TUGAS AKHIR - TE 141599

PENERAPAN KOMPRESI DATA UNTUK EFISIENSI DAYA JARINGAN SENSOR NIRKABEL

MUHAMMAD NAUFAL FARID
NRP 2211 100 162

Dosen Pembimbing
Dr. Ir. Wirawan, DEA.
Dr. Istas Pratomo, ST, MT

JURUSAN TEKNIK ELEKTRO
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya 2016

FINAL PROJECT – TE 141599

IMPLEMENTATION OF DATA COMPRESSION FOR WIRELESS SENSOR NETWORK POWER EFICIENCY

MUHAMMAD NAUFAL FARID
NRP 2211 100 162

Supervisors
Dr. Ir. Wirawan, DEA.
Dr. Istas Pratomo, ST, MT

DEPARTMENT OF ELECTRICAL ENGINEERING
Faculty of Industrial Technology
Sepuluh Nopember Institute of Technology
Surabaya 2016

PENERAPAN KOMPRESI DATA UNTUK EFISIENSI DAYA JARINGAN SENSOR NIRKABEL

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan Untuk
Memperoleh Gelar Sarjana Teknik Elektro
Pada
Bidang Studi Telekomunikasi Multimedia
Jurusan Teknik Elektro
Institut Teknologi Sepuluh Nopember

Menyetujui

Dosen Pembimbing I,	Dosen Pembimbing II,
Dr. Ir. Wirawan, DEA.	**Dr. Istas Pratomo, ST, MT**
NIP. 196311091989031011	**NIP. 197903252003121001**

SURABAYA
JANUARI, 2016

PENERAPAN KOMPRESI DATA UNTUK EFISIENSI DAYA JARINGAN SENSOR NIRKABEL

Muhamad Naufal Farid
2211 100 162

Dosen Pembimbing I : Dr. Ir. Wirawan, DEA.
Dosen Pembimbing II : Dr. Istas Pratomo, ST., MT.

ABSTRAK

Jaringan sensor nirkabel (WSN) biasanya terdiri dari alat yang ringkas, rendah energi dan mampu mempertahankan berbagai macam topologi *ad-hoc* dan beroperasi untuk waktu yang lama juga biasanya menggunakan catu daya yang terbatas dan tidak dapat terbarukan. Batasan pada daya inilah yang menuntun ke banyak penelitian pada peminimalan pemakaian sumber daya baik pada hardware ataupun software. Transmisi data menghabiskan daya paling banyak pada peralatan WSN dan peneliti memutuskan untuk menyelidiki beberapa teknik kompresi pada software yang cocok untuk hardware yang memiliki sumber daya yang terbatas, dengan tujuan untuk meminimalisir daya yang dipakai untuk mengirimkan data. Daya yang nantinya akan dipakai setelah dilakukan metode kompresi tersebut nantinya akan dibandingkan dengan daya yang belum dilakukan kompresi apapun. Dikarenakan keterbatasan peralatan yang dipakai, metode kompresi yang nantinya akan digunakan adalah dengan menggunakan kompresi yang sederhana. Nantinya pemakaian daya kedua metode ini akan dibandingkan dengan pemakaian daya peralatan WSN yang belum diberikan metode kompresi. Peralatan yang digunakan adalah Arduino Leonardo sebagai mikrokontroler, X-bee sebagai antena dan peralatan lainnya sebagai penunjang. Rasio kompresi yang dapat dicapai menggunakan *look-up table* adalah 2.3333:1 dan menggunakan algoritma huffman adalah Min 1:1 & Max. 3.5:1. Sedangkan algoritma *distributed consensus* 2:1. Hasil efisiensi penggunaan daya rata-rata setelah dilakukan kompresi adalah 7.23903753% jika menggunakan *look-up table* dan 13.44170648% jika menggunakan algoritma huffman Sedangkan untuk *distributed consensus* dapat menghemat 11.92%.

Kata Kunci :kompresi, *Huffman code*,WSN,daya yang terpakai

[Halaman ini sengaja dikosongkan]

IMPLEMENTATION OF DATA COMPRESSION FOR WIRELESS SENSOR NETWORK POWER EFICIENCY

Muhammad Naufal Farid
2211 100 162

Supervisor I : Dr. Ir. Wirawan, DEA.
Supervisor II : Dr. Istas Pratomo, ST., MT.

ABSTRACT

Wireless sensor network (WSN)generaly contains compact, low-energy and can maintain various ad-hoc topology and operating for many years using limited energy. This problem then leads to many research on minimalization of energy consumed either in hardware or in software. Data transmition is the most energy-consuming task on WSN device and researcher decided to look for various simple compression method that can be implemented in WSN device in order to reduce energy consumed for transmitting data.Energy used from the implemented compression method will be compared to energy used in WSN device that are not using compression method. Due to limitation of the device used, the compression method used is simple definition using table that are designated beforehand and using very simple Huffman code. The energy used in this two method wil be compared with the energy used in WSN device without compression method. Device used is Arduino Leonardo as micro controller, X-bee as radio communication device, X-ctu, and arduino IDE as supporting software to configure this two device. Compression ratio that can be achieved with look-up table is 2.3333:1 and if we use huffman code will receive Min 1:1 & Max. 3.5:1 if using distributed consensus 2:1 compresssion ratio can be achieved. The result of power eficiency is 7.23903753 % using look-up table and 13.44170648 % using huffman at best. As for distributed consensus is 11.92%.

Keyword :compression, Huffman code, WSN, energy consumed.

[Halaman ini sengaja dikosongkan]

PERNYATAAN KEASLIAN
TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan tugas akhir saya dengan judul **"PENERAPAN KOMPRESI DATA UNTUK EFISIENSI DAYA JARINGAN SENSOR NIRKABEL"** 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, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Surabaya, Januari 2016

Muhammad Naufal Farid
NRP. 2211100162

KATA PENGANTAR

PENERAPAN KOMPRESI DATA UNTUK EFISIENSI DAYA JARINGAN SENSOR NIRKABEL

Tugas akhir yang memiliki beban 4 SKS (Satuan Kredit Semester) merupakan salah satu syarat yang harus dipenuhi oleh penulis untuk menyelesaikan studi Strata-1 di Jurusan Teknik Elektro Institut Teknologi Sepuluh Nopember (ITS) Surabaya.

Pada kesempatan ini, penulis juga ingin mengucapkan terima kasih kepada pihak-pihak yang telah membantu penulis dalam menyelesaikan tugas akhir ini, antara lain :

1. Kedua orang tua penulis yang selalu memberikan dukungan, semangat, dana, dan doa.
2. Bapak Wirawan dan bapak Istas Pratomo selaku Dosen Pembimbing atas segala bantuan, perhatian, dan arahan selama pengerjaan Tugas Akhir ini.
3. Bapak Endroyono selaku Koordinator Bidang Studi Telekomunikasi Multimedia Jurusan Teknik Elektro ITS.
4. Bapak dan Ibu dosen pengajar Bidang Studi Telekomunikasi Multimedia Teknik Elektro ITS.
5. Anak-anak LJ maupun reguler telekomunikasi multimedia khususnya Depa, Tiyan, Joko, Danar, Ummu, Chipe, Sherly, Bambang, Banyu.

Dalam penyusunan laporan Tugas Akhir ini, penulis menyadari banyak kekurangan, baik dalam penulisan laporan maupun pembahasan masalah. Untuk itu, penulis mengharapkan kritik dan saran yang membangun dari semua pihak agar lebih baik di masa yang akan datang. Terima kasih..

Surabaya, Mei 2015

Penulis

[Halaman ini sengaja dikosongkan]

TABLE OF CONTENTS

[Halaman ini sengaja dikosongkan]

DAFTAR ISI

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan yang cepat pada bidang jaringan sensor nirkabel telah memacu peneliti untuk bekerja membangun sensor yang memiliki physical yang lengkap. Jaringan yang seperti itu dinamakan sebagai wireless sensor network (WSN). Salah satu dari aplikasi yang potensial dari WSN adalah memonitoring benda secara real-time. Tiap benda yang dipantau terhubung dengan node di WSN. Hal ini secara khusus penting jika benda yang dipantau besar dan diluar jangkauan. Namun, aplikasi seperti ini membutuhkan data yang real-time untuk tiap node melalui WSN pada topologi jaringan mesh. Data ini biasanya disebar melalui node network sink. Node sink dipakai sebagai gateway dari WSN ke backbone. Proses penyebaran data harus efisien dikarenakan energi, pemrosesan, dan memori membebani node WSN. Dan biasanya catu daya dari node di WSN berasal dari baterai yang memiliki daya terbatas dan juga sulit jika harus sering diganti di lingkungan yang dideskripsikan. maka diperlukan metode untuk mengurangi pemakaian daya sebaik mungkin dan seminimal mungkin. Salah satu faktor terbesar pemakaian daya adalah proses korelasi dari sensor tersebut. Semakin sering sensor berorelasi maka semakin banyak pula daya yang terpakai.

Tugas akhir ini menawarkan solusi untuk masalah tersebut dengan menggunakan kompresi data untuk mengurangi waktu korelasi antar node dan juga mengurangi pemakaian daya. Sehingga bila algoritma kompresi data tersebut digunakan akan mengurangi konsumsi energi pada jaringan sensor nirkabel

1.2 Perumusan Masalah

Permasalahan yang akan diteliti adalah apakah kompresi yang dilakukan berhasil atau tidak, berapa banyak jumlah data yang terkompresi, berapa energi yang terpakai pada masing-masing metode, dan adakah pengaruh jumlah data yang dikirim terhadap peralatan yang digunakan.

1.3 Batasan Masalah

Hal-hal yang akan dilakukan dalam penelitian ini adalah sebagai berikut:

1. Keluaran dari tugas akhir ini berupa prototipe penerapan kompresi data.
2. Perangkat yang digunakan adalah Arduino Leonardo dan X-bee S2.
3. Yang dikompresi adalah hasil angka dari sensor DHT11 dan LDR.
4. Tabel kompresi telah ditetapkan sebelumnya.
5. Pengukuran dilakukan pada lingkungan Teknik Elektro ITS. Khususnya laoratorium B.304

1.4 Tujuan

Tugas akhir ini bertujuan untuk meneliti metode kompresi mana yang cocok untuk diterapkan pada peralatan WSN khususnya pada Arduino Leonardo. Dan membandingkan jumlah energi yang terpakai pada peralatan yang telah diterapkan metode kompresi dan yang belum diterapkan.

1.5 Metodologi

Metodologi yang diterapkan dalam tugas akhir ini terdiri dari tahapan-tahapan sebagai berikut :

1. Studi Literatur
   Studi literatur merupakan tahap awal dari pengerjaan tugas akhir. Pada tahap ini, dilakukan penggalian informasi dari berbagai sumber referensi diantaranya buku, internet, dan tugas akhir sebelumnya. Di tahap ini juga dilakukan observasi mengenai perangkat yang digunakan di dalam WSN.
2. Simulasi dan Perancangan
   Pada tahap ini dilakukan percobaan, simulasi, dan perancangan dari metode kompresi yang akan digunakan. Simulasi diterapkan di MATLAB.
3. Implementasi di peralatan WSN
   Perintah-perintah yang telah didapat dan dicoba di simulasi, diterapkan langsung pada perangkat WSN.
4. Pengujian
   Pada tahap ini dilakukan pengujian kerja darimetode yang telah terimplementasi.

5. Analisa

Tahap analisa dilakukan dengan mengamati dan menganalisa data pengujian yang bertujuan untuk memudahkan menarik kesimpulan.

1.6 Sistematika Pembahasan

Pembahasan tugas akhir ini dibagi menjadi 5 bab dengan sistematika sebagai berikut :

Bab I Pendahuluan

Pada bab ini dijelaskan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, metodologi, sistematika pembahasan, dan relevansi.

Bab II Teori Dasar

Pada bab ini dijelaskan mengenai tinjauan pustaka yang mendukung dalam pengimplementasian kompresi data pada WSN khususnya Arduino Leonardo.

Bab III Perancangan dan Implementasi

Pada bab ini dijelaskan simulasi dan implementasi kompresi data pada peralatan WSN.

Bab IV Analisa Data dan Pembahasan

Bab ini berisi data pengujian dari sistem yang dibangun dimana data yang didapat dianalisa untuk memudahkan menarik kesimpulan.

Bab V Penutup

Pada bab ini berisi kesimpulan dan saran.

1.7 Relevansi

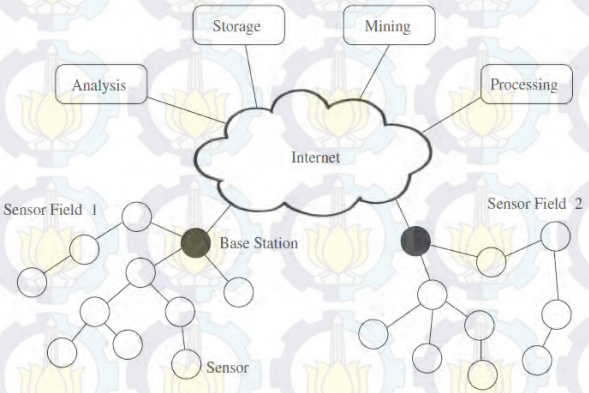
Dengan selesainya tugas akhir ini, diharapkan dapat memberikan referensi dalam menerapkan kompresi data pada WSN khususnya Arduino Leonardo, dan di kemudian hari dapat dikembangkan lebih lanjut untuk kepentingan orang banyak.

[Halaman ini sengaja dikosongkan]

BAB II
TINJAUAN PUSTAKA

2.1 Jaringan Sensor Nirkabel

Perkembangan teknologi elektronika telekomunikasi yang seiring dengan perkembangan protokol komunikasi dan informasi yang ada sekarang telah menuju suatu generasi baru yang murah, akurat, dan memiliki daya jangkau yang lebih luas.Ketika banyak sensor terhubung pada suatu jaringan dan berkomunikasi secara nirkabel, maka hal tersebut dapat disebut dengan jaringan sensor nirkabel (WSN). Maka dari itu, jaringan sensor tidak hanya sebuah komponen untuk mengarahkan, namun juga sebuah *on-board processing*, komunikasi dan kapabilitas penyimpanan. Dengan peningkatan ini, sebuah node sensor sering tidak hanya bertanggung jawab untuk pengumpulan data, namun juga untuk analisis jaringan, korelasi dan penggabungan dari data sensor itu sendiri dan data dari sensor yang lain. Ketika banyak sensor secara kooperatif memonitor *physical environments* yang besar, akan menjadi Jaringan Sensor Nirkabel. Node sensor berkomunikasi tidak hanya kepada sesamanya namun juga dengan *Base Station* (BS) menggunakan radio nirkabelyang mereka miliki, mengijinkan node sensor tersebut menyebarkan data sensor mereka ke penerima. Ilustrasi sederhana sebuah jaringan sensor dapat dilihat pada **Gambar 2.1**.



Gambar 2.1 Ilustrasi Jaringan Sensor Nirkabel

Kebutuhan akan WSN pada dasarnya dikembangkan pada bidang militer yang membutuhkan banyak informasi pada medan perang, namun saat ini WSN sudah banyak digunakan pada banyak bidang seperti pada industri dan digunakan juga untuk melakukan pengawasan baik pada bidang kedokteran, lingkungan, industri, bangunan, kendaraan, dan lain-lain. Terdapat banyak jenis pengaplikasian dan penggunaan dari WSN, namun pada umumnya digunakan untuk mengawasi dan mengambil data pada tempat yang sulit untuk dijangkau dan kecil sekali kemungkinan untuk mengganti sumber tenaga untuk mengoperasikan peralatan tersebut. Selain itu terdapat juga pengaplikasian yang digunakan untuk mengamati suatu parameter tertentu pada suatu daerah seperti memonitoring suhu, kelembaban, dan lux pada suatu lingkungan tertentu.Karakteristik yang unik pada WSN antara lain :

- Daya yang terbatas.
- Dapat bertahan pada lingkungan yang tidak mudah dicapai oleh manusia.
- Adaptif terhadap perubahan node
- *Mobile*
- Disebar dalam suatu wilayah

WSN biasanya terdiri dari dua penugasan yaitu node dan koordinator, node sensor biasanya disebar pada suatu lingkungan dan bertugas untuk mengambil suatu data dan mengirimkannya ke koordinator. Untuk melakukan hal tersebut dibutuhkan perangkat pendukung yang biasanya terdiri dari :

- Transceiver berfungsi untuk menerima/mengirim data dengan menggunakan protokol IEEE 802.15.4 kepada device lain seperti*concentrator*,*modem* Wifi, dan *modem* RF.
- *Mikrokontroler*,berfungsi untuk melakukan fungsi perhitungan, mengontrol dan memproses beberapa alat yang terhubung dengan *mikrokontroler*.
- sumber tenaga, berfungsi sebagai sumber energi bagi sistem WSN secara keseluruhan.
- Sensor, berfungsi untuk mambaca besaran-besaran fisis yang akan diukur

2.2 Kompresi

Bertumbuhnya ukuran *database* dan makin banyaknya pengguna dan durasi dari pemakaian pada lokasi tertentu membuat data dalam jumlah yang sangat banyak ditransfer antara suatu komputer dengan komputer atau alat lain. Untuk memfasilitasi transmisi yang dibutuhkan pada transfer data, jalanya komunikasi dan membantu alat – alat seperti modem dan multiplexer, telah secara kontinyu diperbrui oleh banyak organisasi untuk memungkinkan transfer-data dengan kapabilitas yang tinggi

Walaupun solusi yang nyata untuk masalah penyimpanan data dan transfer informasi adalah menginstal alat penyimpanan tambahan dan menambah fasilitas komunikasi yang ada, untuk melakukan hal itu diperlukan suatu biaya dan peralatan tambahan. Satu metode yang bisa diimplementasikan untuk meringankan masalah porsi dari penyimpanan data dan transfer informasi adalah lewat representasi data oleh kode yang lebih efisien. Jika salah satunya menguji sebuah database atau memonitor sebuah transmisi, maka ada kesempatan yang bagus untuk memperbaiki database dan *transmission sequence*bisa di *encoded* lebih efisien. Maka dari itu kompresi merupakan suatu solusi yang tepat untuk mengatai hal ini

2.2.1 Keuntungan kompresi

Ketika data kompresi digunakan untuk mengurangi kebutuhan penyimpanan, waktu eksekusi dari keseluruhan program akan berkurang. Ini disebabkan karena pengurangan dari penyimpanan tersebut akan menghasilkan sebuah pengurangan dari usaha untuk *disc-access,* ketika encoding dan decoding digunakan untuk teknik kompresi maka instruksi program tambahan juga akan dieksekusi. Sejak waktu eksekusi dari sejumlah instruksi program secara signifikan kurang dari waktu yang dibutuhkan untuk mengakses dan data tranfer ke alat lain, waktu eksekusi program secara keseluruhan bisa dikurangi.

Dengan memperhatikan transmisi dari data, kompresi menyediakan perencanaan jaringan dengan beberapa keuntungan tambahan pada penghematan biaya dengan mengirim data yang lebih sedikit ke jaringan telepon dimana biaya panggilan biasanya berdasarkan durasi. Pertama, kompresi bisa mengurangi kemungkinan eror transmisi karena karakter yang lebih sedikit ditransmisikan ketika data dikompres dan kemungkinan eror tetap konstan. Kedua, ketika kompresi meningkatkan efisiensi, hal tersebut bisa mengurangi atau

mungkin menghilangkan kerja yang ekstra. Akhirnya, dengan mengkonversi teks yang direpresentasikan oleh kode konvolusional seperti ASCII kedalam sebuah kode yang berbeda, algoritma kompresi bisa menyediakan tingkat keamanan terhadap monitoring yang ilegal.

Untuk komunikasi data, transfer dari data kompresi lewat sebuah medium menghasilkan peningkatan pada tingkat efektifitas pada transfer informasi, walaupun tingkat data transfer yang sebenarnya mengekspresikan bits per detik tetap sama. Kompresi data bisa diimplementaikan pada perangkat keras manapun dengan perangkat lunak atau menggunakan perangkat keras yang khusus yang mendukung satu atau lebih teknik kompresi.

Pada gambar, blok diagram dasar dari kompresi data dijelaskan. Ditunjukkan pada gambar, kompresi dan dekompresi diperlukan pada *processor* penggunauntuk memasukkan PC, *intelligent terminals* atau pada alat lain ke *processor*, seperti komponen komunikasi secara khusus. Banyak diantara komponen ini adalah konsentrator data dan multiplexer statik.

