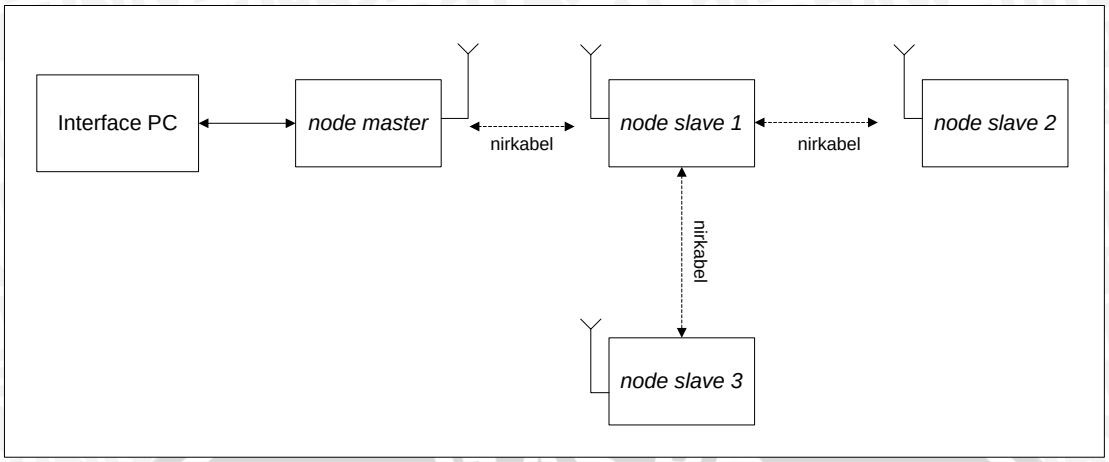
BAB II
TINJAUAN PUSTAKA

Bab tinjauan pustaka membahas kajian pustaka dan dasar teori yang digunakan untuk menunjang penulisan skripsi mengenai *relay routing protocol* jaringan sensor nirkabel pada jaringan lokal. Pada penelitian ini, dasar teori yang diperlukan berdasarkan latar belakang dan rumusan masalah adalah: konsep dasar Jaringan Sensor Nirkabel, konsep dasar TCP/IP, konsep dasar Arduino, konsep dasar XBee, dan konsep dasar Protokol Routing.

2.1 Kajian Pustaka

Telah ada kajian pustaka atau penelitian terkait yang dilakukan sebelumnya yang membahas jaringan sensor nirkabel. Salah satunya adalah penelitian proyek akhir oleh R.Sumiharto yang berjudul "Implementasi Sistem Pemantauan Suhu Menggunakan Jaringan Sensor Nirkabel Multihop". Isu atau teori yang di bahas pada penelitian tersebut adalah jaringan sensor nirkabel multi-hop dengan implementasi sistem pemantauan suhu, menggunakan modul transceiver XBee-PRO yang terintegrasi dengan mikrokontroler ATmega128 dengan menggunakan sensor suhu LM35DZ sebagai *sensing device*. Sistem terdiri dari empat *node*, yaitu *nodemaster* yang bertindak sebagai pengendali dan tiga *node slave*, yang saling terhubung dengan koneksi *multi-hop* nirkabel. Masing-masing *node* akan melakukan pengukuran suhu dengan data yang dapat dikirimkan ke PC yang terhubung dengan *node master* sesuai dengan modus yang diinginkan pengguna (*user*) [SUM-10].

Sistem mempunyai lima mode pengoperasian sistem, yaitu: Mode 1; Menampilkan data sensor node master. Mode 2; Menampilkan data sensor node master dan node slave1. Mode 3; Menampilkan data sensor node master dan node slave2. Mode 4; Menampilkan data sensor node master dan node slave3. Mode 5; Menampilkan data sensor dari semua node [SUM-10]. Diagram blok umum sistem dapat dilihat pada gambar 2.1.



Gambar 2.1 Diagram Blok Umum Sistem
Sumber:[SUM-10]

Hasil perbandingan dari proyek akhir diatas terletak pada perancangan sistem yang digunakan. Sistem yang saya gunakan memiliki kesamaan yaitu pengiriman data dengan 4 node yang dikirim secara multihop ke *sink node* ditambahkan dengan adanya *relay routing* pada routing protocol pada *source node*. Reliabilitas data yang dikirimkan sangat diperhatikan saat data yang dikirim sampai di *sink node*.

2.2 Jaringan Sensor Nirkabel

Jaringan Sensor Nirkabel (*Wireless Sensor Network)* adalah suatu jaringan yang terdiri dari banyak node sensor-sensor yang diletakkan di dekat objek yang akan diteliti. *Sensor node* ini terdiri dari penginderaan, pengolahan data, dan komponen komunikasi. Posisi node sensor bisa ditentukan sebelumnya atau disebar secara random tidak perlu ditentukan sebelumnya [ZHA-04:675]. Sensor tersebut bekerja bersama-sama dan umumnya digunakan untuk memonitor kondisi lingkungan fisik dan gejala alam, antara lain suhu, gerakan, suara, getaran, perubahan warna, dan lain-lain. Beberapa penggunaan Jaringan Sensor Nirkabel lainnya seperti lain dibidang industri, monitoring lingkungan, monitoring bencana alam, monitoring arus lalu lintas, dan lain-lain.

Setiap titik/node sensor biasanya dilengkapi juga dengan mikrokontroler dan sumber energi (biasanya battery atau solar cell). Sebuah Jaringan Sensor

Nirkabel biasanya merupakan jaringan wireless ad-hoc, yang berarti bahwa setiap sensor mendukung algoritma routing *multi-hop* dimana node-node juga berfungsi sebagai *forwarder* yang me-*relay* paket data ke stasiun pusat. Penggunaan arsitektur adhoc dalam jaringan sensor nirkabel dikarenakan arsitektur ini yang paling tepat dan paling murah untuk diterapkan dalam lingkungan nirkabel, mengurangi biaya *key factor* pada banyak jaringan, seperti instalasi, *maintenance* dan kebutuhan operasional lainnya.

2.2.1 Karakteristik Jaringan Sensor Nirkabel

Sebuah jaringan sensor nirkabel memilki karakteristik biaya yang rendah, rendah energi, dan sensor node yang multi fungsi yang dapat mengambil data dan kondisi disekitar sensor node. Sensor node berukuran kecil, akan tetapi memiliki peranan yang sangat besar dalam melakukan *sensing* data yang kemudian data *sensing* tersebut dapat diproses dan dikirim antar sensor node. Proses kirim data antar node inilah yang disebut dengan komunikasi antar *sensor node*. Berikut beberapa karakteristik jaringan sensor nirkabel lainnya [WSN-09]:

a. *Dense Node Deployment*. Pada Jaringan Sensor Nirkabel, jumlah sensor node dapat berjumlah banyak dan lebih banyak daripada MANET (*Mobile Adhoc Networks*).
b. *Battery-Powered Sensor Nodes*. Sensor node dapat menggunakan baterai sebagai daya utama, sehingga memudahkan sensor node untuk diletakkan ditempat yang tidak terdapat daya listrik
c. *Severe Energy, Computation, and Storage Constraints*. Sensor node membtuhkan energi yang sangat terbatas, komputasi dan kapasitas *storage* yang terbatas.
d. *Self - Configurable*. Sensor node biasanya dapat mengatur kemana data yang akan dikirimkan.
e. *Application Specific*. Jaringan sensor nirkabel adalah aplikasi yang spesifik. Sebuah jaringan sensor nirkabel didesain untuk aplikasi yang spesifik.

f. *Unreliable Sensor Nodes.*Sensor node dapat bekerja sendiri tanpa harus dikendalikan secara manual.

g. *Frequent Topology Change*. Topology pada jaringan sensor nirkabel dapat berulang kali berubah menyesuaikan dengan node yang rusak, dsb.

h. *No Global Identification.*. Sendor node tidak mungkin dapat dibuat sebuah *global addressing*.

