Gambar 4.3 Grafik Sensor Arus Fasa S	50
Gambar 4.4 Grafik Sensor Arus Fasa T	52
Gambar 4.5 Grafik Sensor Tegangan Pada Fasa R	54
Gambar 4.6 Grafik sensor tegangan Pada Fasa S	55
Gambar 4.7 Grafik Sensor Tegangan Pada Fasa T	57
Gambar 4.8 Output dari Trafo	57
Gambar 4.9 Bentuk Gelombang Tegangan AC	58
Gambar 4.10 Output tegangan DC	58
Gambar 4.11 Uji Komunikasi	60
Gambar 4.12 Uji IO Pada Dengan LED Blink	61
Gambar 4.13 Uji Analog Input	62
Gambar 4.14 Uji Web	63
Gambar 4.15 Kondisi Breaker Terputus	64
Gambar 4.16 Kondisi Normal Breaker Terputus	64
Gambar 4.17 Kondisi Breaker Terhubung	65
Gambar 4.18 Kondisi Normal Breaker Terhubung	65
Gambar 4.19 Status Alarm Menyala	66
Gambar 4.20 Status Alarm Mati	66

DAFTAR TABEL

HALAMAN

Tabel 4.1	Kalibrasi Sensor Arus ACS712 Fasa R.....	46
Tabel 4.2	Hasil pengujian Sensor Arus Fasa R.....	47
Tabel 4.3	Kalibrasi Sensor Arus ACS712 Pada Fasa S	48
Tabel 4.4	Hasil pengujian Sensor Arus Fasa S	49
Tabel 4.5	Kalibrasi Sensor Arus Pada Fasa T	50
Tabel 4.6	Hasil Pengujian Sensor Arus Fasa T	51
Tabel 4.7	Kalibrasi Tegangan Pada Fasa R	53
Tabel 4.8	Hasil Pengujian Sensor tegangan Pada Fasa R	53
Tabel 4.9	Kalibrasi Tegangan Pada Fasa S	54
Tabel 4.10	Hasil Pengujian Sensor Tegangan Pada Fasa S	55
Tabel 4.11	Kalibrasi Tegangan Pada Fasa T	56
Tabel 4.12	Hasil Pengujian Sensor Tegangan Pada Fasa T	56
Tabel 4.13	Tabel Pengondisi Sinyal	59
Tabel 4.14	Tabel Driver Relay.....	59



BAB I

PENDAHULUAN

1.1 Latar Belakang

Sistem Distribusi merupakan bagian penting dari sistem tenaga listrik. Sistem distribusi ini berguna untuk menyalurkan tenaga listrik dari sumber daya listrik besar sampai ke konsumen. Jadi fungsi distribusi tenaga listrik adalah pembagian atau penyaluran tenaga listrik ke beberapa tempat (pelanggan) dan merupakan sub sistem tenaga listrik yang langsung berhubungan dengan pelanggan, karena catu daya pada pusat-pusat beban (pelanggan) dilayani langsung melalui jaringan distribus. Apabila terjadi gangguan pada jaringan distribusi maka sangat merugikan para pelanggan yang merasakan dampak gangguan tersebut, dan selama ini sistem *monitoring* kondisi panel distribusi yang dilakukan PLN masih menggunakan Sistem konvensional. Sistem ini memiliki banyak kekurangan bila sistem mengalami gangguan, tidak ada alarm atau indikator yang tertampil di pusat pemantauan distribusi. Sehingga dalam pemantauan pada distribusi masih kurang dalam pelayanannya ke masyarakat. Contohnya di daerah terpencil bila ada gangguan jaringan penanganan sering tertunda, dan sistem kontrol berfungsi untuk menyambung dan memutuskan arus listrik dari panel distribusi ke pelanggan jika terjadi suatu kebakaran pada salah satu pelanggan, menjegah terjadinya kebakaran pada listrik secara luas. Hal tersebut yang mengganggu kenyamanan masyarakat dalam menggunakan energi listrik. Berdasarkan kondisi tersebut pada Tugas Akhir ini akan dibuat sistem *monitoring* serta kontrol panel distribusi PLN Berbasis *Web* Menggunakan Sistem *Board Intel Galileo*. Pada alat yang telah dibuat menyambungkan modem ke *intel galileo* menggunakan MQTT (*Message Queuing Telemetry Transport*) kemudian ke komputer sehingga dapat dapat di *monitoring* serta kontrol oleh operator. Sistem ini memanfaatkan layanan jaringan PT. Indonesia *Comnets Plus* (ICON+) sebagai media komunikasi antara panel distribusi dengan pusat pemantauan distribusi. Alat ini memiliki kemampuan untuk *monitoring* jaringan distribusi tegangan rendah, dan sebagai kontrol jarak jauh pengaman apabila terjadi arus lebih dalam keadaan darurat untuk segera ditangani oleh petugas PLN.

1.2 Permasalahan

Permasalahan pada Tugas Akhir ini adalah sebagai berikut :

1. Tidak adanya Monitoring Serta Kontrol Jarak Jauh Untuk Distribusi Tenaga Listrik.
2. Tidak adanya sistem kontrol pada jaringan maupun peralatan pada tegangan listrik.
3. Tidak adanya sistem otomatis untuk memutus dan menyambungkan arus listrik pada gardu distribusi.
4. Tidak adanya pengaman jaringan, jika terjadi gangguan arus berlebih pada gardu distribusi listrik PLN.

1.3 Batasan Masalah

Dari perumusan masalah di atas, maka batasan masalah dari tugas akhir ini adalah :

1. Komunikasi antara perangkat dengan komputer master menggunakan jaringan ethernet.
2. Penggunaan *web browser* sebagai tampilan pada komputer master.
3. Pengembangan Instrumentasi peralatan tegangan tinggi menggunakan perangkat IOT.
4. Maksimal arus yang dapat dihantarkan sebesar 3 ampere pada tegangan 3 fasa dengan range 0-250Volt.

1.4 Tujuan

Tujuan kami membuat Tugas Akhir ini adalah sebagai berikut :

1. Merancang dan membuat sistem Aplikasi Berbasis *Web* Menggunakan Sistem *Board Intel Galileo* Untuk Memonitor Serta Mengontrol Kondisi Listrik Di Panel Distribusi Tegangan Listrik PLN.
2. Memberikan sistem kontrol pada jaringan agar panel dapat di kontrol dengan jarak jauh
3. Memonitoring kondisi listrik di panel distribusi tegangan listrik PLN.
4. Sebagai pengaman jaringan apabila terjadi gangguan arus lebih

1.5 Metodologi

Dalam pelaksanaan tugas akhir yang berupa Aplikasi Berbasis *Web* Menggunakan Sistem *Board Intel Galileo* Untuk Memonitor Serta

Kontrol Kondisi Listrik Di Panel Distribusi Tegangan Listrik PLN., ada beberapa kegiatan yang dapat diuraikan sebagai berikut :

1. Studi Pustaka dan Survey Data Awal
Studi pustaka yang dilakukan dalam hal ini adalah dengan mencari literatur di ruang baca, perpustakaan atau di internet mengenai cara kerja trafo arus, trafo tegangan, desain *web* dari *intel galileo*, teknik pembacaan analog dan digital serta interfacing dari sistem *galileo*. Materi yang dicari meliputi bagaimana teknik pengukuran arus dan tegangan serta sistem isolasi agar dapat dirubah menjadi level tegangan TTL sehingga dapat dibaca oleh prosesor *intel galileo*.
2. Perencanaan dan Pembuatan Alat
Perencanaan alat dilaksanakan setelah studi pustaka memenuhi semua kriteria dalam perancangan alat. Meliputi perancangan aplikasi berbasis *web* menggunakan sistem *board intel galileo* untuk memonitor serta kontrol kondisi listrik di panel distribusi tegangan listrik PLN, perancangan sistem kontrol arus dan tegangan pada listrik.
3. Perencanaan dan Pembuatan *Software*
Pembuatan *software* dilakukan setelah perancangan alat selesai dilakukan, perancangan *software* ini bertujuan untuk memberikan perintah pada sistem mikrokontroller *galileo* agar dapat bekerja membaca semua perintah untuk mengakses IO pada prosesor agar dapat membaca tegangan, arus, serta status kondisi dari panel.
4. Uji Coba dan Analisis Data
Uji coba dan analisis data dilakukan dengan cara mengaplikasikan langsung aplikasi berbasis *web* menggunakan sistem *board intel galileo* untuk memonitor serta kontrol kondisi listrik di panel distribusi tegangan listrik PLN dengan cara membaca arus dan tegangan yang kemudian di kirimkan datanya melalui *ethernet* ke komputer server.
5. Penyusunan Laporan
Penyusunan laporan dilakukan setelah beberapa data pengujian aplikasi berbasis *web* menggunakan sistem *board intel galileo* untuk memonitor serta kontrol kondisi listrik di panel distribusi tegangan listrik PLN sudah mencukupi.

1.6 Sistematika Laporan

Pembahasan Tugas Akhir ini akan dibagi menjadi lima Bab dengan sistematika sebagai berikut:

Bab I Pendahuluan

Bab ini meliputi latar belakang, permasalahan, tujuan penelitian, metodologi penelitian, sistematika laporan, dan relevansi.

Bab II Teori Dasar

Membahas tentang teori - teori penunjang yang diperlukan dan dipergunakan sebagai penunjang pengerjaan Tugas Akhir

Bab III Perancangan Sistem

Membahas tentang perencanaan dan pembuatan perangkat keras (*hardware*) yang terdiri dari perancangan elektronik dan perancangan mekanik serta pembuatan dan perancangan perangkat lunak (*software*).

Bab IV Simulasi, Implementasi dan Analisis Sistem

Bab ini memuat hasil simulasi dan implementasi serta analisis dari hasil tersebut.

Bab V Penutup

Bab ini berisi kesimpulan dan saran dari hasil pembahasan yang telah diperoleh.

1.7 Relevansi

1. Memudahkan para teknisi PLN untuk mengetahui letak gangguan listrik sehingga pada saat gangguan itu terjadi teknisi bisa langsung mengetahui dan bisa langsung memperbaikinya.
2. Memudahkan untuk menyalakan dan mematikan aliran tenaga listrik distribusi
3. Memberikan kemudahan untuk memonitoring serta mengontrol jaringan distribusi PLN.

BAB II TEORI DASAR

2.1 Tinjauan Pustaka

Ada beberapa penelitian yang telah dilakukan sebelumnya untuk menyelesaikan masalah *monitoring gardu distribusi* pada PLN. Di antaranya menggunakan berbasis *web* dan mikrokontroler Atmega32, Metode ini digunakan untuk memonitoring gardu induk dan gardu distribusi PLN. Hasil yang dicapai terdapat kekurangan, yaitu belum adanya kontroler, untuk mengontrol kondisi gardu distribusi PLN.

Penelitian di lakukan oleh Dadang Iskandar dalam seminar nasional informatika 2011 tentang “Sistem Informasi Gardu Induk Dan Gardu Distribusi PLN”, dengan menggunakan sensor arus, sensor tegangan dan sensor suhu, menggunakan metode *mikrokontroler* keluaran dari sensor tersebut di baca oleh mikrokontroler pada antarmuka berbasis *java*. Data yang dihasilkan penelitian tersebut diantaranya arus dan tegangan. Pengambilan data data sensor arus yang dilakukan dengan melakukan percobaan sementara dengan menggunakan beberapa peralatan diantaranya menggunakan beban strika dan beban berupa computer, dari beban setrika yang dilakukan secara manual 1.35 A sedangkan secara monitoring 1.25, beban komputer secara manual Komputer 1.22 A secara monitoring 1.20, Pada pengukuran yang dilakukan masih ada perbedaan hal ini dikarenakan kalibrasi alat yang dilakukan masih belum sempurna. data yang diperoleh oleh sensor tegangan pengukuran yang dilakukan secara manual dan *realtime* namun masih menggunakan satu input. Trafo yang digunakan menggunakan trafo *step down* biasa dengan tegangan *output* 3 V dan untuk *realtime* hasil yang didapatkan pada percobaan tersebut mata tidak dapat membedakan *delay* dari proses pengiriman data. Sehingga data yang dikirimkan dari RTU langsung ditampilkan kedalam *website* sistem informasi monitoring gardu induk dan gardu distribusi. Dari semua data yang di dapatkan masih belum sempurna karen masih terdapat banyak kurang dalam pengambilan data terutama dalam pengambilan data pada sensor arus.[1]

Pada penelitian ke dua di lakukan oleh Riza Agung Firmansyah, Titiek Suheta, Dedi Antoni Pada Seminar Nasional Sains Dan Teknologi Terapan III 2015 Tentang “Perancangan Alat Monitoring Dan Penyimpan Data Pada Panel Hubung Tegangan Rendah Di Trafo Gardu

Distribusi Berbasis Mikrokontroler” dari penelitian tersebut dihasilkan data Besar nilai arus pada fasa R = 43,8 A, fasa S= 45,8 A dan fasa T = 45 A, untuk nilai tegangan pada saat beban puncak terjadi pada fasa R = 222,7 volt, fasa S = 223,4 volt dan fasa T = 222,5 volt dengan proses pengukuran *line to netral* / 1 Fasa R, S dan T. Besar nilai daya pada saat beban puncak pada fasa R = 9754,26 VA, S= 10231 VA dan fasa T = 10013 VA, sehingga di ketahui prosentase pembebanan pada trafo sebesar 60% dan pembebanan tersebut masih normal. Kesimpulan dari penelitian ke dua Perlu adanya monitoring dengan mengirim data dengan sistem *online*. Perlu penambahan fitur – fitur baru untuk lebih memudahkan operator dalam memantau kondisi transformator seperti dapat memprediksi umur transformator.[9]

Pada Tugas Akhir ini akan dilakukan pembuatan aplikasi untuk memonitoring dan mengontrol panel distribusi tegangan listrik PLN menggunakan metode *web intel galileo*. Pada Tugas Akhir ini selain memonitoring juga mengontrol kondisi panel gardu distribusi tegangan listrik PLN, rangkaian breaker pada Tugas Akhir ini berfungsi untuk memutus dan menyambungkan arus listrik, dan memberi rangkaian pengaman pada panel distribusi gardu PLN untuk memutus arus listrik jika terjadi arus berlebih maupun arus yang kurang dari ketentuan untuk memutus dan menyambungkan arus listrik. Hasil yang diharapkan dari Tugas Akhir ini adalah untuk memberi pelayanan yang baik untuk masyarakat, dan mempermudah para teknisi pegawai PLN untuk mengontrol dan memonitoring kondisi panel distribusi tegangan listrik PLN.

2.2 Gardu Distribusi[2]

Gardu distribusi Merupakan salah satu Komponen dari suatu sistem distribusi PLN yang berfungsi untuk menghubungkan jaringan ke Konsumen atau untuk mendistribusikan tenaga listrik pada konsumen atau pelanggan, baik itu pelanggan tegangan menengah maupun pelanggan tegangan rendah. Pengertian Gardu Distribusi tegangan Listrik yang Paling di kenal adalah sebuah bangunan Gardu Listrik yang berisi atau terdiri dari instalasi Perlengkapan Hubung Bagi Tegangan Menengah (*PHB-TM*), Transformator Distribusi, dan Perlengkapan Hubung Bagi Tegangan Rendah (*PHBTR*) Untuk memasok kebutuhan tenaga listrik bagi para pelanggan baik dengan tegangan menengah (TM 20 KV) maupun Tegangan rendah (TR 220/380 Volt).



Gambar 2.1 Trafo Distribusi

Gambar 2.1 menjelaskan tentang gardu distribusi yang menggunakan *transformator* distribusi berfungsi untuk menurunkan tegangan listrik dari jaringan distribusi tegangan tinggi menjadi tegangan terpakai pada jaringan distribusi tegangan rendah (*step down transformator*); misalkan tegangan 20 KV menjadi tegangan 380 volt atau 220 volt. Sedang transformator yang digunakan untuk menaikkan tegangan listrik (*step up transformator*), hanya digunakan pada pusat pembangkit tenaga listrik agar tegangan yang didistribusikan pada suatu jaringan panjang (*long line*) tidak mengalami penurunan tegangan (*voltage drop*) yang berarti; yaitu tidak melebihi ketentuan *voltage drop* yang diperkenankan 5% dari tegangan semula. Dalam Gardu Distribusi ini Terdapat Macam macam komponen dan jenis jenis bagian dari sistem distribusi tegangan listrik.



Gambar 2.2 Panel Distribusi PLN

Gambar 2.2 menjelaskan tentang pengertian dari Operasi Sistem Distribusi adalah segala kegiatan yang mencakup pengaturan, pembagian, pemindahan, dan penyaluran tenaga listrik dari pusat pembangkit kepada konsumen dengan efektif serta menjamin

kelangsungan penyalurannya / pelayanannya. Sebagai tolak ukur pada kegiatan operasi terdapat beberapa parameter, yaitu :

1. Mutu Listrik

Ada 2 hal yang menjadi ukuran mutu listrik yaitu tegangan dan frekuensi. Batas toleransi tegangan pelayanan yaitu pada konsumen TM adalah $\pm 5\%$, dan pada konsumen TR adalah maksimum 5% dan minimum 10% . Sedangkan untuk batas toleransi frekuensi adalah $\pm 1\%$ dari frekuensi standar 50 Hz.

2. Keandalan Penyaluran Tenaga Listrik

Sebagai indikator keandalan penyaluran adalah angka lama pemadaman / gangguan atau yang disebut Sistem *Average Interruption Duration Index (SAIDI)* dan angka seringnya pemadaman / gangguan atau yang disebut Sistem *Average Interruption Frequency Index (SAIFI)*.

3. Keamanan dan Keselamatan

Sebagai indikator dari keamanan dan keselamatan adalah jumlah angka kecelakaan akibat listrik pada personel dan kerusakan pada instalasi / peralatan serta pada lingkungan.

4. Biaya Pengoperasian

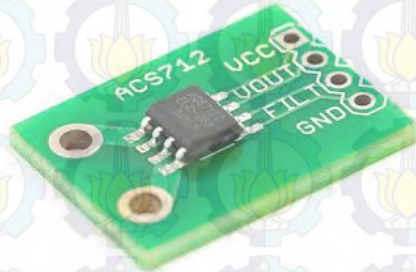
Sebagai indikatornya adalah angka susut jaringan, yaitu selisih antara energi yang dikeluarkan oleh pembangkit dengan energi yang digunakan oleh pelanggan. Penyebab susut jaringan antara lain yaitu pencurian listrik, kesalahan alat ukur, jaringan yang terlalu panjang, faktor daya rendah serta konfigurasi jaringan yang kurang tepat.

5. Kepuasan Pelanggan

2.3 Sensor Arus [3]

Pengukuran arus biasanya membutuhkan sebuah *resistor shunt* yaitu *resistor* yang dihubungkan secara seri pada beban dan mengubah aliran arus menjadi tegangan. Tegangan tersebut biasanya diumpankan ke *current transformer* terlebih dahulu sebelum masuk ke rangkaian pengkondisi *signal*. Teknologi *Hall effect* yang diterapkan oleh *Allegro* menggantikan fungsi *resistor shunt* dan *current transformer* menjadi sebuah sensor dengan ukuran yang relatif jauh lebih kecil. Aliran arus listrik yang mengakibatkan medan magnet yang menginduksi bagian *dynamic offset cancellation* dari *ACS712*. bagian ini akan dikuatkan oleh *amplifier* dan melalui *filter* sebelum dikeluarkan melalui kaki 6 dan 7,

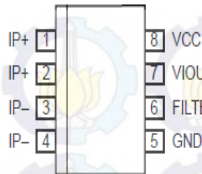
modul tersebut membantu penggunaan untuk mempermudah instalasi arus ini ke dalam sistem.



Gambar 2.3 Sensor Arus ACS712

Gambar 2.3 menjelaskan ACS712 adalah *Hall Effect current sensor*. *Hall effect allegro ACS712* merupakan sensor yang presisi sebagai sensor arus AC atau DC dalam pembacaan arus didalam dunia industri, otomotif, komersil dan sistem-sistem komunikasi. Pada umumnya aplikasi sensor ini biasanya digunakan untuk mengontrol motor, deteksi beban listrik, *switched-mode power supplies* dan proteksi beban berlebih. Sensor ini memiliki pembacaan dengan ketepatan yang tinggi, karena didalamnya terdapat rangkaian *low-offset linear Hall* dengan satu lintasan yang terbuat dari tembaga. cara kerja sensor ini adalah arus yang dibaca mengalir melalui kabel tembaga yang terdapat didalamnya yang menghasilkan medan magnet yang di tangkap oleh *integrated Hall IC* dan diubah menjadi tegangan proporsional. Ketelitian dalam pembacaan sensor dioptimalkan dengan cara pemasangan komponen yang ada didalamnya antara penghantar yang menghasilkan medan magnet dengan *hall transducer* secara berdekatan. Persisnya, tegangan proporsional yang rendah akan menstabilkan *Bi CMOS Hall IC* yang didalamnya yang telah dibuat untuk ketelitian yang tinggi oleh pabrik.

Pin-out Diagram

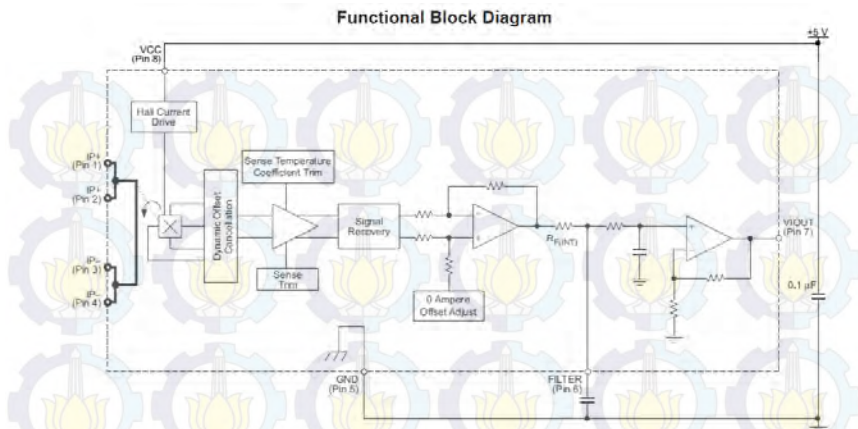


Terminal List Table

Number	Name	Description
1 and 2	IP+	Terminals for current being sensed; fused internally
3 and 4	IP-	Terminals for current being sensed; fused internally
5	GND	Signal ground terminal
6	FILTER	Terminal for external capacitor that sets bandwidth
7	VIOUT	Analog output signal
8	VCC	Device power supply terminal

Gambar 2.4 Contoh Skema ACS712

Output/keluaran dari sensor ini sebesar ($>VIOUT(Q)$) saat peningkatan arus pada penghantar arus (dari pin 1 dan pin 2 ke pin 3 dan 4), yang digunakan untuk pendeteksian atau perasa arus. Hambatan dalam penghantar sensor sebesar $1,2 \text{ m}\Omega$ dengan daya yang rendah. Jalur terminal konduktif secara kelistrikan diisolasi dari sensor mengarah (pin 5 sampai pin 8). Hal ini menjadikan sensor arus *ACS712* dapat digunakan pada aplikasi-aplikasi yang membutuhkan isolasi listrik tanpa menggunakan opto-isolator atau teknik isolasi lainnya yang mahal. Ketebalan penghantar arus didalam sensor sebesar 3x kondisi *overcurrent*. Sensor ini telah dikalibrasi oleh pabrik. Untuk lebih jelas dapat dilihat pada Gambar 1.3.



Gambar 2.5 Blok Diagram Sensor Arus ACS712

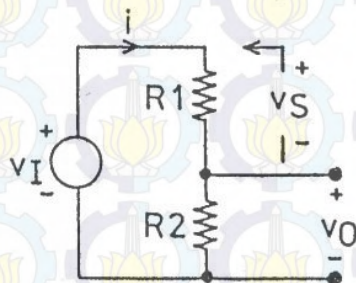
Beberapa fitur penting dari sensor arus ACS712 adalah:

1. Jalur sinyal *analog* yang rendah *noise*
2. *Bandwidth* perangkat diatur melalui pin *FILTER* yang baru
3. Waktu naik keluaran 5 mikrodetik dalam menanggapi langkah asukan aktif
4. *Bandwith* 50 kHz
5. Total error keluaran 1,5% pada $T_A = 25^\circ$, dan 4% pada -40° C sampai 85° C
6. Bentuk yang kecil, paket SOIC8 yang kompak.
7. *Resistansi* internal 1.2 m Ω .
8. 2.1 kVRMS tegangan *isolasi minimum* dari pin 1-4 ke pin 5-8
9. Operasi catu daya tunggal 5.0 V
10. Sensitivitas keluaran 66-185 mV/A
11. Tegangan keluaran sebanding dengan arus AC atau DC
12. Akurasi sudah diatur oleh pabrik
13. Tegangan *offset* yang sangat stabil
14. *Hysteresis magnetic* hampir mendekati nol
15. Keluaran *ratioetric* diambil dari sumber daya

2.4 Voltage Divider [4]

Rangkaian pembagi tegangan biasanya digunakan untuk membuat suatu tegangan referensi dari sumber tegangan yang lebih besar, titik tegangan referensi pada sensor, untuk memberikan bias pada rangkaian penguat atau untuk memberi bias pada komponen aktif. Rangkaian

pembagi tegangan pada dasarnya dapat dibuat dengan 2 buah resistor, contoh rangkaian dasar pembagi tegangan dengan output V_O dari tegangan sumber V_I menggunakan resistor pembagi tegangan R_1 dan R_2 seperti pada gambar 1.4.