Gambar 2.2 blok diagram dasar kompresi data

2.2.2 Terminologi

Seperti yang digambarkan pada gambar x.x, sebuah aliran data dari sumber dioperasikan tergantung pada algoritma tertentu untuk menghasilkan sebuah aliran data kompresi. kompresi dari aliran data sumber ini terkadang bisa disebut sebagai proses enkoding dengan hasil bahwa aliran data kompresi juga disebut aliran data enkoded. Membalikan proses, aliran data kompresi didekompresi untuk memproduksi kembali sebuah aliran data sumber. Karena proses kompresi ini menghasilkan aliran data kompresi pada decoding, hasilnya terkadang disebut aliran data dekoded. Digunakan aliran data sumber dan aliran data dekoded secara sembarang, juga pada aliran data kompresi dan aliran data enkoded.

Derajat dari pengurangan data merupakan hasil dari proses kompresi yang biasa disebut rasio kompresi. rasio ini mengukur

kuantitas dari data kompresi terhadap perbandingan kuantitas dari data sumber, seperti (Ruth dan Krentzler,1972):

$$\text{Compression ratio} = \frac{\text{Length of original data string}}{\text{Length of compressed data string}}$$

Dari persamaan diatas, terlihat bahwa makin besar rasio kompresi maka makin efektif teknik kompresi yang digunakan. Cara lain digunakan ketika berbicara tentang kompresi adalah *figure of merit*, dimana :

$$\text{Figure of merit} = \frac{\text{Length of compressed data string}}{\text{Length of original data string}}$$

Figure of merit harus selalu kurang dari proses kompresi supaya efektif. Fraksi dari reduksi data adalah satu kurang dari *figure of merit*. Maka dari itu, teknik kompresi yang menghasilkan satu karakter dari data kompresi untuk tiap tiga karakter pada aliran data sumber akan mempunyai rasio kompresi 3, figure of merit 0.33, dan fraksi reduksi data menjadi 0.66.

2.2.3 Huffman Coding

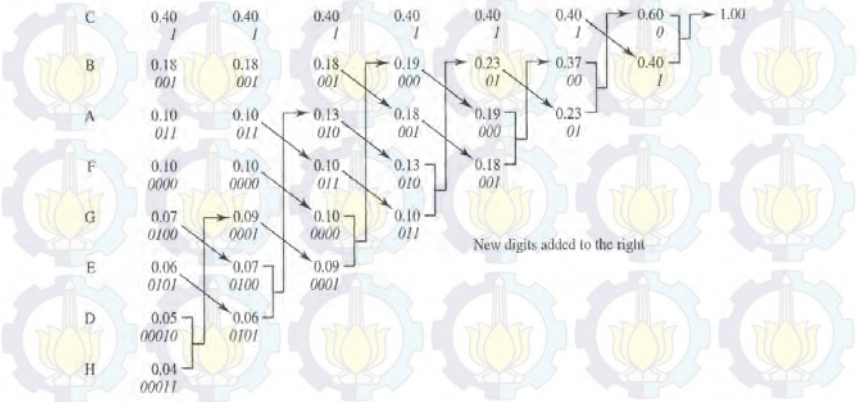
Kode Huffman merupakan kode prefix dan optimum pada model tertentu. Prosedur Huffman berdasarkan dua observeasi dibawah ini :

- Sebuah kode optimum, simbol yang banyak muncul (mempunyai probabilitas kemunculan tinggi) akan mempunyai panjang kode yang pendek daripada simbol yang jarang muncul
- Sebuah kode optimum, dua simbol dapat muncul dengan panjang kode yang sama.

Sebuah cara alternatif untuk membuat kode huffman ini adalah menggunakan fakta bahwa kode huffman, dengan menjadi kode prefix, bisa direpresentasikan sebagai *binary tree* yang node eksternalnya

merupakan simbol – simbol. Kode huffman untuk simbol manapun bisa diterima dengan menelusuri akar hingga ke ujung node dari simbol. Menambah 0 ke *codeword* tiap kali berpindah ke tingkat yang lebih tinggi dan 1 ketika tiap kali berpindah ke tingkat yang lebih rendah.

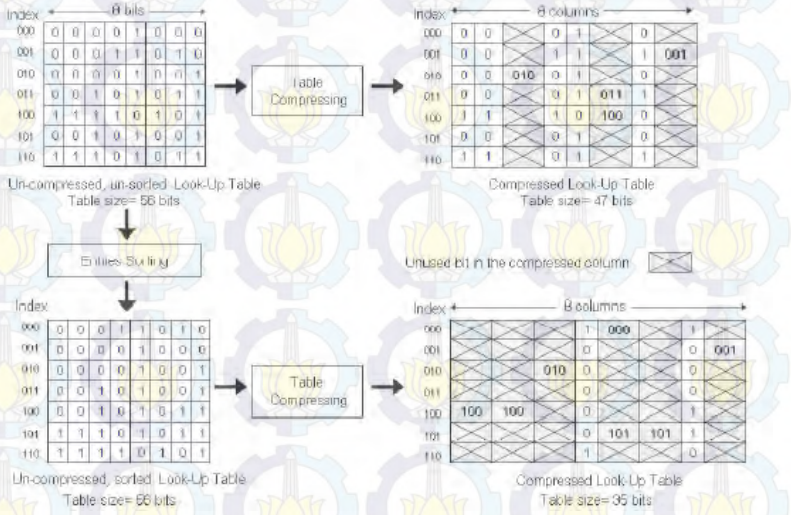
Dibangunlah sebuah *binary tree* yang berawal dari ujung node. Telah diketahui bahwa *codeword* untuk dua simbol dengan kemungkinan paling kecil identikal kecuali bit terakhir. Ini berarti bahwa traversal dari akar ke ujung pada dua simbol ini harus sama kecuali pada step akhir. Ini juga berarti bahwa ujung dari dua simbol dengan probabilitas yang lebih rendah turun pada node yang sama. Ketika diketahui mempunyai hubungan dengan ujung dari simbl dengan probabilitas yang lebih rendah ke node tunggal, node diperlakukan sebagai simbol untuk mengurangi alfabet. Kemungkinan dari simbol ini adalah penjumlahan dari probabilitas turunya simbol. Sekarang bisa diurutkan node – node yang lain untuk mengurangi alfabet dan menggunakan peraturann yang sama untuk membangkitkan node awal untuk node – node yang lain ke dua simbol pada alfabet yang dikurangi dengan probabilitas paling kecil. Melanjutkan dengan cara ini, diakhiri dengan node tunggal, yang mana merupakan akar node. Untuk menerima kode dari tiap simbol, pohon di traverse dari akar ke node – node ujung, memberikan 0 ke tingkat yang lebih tinggi dan 1 ke tingkat yang lebih rendah. Prosedur ini akan terlihat lebih jelas jika dilihat pada gambar.



Gambar 2.3 *binary tree* pada huffman *coding*

2.2.4 Look-up Table

Look-up table adalah metode kompresi yang berdasarkan pada *dictionary* dimana instruksi sebelumnya diganti dengan kode indeks dari tabel yang menyimpan data original tersebut. *Look-up Table* digunakan untuk menyimpan data semula dan indeksnya bertugas sebagai penunjuk seperti apa data asal dari indeks tersebut. Salah satu dari masalah terbesar metode ini adalah tabel ini dapat sangat besar ukurannya dan menghilangkan keuntungan dari penghematan yang didapatkan dari hasil kompresi dan menggunakan lebih banyak daya pada proses kompresinya. Keuntungan dari metode ini adalah waktu pemrosesan dapat lebih cepat dan jika tabel kompresi yang dibuat optimal dapat menghemat lebih banyak daya lebih banyak jika dibandingkan dengan energi yang dipakai untuk melakukan proses kompresinya.



Gambar 2.4 Contoh Kompresi Menggunakan *look-up table*

Untuk membuat tabel kompresi dan kode kompresinya dilakukan beberapa langkah yaitu : (1) Mulai dari kode biner originalnya lalu gabungkan ke seluruh instruksi. (2) Simpan semua instruksi yang berbeda ke *look-up table*. (3) Pada kode original ganti semua instruksi dengan indeks dari *look-up table* dimulai dari 0. Disini indeks memiliki

panjang yang merupakan kelipatan $\log_2$ dari banyaknya instruksi yang berbeda[9].

2.3 Distributed Consensus

Distributed consensus adalah salah satu cara yang digunakan untuk mencapai sistem yang reliabel. Hal ini biasanya membutuhkan proses yang membuat node-node tersebut memiliki persetujuan atau memiliki data yang sama satu sama lain. Konsensus ini menggunakan pertukaran informasi yang terus diulang untuk mencapai dan menghitung fungsi tertentu. Setiap node dari algoritma terdistribusi ini akan mengupdate nilai berdasarkan nilai yang dimiliki dan informasi node-node tetangganya. Atau secara sederhana dapat dituliskan sebagai persamaan dibawah :

$$X_l(t+1) = \sum_{m \in N_l} w_{l,m} x_m(t), l = 1, \dots, L, \qquad (2.1)$$

dimana $w_{l,m}$ adalah bobot dari node yang dipilih dengan node tetangganya yang memenuhi syarat

$$\sum w_{l,m} = 1 \qquad (2.2)$$

dan nilai x pada t = 0 atau x(0) adalah hasil dari pengukuran awal pada tiap node yang nantinya akan di iterasikan. Tiap node dapat direpresentasikan kedalam satu matriks menjadi :

$$x(t) = \begin{bmatrix} x_1(0) & x_1(1) & \cdots & x_1(N) \\ x_2(0) & & \vdots & \\ x_3(0) & & \vdots & \\ x_N(0) & & \vdots & \end{bmatrix} \qquad (2.3)$$

Fungsi yang dapat dihitung oleh fungsi konsensus adalah rata-rata, rata-rata dengan jaringan berbobot, nilai maksimum, dan nilai minimum. Matriks bobot dapat berubah sesuai dengan fungsinya yang berubah terhadap waktu (*time variant*) atau yang tidak berubah terhadap waktu (*time invariant*). Untuk jaringan yang menggunakan topologi *time invariant* menggunakan persamaan

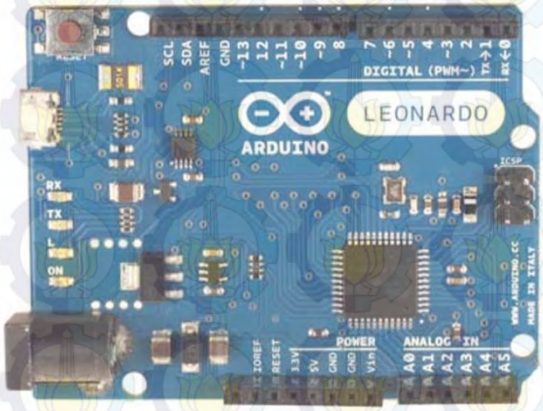
$$X_l(t+1) = w_{l,l}x_l(t) + \sum_{m \in N_l} w_{l,m}x_m(t), l = 1, \dots, L, \qquad (2.4)$$

State dari semua node akan dianggap sama dan mencapai konsensus jika dianggap sudah cukup mendekati nilai hasil dari iterasi sebelumnya atau kondisi dari nilai konvergen dapat dilihat pada [9].

2.4 Mikrokontroler

Mikrokontroler adalah sebuah chip yang berfungsi sebagai pengontrol rangkaian elektronik dan umunya dapat menyimpan program didalamnya. Mikrokontroler umumnya terdiri dari CPU (Central Processing Unit), memori, I/O tertentu dan unit pendukung seperti Analog-to-Digital Converter (ADC) yang sudah terintegrasi di dalamnya.

Kelebihan utama dari mikrokontroler ialah tersedianya RAM dan peralatan I/O pendukung sehingga ukuran board mikrokontroler menjadi sangat ringkas. Flash PEROM on-chip tersebut memungkinkan memori program untuk diprogram ulang dalam sistem (in-system programming) atau dengan menggunakan programmer non-volatile memory konvensional. Kombinasi CPU 8 bit serba guna dan Flash PEROM, menjadikan mikrokontroler MCS51 menjadi microcomputer handal yang fleksibel.



Gambar 2.5 Arduino Leonardo

Arduino Leonardo adalah salah satu board mikrokontroler berbasis ATmega32U4 (datasheet). Memiliki 20 pin input dari output digital dimana 7 pin input tersebut dapat digunakan sebagai output PWM dan 12 pin input analog, 16 MHz osilator kristal, koneksi USB, jack power, ICSP header, dan tombol reset. Untuk mendukung mikrokontroler agar dapat digunakan. Board mikrokontroler ini dapat dihubungkan dengan modul – modul lain sesuai kebutuhan dengan menggunakan sambungan atau yang disebut dengan shield.

Mikrokontroler ini digunakan untuk memproses data dari sensor untuk dimasukkan dalam sebuah frame. Frame ini merupakan barisan kode yang terdiri dari data sensor dan tujuan pengiriman. Selain melakukan proses data sensor, mikrokontroler bertugas untuk melakukan pengubahan alamat tujuan yang dituju dalam pengiriman agar sesuai yang diinginkan. Arduino menyediakan sebuah Integrated Development Environment yang disebut dengan Arduino IDE. IDE ini dijalankan pada komputer yang dimana pada software ini programmer dapat menuliskan dan mengkompile sketches dan kemudian meng-uploadnya ke Arduino melalui koneksi USB. Desainnya berupa text editor yang didesain khusus untuk menuliskan kode dan terdapat sekumpulan fungsi yang didesain untuk membantu dalam melakukan kompilasi dan membuka sketches.

Mikrokontroler	ATmega32u4
Tegangan Operasi	5V
Input Voltage (disarankan)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pin	20 pin
Channel PWM	7 pin
Input Analog	12 pin
Arus DC per pin I/O	40 mA
Arus DC untuk pin 3.3V	50 mA
Flash Memory	32 KB (ATmega32u4) 4 KB digunakan bootloader
SRAM	2.5 KB (ATmega32u4)
EEPROM	1 KB (ATmega32u4)
Clock Speed	16 MHz

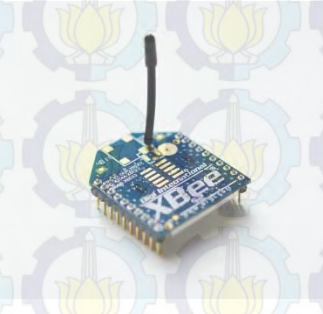
Gambar 2.6 spesifikasi Arduino Leonardo

Sketch merupakan sebuah format coding khusus yang digunakan pada Arduino yang terdiri dari dua metode yang dibutuhkan,

yang satu dijalankan saat Arduino sesaat setelah direset dan yang satunya dijalankan secara terus – menerus. Kode yang dituliskan pada sketch menggunakan bahasa pemrograman seperti bahasa C sehingga menjadikan pemrograman lebih mudah untuk dilakukan meskipun memiliki beberapa batasan yang tidak ada pada bahasa pemrograman C.

2.5 Zigbee 802.15.4

Zigbee adalah sebuah spesifikasi protokol komunikasi radio digital berdaya rendah berdasarkan spesifikasi IEEE 802.15. Jangkauan yang dapat dijangkau oleh zigbee ini mencapai 100 meter dengan catatan *line of sight*. Biayanya yang murah memungkinkan teknologi ini banyak digunakan sebagai pengendali jaringan nirkabel dan aplikasi pemantauan. Penggunaan daya yang rendah membuat alat ini dapat bertahan lama bahkan dengan baterai berukuran lebih kecil. Dan jaringan mesh memberikan realibilitas yang tinggi serta jangkauan yang lebih luas. Pada tugas akhir ini modul Xbee yang digunakan adalah Xbee S2.



Gambar 2.7 Xbee Module berprotokol ZigBee

XBee adalah brand yang mensupport dari berbagai protokol komunikasitermasuk ZigBee 802.15.4 dan WiFi. Gambar Xbee modul diperlihatkan pada gambar diatas.

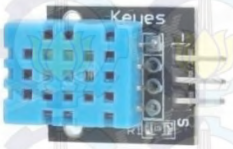
Standar protokol ZigBee sama dengan standar Bluetooth. Manufaktur perangkat suatu pabrik sepenuhnya support dengan standar ZigBee yang dapat berkomunikasi dengan perangkat ZigBee buatan pabrik lainnya. Contohnya Bluetooth headset Motorola dapat berkomunikasi dengan Apple iPhone, saklar lampu Centalite ZigBee dapat berkomunikasi dengan kunci pintu Black & Decker (Faludy, 2010).

Specification	XBee
Performance	
Indoor/Urban Range	up to 133 ft. (40 m)
Outdoor RF line-of-sight Range	up to 400 ft. (120 m)
Transmit Power Output	2mW (+3dBm), boost mode enabled 1.25mW (+1dBm), boost mode disabled
RF Data Rate	250,000 bps
Data Throughput	up to 35000 bps (see chapter 4)
Serial Interface Data Rate (software selectable)	1200 bps - 1 Mbps (non-standard baud rates also supported)
Receiver Sensitivity	-96 dBm, boost mode enabled -95 dBm, boost mode disabled

Gambar 2.8 Spesifikasi Xbee

2.6 Sensor

Sensor adalah suatu alat yang mampu untuk mengubah suatu bentuk energi ke bentuk energi lain, dalam hal ini adalah mengubah dari energi besaran yang diukur menjadi energi listrik. Biasanya besaran yang diukur oleh sensor ini berupa fenomena yang terjadi pada suatu lingkungan seperti suhu, kelembaban, getaran, kecepatan, dan lain-lain. Setelah mengukur fenomena ini, tugas sensor untuk menyampaikan pengukuran dalam bentuk baik representasi tegangan atau angka yang kemudian pada tugas akhir ini akan dibaca oleh *mikrokontroler* dan diolah lalu dikirim melalui jaringan nirkabel.



Gambar 2.9 Sensor DHT11.

Terdapat banyak bentuk sensor. Biasanya perangkat yang murah dan dirancang untuk satu tujuan dan yang memiliki kemampuan terbatas untuk diproses. Kebanyakan sensor sederhana adalah komponen yang diskrit bahkan mereka yang memiliki bagian-bagian yang lebih canggih jika diperlakukan sebagai komponen terpisah. Sensor analog

atau digital biasanya dirancang untuk mengukur hanya satu hal seperti sensor temperatur dan kelembaban pada gambar diatas.

2.6.1 Cara Sensor Mengambil Data

Sensor adalah perangkat elektronik yang membangkitkan tegangan berdasarkan sifat unik dari bahan kimia dan konstruksi mekanik yang dibuat untuk membentuk sensor tersebut. Sensor mendapatkanmenangkap adanya perubahan energi eksternal dan mengubahnya menjadi sinyal listrik proporsional (tegangan, arus, digital, dan sebagainya). Sebagai contoh, sebuah sensor kelembaban kelembaban di udara. Sensor kelembaban bereaksi terhadap fenomena tersebut dan menghasilkan tegangan sehingga mikrokontroler atau perangkat serupa dapat membaca dan digunakan untuk menghitung nilai pada sebuah skala.

2.6.2 Contoh – contoh Sensor

Semua dari jaringan sensor dimulai dari satu sensor yang digunakan untuk membaca dan menginterpretasikan sebuah data. Akan terpikirkan dari segala hal yang ada di rumah, kantor, halamanmu atau di sekelilingnya dapat diukur. Suhu yang ada di kamar, mendeteksi pembawa surat telah mengirimkan surat terbaru ke kotak surat, atau menyimpan log dari beberapa kali hewan peliharaan seperti kucing atau anjing melewati pintu rumah. Contoh sensor yang tersedia adalah sebagai berikut :

- *Humidity sensors*:. suatu alat ukur yang digunakan untuk membantu dalam proses pengukuran atau pendifinisian yang suatu kelembaban uap air yang terkandung dalam udara.
- *Light sensors*: Sensor yang mengukur intensitas atau kurangnya cahaya adalah jenis khusus dari resistor: resistor yang bergantung pada cahaya biasa disebut dengan *light dependent resistor*(LDR), kadang-kadang disebut resistor foto atau photocells.
- *Temperature sensors*: suatu komponen yang dapat mengubah besaran panas menjadi besaran listrik sehingga dapat mendeteksi gejala perubahan suhu pada obyek tertentu. Sensor suhu melakukan pengukuran terhadap jumlah energi panas yang dihasilkan oleh suatu obyek sehingga memungkinkan untuk mengetahui atau mendeteksi gejala perubahan-perubahan suhu dalam bentuk output analog atau digital.