2.2.2 Arsitektur Jaringan Sensor Nirkabel

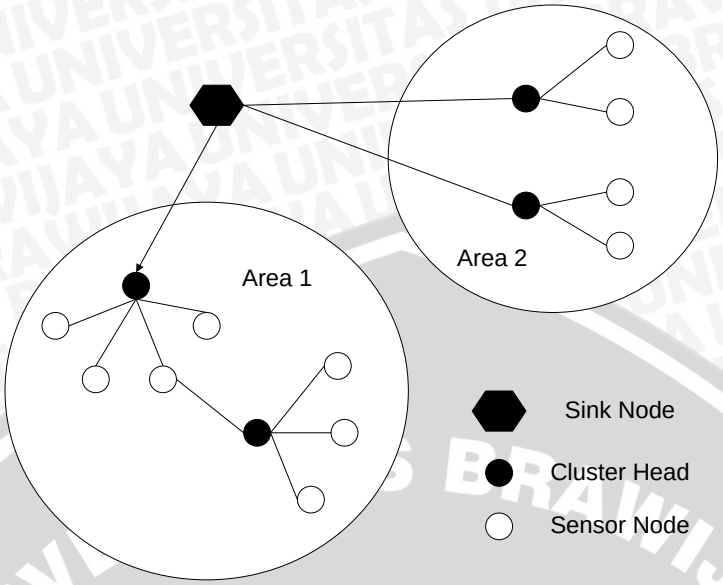
Karakteristik jaringan pada jaringan sensor nirkabel, dibagi menjadi dua kelompok, yaitu karakteristik *sink node*-nya dan karakteristik dari node sensor. Berikut ini adalah arsitektur jaringan sensor nirkabel [ACH-11]:

- ***Flat-based***

  Dalam arsitektur *flat-based* semua node memainkan peran yang sama dan tidak ada yang secara terpusat (hirarki). *Flat routingprotocol* mendistribusikan informasi yang diperlukan untuk setiap *node* sensor yang terjangkau dalam jaringan sensor. Tidak ada upaya dilakukan untuk mengatur jaringan atau trafik, hanya untuk menemukan rute terbaik dengan hop-hop ke tujuan dengan jalan manapun.

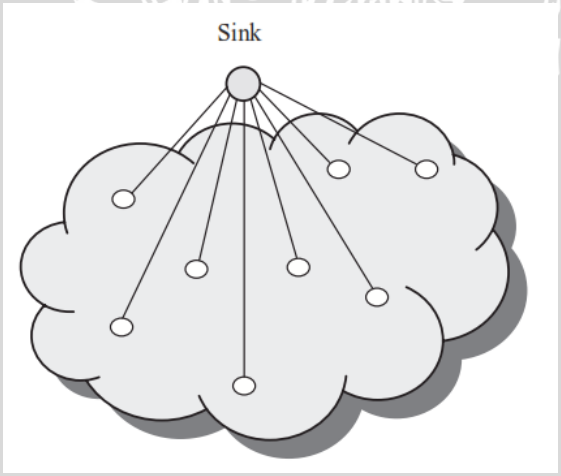
- ***Hierarchical-based***

  Arsitektur ini menetapkan *routing protocol* untuk mencoba menghemat energi dengan mengatur node dalam cluster. Node-node dalam cluster mengirimkan data ke cluster head, dan cluster head inilah yang meneruskan data ke base station. Clustering yang bagus memainkan peran penting dalam skalabilitas jaringan serta penghematan energy. Sisi negative dari struktur jaringan ini adalah cluster dapat menyebabkan kemacetan, ini karena hanya satu kepala berkomunikasi atas nama seluruh cluster, sehingga penurunan energy terbesar akan terjadi pada cluster head tersebut. Jaringan berbasis hirearki dapat dilihat di gambar berikut.



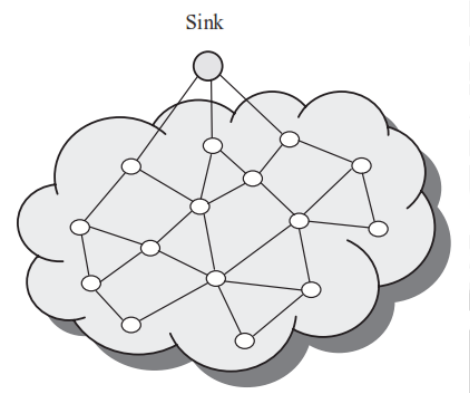
Gambar 2.2 Jaringan Hierarchical-based
Sumber: [Perancangan]

Sedangkan arsitektur jaringan sensor nirkabel berdasarkan karakteristik node sensornya dibagi menjadi dua kategori, yaitu *Single-Hop* dan *Multi-Hop*. Berikut gambar diagram dari single hop :



Gambar 2.3 Topology WSN Single-Hop
Sumber : [WSN-09]

Pada topologi *Single-Hop*, data yang diperoleh dari sensor node langsung dikirimkan ke data sink tanpa melalui sensor node yang lainnya. Sedangkan pada topologi multi-hop seperti gambar 2.4, data yang diperoleh dari sensor node dikirimkan melalui antar sensor node terdekat yang kemudian dikirimkan ke data sink.



Gambar 2.4 Topology WSN Multi-Hop
Sumber : [WSN-09]

2.3 Protokol Routing

Protokol adalah sebuah aturan atau standar yang mengatur atau mengijinkan terjadinya hubungan, komunikasi, dan perpindahan data antara dua atau lebih perangkat [PRO-13]. Protokol dapat diterapkan pada perangkat keras, perangkat lunak atau kombinasi dari keduanya. Pada tingkatan yang terendah, protokol mendefinisikan koneksi perangkat keras. Hal-hal yang perlu diperhatikan mengenai protokol adalah [PRO-13]:

1. melakukan deteksi adanya koneksi fisik atau ada tidaknya komputer atau mesin lainnya.
2. Melakukan metode "jabat-tangan"(*handshaking*).
3. Negosiasi berbagai macam karakteristik hubungan.
4. Prosedur untuk mengawali dan mengakhiri suatu pesan.
5. Format pesan yang digunakan.
6. Hal yang harus dilakukan saat terjadi kerusakan pesan atau pesan yang tidak sempurna.
7. Mendeteksi *failure* pada hubungan jaringan dan langkah-langkah yang dilakukan selanjutnya.
8. Mengakhiri suatu koneksi.

Pada pembuatan protokol ada tiga hal yanng harus dipertimbangkan, yaitu efektivitas, kehandalan, dan fleksibilitas. Pada suatu komunikasi data, informasi yang dikirimkan/diterima terletak antara *header* (penanda awal protokol) dan *end delimier* (penanda akhir data).

Routing protocol adalah suatu protokol yang digunakan untuk mendapatkan rute dari satu jaringan ke jaringan yang lain. Rute ini disebut dengan rute dan informasi route secara dinamis dapat diberikan ke router yang lain ataupun dapat diberikan secara statis ke router lain. *Routing protocol* disebut juga komunikasi antara router-router. Routing protocol mengijinkan router-router untuk *sharing* informasi tentang jaringan dan koneksi antar *router*. *Router* menggunakan informasi tersebut untuk membangun dan memperbaiki table routingnya [ACH-11].

Semua *routing protocol* bertujuan mencari rute tersingkat untuk mencapai tujuan. Masing-masing protokol mempunyai cara dan metodenya sendiri. Secara garis besar, routing protokol dibagi menjadi dua, antara lain [KUN-11]:

a. *Interior Routing Protocol*

Interior routing protocol digunakan dalam sebuah jaringan yang dinamakan *autonomus systems* (AS). AS dapat diartikan sebagai sebuah jaringan(bisa besar atau pun kecil) yang berada dalam satu kendali teknik. AS bisa terdiri dari beberapa sub jaringan yang masing-masingnya mempunyai *gateway* untuk saling berhubungan. *Interior routing protocol* mempunyai beberapa macam implementasi protokol, yaitu [MOD-11]:

- RIP (*Routing Information Protocol*)

  RIP mengirimkan *routing table* yang lengkap ke semua *interface* yang aktif setiap 30 detik, RIP hanya menggunakan jumlah *hop* untuk menentukan cara terbaik ke sebuah *remote* jaringan, tetapi RIP secara *default* memiliki jumlah *hop maximum*, yaitu 15 hop. Hal tersebut berarti nilai 16 dianggap tidak terjangkau (*unreachable*). RIP bekerja dengan baik di jaaringan yang kecil, tetapi RIP tidak efisien pada network yang besar dengan link WAN yang lambat atau pada jaringan yang memiliki jumlah *router* yang banyak.

