BAB 1

PENDAHULUAN

1.1 Latar Belakang

Wireless Sensor Network (WSN) adalah suatu jaringan nirkabel yang terdiri dari kumpulan node sensor dengan kemampuan deteksi (*sensing*), komputasi, dan komunikasi yang tersebar pada suatu tempat. Setiap sensor akan mengumpulkan data dari area yang dideteksi seperti temperatur, suara, getaran, tekanan, gerakan, kelembaban udara dan deteksi lainnya tergantung kemampuan sensor tersebut. Data yang diterima ini kemudian akan diteruskan ke *base station* untuk diolah sehingga memberikan suatu informasi. *Wireless Sensor Network* dapat diimplementasikan pada berbagai bidang kehidupan manusia diantaranya bidang militer untuk deteksi musuh, bidang pertanian untuk pemantauan pertumbuhan tanaman, bidang kesehatan, deteksi bahaya dan bencana alam, bidang pembangunan dan tata kota, dan bidang pendidikan.

Terdapat dua macam arsitektur *Wireless Sensor Network*, yaitu kluster dan flat. Pada arsitektur kluster, *node-node* akan disusun secara hierarki dan memiliki peran sebagai *cluster head*, *child node*, atau *parent node*. *Cluster head* berfungsi sebagai pengatur beberapa *child node*. Sedangkan pada arsitektur flat hanya terdapat dua macam *node* secara fungsional, yaitu *source node* dan *sink node*. Semua node sensor (*source node*) akan mengirim data ke satu tujuan akhir yaitu *sink node*.

Dalam praktiknya pengiriman data merupakan suatu hal yang penting pada *Wireless Sensor Network*. Data yang didapat dari sensor harus dapat sampai ke *base station* dengan utuh dan akurat (*reliable*). Data yang *reliable* ini sangat penting karena hasil pengukuran dan tindakan selanjutnya yang akan diambil akan bergantung pada data-data tersebut. Terdapat beberapa cara untuk memastikan transfer data *reliable* yaitu dengan *Link-Layer Retransmission*, *End-to-End Retransmission*, dan *Erasure Code*.

Pada skripsi ini dibangun aplikasi untuk transfer data pada *Wireless Sensor Network*. *Wireless Sensor Network* yang dibuat juga dapat melakukan transfer data ke node sensor tetangganya hingga sampai ke node sensor yang berperan sebagai *base station*. Karena node sensor memiliki kapasitas penyimpanan yang kecil dan data yang *reliable* sangat dibutuhkan untuk menentukan tindakan selanjutnya, maka akan dibangun juga *Wireless Sensor Network* yang memiliki sifat *reliable* tersebut.

1.2 Rumusan Masalah

- Bagaimana cara membangun aplikasi transfer data dari setiap node sensor pada *Wireless Sensor Network*?
- Bagaimana cara membangun aplikasi transfer data yang *reliable* pada *Wireless Sensor Network*?

1.3 Tujuan

- Membangun aplikasi transfer data yang *reliable* pada *Wireless Sensor Network*.

1.4 Batasan Masalah

Penelitian ini dibuat berdasarkan batasan-batasan sebagai berikut:

1. Sensor yang digunakan sebagai penelitian hanya sensor untuk mengukur temperatur, kelembapan, getaran, dan tekanan udara.
2. Arsitektur yang digunakan untuk membangun *Wireless Sensor Network* ini adalah flat dan kluster.

1.5 Metodologi

Berikut adalah metode penelitian yang digunakan dalam penelitan ini:

1. Melakukan studi literatur mengenai *Wireless Sensor Network*.
2. Mempelajari protokol transfer data yang biasa pada *Wireless Sensor Network*.
3. Mempelajari prinsip *Reliable Data Transfer* pada *Wireless Sensor Network*.
4. Mempelajari pemrograman pada *Wireless Sensor Network* dengan Bahasa Pemrograman JAVA.
5. Melakukan perancangan perangkat lunak.
6. Mengimplementasi rancangan perangkat lunak pada *Wireless Sensor Network*.

1.6 Sistematika Pembahasan

Setiap bab dalam penelitian ini memiliki sistematika penulisan yang dijelaskan ke dalam poin-poin sebagai berikut:

1. Bab 1: Pendahuluan yaitu membahas mengenai gambaran umum penelitian ini. Berisi tentang latar belakang, rumusan masalah, tujuan, batasan masalah, metode penelitian, dan sistematika penulisan
2. Bab 2: Dasar Teori, yaitu membahas teori-teori yang mendukung berjalannya penelitian ini. Berisi tentang *Wireless Sensor Network*,
3. Bab 3: Analisis,
4. Bab 4: Perancangan,
5. Bab 5: Implementasi dan Pengujian,
6. Bab 6: Kesimpulan,

BAB 2

LANDASAN TEORI

Pada bab ini dijelaskan dasar-dasar teori mengenai *Wireless Sensor Network*, transfer data pada wireless sensor network, prinsip *reliable data transfer* pada Wireless Sensor Network, dan pengembaganan pemrograman pada WSN.

2.1 Wireless Sensor Network

Wireless Sensor Network merupakan jaringan nirkabel yang terdiri dari sekumpulan node sensor yang diletakan pada suatu tempat dan memiliki kemampuan untuk mengukur kondisi lingkungan sekitar(*sensing*), komputasi dan dilengkapi dengan alat komunikasi *wireless* untuk komunikasi antara node sensor. Sensor ini akan mengumpulkan data dari kondisi lingkungannya, seperti: cahaya, suara, kelembapan, getaran, gerakan, temperatur, tekanan udara, kualitas air, komposisi tanah, dan lain-lain. Data ini kemudian dikirimkan ke node sensor tetangganya hingga sampai ke *base station* sebagai pusat untuk dikelola.

2.1.1 Penerapan Wireless Sensor Network

Pada awalnya *sensor network* (jaringan sensor) digunakan dalam teknologi militer untuk mendeteksi musuh di laut dan di darat. Semakin lama node sensor ini banyak dikembangkan untuk membantu berbagai bidang kehidupan manusia. Pemanfaatan Wireless Sensor Network pada kehidupan manusia dapat dilihat pada ilustrasi Gambar 2.1. Berikut adalah beberapa penerapan wireless sensor network pada berbagai bidang kehidupan manusia:

- Bidang militer
  Wireless Sensor Network digunakan sebagai bagian dari komunikasi pada bidang militer dan deteksi target atau musuh.

- Monitoring area
  Pada *monitoring area*, node sensor akan disebar pada suatu tempat yang akan di monitoring. Saat node sensor mendeteksi kejadian(panas, tekanan, dan lain-lain) pada suatu tempat, data akan dikirimkan ke *base station* untuk ditentukan tindakan selanjutnya.

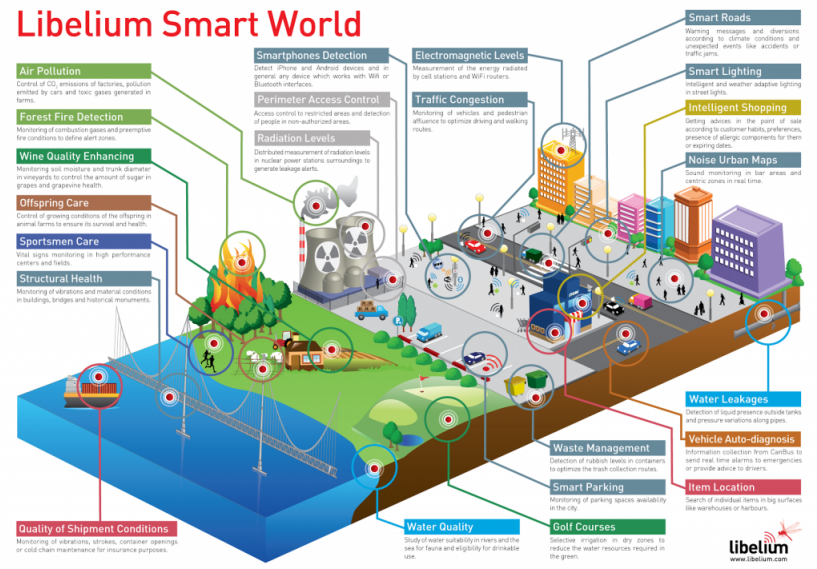
- Transportasi
  Pada bidang transportasi *Wireless Sensor Network* digunakan untuk mendeteksi lalu lintas secara aktual yang nantinya akan disampaikan kepada pengendara tentang kejadian di depan mereka seperti kemacetan lalu lintas.