Gambar 2.6 Rangkaian Voltage Divider

Gambar 2.6 dari rangkaian pembagi tegangan diatas dapat dirumuskan tegangan output V_O . Arus (I) mengalir pada R_1 dan R_2 sehingga nilai tegangan sumber V_I adalah penjumlahan V_S dan V_O sehingga dapat dirumuskan sebagai berikut.

$$V_I = V_S + V_O = i \cdot R_1 + i \cdot R_2$$

Dari rangkaian pembagi tegangan diatas dapat dirumuskan tegangan output V_O . Arus (I) mengalir pada R_1 dan R_2 sehingga nilai tegangan sumber V_I adalah penjumlahan V_S dan V_O sehingga dapat dirumuskan sebagai berikut.

$$V_O = V_I \cdot \left(\frac{R_2}{R_1 + R_2} \right)$$

Berdasarkan Hukum Ohm:

$$V = I \cdot R$$

Dan selanjutnya dikatakan bahwa nilai resistansi, R , tidak tergantung terhadap I atau V . Dengan demikian nilai resistansi, R , adalah bergantung terhadap nilai resistansi, R , yang diberikan. Jika dilihat dari rangkaian, nilai tegangan sumber, V , sudah ditetapkan. Dengan demikian, variabel yang berubah adalah besar arus, I . Sehingga hukum *Ohm* dituliskan menjadi:

$$I = V / R$$

Dan karena R_1 dan R_2 disusun secara seri, dan sistem di atas hanya terdiri atas satu *loop*. Maka nilai $R = R_1 + R_2$. Sehingga:

$$I = V / (R_1 + R_2)$$

Dengan demikian, nilai $VR1$ dapat dipenuhi dengan persamaan:

$$VR1 = I.R1$$

$$VR1 = [V/(R1+R2)].R1$$

$$VR1 = [R1/(R1+R2)].V$$

Dan dengan proses yang sama, $VR2$ dapat dipenuhi dengan persamaan:

$$VR2 = [R2/(R1+R2)].V$$

2.5 Intel Galileo [5]

Intel Galileo adalah papan pengembangan (*development board*) berbasis *Intel* pertama yang dirancang untuk diprogram dengan menggunakan perangkat lunak *Arduino – IDE*. *Galileo* memiliki pin yang *compatible* dengan *Arduino Uno R3* sehingga dapat beroperasi dengan *shield Arduino standar*. Mikroprosesor yang digunakan pada *Galileo* adalah mikroprosesor *Intel® Quark™ SoC X1000 application processor*, yang merupakan mikroprosesor 32 bit dengan kecepatan operasi 400 MHz. Beberapa fitur yang terdapat pada *Galileo* :



Gambar 2.7 Intel Galileo

1. Mendukung berbagai antar muka I/O standar industri, seperti *full-sized mini-PCI Express** slot, 100 Mb *Ethernet port*, *microSD** slot, USB host port, dan USB client port.
2. Memiliki media penyimpanan berupa 256 MB DDR3, 512 kb embedded SRAM, 8 MB *NOR Flash*, dan 8 kb *EEPROM standard on the board*, serta dapat dipasang *microSD card* maksimal 32 GB.
3. Memiliki susunan pin standar *Arduino (Digital, Analog, dan catu daya)* sehingga *compatible* dengan sebagian besar *shield Arduino*.

Galileo yang kami terima adalah merupakan generasi 2, yang memiliki beberapa perbedaan dibandingkan generasi pertama, antara lain :

1. *Header 6-pin 3.3V USB TTL UART* menggantikan *3.5 mm jack RS-232 console port* untuk akses menggunakan *OS Linux*.
2. *Konektor standar 6 pin kabel konverter serial USB ke TTL 3.3V*
3. Peningkatan kecepatan dan kemampuan *12 pin GPIOs*.
4. *PWM 12 bit* untuk pengendalian gerakan yang lebih halus pada motor *servo* dan *DC*
5. *UART 1* dapat diarahkan ke *header Arduino* sehingga menghilangkan kebutuhan *serial software*.
6. Memiliki kemampuan *12V power-over-Ethernet (PoE) capable*
7. *Catu daya* dapat menerima tegangan maksimal *15V*, dibandingkan *7V* pada generasi 1.

Apa yang dapat dibuat menggunakan *Galileo* ini? Secara sederhana *Galileo* ini adalah peningkatan kapasitas dan kemampuan dari *Arduino* biasa, sehingga semua hal yang dapat dilakukan *Arduino* juga dapat dilakukan oleh *Galileo*. Dengan kelebihan yang dimiliki, *Galileo* dapat digunakan untuk melakukan *komputasi* yang rumit seperti memproses citra dari kamera, menganalisis suara, ataupun memadukan informasi dari beberapa sensor. Semoga *Galileo* yang diterima ini dapat bermanfaat bagi pengembangan pendidikan dan teknologi di bidang *embeded system*. Teknologi *IoT (Internet of Thing)* mendorong *Intel* mengeluarkan teknologi yang menggabungkan komputer mini dengan *Arduino*. Teknologi tersebut diberi nama *Intel Galileo*. Berbeda dengan *Arduino* yang menggunakan *microcontroller* sebagai dasarnya, *Intel Galileo* menggunakan *Processor Intel Quark SoC X1000*. Prosesor *Quark X1000* sendiri adalah sebuah prosesor 32-bit berarsitektur yang sama dengan *Intel Pentium* dan memiliki *memori cache* tertanam sebesar *512KB*. Selain *hardware* yang kompatibel dengan *Arduino*, pemrograman untuk *Intel Galileo* dapat menggunakan pemrograman untuk *Arduino* yaitu *Arduino IDE (Integrated Development Environment)*. Selain dapat menggunakan bahasa pemrograman *C* dan *C#* (bahasa pemrograman yang didukung dalam *Arduino IDE*), dapat juga digunakan bahasa pemrograman lain yang di antaranya adalah *Phyton* dan *Node.js*.

alat yang berbasis *IoT*. *Intel Galileo* merupakan sistem yang mudah digunakan. Selain karena *kompatibel* dengan *Arduino*, *Intel Galileo* juga memiliki sifat yang *open source* dari sisi *hardware* maupun *software*. *Hardware* dan *software*-nya masih sangat mudah untuk dioprek. Selain kita dapat menggunakan *Arduino IDE* untuk melakukan pemrograman pada *Intel Galileo*, kelebihan dari *Intel Galileo* adalah kita dapat menggunakan *library* yang ada pada *Arduino*. Kelebihan lainnya adalah bagi yang terbiasa dengan pemrograman *web* dapat menggunakan *node.js* untuk pemrogramannya. Bahkan bahasa pemrograman *python* juga dapat kita gunakan. *Intel Galileo* merupakan sistem yang membutuhkan sedikit tegangan saja untuk beroperasi. Sistem ini hanya membutuhkan tegangan sebesar 5v DC sehingga power dari kabel *USB* komputer mampu untuk memberikan energi untuk menjalankan *Arduino*.

Intel Galileo dalam Tugas Akhir yang telah kami buat digunakan sebagai client untuk mengakuisi data dari sensor dan dikirimkan ke MQTT server (pubnub), sebelum dikirim ke komputer remote.

2.6 Javascript [6]

Javascript adalah bahasa *script* yang dikembangkan oleh *netscape* untuk membuat dokumen yang dinamis. *Java script* adalah bahasa *script* sederhana yang mempunyai kemiripan dengan bahasa pemrograman C. *JavaScript* juga dikenal sebagai sebuah kode pemrograman berorientasi objek (*objek Oriented Programming*) disingkat OOP. *javascript* memiliki keistimewaan untuk ditambahkan pada kode *HTML* dan membuat dokumen menjadi lebih interaktif. *Javascript* adalah sebuah bahasa komputer atau kode pemrograman yang digunakan pada *website* agar *website* tersebut menjadi lebih interaktif dan dinamis. *Javascript* adalah jenis bahasa pemrograman *client side*. Penggunaan kode *javascript* pada sebuah *website* bersifat *opsional*, artinya tidak harus selalu ada. Namun, *website-website* maupun *blog modern* saat ini hampir semuanya menggunakan kode *javascript* walaupun sedikit. *javascript* bukan *Java*, meskipun antara keduanya ada persamaan. Bahasa *JavaScript* menyurupai *Java* tetapi tidak memiliki penulisan yang statis dan kontrol yang kuat. *javascript* mendukung banyak *syntax ekspresi* dan konstruksi alur kendali dasar *Java*. Manfaat, Fungsi & Peran *JavaScript* Seperti yang disebutkan diatas, *javascript* berfungsi membuat sebuah halaman *website* lebih *interaktif* dan *dinamis*. Penggunaan kode *javascript* sangat dibutuhkan untuk *website-website*

yang *berorientasi* pada kenyamanan pengakses (*user-experience*). Namun, penggunaan kode *javascript* ini tidak akan maksimal jika tidak dibarengi dengan penggunaan *css* karena *javascript* juga tidak akan lepas dari *html*. Ketiga bahasa pemrograman itu sangat erat kaitannya. Jika saya ibaratkan sebuah *website* adalah rumah, maka kode *html* adalah segala yang membangun dan ada dirumah tersebut (tembok, tiang, atap, pondasi, ruang keluarga, kamar, dapur, peralatan rumah tangga, dan sebagainya). Lalu, kode *css* adalah pengatur dan penghias rumah tersebut(tata letak, warna dan sebagainya). Dan peran *javascript* bisa sebagai ‘pembantu’ kode *css* agar pengaturan dan hiasannya lebih bagus dengan tambahan pernik-pernik misalnya. Bisa juga untuk memanipulasi kode *html* dengan nama/identitas tertentu. Manipulasi ini bisa berupa menyembunyikan, mengganti bahkan menambahkan *elemen html* yang baru.

2.7 Bootstrap [7]

Bootstrap merupakan *framework* yang mampu mengatasi kesulitan maupun permasalahan yang ada saat mendesign *website*. *Bootstrap* dengan segala kelengkapannya yang dibutuhkan pada masalah design baik itu menggunakan *css* maupun *javascript*, sudah disajikan begitu mudah, menarik dan tentunya mampu menyelesaikan design secara cepat. Tampilan yang juga responsive pada banyak *platform*, tetap rapi baik pada tampilan versi desktop maupun pada versi mobile. Salah satu contoh *website* besar yang menggunakan *framework bootstrap* adalah *Twitter*, *interface twitter* di bangun dengan menggunakan *bootstrap*. karena sebenarnya *bootstrap* di kembangkan oleh *developer twitter* sendiri sehingga *bootstrap* sering juga di sebut “*twitter bootstrap*”. *Bootstrap* berisi kumpulan class yang siap pakai. misalnya anda ingin membuat sebuah tombol dengan cepat dan tana harus merancang dan mengetikkan *syntax css* untuk membuat sebuah tombol, maka dengan *bootstrap* anda dapat membuatnya dengan sangat cepat, yaitu hanya dengan menambahkan class “*btn*” dan menambahkan class *btn*. Berikut ini akan di jelaskan beberapa kegunaan yang telah menjadi kelebihan pada *bootstrap*. adapun beberapa kelebihan *bootstrap* adalah sebagai berikut.

2.8 Pubnub

PubNub adalah jasa global *Data Stream Network* (*DSN*) dan *realtime infrastruktur - as-a -service (IaaS)* perusahaan yang berbasis di

San Francisco , California . perusahaan tersebut membuat produk untuk pengembang perangkat lunak dan perangkat keras untuk membangun *web realtime*, *mobile*, dan *Internet of Things* (IOT) aplikasi . produk utama *PubNub* adalah *realtime* mempublikasikan / berlangganan *messaging* API dibangun pada jaringan aliran data global mereka yang terdiri dari jaringan direplikasi setidaknya 14 pusat data yang terletak di Amerika Utara , Amerika Selatan , Eropa , dan Asia . jaringan saat ini melayani lebih dari 300 juta perangkat dan sungai lebih dari 750 miliar pesan per bulan. Cara penggunaan *pubnub* adalah dengan mendaftarkan akun *email* yang resmi dan sudah terverifikasi dengan *email* yang sudah terdaftar, selanjutnya *email* dapat digunakan untuk *login* ke *dashboard* dari *pubnub*. Penggunaan dengan cara membuat aplikasi *pubnub*. Didalam aplikasi *pubnub* tersebut terdapat *subscription key* dan *publish key* yang dapat digunakan untuk berlangganan *pubnub*. Didalam *pubnub*, terdapat *channel* yang berfungsi sebagai saluran utama untuk menerima dan mengirimkan data.

2.9 Jade (java agent development framemwork)

JADE (Java Agent Development Framework) adalah *middleware* yang dapat digunakan untuk mengembangkan dan menjalankan aplikasi *peer to peer* yang berdasarkan pada paradigma agent. Sesuai namanya bahasa pemrograman yang digunakan untuk mengembangkan agent dalam *JADE* adalah *Java*. Prinsip Kerja *JADE*. *Jade* sebagai *middleware* yang memberikan fasilitas untuk pengembangan sistem berbasis agent menyediakan :

1. Runtime environment yang menjadi tempat di mana agent dapat berjalan dan harus aktif pada host dimana agent akan bekerja.
2. Pustaka berupa kelas-kelas yang dapat/harus digunakan untuk mengembangkan agent.
3. Sekumpulan graphic tool yang digunakan untuk melakukan administrasi dan monitoring terhadap aktivitas agent yang sedang berjalan pada runtime environment.

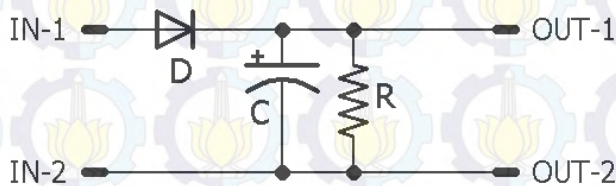
Runtime environement dalam *JADE* dikenal dengan istilah container. Satu host dapat menjalankan lebih dari satu container dan setiap container bisa menangani beberapa agent. Sekumpulan container yang aktif disebut sebagai platform. Sebuah platform dapat memiliki container yang berasal dari host yang berbeda-beda. Satu platform harus memiliki satu container yang memiliki atribut sebagai main container

yang aktif. Semua container yang aktif dan ingin bergabung dalam sebuah platform harus bergabung dengan mendaftarkan diri pada main container dan tidak boleh beratribut sebagai main container atau disebut juga normal container. Selain menerima pendaftaran dari normal container sebuah main container selalu memiliki dua buah agent yang aktif secara otomatis ketika main container dijalankan. Kedua agent itu adalah:

1. *AMS (Agent Management System)* yang menyediakan naming service yang memastikan setiap *agent* dalam *platform* memiliki identitas yang unik. Selain itu AMS dapat merepresentasikan otoritas dalam platform di mana melalui AMS kita dapat menjalankan atau menghentikan *agent* dalam container yang terdaftar.
2. *DF (Directory Facilitator)* adalah *agent* yang berfungsi sebagai “yellow pages” bagi *platform*. Melalui DF sebuah *agent* dapat mencari *agent* yang aktif dan layanan yang diberikan *agent* tersebut.

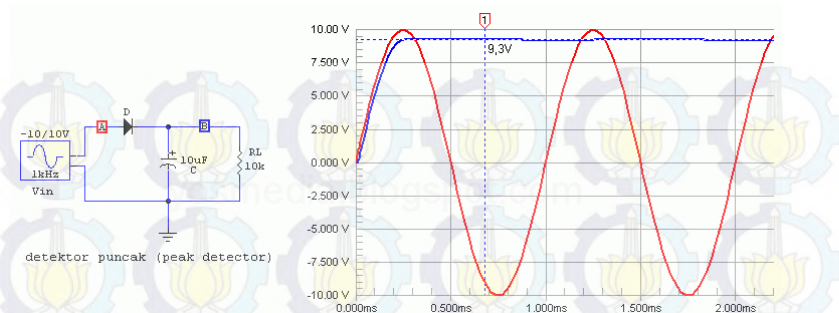
2.10 Peak Detector

peak detector adalah rangkaian yang terdiri dari hubungan seri sebuah dioda dengan kapasitor yang menghasilkan output, secara teori, berupa tegangan DC yang sama dengan amplitudo puncak (V_p) tegangan AC sebagai *input*. Tetapi karena dioda yang ada tidaklah ideal maka tegangan *output* DC yang dihasilkan adalah hasil pengurangan dari amplitudo puncak tegangan AC sebagai *input* dengan tegangan buka dioda sebesar 0,7V.



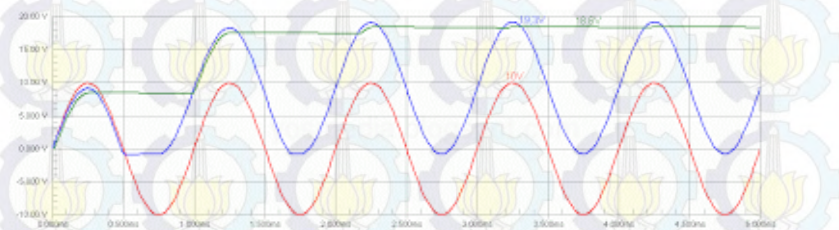
Gambar 2.9 Rangkaian Peak Detector

Gambar 2.9 menjelaskan tentang rangkaian *peak detektor* yang menggunakan *input* sensor arus ACS712 untuk mendeteksi *peak* pada sensor arus.



Gambar 2.10 Peak Detector

Sama seperti rangkaian detektor puncak, *output* dari rangkaian detektor puncak ke puncak ini juga dipengaruhi oleh tegangan buka diode seperti pada gambar 2.9. Hal ini dapat ditunjukkan pada grafik *output* rangkaian detektor puncak ke puncak berikut.



Gambar 2.11 Gelombang Peak Detector

Untuk rangkaian penjepit DC positif dapat kita ketahui *output*, yaitu : $V_{out\text{clamper}} = V_{in(\text{peak-to-peak})} - V_D = 20 - 0,7 = 19,3V$

Output rangkaian penjepit DC positif ini kemudian dijadikan sebagai *input* rangkaian detektor puncak, yang juga merupakan *output* rangkaian detektor puncak ke puncak, yang besarnya adalah :

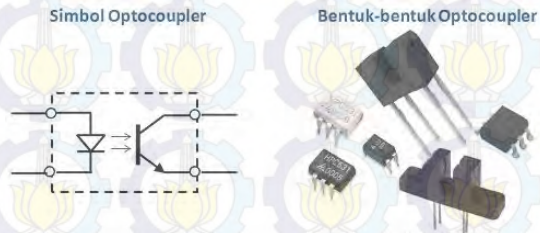
$$V_{out(\text{peak-to-peak detector})} = V_{in(\text{peak-to-peak})} - V_D$$

$$V_{out(\text{peak-to-peak detector})} = V_{out(\text{clamper})} - V_D = 19,3 - 0,7 = 18,6V$$

2.11 Optocoupler [8]

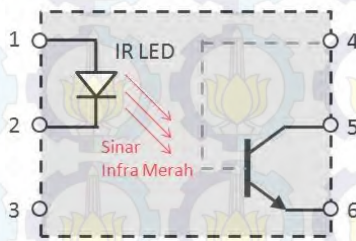
Optocoupler adalah komponen elektronika yang berfungsi sebagai penghubung berdasarkan cahaya optik. Pada dasarnya *Optocoupler* terdiri dari 2 bagian utama yaitu *Transmitter* yang berfungsi sebagai pengirim cahaya optik dan *Receiver* yang berfungsi sebagai pendeteksi sumber cahaya. Masing-masing bagian *Optocoupler* (*Transmitter* dan

Receiver) tidak memiliki hubungan konduktif rangkaian secara langsung tetapi dibuat sedemikian rupa dalam satu kemasan komponen.



Gambar 2.12 Simbol Dan Bentuk Optocoupler

Pada gambar 2.11 menjelaskan jenis-jenis *Optocoupler* yang sering ditemukan adalah *Optocoupler* yang terbuat dari bahan Semikonduktor dan terdiri dari kombinasi *LED (Light Emitting Diode)* dan *Phototransistor*. Dalam Kombinasi ini, *LED* berfungsi sebagai pengirim sinyal cahaya optik (*Transmitter*) sedangkan *Phototransistor* berfungsi sebagai penerima cahaya tersebut (*Receiver*). Jenis-jenis lain dari *Optocoupler* diantaranya adalah kombinasi *LED-Photodiode*, *LED-LASCR* dan juga *Lamp-Photoresistor*. Prinsip Kerja *Optocoupler*. Pada prinsipnya, *Optocoupler* dengan kombinasi *LED-Phototransistor* adalah *Optocoupler* yang terdiri dari sebuah komponen *LED (Light Emitting Diode)* yang memancarkan cahaya infra merah (*IR LED*) dan sebuah komponen semikonduktor yang peka terhadap cahaya (*Phototransistor*) sebagai bagian yang digunakan untuk mendeteksi cahaya infra merah yang dipancarkan oleh *IR LED*. Untuk lebih jelas mengenai Prinsip kerja *Optocoupler*, silakan lihat rangkaian internal komponen *Optocoupler* pada gambar 2.12 :



Gambar 2.13 Rangkaian Optocoupler

Dari gambar 2.12 dapat dijelaskan bahwa Arus listrik yang mengalir melalui *IR LED* akan menyebabkan *IR LED* memancarkan

sinyal cahaya Infra merahnya. Intensitas Cahaya tergantung pada jumlah arus listrik yang mengalir pada *IR LED* tersebut. Kelebihan Cahaya Infra Merah adalah pada ketahanannya yang lebih baik jika dibandingkan dengan Cahaya yang tampak. Cahaya Infra Merah tidak dapat dilihat dengan mata telanjang. Cahaya Infra Merah yang dipancarkan tersebut akan dideteksi oleh *Phototransistor* dan menyebabkan terjadinya hubungan atau *Switch ON* pada *Phototransistor*. Prinsip kerja *Phototransistor* hampir sama dengan Transistor Bipolar biasa, yang membedakan adalah Terminal Basis (Base) *Phototransistor* merupakan penerima yang peka terhadap cahaya.

2.12 MQTT (Message Queuing Telemetry Transport)

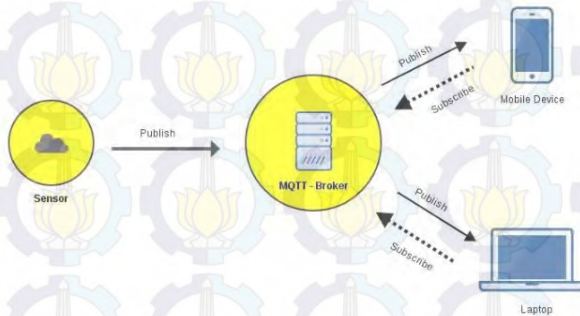
MQTT (*Message Queuing Telemetry Transport*) adalah protokol konektivitas m2m (*Machine To Machine*) atau *IoT* berbasis *open source* dengan *standard* terbuka dirancang untuk perangkat terbatas dengan *bandwidth* rendah, *latency* tinggi, dan jaringan buruk. MQTT ideal untuk perangkat m2m (*Machine To Machine*) dimana *bandwidth* dan daya tahan baterai menjadi pertimbangan utama. Ada beberapa pertimbangan mengapa transfer data di sistem *IoT* harus seefisien mungkin. Seperti anda ketahui bahwa pada sistem *IoT* terutama pada bagian perangkat akuisisi data seperti sensor & *embedded device* biasanya memakai catu daya yang disediakan oleh baterai, solar panel dll. Koneksi perangkat ini dengan server atau internet biasanya memakai koneksi WiFi, Bluetooth, GSM, gelombang radio dll yang memerlukan catu daya cukup besar saat koneksi terjadi apalagi kalau koneksi terjadi tiap waktu, batasan lainnya yaitu ukuran media penyimpanan internal dan RAM dari perangkat-perangkat ini biasanya relatif cukup kecil. Jadi ada beberapa optimasi yang harus diperhatikan dalam pertukaran data pada sistem *IoT* agar transfer data bisa seefisien mungkin

1. Menekan ukuran paket data sekecil mungkin sehingga trafik bisa meningkat.
2. Meminimalisasi proses komputasi untuk *encoding* dan *decoding* dari paket data.
3. Data hanya menggunakan ruang penyimpanan yang sekecil mungkin.