- Sensor proximity merupakan sensor atau saklar yang dapat mendeteksi adanya target jenis logam dengan tanpa adanya kontak fisik. Biasanya sensor ini tediri dari alat elektronis solid-state yang terbungkus rapat untuk melindungi dari pengaruh getaran, cairan, kimiawi, dan korosif yang berlebihan. Sensor proximity dapat diaplikasikan pada kondisi penginderaan pada objek yang dianggap terlalu kecil atau lunak untuk menggerakkan suatu mekanis saklar.
- Proses penginderaan sensor kecepatan merupakan proses kebalikan dari suatu motor, dimana suatu poros/object yang berputar pada suatui generator akan menghasilkan suatu tegangan yang sebanding dengan kecepatan putaran object. Kecepatan putar sering pula diukur dengan menggunakan sensor yang mengindera pulsa magnetis (induksi) yang timbul saat medan magnetis terjadi.

DHT11 adalah sensor digital yang dapat mengukur suhu dan kelembaban udara di sekitarnya. Sensor ini sangat mudah digunakan bersama dengan Arduino. Memiliki tingkat stabilitas yang sangat baik serta fitur kalibrasi yang sangat akurat. Koefisien kalibrasi disimpan dalam OTP program memory, sehingga ketika internal sensor mendeteksi sesuatu, maka module ini menyertakan koefisien tersebut dalam kalkulasinya.

Gambar 2.10 Macam-Macam Sensor Proximity

DHT11 termasuk sensor yang memiliki kualitas terbaik, dinilai dari respon, pembacaan data yang cepat, dan kemampuan anti-interference. Ukurannya yang kecil, dan dengan transmisi sinyal hingga 20 meter, membuat produk ini cocok digunakan untuk banyak aplikasi-aplikasi pengukuran suhu dan kelembaban.

BAB III
RANCANGAN IMPLEMENTASI KOMPRESI DATA PADA JARINGAN SENSOR NETWORK

3.1 Metodologi Penelitian

Tahapan pengerjaan tugas akhir ini diselesaikan dengan beberapa tahapan. Pada awalnya implementasi kompresi data ini diawali dengan kegiatan perancangan algoritma kompresi apa yang akan dipakai. Karena kompresi akan di implementasikan pada perangkat yang mengukur suhu, kelembaban dan luminansi, maka sebelum menentukan tabel kompresi harus diketahui terlebih dahulu probabilitas nilai yang sering muncul pada tempat yang akan diukur dikarenakan keterbatasan perangkat yang tidak dapat membuat tabel kompresi sendiri. Setelah itu akan dibuat tabel kompresi yang berdasarkan pada hasil pengukuran suhu, kelembaban, dan luminansi.

Selain hardware, yang perlu dipersiapkan untuk mencapai tujuan tugas akhir ini adalah instalasi software serta konfigurasinya. Software yang digunakan dalam tugas akhir ini adalah arduino IDE dan X-CTU. Software arduino digunakan untuk konfigurasi Arduino Leonardo pada node dan koordinator dan digunakan untuk mengambil data dan menampilkan data yang terbaca pada sensor dan pada perangkat Xbee, sedangkan software X-CTU nantinya akan digunakan untuk mengkonfigurasi Xbee agar dapat saling terhubung sesuai dengan keiinginan pengguna. Terdapat beberapa topologi yang biasanya digunakan untuk menghubungkan perangkat Xbee dan diantaranya adalah *point-to-point, multi-hop,* dan *mesh*. Pada tugas akhir ini topologi yang dipakai adalah *point-to-point* yang dipakai pada implementasi kompresi data menggunakan *huffman coding* dan *look-up* table, sedangkan topologi *mesh* digunakan pada skenario *distributed consensus*.

Gambar 3.1 Metodologi Pengerjaan Sistem.

3.2 Instalasi Perangkat Lunak

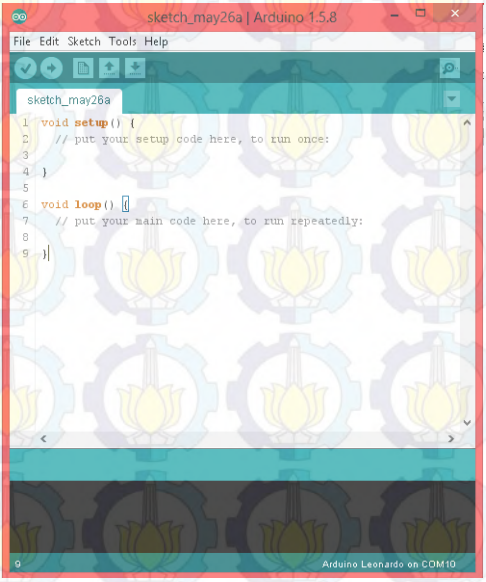
Dari berbagai komponen – komponen yang ada dalam tugas akhir ini, diperlukan software atau perangkat lunak yang spesifik untuk digunakan dalam konfigurasi. Software yang dibutuhkan meliputi kebutuhan dari node – node sensor hingga server. Berikut software yang digunakan ,cara menginstalasi, dan cara mengkonfigurasikannya.

3.2.1 Instalasi Software Mikrokontroler

Untuk melakukan konfigurasi node sensor dan koordinator pada jaringan sensor nirkabel ini yang bermikrokontroler Arduino Leonardo, diperlukan software *Arduino* untuk melakukan pemrograman.

Software arduino dapat diunduh pada website resmi dari arduino sendiri. Versi yang digunakan pada tugas akhir ini adalah versi 1.5.8. Untuk instalasi arduino dan konfigurasi, dibutuhkanpenginstalan software arduino pada PC terlebih dahulu. Setelah berhasil diinstal, maka aplikasi arduino dapat dijalankan dan akan tampil tampilan arduino seperti pada Gambar 3.2.Untuk melakukan pemrograman pada software arduino, maka diperlukan konfigurasi awal pada parameter berikut :

Konfigurasi parameter – parameter diatas dapat dilakukan dengan memilih *menu tool* alat. Untuk melakukan pemrograman pada arduino, bahasa yang digunakan adalah bahasa arduino. Bahasa arduino ini sedikit berbeda bahasa C walaupun beberapa *syntax*-nya yang menyerupai atau sama. Bahasa yang digunakan oleh arduino sebagian besar sama dan tidak seperti bahasa C, bahasa yang digunakan pada software ini banyak memiliki keterbatasan dan sangat diperlukan *error detection* yang tinggi untuk dapat mengetahui bagian mana yang menyebabkan suatu kesalahan dapat terjadi.

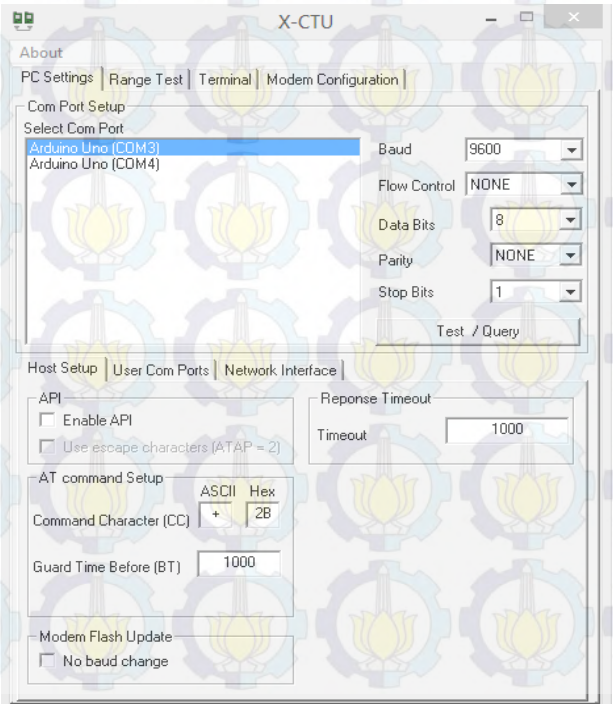


Gambar 3.2 Tampilan Awal Pada Software Arduino Versi 1.5.8.

3.2.2 Instalasi X-CTU

Agar dua buah Xbee dapat berkomunikasi dengan satu sama lain maka perlu diatur bagaimana cara mereka berkomunikasi. Untuk melaksanakan hal tersebut diperlukan software bernama X-CTU untuk mengkonfigurasikan perangkat Xbee tersebut. Versi software yang digunakan adalah versi 5.2.8.6 dan software ini terdapat secara gratis pada web resmi dari digi. Software ini menyajikan banyak sekali keuntungan jika ingin mengkonfigurasikan perangkat zigbee seperti Xbee S2 dan cara menginstalnya juga mudah.

Jika instalasi telah berhasil dilakukan maka selanjutnya dapat dilakukan konfigurasi pada perangkat Xbee yang diinginkan dengan memilih perangkat tersebut akan menjadi *end device* atau *coordinator* dan juga mengkonfigurasikan alamat pengiriman dan cara pengiriman. Untuk lebih jelasnya dapat dilihat gambar 3.3.

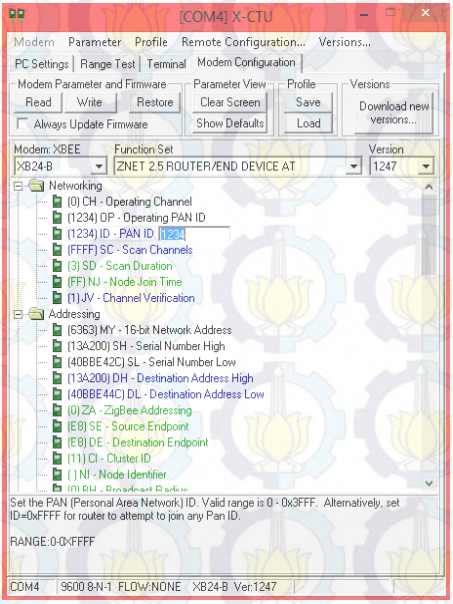


Gambar 3.3 Tampilan Awal pada X-CTU

3.3 Konfigurasi Xbee

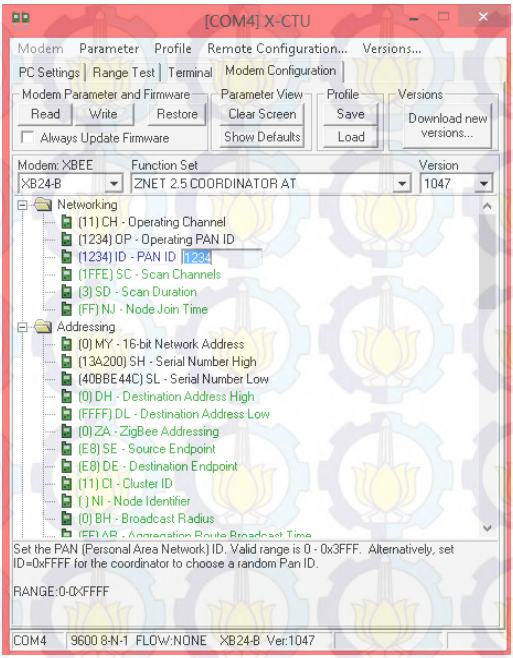
Pada tugas akhir ini akan digunakan perangkat Xbee S2 untuk mengirimkan data yang berasal dari Arduino Leonardo dan akan dikirim secara nirkabel. Namun dikarenakan cara pengiriman yang akan dilakukan adalah secara *point-to-point* dan *mesh* yang menggunakan dua ampai tiga perangkat maka yang perlu dikonfigurasi hanyalah masksimal dua node dan satu koordinator saja.

Untuk mengkonfigurasikan perangkat X-bee ini pertama-tama install X-CTU terlebih dahulu, kemudian setelah terinstal dengan benar buka X-CTU dan buka *Modem Configuration* pada *taskbar*, buka XB24-B pada modem dan ZNET 2.5 ROUTER/END DEVICE AT pada function set untuk mengkonfigurasi node sensor. Jika ingin mengkonfigurasi koordinator maka buka ZNET 2.5 COORDINATOR AT pada *function set*. Atau jika ingin melihat konfigurasi sebelumnya klik read pada *modem parameter and firmware*. Jika telah benar dikonfigurasi maka klik *write* pada pilihan *modem parameter and firmware*.



Gambar 3.4 Konfigurasi Xbee pada Node Sensor *point-to-point*

Lalu scan channel yang digunakan sampai dengan FFFF agar kanal yang ditelaah sampai dengan kanal terakhir yaitu FFFF. Lalu yang harus dikonfigurasikan kemudian adalah *Destination Address High*(DH) dan *Destination Address Low*(DL). DH yang diberikan adalah serial number dari peraangkat *zigbee* yang digunakan dan dalam tugas akhir ini digunakan X-bee S2 yang berserial 13A200 pada node sensor dan koordinator. Sedangkan untuk DL menggunakan delapan serial terakhir dari perangkat X-bee. Untuk DL pada node sensor menggunakan 40BBE44C dan pada koordinator menggunakan 40BBE44C.



Gambar 3.5 Konfigurasi Xbee pada Koordinator *point-to-point*

Jika mengkonfigurasikan dengan topologi *mesh* akan menjadi :

- Node : ZNET 2.5 ROUTER/END DEVICE AT
- Koordinator : 2.5 COORDINATOR AT

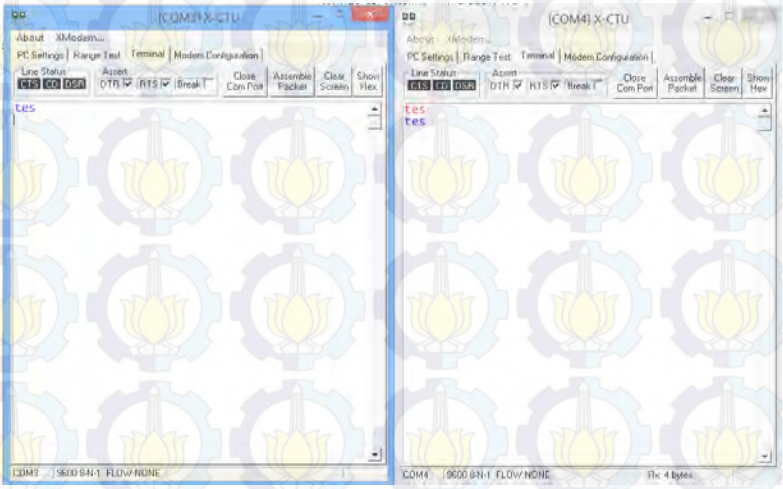
Sedangkan Konfigurasi node sensor topologi *mesh* pada tiap perangkat dilakukan dengan cara :

- PAN ID : 1234
- ATSH : 13A200
- ATSL : 40BBE42C
- ATDH : 0
- ATDL : FFFF
- ATBH : 0
- ATBD : 9600

Sedangkan untuk koordinator menggunakan konfigurasi :

- PAN ID : 1234
- ATSH : 13A200
- ATSL : 40BBE42C
- ATDH : 0
- ATDL : FFFF
- ATBH : 0
- ATBD : 9600

Setelah kedua perangkat Xbee tersebut dikonfigurasikan kemudian dilakukan pengujian apakah kedua perangkat tersebut saling terhubung dengan memilih tab terminal yang terdapat dalam satu tab dengan PC settings. Untuk lebih jelasnya dapat dilihat pada gambar dibawah.



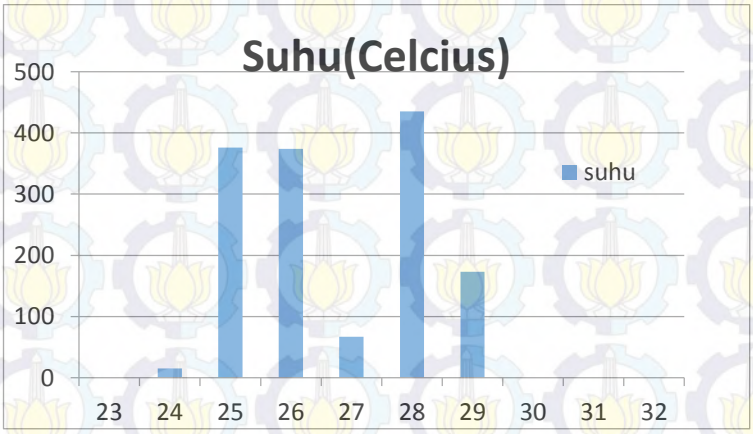
Gambar 3.6 Tes Koneksi Antar Perangkat Xbee

3.4 Pengambilan Data untuk Tabel Kompresi

Agar dapat melakukan kompresi menggunakan algoritma huffman maka harus diketahui terlebih dahulu berapa probabilitas nilai kelembaban, suhu, dan luminansi dari lingkungan yang akan diukur. Dalam tugas akhir ini lingkungan yang diukur adalah didepan ruangan B.304 dan pengukuran ini berlangsung selama 2 jam pada subuh, pagi, siang, sore, dan malam hari dengan waktu antar pengambilan data 3 menit sekali. Setelah pengukuran didapatkan hasil seperti pada tabel dibawah. Pada tabel diatas terlihat bahwa suhu yang paling sering terbaca oleh sensor adalah 28^{o}C dan luminansi terbesar adalah yang berada dibawah 45 x 2048 lux.

Tabel 3.1 Hasil Pengukuran Suhu

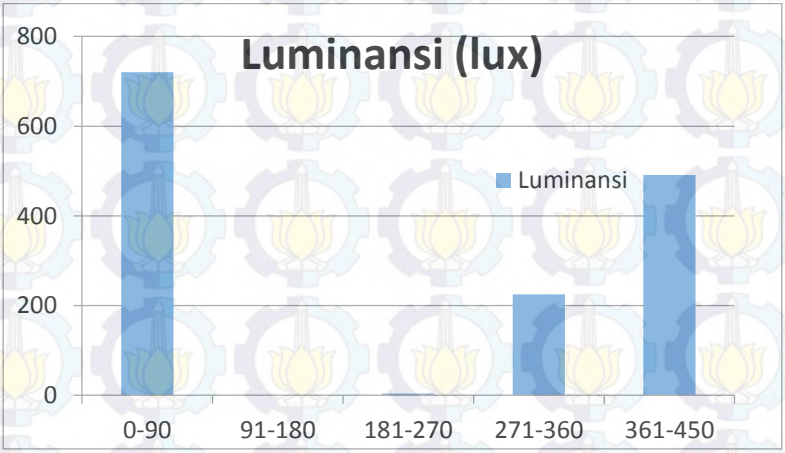
suhu (Celcius)	jumlah	probabilitas
23-24	15	1.041666667
25-26	750	52.08333333
27-28	502	34.86111111
29-30	173	12.01388889
31-32	0	0



Gambar 3.7 Suhu Terhadap Banyak Pembacaan

Tabel 3.2 Hasil pengukuran Luminansi

angka	jml	%
0 ≤ x ≤ 90	720	50
90 < x ≤ 180	0	0
180 < x ≤ 270	4	0.277778
270 < x ≤ 360	225	15.625
360 < x ≤ 450	491	34.09722

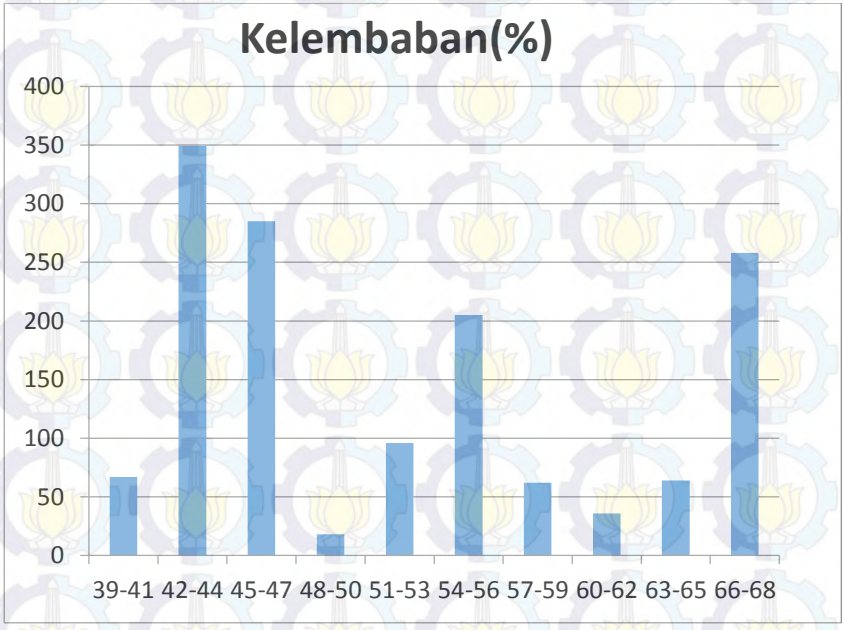


Gambar 3.8 Luminansi Terhadap Banyak Pembacaan

Setelah melakukan pengukuran terhadap suhu dan luminansi pada peralatan yang dibuat kemudian dilakukan pengukuran terhadap kelembaban yang kemudian didapatkan nilai yang paling banyak terbaca oleh sensor adalah 45%, setelah itu hasil pembacaan dikelompokkan menjadi tiga jarak nilai dan probabilitas ketiga nilai tersebut dijumlahkan seperti yang tertera pada tabel dibawah kemudian diberikan analisa hasil dari pengamatan secara statistik data yang berhasil dibaca oleh masing-masing sensor.