- Kesehatan
  Beberapa aplikasi kesehatan seperti membantu pada disabilitas, monitoring pasien, diagnosis, pengaturan penggunaan obat, dan pelacakan dokter dan pasien di rumah sakit.

- Deteksi lingkungan
  Deteksi lingkungan yang dapat dilakuan antara lain deteksi gunung berapi, polusi udara, kebakaran hutan, efek rumah kaca, dan deteksi longsor.

- Monitoring struktur
  Wireless Sensor Network dapat melakukan deteksi pergerakan bangunan dan infrastruktur seperti jembatan, *flyover*, terowongan dan fasilitas lain tanpa mengeluarkan biaya untuk melakukan dektesi manual dengan mendatangi tempatnya secara langsung.

- Bidang pertanian
  Pada bidang pertanian dapat membantu pengelola pertanian untuk melihat penggunaan air dalam irigasi dan mengelola buangan pertanian mereka.

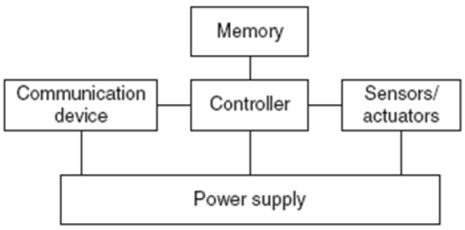


Gambar 2.1: Ilustrasi Pemanfaatan *Wireless Sensor Network*

2.1.2 Node Sensor

Struktur Node Sensor

Setiap node sensor memiliki kemampuan deteksi, komputasi dan komunikasi. Node sensor memiliki lima komponen utama yaitu *controller*, *memory*, *sensor and actuator*, *communication device*, dan *power supply*, (Gambar 2.2). Semua komponen akan bekerja secara seimbang dalam melakukan sensing, komputasi, komunikasi, dan menjaga penggunaan energi seminimal mungkin.



Gambar 2.2: Struktur Node Sensor

Controller

Controller adalah inti utama pada node sensor. Controller mengumpulkan data dari sensor dan memproses data tersebut hingga menentukan kapan dan kemana data tersebut dikirim. Controller menerima data dari node sensor lain. Pada controller biasanya terdapat microcontroller atau microprocessor yang mengatur dan melakukan komputasi data dari node sensor. Microcontroller ini

juga dapat mengurangi penggunaan energi dengan masuk ke sleep states yang berarti hanya bagian dari controller saja yang aktif.

Beberapa microcontroller yang digunakan dalam Wireless Sensor Node:

- Intel StrongARM (32-bit RISC, up to 206 MHz)
- Texas Instrument MSP 430 (16-bit RISC, up to 4 MHz,RAM 2-10 kB)
- Atmel Atmega 128L (8-bit)

Memory

Random Access Memory (RAM) digunakan untuk menyimpan sementara hasil yang didapat dari sensor. RAM juga menyimpan sementara paket dari node sensor lain. Jika power supply terjadi masalah maka data pada RAM ini akan hilang. Ini merupakan salah satu kekurangan dari penggunaan RAM. Maka untuk menyimpan kode program digunakanlah Read Only Memory (ROM). ROM ini biasa disebut Electrically Erasable Programmable Read-Only Memory (EEPROM) atau Flash Memory. Flash memory dapat menyimpan data jika suatu saat data pada RAM hilang atau energi sudah habis (intermediate storage).

Communication Device

Communication Device digunakan untuk bertukar data antar node sensor. Pada aplikasi Wireless Sensor Network, Radio Frequency(RF)-based adalah media komunikasi yang paling relevan saat ini. RF-based mendukung jangkauan yang jauh, data rate yang tinggi, dapat menerima error rate pada penggunaan energi, dan tidak perlu saling mengetahui posisi antara penerima dan pengirim.

Pada node sensor dibutuhkan transmitter dan receiver untuk menerima dan mengirim data. Kedua hal ini dapat digabung dan disebut dengan transceiver. Tugas transceiver adalah mengubah stream bit dari microcontroller dan mengubahnya menjadi gelombang radio. Selain itu transceiver juga dapat mengubah gelombang radio menjadi stream bit.

Sensor dan Actuator

Sensor dan Actuator adalah hal yang penting pada Wireless Sensor Network. Tanpa sensor dan actuator maka node sensor tidak berguna dan tidak dapat digunakan. Tabel 2.1 adalah jenis - jenis sensor yang dapat dimiliki node sensor. Sensor dikategorikan menjadi tiga:

1. **Passive, omnidirectional sensors** Sensor ini dapat mengukur kualitas dari lingkungan fisik tempat node sensor tersebut tanpa mengubah lingkungannya. Beberapa sensor dikategori ini *self-powered*, sensor mendapatkan energi yang mereka butuhkan dari lingkungannya. Energi ini digunakan untuk memperkuat sinyal analog. *Omnidirectional* berarti tidak ada arah pada sensor ini. Sensor akan memancarkan sinyalnya ke segala arah. Contoh sensor ini adalah termometer, sensor cahaya, sensor getaran, mikrofon, sensor kelembapan, sensor tekanan udara, sensor deteksi asap, dan lain-lain.

2. **Passive, narrow-beam sensors** Sensor ini memiliki sifat yang sama yaitu pasif, tidak mengubah lingkungannya. Sensor ini dapat melakukan gerakan dan memiliki arah atau daerah pengukuran. Contoh dari sensor ini adalah kamera yang bisa mengukur sesuai dengan arah yang dituju.

3. **Active Sensor** Sensor ini aktif dalam memeriksa lingkungannya. Contoh dari sensor ini adalah sonar, radar atau sensor seismik. Sensor ini menghasilkan gelombang dengan ledakan kecil untuk melakukan deteksi.

Actuator jumlahnya beragam seperti sensor. Actuator adalah penerima sinyal dan yang mengubahnya menjadi aksi fisik. Contoh aktuator adalah LED, yang mengubah listrik menjadi cahaya dan motor (electrical motor) juga mengubah listrik menjadi gerakan.

Tabel 2.1: Jenis - jenis sensor yang dapat dimiliki node sensor

Sensor	Sense Event	Remark
Accelerometer		
Acoustic emission sensor		
Acoustic sensor		
Capacitance sensor		
ECG		
EEG		
EMG		
Electrical/electromagnetic sensor		
Gyroscope		
Humidity Sensor		
Infrasonic sensor		
Magnetic sensor		
Oximeter		
pH sensor		
Photo acoustic spectroscopy		
Piezoelectric cylinder		
Soil moisture sensor		
Temperature sensor		
Barometer sensor		
Passive infrared sensor		
Seismic sensor		
Oxygen sensor		
Blood flow sensor		

Power Supply

Power supply atau sumber energi pada Wireless Sensor Network bisa berasal dari dua cara yaitu ***storing energy*** dan ***energy scavenging***. *Storing energy* adalah dengan menggunakan baterai sebagai sumber energinya. Baterai yang digunakan dapat diisi ulang maupun yang tidak dapat diisi ulang. *Energy scavenging* digunakan saat membuat Wireless Sensor Network yang akan digunakan dalam waktu yang lama. Dibutuhkan energi yang bisa dikatakan tidak terbatas. Salah satu cara *energy scavenging* adalah *Photovoltaics*. *photovoltaics* dapat disebut juga *solar cell* yang memanfaatkan cahaya matahari dan mengubahnya menjadi energi sebagai pembangkit daya. Cara lain yang bisa digunakan adalah pemanfaatan angin dan air untuk mengerakan kincir atau turbin yang akan menghasilkan listrik dan digunakan sebagai sumber energi pada node sensor.