Optimasi-optimasi diatas disediakan oleh protokol yang bernama MQTT.

Protokol MQTT (*Message Queuing Telemetry Transport*) adalah protokol yang berjalan pada diatas stack TCP/IP dan mempunyai ukuran

paket data dengan *low overhead* yang kecil (minimum 2 bytes) sehingga berefek pada konsumsi catu daya yang juga cukup kecil. Protokol ini adalah jenis protokol *data-agnostic* yang artinya anda bisa mengirimkan data apapun seperti data binary, text bahkan XML ataupun JSON dan protokol ini memakai model *publish/subscribe* daripada model *client-server*.



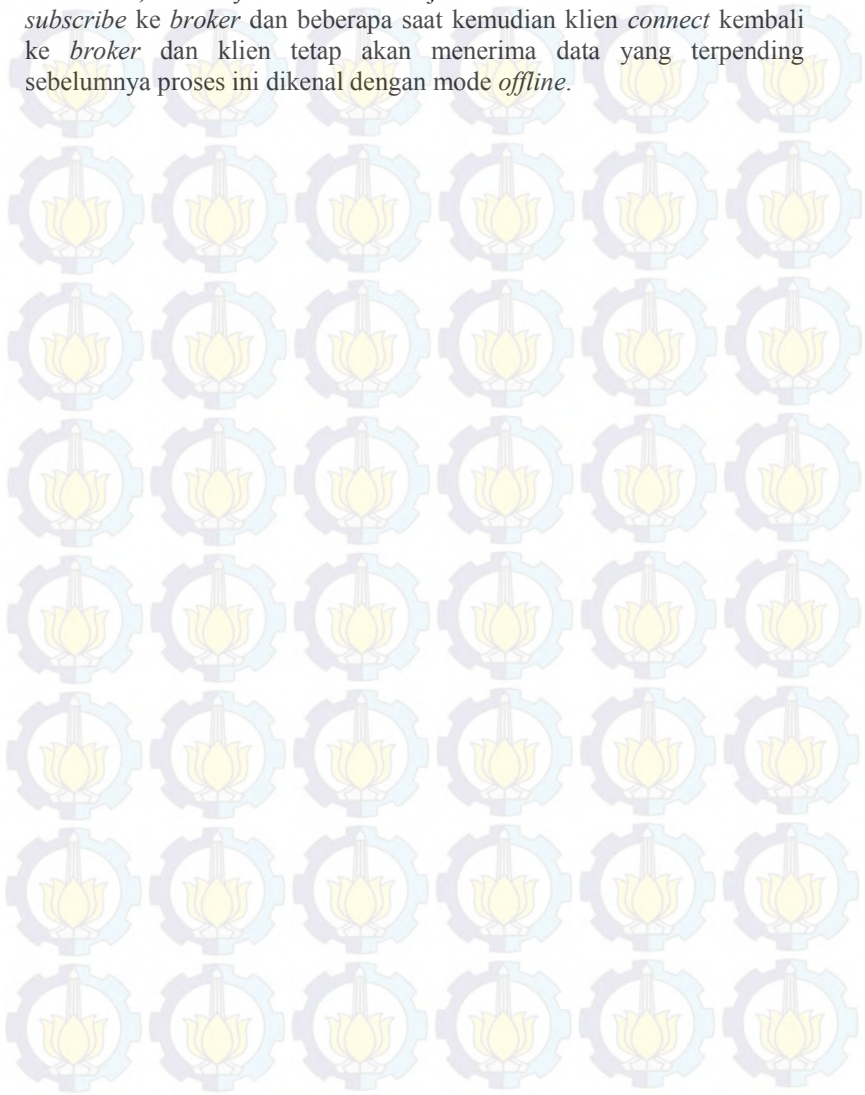
Gambar 2.14 Sistem Umum *IoT* Memakai *MQTT*

Pada gambar 2.13 menjelaskan tentang, *Stack TCP/IP* sudah banyak di dukung oleh mikrokontroler seperti seri STM32Fx7 maupun device board yang umum dipasaran seperti *Arduino Yun*, *Arduino + Ethernet Shield*, *ESP8266 WiFi SoC*, *Raspberry Pi*, *Intel Galileo* dll. Jadi sebenarnya banyak pilihan untuk belajar *IoT* dengan memakai protokol ini ditambah lagi harga device yang semakin murah dan terjangkau. Sistem umum *MQTT* seperti pada gambar diatas membutuhkan dua komponent perangkat lunak utama yaitu

1. *MQTT Client* yang nantinya akan di install di device. Untuk Intel Galileo bisa memakai *pubsubclient*, pustaka seperti *mqtt.js* bisa dipakai pada platform *Node.js* di *Raspberry Pi* ataupun laptop.
2. *MQTT Broker* yang berfungsi untuk menangani publish dan subscribe data. Untuk platform *Node.js* bisa memakai *broker mosca* sedangkan untuk platform yg lain banyak broker tersedia seperti *mosquitto*, *HiveMQ* dll.

Keuntungan dari sistem *publish/subscribe* adalah antara sumber pengirim data (*publisher*) dan penerima data (klien) tidak saling mengetahui karena ada *broker* diantara mereka atau istilah kerennya yaitu *space decoupling* dan yang lebih penting lagi yaitu adanya *time*

decoupling dimana *publisher* dan klien tidak perlu terkoneksi secara bersamaan, misalnya klien bisa saja *disconnect* setelah melakukan *subscribe* ke *broker* dan beberapa saat kemudian klien *connect* kembali ke *broker* dan klien tetap akan menerima data yang terpending sebelumnya proses ini dikenal dengan mode *offline*.



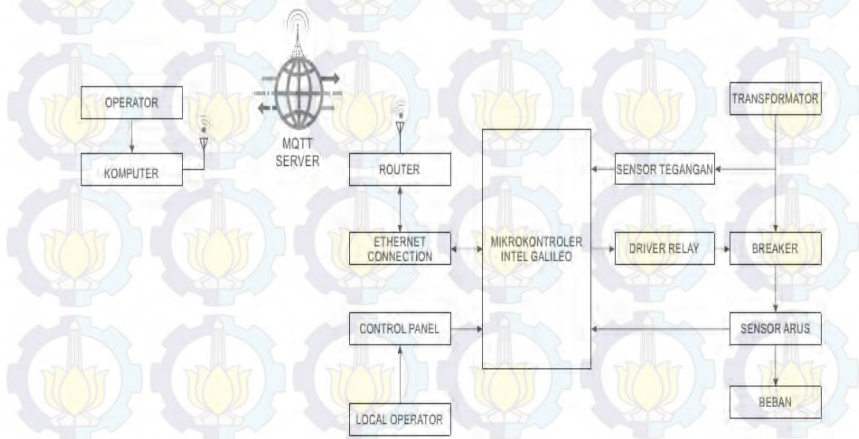
BAB III

PERANCANGAN SISTEM KONTROL

Bab ini membahas tentang sistem proses kerja dari Tugas Akhir dalam bentuk diagram, perancangan perangkat keras dan perancangan perangkat lunak. Perancangan perangkat keras dibagi menjadi perancangan elektronik dan juga perancangan mekanik. Perancangan elektronik meliputi diagram *input-output* sistem *intel galileo*, perancangan sensor arus, perancangan pengondisian sinyal sensor arus, perancangan sensor tegangan, perancangan power *supply* untuk pengondisi sensor arus, dan *driver relay*. Untuk perancangan mekanik, perancangan blok *circuit breaker*, dan perancangan box panel yang mendukung kerja alat. Untuk perancangan perangkat lunak perancangan pemrograman *Intel Galileo* menggunakan bahasa *JavaScript*, perancangan pemrograman analog *input intel galileo* yang terhubung dengan pengondisi sinyal dan sensor tegangan.

3.1 Blok Fungsional Sistem

Pada bab ini dibahas mengenai perancangan alat secara keseluruhan. Dijelaskan pula lebih terinci tiap-tiap bagian atau blok-blok penyusun alat ini berupa blok *funksional* sistem.



Gambar 3.1 Blok Fungsional

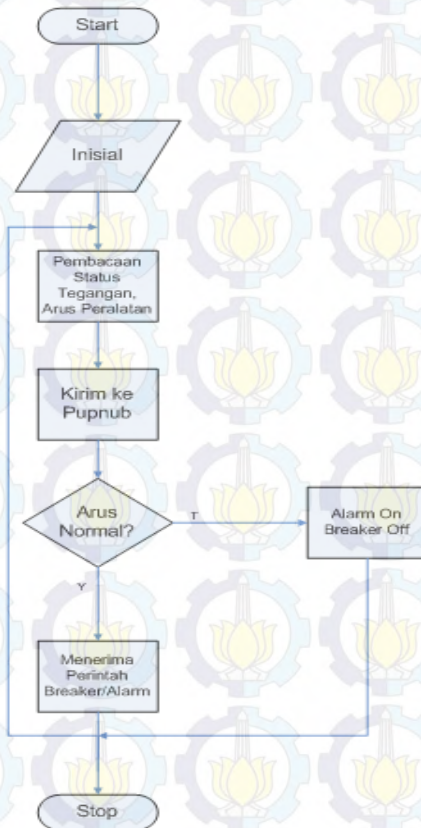
Gambar 3.1 merupakan blok fungsional untuk perancangan Monitoring gardu distribusi listrik PLN. Diagram blok tersebut memiliki tahapan sebagai berikut :

1. Operator memantau kondisi jaringan di lapangan pada tampilan. Sesuai kondisi, operator dapat mengirimkan perintah untuk dikondisikan pada peralatan. Perintah yang dapat dikirimkan berupa sinyal untuk menghubungkan atau memutuskan breaker serta menyalakan atau mematikan alarm peringatan.
2. Data dari sensor dan kondisi sistem yang ada di lapangan akan di transfer ke komputer operator melalui jaringan ethernet yang telah disediakan oleh *ICON+* dan terhubung ke jaringan Internet.
3. Data hasil akuisisi pada *intel galileo* akan dikirimkan ke *MQTT* server secara streaming sebelum ditampilkan pada tampilan *visual* pada komputer master. Pada *mqtt server*, semua data akan terekam dan terdapat log dari keseluruhan dari keseluruhan data yang dikirim.
4. Keseluruhan komunikasi antara perangkat di lapangan dengan komputer pada operator dilaksanakan melalui layanan *MQTT streaming message* yaitu *Pubnub*.
5. Pada Box, terdapat panel yang digunakan untuk operasi operator lapangan. Panel ini sebagai backup atau kontrol lokal apabila terdapat masalah koneksi antara *remote* operator dengan peralatan.
6. Pada peralatan ini, terbagi menjadi 2 bagian yaitu elektronik dan mekanik. Bagian elektronik pada peralatan ini terdiri dari CPU berupa *Intel Galileo*, sensor arus *ACS712*, sensor tegangan, *driver relay*, *push button local* operator, *led* indikator, dan rangkaian *circuit breaker*. Mekanik pada peralatan ini, menggunakan *relay/kontaktor* sebagai pemutus dan penghubung antara sumber tegangan(*transformator*) dengan beban. Semua komponen tersebut, disusun pada papan akrilik dan dicover oleh *box* panel yang terbuat dari besi.
7. Ketika CPU menerima perintah dari *remote* operator, maka peralatan akan mengeksekusi sesuai perintah yang dikirimkan secara *remote* baik untuk mematikan atau menyalakan alarm maupun memutus atau menyambungkan *circuit breaker*.

8. Pada panel operator *lokal*, terdapat saklar untuk memfungsikan panel operator lokal dan saklar untuk memutus dan menyambungkan *circuit breaker* secara lokal. Selain saklar, terdapat indikator yang digunakan untuk memantau kondisi dari *circuit breaker* dan status fungsi panel.

3.2 Perancangan Elektronik

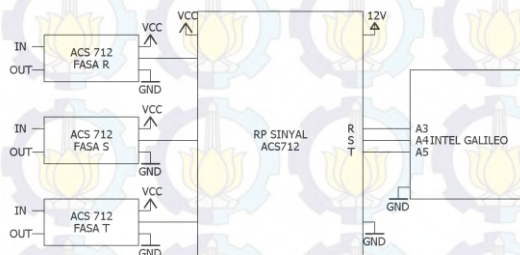
Adapun *flowchart* untuk alur cara kerja dan beberapa perancangan elektronik yang digunakan pada proyek Tugas Akhir ini, sebagai berikut perancangan elektronik:



Gambar 3.2 Flowchart Pada Hardware

3.2.1 Perancangan Pengondisi Sinyal Sensor Arus ACS712

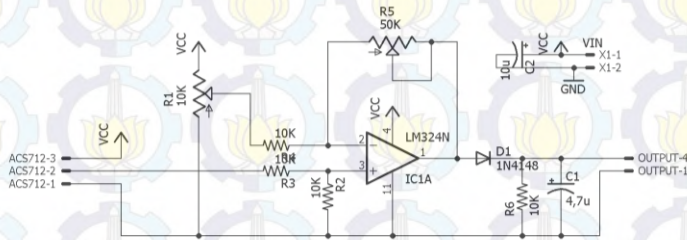
Sensor Arus ACS712 adalah komponen yang digunakan untuk mengukur besar arus yang mengalir pada satu rangkaian dengan memanfaatkan prinsip *Hall-Effect*. *Hall-Effect* sensor bekerja berdasarkan medan magnet yang terjadi akibat adanya aliran arus. Oleh sensor, medan magnet tersebut dikonversi menjadi tegangan. Berikut gambar hubungan sensor arus ACS712 dengan *Intel Galileo* :



Gambar 3.3 Diagram Koneksi Sensor Arus ACS712

Pada Gambar 3.3 terdapat blok diagram antara sensor ACS712, rangkaian pengondisi sinyal, dan *Intel Galileo*. Bentuk gelombang *output* dari sensor ACS712 tergantung dari jenis arus yang diukur. ACS712 pada alat ini digunakan untuk mengukur AC yang memiliki sifat arus yang selalu berubah arah (bolak-balik) membentuk gelombang sinusoidal sehingga *output* dari ACS712 akan ikut berbentuk sinusoidal dengan referensi apabila arus yang diukur oleh *hall-effect* bernilai 0 maka *output* dari ACS akan bernilai $VCC/2$. Dalam rangkaian ini, ACS712 disuplai tegangan dari rangkaian pengondisi sinyal sebesar 5Volt. Maka, apabila sensor tidak dialiri arus maka tegangan *output* dari sensor akan bernilai $\pm 2,5$ Volt dan berubah naik turun sesuai dengan arus inputnya yang berupa arus bolak-balik. ACS712 memiliki berbagai macam kapasitas arus, setiap perbedaan kapasitas memiliki resolusi perubahan tegangan yang berbeda dalam satuan mV/A. Kapasitas ACS yang dipakai pada peralatan ini memiliki batas maksimal sebesar 30 Ampere dengan perubahan setiap amperenya sebesar 66mV (66mV/A). Karena nilai dari *output* ACS yang berbentuk sinusoidal dengan nilai rata-rata sebesar 2,5Volt, maka dibutuhkan rangkaian tambahan berupa differential amplifier dan *peak detector* yang berfungsi menggeser nilai rata-rata *output* ACS712 dari $VCC/2$ menjadi 0 dan mendeteksi titik

puncak dari tegangan yang dihasilkan oleh ACS712 sehingga pembacaan ADC pada mikrokontroller menjadi lebih stabil. Rangkaian dasar dari sistem pengondisi sinyal digambarkan pada skema gambar 3.4.

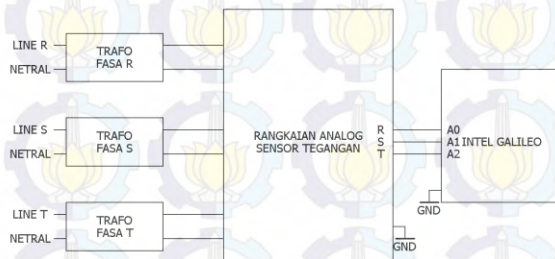


Gambar 3.4 Skematik Pengondisi Sinyal Output ACS712.

Dari Gambar 3.4 ada beberapa komponen penyusun dari sistem pengondisi sinyal ACS712 diantaranya *Op-Amp* dan rangkaian *peak detector*. *Op-Amp* pada rangkaian ini disusun menggunakan prinsip differential Amplifier, dimana cara kerja dari rangkaian *Op-Amp* ini dengan menghitung perbedaan tegangan antara kedua kaki inputnya. Sinyal *input* pada kaki *Non-Inverting* dikurangi dengan sinyal *input* pada kaki *Inverting*. Kaki *Non Inverting* pada *Op-Amp* mendapatkan sinyal *output* dari ACS712, sedangkan kaki *inverting*nya dihubungkan dengan multiturn sebagai pengatur tegangan referensinya. *Output* tegangan dari *Op-Amp* akan bernilai sama dengan nilai hasil dari pengurangan antara kedua inputnya. Hasil dari proses pengurangan oleh *Op-Amp* akan diumpankan menuju rangkaian *peak detector* yang terdiri dari kapasitor dan resistor. Pada rangkaian ini, tegangan output dari *Op-Amp* yang berupa setengah gelombang sinusoidal akan akan mencharge kapasitor pada saat nilai tegangan gelombang bertambah. Pada saat nilai gelombang turun, maka kapasitor yang masih menyimpan tegangan akan melakukan *discharge* oleh resistor. Waktu *discharge* kapasitor ini ditentukan oleh kapasitas kapasitor dan besar nilai resistor. *Output* dari rangkaian ini berupa tegangan searah dengan *minimum ripple*. Pada ujung rangkaian, terdapat dioda zener yang berfungsi sebagai *filter* tegangan agar nilai outputnya tidak melebihi 5,1 Volt. Sehingga *output* dari rangkaian aman sebelum dimasukkan menuju kaki *input* ADC pada mikrokontroler.

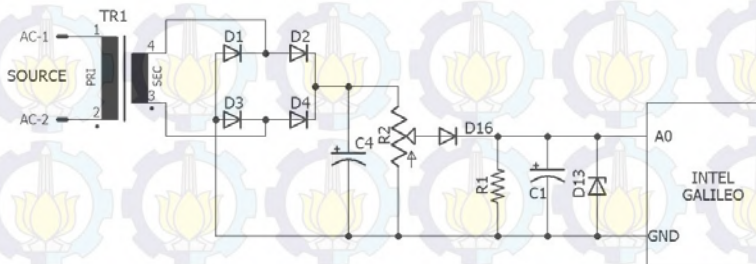
3.2.2 Perancangan Sensor Tegangan

Untuk sensor tegangan digunakan transformator, hal ini dimaksudkan agar keluaran dari trafo stabil dengan keluarannya sehingga nantinya sinyal yang masuk dapat dibaca oleh ADC. Sensor tegangan berfungsi sebagai pendeteksi adanya gangguan pada jaringan yang berupa pada tegangan listrik tersebut dengan begitu gangguan yang terjadi akan di baca oleh *Intel Galileo*.



Gambar 3.5 Diagram Analog Sensor Tegangan

Pada gambar 3.5 terdapat blok diagram antara sensor tegangan dan *intel Galileo*. Menjelaskan tentang rangkaian sensor tegangan yang terdiri dari 3 yang berfungsi sebagai penurun tegangan yang dikendalikan didalam rangkaian *analog* sensor tegangan selanjutnya dibaca oleh *Intel Galileo*, untuk dijalankan sebagai mana mestinya.



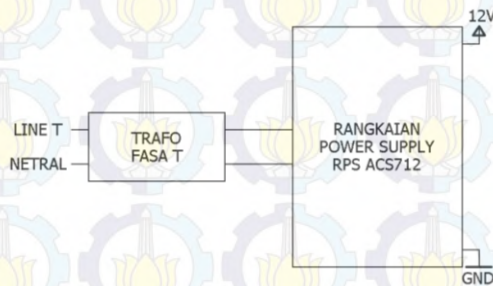
Gambar 3.6 Skematik Sensor Tegangan

Pada Gambar 3.6 ada beberapa komponen penyusun dari sistem pengondisi tegangan diantaranya trafo dan rangkaian sensor tegangan, trafo pada rangkaian tegangan berfungsi sebagai penurun tegangan yang berlebih. Pada gambar 3.6 menggunakan trafo dengan rasio primer dan

sekunder sebesar 220 banding 6 volt, trafo yang digunakan sebesar 100 mA.

3.2.3 Perancangan Power Supply

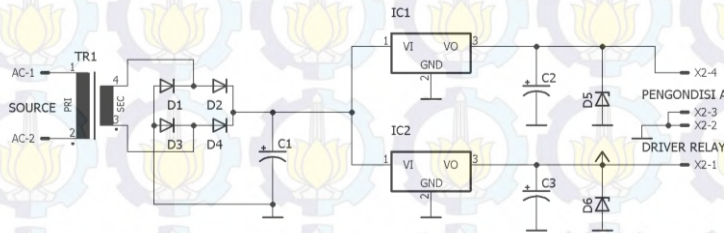
Digunakan untuk memberi masukan tegangan pada keseluruhan komponen yang terdapat pada sistem alat ini. Rangkaian *Power Supply* ini berguna sebagai sumber daya listrik untuk memberikan sumber pada perangkat lainnya, rangkaian *Power supply* pada gambar 3.7:



Gambar 3.7 Diagram Power Supply

Pada gambar 3.7 menjelaskan tentang *power supply* yang merupakan sebuah disain *power supply* dengan efisiensi daya yang baik. Saat ini peralatan elektronika yang menggunakan adaptor semakin banyak dan semakin beraneka ragam. Mulai dari peralatan elektronika yang murah seperti radio sampai dengan handphone. Kebutuhan adaptor sebagai sebuah alternatif sebagai pengganti baterai lebih disukai karena baterai tidak dapat tahan lama dan secara otomatis membuat biaya operasional sebuah alat elektronika tersebut menjadi lebih besar. Dengan sebuah adaptor tidak lagi dibutuhkan baterai tetapi kelemahannya tidak dapat dibawa-bawa dengan mudah karena adaptor harus selalu tersambung ke jaringan listrik PLN. Tetapi walaupun demikian adaptor tetap digunakan. Dari berbagai macam adaptor yang terdapat dipasaran, adaptor konvensional dengan transformator penurun tegangan serta regulator tegangan sederhana lebih banyak ditemukan daripada adaptor dengan teknologi *switching*. Power supply yang baik harus mampu memberikan tegangan regulasi yang baik serta mampu memberikan arus yang cukup kepada beban. Tegangan yang tidak terregulasi pada *output power supply* dapat menyebabkan peralatan elektronika yang menggunakan *power supply* tersebut akan rusak

terutama bagian regulasi tegangan (jika ada) tetapi jika peralatan tersebut tidak membunyai rangkaian regulasi tegangan internal maka dapat dipastikan peralatan elektronik tersebut akan rusak.

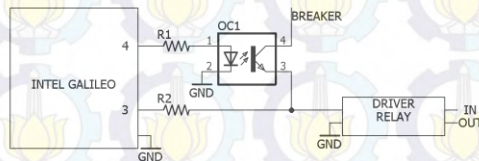


Gambar 3.8 Skematik Rangkaian Power Supply

Rangkaian *power supply* ini sangat penting karena rangkaian tersebut yang nantinya akan memberikan tegangan pada perangkat elektronik lainnya sebagai mana fungsinya rangkaian *power supply* adalah rangkaian yang memberikan sumber untuk perangkat elektronik yang terdapat pada alat ini, jadi *power supply* ini memberikan sumber untuk menjalankan alat ini.

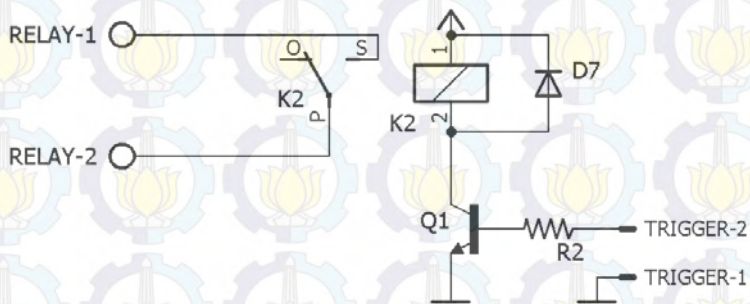
3.2.4 Perancangan Breaker

Pada dasarnya rangkaian breaker berfungsi sebagai pengaman pada perangkat listrik yang apabila terjadi hubung singkat ataupun ada arus berlebih, maka rangkaian breaker yang akan memutus aliran listrik secara otomatis bila terjadi gangguan arus berlebih ataupun hubung singkat, memberikan perlindungan terhadap lonjakan tegangan yang dikhawatirkan bisa menjadi penyebab kerusakan pada sirkuit penyearah atau bisa disebut sebagai pengaman arus (*overload*). Rangkaian Breaker pada gambar 3.10.