Tabel 3.3 Hasil Pengukuran Kelembaban

kelembaban(%)	jumlah	probabilitas
39-41	67	4.65
42-44	349	24.24
45-47	285	19.79
48-50	18	1.25
51-53	96	6.67
54-56	205	14.24
57-59	62	4.31
60-62	36	2.50
63-65	64	4.44
66-68	258	17.92



Gambar 3.9 Kelembaban Terhadap Banyak Pembacaan

Dari hasil tabel diatas kemudian dapat ditarik kesimpulan bahwa suhu, kelembaban, dan luminansi memiliki statistik sebagai berikut :

Tabel 3.4 Statistik Kelembaban

Kelembaban		
mean	53.5	%
median	53.5	%
modus	45	%
jangkauan	30	%
varians	75	%
standar deviasi	8.66	%

Tabel 3.5 Statistik Suhu

Suhu		
mean	27.5	^{o}C
median	27.5	^{o}C
modus	28	^{o}C
jangkauan	6	
varians	13.75	^{o}C
standar deviasi	3.71	^{o}C

Tabel 3.6 Statistik Luminansi

Luminansi		
mean	198.6722	lux
median	89.5	lux
modus	44.5	lux

Setelah diketahui probabilitas tiap nilai suhu, kelembaban, dan luminansi dari lingkungan yang diukur kemudian data tersebut digunakan untuk menentukan kode huffman seperti yang tertera pada Tabel 3.7, Tabel 3.8, dan Tabel 3.9. Namun dikarenakan matlab dan IDE arduino membaca angka 0 didepan angka 1 maka angka 0 tersebut

kemudian akan dihilangkan dan akan direpresentasikan dengan desimal dari angka biner tersebut.

Tabel 3.7 Kode huffman untuk nilai kelembaban

Nilai kelembaban	Kode
39~41	1101
42~44	00
45~47	10
48~50	010100
51~53	0100
54~56	111
57~59	11001
60~63	010101
64~66	1100
67~68	011

Tabel 3.8 Kode huffman untuk nilai suhu

Nilai suhu	Kode
21~22	111110
23~24	1110
25~26	0
27~28	10
29~30	110
31~32	11110
33~34	11111

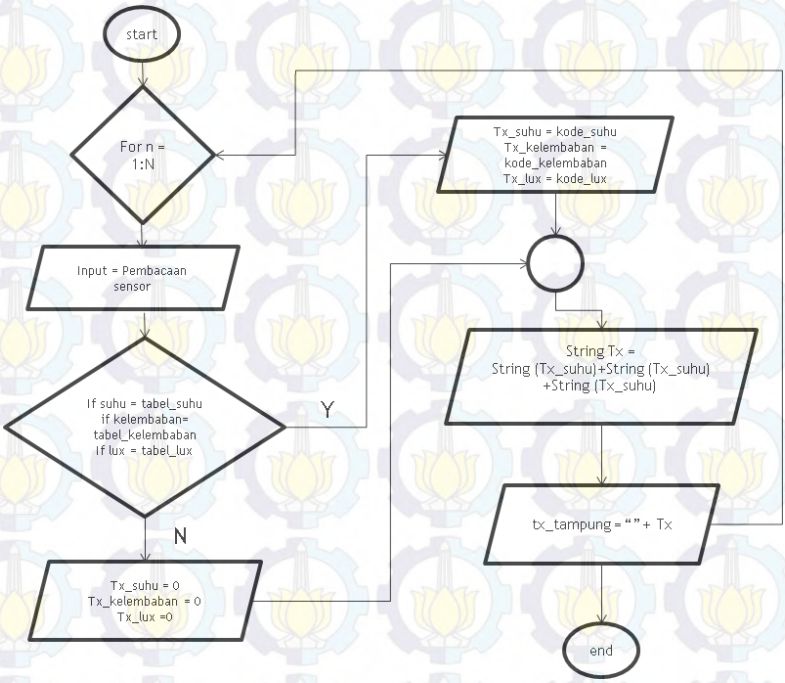
Tabel 3.9 Kode huffman untuk nilai suhu

Nilai ldr	Kode
0~90	0
91~180	1111
181~270	1110
271~360	110
361~450	10

3.5 Simulasi Kompresi pada MATLAB

Sebelum dilakukan implementasi pada perangkat keras, sebelumnya terlebih dahulu dilakukan simulasi dengan menggunakan MATLAB agar nantinya lebih mudah melakukan pemrograman pada

IDE Arduino dan akan menjadi acuan untuk mengubah program yang dibuat kedalam bahasa C. Simulasi pada MATLAB ini akan mensimulasikan dua macam cara kompresi, yaitu dengan menggunakan *look-up table* dan dengan menggunakan algoritma *huffman coding.* Secara garis besar kompresi menggunakan *look-up table* dan huffman menggunakan flowchart seperti gambar dibawah



Gambar 3.10 Flowchart Kompresi

3.5.1 simulasi Algoritma *Look-up Table* pada MATLAB

Pada tugas akhir ini akan dilakukan dua metode kompresi dan salah satunya adalah kompresi dengan cara *look-up table*. Maksud dari *look-up table* ini adalah merepresentasikan suatu nilai yang menggunakan dua *integer* dengan cara menggantinya dengan suatu simbol. Pada tugas akhir ini simbol yang diberikan adalah alfabet "a" sampai dengan "z". algoritma ini akan menjadi perbandingan untuk metode kompresi selanjutnya.

```
number=[17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
37 38 39 40 41 42];
number2=[39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57
58 59 60 61 62 63 64];
code=['a' 'b' 'c' 'd' 'e' 'f' 'g' 'h' 'i' 'j' 'k' 'l' 'm' 'n' 'o' 'p' 'q' 'r' 's' 't' 'u' 'v' 'w' 'x'
'y' 'z'];
counter=0;
inputhum=[input(1) input(2)];
nilai_hum=str2num(inputhum);

inputtemp=[input(3) input(4)];
nilai_temp=str2num(inputtemp);

inputldr=[input(5) input(6)];
nilai_ldr=str2num(inputldr);
```

Gambar 3.11 Inisialisasi Tabel dan Pemisahan Isi

Pada inisialisasi gambar diatas diasumsikan nilai suhu(pada number) berkisar dari 17^{o} C sampai dengan 42^{o} C dan kelembaban(number2) berkisar antara 39 sampai dengan 64%. Counter digunakan untuk nantinya menghitung nilai hasil pembacaan LDR yang terdapat pada jarak nilai tertentu. Tabel inilah yang kemudian akan dicocokkan dengan hasil pembacaan sensor, namun pada simulasi menggunakan MATLAB ini hasil pembacaan sensor ditentukan sendiri. Setelah itu hasil dari kompresi akan dicocokkan kembali dengan tabel dekompresi yang telah diberikan pada penerima. Pada simulasi ini *encoder* dan *decoder* terdapat dalam satu fungsi dan menggunakan tabel yang sama. Untuk melakukan kompresi dan dekompresi. Untuk menentukan nilai luminansi digunakan fungsi dibawah

```
for i=0:(nilai_ldr)
        if mod(i,90)==0
                counter=counter+1;
        end
end
```

Gambar 3.12 Fungsi Menentukan Luminansi

3.5.2 Simulasi Algoritma Huffman pada MATLAB

Pada tugas akhir ini algoritma huffman yang dipakai adalah algoritma yang tidak adaptif dikarenakan tabel kompresinya telah ditentukan sebelumnya dan nilai binernya pun dikonversikan kedalam nilai integernya (misal nilai biner 1001 menjadi 9). Nilai ini kemudian diubah menjadi nilai binernya dan digabungkan dengan nilai biner dari sensor sesudahnya. Setelah semua nilai digabungkan menjadi satu string, kemudian nilai biner tersebut diubah menjadi nilai integernya.

```
number=[21,22,23,24,25,26,27,28,29,30,31,32,33,34];
number2=[39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,5
8,59,60,61,62,63,64,65,66,67,68];
code=[62,62,14,14,0,0,2,2,6,6,30,30,31,31];
code_hum=[13,13,13,0,0,0,2,2,2,20,20,20,4,4,4,7,7,7,25,25,25,21,21,21,
12,12,12,3,3,3];
code_lux=[0,15,14,6,2];
lena=length(number);
lenc=length(number2);
lenb=length(code);
counter=0;
```

Gambar 3.13 Inisialisasi Tabel dan Pemisahan Nilai Hasil Pengukuran

Pada inisialisasi tabel diatas telah dilakukan pengukuran selama 24 jam dan diberikan nilai sesuai dengan tabel yang telah ditentukan sebelumnya menggunakan *huffman tree*. Nilai suhu(pada number) berkisar dari 21^{o} C sampai dengan 34^{o} C dan kelembaban(number2) berkisar antara 39 sampai dengan 64%. Counter digunakan untuk nantinya menghitung nilai hasil pembacaan LDR yang terdapat pada jarak nilai tertentu. Tabel inilah yang kemudian akan dicocokkan dengan hasil pembacaan sensor, namun pada simulasi menggunakan MATLAB ini hasil pembacaan sensor ditentukan sendiri. Setelah itu hasil dari kompresi akan dicocokkan kembali dengan tabel dekompresi yang telah diberikan pada penerima. Pada simulasi ini *encoder* dan *decoder* terdapat dalam satu fungsi dan menggunakan tabel yang sama. Untuk melakukan kompresi dan dekompresi. Untuk menentukan nilai luminansi digunakan fungsi seperti pada simualsi sebelumnya

3.6 Implementasi Kompresi pada Arduino Leonardo

Pada sub-bab ini penulis mengimplementasikan simulasi pada MATLAB dan mencobanya pada perangkat keras yang sudah dikonfigurasikan sebelumya dan diadaptasikan pada kondisi lingkungan yang akan dimonitoring nantinya. Oleh sebab itu pada fungsi IDE ini akan ada yang diubah dari fungsi pada MATLAB. Seperti inisialisasi pada tabel pengkodean yang akan disesuaikan dengan kondisi lingkungan.

3.6.1 Implementasi Algoritma *Look-up Table*

Pada pengimplementasian ini hampir sama dengan yang dilakukan pada simulasi, namun perbedaan terdapat pada inisialisasi tabel dan pada header. Header dihilangkan dikarenakan pada *xbee* telah dilakukan konfigurasi pengalamatan dan menyebabkan tidak diperlukannya header untuk dicantumkan agar dapat saling berkomunikasi. Selain itu diberikan juga pengulangan simbol agar dapat mencakup nilai yang lebih banyak.

```
#include <dht.h>
#define dht_dpin A0
dht DHT;

String tampung;
String buff;
int number[26]={19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32,
33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44};
int number2[40]={35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48,
49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59,
60,61,62,63,64,65,66,67,68,69,70,71,72,73,74};
char code[26]={'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q',
'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z'};
char code2[40]={'a','a','b', 'b', 'c','c','d', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm',
'n', 'o','p','q', 'q', 'r','r','s', 's', 't','t','u', 'u', 'v','v','w', 'w', 'x','x','y', 'y', 'z','z'};
char tx[3];
int counter=0;
String bin[3];
```

Gambar 3.14 inisialisasi tabel dan array

Pada gambar diatas terdapat inisialisasi tabel kompresi dan string juga array yang nantinya akan dipakai untuk melakukan kompresi. number digunakan untuk nilai suhu, number2 digunakan untuk nilai kelembaban, sedangkan pada code digunakan untuk mengubah nilai suhu yang terbaca pada sensor menjadi yang terdapat di dalam array code dan pada code2 untuk mengubah nilai kelembaban. Pada implementasi kompresi ini nilai luminansi diubah menggunakan tabel kompresi yang sama dengan nilai suhu. Pada implementasi ini tidak terdapat banyak perbedaan dari apa yang telah disimulasikan pada perangkat lunak MATLAB dan jenis data seperti String, long, char, int tidak terlalu berpengaruh pada proses kompresi.

Hasil dari pembacaan sensor kemudian akan dicocokkan dengan kodenya untuk diubah menjadi kode yang telah ditetapkan dan kemudian dijadikan satu string. Satu string tersebut kemudian disatukan kembali dengan hasil pembacaan selanjutnya sebanyak yang ditentukan pengguna dan kemudian satu string tersebut dikirimkan melalui antena *xbee*.

```
for(int j=0;j<40;j++){
  if(data[0]==number2[j]){
   tx[0]=code2[j];
  }
 }


 for (int i=0;i<26;i++){


  if(data[1]==number[i]){
    tx[1]=code[i];
  }
 }
  for(int k=0;k<data[2];k++)
  {
   if (k%77==0){
   counter=counter+1;
    }
  }
```

```
 tx[2]=code[counter-1];
 counter=0;
 bin[0]=String(tx[0]);
 bin[1]=String(tx[1]);
 bin[2]=String(tx[2]);
 buff=String(bin[0]+bin[1]+bin[2]);
 delay(50);
 tampung = tampung+ buff+",";
 delay(1000);
 }
 Serial1.print(tampung);
 delay(50);
}
```

Gambar 3.15 fungsi pengkodean *look-up table*

3.6.2 Implementasi Algoritma Huffman

Pada algoritma ini akan sangat berbeda dari apa yang disimulasikan dikarenakan terdapat banyak batasan pada IDE milik Arduino. Seperti pada saat definisi integer yang hanya dapat mengolah data sebanyak 2 byte, tidak adanya library untuk mengubah dari desimal menjadi biner atau sebaliknya, dan pengubahan kembali dari string yang panjang menjadi desimal. Karena keterbatasan ini, maka data yang telah dikompresikan akan dibagi menjadi dua bagian, yaitu 10 Byte pertama dan 10 Byte kedua kemudian keduanya dikonversikan menjadi desimal yang terpisah.

```
#include <dht.h>

#define dht_dpin A0 //set pin DHT

dht DHT;

//const int teg5 = A2; //set pin 8 sebagai sumber tegangan 5 volt
const int relay = 9; //set pin 9 sebagai aktuator
const int ldry = A1; //set pin LDR

int number[14]={21,22,23,24,25,26,27,28,29,30,31,32,33,34};
```

```
int number2[30]=
{39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61
,62,63,64,65,66,67,68};
int code_temp[14]={62,62,14,14,0,0,2,2,6,6,30,30,31,31};

int code_hum[30]=
{13,13,13,0,0,0,2,2,2,20,20,20,4,4,4,7,7,7,25,25,25,21,21,21,12,12,12,3,
3,3};
int code_lux[5]={0,15,14,6,2};
int dec_tx[35]=
{124,1007,1006,502,250,28,239,238,118,58,0,15,14,6,2,4,47,46,22,10,1
2,111,110,54,26,60,495,494,246,122,62,511,510,254,126};

unsigned long bin_tx[35]=
{1111100,1111101111,1111101110,111110110,11111010,11100,11101
111,11101110,1110110,111010,0,1111,1110,110,10,100,101111,10111
0,10110,1010,1100,1101111,1101110,110110,11010,111100,11110111
1,111101110,11110110,1111010,111110,111111111,111111110,11111
110,1111110};

unsigned long bin_tx2[10]=
{1101,0,10,10100,100,111,11001,10101,1100,11};
int dec_tx2[10]={13,0,2,20,4,7,25,21,12,3};
int tx[3];
unsigned long binh;
unsigned long bint;
unsigned long binl;
String bin[3];
unsigned long btot1;
unsigned long btot2;
String trans;
int trans1;
int trans2;
int counter=0;
String tampung;
String buff;
String str[3];
```

Gambar 3.16 inisialisasi tabel, string tampung dan array penampung

Berdasarkan pada gambar diatas, terdapat banyak sekali string dan array untuk mengubah data yang terbaca oleh sensor menjadi biner, string, kembali menjadi integer atau mengubah biner menjadi angka desimalnya. Pada implementasi menggunakan *look-up table* tidak terdapat unsigned long dikarenakan pemakaian simbol hanya menggunakan 1 Byte saja, namun pada algoritma huffman ini harus digunakan unsigned long yang dapat menampung hingga 10 Byte dikarenakan angka biner yang digunakan di *input*-kan sebagai integer. Dan untuk mengubah nilai desimal tersebut menjadi nilai biner digunakan fungsi :

```
unsigned long dec2bin(unsigned long n)
{
  unsigned long rem, i=1, binary=0;
  while (n>=1)
  {
    rem=n%2;
    n/=2;
    binary+=rem*i;
    i*=10;
  }
  return binary;
}
```

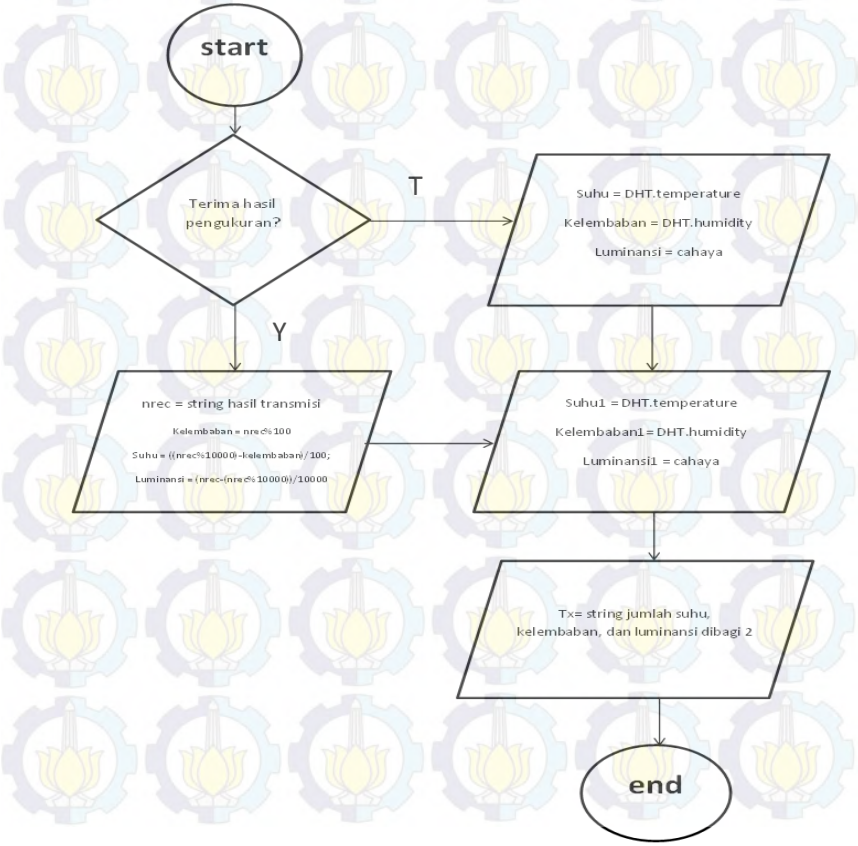
Gambar 3.17 fungsi pengubah data menjadi biner

Gambar diatas adalah fungsi untuk membuat nilai desimal menjadi nilai binernya, agar data tersebut dapat diubah kembali menjadi Selanjutnya akan digunakan fungsi yang sama dengan algoritma sebelumnya untuk menentukan nilai yang terbaca pada sensor akan diubah menjadi kode huffman yang terdapat pada tabel. Dengan membuat tipe data tersebut menjadi unsigned long maka panjang data yang dapat dimasukkan adalah 2^{32}-1 atau sebanyak 10 digit bilangan biner.

3.6.3 Implementasi Distributed Consensus

Algoritma ini sebenarnya bukan algoritma kompresi, namun algoritma untuk merata-rata hasil pengukuran sensor dan

mengirimkannya ke koordinator. Jika dilihat pada tingkat penghematan bit, maka algoritma ini juga meminimalisir pengiriman jumlah bit pada tiap node dengan merata-rata hasil pengukuran sensor kemudian dikirimkan. Namun jika algoritma ini diiterasikan berkali-kali maka pemakaian energi bisa saja lebih besar dari yang tidak memakai algoritma apapun. Oleh karena itu iterasi yang dilakukan untuk algoritma ini hanya sekali saja selain karena persoalan diatas dikarenakan juga keterbatasan dari perangkat. Flowchart dari algoritma ini dapat dilihat pada gambar dibawah.