2.1.3 Arsitektur dan Topologi Wireless Sensor Network

Pada *Wireless Sensor Network* biasanya akan terdapat banyak node sensor yang disebar pada suatu tempat. Terdapat satu atau lebih *sink node* atau *base station* dalam area sensing tersebut (Gambar 2.10). *sink node* atau *base station* adalah node sensor yang bertugas untuk mendapatkan data dari node sensor lain. Dalam membuat *Wireless Sensor Network* perlu diperhatikan arsitektur dan topologi yang akan digunakan. Tidak semua topologi jaringan komputer dapat digunakan untuk *Wireless Sensor Network*.

Ada banyak topologi pada jaringan sensor (*sensor network*). Pada jaringan sensor dengan menggunakan kabel topologi yang sering digunakan adalah topologi *star*, *line*, atau *bus*. Sedangkan pada jaringan sensor tanpa kabel (WSN) topologi yang biasa digunakan adalah *star*, *tree*, atau *mesh*.

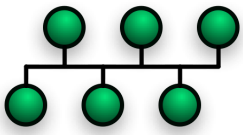
Topologi Point-to-Point

Topologi point-to-point adalah topologi yang menghubungkan dua titik (Gambar 2.3). Topologi point-to-point dibagi menjadi dua yaitu permanent point-to-point dan switched point-to-point. Permanent point-to-point adalah koneksi perangkat keras antara dua titik dan tidak dapat diubah. Switched point-to-point adalah koneksi point-to-point yang dapat berpindah antara node yang berbeda.

Gambar 2.3: Topologi Point-to-Point

Topologi Bus

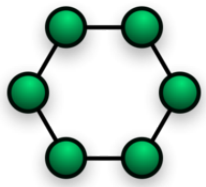
Topologi bus seperti pada Gambar 2.4 akan terdiri dari node-node dan sebuah jalur. Setiap node akan terhubung dengan jalur yang sama. Untuk mengirim data atau komunikasi akan dilakukan bergantian antar node. Kekurangan dari topologi bus ini adalah jika suatu saat jalur / bus ini mengalami kerusakan maka setiap node tidak dapat berkomunikasi lagi.



Gambar 2.4: Topologi Bus

Topologi Ring

Pada topologi ring node akan disusun dengan bentuk melingkar (Gambar 2.5). Setiap node akan terkoneksi dengan dua node lain. Transfer data terjadi dengan cara data akan berjalan dari satu node ke node lain mengikuti jalur melingkar tersebut hingga menemukan node tujuan yang tepat. Topologi ini mudah untuk diimplementasikan tapi kekurangan dari topologi ring adalah saat ada node yang rusak maka perlu biaya lebih untuk memperbaikinya. Biasanya node untuk menangani kegagalan komunikasi akibat node yang rusak, akan di atur komunikasi node tidak hanya satu arah tetapi dapat ke arah sebaliknya.



Gambar 2.5: Topologi Ring

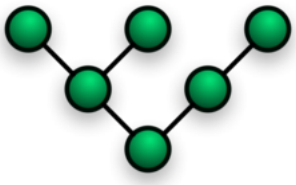
Topologi Star

Topologi star terdiri dari satu node yang berada di tengah biasanya berupa hub atau switch seperti pada Gambar 2.6. Setiap node akan terkoneksi dengan node yang berada di tengah ini. Saat node akan berkomunikasi dengan node lain, node tersebut harus mengirimkan data tersebut ke node yang ada ditengah dahulu dan node yang ditengah ini akan meneruskan data tersebut ke node tujuan. Yang paling penting pada topologi ini adalah node yang berada di tengah, karena semua komunikasi harus melalui node tersebut. Jika node tengah mengalami kerusakan maka tidak akan terjadi komunikasi antar node pada jaringan tersebut.

Gambar 2.6: Topologi Star

Topologi Tree

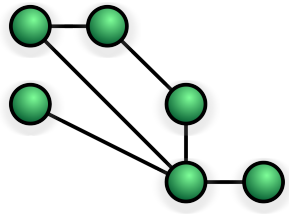
Pada topologi tree node-node akan disusun secara hierarki dengan satu node yang berada pada level paling atas sebagai *root node* (Gambar 2.7). *Root node* akan terhubung dengan satu atau lebih node level dibawahnya. Dengan topologi tree lebih mudah untuk melakukan identifikasi dan meminimalisir kesalahan, namun jika tree sudah sangat besar / level tree sudah sangat banyak maka akan sulit untuk mengaturnya.



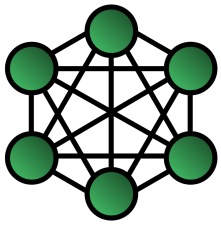
Gambar 2.7: Topologi Tree

Topologi Mesh

Topologi mesh dibagi menjadi dua yaitu partially connected mesh dan fully connected mesh. Pada partially connected mesh (Gambar 2.8), node akan terhubung dengan lebih dari satu node. Pada fully connected mesh (Gambar 2.9), setiap node akan terhubung dengan semua node lain pada jaringan tersebut

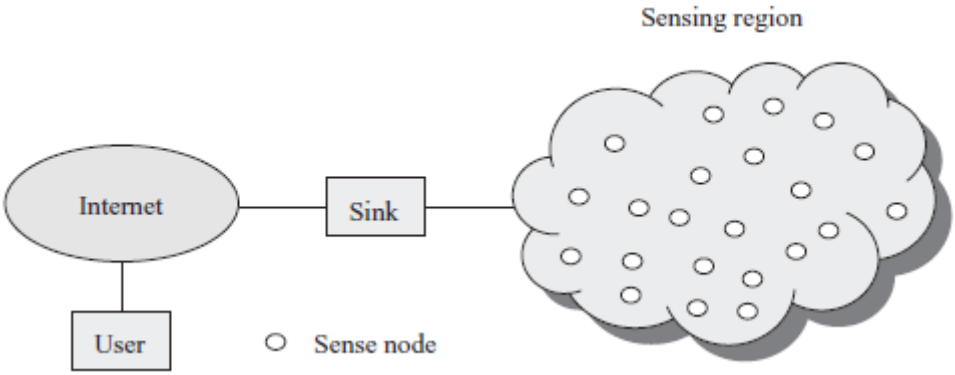


Gambar 2.8: Topologi Partially Connected Mesh



Gambar 2.9: Topologi Fully Connected Mesh

Arsitektur yang biasanya dipakai pada *Wireless Sensor Network* adalah **arsitektur flat / peer-to-peer** dan **arsitektur hierarki**. Selain itu dalam membangun *Wireless Sensor Network* perlu juga diperhatikan jalur komunikasi yang digunakan untuk menghubungkan antar node sensor saat transfer data. Untuk area *sensing* yang tidak terlalu luas dan hanya menggunakan sedikit node sensor dapat menggunakan cara komunikasi ***single hop***. Sedangkan untuk daerah yang luas dan memerlukan banyak node sensor dapat menggunakan cara komunikasi ***multi hop***.