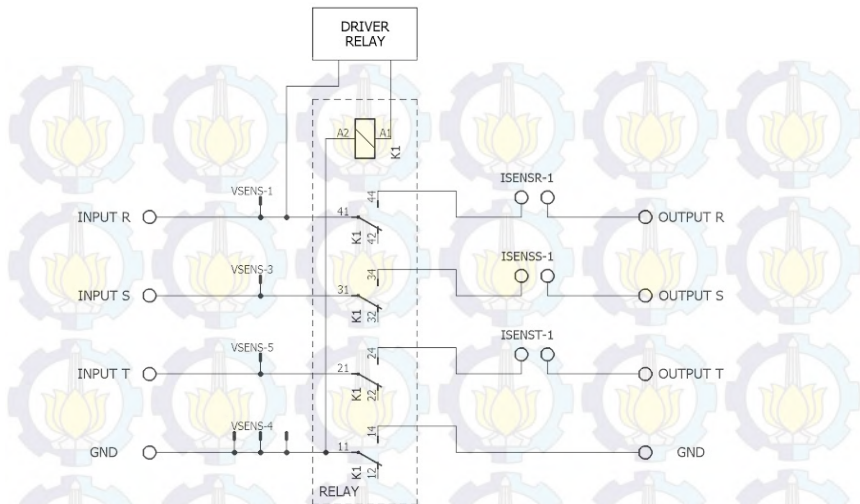
Gambar 3.9 Rangkaian Breaker

Rangkaian *Breaker* atau Sakelar Pemutus Tenaga adalah suatu peralatan pemutus rangkaian listrik pada suatu sistem tenaga listrik, yang mampu untuk membuka dan menutup rangkaian listrik pada semua kondisi, termasuk arus hubung singkat, sesuai dengan ratingnya. Juga pada kondisi tegangan yang normal ataupun tidak normal., dan berikut skematik *driver breaker* pada gambar 3.11.



Gambar 3.10 Skematik Driver Breaker

Pada gambar 3.11 menjelaskan tentang skematik driver breaker, terdapat bagian-bagian yang memiliki fungsi yang berbeda-beda untuk menjalankan breaker dengan sebagai mana mestinya, driver breaker berfungsi untuk menjalankan rangkaian breaker agar bekerja sebagai mana fungsinya yang dapat memutus dan menyambungkan arus listrik pada jaringan distribusi PLN.

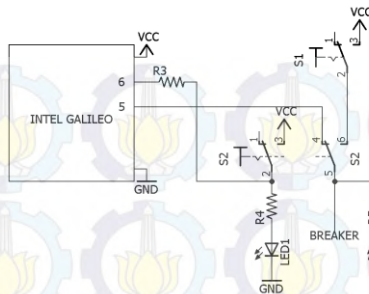


Gambar 3.11 Driver Relay

Gamba 3.11 menjelaskan *driver relay* bekerja berdasarkan prinsip kerja *relay* yaitu Secara prinsip, *relay* merupakan tuas saklar yang berfungsi untuk memutuskan dan menyambungkan kontak sesuai dengan perintah yang di berikan pada *driver relay*. Penggunaan *driver relay* untuk mengontrol kondisi panel distribusi listrik PLN jika terjadi gangguan yang memerlukan pemutusan pada panel distribusi PLN, maka *driver reley* inilah yang akan menjalankan perintah yang diberikan.

3.2.5 Perancangan Panel

Rangkaian panel berfungsi sebagai pengontrol panel secara lokal yang digukan untuk mengontrol kondisi panel ketika dilapangan secara langsung, rangkaian Panel pada gambar 3.12.

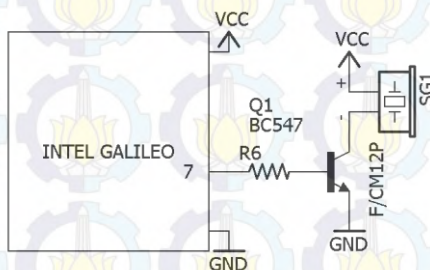


Gambar 3.12 Rangkaian Panel

Gambar 3.12 menjelaskan Panel adalah susunan beberapa bidang yang membentuk suatu kesatuan bentuk dan fungsi. Panel listrik merupakan tempat pengaturan pembagi dan pemutus aliran listrik. Panel kontrol listrik adalah peralatan yang berfungsi untuk mengatur dan mengendalikan beban listrik di bengkel listrik atau industri yang menggunakan motor listrik sebagai penggerak. Pada umumnya pengontrolan di industri ada dua jenis yaitu jenis manual dan jenis otomatis. Pengontrolan manual adalah pengontrolan motor listrik yang dilayani dengan alat kontrol manual.

3.2.6 Perancangan Buzzer

Rangkaian Buzzer sebagai pengingat pesan dan tanda, jika terjadi adanya gangguan pada sensor-sensor yang di gunakan untuk mendeteksi terjadinya gangguan, rangkaian Buzzer pada gambar 3.13.

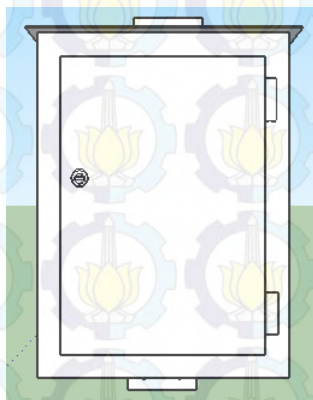


Gambar 3.13 Rangkaian Buzzer

Gambar 3.13 menjelaskan rangkaian *buzzer* jika terjadi adanya gangguan pada sensor maupun alat elektronik lainnya yang terdapat pada *box* panel maka akan ada pesan yang masuk pada *Intel Galileo* kemudian akan memerintahkan *buzzer* untuk menjalankan fungsinya yaitu sebagai pengingat dan alarm, selain *buzzer* rangkaian elektronik lainnya juga akan menjalankan tugasnya sebagai mana fungsinya jika terjadi adanya gangguan.

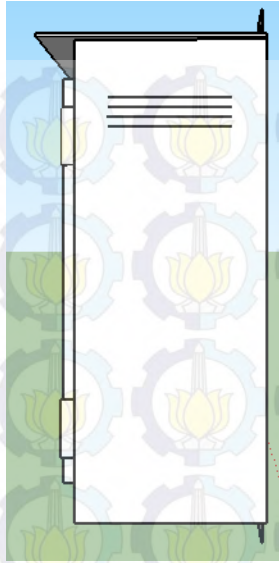
3.3 Perancangan Mekanik

Perancangan mekanik pada gardu distribusi listrik PLN terdiri dari *Box* Panel dan Rangkaian Breaker yang berfungsi untuk memutus dan menyambung arus listrik. Berikut adalah gambaran *box* panel gardu distribusi listrik PLN:



Gambar 3.14 Box Panel Tampak Depan

Gambar 3.14 menjelaskan *Box* panel berfungsi untuk pengan dan pelindung panel rancangan alat yang sudah dibuat selain itu juga *box* panel terbut berfungsi untuk kerapian suatu instalasi listrik. *Box* panel yang digunakan untuk pelindung panel dan rangkaian elektronik jika terlihat tampak depan memiliki spesifikasi pada pintu tinggi 29,5 cm , lebar pada pintu 20 cm dan lebar panel 25 cm. Dilihat pada sisi samping pada gambar 3.15.

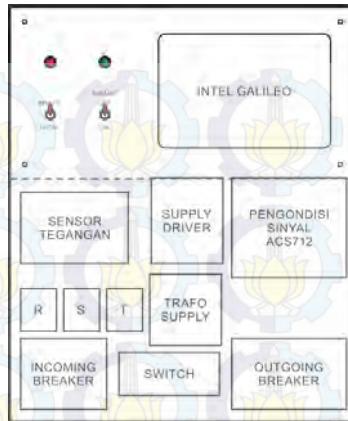


Gambar 3.15 Box Panel Tampak Samping

Box panel jika dilihat dari samping seperti pada gambar 3.15 yang memiliki spesifikasi tinggi box 35 cm dan luas sebesar 15 cm. Dilihat pada sisi samping box terdapat lubang-lubang yang telah di desain sedemikian rupa agar tetap terjaga keamanannya terkena air hujan, fungsi lubang-lubang tersebut untuk sirkulasi udara didalam panel.

3.3.1 Panel

Panel yang terdapat pada dalam box terlihat pada gambar 3.16 terlihat dari atas, pada panel dalam terdapat *Intel Galileo*, rangkain ACS712, rangkaian sensor tegangan dan rangkaian lainnya.

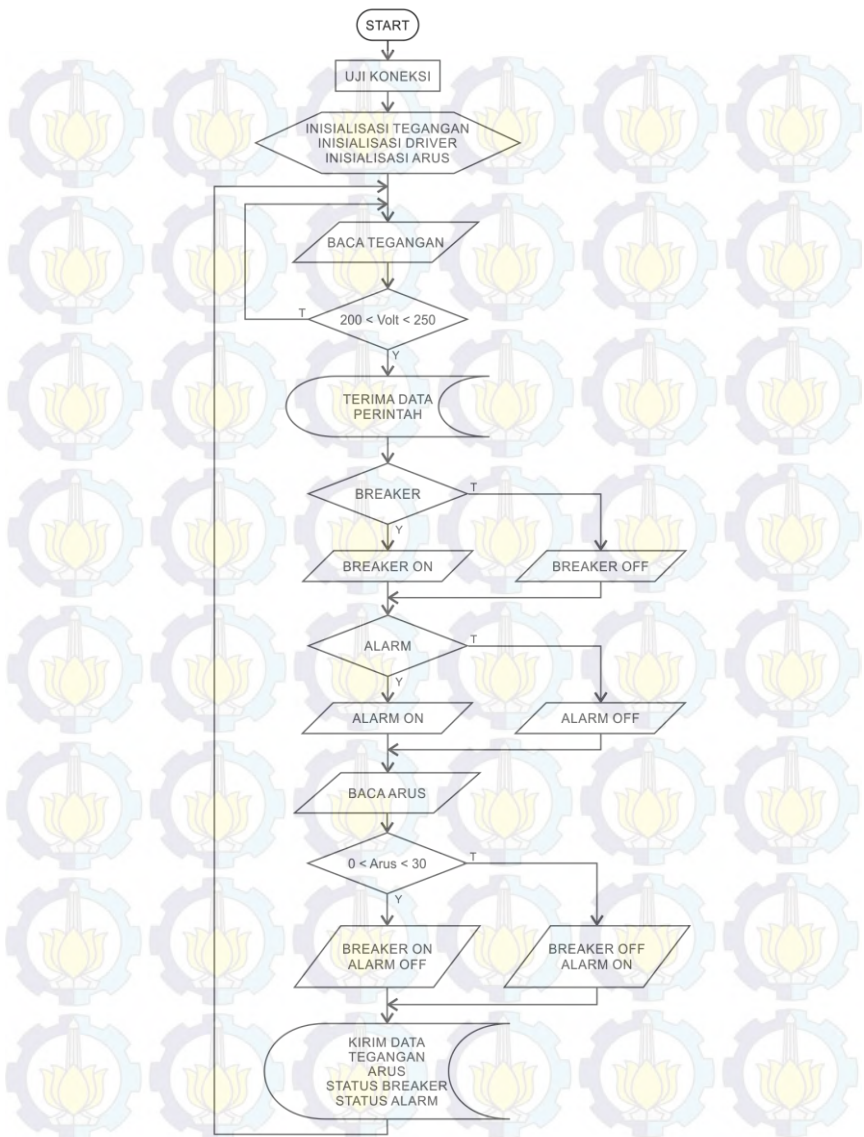


Gambar 3.16Panel

Pada dalam box panel terdapat rancangan mekanik maupun elektronik, yang bisa di lihat pada gambar 3.16. Dari desain pada gambar 3.16, terdapat blok-blok yang terpisah pisah antara satu dengan yang lainnya dikarenakan berbagai macam rangkaian yang digunakan berbeda antara satu dengan yang lainnya antara pengondisi sinyal, trafo, dan bagian *breaker*.

3.4 Perancangan Perangkat Lunak (Software)

Perancangan perangkat lunak (*software*) pada Tugas Akhir ini terdiri dari pemrograman *Intel Galileo Gen 2* dengan beberapa teknik diantaranya adalah percobaan koneksi, pemrograman *blink input-output*, pemrograman ADC, dan dasar pemrograman *web* dengan tampilan “*Hello World*”. Setelah semua program sukses, maka selanjutnya adalah menggabungkan semua program yang telah disusun dan dimodifikasi sehingga dapat diakses melalui *web browser* pada komputer. Secara spesifik, diagram alir dari keseluruhan program yang akan dituliskan pada *Intel Galileo* ada pada Gambar 3.17.



Gambar 3.17 Diagram Alir program keseluruhan pada Intel Galileo

3.4.1 Percobaan Koneksi Intel Galileo dengan PC

Alat ini memanfaatkan koneksi menggunakan jaringan *Ethernet* antara komputer *master* dengan *Intel Galileo* menggunakan metode TCP/IP. Syarat kedua peralatan dapat berinteraksi adalah diantara kedua peralatannya memiliki masing-masing *IP address* yang berbeda sehingga dapat di panggil satu sama lain. Pada percobaan koneksi, *Intel Galileo* memiliki IP 192.168.1.131 sedangkan komputer memiliki IP 192.168.1.6. percobaan komunikasi menggunakan *Command Prompt* dengan mengetikkan *Ping+Alamat IP tujuan*. Dengan menggunakan *Ping*, maka dapat diketahui kualitas hubungan antara komputer dengan *Intel Galileo*. Koneksi antara kedua peralatan ini sangat penting, karena penulisan program pada *Intel Galileo* menggunakan bahasa *JavaScript* yang dapat dituliskan secara *remote* melalui jaringan *Ethernet*. Akses remote dari *Intel Galileo* menggunakan *Putty*.

3.4.2 Pemrograman Blink Input-Output

Pemrograman IO dengan metode *blinking* bertujuan untuk menentukan kondisi dari IO secara otomatis bergantian dengan fungsi waktu. Pada percobaan *blink*, pin yang digunakan sebagai *output* adalah pin 13 yang secara *default*, pin 13 pada *board intel galileo* sudah terhubung dengan *led onboard* sehingga kondisi dari pin 13 dapat dipantau. Contoh kode program untuk *Blinking IO pin 13* :

```
var m=require('mraa');
console.log('MRAA Version: '+m.getVersion());

function sleep(delay)
{
    delay += new Date().getTime();
    while (new Date() < delay) { }
}

var led=new m.Gpio(13);
led.dir(m.DIR_OUT);

while(true)
{
    led.write(1);
```

```
sleep(200);  
led.write(0);  
sleep(200);  
}
```

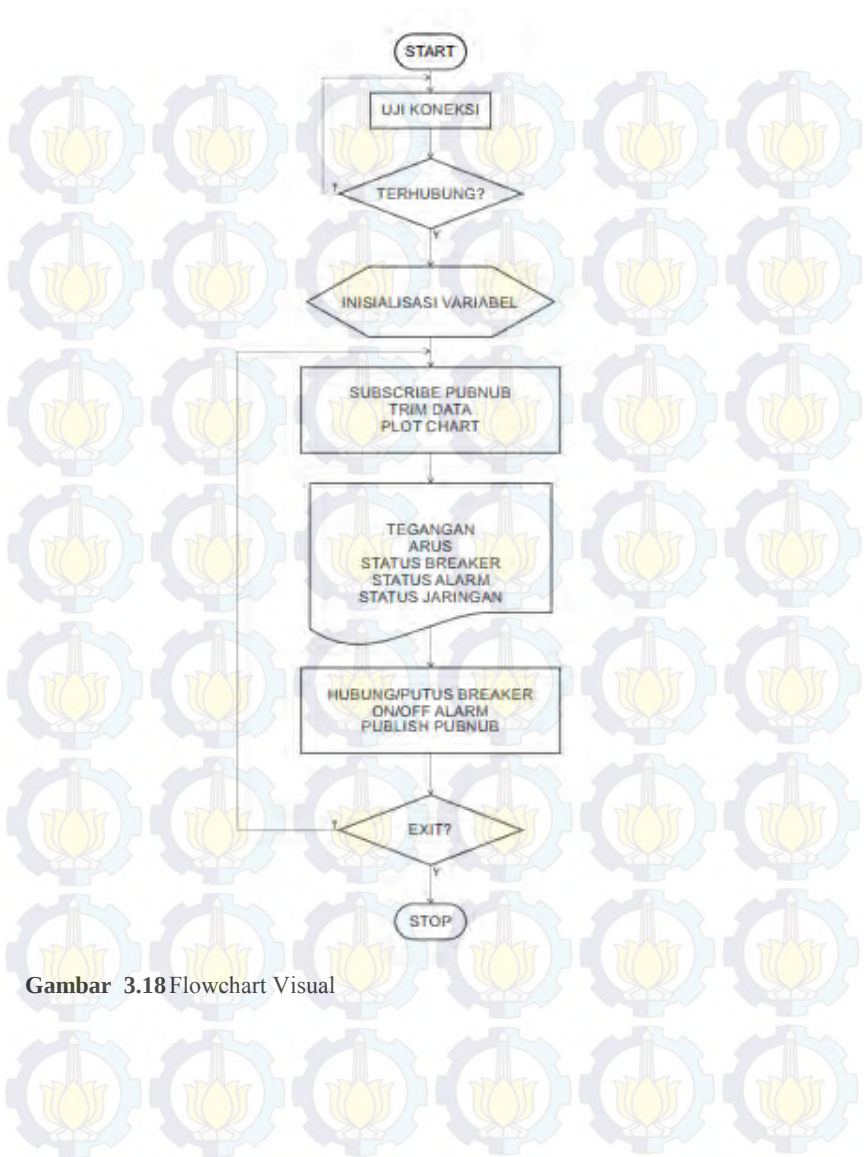
3.4.3 Pemograman Pembacaan ADC

Output dari sensor pada peralatan ini berupa tegangan, maka untuk pembacaannya menggunakan pin ADC pada *Intel Galileo*. Program yang digunakan menggunakan bahasa *JavaScript* dan menggunakan bantuan *library eksternal* yaitu “*mraa*”. Program ini bertujuan untuk membaca *analog input* pada *channel 0* dari *Intel Galileo* dan hasil pembacaan dari nilai ADC dan Tegangan akan ditampilkan pada console terminal. Contoh kode program untuk pembacaan ADC :

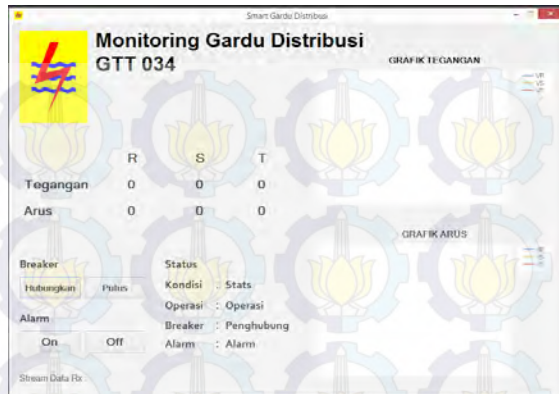
```
var mraa = require("mraa");  
var util = require('util');  
var express = require('express');  
var app = express();  
  
function sleep(delay)  
{  
  delay += new Date().getTime();  
  while (new Date() < delay) { }  
}  
  
var ch0 = new mraa.Aio(0);  
while(true)  
{  
  var e = ch0.read();  
  var val = (e/1024)*5;  
  console.log("Nilai ADC & Tegangan : "+ e +" & " +  
  val.toFixed(3));  
}
```

3.4.4 Pemograman Tampilan HMI

Human Machine Interface pada peralatan ini menggunakan *web browser* yang akan menampilkan besaran nilai dari tegangan, arus, *breaker*, dan status panel berikut tampilan pada gambar 3.18.



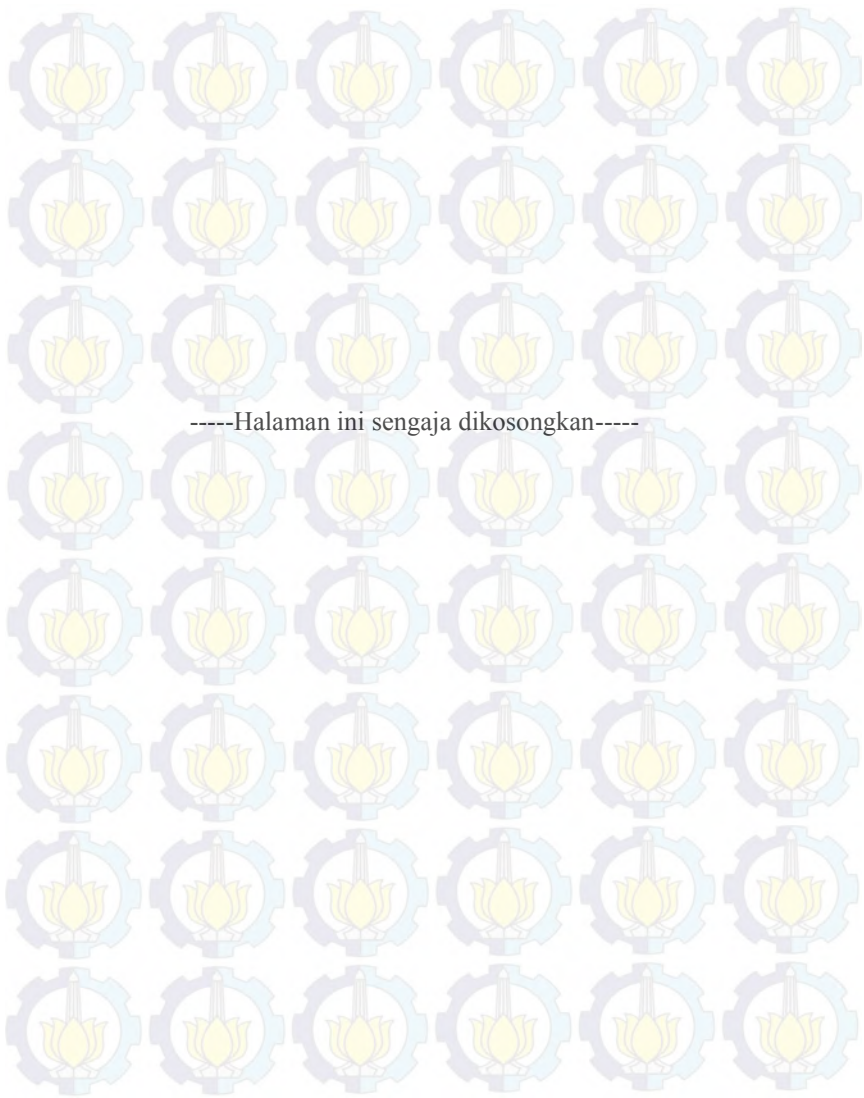
Gambar 3.18Flowchart Visual



Gambar 3.19 Tampilan HMI

Pada proyek tugas akhir monitoring dan kontrol panel distribusi PLN menggunakan *web* untuk komunikasi dalam pengiriman data dari distribusi PLN ke intel galileo maupun perintah yang dikirim dari *intel galileo* ke panel distribusi. Pada tampilan *HMI* terdapat status tegangan dan arus yang menggunakan 3 fasa yaitu R S T.

1. Menunjukkan tegangan yang pada panel distribusi yang nantinya bisa di monitoring jika terdapat gangguan pada tegangan 3 fasa, dalam kondisi normal pada tegangan akan menunjukana nilai-nilai dalam kondisi stabil.
2. Arus, memonitoring arus pada panel distribusi yang memberi informasi jika terjadi gangguan pada arus maka petugas PLN akan mengetahui gangguan yang terjadi pada arus panel distribusi PLN.
3. *Breaker*, berfungsi untuk mengontrol menyambung dan memutus tegangan arus listrik pada panel distribusi PLN jika terjadi gangguan dan digunakan jika diperlukan.
4. Alarm berfungsi sebagai pengingat atau pertanda adanya gangguan pada panel distribusi PLN, pada alarm terdapat perintah *on off*, *on* untuk menyalakan alarm jika terjadi gangguan *off* untuk mematikan alarm pada panel distribusi jika terjadi gangguan.
5. Status yang memberitahu kondisi-kondisi pada panel distribusi tegangan listrik PLN.