  RIP versi 1 menggunakan hanya *classful routing*, yang berarti semua alat di *network* harus menggunakan *subnet mask* yang sama, hal tersebut dikarenakan RIP versi 1 tidak mengirimkan *update* dengan informasi subnetmask didalamnya. RIP versi 2 menyediakan *prefix routing* yang bisa mengirimkan informasi *subnet mask* bersama dengan *update-update* dari *route*r.

- OSPF (*Open Shortest Path First*)

Open Shorthest Part First (OSPF) didesain sebagai penggantian dari RIP dan mengambil dari versi sebelumnya dari *Intermediate System to IntermediateSystem* (IS-IS). OSPF adalah protokol yang handal dengan fasilitas *least-costrouting*, *multipath routing* dan *load balancing*. Penetuan jalur tercepat dan terbaik pada jaringan dihitung melalui metode algoritma Djikstra. Pertama, sebuah pohon jalur terpendek (*shortest path tree*) akan di bangun, dan kemudian *routing table* akan diisi dengan jalur-jalur terbaik yang dihasilkan dari pohon tersebut. OSPF melakukan *coverage* dengan cepat dan OSPF mendukung *multiple route* dengan *cost* (biaya) yang sama, ke tujuan yang sama. OSPF seharusnya dirancang dengan cara hierarkis, yang berarti dapat memisahkan internetwork yang lebih besar menjadi *internetwork-internetwork* yang lebih kecil yang disebut dengan area.

b. *Exterior Protocol*

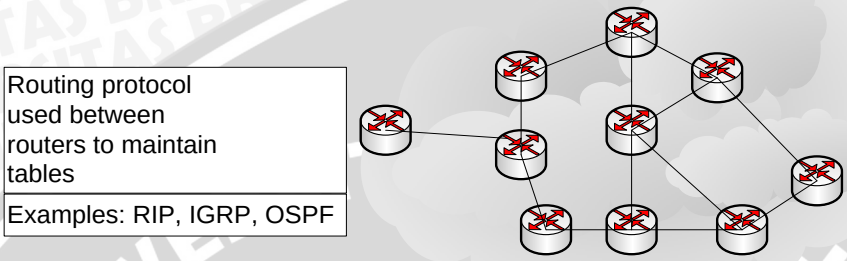
AS merupakan sebuah network dengan sistem *policy* yang pegang dalam satu pusat kendali. Internet terdiri dari ribuan AS yang saling terhubung. Untuk bisa saling berhubungan antara AS, maka tiap-tiap AS menggunakan *exterior protocol* untuk pertukaran informasi routingnya. Informasi routing yang dipertukarkan bernama informasi keterjangkauan (*reachability information*). Tidak banyak *router* yang menjalankan protokol routing ini. Hanya router utama dari sebuah AS yang menjalankannya. Protokol yang mengimplementasikan *exterior protocol* antara lain [ACH-11]:

- EGP (*Exterior Gateway Protocol*)

Protokol ini mengumumkan ke AS lainnya tentang jaringan yang berada di bawahnya. Pengumumannya kira-kira berbunyi: ”Jika hendak pergi ke AS nomor sekian dengan nomor jaringan sekian, maka silahkan melewati saya”. *Router* utama menerima *routing* dari router-router AS yang lain tanpa mengevaluasinya. Maksudnya, rute untuk ke sebuah AS bisa jadi lebih dari satu rute dan EGP menerima semuanya tanpa mempertimbangkan rute terbaik.

- BGP (*Border Gateway Protocol*)

BGP sudah memiliki cara kerja mempertimbangkan rute terbaik untuk dipilih. Seperti EGP, BGP juga bertukar reachability information. Router menggunakan informasi ini untuk membangun dan memperbaiki table routingnya. Seperti terlihat pada gambar 2.5.



Gambar 2.5 Routing Protocol
Sumber: [ACH-11]

2.4 Relay Routing Protocol

Pada sebuah jaringan sensor nirkabel jika terdapat simpul-simpul khusus yang mempunyai fungsi utama untuk meneruskan paket yang dipergunakan maka operasional jaringan dan pengelolaan jaringan sensor nirkabel tersebut secara potensial dapat disederhanakan secara drastis. Simpul-simpul yang meneruskan paket disebut "*relay nodes*" [AAN-12]. Masalah pada penempatan relay node dapat dikategorikan pada *single tiered* atau *two tiered* sesuai dengan skema penerusan data yang diadopsi oleh jaringan sensor nirkabel. Jika relay nodes dan *ordinay sensor nodes* dapat melewatkan paket maka hal ini disebut sebagai "*single–tiered relay node placement*". Untuk kondisi "*two–tiered relay node placement*" hanya *relay nodes* yang dapat melewatkan paket. Pada sistem *two tiered*, *relay nodes* membentuk sebuah *backbone virtual* dan harus terhubung dengan jaringan sensor nirkabel [AAN-12].

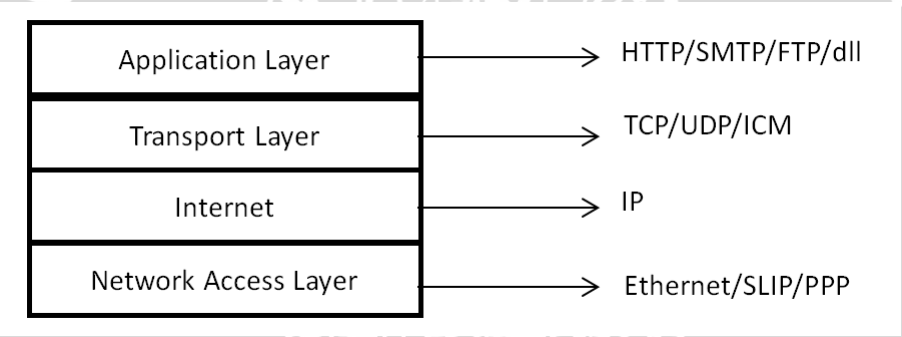
Relay node sendiri fungsi utamanya adalah sebagai pencegah kegagalan *link*. *Relay node* ini nantinya akan menghasilkan topologi jaringan baru agar proses pengiriman data tetap berlangsung. Ketika tidak ada rute lain antara *source node* dan node tujuan, *source node* akan mengirimkan salinan paket ke node tetangga bahwa memenuhi untuk pengiriman ke tujuan. Relay node hanya

diperbolehkan untuk mengirim informasi ke node tujuan, dan tidak diperbolehkan untuk mengirim paket ke node relay lain. Hal tersebut dapat dilakukan dengan asumsi bahwa paket-paket pada node relay memiliki umur terbatas dalam jaringan. Protokol relay terkait dengan protokol relay dua-hop yang disebut epidemi routing protokol.

2.5 Jaringan TCP/IP

TCP/IP adalah sebuah arsitektur jaringan yang dipakai dalam jaringan internet. TCP/IP dilahirkan oleh ARPANET, sebuah pusat riset yang disponsori oleh DoD (Departemen Pertahanan AS). Sampai sekarang model ini sudah berkembang sedemikian pesatnya [DDU-07:2].

TCP/IP terdiri dari 4 layer (lapis) yang berisikan bermacam-macam protokol. Susunan layer tersebut adalah sperti gambar 2.6 berikut [DDU-07:2]:



Gambar 2.6 Susunan Layer pada Protokol TCP/IP
Sumber: [DDU-07:2]

Tiap layer dari TCP/IP ini mempunyai fungsi-fungsi yang berbeda dan didalam tiap layer ini dapat bervariasi protokol yang bekerja pada layer tersebut, karena itu struktur dan ukuran data yang terdapat pada tiap layer juga tidak sama. Meskipun demikian format data setiap layer-nya dibuat saling kompatibel agar transmisi data menjadi efisien [DDU-07:2].