Gambar 2.10: Arsitektur Wireless Sensor Network

Single-Hop dan Multi-Hop

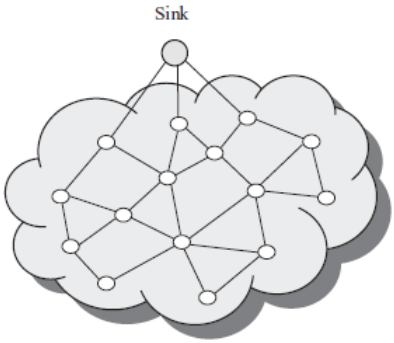
Untuk mengirim data ke sink node setiap node sensor dapat menggunakan single-hop long-distance transmission. Single-hop long-distance ini berarti setiap node sensor akan mengirimkan data ke sink node hanya satu kali lompatan walaupun jarak antara sink node dengan node sensor itu sangat jauh. Dalam jaringan sensor, penggunaan energi paling besar adalah saat melakukan komunikasi dibandingan saat sensing. Penggunaan energi akan semakin bertambah jika jarak sink dan node sensor semakin jauh. Untuk menangani masalah tersebut muncul protokol multi-hop.

Pada protokol multi-hop node sensor akan disusun saling berdekatan dan terhubung dengan yang lain. Jadi saat akan berkomunikasi dengan sink node, node sensor harus mengirimkan data

tersebut ke node sensor tetangganya dan diteruskan hingga sampai ke sink node. Karena jarak yang saling berdekatan maka penggunaan energi dapat efektif. Single-hop dan multi-hop ini dapat digunakan dengan topologi flat maupun hierarki sesuai dengan kebutuhan sistem.

Arsitektur Flat / Peer-to-Peer

Pada arsitektur flat, setiap node sensor memiliki peran atau *role* yang sama dalam melakukan *sensing*. Secara fungsional hanya terdapat dua macam node sensor pada arsitektur flat, yaitu *source node* dan *sink node*. Karena jumlah node sensor yang banyak maka tidak mungkin menentukan *global identified* untuk setiap node sensor pada jaringan ini. Untuk mendapatkan data dilakukan dengan cara *sink node* melakukan pengiriman data ke semua node sensor pada area *sensing* dengan cara *flooding* dan hanya node sensor yang sesuai yang akan merespon *sink node*. Setiap node sensor mengirimkan data ke *sink node* dengan *multi hop* dan melalui node tetangganya yang terhubung dengannya untuk meneruskan data (Gambar 2.11).

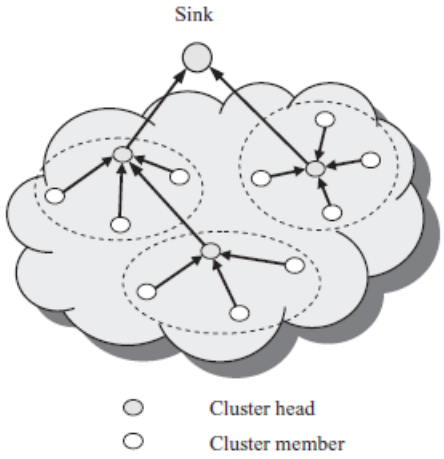


Gambar 2.11: Arsitektur flat pada *Wireless Sensor Network*

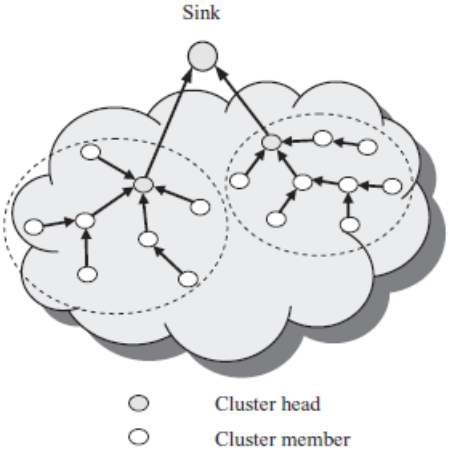
Arsitektur Hierarki

Pada arsitektur hierarki, semua node sensor dikelompokan ke dalam cluster-cluster. Terdapat cluster head pada setiap cluster. Cluster head ini yang mengumpulkan data dari setiap node sensor di bawahnya dan meneruskan data yang telah diterima ke base station atau sink. Hal yang perlu diperhatikan pada arsitektur hierarki adalah pemilihan node sensor sebagai cluster head dan node sensor yang melakukan sensing. Penggunaan energi yang paling besar dalam Wireless Sensor Network ini adalah saat melakukan komunikasi yaitu saat mengirimkan data ke node sensor lain. Maka untuk node sensor yang memiliki energi kecil dapat digunakan untuk sensing, karena node sensor sensing ini hanya melakukan komunikasi ke cluster head. Cluster head harus memiliki energi atau daya yang lebih banyak, karena cluster head akan bertugas menerima hasil sensing node sensor di bawahnya dan meneruskan data ke sink node.

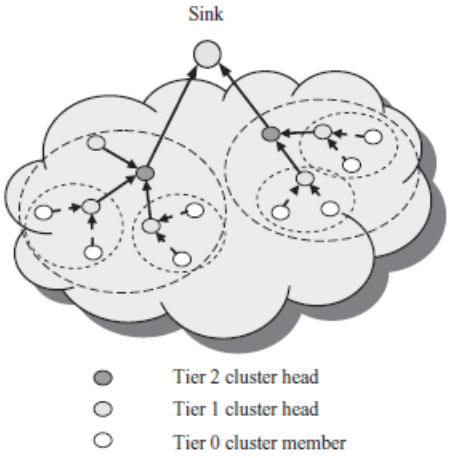
Masalah yang utama pada clustering ini adalah pemilihan cluster head dan bagaimana cara mengatur setiap cluster. Terdapat beberapa cara untuk membuat clustering ini. Bedasarkan jarak antara cluster head dengan cluster member, dapat dibuat clustering dengan single hop atau multi hop seperti pada Gambar 2.12 dan Gambar 2.13. Sedangkan jika berdasarkan jumlah tier dapat dibangun clustering single tier atau multi tier (Gambar 2.14).



Gambar 2.12: Arsitektur hierarki pada *Wireless Sensor Network* dengan *single hop* terhadap *Cluster Head*



Gambar 2.13: Arsitektur hierarki pada *Wireless Sensor Network* dengan *multi hop*



Gambar 2.14: Clsutering dengan multi tier

2.1.4 Sistem Operasi

Setiap node sensor memerlukan sistem operasi (OS) untuk mengontrol perangkat keras dan perangkat lunak. Sistem operasi tradisional tidak dapat digunakan pada WSN. Pada sistem operasi tradisional digunakan untuk mengatur proses, memory, CPU time, dan file system. Terdapat beberapa hal yang harus ditangani oleh sistem operasi dalam WSN yaitu:

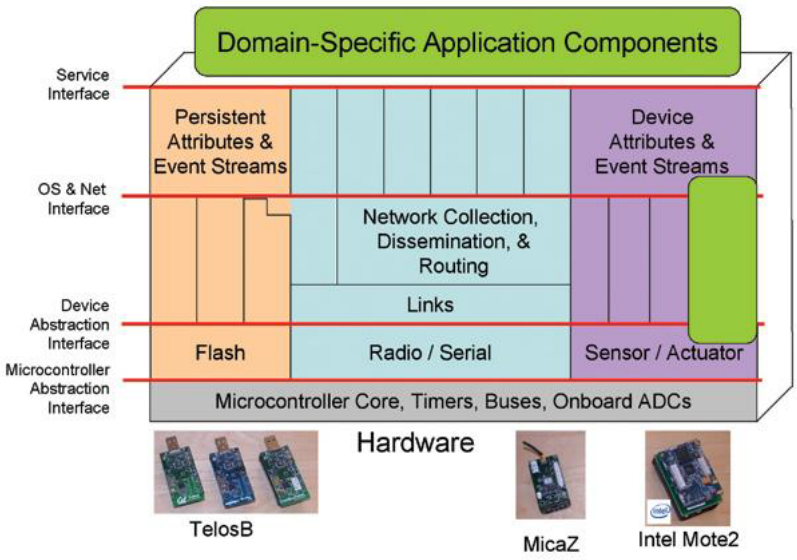
1. WSN memerlukan real-time scheduler. Data yang didapat harus segera dikirim atau diproses.
2. Pengaturan memori karena memory pada WSN sangat kecil.
3. Pengaturan data yang efisien terkait dengan microprocessor dan memori yang terbatas
4. Mendukung kode pemrograman yang efisien dan reliable karena dapat terjadi perubahan kode saat implementasi.
5. Mendukung pengaturan sumber daya untuk menambah waktu hidup dari node sensor dan meningkatkan performa dengan sleep time atau wake up time saat terdapat interupsi dari lingkungan.
6. Mendukung antarmuka untuk pemrograman dan antarmuka perangkat lunak.