-----Halaman ini sengaja dikosongkan-----

BAB IV

HASIL REAL DAN IMPLEMENTASI

Pada bab ini akan dibahas mengenai pengujian dan analisa alat dari monitoring gardu distribusi tenaga listrik PLN yang telah dibuat.. Pengujian alat ini ditujukan untuk memastikan agar peralatan yang dibuat dapat berfungsi dengan baik. Pengujian tugas akhir ini meliputi pengujian sensor arus ACS172, sensor tegangan dan pengujian program *Intel Galileo*. Setelah melakukan beberapa pengujian alat, data yang diperoleh akan dianalisa untuk mengetahui proses kerja dari seluruh sistem alat yang dibuat.

4.1 Pengujian Sensor Arus ACS172

Untuk mengetahui arus yang mengalir dari sumber diperlukan suatu rangkaian yang dapat mengubah besaran suatu arus menuju ke sebuah peralatan elektronik dari sebuah sumber tegangan.[10] Sensor yang digunakan dalam rangkaian ini adalah sensor arus ACS712. Pengujian dari sensor arus ini secara keseluruhan dilakukan setelah semua sistem minimum terpasang pada sensor ini. Sensor arus ACS712 sendiri sebenarnya sudah memiliki datasheet pengukuran. Karakteristik sensor arus ACS712 ini adalah jika tidak diberi beban listrik yang dilewatkan secara seri pada sensor arus tersebut maka tegangan keluarannya adalah 2,5 volt pada saat arus nyala. Dalam pengujian sensor dilakukan dengan merubah beban, dengan cara menggunakan *clamp* meter sebagai pengukur yang mengalir pada arus *input*, hasil pengujian sensor yang didapat pada tabel 4.1.



Gambar 4.1 Hasil Gelombang Pengujian ACS712

Gambar 4.1 pengujian ACS712 menggunakan *oscilloscop* menghasilkan gelombang seperti pada gambar 4.1 dengan *input* arus 1 amper, selanjutnya pengujian sensor arus dan pengambilan data

menggunakan beban R 10 Ω 5,4 A, tang amper untuk mengatur *input* yang kita inginkan untuk pengambilan data dan AVOMeter digunakan untuk mengetahui *output* dari rangkaian pengondisi sinyal ACS712.

Pengujian Sensor Arus 3 fasa dilakukan secara manual menggunakan clamp meter sebagai input dari sensor arus, dengan tujuan untuk membandingkan data yang dihasilkan dengan data yang dilakukan secara sistem, hasil perbandingan akan diketahui hasil error dari dua perbandingan pengujian data. Untuk toleransi error ditentukan senilai 10% , hasil pengujian Sensor Arus pada tabel 4.2. untuk mengetahui hasil error dan rata-rata error digunakan rumus sebagai berikut:

$$Error = \frac{Output - Input}{Input} \times 100\%$$

$$Rata - Rata = \frac{jumlah\ error}{Banyak\ error}$$

4.2 Kalibrasi Pengujian Sensor Arus ACS712

Sebelum melakukan pengujian sensor arus ACS712 pada alat yang telah kami buat, dilakukan kalibrasi untuk mengetahui hasil yang sesuai antara pengukuran data secara real dengan data yang tertampil secara *visual* pada *remote* komputer, tabel 4.1 adalah hasil kalibrasi pada sensor arus ACS712

Tabel 4.1 Kalibrasi Sensor Arus ACS712 Fasa R

No	Input Clamp	ADC
1.	0 A	209
2.	0,5 A	267
3.	1 A	312
4.	1,5 A	362
5.	2 A	400
6.	2,5 A	452
7.	3 A	495
8.	3,5 A	542
9.	4 A	594

No	Input Clamp	ADC
10.	4,5 A	645
11.	5 A	690

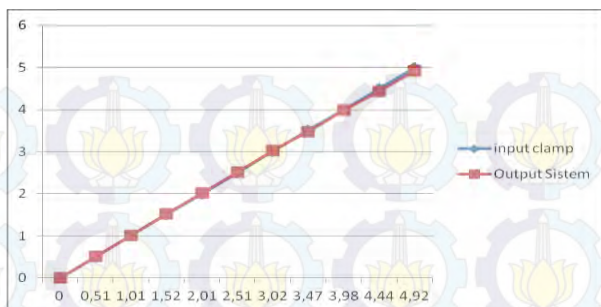
Tabel 4.1 menjelaskan proses kalibrasi pada sensor arus ACS712 pada fasa R menggunakan *clamp* meter untuk mengetahui arus yang mengalir menuju ke beban. Dari tabel 4.1 data ADC yang dibandingkan dengan data arus, didapatkan rumus fungsi hubungan besar arus dengan nilai ADC yang didapatkan dari regresi.

$$\text{Arus R} = (((0.010533644973698 * \text{ADC}) - 2.2573771117574))$$

Tabel 4.2 Hasil pengujian Sensor Arus Fasa R

No	Input Clamp	Output Sistem	Error %
1	0 A	0 A	0%
2	0,5 A	0,51 A	2%
3	1 A	1,01 A	1%
4	1,5 A	1,52 A	1,3%
5	2 A	2,01 A	0,5%
6	2,5 A	2,51 A	0,4%
7	3 A	3,02 A	0,6%
8	3,5 A	3,47 A	0,8%
9	4 A	3,98 A	0,5%
10	4,5 A	4,44 A	1,3%
11	5 A	4,92 A	1,6%
Rata-rata Error			1,5 %

Hasil dari perbandingan data pengujian sensor arus pada tabel 4.1 diperoleh nilai error 0,66%, untuk mengetahui linier dibuat grafik pada gambar 4.2.



Gambar 4.2 Grafik Sensor Arus Fasa R

Gambar 4.2 menjelaskan grafik pengujian data sensor arus Fasa R dilakukan secara manual menggunakan beban R 10 Ω 5,7 A di dapatkan hasil seperti pada tabel 4.2 pengujian diambil untuk mengetahui hasil eror perbandingan dari pengujian secara manual dengan menggunakan *input clamp* meter dengan *output* sistem, dari hasil perbandingan dapat disimpulkan bahwa hasil pengujian sensor arus 3 fasa dengan cara menggunakan *clamp* meter dan *output* sistem menghasilkan eror rata-rata 0,7%, masih didalam toleransi aman karena toleransi eror senilai 10%, pengambilan data selanjutnya sensor arus Fasa S yang diperoleh pada tabel 4.4 sebelum melakukan pengujian data pada sensor arus ACS712 pada fasa S dilakukan kalibrasi terlebih dahulu, pada tabel 4.3.

Tabel 4.3 Kalibrasi Sensor Arus ACS712 Pada Fasa S

No	Input Clamp	ADC
1.	0 A	206
2.	0,5 A	260
3.	1 A	310
4.	1,5 A	358
5.	2 A	403
6.	2,5 A	454
7.	3 A	500
8.	3,5 A	549
9.	4 A	597

No	Input Clamp	ADC
10.	4,5 A	646
11.	5 A	696

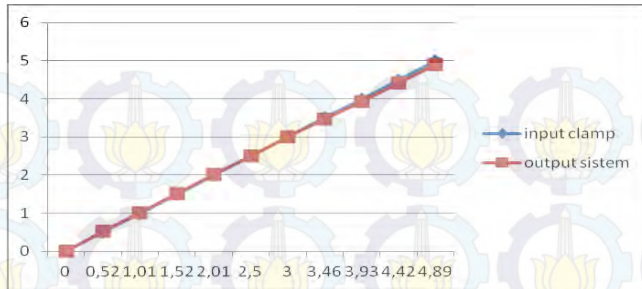
Tabel 4.3 menjelaskan proses kalibrasi pada sensor arus ACS712 pada fasa S menggunakan *clamp* meter untuk mengetahui arus yang mengalir menuju ke beban. Dari tabel 4.3 data ADC yang dibandingkan dengan data arus, didapatkan rumus fungsi hubungan besar arus dengan nilai ADC yang didapatkan dari regresi.

$$\text{Arus S} = (((0.010309615279189 * \text{ADC}) - 2.1665067704621))$$

Tabel 4.4 Hasil pengujian Sensor Arus Fasa S

No	Input Clamp	Output Sistem	Error %
1	0 A	0 A	0%
2	0,5 A	0,52 A	4%
3	1 A	1,01 A	1%
4	1,5 A	1,52 A	1,3%
5	2 A	2,01 A	0,5%
6	2,5 A	2,50 A	0%
7	3 A	3,00 A	0%
8	3,5 A	3,46 A	1,1%
9	4 A	3,93 A	1,7%
10	4,5 A	4,42 A	1,7%
11	5 A	4,89 A	2,2%
Rata-rata Error			1,5%

Hasil pengujian pada tabel 4.4 diperoleh hasil error senilai 1,13%, dari hasil perbandingan dibuat grafik seperti pada gambar 4.3 untuk mengetahui hasil linier pada grafik 4.3



Gambar 4.3 Grafik Sensor Arus Fasa S

Gambar 4.3 menjelaskan hasil pengujian dari perbandingan data sensor arus yang dilakukan secara otomatis menggunakan *input clamp* meter dengan data pengujian *output* dari sistem menghasilkan grafik pada gambar 4.3 yang menunjukkan hasil perbandingan *input* dari *clamp* meter dan *output* dari sistem masih terdapat error namun masih dalam toleransi, jadi grafik menunjukkan hasil dari pengujian data linier. Pengujian selanjutnya dilakukan pada fasa T hasil pengujian pada tabel 4.6. sebelum melakukan pengujian data pada sensor arus ACS712 pada fasa T dilakukan kalibrasi terlebih dahulu, pada tabel 4.5.

Tabel 4.5 Kalibrasi Sensor Arus Pada Fasa T

No	Input Clamp	ADC
1.	0 A	208
2.	0,5 A	266
3.	1 A	318
4.	1,5 A	368
5.	2 A	418
6.	2,5 A	472
7.	3 A	521
8.	3,5 A	570
9.	4 A	622
10.	4,5 A	670
11.	5 A	720

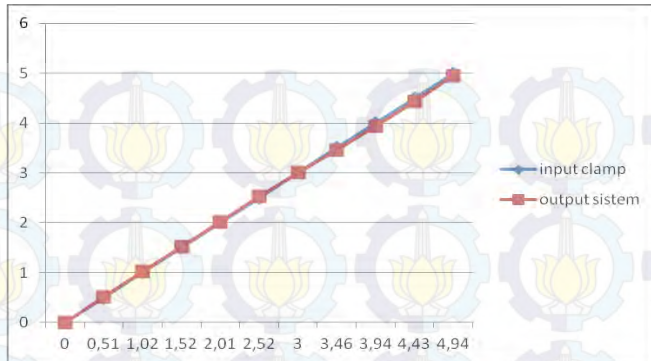
Pada proses kalibrasi pada sensor arus ACS712 pada fasa T menggunakan *clamp* meter untuk mengetahui arus yang mengalir menuju ke beban. Dari tabel 4.5 maka data ADC yang dibandingkan dengan data arus, didapatkan rumus fungsi hubungan besar arus dengan nilai ADC yang didapatkan dari regresi

$$\text{Arus T} = (((0.0098276955622793 * \text{ADC}) - 2.1038286574932))$$

Tabel 4.6 Hasil Pengujian Sensor Arus Fasa T

No	Input Clamp	Output Sistem	Error %
1	0 A	0 A	0%
2	0,5 A	0,51 A	2%
3	1 A	1,02 A	2%
4	1,5 A	1,52 A	1,3%
5	2 A	2,01 A	0,5%
6	2,5 A	2,52 A	0,8%
7	3 A	3,00 A	0
8	3,5 A	3,46 A	1,1%
9	4 A	3,94 A	1,5%
10	4,5 A	4,43 A	1,5%
11	5 A	4,94 A	1,2%
Rata-rata Error			2,2 %

Pada tabel 4.6 menjelaskan perbandingan dari pengujian data menggunakan *input clamp* meter dengan *output* sistem dan mengetahui hasil error pada perbandingan data, setelah diketahui hasil perbandingan dalam pengujian dan di peroleh error senilai 2,2% maka sensor tegangan bisa di gunakan pada Proyek Tugas Akhir karena nilai error masih dalam toleransi yang di tentukan senilai 10% dengan mengetahui nilai error masih dalam toleransi, dan selanjutnya dibuat grafik pada gambar 4.4.



Gambar 4.4 Grafik Sensor Arus Fasa T

Gambar 4.4 menjelaskan hasil pengujian dari perbandingan data sensor arus pada fasa T yang dilakukan secara otomatis menggunakan *input clamp* meter dengan data pengujian *output* dari sistem menghasilkan grafik pada gambar 4.4 yang menunjukkan hasil perbandingan *input* dari *clamp* meter dan *output* dari sistem masih terdapat error namun masih dalam toleransi, jadi grafik 4.4 menunjukkan data linier dari hasil yang telah dilakukan.

4.3 Pengujian Sensor Tegangan

Pengujian Sensor Tegangan 3 fasa pada panel Tugas Akhir dilakukan secara manual menggunakan *clamp* meter sebagai *input* dari sensor tegangan, dengan tujuan untuk membandingkan data yang diperoleh dengan data yang dilakukan secara sistem, hasil perbandingan akan diketahui hasil error dari dua perbandingan pengujian data. Untuk toleransi error ditentukan senilai 10% , hasil pengujian Sensor tegangan pada fasa R tabel 4.8. untuk mengetahui hasil error digunakan rumus sebagai berikut:

$$\text{Error} = \frac{\text{Output} - \text{Input}}{\text{Input}} \times 100\%$$

$$\text{Rata - Rata} = \frac{\text{jumlah error}}{\text{Banyak error}}$$

4.4 Kalibrasi Pengujian Tegangan

Sebelum melakukan pengujian tegangan pada alat yang telah dibuat, dilakukan kalibrasi untuk mengetahui hasil yang sesuai antara

pengukuran data secara *real* dengan data yang tertampil secara *visual* pada *remote* komputer, tabel 4.7 adalah hasil kalibrasi pada tegangan pada fasa R.

Tabel 4.7 Kalibrasi Tegangan Pada Fasa R

No	V Input Clamp	ADC
1.	100 V	141
2.	120 V	199
3.	140 V	257
4.	160 V	316
5.	180 V	375
6.	200 V	433

Pada proses kalibrasi tegangan pada fasa R menggunakan *clamp* meter untuk mengetahui arus yang mengalir menuju ke beban. Dari tabel 4.7 maka data ADC yang dibandingkan dengan data tegangan, didapatkan rumus fungsi hubungan besar tegangan dengan nilai ADC yang didapatkan dari regresi.

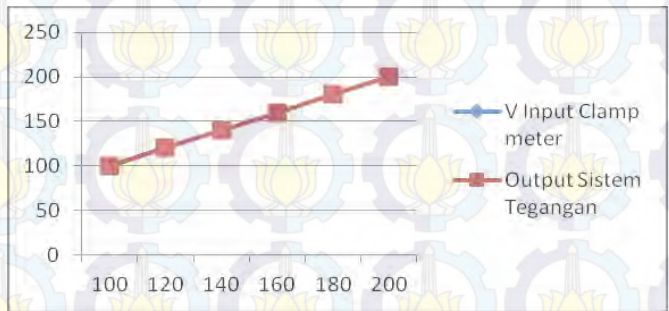
$$\text{Tegangan} = (51.914523965309 + (0.34195982347946 * \text{ADC}))$$

Tabel 4.8 Hasil Pengujian Sensor tegangan Pada Fasa R

No	V Input Clamp meter	Output Sistem Tegangan	Error
1	100 V	100 V	0%
2	120 V	120 V	0%
3	140 V	140 V	0%
4	160 V	160 V	0%
5	180 V	180 V	0%
6	200 V	200 V	0%
Rata-rata Error			0 %

Pada tabel 4.4 menjelaskan perbandingan dari pengujian data menggunakan *input clamp* meter dengan *output* sistem dan mengetahui hasil error pada perbandingan data, setelah diketahui hasil perbandingan dalam pengujian dan di peroleh error senilai 0% makan sensor tegangan

bisa di gunakan pada Proyek Tugas Akhir karena dengan mengetahui nilai eror masih dalam toleransi,dan selanjutnya dibuatl grafik pada tabel 4.6.



Gambar 4.5 Grafik Sensor Tegangan Pada Fasa R

Pada gambar 4.5 menjelaskan hasil pengujian dari perbandingan data sensor tegangan pada fasa R yang dilakukan secara otomatis menggunakan *input clamp* meter dengan data pengujian *output* dari sistem menghasilkan grafik pada gambar 4.5 yang menunjukana hasil perbandingan *input* dari *clamp* meter dan *output* dari sistem masih terdapat eror namun masih dalam toleransi, jadi grafik menunjukan hasil dari pengujian data linier. Pengujian, selanjutnya dilakukan pada fasa S hasil pengujian pada tabel 4.9 sebelum melakukan pengujian data tegangan pada fasa S dilakukan kalibrasi terlebih dahulu, pada tabel 4.9.

Tabel 4.9 Kalibrasi Tegangan Pada Fasa S

No	V Input Clamp	ADC
1.	100 V	140
2.	120 V	199
3.	140 V	257
4.	160 V	316
5.	180 V	376
6.	200 V	434

Pada proses kalibrasi tegangan pada fasa S menggunakan *clamp* meter untuk mengetahui arus yang mengalir menuju ke beban. Dari tabel 4.9 maka data ADC yang dibandingkan dengan data tegangan,

didapatkan rumus fungsi hubungan besar tegangan dengan nilai ADC yang didapatkan dari regresi.

$$\text{Tegangan S} = (52.477566640274 + (0.33979941937187 * \text{ADC}))$$

Tabel 4.10 Hasil Pengujian Sensor Tegangan Pada Fasa S

No	V Input Clamp meter	Output Sistem Tegangan	Error
1	100 V	100 V	0%
2	120 V	120 V	0%
3	140 V	140 V	0%
4	160 V	160 V	0%
5	180 V	180 V	0%
6	200 V	200 V	0%
Rata-rata Error			0 %

Hasil pengujian yang telah dilakukan diperoleh data pada tabel 4.10 pengujian untuk membandingkan data yang telah dilakukan dengan cara pengambilan data *input* menggunakan *clamp* meter dengan pengambilan data dari *output* sistem, selanjutnya hasil perbandingan diketahui hasil error, selama error masih dalam toleransi senilai 10% maka sensor tegangan pada fasa S bisa digunakan proyek Tugas Akhir.



Gambar 4.6 Grafik sensor tegangan Pada Fasa S

Hasil pengujian dari perbandingan data sensor tegangan pada fasa S yang dilakukan secara otomatis menggunakan *input clamp* meter dengan data pengujian *output* dari sistem menghasilkan grafik pada gambar 4.6 yang menunjukkan hasil perbandingan, tujuan dari grafik 4.6 untuk mengetahui hasil linier dari data perbandingan. Pengujian

selanjutnya sensor tegangan pada fasa T hasil pengujian pada tabel 4.12. sebelum melakukan pengujian data tegangan pada fasa T dilakukan kalibrasi terlebih dahulu, pada tabel 4.11.

Tabel 4.11 Kalibrasi Tegangan Pada Fasa T

No	V Input Clamp	ADC
1.	100 V	140
2.	120 V	199
3.	140 V	257
4.	160 V	316
5.	180 V	376
6.	200 V	434

Pada proses kalibrasi tegangan pada fasa T menggunakan *clamp* meter untuk mengetahui arus yang mengalir menuju ke beban. Dari tabel 4.11 maka data ADC yang dibandingkan dengan data tegangan, didapatkan rumus fungsi hubungan besar tegangan dengan nilai ADC yang didapatkan dari regresi

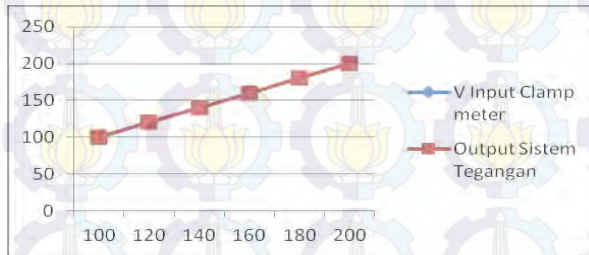
$$\text{Tegangan T} = (52.390888963028 + (0.3402990506809 * \text{ADC})).$$

Tabel 4.12 Hasil Pengujian Sensor Tegangan Pada Fasa T

No	V Input Clamp meter	Output Sistem Tegangan	Error
1	100 V	100 V	0%
2	120 V	120 V	0%
3	140 V	140 V	0%
4	160 V	160 V	0%
5	180 V	180 V	0%
6	200 V	200 V	0%
Rata-rata Error			0 %

Hasil pengujian sensor tegangan pada fasa T diperoleh data pada tabel 4.12 yang membandingkan pengujian data secara manual menggunakan *input clamp* meter dengan pengujian secara sistem dengan pengambilan data *output* sistem, untuk mengetahui hasil error sensor

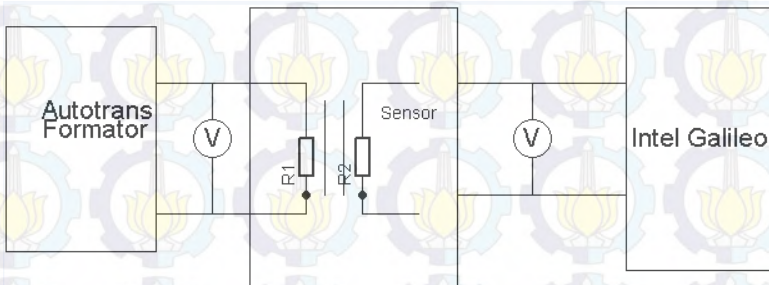
tegangan pada fasa T dan hasil rata-rata eror 0%, jadi dari hasil eror disimpulkan bahwa sensor tegangan pada fasa T bisa digunakan untuk Poyek tugas Akhir, selanjutnya dibuat grafik pada gambar 4.7



Gambar 4.7 Grafik Sensor Tegangan Pada Fasa T

Gambar 4.7 menjelaskan hasil pengujian dari perbandingan data sensor tegangan pada fasa T yang dilakukan secara otomatis menggunakan *input clamp* meter dengan data pengujian *output* dari sistem menghasilkan grafik pada gambar 4.7 yang menunjukkan hasil perbandingan, tujuan dari grafik 4.7 untuk mengetahui hasil linier dari data perbandingan.

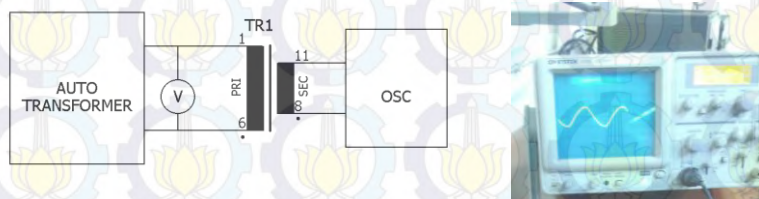
Untuk pengujian trafo dilakukan pengujian seperti pada gambar 4.8, pengujian dilakukan untuk memastikan trafo bisa digunakan untuk kelanjutan kegunaan Tugas Akhir nantinya, pengujian dilakukan dengan menggunakan alatukur avometer.



Gambar 4.8 Output dari Trafo

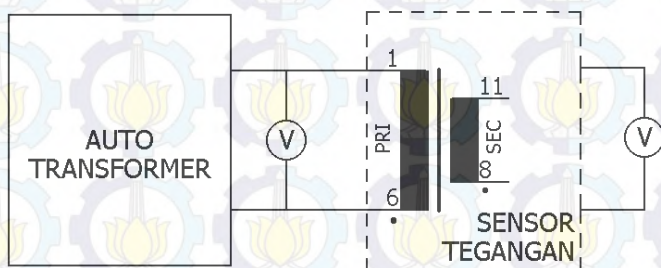
Pada saat pengujian tegangan pada trafo menggunakan avo meter AC dan selanjutnya *output* dibaca oleh *oscilloscop* yang mengeluarkan gelombang AC seperti yang di tunjukan pada gambar 4.9, hasil pengujian data sesuai dengan keinginan yaitu 23,1 data yang di peroleh

menunjukkan bahwa trafo yang di gunakan pada Tugas Akhir ini bisa digunakan sebagai mana mestinya dengan cara kerjanya.



Gambar 4.9 Bentuk Gelombang Tegangan AC

Gambar 4.9 hasil yang dipeoleh dari pengujian trafo yang kemudian data di baca oleh *oscilloscop* untuk mengetahui hasil gelombangnya, dari hasil yang di peroleh pengujian, trafo bisa digunakan pada Tugas Akhir ini dan bekerja dengan baik. Setelah menguji *output* trafo dengan pengukuran AC selanjutnya, pengujian *output* Tegangan DC seperti pada gambar 4.10.