Masing-masing layer TCP/IP mempunyai headeryang berisi informasi kontrol. Proses penambahan dan pengurangan setiap paket data dengan headermasing-masing layer dilakukan dengan proses enkapsulasi dan dekapsulasi. Untuk mengirimkan sebuah data mesin akan melakukan enkapsulasi header, dan sebaliknya untuk menerima data mesin akan melakukan dekapsulasi header lalu mengolah informasi-informasi kontrol yang terdapat di dalamnya [DDU-07:2].

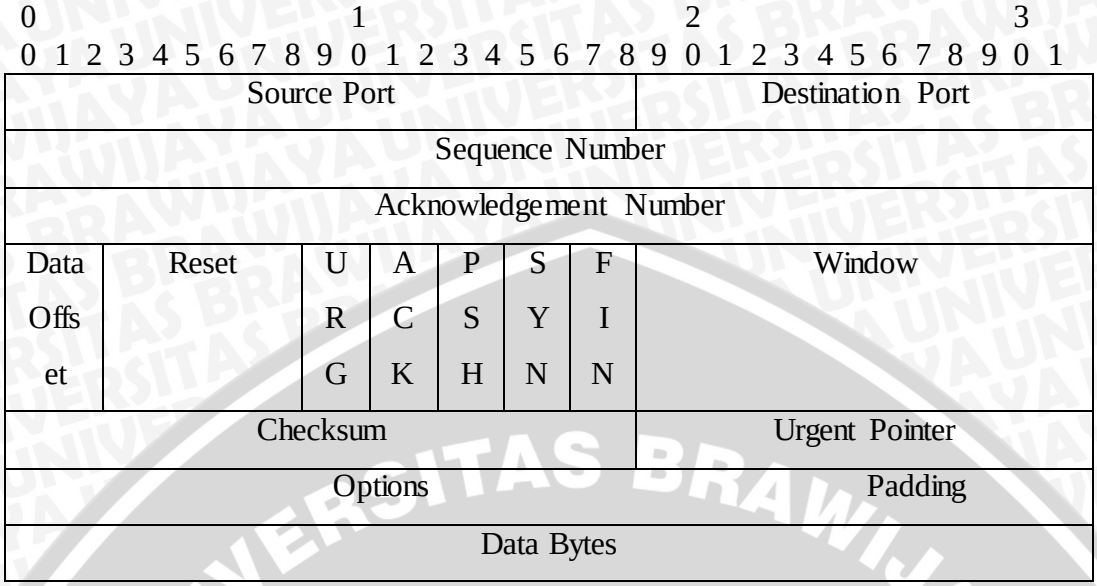
Pada implementasi ini, protokol pada Network Access Layer dipakai SLIP karena merupakan protokol yang dibutuhkan jika koneksinya menggunakan modem, pada Internet Layer dipilih IP yang merupakan protokol yang harus ada untuk bisa koneksi ke internet, pada Transport Layer dipilih TCP untuk mengarahkan port aplikasi ke HTTP, dan pada Application Layer dipilih HTTP [DDU-07:2].

Protokolnya sangat sederhana, host cukup mengirimkan paket-paket IP melalui saluran dengan byte flagkhusus (0xC0) untuk keperluan pembuatan frame. Bila terjadi byte flag di dalam paket data IP, maka digunakan pengisian karakter (byte stuffing), dan dua byte yang berturutan (0xDB dan 0xDC) disisipkan menggantikan byte tersebut. Bila yang terjadi di dalam paket IP adalah 0xDB, maka dalam hal ini juga akan digunakan pengisian karakter, yaitu 0xDB dan 0xDD [DDU-07:2].

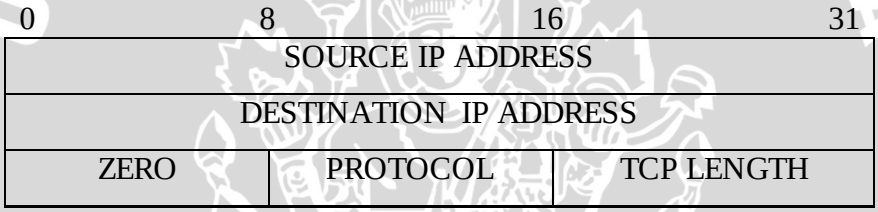
2.5.1 Transmission Control Protocol (TCP)

Satuan paket data dari TCP disebut segment. Sebuah segment terdiri dari header dan data. Header mempunyai bagian tetap 20 bytes dan bagian optional yang panjangnya dapat berubah-ubah. Berikut adalah format header TCP. Header dikirimkan dari kiri ke kanan [DDU-07:2].

TCP menambahkan pseudo-header untuk keperluan perhitungan checksum pada segment-nya. Sehingga perhitungan checksum pada TCP meliputi headerdan data TCP serta pseudo-header TCP. Pseudoheader ini hanya berguna untuk perhitungan checksum saja dan tidak ikut dikirimkan bersama dengan header dan data. Maksud penggunaan pseudo-header ini adalah untuk memastikan bahwa segment telah mencapai tujuannya dengan benar [DDU-07:3]. Berikut gambar 2.7 dan 2.8 menjelaskan format TCP dan pseudo-header TCP.



Gambar 2.7 Format TCP Header Segment
Sumber: [DDU-2007:3]



Gambar 2.8 Format Pseudo-header TCP
Sumber: [DDU-2007:3]

Keterangan [PJK-07:64]:

- Source Port : 16 bit nomer port. Digunakan untuk menerima reply
- Destination port : 16 bit nomer port tujuan
- Sequence Number : nomer awal data pada segmen
- Acknowledge number : apabila ACK diset maka ini menjadi nomer urut data yang dikirim
- Data offset : nomer dimana bagian data mulai
- Reserved : untuk kegunaan masa depan, diset 0
- URG : mengaktifkan titik yang darurat pada suatu segmen
- ACK : kolom acknowledge
- PSH : fungsi push
- RST : mereset suatu koneksi
- SYN : untuk mensinkronisasi nomer urutan

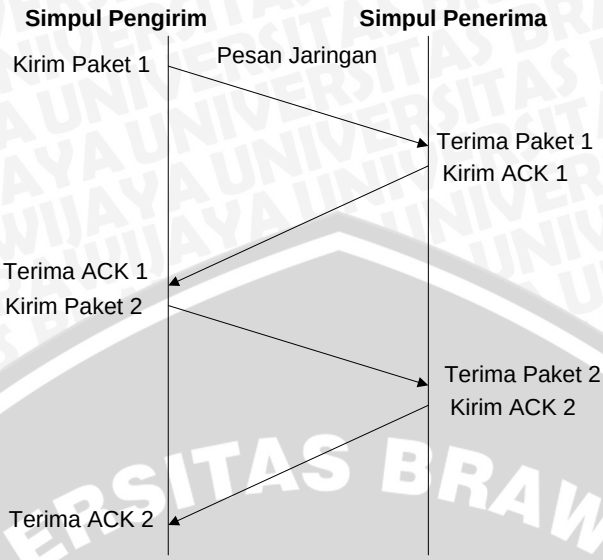
- FIN : batas akhir data
- Window : nomer window untuk proses windowing
- Checksum : nomer yang digunakan untuk mengecek validitas pengirim dan penerima
- Urgent Pointer : menunjuk pada titik yang darurat pada suatu segmen
- Options : digunakna untuk pilihan lain pada datagram
- Padding : digunakan untuk membulatkan data pada bagian options