Beberapa sistem operasi yang umum digunakan pada WSN antara lain :

1. TinyOS
2. Contiki
3. LiteOS

TinyOS

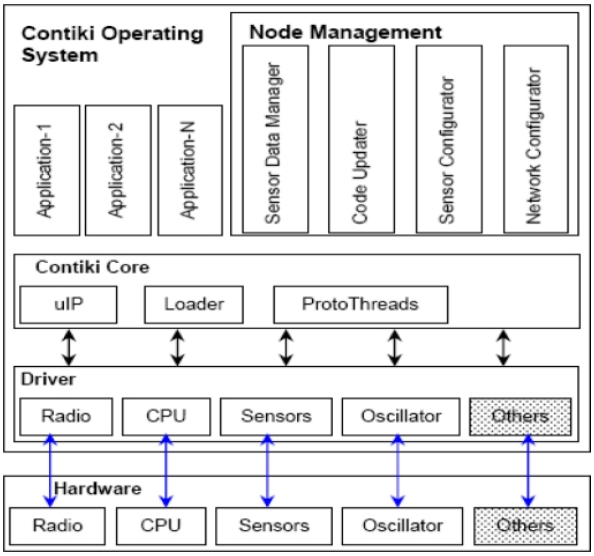
TinyOS adalah sistem operasi open-source yang digunakan pada WSN. TinyOS dapat menjalankan program dengan memori yang sangat kecil. Ukurannya hanya 400 Byte. Komponen librari TinyOS terdiri dari protokol jaringan, layanan distribusi sensor, driver sensor, dan software pengamatan data sensor yang dapat digunakan untuk melakukan monitoring jaringan sensor. Gambar 2.15 adalah arsitektur pada TinyOS.



Gambar 2.15: Arsitektur TinyOS

Contiki

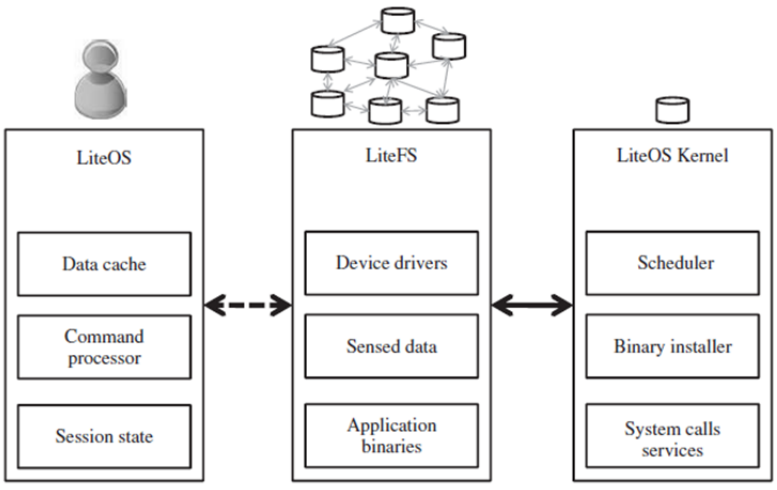
Contiki adalah sistem operasi open-source dengan Bahasa Pemrograman C yang digunakan pada WSN. Contiki sudah mendukung multitasking pada suatu proses. Pengaturan Contiki hanya memerlukan 2KB dari RAM dan 40KB dari ROM. Fitur yang dimiliki oleh Contiki antara lain: multitasking, multithreading, jaringan TCP/IP, IPv6, GUI, Web Browser, Web Server, telnet, dan komputasi jaringan virtual. Gambar 2.16 adalah arsitektur pada Contiki



Gambar 2.16: Arsitektur Contiki

LiteOS

LiteOS adalah sistem operasi mirip UNIX yang didisain untuk WSN. Tujuan dibuat LiteOS adalah membuat sistem operasi yang mirip dengan UNIX agar lebih lebih familiar dengan paradigma pemrograman UNIX. Pada LiteOS terdapat sistem berkas yang hiearki, dan mendukung Bahasa Pemrograman LiteC++ dan UNIX Shell. LiteOS dapat digunakan untuk MicaZ yang memiliki 8 Mhz CPU, 128 byte flash, dan 4KB RAM. LiteOS memiliki tiga komponen utama yaitu: LiteShell, LiteFS, dan Kernel. Gambar 2.16 adalah arsitektur pada LiteOS



Gambar 2.17: Arsitektur LiteOS

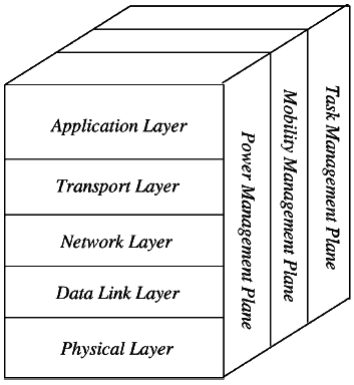
Perbandingan dari TinyOS, Contiki, dan LiteOS dapat dilihat pada Tabel 2.2 berikut ini:

Tabel 2.2: Tabel Perbandingan Sistem Operasi

Operating System	Programming Paradigm	Scheduling
TinyOS	Event-based	FIFO
Contiki	Predominant event-based	FIFO
LiteOS	Thread-based	Priority-based scheduling with optional round-robin support

2.1.5 Protokol Stack pada Wireless Sensor Network

Wireless Sensor Network memiliki lima layer protokol: physical layer, data link layer, network layer, transport layer, dan application layer, seperti pada Gambar 2.18.



Gambar 2.18: Layer pada *Wireless Sensor Network*

Physical Layer

Physical layer bertanggung jawab untuk mengubah bit stream dari data link layer menjadi sinyal agar bisa dilakukan transmisi melalui media komunikasi yang terdapat (transceiver). Pemilihan media dan frekuensi adalah hal yang penting dalam komunikasi antar node sensor. Salah satu cara yang bisa digunakan adalah dengan menggunakan radio frekuensi (RF). Jaringan sensor memerlukan biaya yang kecil, ukuran yang kecil, dan penggunaan daya yang kecil untuk tranceivernya. Karena itu banyak yang menggunakan Radio Frequency (RF) untuk desain perangkat keras node sensornya.

Data Link Layer

Data link layer bertanggung jawab untuk aliran data multiplexing, membentuk data frame, mendeteksi data frame, medium access, dan mengatur kesalahan saat transmisi data. Fungsi paling penting data link layer adalah Medium Access Control (MAC). Protokol MAC menentukan kapan node sensor mengakses media untuk mengirim data, melakukan kontrol dan mengatur paket ke node sensor lain. Hal itu dilakukan agar tidak terjadi paket yang bertabrakan.

Network Layer

Network layer bertanggung jawab untuk routing dari node sensor ke sink node. Pada WSN node sensor tersebar pada suatu tempat untuk melakukan sensing. Data sensing tersebut harus dikirimkan ke sink node untuk diolah. Dalam mengirimkan data tersebut dapat menggunakan single hop atau multi hop. Dalam mengirim data diperlukan protokol routing yang hemat energi.