Gambar 4.10 Output tegangan DC

Gambar 4.10 menjelaskan pada saat pengujian tegangan DC menggunakan avo meter, pengujian pengambilan data dilakukan untuk mengetahui *output* yang dihasilkan trafo menggunakan avometer DC, memastikan *output* sesuai dengan keinginan dan dapat dilakukan langkah pengujian selanjutnya dengan menggunakan *oscilloscop*, untuk mengetahui hasil gelombang yang dihasilkan, seperti apa yang di inginkan. Bentuk gelombang output tegangan DC pada gambar 4.11.

4.5 Pengujian Supply Rangkaian Pengondisi Sinyal ACS712

Pada tahap pengujian ini menggunakan Avometer untuk mengetahui *output* dari *supply* rangkaian pengondisi sinyal ACS712, dengan data yang dihasilkan saat pengujian maka rangkaian pengondisi sinyal dapat digunakan, terlihat pada tabel 4.13.

Tabel 4.13 Tabel Pengondisi Sinyal

Output Trafo	Output Dioda	Output Regulator
11,76 Vac	13,34 V	11,87 V

ari data pengujian supply rangkaian pengondisi sinyal ACS712 didapatkan hasil yang sesuai untuk kegunaan alat Tugas Akhir nantinya dengan data tersebut rangkaian supply pengondisi sinyal bisa dijalankan dengan alat elektronik lainnya.

4.6 Pengujian Driver Relay

pada pengujian *Driver Relay* menggunakan Avometer untuk melihat hasil output dioda, regulator dan *input trigger*, dihasilkan data pada tabel 4.14

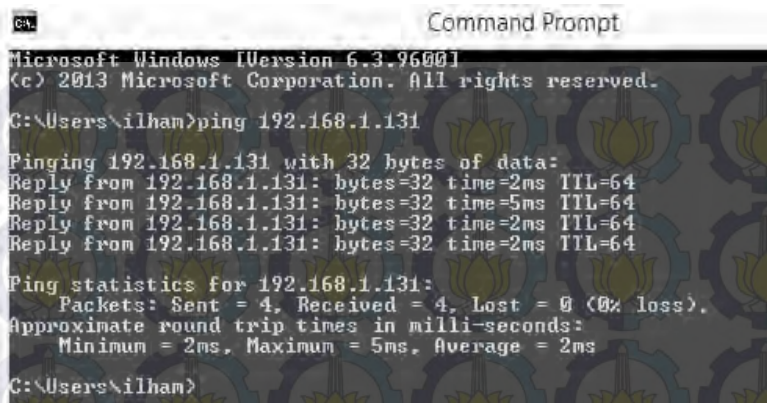
Tabel 4.14 Tabel Driver Relay

Output Dioda	Output Regulator	Input Triger	V Relay
13,38 V	11,95 V	3,220 V	11,61 V

Gambar 4.14 menjelaskan hasil percobaan awal pada *relay* komponen listrik yang bekerja berdasarkan prinsip induksi medan elektromagnetis. Jika sebuah penghantar dialiri oleh arus listrik, maka di sekitar penghantar tersebut timbul medan *magnet*. Medan magnet yang dihasilkan oleh arus listrik tersebut selanjutnya diinduksikan ke logam *ferromagnetis*, *relay* juga berfungsi sebagai saklar.

4.7 Pengujian Sistem Intel Galileo

Dalam pengujian program *intel Galileo* ini adalah menguji jaringan yang di akses oleh *intel galileo* hasil pengujian terdapat pada gambar.



```
C:\>
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\ilham>ping 192.168.1.131

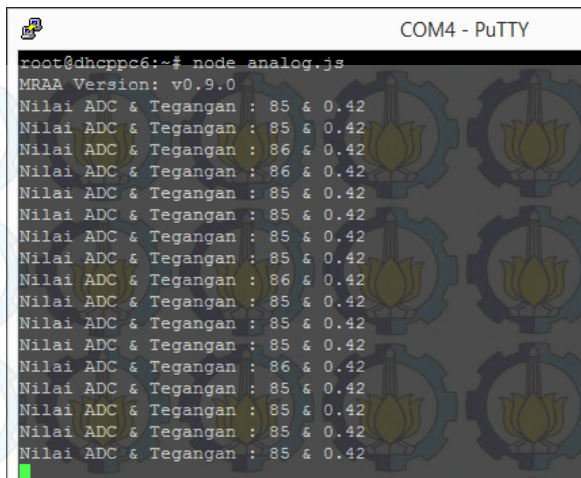
Pinging 192.168.1.131 with 32 bytes of data:
Reply from 192.168.1.131: bytes=32 time=2ms TTL=64
Reply from 192.168.1.131: bytes=32 time=5ms TTL=64
Reply from 192.168.1.131: bytes=32 time=2ms TTL=64
Reply from 192.168.1.131: bytes=32 time=2ms TTL=64

Ping statistics for 192.168.1.131:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 2ms, Maximum = 5ms, Average = 2ms

C:\Users\ilham>
```

Gambar 4.11 Uji Komunikasi

Pada pengujian Koneksi, *Intel Galileo* dengan PC dihubungkan langsung dengan kabel RJ45 dengan metode *point to point*. Dengan metode *point to point*, maka koneksi antara kedua *hardware* harus memiliki alamat IP pada masing-masing peralatan sebagai identifikasi dari perangkat yang terhubung. Pengujian kali ini menggunakan IP yang di *set Static* pada kedua perangkat, *Intel Galileo* di set pada 192.168.1.131 dan pada master komputer di set pada 192.168.1.6. pengujian menggunakan teknik *ping* pada *command prompt* master komputer. Dengan mengetikkan “ping 192.168.1.131” pada komputer maka akan tampil hasil pada gambar 4.13. Hasil dari pengujian ping dengan *command prompt* menunjukkan bahwa antara kedua perangkat sukses komunikasi dengan waktu jawaban antara kedua perangkat rata-rata memiliki waktu 2ms. Pada akhir dari pengujian maka ping juga akan menampilkan berapa jumlah paket data yang dikirimkan, diterima, dan paket yang gagal diterima. Dari hasil pengujian, paket yang dikirimkan berjumlah 4 dan yang diterima berjumlah 4 dengan *lost* 0 maka dapat disimpulkan bahwa komunikasi diantara kedua perangkat tidak memiliki kegagalan dan komunikasi berjalan dengan baik ditandai dengan jumlah *lost* nya bernilai 0%. Dalam keadaan lain, apabila ada kegagalan maka salah satu data yang terkirim tidak diterima lagi maka yang tampil bukan waktu dari penerimaan, melainkan tulisan “*Request Timed Out*”.



```
COM4 - PuTTY
root@dncppc6:~# node analog.js
MRAA Version: v0.9.0
Nilai ADC & Tegangan : 85 & 0.42
Nilai ADC & Tegangan : 85 & 0.42
Nilai ADC & Tegangan : 86 & 0.42
Nilai ADC & Tegangan : 86 & 0.42
Nilai ADC & Tegangan : 85 & 0.42
Nilai ADC & Tegangan : 85 & 0.42
Nilai ADC & Tegangan : 85 & 0.42
Nilai ADC & Tegangan : 85 & 0.42
Nilai ADC & Tegangan : 86 & 0.42
Nilai ADC & Tegangan : 85 & 0.42
Nilai ADC & Tegangan : 85 & 0.42
Nilai ADC & Tegangan : 85 & 0.42
Nilai ADC & Tegangan : 86 & 0.42
Nilai ADC & Tegangan : 85 & 0.42
Nilai ADC & Tegangan : 85 & 0.42
Nilai ADC & Tegangan : 85 & 0.42
Nilai ADC & Tegangan : 85 & 0.42
Nilai ADC & Tegangan : 85 & 0.42
Nilai ADC & Tegangan : 85 & 0.42
```

Gambar 4.13 Uji Analog Input

Analog *Input Channel* merupakan jenis pin *input* yang terdapat pada *Intel Galileo*. Cara kerja dari analog *input* adalah dengan membaca tegangan yang masuk pada pin analog *input* dengan *range* tegangan antara 0-5 Volt akan dicacah dan dikonversi menjadi satuan digital dengan *range* sesuai resolusi dari *Intel Galileo* yaitu sebesar 10 bit. Maka tegangan yang masuk pada pin analog yang besarnya antara 0-5Volt akan dikonversi menjadi satuan dengan besar *range* antara 0-1023. Resolusi dari pin adc pada *board* akan mempengaruhi sensitivitas pin analog *input* terhadap perubahan tegangan. Pada pengujian analog *input*, *board intel galileo* yang memiliki 6 *channel* pin analog *input* dipilih *channel* analog 0 dan dihubungkan dengan sebuah potensiometer. Nilai hasil konversi berupa ADC dan tegangan akan dikirimkan ke komputer dan tampil pada *Putty* terminal. Nilai ADC yang terbaca pada analog pin adalah hasil konversi dari nilai tegangan yang terukur pada pin analog *input* yang telah tercacah dan dikonversi menjadi satuan dengan *range* 0-1023. Hasil pengujian tampil pada *putty* terminal dengan nilai ADC pada kisaran 805-810. Setelah terukur nilai ADC, maka selanjutnya adalah pengkonversian kembali menjadi nilai tegangan dengan cara mengalikan nilai ADC dikalikan 5 Volt dibagi dengan nilai maksimal dari resolusi ADC. Setelah proses kalkulasi, maka akan tampil pada *putty* terminal nilai tegangan yang masuk pada pin analog *input channel* 0. Dengan resolusi ADC pada *Intel Galileo* sebesar 10 bit, maka

setiap perubahan tegangan sebesar 4mV dapat terukur. Dengan adanya pin analog *input*, maka tegangan keluaran dari rangkaian pengondisi sinyal ACS dan rangkaian sensor tegangan dapat terukur nilainya dan dapat dikonversi menjadi tegangan asli dari sumber tegangan dan arus asli yang mengalir dari sumber tegangan menuju beban.



Gambar 4.14 Uji Web

Gambar 4.14 pada tahap uji *web* ini pengujian terakhir untuk pengujian *intel galileo* dari hasil pengujian program *intel galileo* dan di pastikan program bisa digunakan dalam menjalankan monitoring gardu panel distribusi listrik PLN. Pengujian web ini digunakan untuk mencoba pembuatan *web server* sederhana yang menggunakan *Intel Galileo* dan ditampilkan pada *web browser* komputer master. Dengan bahasa *JavaScript*, maka pembuatan *web* menjadi lebih mudah dikarenakan dasar dari pemrograman yang mengatur bagaimana perilaku tampilan web adalah javascript. Selain lebih mudah, bahasa *javascript* juga lebih sederhana dan mudah dimengerti. Pengujian ini dilakukan dengan cara pemanggilan *IP Intel Galileo* diikuti dengan *Port Web* yang digunakan sehingga apabila sukses terkoneksi maka akan muncul tampilan *web* yang telah di buat.

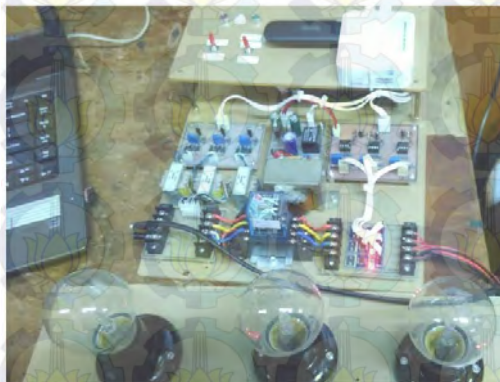
4.8 Pengujian Keseluruhan

Pengujian alat secara keseluruhan menunjukkan bahwa sistem monitoring dan sistem kontrol pada panel distribusi tegangan listrik PLN sudah berjalan tanpa gangguan dan berjalan sesuai fungsinya. Pengujian dilakukan secara bertahap. Cara pengujian alat dimulai dari konsi panel normal seperti pada gambar 4.16.



Gambar 4.15 Kondisi Breaker Terputus

Gambar 4.15 kondisi *breaker* pada saat panel dalam status normal dan breaker terputus menyatakan bahwa kondisi panel diatribusi dalam kondisi normal tidak ada gangguan, untuk mengetahui *breaker* berkerja dengan baik atau tidak pada alat tugas akhir ini menggunakan lampu pijar 100 watt sebagai beban, seperti pada gambar 4.16.



Gambar 4.16 Kondisi Normal Breaker Terputus

Pada saat *breaker* terputus maka pada beban lampu tidak menyala namun kondisi panel normal, breaker disini sebagi kontrol untuk

menyambung dan memutus tegangan listrik ke beban. Dan selanjutnya *breaker* pada status terputus pada gambar 4.17.



Gambar 4.17 Kondisi Breaker Terhubung

Gambar 4.17 pada saat kondisi *breaker* terhubung, status normal dan status *breaker* terhubung maka beban pada lampu yang sebagai indikator status *breaker* akan menyala dan itu menyatakan status *breaker* terhubung seperti pada gambar 4.18



Gambar 4.18 Kondisi Normal Breaker Terhubung

Dalam kondisi *breaker* terhubung maka beban pada lampu akan menyala secara otomatis, tanpa harus menghubungkan dari panel secara langsung, jadi *breaker* bisa dikontrol melalui *web* yang telah disediakan,

The screenshot displays a PLC ladder logic program for a motor control system. The program includes a main circuit with a 3-phase supply (228V, 5.0A) and a control circuit with a stop button (S1), a start button (S2), and a thermal relay (T1). The control logic uses a timer (T1) to control the motor's operation. The status of the system is displayed on a screen, showing 'Kondisi : Normal', 'Operasi : Remote', 'Breaker : Terputus', and 'Alarm : Menyala'. The screen also shows 'Stream Data Received: 00000000' and '1454415152128/5431'.

Gambar 4.19 dalam pengujian *web* untuk menyalakan alarm secara otomatis menggunakan operasi *remote* dapat dilakukan dengan baik dan bisa digunakan pada alat Tugas Akhir, pada alarm terdapat perintah *on* dan *off*, dalam pengujian saat alarm di beri perintah *on* maka status alarm berubah menjadi status menyala, menandakan bahwa alarm berbunyi, selanjutnya pada perintah *off* seperti pada gambar 4.20.

[illegible]

Gambar 4.20 Status Alarm Mati

Pada pengujian *web* untuk mematikan alarm secara otomatis menggunakan operasi *remote* dapat dilakukan dengan baik dan bisa digunakan pada alat tugas Akhir, pada alarm terdapat perintah *on* dan *off*, dalam pengujian yang telah dilakukan menggunakan perintah *on* dan selanjutnya menggunakan perintah *off*. Dalam pengujian mematikan alarm secara otomatis dengan memberi perintah *off* maka status alarm akan berubah menjadi mati menandakan alarm sudah tidak berbunyi atau mati. Kesimpulan dari seluruh pengujian alat dapat dilakukan dengan baik dan bisa digunakan sebagaimana mestinya.



BAB V

PENUTUP

Bab penutup ini berisi tentang kesimpulan proyek tugas akhir yang berjudul Monitoring panel distribusi PLN Berbasis *Web* Menggunakan Sistem *Board Intel Galileo*, kesimpulan dari hasil pengujian dan analisa data, serta saran untuk Monitoring panel distribusi PLN ini kedepannya.

5.1 Kesimpulan

Dari Tugas Akhir yang telah dilakukan, dapat diambil beberapa kesimpulan sebagai berikut:

1. Dari pengujian arus di peroleh nilai rata-rata eror 1,5% dengan toleransi 10% dan hasil pengujian tegangan diperoleh nilai eror 0% .
2. Pada alat yang telah dibuat komunikasi antara perangkat yang ada di lapangan dengan MQTT Server menggunakan router yang terhubung ke Internet. Sehingga dapat menjangkau jaringan dengan keterandalan yang rendah.
3. Penggunaan tampilan pada komputer master menggunakan bahasa *Visual C#* yang memiliki keunggulan berupa program yang berukuran kecil dan memiliki efektifitas koneksi tinggi serta tampilan visual yang lebih mudah dipahami.
4. Tampilan *monitoring* pada komputer dapat menampilkan hasil *monitoring* berupa arus dan tegangan, dan terdapat alarm sebagai indikator jika terjadi gangguan sesuai yang diinginkan.
5. Modem *router* dapat mengirimkan data yang diproses oleh *intel galileo* ke komputer dengan menggunakan MQTT, proses kecepatan dalam pengiriman data tergantung jaringan yang mendukung.

5.2 Saran

Dengan memperhatikan beberapa kelemahan dan kekurangan proyek Tugas Akhir ini, maka diberikan beberapa saran yang dapat dikembangkan demi kebempurnaan dari proyek Tugas Akhir ini, adapun beberapa saran tersebut yaitu:

1. Penggunaan tampilan pada web memiliki kelemahan yaitu semakin banyak elemen web yang digunakan maka akan

mempengaruhi kecepatan transfer datanya. Tampilan web yang sederhana lebih disarankan untuk penggunaan secara realtime.

2. Penggunaan sensor arus berupa ACS712 tidak disarankan pada peralatan real dikarenakan kapasitas hantarnya yang kecil.
3. Pemasangan CT (*current transformer*) pada *element* pengukur arus, untuk pemasangan pada panel distribusi PLN secara *real*.

DAFTAR PUSTAKA

- [1] Dadang Iskadar, “*Sistem Informasi Gardu Induk Dan Gardu Distribusi PLN*”, Universitas Gadjah Mada Yogyakarta 2011
- [2] Aprianto, Agung (2012), “*Pemeliharaan Trafo Distribusi*”, Universitas Diponegoro –Semarang
- [3] Alif Rochman Aji, “*Monitoring Arus Dan tegangan di cabang Saluran Rumah Dengan Mikrokontroler Media Wifi*”, Proyek Tugas Akhir, Institut Teknologi Sepuluh Nopember, 2013.
- [4] _____, “*pembagi tegangan (Voltage Divider)*”, <http://elektronika-dasar.web.id/pembagi-tegangan-voltage-divider/>, 11 Maret 2016
- [5] Aulia Faqih, “*Perkenalan Dengan Intel Galileo*”, <http://teknojurnal.com/berkenan-dengan-intel-galileo/>, 12 Maret 2016
- [6] Wahana Komputer Semarang. “*Web Berbasis Java Scrip Dan CGI*”, Andi, Yogyakarta, 2001
- [7] Robert J.tibshirani, “*An Introduction to the Bootstrap*”, CRC press, London, June 1993.
- [8] _____, “*Pengertian Optocoupler dan Prinsip Kerjanya*” <http://teknikelektronika.com/pengertian-optocoupler-fungsi-prinsip-kerja-optocoupler/>, 14 Maret 2016
- [9] Riza Agung Firmansyah, Titiek Suheta, Dedi Antoni, ”*Perancangan Alat Monitoring Dan Penyimpan Data Pada Panel Hubung Tegangan Rendah Di Trafo Gardu Distribusi Berbasis Mikrokontroler*”, Institut Teknologi Adhi Tama Surabaya 2015



LAMPIRAN A

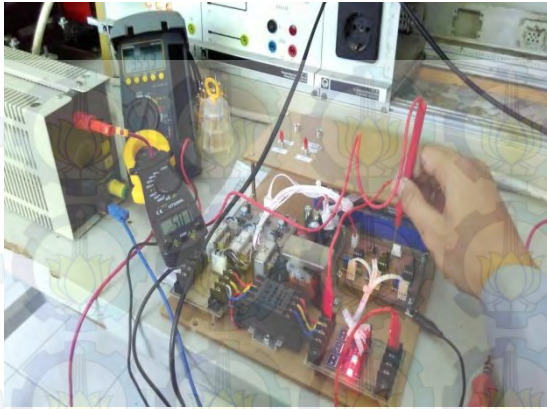
FOTO

Box Panel Monitoring Dan Kontrol Sistem Distribusi PLN



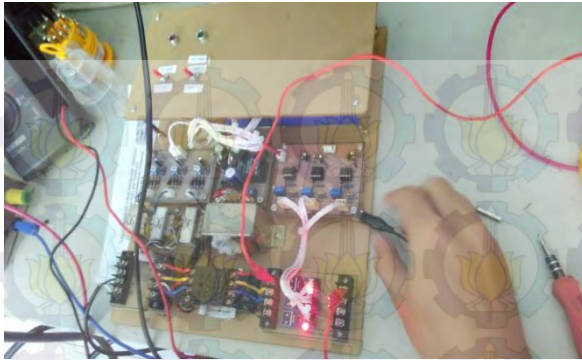
Saat pengujian sensor Arus ACS712





Saat Pengujian Sensor Tegangan





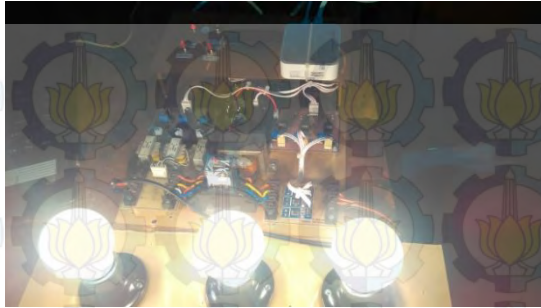
Saat pengujian Alat Dalam Kondisi Normal



Bagian Panel Local



Pengujian Breaker Terhubung



Breaker Saat Terputus



Saat terhubung dengan arus listrik dalam kondisi normal dan status local



LAMPIRAN B PROGRAM

/**

Project : Tugas Akhir

Author : Ifa Fitrianti (2213039001)

Author 2 : Ilham Sudrajat (2213039006)

Counseller : Suwito, S.T., M.T

Laboratory : Elektronika Terapan Sepuluh Nopember

Company : Institut Teknologi Sepuluh Nopember

Program type : Intel Galileo gen 2

*/

```
var mraa = require('mraa');
console.log('MRAA Version: ' + mraa.getVersion());
var PUBNUB = require('pubnub');
var pubnub = PUBNUB.init({
  publish_key: 'demo',
  subscribe_key: 'demo'
});
```

```
function sleep(delay)
{
  delay += new Date().getTime();
  while (new Date() < delay) { }
}
```

```
var analog = new mraa.Aio(1);

var cbo = new mraa.Gpio(4);
cbo.dir(mraa.DIR_OUT);
var cbstats = new mraa.Gpio(3);
cbstats.dir(mraa.DIR_IN);
var opstats = new mraa.Gpio(6);
opstats.dir(mraa.DIR_IN);
var buzzer = new mraa.Gpio(7);
buzzer.dir(mraa.DIR_OUT);
```

```

var breaker = new mraa.Gpio(5);
breaker.dir(mraa.DIR_OUT);
var x = new mraa.Gpio(8);
x.dir(mraa.DIR_OUT);
var y = new mraa.Gpio(9);
y.dir(mraa.DIR_OUT);
var z = new mraa.Gpio(10);
z.dir(mraa.DIR_OUT);

var lastValue = 0;
var ir=0,is=0,it=0;

function cekvr(){
  x.write(0);
  y.write(0);
  z.write(0);
  var analogValue = analog.read();
  var max = 1023;
  var percent =
(51.914523965309+(0.34195982347946*analogValue)).toFixed(0);
  console.log(percent);
  if(percent !== lastValue) {
    pubnub.publish({
      channel: 'sdis',
      message: {"vr" : percent}
    });
    lastValue = percent;
  }
}

function cekvs(){
  x.write(1);
  y.write(0);
  z.write(0);
  var analogValue = analog.read();
  var max = 1023;
  var percent =
(52.477566640274+(0.33979941937187*analogValue)).toFixed(0);
  console.log(percent);
}

```

```

if(percent !== lastValue) {
  pubnub.publish({
    channel: 'sdis',
    message: {"vs" : percent}
  });
  lastValue = percent;
}
}

function cekvt(){
  x.write(0);
  y.write(1);
  z.write(0);
  var analogValue = analog.read();
  var max = 1023;
  var
                                percent
(52.390888963028+(0.3402990506809*analogValue)).toFixed(0);
  console.log(percent);
  if(percent !== lastValue) {
    pubnub.publish({
      channel: 'sdis',
      message: {"vt" : percent}
    });
    lastValue = percent;
  }
}

function cekir(){
  x.write(1);
  y.write(1);
  z.write(0);
  var analogValue = analog.read();
  var max = 1023;
  var ir = (((0.010533644973698*analogValue)-2.2573771117574)-
5).toFixed(1);
  console.log(ir);
  if(ir !== lastValue) {
    pubnub.publish({
      channel: 'sdis',

```

```

        message: {"ir" : ir}
    });
    lastValue = ir;
}
}

function cekis(){
    x.write(0);
    y.write(0);
    z.write(1);
    var analogValue = analog.read();
    var max = 1023;
    var is = (((0.010309615279189*analogValue)-2.1665067704621)-
8).toFixed(1);
    console.log(is);
    if(is !== lastValue) {
        pubnub.publish({
            channel: 'sdis',
            message: {"is" : is}
        });
        lastValue = is;
    }
}

function cekit(){
    x.write(1);
    y.write(0);
    z.write(1);
    var analogValue = analog.read();
    var max = 1023;
    var it = (((0.0098276955622793*analogValue)-2.1038286574932)-
5.6).toFixed(1);
    console.log(it);
    if(it !== lastValue) {
        pubnub.publish({
            channel: 'sdis',
            message: {"it" : it}
        });
        lastValue = it;
    }
}