2.5.2 Cara Kerja TCP

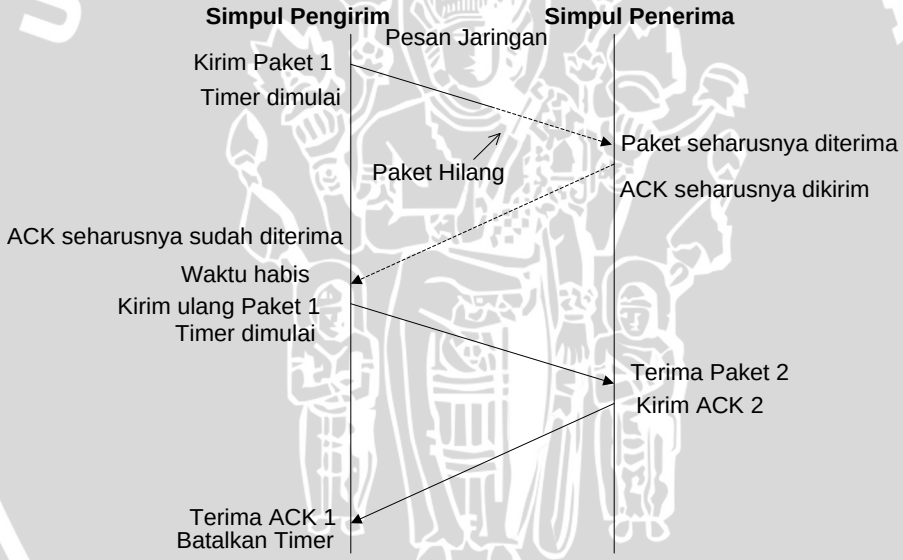
Seperti telah disebutkan di atas, TCP melakukan *handshaking* untuk memulai koneksi, mengakhiri koneksi, dan mengirim data. Untuk menjelaskan cara kerja TCP dalam melakukan handshaking dapat dimisalkan dengan sebuah jaringan client-server sederhana [DDU-07:3].

Suatu simpul memulai hubungan (establish connection) bila simpul tersebut akan berkomunikasi dengan simpul yang lain dalam jaringan. Proses ini dilakukan dengan membentuk proses client dengan menggunakan TCP. Selanjutnya proses client akan mengirimkan permintaan ke simpul tujuan, yang cara pengalamatannya diatur oleh IP. Proses server pada simpul tujuan akan menanggapi permintaan ini dengan mengirimkan jawaban ke client. Dengan demikian terbentuklah hubungan semu antara dua simpul. Aplikasi pada lapis yang lebih tinggi dari Transport Layer akan menggunakannya untuk komunikasi. Setelah berkomunikasi, lapis yang lebih tinggi akan memberitahukan Transport Layer bahwa jalur telah selesai digunakan dan hubungan dapat diputuskan.TCP mengirim paket data dalam bentuk urutan yang disebut stream. Data dari lapis di atasnya dipecah menjadi beberapa paket dan dikirimkan secara berurutan. Penerima akan menerimanya secara berurutan pula. Pengiriman ini dilakukan melalui untai semu yang dibentuk pada awal hubungan [DDU-07:3].

Hubungan yang dibentuk oleh TCP bersifat full duplex, yaitu pertukaran data dapat berlangsung dua arah pada waktu yang sama. Sifat full duplex ini juga dapat dimanfaatkan untuk menyisipkan informasi kontrol [DDU-07:3].



Gambar 2.9 Teknik Positive Acknowledgement
Sumber: [DDU-07:3]



Gambar 2.10 Timeout dan Retransmission
Sumber: [DDU-07:3]

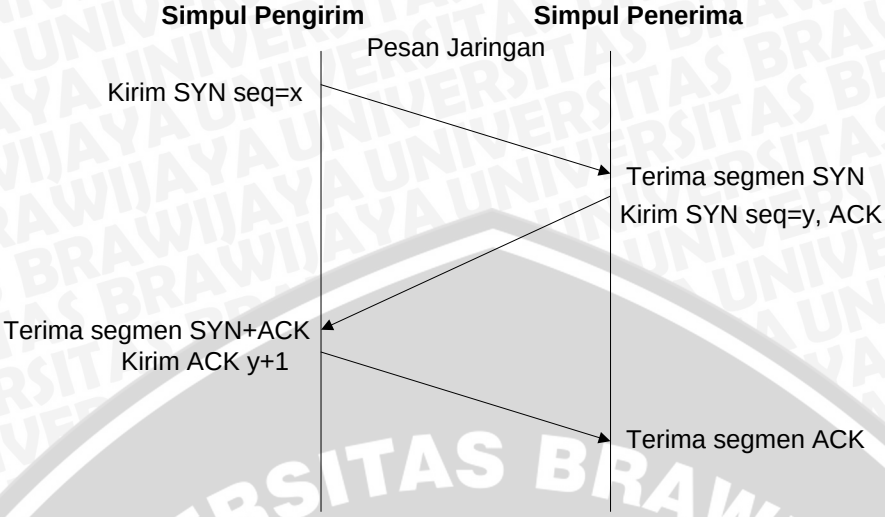
Pada gambar 2.9 dan 2.10 diatas menjelaskan proses teknik positive acknowledgement dan timeout retransmission. Informasi kontrol ini berupa bukti terima (ACK) untuk setiap paket data yang dikirim. Teknik pengiriman data dengan bukti terima ini disebut positive acknowledment. Simpul pengirim mengirimkan permintaan data ke simpul penerima. Simpul penerima membalasnya dengan mengirimkan data yang diminta dan disisipi dengan

informasi kontrol bukti terima, yaitu bahwa simpul penerima telah menerima data dari simpul pengirim dengan sempurna [DDU-07:3].

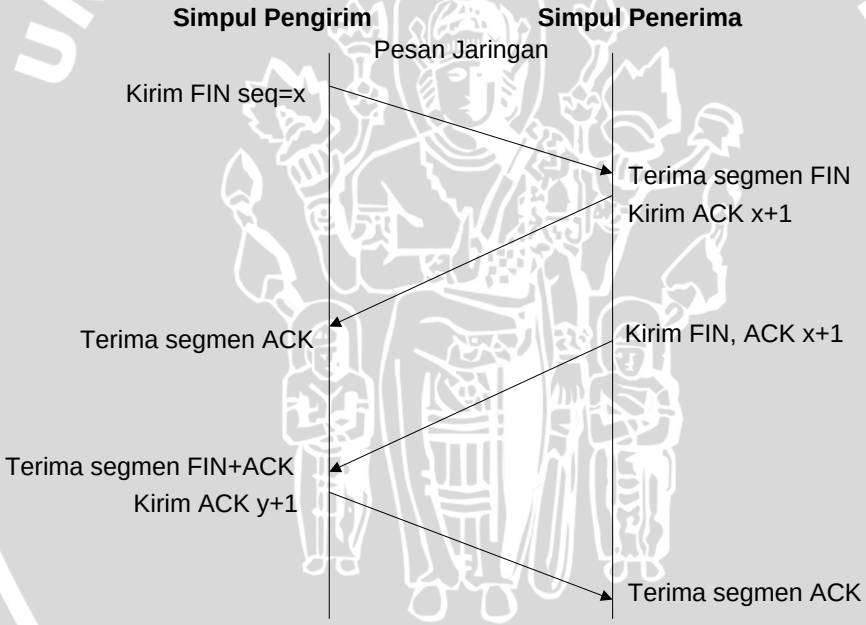
Bila suatu saat terjadi kehilangan atau kerusakan pada paket data, simpul penerima akan menunggu terus sedangkan simpul pengirim tidak akan menerima bukti terima dan kemungkinan akan menungguterus datangnya bukti terima tersebut. Akibatnya sistem jaringan akan menggantung (*hang up*). Untuk menghindari hal tersebut di atas digunakan cara timeout, yaitu setelah mengirimkan data, simpul pengirim akan menunggu datangnya bukti terima selama selang waktu tertentu. Setelah selang waktu ini habis dan bukti terima belum diterima, simpul pengirim akan timeout. Kemudian data yang hilang tersebut akan dikirimkan kembali oleh simpul pengirim (*retransmission*) [DDU-07:4].