Transport Layer

Secara umum transport layer bertanggung jawab untuk pengiriman data yang reliable antara node sensor dan sink node. Protokol transport yang biasa tidak bisa diterapkan pada WSN tanpa modifikasi. Setiap jaringan sensor memiliki fungsi khusus. Untuk aplikasi yang berbeda memerlukan kebutuhkan reliabilitas yang berbeda. Pengiriman data pada WSN dibagi menjadi dua yaitu: downstream dan upstream. Upstream berarti node sensor mengirimkan hasil sensing ke sink node. Downstream berarti data berasal dari sink node contohnya, queries, dan perintah-perintah yang dikirimkan ke setiap node sensor. Aliran data yang reliable untuk kedua jenis pengiriman ini mungkin berbeda. Pada upstream reliable data dapat ditoleransi karena sensor akan melakukan sensing terus menerus dan dapat terjadi pengulangan data sehingga data yang hilang tadi dapat dikoreksi. Sedangkan pada downstream memerlukan 100% pengiriman data yang reliable.

Application Layer

Application layer meliputi berbagai macam protokol yang ada pada layer ini untuk menjalankan berbagai macam aplikasi seperti query dissemination, node localization, time synchronization, dan network security. Protokol yang ada pada layer ini antara lain:

1. Sensor management protocol (SMP) adalah protokol untuk melakukan pertukaran lokasi data, sinkornisasi node sensor, memindahkan node sensor, mengatur ulang node sensor, dan menyimpan status dari node sensor.

2. Sensor query and data dissemination protocol (SQDDP) adalah protokol yang mendukung antarmuka aplikasi untuk memasukan query, merespon query, dan mengumpulkan respon.

3. Sensor query and tasking language (SQTL) adalah protokol yang mendukung bahasa pemrograman pada WSN.

2.2 Protokol Transport

WSN terdiri dari sink node dan banyak node sensor. Setiap node sensor akan mengirimkan data hasil sensing ke sink node. Fungsi protokol transport antara lain:

1. Mengatasi kemacetan jaringan (Congestion Control)

2. Mengurangi paket yang hilang

3. Memastikan alokasi bandwidth yang adil

4. Menjamin reliabilitas

Pada protokol transport tradisional dikenal istilah TCP dan UDP, namun keduanya tidak bisa diimplementasikan pada WSN. UDP sendiri tidak mendukung pengiriman data yang reliable. Dalam mengirimkan data TCP menggunakan proses three-handshake. Proses ini memerlukan penggunaan data yang besar sedangkan pada WSN sangat terbatas untuk memorinya.

Protokol transport pada WSN harus mendukung pengiriman data yang reliable. Performa dari sebuah protokol transfer dapat dinilai dari empat ukuran yaitu: energi yang efisien, keandalan (reliability), Quality of Service, dan keadilan (fairness).

Energi Efisien setiap node sensor memiliki keterbatasan energi atau daya yang dimiliki. Membuat protokol untuk menghemat penggunaan energi adalah yang dibutuhkan dalam WSN.

Keandalan (Reliability) pada WSN reliability dibutuhkan untuk mendapatkan data yang lengkap dan sehingga dapat dilakukan tindakan yang tepat terhadap suatu kejadian.

Quality of Service digunakan untuk melihat rasio dari paket yang hilang, paket yang berhasil dikirim, dan jeda waktu pengiriman data.

Keadilan (Fairness) protokol pada WSN ini harus adil dalam hal pembagian bandwidth untuk setiap node sensor sehingga setiap node sensor dapat mengirimkan data ke sink node dengan jumlah yang sama.

2.2.1 Protokol Pada Layer Transport

Protokol pada WSN dikategorikan menjadi tiga tipe. Tipe yang pertama adalah protokol untuk mengontrol kemacetan jaringan (congestion control). Tipe kedua adalah protokol untuk memastikan keandalan (reliability). Tipe ketiga adalah protokol untuk memastikan keandalan dan mengontrol kemacetan.

Protocol Congestion Control

Protokol pada Congestion Control antara lain:

- Congestion Detection and Avoidance (CODA)
- Congestion Control and Fairness (CCF)
- Priority-Based Congestion Control Protocol (PCCP)

Protocol Reliability

Protokol pada Reliability antara lain:

- Pump Slowly, Fetch Quickly (PSFQ)
- GARUDA

Protocol Congestion Control and Reliability

Protokol pada Congestion Control and Reliability antara lain:

- Sensor Transmission Control Protocol (STCP)
- Event-to-Sink Reliable Transport (ESRT)

2.3 Prinsip Reliable Data Transfer pada Wireless Sensor Network

Banyak aplikasi jaringan komputer tidak hanya Wireless Sensor Network yang memerlukan pengumpulan data tanpa ada data yang hilang (loss). Terdapat empat cara yang dapat digunakan untuk membuat aplikasi transfer data yang reliable pada jaringan komputer:

1. Link-Level Retransmission
2. End-to-End Retransmission
3. Erasure Code
4. Alternative Route

2.3.1 Link-Level Retransmission

Tingkat data yang hilang (loss) pada jaringan wireless lebih tinggi dibandingkan dengan jaringan kabel (wired). Jika persentase data hilang 10% per hop, maka setelah 15 hop persentase menjadi 80%. Saat pada hop ke-n terjadi loss data, maka semua pengiriman data sampai hop ke n-1 akan menjadi tidak berguna. Untuk mengirimkan paket hop ke-n lagi diperlukan sebanyak n-1 kali tambahan pengiriman jika berhasil. Jika menggunakan Link-Level Retransmission hanya dibutuhkan satu kali retransmission (pengiriman ulang) untuk sampai ke posisi yang sama. Untuk masalah efisien link-level retransmission adalah pilihan yang tepat.

Waktu pengiriman bergantung pada berapa jumlah retransmission dan Round trip time (RTT) yang bervariasi. Ini membuat End-to-End Retransmission tidak efisien karena tidak diketahui jumlah RTT. Untuk masalah paket yang loss, pengirim paket harus menyimpan buffer untuk waktu yang lama dan menunggu hingga menerima acknowledgement dari hop berikutnya. Menyimpan buffer pada memori bukan hal yang tepat pada WSN.

2.3.2 End-to-End Retransmission

End-to-End Retransmission adalah salah satu metode yang digunakan di Internet. Cara ini dapat memastikan reliable data tanpa harus mengetahui apa yang terjadi di tengah jaringan. Pada End-to-End Retransmission diperlukan handshake seperti pada komunikasi jaringan komputer. Pada awalnya data dikirim sebagai permintaan transfer. Jika penerima (receiver) memiliki cukup RAM, dan layer aplikasi dapat menerima data tersebut, maka receiver mengirimkan acknowledge untuk permintaan data tersebut. Saat koneksi sudah terbentuk, data yang sebenarnya dapat dikirim. Data tersusun dari beberapa round. Setiap round, pengirim (sender) mengirim paket yang hilang pada round sebelumnya. Diakhir setiap round receiver mengirimkan acknowledge kepada sender yang berisi informasi paket yang hilang. Sender menerima acknowledge tersebut dan mengirim paket yang hilang tersebut. Untuk round pertama terdapat pengecualian karena semua pake pada round sebelumnya tidak ada (belum dikirim).

2.3.3 Erasure Code

Erasure code adalah kode dimana kita dapat membuat ulang m dari n buah pesan ($n > m$). Jika n lebih besar dibanding tingkat data yang hilang, maka akan didapat reliabilitas yang tinggi tanpa transmisi ulang atau pengiriman ulang.

2.3.4 Alternate Route

Menambahkan rute alternatif untuk menangani kegagalan pada link atau jalur komunikasi adalah cara lain untuk meningkatkan keandalan. Saat link antara dua node sensor terjadi kerusakan maka data yang sedang dikirim akan dibuang (drop) sampai terdapat rute baru untuk mengirim data.

2.4 Pengembangan Pemrograman pada Wireless Sensor Network

2.4.1 Tabel

Berikut adalah contoh pembuatan tabel. Penempatan tabel dan gambar secara umum diatur secara otomatis oleh LaTeX, perhatikan contoh di file bab2.tex untuk melihat bagaimana cara memaksa tabel ditempatkan sesuai keinginan kita.