Gambar 3.18 Flowchart *distributed consensus*

Berdasarkan flowchart diatas, maka node terlebih dahulu memeriksa apakah dia menerima dta hasil pengukuran dari node lain atau tidak, jika iya maka node akan menerima data terlebih dahulu dan menampungnya dalam satu string lalu mengubahnya kedalam bentuk integer untuk dipisahkan antara kelembaban, suhu, dan luminansinya. Jika node tidak menerima transmisi data maka dia akan mengambil data sensor terlebih dahulu untuk kemudian ditambahkan dengan hasil pembacaan sensor selanjutnya. Berikut syntax untuk mengecek transmisi data dan memisahkannya

```
while(Serial1.available() !=0)
  {
      rec=(rec)+char(Serial1.read());
  }
  delay(500);
  long int nrec=rec.toInt();
  x=nrec%100;
  y=((nrec%10000)-x)/100;
  z=(nrec-(nrec%10000))/10000;
```

Gambar 3.19 fungsi pengecekan transmisi dan memisahkannya

Setelah dipisahkan, node kemudian mengambil data dari sensornya sendiri dan menambahkannya ke hasil pemisahan tadi. Setelah itu membaginya dengan jumlah node yang terlibat, dalam hal ini node yang terlibat hanya dua. Setelah itu hasil dari rata-rata tersebut dijadikan kedalam satu string kembali dan dikirimkan untuk selanjutnya diterima koordinator atau diolah untuk dirata-rata kembali oleh node selanjutnya. Berikut syntax yang dimaksud :

```
long int hum=(x+data[0])/2;
long int temp=(y+data[1])/2;
long int lux=(z+data[2])/2;
bin[0]=String(lux); bin[1]=String(temp); bin[2]=String(hum);
buff=String(bin[0]+bin[1]+bin[2]);
  Serial1.print(buff);
```

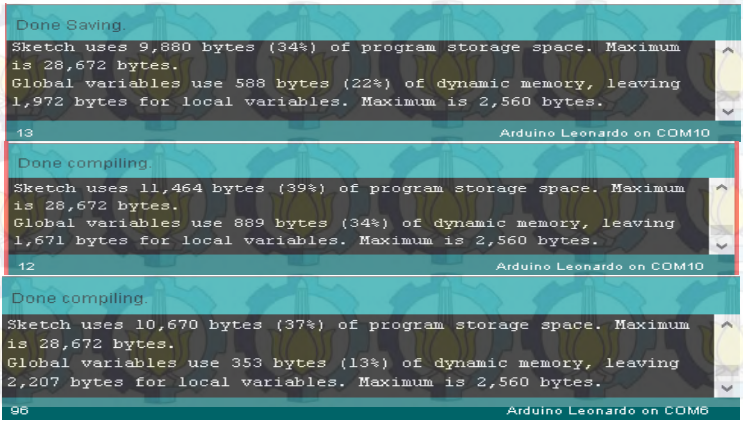
Gambar 3.20 fungsi merata-rata, penggabungan, dan pengiriman

BAB IV
ANALISIS DATA DAN PEMBAHASAN

Pada bab ini, akan dibahas mengenai analisis dari sistem yang dibuat. Pengimplementasian kompresi data pada jaringan sensor nirkabel diuji dengan beberapa hasil data dan dianalisis untuk mengetahui performa kerja terbaiknya. Berikut beberapa bahasan analisis yang terdiri dari analisis penggunaan daya pada tegangan sumber 9V, dan banyak data yang dikirim tiap satu kali pengiriman. Jumlah data yang dibangkitkan dalam satu kali pengambilan data berkisar antara 3 byte (2 Byte hasil pengukuran dan 1Byte pemisah antar hasil) hingga 8 Byte. Pada pengukuran pemakaian daya ini diasumsikan bahwa daya pancar tiap Byte sama saat memancarkan data yang telah dikompresi dan yang belum dikompresi. Dan diasumsikan juga bahwa jarak antar node tidak berpengaruh terhadap daya pancar.

4.1 Data Pengukuran

Sebelum dilakukan pengukuran penggunaan daya pada algoritma yang telah ditetapkan, dilakukan terlebih dahulu perbandingan antara kompresi menggunakan metode huffman, kompresi menggunakan metode *look-up table*, dan yang tidak menggunakan kompresi. parameter yang dibandingkan adalah pemakaian memori untuk penyimpanan, memori untuk pemrosesan, dan berapa persentasi Byte yang terkompresi menggunakan algoritma tersebut.



Gambar 4.1 Pemakaian memori dan RAM tiap algoritma

Pada gambar diatas terlihat bahwa algoritma *look-up table* (gambar pertama) lebih sedikit menggunakan memori penyimpanan dan RAM jika dibandingkan dengan algoritma huffman (gambar kedua).Hal ini dapat terlihat langsung dari banyaknya baris program yang dipakai oleh kedua algoritma. Pada algoritma pertama, digunakan 64 baris sudah termasuk inisialisasi tabel dan fungsi untuk mengubah nilai hasil pembacaan sensor menjadi nilai hasill kompresi sedangkan pada algoritma huffman menggunakan 142 baris untuk kodenya. Penggunaan RAM yang besar juga dikarenakan algoritma huffman menggunakan fungsi FOR lebih banyak daripada algoritma sebelumnya dan banyak terdapat pengubahantipe data dari integer ke string, diubah kembali ke integer lalu disatukan kembali dalam bentuk string lalu dikirimkan secara serial. Sedangkan untuk algoritma *distributed consensus* menggunakan sedikit lebih banyak memori jika dibandingkan dengan *look-up* table namun memakai jumlah RAM yang jauh lebih kecil dibandingkan dengan semua algoritma lainnya.

Jika dilihat pada jumlah Byte yang dihasilkan, pada algoritma *look-up table* akan menghasilkan 3 Byte hasil pengukuran dari 7 Byte yang akan dikirim dan belum termasuk pemisah antar hasil pengukurannya. Sedangkan pada algoritma huffman dapat memberikan 7 Byte pada kemungkinan terburuk dan 2 Byte pada kemungkinan terbaik. Hasil 7 Byte didapatkan dikarenakan pada inisialisasi tabel terdapat empat digit angka setelah hasil penggabungan biner antara biner suhu dan biner lux dan kemudian string tersebut ditambah dengan dua digit angka hasil pengubahan bentuk biner pada kelembaban kemudian ditambah satu digit pemisah. Sedangkan 2 Byte didapatkan dengan hasil 0 pada kelembaban yang kemudian direpresentasikan sebagai 0 dan nilai 00 dari hasil penggabungan angka biner dari suhu dan lux yang kemudian juga dibaca sebagai 0 oleh IDE, kemudian hasil tersebut digabungkan menjadi satu string dan akan terbaca menjadi 00. Sedangkan pada algoritma *distributed consensus* jumlah Byte yang dapat dihemat adalah setengahnya karena node yang terlibat adalah dua buah node. Jika diterapkan rumus yang terdapat pada bab 2 dan dijadikan tabel perbandingan maka persentase jumlah Byte yang dikirim setelah dilakukan kompresi terhadap data aslinya maka akan terlihat seperti :

Tabel 4.1 Tabel Perbandingan Jumlah Byte

Jenis Kompresi	Rasio Kompresi
Look-up table	2.3333 : 1
Huffman	Min 1 : 1 & Max. 3.5 : 1
Distributed Consensus	2 : 1

Tabel 4.2 Tabel Perbandingan Jumlah Byte Pada Simulasi

Jenis Kompresi	Rasio Kompresi
Look-up table	2.3333 : 1
Huffman	Min 1.75 : 1 & Max. 7 : 1

4.2 Analisa Data

Setelah diketahui berapa jumlah data yang dapat dihemat dengan menggunakan kedua algoritma tersebut, maka harus juga diketahui berapa jumlah daya yang digunakan untuk mengirim hasil kompresi tersebut secara nirkabel ke koordinator. Parameter yang diukur pada tugas akhir ini adalah penggunaan daya pada catu daya terbatas yang berbeda yaitu baterai kotak 9V dan *power bank* dengan kapasitas 6000 mAh.

4.2.1 Data Penggunaan Daya

Dalam sub bab ini akan dilakukan perbandingan penggunaan daya yang menggunakan *power bank* dan yang tidak menggunakan *power bank*. Dan pada bagian yang tidak menggunakan *power bank* akan dilakukan perbandingan pemakaian daya pada kompresi data menggunakan *huffman coding* dan *look-up table* dengan skenario 40, 25, dan 15 pembacaan data tiap pengiriman informasi. Hal yang sama tidak dilakukan pada pemakaian daya yang menggunakan *power bank* dikarenakan alat ukur yang digunakan tidak terlalu akurat. Dan pada *distributed consensus* tidak dilakukan skenario diatas dikarenakan sulitnya memisahkan data yang banyak pada pengiriman yang banyak.

4.2.1.1 Penggunaan Daya Menggunakan *Power Bank*

Pemakaian daya pada skenario ini akan diukur menggunakan alat yang bernama *charger doctor*. Permasalahannya adalah alat ukur ini tidak seperti wattmeter yang dapat mengukur tegangan dan arus dengan teliti. Ketelitian dari alat ukur yang dipakai ini hanya sampai 0.01 A dan

0.01 V saja sehingga membuat data pengukuran tidak terlalu akurat. Pada skenario ini digunakan pengiriman data 15 kali pengambilan data.

Tabel 4.3 Statistik Penggunaan Daya pada *power bank*

Daya (Watt)	Tanpa Kompresi	*look-up Table*	Huffman
Mean	0.451470139	0.449229097	0.445804
Jangkauan	0.0512	0.0512	0.512
Varians	0.000104901	0.000197754	0.00032516
standar deviasi	0.010242109	0.014062495	0.0180322

Dapat dilihat pada tabel bahwa jangkauan dari ketiga skenario pengiriman data adalah sama, hal ini dikarenakan nilai maksimun dan nilai minimum dari ketiga skenario tersebut juga sama yaitu 0.4535 Watt pada maksimum dan 0.4024 Watt pada nilai minimum. Hal seperti ini terjadi dikarenakan kurang presisinya alat ukur yang digunakan pada pengukuran ini dan mengakibatkan varians dan standar deviasi menjadi sangat kecil juga. Dengan kecilnya standar deviasi dan varians membuktikan bahwa data yang diambil tidak valid berdasarkan statistik.

4.2.1.2 Perbandingan Kompresi 40 Pembacaan Data Tiap Transmisi

Pada pengiriman 40 data hasil pembacaan sensor selama 40 detik terlihat bahwa setelah 44 detik terjadi penurunan arus yang cukup drastis selama 8 detik dan terjadi secara periodik. Hal ini disebabkan data yang diterima kurang dari jumlah data yang dikirim yaitu pada pengiriman yang tidak memakai kompresi jumlah data yang diterima adalah 320 Byte sedangkan pada algoritma *look-up table* jumlah data yang diterima adalah 160 Byte dan pada algoritma huffman 160 Byte. Pada algoritma huffman dihasilkan 120 Byte dikarenakan pada saat pengukuran didapati tiap hasil pembacaan sensor diubah menjadi 2 Byte dan ditambah satu byte sebagai pemisah antar data dan kemudian dikirimkan menggunakan perangkat Xbee Hasil dari penggunaan daya ini dapat dilihat pada lampiran.

Tabel 4.4 Statistik Penggunaan Daya 40 Data per Transmisi

Daya (Watt)	tanpa kompresi	*look-up table*	huffman
mean	0.608317221	0.609037805	0.580598699
jangkauan	0.104165	0.0955	0.081665
varians	0.000528366	0.000490703	0.000332266
standar deviasi	0.022986213	0.022151819	0.018228167

Peristiwa ini berhubungan dengan perangkat Xbee yang memiliki batas pengiriman yaitu 153 Byte dalam satu kali pengiriman. Setelah melakukan pengiriman kemudian arduino melakukan *reboot* dan menyebabkan penurunan arus pada saat booting tersebut terjadi penurunan pemakaian arus, namun menyebabkan sedikit kenaikan pada tegangan. Perubahan arus ini hanya sedikit mempengaruhi pemakaian daya pada peralatan sensor nirkabel. Pada skenario ini pemakaian daya rata-rata paling kecil adalah 0.609694997 Watt menggunakan algoritma huffman dan pemakaian daya rata-rata paling besar adalah 0.609694997 Watt tanpa menggunakan kompresi.

4.2.1.3 Perbandingan Kompresi 25 Pembacaan Data Tiap Transmisi

Pengiriman dengan metode ini tidak terjadi penurunan tegangan seperti yang terjadi pada pengukuran menggunakan 40 rentetan hasil pembacaan sensor sebelumnya. Namun hal itu terjadi hanya pada algoritma *look-up table* dan huffman saja sedangkan pada skenario tanpa menggunakan kompresi tetap terjadi penurunan arus yang drastis dengan penurunan tiap 44 detik dan terjadi selama 8 detik. Pada pengukuran ini pemakaian daya pada skenario tanpa kompresi hampir sama dari pemakaian daya tanpa kompresi pada skenario sebelumnya.

Pada skenario ini jumlah Byte tiap pengiriman yang dikirim oleh perangkat sama dengan jumlah Byte sebelumnya yaitu 8 Byte tanpa menggunakan kompresi, 4 Byte dengan menggunakan simbol, dan 3 Byte untuk algoritma huffman. Jika ditotal tiap satu kali pengiriman maka untuk tanpa kompresi mengirimkan 200 Byte, sedangkan dengan menggunakan simbol mengirimkan 100 Byte tiap pengiriman dan algoritma huffman mengirimkan 75 Byte tiap pengiriman. Penggunaan daya dapat dilihat pada gambar.

Tabel 4.5 Statistik Penggunaan Daya 25 Data per Transmisi

Daya (Watt)	Tanpa Kompresi	*Look-upTable*	Huffmann
mean	0.610347267	0.554607408	0.525422092
jangkauan	0.097448	0.048192	0.116512
varians	0.00049102	0.000109875	0.000290108
standar dev.	0.022158964	0.010482148	0.017032565

Pada skenario ini perbedaan penggunaan daya lebih terlihat pada skenario ini dikarenakan pada penggunaan tanpa kompresi mengirim lebih banyak dari yang dapat dikirimkan oleh Xbee dan menambah kebutuhan daya untuk melakukan *reboot*. Penggunaan daya rata-rata paling kecil tetap berasal dari metode huffman dengan daya 0.525332835 Watt dan penggunaan daya rata-rata terbesar adalah yang tidak menggunakan kompresi sama sekali yaitu 0.610356709 Watt.

4.2.1.4 Perbanding Kompresi 15 Pembacaan Data Tiap Transmisi

Pada skenario ini hasil pengukuran tidak terjadi penurunan arus seperti pada skenario sebelumnya. Namun hasil pemakaian daya hampir sama antara menggunakan algoritma huffman dan *look-up table*. Hal ini disebabkan karena pada skenario ini tidak terdapat kelebihan data yang dikirim oleh Xbee seperti pada skenario pertama dan skenario kedua. Pada skenario ini jumlah data yang dikirim adalah 8 Byte tiap pengiriman untuk yang tidak menggunakan kompresi, lalu 4 Byte menggunakan simbol, dan 3 Byte untuk yang menggunakan huffman. Hasil penggunaan daya rata-rata dapat dilihat pada gambar.

Tabel 4.6 Statistik Penggunaan Daya 15 Data per Transmisi

Daya (Watt)	Tanpa Kompresi	*Look-up Table*	Huffmann
mean	0.582047617	0.550116875	0.502233092
jangkauan	0.083791	0.052495	0.073254
varians	0.00034902	0.000122492	0.000166399
standar deviasi	0.018682089	0.011067591	0.012899557

Pada skenario ini perbedaan penggunaan daya lebih terlihat pada tiap penggunaan algoritma dan tidak menggunakan kompresi. penggunaan daya rata-rata jika dibandingkan dengan dua skenario sebelumnya merupakan penggunaan yang paling sedikit dan jika dilihat penggunaan daya rata-rata paling kecil tetap berasal dari metode huffman dengan daya 0.502173981 Watt dan penggunaan daya rata-rata terbesar adalah yang tidak menggunakan kompresi sama sekali yaitu 0.582069023 Watt. Pada skenario ini jumlah Byte tiap pengiriman yang dikirim oleh perangkat sama dengan jumlah Byte sebelumnya yaitu 8 Byte tanpa menggunakan kompresi, 4 Byte dengan menggunakan simbol, dan 3 Byte untuk algoritma huffman. Jika ditotal tiap satu kali pengiriman maka untuk tanpa kompresi mengirimkan 120 Byte, sedangkan dengan menggunakan simbol mengirimkan 60 Byte tiap pengiriman dan algoritma huffman mengirimkan 45 Byte tiap pengiriman.

4.2.1.5 Perbandingan Daya Menggunakan Distributed Consensus

Pada sub-bab ini skenario yang digunakan adalah mengirimkan hanya satu data pengukuran dikarenakan jika dikirim banyak data pengukuran dalam sekali pengiriman akan semakin sulit bagi mikrokontroler untuk memisahkan mana yang data suhu, kelembaban, dan luminansi yang nantinya akan digunakan untuk menentukan rata-rata. Pada skenario ini juga hanya menggunakan baterai kotak 9V saja dikarenakan data yang diambil dari skenario sebelumnya yang menggunakan *power bank* tidak begitu akurat. Untuk pengukuran pemakaian daya, pada skenario ini menggunakan dua node sensor yang masing-masing diprogram menggunakan metode *distributed consensus*. Hasil dari pengukuran dapat dilihat pada tabel dibawah

Tabel 4.7 Statistik Penggunaan Daya pada *Distributed consensus*

Daya(W)	Node 1	Node 2	Rata-rata
Rata-rata	0.512663101	0.516879714	0.514771
Jangkauan	0.093779	0.074734	0.084257
Varians	0.000742312	0.000431608	0.000587
Standar dev.	0.027245399	0.020775167	0.02401

Dapat dilihat bahwa pada skenario ini penggunaan daya terlihat lebih sedikit jika dibandingkan dengan *look-up table* yang rata-rata pemakaian daya skenario ini adalah 0.514771 Watt. Jangkauan, varians, standar deviasi, dan penggunaan daya pada tiap node hampir sama dikarenakan algoritma yang digunakan juga sama. Namun pengiriman data ini hanyalah sekitar 8 Byte. Penggunaan daya yang besar ini dikarenakan pengiriman yang diinginkan iterasinya hanya satu kali ternyata teriterasikan kembali menjadi berkali-kali dan dikarenkan juga pengiriman serial yang tidak sempurna membuat pembacaan pada penerima menjadi tidak benar dan mengubah hasil dari semua proses dari skenario ini.

4.2.2 Rata-Rata Perbandingan Penggunaan Daya

Sub-bab ini bertujuan untuk membandingkan semua hasil dari skenario yang dilakukan pada sub-bab 4.2 dan berdasarkan pada data sebelumnya, semakin sedikit jumlah data dalam satu kali pengiriman maka semakin sedikit energi yang dipakai tiap detiknya. Hal ini dapat terlihat pada tabel dibawah. Penggunaan daya pada skenario 4.2.1.5 tidak dimasukkan pada sebagian besar perbandingan karena tidak dikirimkan secara kumulatif.