2.5.3 Membuka dan Menutup Koneksi TCP

Seperti telah dijelaskan sebelumnya, agar dua simpul dapat berkomunikasi dengan menggunakan protokol TCP, simpul tersebut harus membentuk hubungan TCP. Di sini kedua simpul akan bertukar informasi kontrol (handshake) untuk membentuk suatu untai semu yang menyatakan kedua belah pihak telah siap untuk melakukan transfer data yang sesungguhnya. Untuk membuka koneksi ini TCP menggunakan metode three-way-handshake [DDU-07:4]. Untuk lebih jelasnya, gambar 2.11 dan 2.12 dibawah menjelaskan pembukaan dan penutupan koneksi *three way handshake*



Gambar 2.11 Three-way-handshake Pembukaan Koneksi
Sumber: [DDU-07:4]



Gambar 2.12 Three-Way-Handshake Menutup Koneksi
Sumber: [DDU-07:4]

Secara garis besar proses ini berlangsung untuk tukar menukar nomor urut (Sequence Number) antara kedua simpul agar didapat perjanjian mengenai Initial Sequence Number (nomor urut pertama) yang akan digunakan oleh masing-masing simpul.Segment pertama pada proses handshake dapat diidentifikasi dari bit SYN yang diset '1'. Pada segment kedua bit SYN dan ACK keduanya diset '1'. Segment kedua ini menyatakan bahwa segmentpertama sudah diterima dan

simpul penerima mengirimkan tanda terima (ACK). Pada segmentterakhir hanya bit ACK saja yang diset '1', yang menandakan bahwa antara kedua simpul telah terbentuk hubungan [DDU-07:4].

Hal lain yang perlu mendapat perhatian adalah Sequence Number dan *Acknowledment Number*. Masing-masing simpul akan memulai koneksi dengan *Sequence Number*-nya sendiri-sendiri yang biasanya dibangkitkan secara acak. Misal simpul pertama memulai dengan *Sequence Number* (seq) = x, simpul kedua dengan seq = y. Segment pertama dikirimkan dengan seq = x. Segment kedua dari simpul penerima dikirim dengan seq = y dan tanda terima untuk ack = x+1 (*Acknowledment Number*berisi nomor urut dari byte data yang diharapkan diterima berikutnya). Pada segment terakhir, segment dengan seq = y yang telah diterima diberi tanda terima dengan ack = y+1 [DDU-07:4].

Ketika program aplikasi memberitahu TCP bahwa sudah tidak ada lagi data yang dikirim, TCP akan menutup koneksi dalam satu arah dengan *three-way-handshake* yang dimodifikasi. Untuk menutup setengah koneksi yang lain (satu arah yang lain), simpul yang sedang mengirimkan segment akan menyelesaikan pengiriman sisa data, menunggu tanda terima dari penerima, dan kemudian mengirimkan segment dengan bit FIN diset '1'. TCP penerima akan memberi tanda terima untuk segment FIN tersebut dan memberitahu program aplikasi bahwa sudah tidak ada lagi data dari simpul pengirim. Saat koneksi ditutup satu arah, data hanya dapat mengalir pada arah yang lain saja sampai koneksi ditutup seluruhnya oleh simpul yang lain [DDU-07].

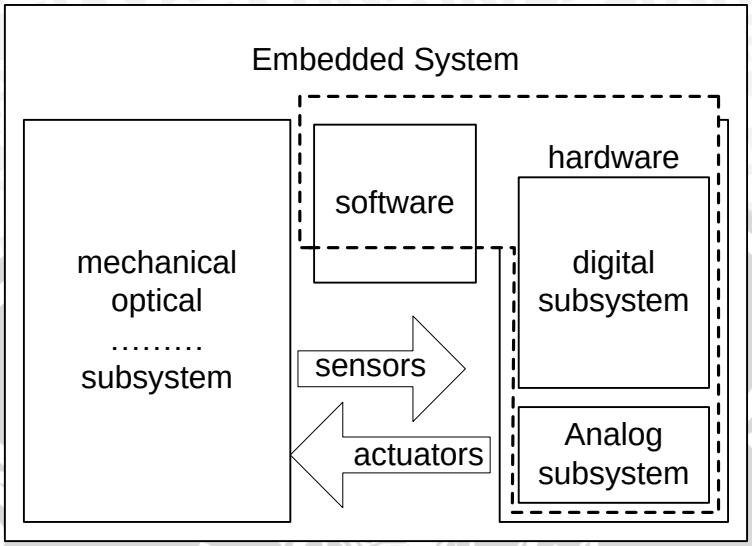
Perbedaan *three-way-handshake* untuk menutup dan membuka koneksi ialah bahwa untuk menutup koneksi ditandai dengan segment dengan bit FIN diset '1'. Segment ini akan dibalas dengan ACK dan diikuti oleh pemberitahuan kepada program aplikasi bahwa koneksi telah ditutup satu arah. Pada akhirnya setelah program aplikasi selesai mengirimkan datanya, TCP diperintahkan untuk menutup koneksi seluruhnya dengan mengirimkan segment dengan bit FIN diset '1'. Yang kemudian akan diberikan tanda terima (ACK) terakhir oleh penerimanya [DDU-07].

2.6 Embedded System

Embedded system atau sistem tertanam adalah sistem komputer yang dibuat untuk melakukan tujuan atau fungsi tertentu yang didalamnya terdapat susunan perangkat keras dan komponen-komponen mekanik. Komponen-komponen penyusun *embedded system* ini dikoordinasikan oleh satu atau lebih *chip* mikrokontroler yang yang telah diprogram untuk melakukan fungsi tertentu. Karena dibangun untuk fungsi tertentu saja, maka pada umumnya komponen utama *embedded system* dapat memiliki ukuran yang lebih kecil dengan jumlah dan jenis komponen penyusunnya sesuai dengan kebutuhan saja. Dengan demikian biaya produksinya dapat ditekan. Hal ini berbeda dari sistem komputer umum, seperti *personal computer*, yang lebih fleksibel karena dapat memenuhi berbagai kebutuhan pengguna. Sebagai konsekuensinya, *personal computer* harus menyediakan berbagai macam komponen yang dapat memenuhi kebutuhan pengguna secara umum. Akibatnya, ukuran komponen utamanya pun menjadi lebih besar dan biaya produksinya juga menjadi lebih mahal. Namun pada beberapa kasus, *embedded system* mungkin saja berupa suatu sistem dengan skala yang sangat besar dan bernilai sangat mahal, seperti *embedded system* pada pembangkit listrik tenaga nuklir atau sistem *control* pabrik. Pada intinya, yang membedakan antara *embedded system* dengan sistem komputer biasa adalah spesifikasi dari fungsinya [EMB-12:2].

Suatu *embedded system* biasanya memiliki sensor-sensor sebagai masukan, seperti sensor panas, sensor posisi (GPS), sensor pengukur jarak, sensor guncangan, dan lain-lain. Selain sensor, *embedded system* juga dilengkapi dengan komponen yang berfungsi menanggapi hasil penerimaan sensor setelah diproses, yaitu dapat berupa motor penggerak, layar, ataupun menggunakan modem untuk berkomunikasi dengan suatu *server*. Perangkat masukan dan keluaran dalam suatu *embedded system* dapat bersifat *digital* jika perangkat tersebut menerima masukan atau memberi hasil dalam bentuk *digital* atau *analog* jika masukan maupun keluarannya berupa sinyal *analog*. Semua perangkat masukan dan keluaran dari *embedded system* ini dikoordinasikan oleh logika yang telah diprogram ke dalam *micro controller* yang ditanamkan dalam sistem ini, atau lebih dikenal sebagai *firmware*. Inilah inti dari pembuatan suatu *embedded system*, karena program

dalam *micro controller* itulah yang menentukan pemrosesan dan hasil yang dikeluarkan. Gambar 2.13 berikut adalah susunan standar dari suatu *embedded system* [EMB-12:2]:



Gambar 2.13 Susunan Standar Embedded System
Sumber :[EMB-12:2]

2.7 Papan Arduino

Arduino adalah papan rangkaian elektronik yang sifatnya open source dan di dalamnya terdapat komponen utama yaitu sebuah chip mikrokontroler dengan jenis AVR dari perusahaan Atmel [SCR-12:4]. Arduino didefinisikan sebagai sebuah platform elektronik yang *open source*, berbasis pada software dan hardware yang fleksibel dan mudah digunakan, yang ditujukan untuk para desainer elektronik dan setiap orang yang tertarik dalam membuat objek atau lingkungan yang instruktif [ART-11:1].