```

```
}  
}
```

```
function cekcb(){  
  var cb = cbstats.read();  
  console.log(cb);  
  pubnub.publish({  
    channel: 'sdis',  
    message: {"breaker" : cb}  
  });  
}
```

```
function cekop(){  
  var op = opstats.read();  
  console.log(op);  
  pubnub.publish({  
    channel: 'sdis',  
    message: {"operation" : op}  
  });  
}
```

```
function sirine(cond){  
  if(cond == "1"){  
    buzzer.write(1);  
    console.log( "> Alarm On" );  
    pubnub.publish({  
      channel: 'sdis',  
      message: {"buzz" : "on"}  
    });  
  }else if(cond=="0"){  
    buzzer.write(0);  
    console.log( "> Alarm Off" );  
    pubnub.publish({  
      channel: 'sdis',  
      message: {"buzz" : "off"}  
    });  
  }  
}
```

```

pubnub.subscribe({
  channel : "sdissend",
  callback : function(message) {
    if(message=="1"){
      breaker.write(1);
      console.log( " > Terhubung " );
    }
    if(message=="2"){
      breaker.write(0);
      console.log( " > Terputus " );
    }
    if(message=="3"){
      sirine(1);
    }
    if(message=="4"){
      sirine(0);
    }
  }
});

```

```

function cek(){
  cekvr();
  sleep(500);
  cekvs();
  sleep(500);
  cekvt();
  sleep(500);
  cekir();
  sleep(500);
  cekis();
  sleep(500);
  cekit();
  sleep(500);
  cekcb();
  sleep(500);
  cekop();
  sleep(500);
  if(ir >= 3){

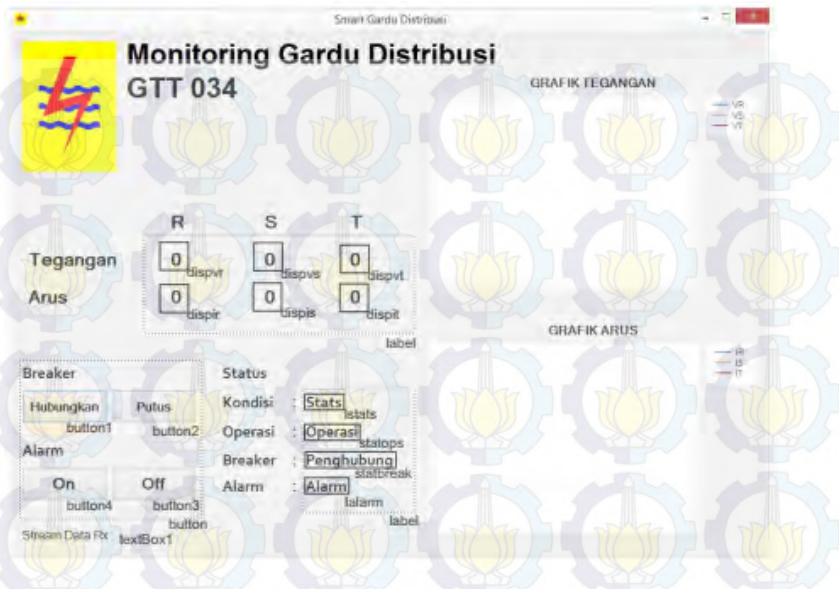
```

```

cbo.write(0);
sirine(1);
pubnub.publish({
    channel: 'sdis',
    message: {"troub" : "a"}
});
} else if(is>=3){
    cbo.write(0);
    sirine(1);
    pubnub.publish({
        channel: 'sdis',
        message: {"troub" : "b"}
    });
} else if(it>=3){
    cbo.write(0);
    sirine(1);
    pubnub.publish({
        channel: 'sdis',
        message: {"troub" : "c"}
    });
} else{
    cbo.write(1);
    pubnub.publish({
        channel: 'sdis',
        message: {"troub" : "d"}
    });
}
//cbo.write(1);
console.log("");
}

setInterval(cek, 1000);

```



/******

Program : Visual C#

*****/

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading;
using System.Windows.Forms;
using PubNubMessaging.Core;
using System.Windows.Forms.DataVisualization.Charting;
```

```
namespace VisualTA
{
    public partial class Form1 : Form
    {
        Pubnub _pubnub;
```

```

const string CHANNELNAME = "sdis";
const string CHANNELSEND = "sdissend";
const string PUBLISH_KEY = "demo";
const string SUBSCRIBE_KEY = "demo";
const string SECRET_KEY = "";
delegate void SetTextCallback(string text);
double i = 0;
double data;
public Form1()
{
    InitializeComponent();
}
private void Form1_Load(object sender, EventArgs e)
{
    Init();
    Subscribe();
    //this.chart1.Titles.Add("ARUS");
}
private void timer1_Tick(object sender, EventArgs e)
{
    i=i++;
}
void Init()
{
    Console.WriteLine("Running Init()");
    _pubnub = new Pubnub(
        PUBLISH_KEY,
        SUBSCRIBE_KEY
    );
}
void Subscribe()
{
    _pubnub.Subscribe<string>(CHANNELNAME,
        DisplaySubscribeReturnMessage, DisplayConnectStatusMessage,
        DisplaySubscribeErrorMessage);
}
void DisplaySubscribeReturnMessage(string result)
{
    Console.WriteLine("Returned:");
}

```

```

    tampil(result);
    var _subsc = result.Split(new[] { '{', '}' },
StringSplitOptions.RemoveEmptyEntries);
    var _key = _subsc[1].Split(new[] { ':' },
StringSplitOptions.RemoveEmptyEntries);
    _key[0] = _key[0].Replace("\\", String.Empty);
    _key[1] = _key[1].Replace("\\", String.Empty);
    if (_key[0].Trim() == "vr")
    {
        SetVR(_key[1].Trim());
    }
    else if (_key[0].Trim() == "vs")
    {
        SetVS(_key[1].Trim());
    }
    else if (_key[0].Trim() == "vt")
    {
        SetVT(_key[1].Trim());
    }
    else if (_key[0].Trim() == "ir")
    {
        SetIR(_key[1].Trim());
    }
    else if (_key[0].Trim() == "is")
    {
        SetIS(_key[1].Trim());
    }
    else if (_key[0].Trim() == "it")
    {
        SetIT(_key[1].Trim());
    }
    else if (_key[0].Trim() == "breaker")
    {
        if (_key[1].Trim() == "1")
        {
            breaker("1");
        }
        else if (_key[1].Trim() == "0")
        {

```

```

        breaker("0");
    }
}
else if (_key[0].Trim() == "operation")
{
    if (_key[1].Trim() == "1")
    {
        operation("1");
    }
    else if (_key[1].Trim() == "0")
    {
        operation("0");
    }
}
else if (_key[0].Trim() == "buzz")
{
    if (_key[1].Trim() == "on")
    {
        buzzer("on");
    }
    else if (_key[1].Trim() == "off")
    {
        buzzer("off");
    }
}
else if (_key[0].Trim() == "troub")
{
    trouble(_key[1].Trim());
}
}
private void tampil(string text)
{
    if (this.textBox1.InvokeRequired)
    {
        SetTextCallback d = new SetTextCallback(tampil);
        this.Invoke(d, new object[] { text });
    }
    else
    {

```

```

        this.textBox1.Text = text;
    }
}
private void SetVR(string text)
{
    // InvokeRequired required compares the thread ID of the
    // calling thread to the thread ID of the creating thread.
    // If these threads are different, it returns true.
    if (this.dispvr.InvokeRequired)
    {
        SetTextCallback d = new SetTextCallback(SetVR);
        this.Invoke(d, new object[] { text });
    }
    else
    {
        this.dispvr.Text = text;
    }
}
private void SetVS(string text)
{
    // InvokeRequired required compares the thread ID of the
    // calling thread to the thread ID of the creating thread.
    // If these threads are different, it returns true.
    if (this.dispvs.InvokeRequired)
    {
        SetTextCallback d = new SetTextCallback(SetVS);
        this.Invoke(d, new object[] { text });
    }
    else
    {
        this.dispvs.Text = text;
    }
}
private void SetVT(string text)
{
    // InvokeRequired required compares the thread ID of the
    // calling thread to the thread ID of the creating thread.
    // If these threads are different, it returns true.
    if (this.dispvt.InvokeRequired)

```

```

    {
        SetTextCallback d = new SetTextCallback(SetVT);
        this.Invoke(d, new object[] { text });
    }
    else
    {
        this.dispvt.Text = text;
    }
}
private void SetIR(string text)
{
    // InvokeRequired required compares the thread ID of the
    // calling thread to the thread ID of the creating thread.
    // If these threads are different, it returns true.
    data = Convert.ToDouble(text);
    //this.chart1.Series["Series 1"].Points.AddXY(i,data);
    if (this.dispir.InvokeRequired)
    {
        SetTextCallback d = new SetTextCallback(SetIR);
        this.Invoke(d, new object[] { text });
    }
    else
    {
        this.dispir.Text = text;
    }
}
private void SetIS(string text)
{
    // InvokeRequired required compares the thread ID of the
    // calling thread to the thread ID of the creating thread.
    // If these threads are different, it returns true.
    if (this.dispis.InvokeRequired)
    {
        SetTextCallback d = new SetTextCallback(SetIS);
        this.Invoke(d, new object[] { text });
    }
    else
    {
        this.dispis.Text = text;
    }
}

```

```

    }
}
private void SetIT(string text)
{
    // InvokeRequired required compares the thread ID of the
    // calling thread to the thread ID of the creating thread.
    // If these threads are different, it returns true.
    if (this.dispit.InvokeRequired)
    {
        SetTextCallback d = new SetTextCallback(SetIT);
        this.Invoke(d, new object[] { text });
    }
    else
    {
        this.dispit.Text = text;
    }
}
private void breaker(string text)
{
    if (this.statbreak.InvokeRequired)
    {
        SetTextCallback d = new SetTextCallback(breaker);
        this.Invoke(d, new object[] { text });
    }
    else
    {
        if (text == "1")
        {
            this.statbreak.Text = "Terhubung";
        }
        else if (text == "0")
        {
            this.statbreak.Text = "Terputus";
        }
    }
}
private void operation(string text)
{
    // InvokeRequired required compares the thread ID of the

```

```

// calling thread to the thread ID of the creating thread.
// If these threads are different, it returns true.
if (this.statops.InvokeRequired)
{
    SetTextCallback d = new SetTextCallback(operation);
    this.Invoke(d, new object[] { text });
}
else
{
    if (text == "1")
    {
        this.statops.Text = "Local";
    }
    else if (text == "0")
    {
        this.statops.Text = "Remote";
    }
}
}
private void buzzer(string text)
{
    // InvokeRequired required compares the thread ID of the
    // calling thread to the thread ID of the creating thread.
    // If these threads are different, it returns true.
    if (this.lalarm.InvokeRequired)
    {
        SetTextCallback d = new SetTextCallback(buzzer);
        this.Invoke(d, new object[] { text });
    }
    else
    {
        if (text == "on")
        {
            this.lalarm.Text = "Menyala";
            System.Media.SystemSounds.Hand.Play();
        }
        else if (text == "off")
        {
            this.lalarm.Text = "Mati";
        }
    }
}

```

```

    }
}
private void trouble(string text)
{
    // InvokeRequired required compares the thread ID of the
    // calling thread to the thread ID of the creating thread.
    // If these threads are different, it returns true.
    if (this.Istats.InvokeRequired)
    {
        SetTextCallback d = new SetTextCallback(trouble);
        this.Invoke(d, new object[] { text });
    }
    else
    {
        if (text == "a")
        {
            this.Istats.Text = "OC Fasa R";
        }
        else if (text == "b")
        {
            this.Istats.Text = "OC Fasa S";
        }
        else if (text == "c")
        {
            this.Istats.Text = "OC Fasa T";
        }
        else if (text == "d")
        {
            this.Istats.Text = "Normal";
        }
    }
}
void DisplayConnectStatusMessage(string result)
{
    Console.WriteLine("ConnectStatus:");
    Console.WriteLine(result);
}

```

```

void DisplaySubscribeErrorMessage(PubnubClientError
pubnubError)
{
    Console.WriteLine(pubnubError.StatusCode);

    //Based on the severity of the error, we can filter out errors for
    handling or logging.
    switch (pubnubError.Severity)
    {
        case PubnubErrorSeverity.Critical:
            //This type of error needs to be handled.
            break;
        case PubnubErrorSeverity.Warn:
            //This type of error needs to be handled
            break;
        case PubnubErrorSeverity.Info:
            //This type of error can be ignored
            break;
        default:
            break;
    }

    Console.WriteLine(pubnubError.StatusCode); //Unique ID of the
error

    Console.WriteLine(pubnubError.Message); //Message received
from client or server. From client, it could be from .NET exception.

    if (pubnubError.DetailedDotNetException != null)
    {
        Console.WriteLine(pubnubError.IsDotNetException); //
Boolean flag to check .NET exception

        Console.WriteLine(pubnubError.DetailedDotNetException.ToString());
// Full Details of .NET exception
    }

    Console.WriteLine(pubnubError.MessageSource); // Did this
originate from Server or Client-side logic

```

```

        if (pubnubError.PubnubWebRequest != null)
        {
            //Captured Web Request details
            Console.WriteLine(pubnubError.PubnubWebRequest.RequestUri.ToString());
            Console.WriteLine(pubnubError.PubnubWebRequest.Headers.ToString());
        }

        if (pubnubError.PubnubWebResponse != null)
        {
            //Captured Web Response details
            Console.WriteLine(pubnubError.PubnubWebResponse.Headers.ToString());
        }

        Console.WriteLine(pubnubError.Description); // Useful for
        logging and troubleshooting and support
        Console.WriteLine(pubnubError.Channel); //Channel name(s) at
        the time of error
        Console.WriteLine(pubnubError.ErrorDateTimeGMT); //GMT
        time of error
    }
    void DisplayReturnMessage(string result)
    {
        Console.WriteLine("PUBLISH STATUS CALLBACK");
        Console.WriteLine(result);
    }
    void DisplayErrorMessage(PubnubClientError pubnubError)
    {
        Console.WriteLine(pubnubError.StatusCode);
    }
    private void button1_Click(object sender, EventArgs e)
    {
        _pubnub.Publish<string>(<

```

```
CHANNELSEND,  
1,  
DisplayReturnMessage,  
DisplayErrorMessage  
);  
}
```

```
private void button2_Click(object sender, EventArgs e)  
{  
    _pubnub.Publish<string>(  
        CHANNELSEND,  
        2,  
        DisplayReturnMessage,  
        DisplayErrorMessage  
    );  
}
```

```
private void button4_Click(object sender, EventArgs e)  
{  
    _pubnub.Publish<string>(  
        CHANNELSEND,  
        3,  
        DisplayReturnMessage,  
        DisplayErrorMessage  
    );  
}
```