Tabel 4.8 Tabel Perbandingan Penggunaan Daya Rata-rata

Daya Rata-rata (Watt)			
Jenis Skenario	Tanpa Kompresi	*Look-up table*	Huffman
40 data	0.609694997	0.566942037	0.549914
25 data	0.610356709	0.554610372	0.525333
15 data	0.582069023	0.550112124	0.502174

Tabel 4.9 Tabel Perbandingan Penggunaan Daya Minimun

Daya Minimum (Watt)			
jenis skenario	tanpa kompresi	*look-up table*	huffman
40 data	0.535947	0.501313	0.492072
25 data	0.537309	0.53244	0.497872
15 data	0.516795	0.52765	0.479892

Tabel 4.10 Tabel Perbandingan Penggunaan Daya Maksimum

Daya Maksimum (Watt)			
jenis skenario	tanpa kompresi	*look-up table*	huffman
40 data	0.667725	0.615896	0.549914
25 data	0.672276	0.602272	0.525333
15 data	0.62652	0.595335	0.502174

Jika dilihat dari tabel diatas maka penggunaan daya pada perangkat yang tidak menggunakan kompresi berkisar antara 0.537 Watt sampai dengan 0.672. Sedangkan untuk perangkat yang menggunakan simbol daya yang digunakan berkisar dari 0.5 Watt sampai dengan 0.61 Watt dan untuk algoritma huffman berkisar dari 0.479 Watt sampai dengan 0.549 Watt. Jika dilihat dari persentase efisiensi daya yang menggunakan rumus dibawah dan dibandingkan maka akan menjadi :

$$\text{Efisiensi} = \frac{\text{Daya awal} - \text{daya kompresi}}{\text{Daya awal}} \text{ X } 100\%$$

Tabel 4.11 Perbandingan Rasio Kompresi Dengan Efisiensi

Jenis Kompresi	Rasio Kompresi	Efisiensi Rata-rata
Look-up table	2.3333 : 1	7.23903753 %
Huffman	Min 1 : 1 & Max. 3.5 : 1	13.44170648 %
Distributed Consensus	2 : 1	11.92 %

Daya (Watt)	tanpa kompresi			*look-up table*			huffman		
	40 data	25 data	15 data	40 data	25 data	15 data	40 data	25 data	15 data
mean	0.609686	0.610347	0.582048	0.566979	0.554607	0.550117	0.549936	0.525422	0.502233
jangkauan	0.104165	0.097448	0.083791	0.091439	0.048192	0.052495	0.12142	0.116512	0.073254
varians	0.00053	0.000491	0.000349	0.000308	0.00011	0.000122	0.000223	0.00029	0.000166
standar deviasi	0.023015	0.022159	0.018682	0.01754	0.010482	0.011068	0.014935	0.017033	0.0129

Gambar 4.2 Statistik Penggunaan Daya pada Masing-masing Skenario

BAB V
KESIMPULAN

Setelah rangkaian penentuan parameter dan perancangan yang telah dilakukan dianalisis maka akan dapat diberikan kesimpulan. Pembahasan dari bab-bab sebelumnya dan kendala-kendala yang terjadi selama pengerjaan tugas akhir ini diharapkan akan menjadi bahan pertimbangan atau referensi dalam melakukan pengembangan dari tugas akhir ini dan membantu membuat alternatif pilihan dalam mengembangkan penerapan jaringan sensor nirkabel menggunakan perangkat yang digunakan dalam tugas akhir ini.

5.1 Kesimpulan

Dari tugas akhir ini dapat ditarik beberapa kesimpulan, yaitu :

1. Kompresi data pada algoritma huffman dapat menghemat lebih banyak daya jika jatuh pada nilai yang memiliki jumlah byte pengiriman lebih sedikit daripada menggunakan simbol.
2. Rasio kompresi yang dapat dicapai jika menggunakan simbol adalah 2.3333 : 1 dan jika menggunakan algoritma huffman adalah Min 1 : 1 & Max. 3.5 : 1. Sedangkan algoritma *distributed consensus* 2 : 1.
3. Hasil efisiensi penggunaan daya rata-rata setelah dilakukan kompresi adalah 7.23903753 % jika menggunakan look-up table dan 13.44170648 % jika menggunakan algoritma huffman Sedangkan untuk *distributed consensus* dapat menghemat 11.92 %.
4. Batas Byte yang dapat dikirim oleh Xbee adalah 153 Byte dalam satu kali pengiriman data.
5. Semakin sedikit jumlah Byte yang dikirimkan maka semakin sedikit pula energi yang digunakan untuk mengirim, dan semakin besar jumlah Byte yang dikirimkan maka akan semakin besar energi yang digunakan juga akan membuat data yang dikirim tidak lengkap atau terpotong.

5.2 Saran

1. Untuk membentuk kompresi data yang lebih efisien diperlukan *mikrokontroler* yang lebih baik seperti TelosB untuk menyimpan tabel kompresi dan mengolah data yang lebih banyak.
2. Dengan banyaknya keterbatasan perangkat yang digunakan mungkin pembuatan tabel kompresi dan kompresi yang digunakan dapat dikembangkan lebih lanjut menggunakan *mikrokontroler* selain Arduino Leonardo.
3. Selain itu, jika kedepannya ingin menggunakan algoritma huffman sebaiknya dilakukan dengan *adaptive encoding* agar lebih daapat menyesuaikan dengan kondisi lingkungan dan perubahan musim..
4. Penerapan sistem monitoring pada tugas akhir ini masih belum melakukan penghitungan estimasi lama baterai dan belum melakukan penghitungan pemakaian energi secara mendetail..
5. Algoritma dekompresi masih belum dilakukan secara otomatis dan harus dilakukan manual, jadi untuk kedepannya dapat ditambahkan algoritma dekompresi pada koordinator agar dapat langsung mengetahui apakah kompresi yang dilakukan sudah benar atau belum.

LAMPIRAN

A. LEMBAR PENGESAHAN PROPOSAL TUGAS AKHIR

Jurusan Teknik Elektro
Fakultas Teknologi Industri

TE-141509 TUGAS AKHIR – 4 SKS

11 FEB 2015

Nama Mahasiswa	: Muhammad Naufal Farid
Nomor Pokok	: 2211100162
Bidang Studi	: Telekomunikasi Multimedia
Tugas Diberikan	: Semester Genap 2014/2015
Dosen Pembimbing	: 1. Dr. Ir. Wirawan, DEA 2. Dr. Istas Pratomo ST. MT.
Judul Tugas Akhir	: Penerapan Kompresi Data untuk Efisiensi Daya Jaringan Sensor Nirkabel (Implementation of Data Compression for Wireless Sensor Network)

Uraian Tugas Akhir

Jaringan sensor nirkabel memiliki potensi yang sangat bagus untuk diterapkan pada banyak aspek seperti pada militer, penanggulangan bencana, penelitian tempat berbahaya, dan masih banyak lagi. Pada sensor yang berdekatan, data yang diinginkan kebanyakan adalah memiliki korelasi yang tinggi. Dikarenakan korelasi yang tinggi tersebut kemudian menimbulkan pemikiran baru dikarenakan banyak data yang sama dan dapat dikompresikan dengan cara yang sederhana pula sehingga dapat menurunkan waktu untuk *sensing* dan dapat lebih menghemat daya. Untuk mewujudkan hal tersebut maka digunakan metode kompresi data. Metode ini terdiri dari tiga tahap dan dapat juga digunakan untuk data *rate control*. Pada tahap pertama, *sensing* data satu dimensi dikelompokkan menjadi blok berukurn 8x8 dan diubah menjadi data dua dimensi. Pada tahap kedua, data dua dimensi dipisah menjadi dua bagian yaitu *upper part* dan *lower* part. Dan pada tahap terakhir tiap bagian yang dipisah kemudian dikompres menggunakan DCT dua dimensi. Pada tugas akhir ini, teknik kompresi yang digunakan termasuk transformasi data dimensi, pemisahan data, FDTC, kuantisasi, koding entropi untuk membangun kembali *frame*, dan transmisi data. Selain akan disimulasikan, metode yang dipelajari ini akan diimplementasikan pada perangkat jaringan sensor nirkabel berbasis arduino. Perangkat yang akan dipakai adalah arduino uno, xbee shield, xbee, X-CTU, dan IDE.

Dosen Pembimbing I,

9/2/15

Dr. Ir. Wirawan, DEA.
NIP. 196311091989031011

Dosen Pembimbing II,

Dr. Istas Pratomo ST. MT.
NIP. 197903252003121001

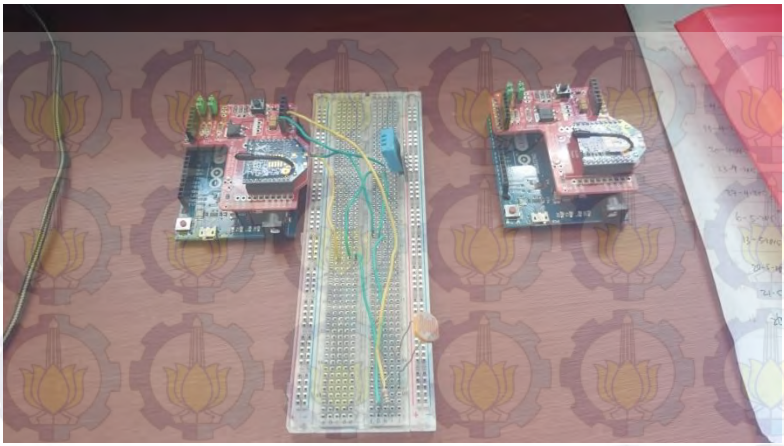
Mengetahui,
Jurusan Teknik Elektro FTI – ITS
Ketua,

Dr. Tri Arief Sardjono, S.T., M.T.
NIP. 197002121995121001

Menyetujui,
Bidang Studi Telekomunikasi Multimedia
Koordinator,

Dr. Ir. Endroyono, DEA
NIP. 196504041991021001

B. BENTUK FISIK ALAT



C. LISTING PROGRAM ARDUINOTANPA KOMPRESI

```
#include <dht.h>
#define dht_dpin A0 //set pin DHT
dht DHT;
//const int teg5 = A2; //set pin 8 sebagai sumber tegangan 5 volt
const int relay = 9; //set pin 9 sebagai aktuator
const int ldry = A1; //set pin LDR
String buff;
String bin[3];

String tampung;
void setup() {
  // put your setup code here, to run once:
//Serial.begin(9600);
Serial1.begin(9600);
  //pinMode(teg5, OUTPUT);
  pinMode(relay, OUTPUT); //disediakan jika dibutuhkan aktuator
  pinMode(ldry, INPUT),
  //analogWrite(teg5, 1023);
  digitalWrite(relay, LOW);
}

void loop() {
```

```
  // put your main code here, to run repeatedly:
int data[3] = {00, 00, 00}; //frame data [NId, Kelembapan, Suhu, Cahaya]
  tampung = "";
  for (int i=0;i<25;i++)
  {
  DHT.read11(dht_dpin);
  int sensorValue = analogRead(ldry);
  int cahaya = sensorValue*(1000.0/2048.0) ; // sensor cahaya
  delay(50);
  data[0] = DHT.humidity ; //input data kelembapan ke array
  data[1] = DHT.temperature ;//input data suhu ke array
  data[2] = cahaya ; // input data cahaya ke array
  bin[0]=String(data[0]);
  bin[1]=String(data[1]);
  bin[2]=String(data[2]);
  buff=String(bin[0]+bin[1]+bin[2]);
  delay(100);
  tampung = tampung+ buff+",";
  delay(2000);
  }
  //Serial.print(tampung);
  Serial1.print(tampung);
}
```

D. LISTING PROGRAM ARDUINO *LOOK-UP TABLE*

```
//inisialisasi tabel
//sensor + aktuator

#include <dht.h>
#define dht_dpin A0 //set pin DHT
dht DHT;

//const int teg5 = A2; //set pin 8 sebagai sumber tegangan 5 volt
const int relay = 9; //set pin 9 sebagai aktuator
const int ldry = A1; //set pin LDR
String tampung;
String buff;
```

```
int
number[26]={19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,3
7,38,39,40,41,42,43,44};
int
number2[40]={34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,
52,53,54,55,56,57,58,59,60,61,62,63,64,65,66,67,68};
char code[26]={'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q',
'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z'};
char code2[40]={'a','a','b', 'b', 'c','c','d', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm',
'n', 'o','p','q', 'q', 'r','r','s', 's', 't','t','u', 'u', 'v','v','w', 'w', 'x','x','y', 'y', 'z','z'};
char tx[3];
int counter=0;
String bin[3];

void setup(){
 Serial1.begin(9600);
 //pinMode(teg5, OUTPUT);
 pinMode(relay, OUTPUT); //disediakan jika dibutuhkan aktuator
 pinMode(ldry, INPUT),
 //analogWrite(teg5, 1023);
 digitalWrite(relay, LOW);
}

void loop(){
 int data[3] = {00, 00, 00}; //frame data [NId, Kelembapan, Suhu,
Cahaya]
 tampung = "";
 for (int i=0;i<15;i++)
 {
 counter=0;
 DHT.read11(dht_dpin);
 int sensorValue = analogRead(ldry);
 int cahaya = sensorValue*(1000.0/2048.0) ; // sensor cahaya
 delay(50);
 data[0] = DHT.humidity ; //input data kelembapan ke array
 data[1] = DHT.temperature ;//input data suhu ke array
 data[2] = cahaya ; // input data cahaya ke array
 for(int j=0;j<40;j++){
  if(data[0]==number2[j]){
```

```
    tx[0]=code2[j];
   }
  }
  for (int i=0;i<26;i++){
   if(data[1]==number[i]){
    tx[1]=code[i];
   }
  }
   for(int k=0;k<data[2];k++)
   {
    if (k%90==0){
    counter=counter+1;
    }
   }
  tx[2]=code[counter-1];
  counter=0;
  bin[0]=String(tx[0]);
  bin[1]=String(tx[1]);
  bin[2]=String(tx[2]);
  buff=String(bin[0]+bin[1]+bin[2]);
  delay(100);
  tampung = tampung+ buff+",";
  delay(2000);
  }
  Serial1.print(tampung);
  delay(50);
}
```

E. LISTING PROGRAM ARDUINO HUFFMAN CODING

```
//inisialisasi tabel
//sensor + aktuator

#include <dht.h>
#define dht_dpin A0 //set pin DHT
dht DHT;

//const int teg5 = A2; //set pin 8 sebagai sumber tegangan 5 volt
const int relay = 9; //set pin 9 sebagai aktuator
const int ldry = A1; //set pin LDR
```

```
int number[14]={21,22,23,24,25,26,27,28,29,30,31,32,33,34};
int
number2[30]={39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,
57,58,59,60,61,62,63,64,65,66,67,68};
int code_temp[14]={62,62,14,14,0,0,2,2,6,6,30,30,31,31};
int
code_hum[30]={13,13,13,0,0,0,2,2,2,20,20,20,4,4,4,7,7,7,25,25,25,21,2
1,21,12,12,12,3,3,3};
int code_lux[5]={0,15,14,6,2};
int
dec_tx[35]={124,1007,1006,502,250,28,239,238,118,58,0,15,14,6,2,4,4
7,46,22,10,12,111,110,54,26,60,495,494,246,122,62,511,510,254,126};
unsigned                                                    long
bin_tx[35]={1111100,1111101111,1111101110,111110110,11111010,1
11100,11101111,11101110,1110110,111010,0,1111,1110,110,10,100,10
1111,101110,10110,1010,1100,1101111,1101110,110110,11010,11110
0,111101111,111101110,11110110,1111010,111110,111111111,11111
1110,11111110,1111110};
unsigned                                                    long
bin_tx2[10]={1101,0,10,10100,100,111,11001,10101,1100,11};
int dec_tx2[10]={13,0,2,20,4,7,25,21,12,3};
int tx[3];
unsigned long binh;
unsigned long bint;
unsigned long binl;
String bin[3];
unsigned long btot1;
unsigned long btot2;
String trans;
int trans1;
int trans2;
int counter=0;
String tampung;
String buff;
String str[3];

unsigned long dec2bin(unsigned long n)  /* Function to convert decimal
to binary.*/
```

```
{
   unsigned long rem, i=1, binary=0;
   while (n>=1)
   {
      rem=n%2;
      n/=2;
      binary+=rem*i;
      i*=10;
   }
   return binary;
}

void setup(){
 Serial1.begin(9600);
 //pinMode(teg5, OUTPUT);
 pinMode(relay, OUTPUT); //disediakan jika dibutuhkan aktuator
 pinMode(ldry, INPUT),
 //analogWrite(teg5, 1023);
 digitalWrite(relay, LOW);
}

void loop(){


 int data[3] = {00, 00, 00}; //frame data [NId, Kelembapan, Suhu,
Cahaya]

 tampung = "";
 for (int i=0;i<15;i++)
 {
 counter=0;
 DHT.read11(dht_dpin);
 int sensorValue = analogRead(ldry);
 int cahaya = sensorValue*(1000.0/2048.0) ; // sensor cahaya
 delay(50);
 data[0] = DHT.humidity ; //input data kelembapan ke array
 data[1] = DHT.temperature ;//input data suhu ke array
 data[2] = cahaya ; // input data cahaya ke array
```

```
delay(100);


 for(int k=0;k<40;k++)
 {
 if (data[0]==number2[k]){
 tx[0]=code_hum[k];
 }
 }
 for(int k=0;k<14;k++)
 {
 if (data[1]==number[k]){
   tx[1]=code_temp[k];
 }
 }
 if (data[2]==0)
 {
   counter=1;
 }
 else{
   for(int k=0;k<data[2];k++)
 {
 if (k%90==0){
 counter=counter+1;
 }
 }
 }
 tx[2]=code_lux[counter-1];

 counter=0;

 binh=dec2bin(tx[0]);
 bint=dec2bin(tx[1]);
 binl=dec2bin(tx[2]);

 bin[0]=String(binh);
 bin[1]=String(bint);
 bin[2]=String(binl);
```

```
str[1]=String(bin[1]+bin[2]);
str[0]=String(bin[0]);

unsigned long a=str[1].toInt();
unsigned long b=str[0].toInt();

for (int i=0;i<35;i++)
{
 if (a==bin_tx[i])
 {
  trans1=dec_tx[i];
 }

}
for (int i=0;i<10;i++)
{
 if (b==bin_tx2[i])
 {
  trans2=dec_tx2[i];
 }
}

String y= String(trans1);
String x= String(trans2);

buff=String(x+y);
tampung = tampung+ buff+",";
delay(1000);
}
Serial1.print(tampung);
delay(50);
}
```

F. LISTING PROGRAM PENERIMA

```
void setup() {
```

```
  // put your setup code here, to run once:
Serial.begin(9600);
Serial1.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:
while (Serial1.available() !=0)
{
  String rec="";

  rec=rec+char(Serial1.read());
  Serial.print(rec);
}

}
```

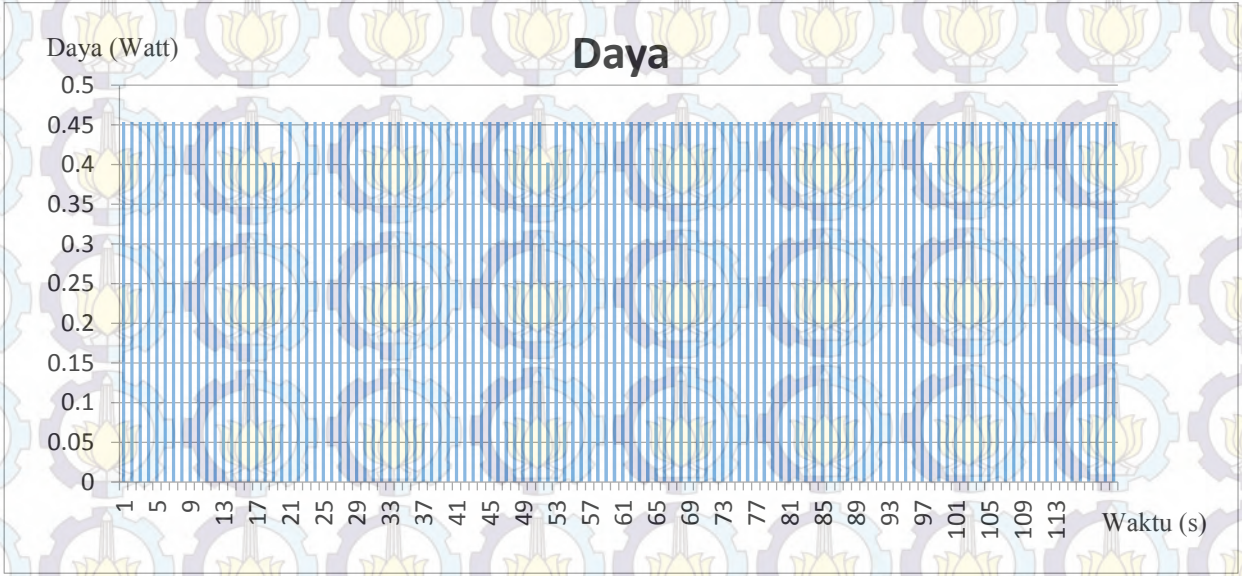
G. LISTING PROGRAM DISTRIBUTED CONSENSUS

```
#include <dht.h>
#define dht_dpin A0 //set pin DHT
dht DHT;


int count = 0;
const int relay = 9; //set pin 9 sebagai aktuator
const int ldry = A1; //set pin LDR

void setup() {
Serial.begin(9600);
Serial1.begin(9600);
}

void loop() {
  String rec="";

int data[3] = {00, 00, 00};
  DHT.read11(dht_dpin);
  int sensorValue = analogRead(ldry);
```