Beberapa keunggulan yang dimiliki oleh Arduino antara lain [ART-11:2]:

1. IDE Arduino merupakan *multiplatform*, yang dapat dijalankan di berbagai sistem operasi, seperti Windows, Macintosh, dan Linux
2. IDE Arduino dibuat berdasarkan pada IDE *Processing*, yang sederhana sehingga mudah digunakan.
3. Pemrograman Arduino menggunakan kabel yang terhubung dengan port USB, bukan port serial. Fitur ini berguna karena banyak komputer yang sekarang ini tidak memiliki port serial.

4. Arduino adalah *hardware* dan *software open source*. User bisa men-download software dan gambar rangkaian Arduino tanpa harus membayar ke pembuat Arduino.
5. Biaya *hardware* yang cukup murah sehingga tidak terlalu menakutkan untuk membuat kesalahan.
6. Proyek Arduino dikembangkan dalam lingkungan pendidikan, sehingga bagi pemula akan lebih cepat dan mudah mempelajarinya.
7. Memiliki begitu banyak pengguna dan komunitas di internet yang dapat membantu setiap kesulitan yang dihadapi.

Bahasa pemrograman Arduino adalah bahasa C. Tetapi bahasa ini sudah dipermudah menggunakan fungsi-fungsi yang sederhana sehingga pemula pun bisa mempelajarinya dengan cukup mudah.

2.7.1 Pemrograman Arduino

Struktur dasar dari bahasa pemrograman arduino itu sederhana hanya terdiri dari dua bagian [KTA-12].

```
void setup( )
{
  // Statement;
}
void while( )
{
  // Statement;
```

Gambar 2.14 Struktur Dasar Pemrograman Arduino
Sumber: [KTA-12]

Dimana *setup*() bagian untuk inisialisasi yang hanya dijalankan sekali di awal program, sedangkan *while*() untuk mengeksekusi bagian program yang akan dijalankan berulang-ulang untuk selamanya.

- ***setup*()**

Fungsi setup() hanya di panggil satu kali ketika program pertama kali di jalankan. Ini digunakan untuk pendefinisian mode pin atau memulai komunikasi

serial. Fungsi *setup*() harus diikut sertakan dalam program walaupun tidak ada statement yang di jalankan [KTA-12].

```
void setup()
{
pinMode(13,OUTPUT); //mengset 'pin' 13 sebagai output
```

Gambar 2.15 Fungsi Setup()
Sumber: [KTA-12]

- ***while*()**

Setelah melakukan fungsisetup() maka secara langsung akan melakukan fungsi *while*() secara berurutan dan melakukan instruksi-instruksi yang ada dalam fungsi *while*() [KTA-12].

```
void loop()
{
digitalWrite(13, HIGH);     // nyalakan 'pin' 13
     delay(1000);           // pause selama 1 detik
digitalWrite(13, LOW);      // matikan 'pin' 13
     delay(1000);           // pause selama 1 detik
```

Gambar 2.16 Fungsi While()
Sumber: [KTA-12]

- ***Function***

Function (fungsi) adalah blok pemrograman yang mempunyai nama dan mempunyai statement yang akan di eksekusi ketika function di panggil. Fungsi void setup() dan void loop() telah di bahas di atas dan pembuatan fungsi yang lain akan di bahas selanjutnya [KTA-12].

Cara pendeklarasian function

```
type functionName(parameters)
   {
       // Statement;
```

Gambar 2.17 Pendeklarasian Fungsi
Sumber: [KTA-12]

Contoh

```
intdelayVal()
{
int v; // membuat variable 'v' bertipe integer
v = analogRead(pot); // bacaharga potentiometer
  v /= 4; // konversi 0-1023 ke 0-255
  return v; // return nilai v
```

Gambar 2.18 Pendeklarasian Fungsi
Sumber: [KTA-12]

Pada contoh di atas fungsi tersebut memiliki nilai balik int (integer), karena jika tidak menghendaki adanya nilai balik maka type function harus void [KTA-12].

- ***{ } curly braces***

Curly brace mendefinisikan awal dan akhir dari sebuah blok fungsi. Apabila ketika memprogram dan programmer lupa memberi curly brace tutup maka ketika di compile akan terdapat laporan error [KTA-12].

- ***; semicolon***

Semicolon harus di berikan pada setiap statement program yang kita buat ini merupakan pembatas setiap statement program yang di buat [KTA-12].

- ***/*...*/ block comment***

Semua statement yang di tulis dalam block comments tidakakan di eksekusi dan tidak akan di compile sehingga tidak mempengaruhi besar program yang di buat untuk dimasukan dalam papan arduino [KTA-12].

- ***// line comment***

Sama halnya dengan *block comments*, *line coments* pun sama hanya saja yang dijadikan komen adalah tiap baris [KTA-12].

- ***Variable***

Variable adalah sebuah penyimpan nilai yang dapat digunakan dalam program.*Variable* dapat dirubah sesuai dengan instruksi yang kita buat. Ketika mendeklarisikan *variable* harus diikutsertakan *typevariable* serta nilai awal *variable*.

```
Type variableName = 0;
```

Gambar 2.19 Pendeklarasian Variable
Sumber: [KTA-12]

Contoh:

```
IntinputVariable = 0;  /* mendefinisikan sebuah variable
                          bernama inputVariable dengan nilai
                          awal 0 */


inputVariable = analogRead(2); /* menyimpan nilai yang ada
```

Gambar 2.20 Contoh Pendeklarasian Variable
Sumber: [KTA-12]

❖ ***variable scope***

Sebuah *variable* dapat dideklarasikan pada awal program sebelum *void setup*(), secara lokal di dalam sebuah *function*, dan terkadang di dalam sebuah *block statement* pengulangan. Sebuah variable global hanya satu dan dapat digunakan pada semua *block function* dan *statement* di dalam program. *Variable global* dideklarasikan pada awal program sebelum fungsi *setup*(). Sebuah variable local di deklarasikan di setiap *block function* atau di setiap *block statement* pengulangan dan hanya dapat digunakan pada block yang bersangkutan saja.

```
int value; /* 'value' adalah variable global dan dapat
                di gunakan pada semua block function
                */
  void setup()
  {
    /* no setup needed */
  }

  void loop()
  {

     for (int i=0; i<20;) /*'i' hanya dapat digunakan
                           dalam pengulangan saja*/

  {
```

Gambar 2.21 Contoh Penggunaan Variable Scope pada Pemrograman Arduino
Sumber: [KTA-12]

2.7.2 Tegangan Analog pada Arduino

Semua hal yang dapat dilakukan oleh Arduino berasal dari dua keadaan tegangan yaitu HIGH dan LOW. Pin I/O Arduino memiliki tegangan 0-5v.

Kondisi HIGH memiliki besar tegangan 3-5V dan 2V kebawah untuk kondisi LOW. Dua keadaan inilah yang dapat mengendalikan segala sesatu di yang dihubungkan dengan Arduino, seperti relay, tombol, dll.

Pada beberapa sistem kontrol, pengolahan input dan output secara digital mungkin sudah memenuhi kinerja yang dibutuhkan. Akan tetapi pada kondisi tertentu ada kemungkinan dihadapkan pada kondisi input dan output yang membutuhkan besaran yang berubah-ubah dengan nilai yang kontinyu dan tidak lagi hanya dengan dua keadaan seperti halnya sinyal digital. Sinyal semacam ini disebut sebagai sinyalanalog, sebagai contoh saat kita menghubungkan sensor yang tegangan keluarannya bervariasi dalam kisaran dari 0 volt sampai 5 volt [DEW-13]. Arduino sebagai kontroler harus mampu mengidentifikasi/mengolah semua variasi tegangan keluaran dari sensor yang dihubungkan pada pin inputnya tersebut. Begitu juga halnya saat diperlukan tegangan output yang membutuhkan nilai tegangan yang bervariasi, seperti misalnya saat mengatur tingkat keterangan sebuah led atau berubahnya kecepatan sebuah motor [DEW-13].