Perhatikan bawa berbeda dengan penempatan judul gambar gambar, keterangan tabel harus diletakkan di atas tabel!! Lihat Tabel 2.3 berikut ini:

Tabel 2.3: Tabel contoh

	v_{start}	$\mathcal{S}_1$	v_{end}
τ_1	1	12	20
τ_2	1		20
τ_3	1	9	20
τ_4	1		20

Tabel 2.4 dan Tabel 2.5 berikut ini adalah tabel dengan sel yang berwarna dan ada dua tabel yang bersebelahan.

Tabel 2.4: Tabel bewarna(1)

	v_{start}	$\mathcal{S}_2$	$\mathcal{S}_1$	v_{end}
τ_1	1	5	12	20
τ_2	1	8		20
τ_3	1	2/8/17	9	20
τ_4	1			20

Tabel 2.5: Tabel bewarna(2)

	v_{start}	$\mathcal{S}_1$	$\mathcal{S}_2$	v_{end}
τ_1	1	12	5	20
τ_2	1		8	20
τ_3	1	9	2/8/17	20
τ_4	1			20

2.4.2 Kutipan

Berikut contoh kutipan dari berbagai sumber, untuk keterangan lebih lengkap, silahkan membaca file referensi.bib yang disediakan juga di template ini. Contoh kutipan:

- Buku: [?]
- Bab dalam buku: [?]
- Artikel dari Jurnal: [?]
- Artikel dari prosiding seminar/konferensi: [?]
- Skripsi/Thesis/Disertasi: [?] [?] [?]
- Technical/Scientific Report: [?]
- RFC (Request For Comments): [?]
- Technical Documentation/Technical Manual: [?] [?] [?]
- Paten: [?]
- Tidak dipublikasikan: [?] [?]
- Laman web: [?]
- Lain-lain: [?]

2.4.3 Gambar

Pada hampir semua editor, penempatan gambar di dalam dokumen LaTeX tidak dapat dilakukan melalui proses *drag and drop*. Perhatikan contoh pada file bab2.tex untuk melihat bagaimana cara menempatkan gambar. Beberapa hal yang harus diperhatikan pada saat menempatkan gambar:

- Setiap gambar **harus** diacu di dalam teks (gunakan *field* LABEL)
- *Field* CAPTION digunakan untuk teks pengantar pada gambar. Terdapat dua bagian yaitu yang ada di antara tanda [dan] dan yang ada di antara tanda { dan }. Yang pertama akan muncul di Daftar Gambar, sedangkan yang kedua akan muncul di teks pengantar gambar. Untuk skripsi ini, samakan isi keduanya.
- Jenis file yang dapat digunakan sebagai gambar cukup banyak, tetapi yang paling populer adalah tipe PNG (lihat Gambar 2.19), tipe JPG (Gambar 2.20) dan tipe PDF (Gambar 2.21)
- Besarnya gambar dapat diatur dengan *field* SCALE.
- Penempatan gambar diatur menggunakan *placement specifier* (di antara tanda [dan] setelah deklarasi gambar. Yang umum digunakan adalah **H** untuk menempatkan gambar **sesuai** penempatannya di file .tex atau **h** yang berarti "kira-kira" di sini.
  Jika tidak menggunakan *placement specifier*, LaTeX akan menempatkan gambar secara otomatis untuk menghindari bagian kosong pada dokumen anda. Walaupun cara ini sangat mudah, hindarkan terjadinya penempatan dua gambar secara berurutan.
  - Gambar 2.19 ditempatkan di bagian atas halaman, walaupun penempatannya dilakukan setelah penulisan 3 paragraf setelah penjelasan ini.
  - Gambar 2.20 dengan skala 0.5 ditempatkan di antara dua buah paragraf. Perhatikan penulisannya di dalam file bab2.tex!
  - Gambar 2.21 ditempatkan menggunakan *specifier* **h**.

Curabitur tellus magna, porttitor a, commodo a, commodo in, tortor. Donec interdum. Praesent scelerisque. Maecenas posuere sodales odio. Vivamus metus lacus, varius quis, imperdiet quis, rhoncus a, turpis. Etiam ligula arcu, elementum a, venenatis quis, sollicitudin sed, metus. Donec nunc pede, tincidunt in, venenatis vitae, faucibus vel, nibh. Pellentesque wisi. Nullam malesuada. Morbi ut tellus ut pede tincidunt porta. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam congue neque id dolor.

Donec et nisl at wisi luctus bibendum. Nam interdum tellus ac libero. Sed sem justo, laoreet vitae, fringilla at, adipiscing ut, nibh. Maecenas non sem quis tortor eleifend fermentum. Etiam id tortor ac mauris porta vulputate. Integer porta neque vitae massa. Maecenas tempus libero a libero posuere dictum. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Aenean quis mauris sed elit commodo placerat. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Vivamus rhoncus tincidunt libero. Etiam elementum pretium justo. Vivamus est. Morbi a tellus eget pede tristique commodo. Nulla nisl. Vestibulum sed nisl eu sapien cursus rutrum.

Nulla non mauris vitae wisi posuere convallis. Sed eu nulla nec eros scelerisque pharetra. Nullam varius. Etiam dignissim elementum metus. Vestibulum faucibus, metus sit amet mattis rhoncus, sapien dui laoreet odio, nec ultricies nibh augue a enim. Fusce in ligula. Quisque at magna et nulla commodo consequat. Proin accumsan imperdiet sem. Nunc porta. Donec feugiat mi at justo. Phasellus facilisis ipsum quis ante. In ac elit eget ipsum pharetra faucibus. Maecenas viverra nulla in massa.

Nulla ac nisl. Nullam urna nulla, ullamcorper in, interdum sit amet, gravida ut, risus. Aenean ac enim. In luctus. Phasellus eu quam vitae turpis viverra pellentesque. Duis feugiat felis ut enim. Phasellus pharetra, sem id porttitor sodales, magna nunc aliquet nibh, nec blandit nisl mauris

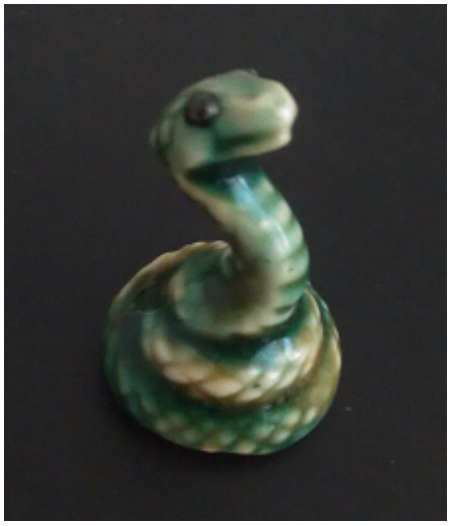
Gambar 2.19: Gambar *Serpentes* dalam format png

at pede. Suspendisse risus risus, lobortis eget, semper at, imperdiet sit amet, quam. Quisque scelerisque dapibus nibh. Nam enim. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nunc ut metus. Ut metus justo, auctor at, ultrices eu, sagittis ut, purus. Aliquam aliquam.

Gambar 2.20: Ular kecil

Etiam pede massa, dapibus vitae, rhoncus in, placerat posuere, odio. Vestibulum luctus commodo lacus. Morbi lacus dui, tempor sed, euismod eget, condimentum at, tortor. Phasellus aliquet odio ac lacus tempor faucibus. Praesent sed sem. Praesent iaculis. Cras rhoncus tellus sed justo ullamcorper sagittis. Donec quis orci. Sed ut tortor quis tellus euismod tincidunt. Suspendisse congue nisl eu elit. Aliquam tortor diam, tempus id, tristique eget, sodales vel, nulla. Praesent tellus mi, condimentum sed, viverra at, consectetuer quis, lectus. In auctor vehicula orci. Sed pede sapien, euismod in, suscipit in, pharetra placerat, metus. Vivamus commodo dui non odio. Donec et felis.