```
private void button3_Click(object sender, EventArgs e)  
{  
    _pubnub.Publish<string>(  
        CHANNELSEND,  
        4,  
        DisplayReturnMessage,  
        DisplayErrorMessage  
    );  
}
```



LAMPIRAN C

DATASHEET

1. Datasheet Intel Galileo Gen 2



Galileo Datasheet



Galileo Front



Galileo Back

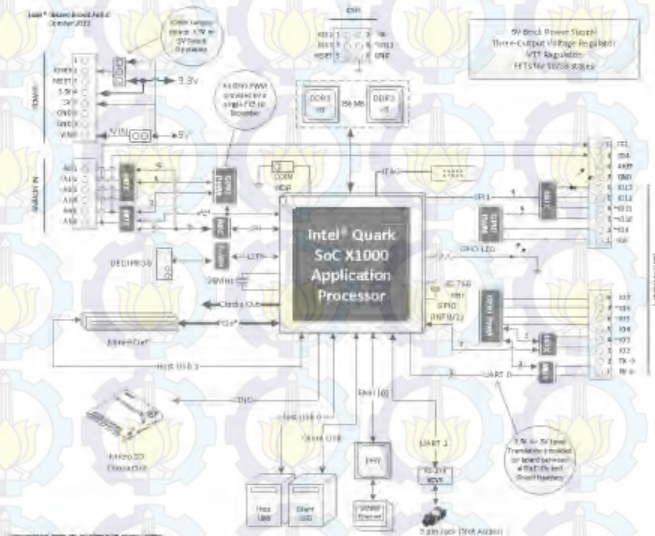
Overview

Galileo is a microcontroller board based on the Intel® Quark SoC X1000 Application Processor, a 32-bit Intel Pentium-class system on a chip ([datasheet](#)). It is the first board based on Intel® architecture designed to be hardware and software pin-compatible with Arduino shields designed for the Uno R3. Digital pins 0 to 13 (and the adjacent AREF and GND pins), Analog inputs 0 to 5, the power header, ICSP header, and the UART port pins (0 and 1), are all in the same locations as on the Arduino Uno R3. This is also known as the Arduino 1.0 pinout.

Galileo is designed to support shields that operate at either 3.3V or 5V. The core operating voltage of Galileo is 3.3V. However, a jumper on the board enables voltage translation to 5V at the I/O pins. This provides support for 5V Uno shields and is the default behavior. By switching the jumper position, the voltage translation can be disabled to provide 3.3V operation at the I/O pins.

Of course, the Galileo board is also SW compatible with the Arduino SW Development Environment, which makes usability and introduction a snap.

In addition to Arduino HW and SW compatibility, the Galileo board has several PC industry standard I/O ports and features to expand native usage and capabilities beyond the Arduino shield ecosystem. A full sized mini-PCI Express® slot, 100Mb Ethernet port, Micro-SD slot, RS-232 serial port, USB Host port, USB Client port, and 8MB NOR flash come standard on the board.



Getting Started

To get started, simply connect the board to power with the 5V AC-to-DC adapter and then connect to the computer with the micro-USB cable.

Galileo has a [dedicated forum](#) for discussing the board.

Details and Specifications

Arduino Shield Supported Features

Galileo is compatible with Arduino Uno shields and is designed to support 3.3V or 5V shields, following the Arduino Uno Revision 3, including:

- **14 digital input/output pins**, of which 6 can be used as Pulse Width Modulation (PWM) outputs;
 - Each of the 14 digital pins on Galileo can be used as an input or output, using `pinMode()`, `digitalWrite()`, and `digitalRead()` functions.
 - The pins operate at 3.3 volts or 5 volts. Each pin can source a max of 10mA or sink a maximum of 25 mA and has an internal pull-up resistor (disconnected by default) of 5.6k to 10 kOhms.
- **A0 - A5 - 6 analog inputs**, via an AD7298 analog-to-digital (A/D) converter ([datasheet](#))
 - Each of the 6 analog inputs, labeled A0 through A5, provides 12 bits of resolution (i.e., 4096 different values). By default they measure from ground to 5 volts.
- **I²C* bus, TWI**, with SDA and SCL pins that are near to the AREF pin.
 - TWI: A4 or SDA pin and A5 or SCL pin. Support TWI communication using the [Wire library](#).
- **SPI**
 - Defaults to 4MHz to support Arduino Uno shields. Programmable up to 25MHz.

- **Note:** While Galileo has a native SPI controller, it will act as a master and not as an SPI slave. Therefore, Galileo cannot be a SPI slave to another SPI master. It can act, however, as a slave device via the USB Client connector.
- **UART** (serial port) Programmable speed UART port (Pins 0 (RX) and 1 (TX))
- **ICSP (SPI)** - a 6 pin in-circuit serial programming (ICSP) header, located appropriately to plug into existing shields. These pins support SPI communication using the SPI library.
- **VIN.** The input voltage to the Galileo board when it's using an external power source (as opposed to 5 volts from the regulated power supply connected at the power jack). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
 - **Warning:** The voltage applied to this pin must be a regulated 5V supply otherwise it could damage the Galileo board or cause incorrect operation.
- **5V output pin.** This pin outputs 5V from the external source or the USB connector. Maximum current draw to the shield is 800 mA
- **3.3V output pin.** A 3.3 volt supply generated by the on-board regulator. Maximum current draw to the shield is 800 mA
- **GND.** Ground pins.
- **IOREF.** The IOREF pin on Galileo allows an attached shield with the proper configuration to adapt to the voltage provided by the board. The IOREF pin voltage is controlled by a jumper on the board, i.e., a selection jumper on the board is used to select between 3.3V and 5V shield operation.
- **RESET** button/pin
 - Bring this line LOW to reset the sketch. Typically used to add a reset button to shields that block the one on the board.
- **AREF** is unused on Galileo. Providing an external reference voltage for the analog inputs is not supported.
 - For Galileo it is not possible to change the upper end of the analog input range using the AREF pin and the `analogReference()` function.

Details of Intel® Architecture Supported Features

The genuine Intel processor and surrounding native I/O capabilities of the SoC provides for a fully featured offering for both the maker community and students alike. It will also be useful to professional developers who are looking for a more simple and cost effective development environment to the more complex Intel® Atom processor and Intel® Core processor-based designs.

- 400MHz 32-bit Intel® Pentium instruction set architecture (ISA)-compatible processor
 - 16 KByte L1 cache
 - 512 KBytes of on-die embedded SRAM
 - Simple to program: Single thread, single core, constant speed
 - ACPI compatible CPU sleep states supported
 - An integrated Real Time Clock (RTC), with an optional 3V "coin cell" battery for operation between turn on cycles.
- 10/100 Ethernet connector
- Full PCI Express* mini-card slot, with PCIe* 2.0 compliant features
 - Works with half mini-PCIe* cards with optional converter plate
 - Provides USB 2.0 Host Port at mini-PCIe* connector
- USB 2.0 Host connector
 - Support up to 128 USB end point devices
- USB Client connector, used for programming
 - Beyond just a programming port - a fully compliant USB 2.0 Device controller
- 10-pin Standard JTAG header for debugging
- Reboot button to reboot the processor
- Reset button to reset the sketch and any attached shields
- Storage options:
 - 8 MByte Legacy SPI Flash whose main purpose is to store the firmware (or bootloader) and the latest sketch. Between 256 KByte and 512 KByte is dedicated for sketch storage. The upload happens automatically from the development PC, so no action is required unless there is an upgrade that is being added to the firmware.

- 512 KByte embedded SRAM that is enabled by the firmware by default.
- 256 MByte DRAM, enabled by the firmware by default.
- Optional micro SD card offers up to 32GByte of storage
- USB storage works with any USB 2.0 compatible drive
- 11 KByte EEPROM can be programmed via the [EEPROM library](#).

Schematic, Reference Design & Pin Mapping

- Schematic: [Galileo-schematic.pdf](#)
- Cadence® Allegro® files: [Galileo-reference-design.zip](#)

Power

Galileo is powered via an AC-to-DC adapter, connected by plugging a 2.1mm center-positive plug into the board's power jack. The recommended output rating of the power adapter is 5V at up to 3A.

Electrical Summary

Input Voltage (recommended)	5V
Input Voltage (limits)	5V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
Total DC Output Current on all I/O lines	80 mA
DC Current for 3.3V Pin	800 mA
DC Current for 5V Pin	800 mA

Communication

Galileo has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers.

Galileo provides **UART** TTL (5V/3.3V) serial communication, which is available on digital pin 0 (RX) and 1 (TX). In addition, a second UART provides RS-232 support and is connected via a 3.5mm jack.

The **USB Client** ports allows for serial (CDC-ACM) communications over USB. This provides a serial connection to the Serial Monitor or other applications on your computer. It is also used to upload sketches to the board.

The **USB Host** port allows Galileo act as a USB Host for connected peripherals such as mice, keyboards, and smartphones. To use these features, see the [USBHost reference pages](#).

Galileo is the first Arduino board to provide a mini **PCI Express*** (mPCIe*) slot. This slot allows full size and half size (with adapter) mPCIe* modules to be connected to the board and also provides an additional USB Host port via the mPCIe* slot. Any standard mPCIe* module can be connected and used to provide applications such as WiFi, Bluetooth or Cellular connectivity. Initially, the Galileo mPCIe* slot provides support for the [WiFi Library](#). For additional information, see the [Intel® Galileo Getting Started Guide](#).

An **Ethernet RJ45 Connector** is provided to allow Galileo to connect to wired networks. Full support of on-board Ethernet interface is fully supported and does not require the use of the SPI interface like existing Arduino shields.

The onboard **microSD** card reader is accessible through the [SD Library](#). The communication between Galileo and the SD card is provided by an integrated SD controller and does not require the use of the SPI interface like other Arduino boards. The native SD interface runs at up to 50MHz depending on the class of card used.

The Arduino software includes a Wire library to simplify use of the **TWI/I²C*** bus; see the [documentation](#) for details.

For **SPI** communication, use the [SPI library](#).

Programming

Galileo can be programmed with the Arduino software ([download](#)). When you are ready to upload the sketch to the board, program Galileo through the USB Client port by selecting "Intel Galileo" as your board in the Arduino IDE. Connect Galileo's port labelled USB Client (the one closest to the Ethernet) to your computer. For details, see the [reference](#), [tutorials](#) and [Intel® Galileo Getting Started Guide](#).

Rather than requiring a physical press of the reset button before an upload, Galileo is designed to be reset by software running on a connected computer.

When the board boots up two scenarios are possible:

- If a sketch is present in persistent storage, it is executed.
- If no sketch present, the board waits for upload commands from the IDE.

If a sketch is executing, you can upload from the IDE without having to press the reset button on the board. The sketch is stopped; the IDE waits for the upload state, and then starts the newly uploaded sketch.

Pressing the reset button on the board restarts a sketch if it is executing and resets any attached shields.

Properties of Pins Configured as OUTPUT

Pins configured as OUTPUT with `pinMode()` are said to be in a low-impedance state. On Galileo, when a pin is configured as OUTPUT, the functionality is provided via an I²C*-based Cypress I/O expander ([datasheet](#)). Digital pins 0 to 13 and Analog pins A0 to A5 can be configured as OUTPUT pins on Galileo.

The I/O expander's pins, when configured as OUTPUT, can source (provide positive current) up to 10 mA (milliamps) and can sink (provide negative current) up to 25 mA of current to other devices/circuits. The individual per pin current sourcing capability of 10 mA is subject to an overall limit of 80 mA combined between all OUTPUT pins. The per pin capability current sinking capability is subject to an overall limit of 200 mA. The following table provides a breakdown of the overall OUTPUT capabilities of the pins.

	Current Source (mA)	Current Sink (mA)
Per Pin Capability	10	25
Digital Pins 3,5,9,10,12,13 Combined	40	100
Digital Pins 0,1,2,4,6,7,8,11 and Analog Pins A0,A1,A2,A3,A4, A5 Combined	40	100
Digital Pins 0-13 and Analog Pins A0-A5 Combined	80	200

Galileo Jumper Configuration

There are three jumpers on Galileo that are used to vary the configuration of the board.

IOREF Jumper

To allow Galileo support both 3.3V and 5V shields, the external operating voltage is controlled via a jumper. When the jumper is connected to 5V, Galileo is configured to be compatible with 5V shields and IOREF is set to 5V. When the jumper is connected 3.3V, Galileo is configured to be compatible with 3.3V shields and IOREF is set to 3.3V.

The input range of the Analog pins is also controlled by the IOREF jumper and must not exceed the chosen operating voltage. However, the resolution of `AnalogRead()` remains at 5 V/1024 units for the default 10-bit resolution or, 0.0049V (4.9mV) per unit regardless of IOREF jumper setting.

Warning: The IOREF jumper should be used to match the board and shield operating voltages. Incorrectly setting the voltage could damage the board or the shield.

I²C* Address Jumper

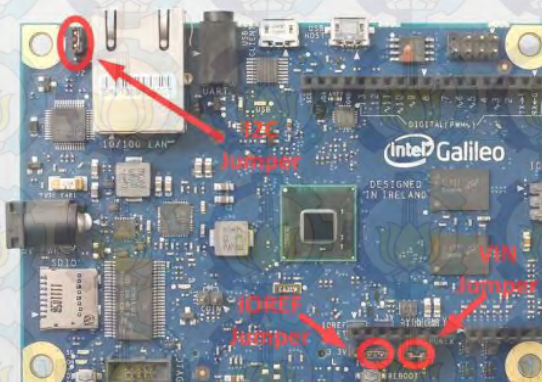
To prevent a clash between the I²C* Slave address of the on board I/O expander and EEPROM with any external I²C Slave devices, jumper J2 can be used to vary the I²C* address of the on-board devices.

With J2 connected to pin 1 (marked with white triangle), the 7-bit I/O Expander address is 0100001 and the 7-bit EEPROM address is 1010001. Changing the jumper position changes the I/O Expander address to 0100000 and the EEPROM address to 1010000.

VIN Jumper

On Galileo, the VIN pin can be used to supply 5V from the regulated power supply connected at the power jack to attached shields or devices. If there is a need to supply more than 5V to a shield using VIN then the VIN jumper should be removed from Galileo to break the connection between the on-board 5V supply and the VIN connection on the board header.

Warning: If the VIN jumper is not removed and more than 5V is connected to VIN, it may damage the board or lead to unreliable operation.





Automatic (Software) Reset

Rather than requiring a physical press of the reset button before an upload, Galileo is designed in a way that allows it to be reset by software running on a connected computer. USB CDC-ACM control signals are used to transition Galileo from run-time to bootloader mode. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. For details, see the [Intel® Galileo Getting Started Guide](#).

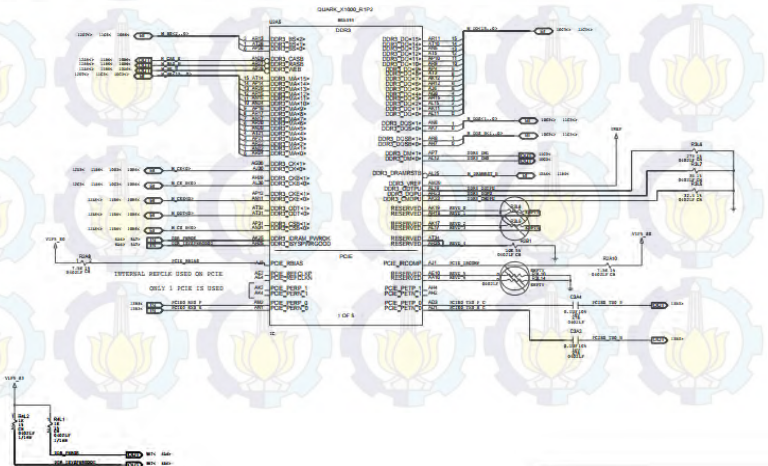
Physical Characteristics

Galileo is 4.2 inches long and 2.8 inches wide respectively, with the USB connectors, UART jack, Ethernet connector, and power jack extending beyond the former dimension. Four screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), is not an even multiple of the 100 mil spacing of the other pins.

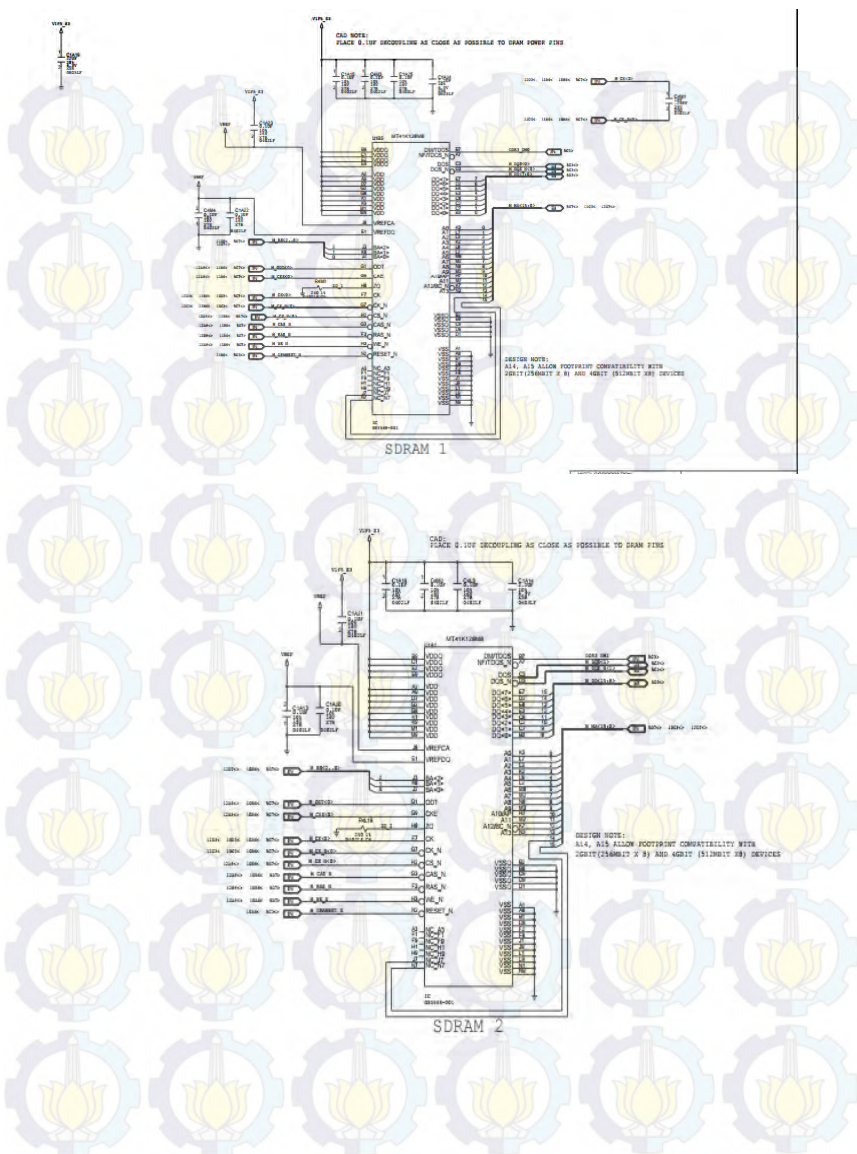
Intel® Galileo Design Document

This Intel® Galileo design document is licensed by Intel under the terms of the Creative Commons Attribution Share-Alike License (ver. 3), subject to the following terms and conditions. The Intel® Galileo design document IS PROVIDED "AS IS" AND "WITH ALL FAULTS." Intel DISCLAIMS ALL OTHER WARRANTIES, EXPRESS OR IMPLIED REGARDING THE GALILEO DESIGN OR THIS GALILEO DESIGN DOCUMENT INCLUDING, BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Intel® may make changes to the specifications, schematics and product descriptions at any time, without notice. The Customer must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel® reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. ENJOY!
*Other names and brands may be claimed as the property of others.

07 Oct 2013



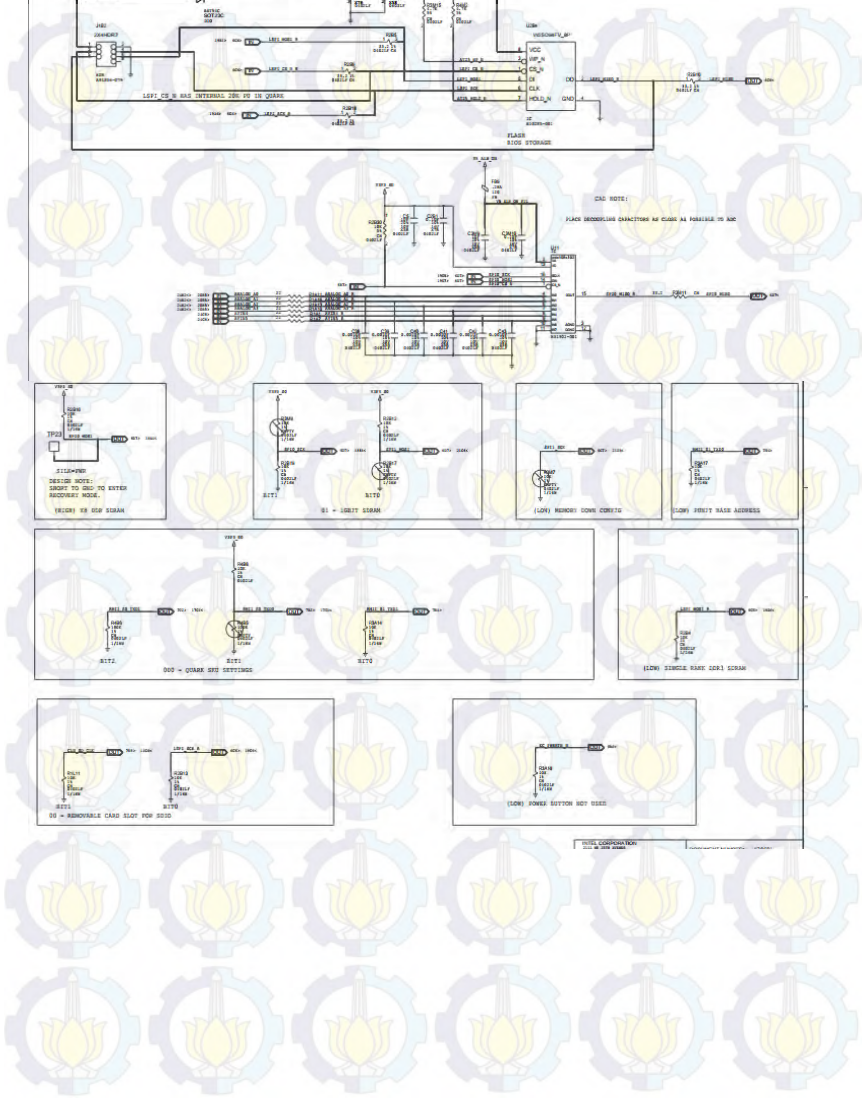


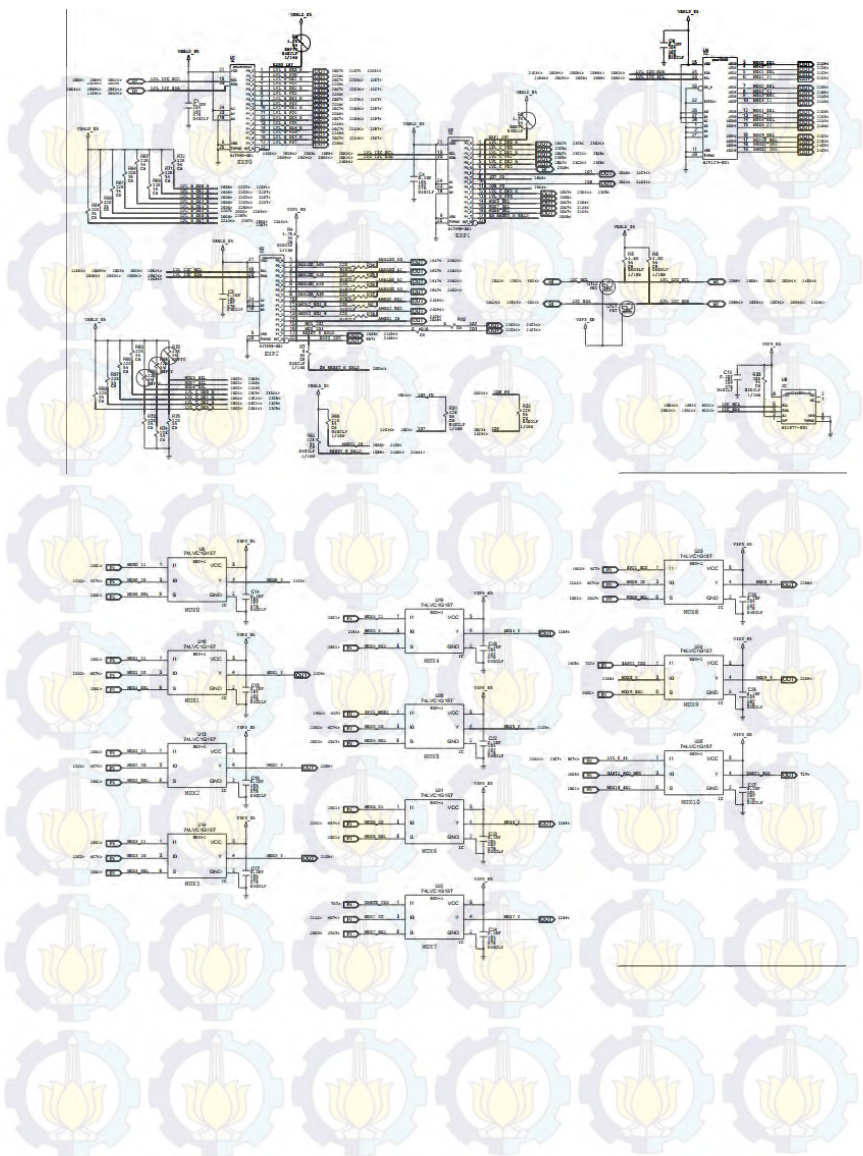




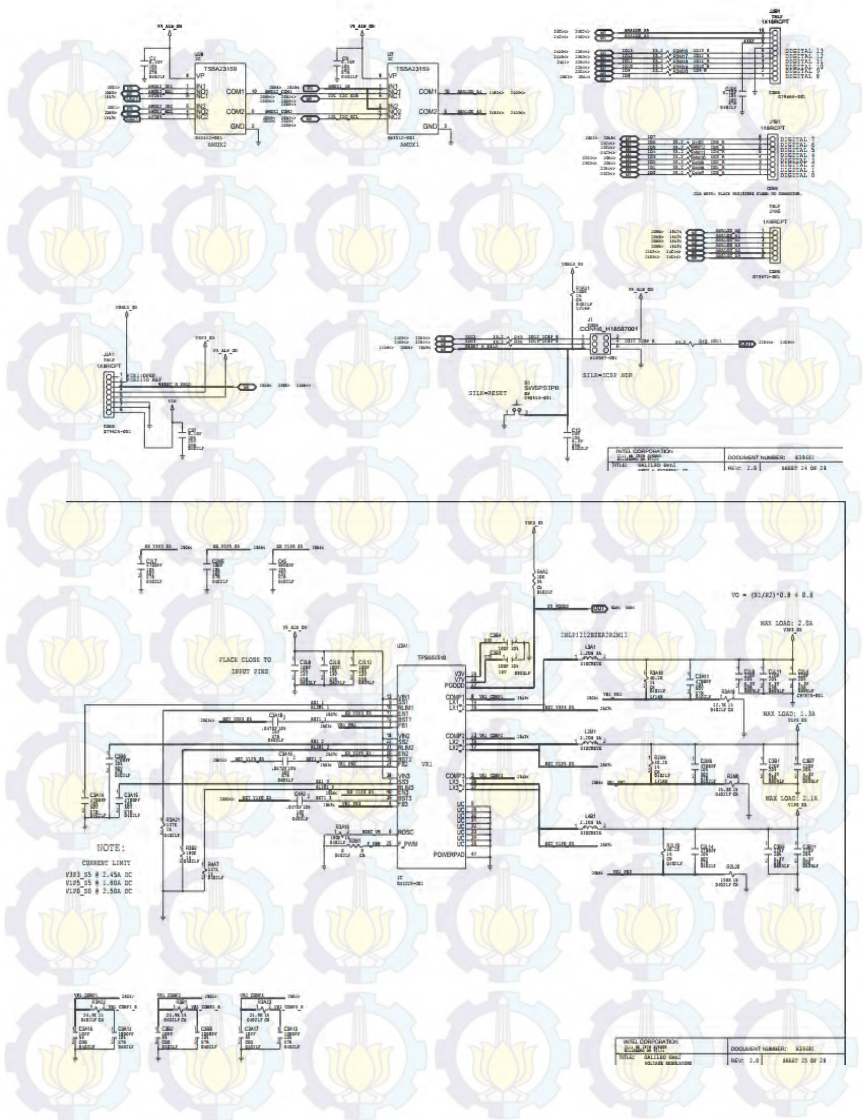


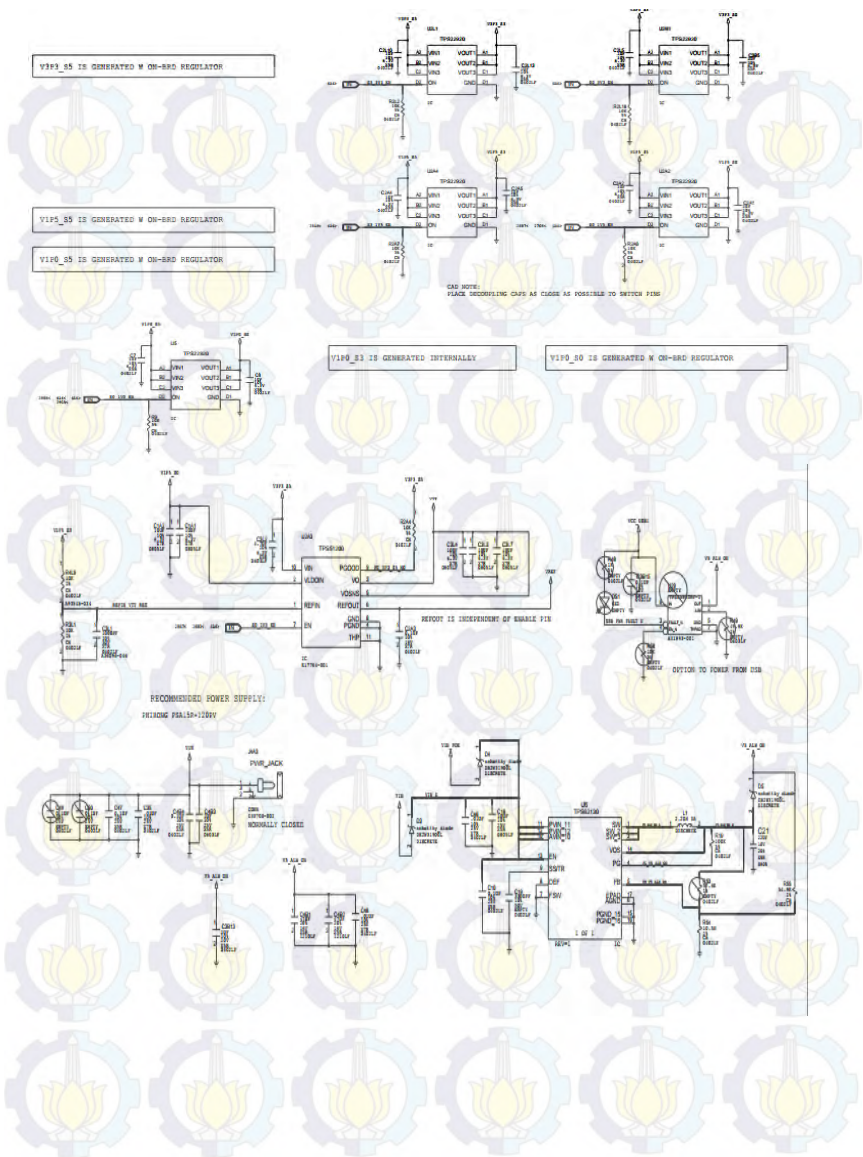














DAFTAR RIWAYAT HIDUP



Nama : Ifa Fitrianti
TTL : Tuban, 12 September 1995
Jenis Kelamin : Perempuan
Agama : Islam
Alamat : Desa Suco, Kec.
Mumbulsari, Kab. Jember
Telp/HP : 089678422558
E-mail : ifafitrianti@gmail.com

RIWAYAT PENDIDIKAN

1. 1998 – 2001 : Tk Bakti Pertiwi Tuban
2. 2001 – 2007 : SD Negeri Suco 5 Jember
3. 2007 – 2010 : MTS Syalafia-Syafiiyah Jember
4. 2010 – 2013 : SMA Negeri 5 Tuban
5. 2013 – 2016 : D3 Teknik Elektro, Program Studi Teknik Elektro Industri - FTI Institut Teknologi Sepuluh Nopember (ITS)

PENGALAMAN KERJA

1. Kerja Praktek di SIER Rungkut Industri, Surabaya, Jawa Timur
2. Kerja Praktek di PT PLN (Persero) APJ Surabaya Selatan, Jawa Timur

PENGALAMAN ORGANISASI

1. Staff Departemen Hubungan Luar Periode 2014/2015
HIMAD3TEKTRO, FTI - ITS



DAFTAR RIWAYAT HIDUP



Nama : Ilham Sudrajat
TTL : Madiun , 11 April 1995
Jenis Kelamin : Laki - Laki
Agama : Islam
Alamat : Desa. Garon, Kec.
Balerejo Kab.Madiun
Telp/HP : 085784849500
E-mail : ilhamimst@gmail.com

RIWAYAT PENDIDIKAN

1. 1998 – 2001 : TK Garon 2
2. 2001 – 2007 : SD Negeri Garon 2 Madiun
3. 2007 – 2010 : SMP Negeri 3 Madiun
4. 2010 – 2013 : SMA Negeri 5 Madiun
5. 2013 – 2016 : D3 Teknik Elektro, Program Studi Teknik Elektro Industri - FTI Institut Teknologi Sepuluh Nopember (ITS)

PENGALAMAN KERJA

1. Kerja Praktik di PT.PLN(Persero) Area Madiun Rayon Caruban

PENGALAMAN ORGANISASI

1. Team Suro Electric Car D3 Teknik Elektro ITS.