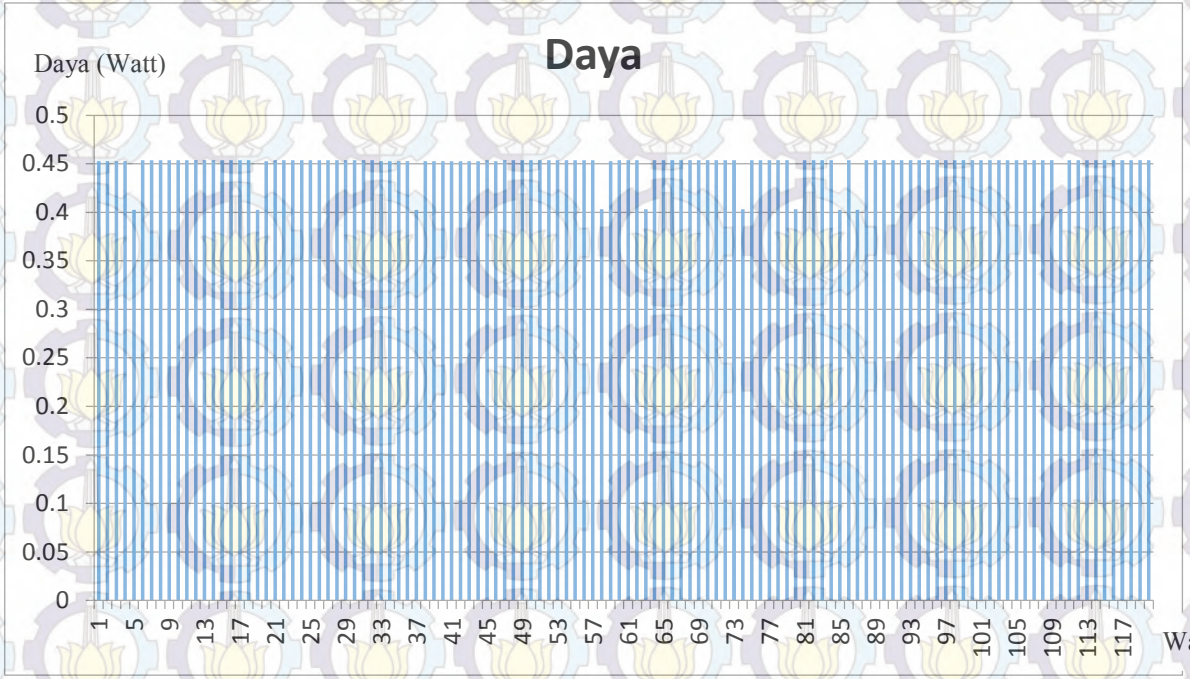
```
  int cahaya = sensorValue*(1000.0/1024.0) ; // sensor cahaya

  data[0] = DHT.humidity ; //input data kelembapan ke array
  data[1] = DHT.temperature ;//input data suhu ke array
  data[2] = cahaya ; // input data cahaya ke array

  long int x;
long int y;
long int z;

  while(Serial1.available() !=0)
  {

    rec=(rec)+char(Serial1.read());
  }
  delay(500);
  long int nrec=rec.toInt();
  x=nrec%100;
  y=((nrec%10000)-x)/100;
  z=(nrec-(nrec%10000))/10000;

  if(x<20){
    x=data[0];
    y=data[1];
    z=data[2];
  }
  if(x<20){
    x=data[0];
    y=data[1];
    z=data[2];
  }
  if(x>60){
    x=data[0];
    y=data[1];
    z=data[2];
  }
  if(y<20){
    x=data[0];
    y=data[1];
```

```
  z=data[2];
 }
 if(y>37){
  x=data[0];
  y=data[1];
  z=data[2];
 }
 if(z<100){
  x=data[0];
  y=data[1];
  z=data[2];
 }
 if(z>800){
  x=data[0];
  y=data[1];
  z=data[2];
 }
  delay(500);
 String bin[3];
 long int hum=(x+data[0])/2;
 long int temp=(y+data[1])/2;
 long int lux=(z+data[2])/2;
 delay(500);
 bin[0]=String(lux);
 bin[1]=String(temp);
 bin[2]=String(hum);
  String buff="";
 buff=String(bin[0]+bin[1]+bin[2]);
delay(500) ;
 Serial1.print(buff);
 Serial.println(buff);
delay(500);
 x=0;
 y=0;
 z=0;
 buff="";
}
```

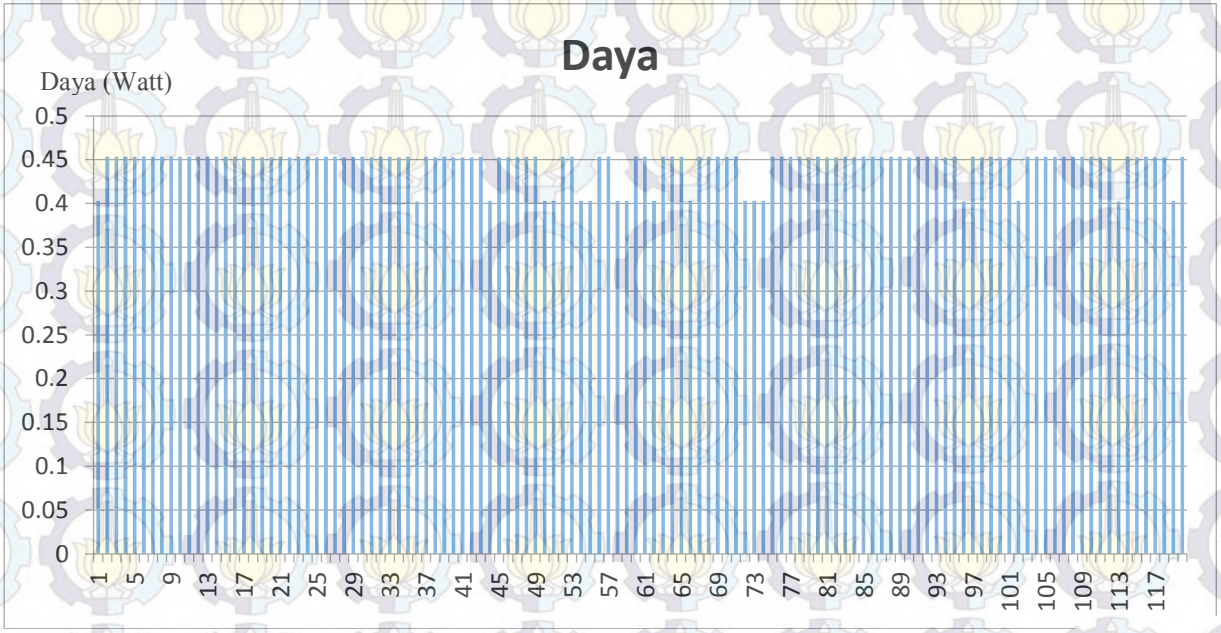
H. HASIL PENGUKURAN DENGAN CATU DAYA POWER BANK



Penggunaan Daya Tanpa Menggunakan Kompresi

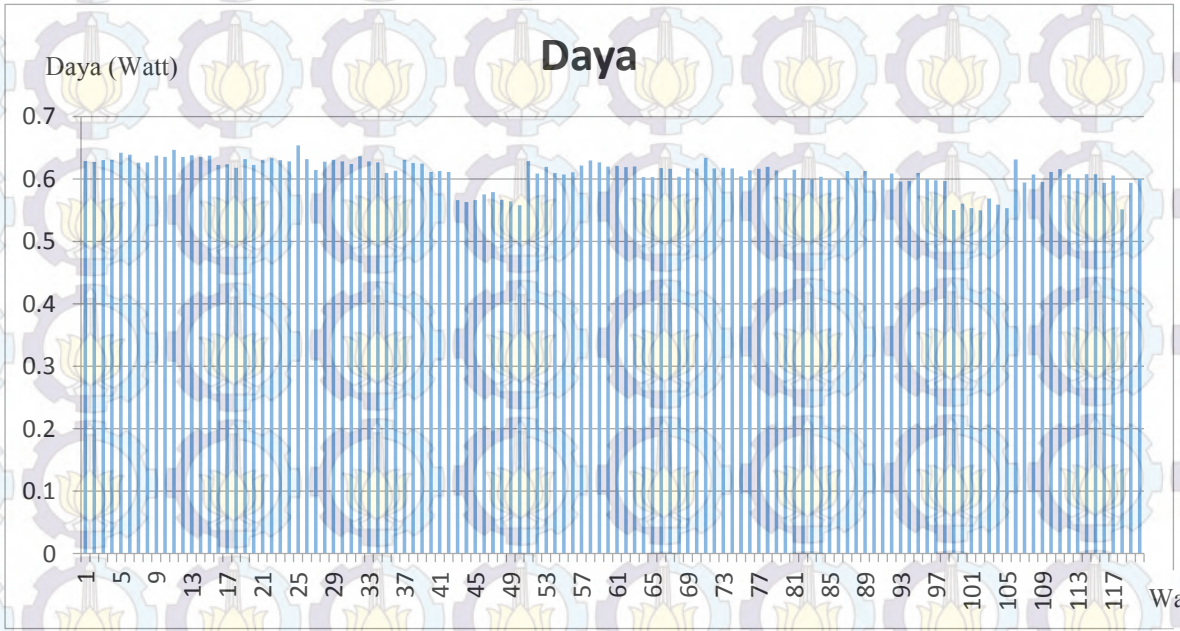


Penggunaan Daya Menggunakan look-up table

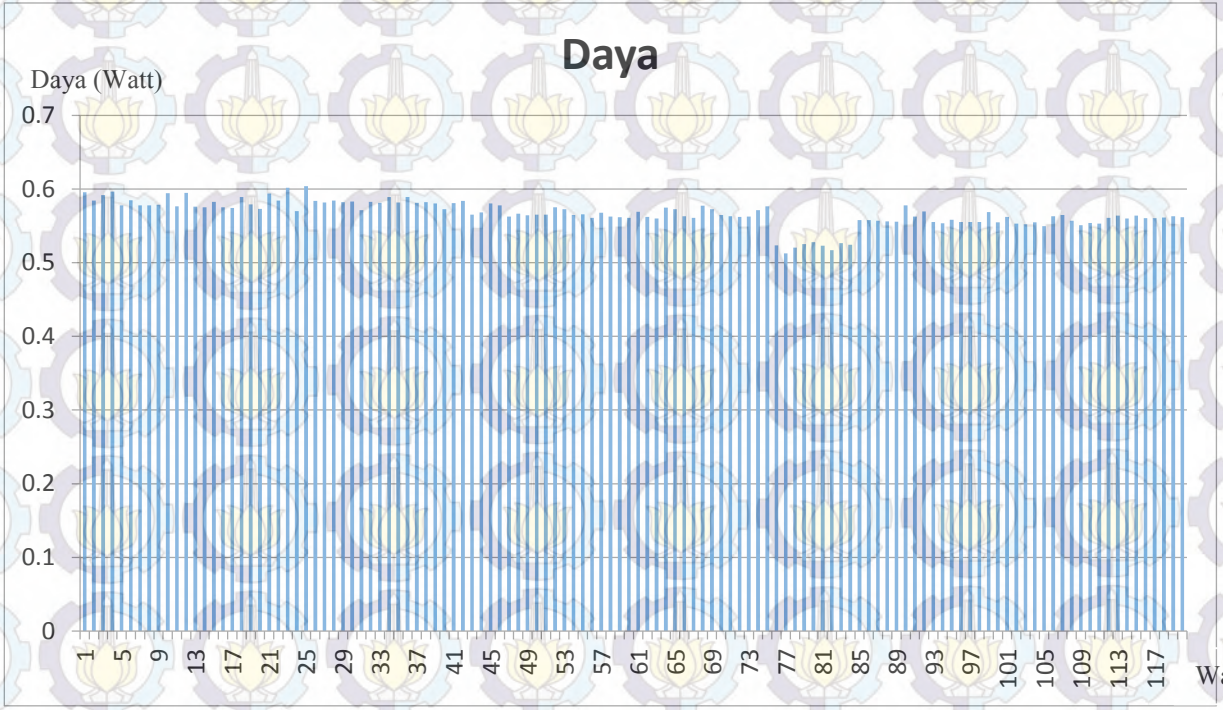


Penggunaan Daya Dengan Metode Huffman

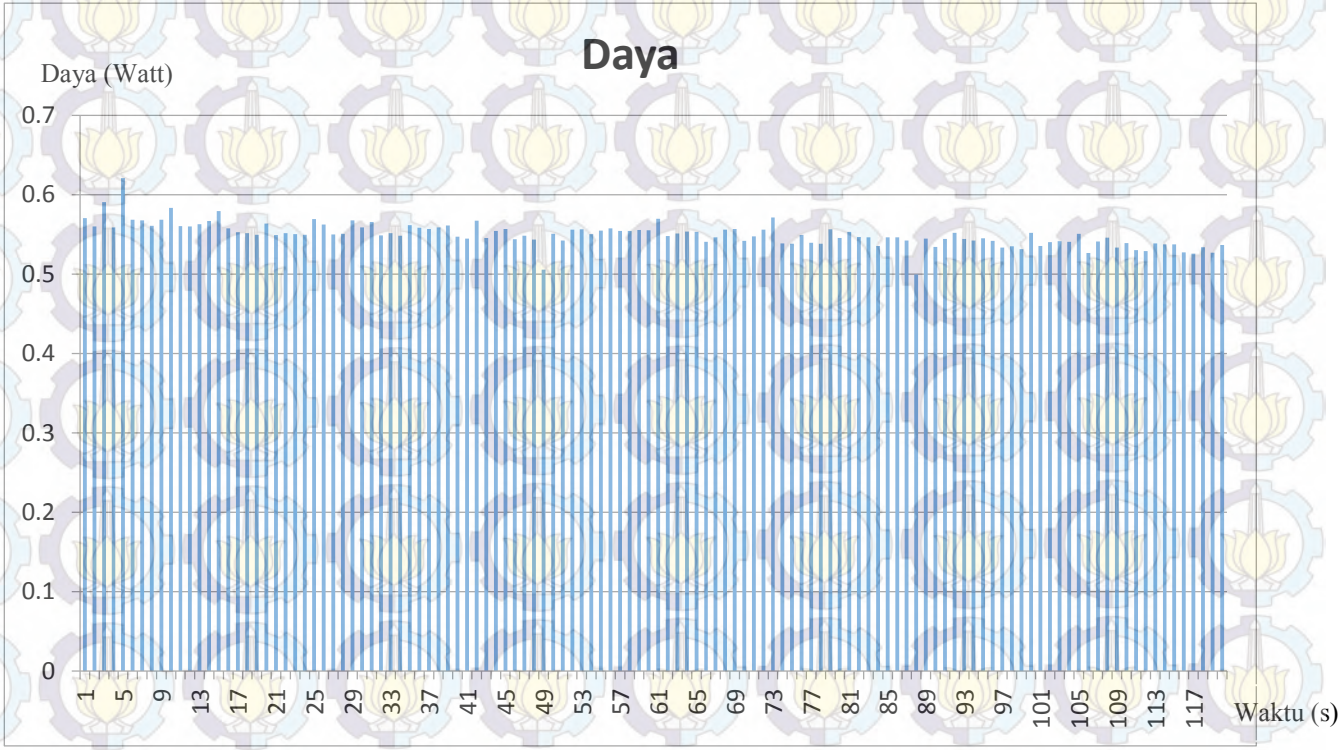
I. HASIL PENGUKURAN DENGAN 40, 25, 15 DATA PER MENIT



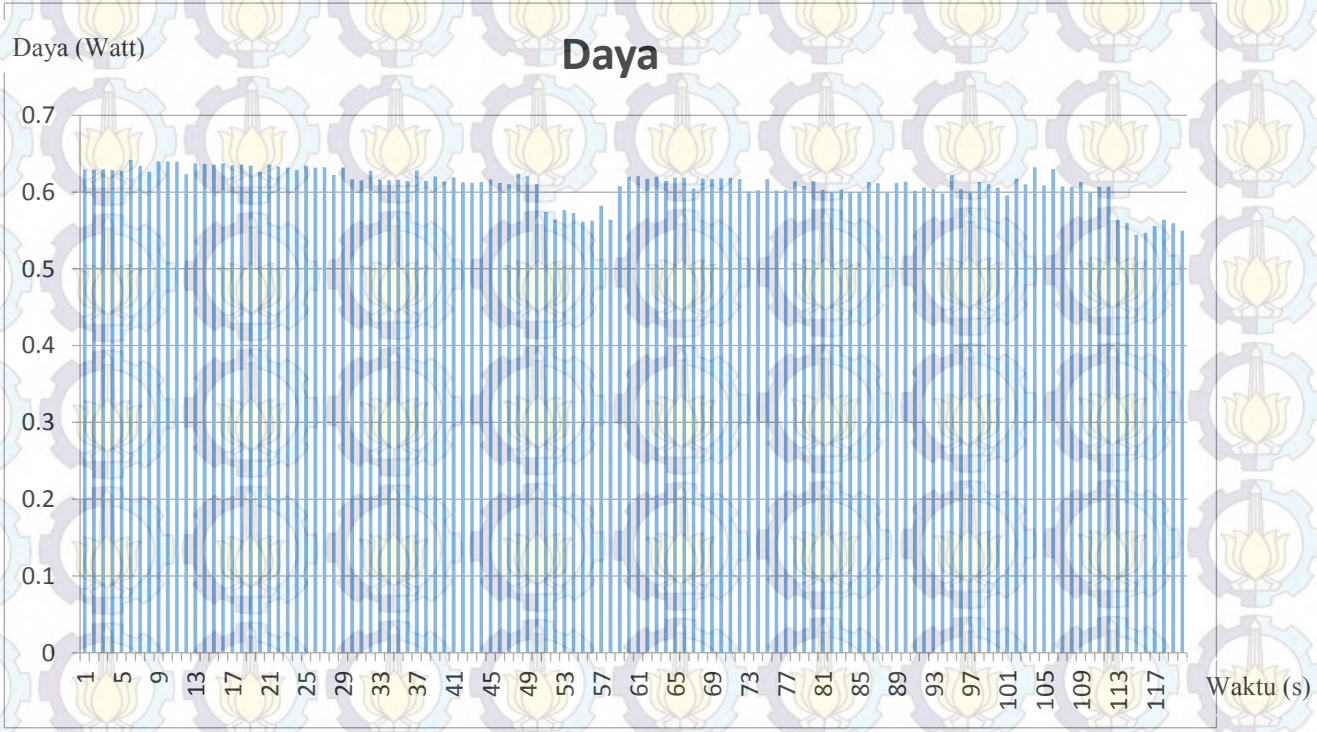
Penggunaan Daya Tanpa Menggunakan Kompresi



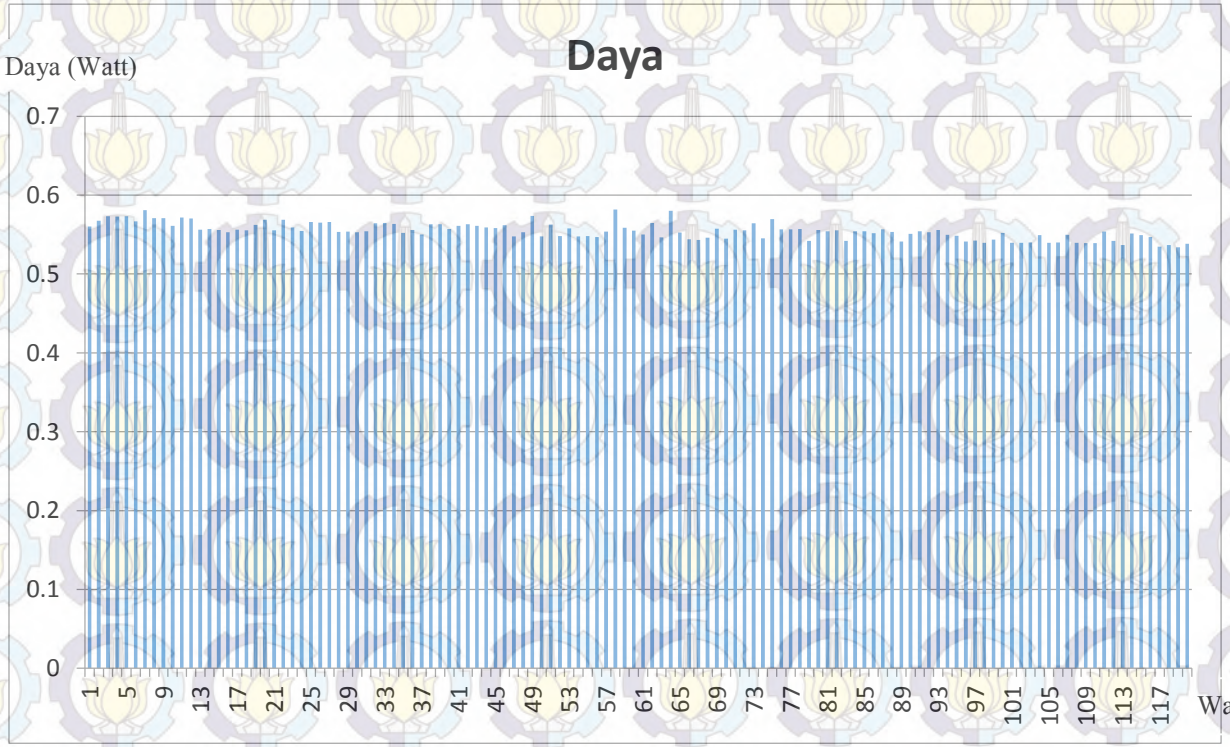
Penggunaan Daya Menggunakan look-up table