Etiam suscipit aliquam arcu. Aliquam sit amet est ac purus bibendum congue. Sed in eros. Morbi non orci. Pellentesque mattis lacinia elit. Fusce molestie velit in ligula. Nullam et orci vitae nibh vulputate auctor. Aliquam eget purus. Nulla auctor wisi sed ipsum. Morbi porttitor tellus ac enim. Fusce ornare. Proin ipsum enim, tincidunt in, ornare venenatis, molestie a, augue. Donec vel pede in lacus sagittis porta. Sed hendrerit ipsum quis nisl. Suspendisse quis massa ac nibh pretium cursus. Sed sodales. Nam eu neque quis pede dignissim ornare. Maecenas eu purus ac urna tincidunt congue.



Gambar 2.21: *Serpentes* jantan

LAMPIRAN A

KODE PROGRAM

Listing A.1: MyCode.c

```c

// This does not make algorithmic sense,
// but it shows off significant programming characters.

#include<stdio.h>

void myFunction( int input, float* output ) {
    switch ( array[i] ) {
        case 1: // This is silly code
            if ( a >= 0 || b <= 3 && c != x )
                *output += 0.005 + 20050;
            char = 'g';
            b = 2^n + ~right_size - leftSize * MAX_SIZE;
            c = (--aaa + &daa) / (bbb++ - ccc % 2 );
            strcpy(a,"hello $@?");
    }
    count = ~mask | 0x00FF00AA;
}

// Fonts for Displaying Program Code in LATEX
// Adrian P. Robson, nepsweb.co.uk
// 8 October 2012
// http://nepsweb.co.uk/docs/progfonts.pdf
```

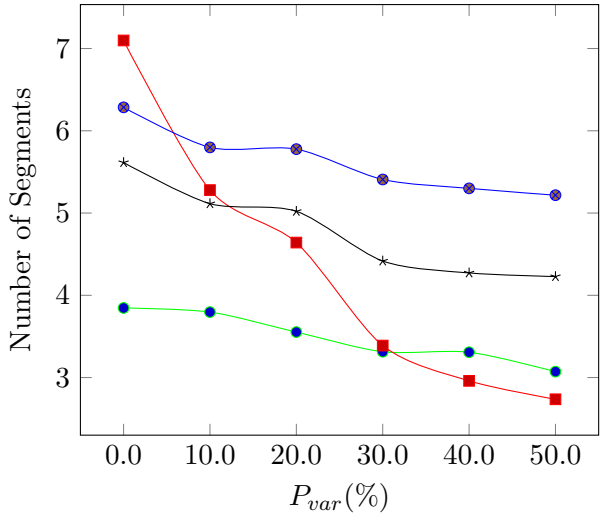
Listing A.2: MyCode.java

```java
import java.util.ArrayList;
import java.util.Collections;
import java.util.HashSet;

//class for set of vertices close to furthest edge
public class MyFurSet {
    protected int id;                                   //id of the set
    protected MyEdge FurthestEdge;                      //the furthest edge
    protected HashSet<MyVertex> set;                    //set of vertices close to furthest edge
    protected ArrayList<ArrayList<Integer>> ordered;    //list of all vertices in the set for each trajectory
    protected ArrayList<Integer> closeID;               //store the ID of all vertices
    protected ArrayList<Double> closeDist;              //store the distance of all vertices
    protected int totaltrj;                             //total trajectories in the set

    /*
     * Constructor
     * @param id : id of the set
     * @param totaltrj : total number of trajectories in the set
     * @param FurthestEdge : the furthest edge
     */
    public MyFurSet(int id,int totaltrj,MyEdge FurthestEdge) {
        this.id = id;
        this.totaltrj = totaltrj;
        this.FurthestEdge = FurthestEdge;
        set = new HashSet<MyVertex>();
        ordered = new ArrayList<ArrayList<Integer>>();
        for (int i=0;i<totaltrj;i++) ordered.add(new ArrayList<Integer>());
        closeID = new ArrayList<Integer>(totaltrj);
        closeDist = new ArrayList<Double>(totaltrj);
        for (int i = 0;i <totaltrj;i++) {
            closeID.add(-1);
            closeDist.add(Double.MAX_VALUE);
        }
    }

}
```

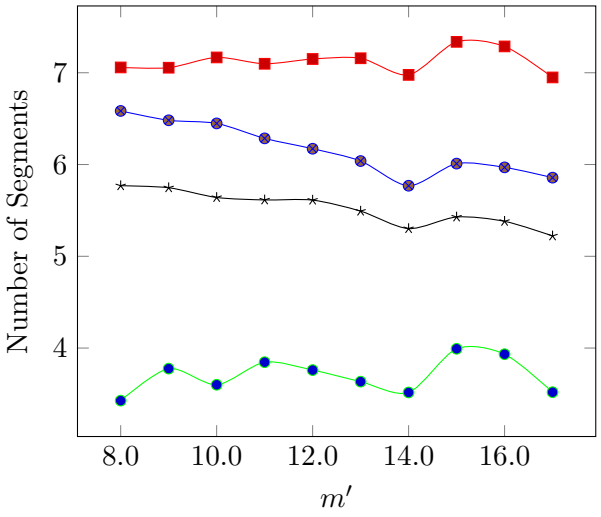
LAMPIRAN B

HASIL EKSPERIMEN

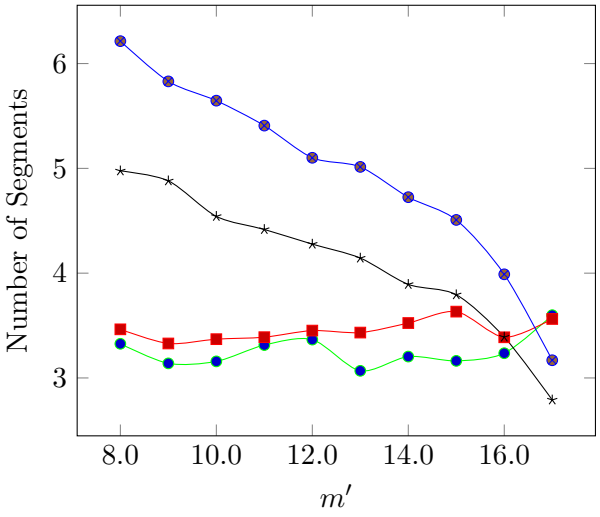
Hasil eksperimen berikut dibuat dengan menggunakan TIKZPICTURE (bukan hasil excel yg diubah ke file bitmap). Sangat berguna jika ingin menampilkan tabel (yang kuantitasnya sangat banyak) yang datanya dihasilkan dari program komputer.



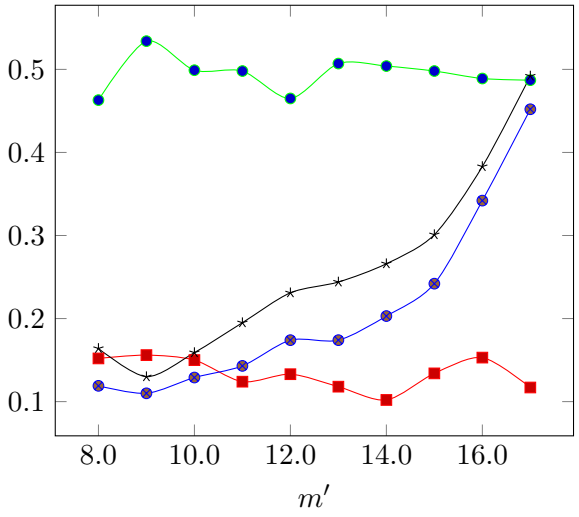
Gambar B.1: Hasil 1



Gambar B.2: Hasil 2



Gambar B.3: Hasil 3



Gambar B.4: Hasil 4