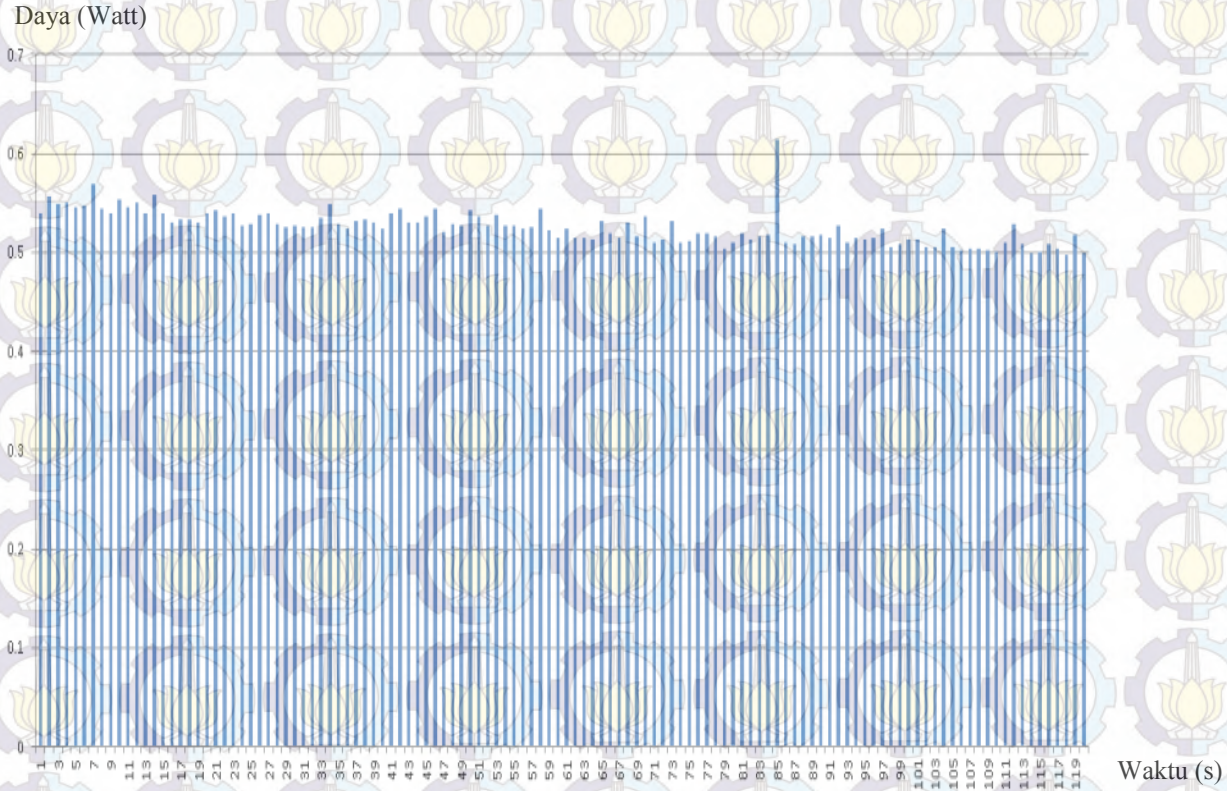
Penggunaan Daya Dengan Metode Huffman

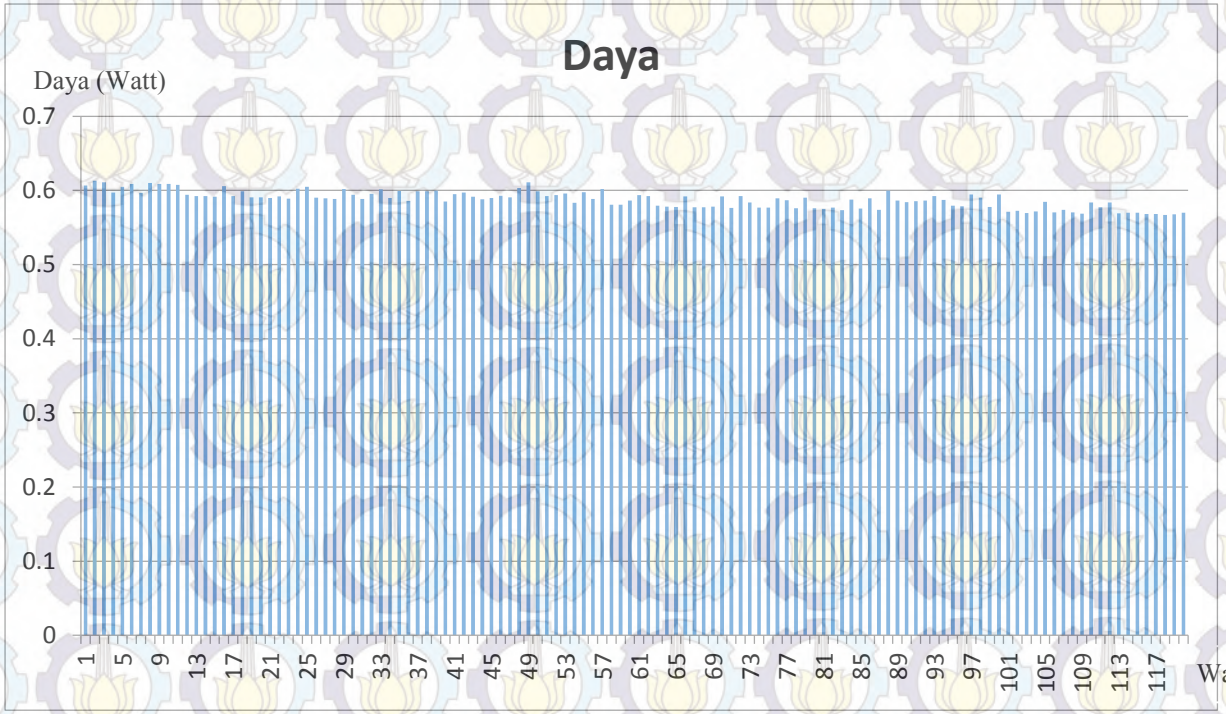


Penggunaan Daya Tanpa Kompresi

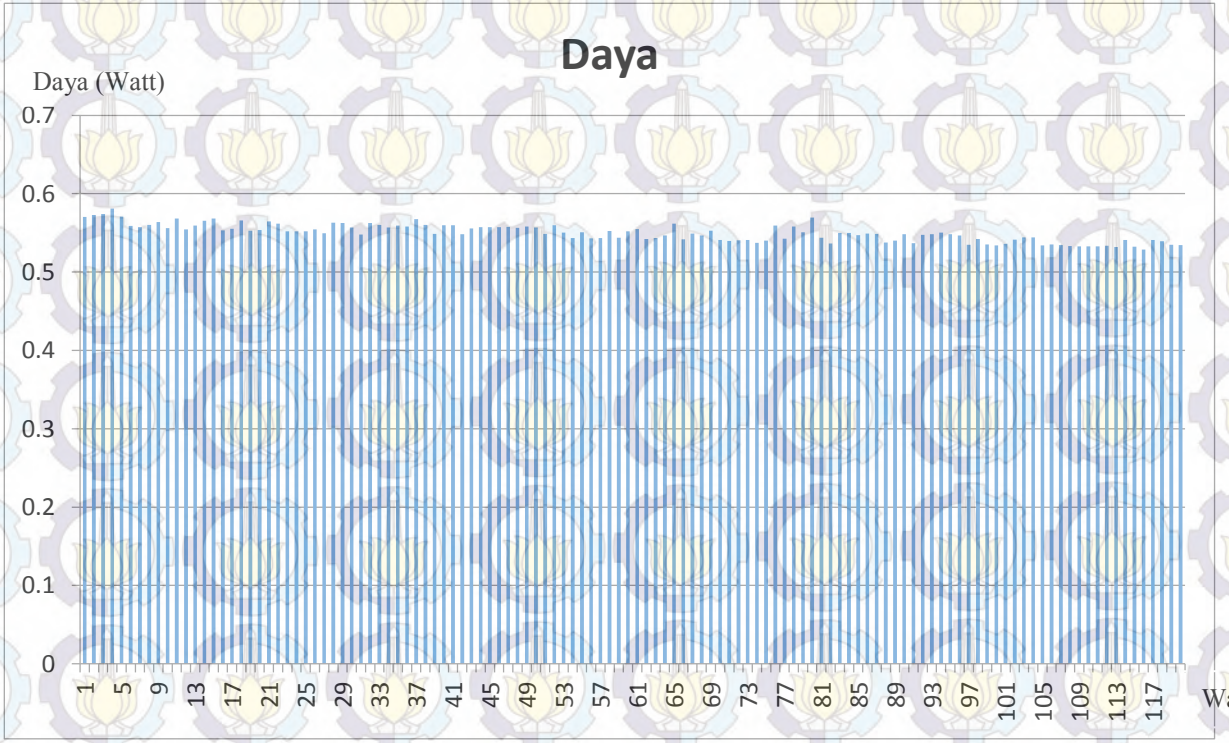


Penggunaan Daya Menggunakan *Look-up Table*

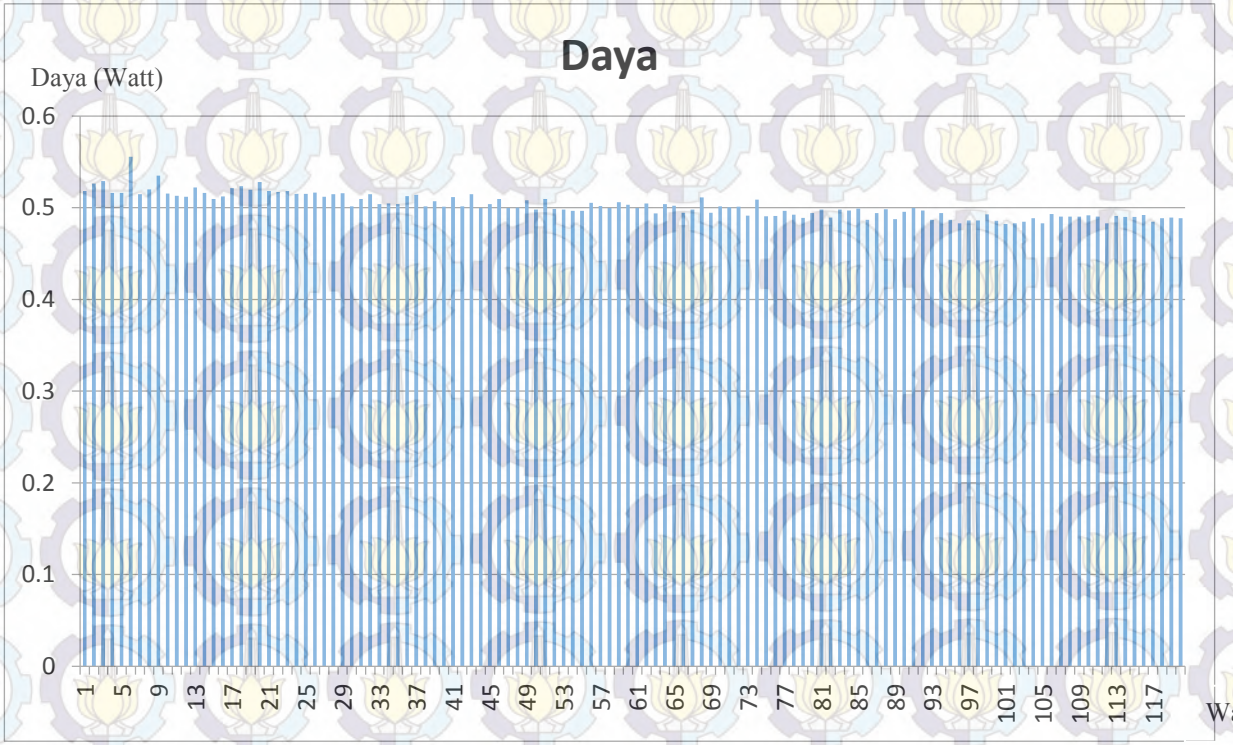

Daya (Watt)
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Waktu (s)



Penggunaan Daya Tanpa Menggunakan Kompresi

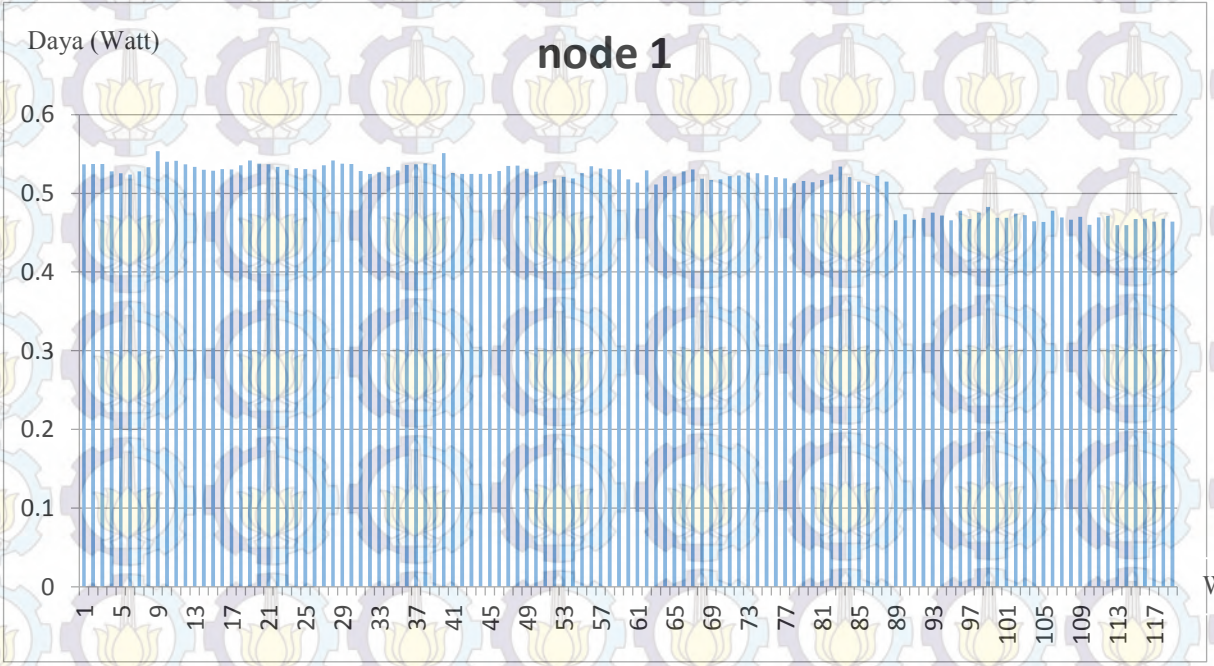


Penggunaan Daya Menggunakan *Look-up Table*

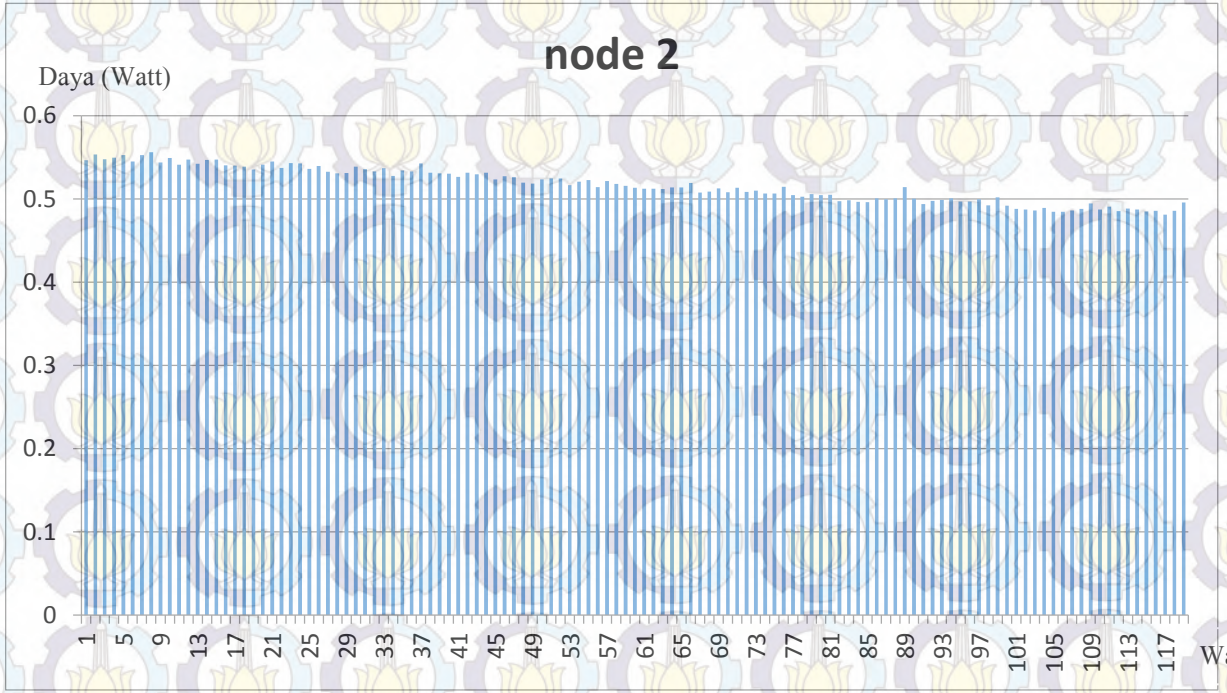


Penggunaan Daya Dengan Metode Huffman

J. HASIL PENGUKURAN DENGAN ALGORITMA DISTRIBUTED CONSENSUS



Penggunaan Daya pada Node 1



Penggunaan Daya pada Node 2

[Halaman ini sengaja dikosongkan]

DAFTAR PUSTAKA

[1] Van der Byl, Andrew, Neilson, Robert, H.W, Richardt.2009. An "Evaluation of Compression Techniques for Wireless Sensor Networks". IEEE Journal AFRICON 2009

[2] Kadir, Abdul. 2012. "Panduan Praktis Mempelajari Aplikasi Mikrokontroler dan Pemrogramannya Menggunakan Arduino".CV. ANDI OFFSET

[3] Held, Gilbert. 1989. "Data Compression Techniques and Aplications, Hardware, and Software Considerations". John wiley & Sons Ltd.

[4] Wang, You-Chiun. "Data Compression Technique in Wireless Sensor Networks".

[5] Bonny, Talal, and Henkel, Joerg. "Efficient Code Density Through Look-up Table Compression".

[6] SĂCĂLEANU*, Ioan, Dragos, etal. "An Adaptive Huffman Algorithm for Data Compression in Wireless Sensor Networks".

[7] Russian, Ishtiaq, Syed, etal. "CoXoH: Low Cost Energy Efficient Data Compression for Wireless Sensor Nodes using Data Encoding".

[8] Jinho, Shancang, Xinheng, 2012. "A General Distributed Consensus Algorithm for Wireless Sensor Networks".

[9] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," Systems and Control Letters, vol. 53, pp. 65-78, 2004.

[Halaman ini sengaja dikosongkan]

BIOGRAFI PENULIS

Muhammad Naufal Farid merupakan salah satu mahasiswa ITS yang mengambil jurusan teknik elektro. Bidang studi yang diambil dalam menjalani kuliah adalah telekomunikasi multimedia. Ia yang berkelahiran di Bandar Lampung, 22 Oktober 1993 merupakan anak ke 2 dari 4 bersaudara dari pasangan Hasnul Abrar dan Herlina Rustam. Semasa perkuliahan, ia tertarik pada ilmu jaringan sensor nirkabel yang dimana menjadi salah satu pembahasan pada tugas akhirnya.