❖ **Analog Input**

Arduino khusus menyediakan 6 kanal (8 kanal pada arduino model Mini dan Nano, dan 16 pada model Mega) untuk difungsikan sebagai analog input. Analog ke digital converternya menggunakan resolusi 10 bit yang berarti range nilai analog dari 0 volt sampai 5 volt terdapat 1024 keadaan (2^10= 1024) dan akan dirubah kenilai integer antara 0 sampai 1023, atau resolusinya adalah 5 volt/1024=4,9 mV per unit dimana itu berarti nilai digital yang dihasilkan akan berubah setiap perubahan 4,9 mV dari tegangan input analognya. Akan tetapi range input analog dan resolusi tersebut dapat dirubah dengan fungsi analogReference(). Perintah yang digunakan untuk fungsi analog input ini adalah [DEW-13]:

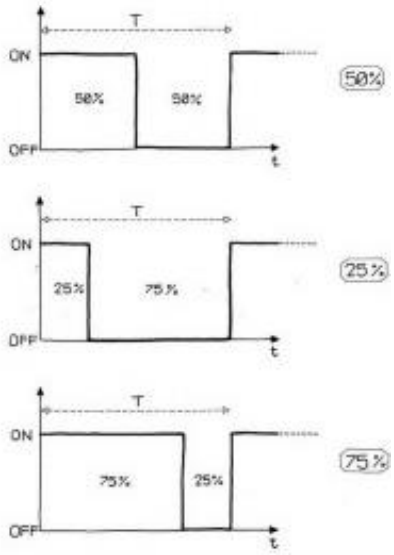
1. analogRead(pin): berfungsi untuk membaca nilai analog pada input pin yang akan menghasilkan nilai integer antara 0-1023.
2. analogReference(parameter): berfungsi untuk menentukan referensi yang digunakan. Parameternya meliputi:

- DEFAULT: default analog reference yaitu 5V (pada board Arduino 5V) atau 3,3 volt (pada board Arduino 3,3 V)
- INTERNAL: built-in referensi internal tergantung pada jenis mikrokontroler yang terpasang pada board Arduino, 1.1 volt pada ATmega168 atau ATmega328 dan 2.56 volt pada ATmega8.
- INTERNAL1V1: a built-in referensi internal 1.1V (khusus Arduino Mega)
- INTERNAL2V56: a built-in referensi internal 2,56V (khusus Arduino Mega)
- EXTERNAL: pilihan referensi yang tergantung pada tegangan yang diberikan pada pin AREF(hanya dengan range tegangan 0 sampai 5V).
- Perlu untuk diperhatikan, jangan menggunakan referensi dibawah 0 volt atau lebih dari 5 volt dan pastikan memilih referensi external sebelum perintah analogRead() jika menghubungkan pin AREF dengan referensi eksternal karena jika tidak akan bisa merusak mikrokontrol.

❖ **Analog Output**

Secara teori suatu analog output akan mengeluarkan output tegangan bervariasi sesuai dengan nilai yang dikehendaki, maka seharusnya pin output analog Arduino seharusnya mampu mengeluarkan tegangan output dengan kisaran tegangan dari 0 V sampai 5V. Akan tetapi tidak demikian adanya, karena pin-pin Arduino yang difungsikan sebagai output sebenarnya hanya mampu sebagai digital output yaitu hanya mampu mengeluarkan tegangan 0V atau 5V. Arduino menggunakan cara *Pulse Wide Modulation* (PWM) atau modulasi lebar pulsa untuk menghasilkan analog output yang dikehendaki. Metode PWM ini menggunakan pendekatan perubahan lebar pulsa untuk menghasilkan nilai tegangan analog yang diinginkan. Pin yang difungsikan sebagai PWM analog output akan mengeluarkan sinyal pulsa digital dengan frekwensi 490 Hz dimana nilai tegangan analog diperoleh dengan merubah *Duty Cycle* atau perbandingan lamanya pulsa HIGH terhadap periode (T) dari sinyal digital tersebut. Jika pulsa HIGH muncul selama setengah dari periode sinyal maka akan menghasilkan duty cycle 50% yang berarti sinyal analog yang dihasilkan sebesar setengah dari tegangan analog maksimal yaitu 1/2 dari 5 V atau sama dengan 2,5 V begitu juga

halnya jika pulsa HIGH hanya seperempat bagian dari periode sinyal maka tegangan analog identik yang dihasilkan adalah 1/4 dari 5V = 1,25 V dan seterusnya.



Gambar 2.22 *Pulse Wide Modulation*
Sumber: [DEW-13]

Perintah yang digunakan untuk output analog adalah analogWrite (pin,value), dimana:

- Pin: nomor pin Arduino yang akan digunakan sebagai analog output
- value: nilai duty cycle yang diinginkan dengan nilai 0-255, yang berarti nilai 0 untuk 0 Volt dan 255 untuk tegangan keluaran maksimum atau 5 Volt.

2.8 Radio Frequency

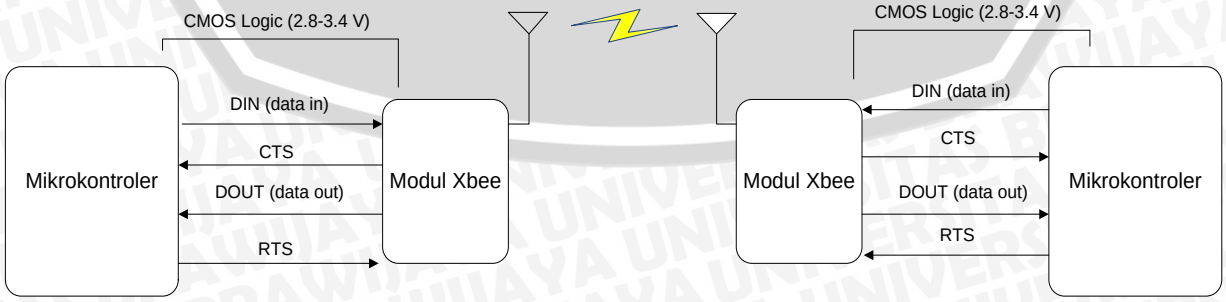
Radio Frequency merupakan sebuah daya dalam bentuk arus bolak-balik yang dihasilkan oleh sebuah peralatan elektronik yang mengalir melalui kabel menuju antena. Kemudian antena akan menyebarkan daya ini yang menciptakan suatu gelombang radio yang bergerak melewati udara ke segala arah. Gelombang radio merupakan salah satu bentuk dari radiasi elektromagnetik dan terbentuk ketika objek bermuatan listrik dipercepat dengan frekuensi yang terdapat pada frekuensi radio dalam spektrum elektromagnetik. Daya jangkau gelombang ini antara 10 hertz hingga mencapai 300 Gigahertz.

2.9 Modul XBee

Modul XBee mendukung 2 jenis protokol antarmuka serial yaitu mode *transparent* dan mode API (*Application Programming Interface*). Pada mode transparent, terdapat 2 jenis kondisi yaitu *command* dan data. Pada mode data, modul XBee dapat dianalogikan sebagai pengganti kabel dalam komunikasi serial. Mode *command* digunakan untuk melakukan konfigurasi pada modul XBee ZB melalui AT Command. Pada API mode, seluruh data yang keluar atau masuk dari/ke XBee ZB terdiri dari paket data yang berisi informasi atau operasi yang sedang terjadi di dalam modul. API mode menyediakan fasilitas untuk memudahkan host untuk mengkonfigurasi modul tanpa harus meninggalkan data mode ataupun memudahkan proses routing data. Pada API mode tidak ada perbedaan langsung antara command mode & data mode seperti pada transparent mode. Semua data maupun operasi baik data maupun command mode hanya dibedakan oleh frame API Identifier. API mode sesuai jika digunakan untuk beberapa aplikasi yang membutuhkan:

1. Pengiriman data ke banyak tujuan (*multiple destinations*).
2. Informasi status pengiriman data (sukses/gagal).
3. Identifikasi alamat pengirim data yang diterima oleh modul

XBee merupakan sebuah modul yang didalamnya terdiri dari XBee *Receiver* dan XBee *transmitter* dengan sistem antarmuka serial UART (*Universal Asynchronous Receiver Transmitter*). Gambar 2.23 merupakan cara kerja komunikasi serial XBee.



Gambar 2.23 Komunikasi serial XBee
Sumber: [QIS-12]

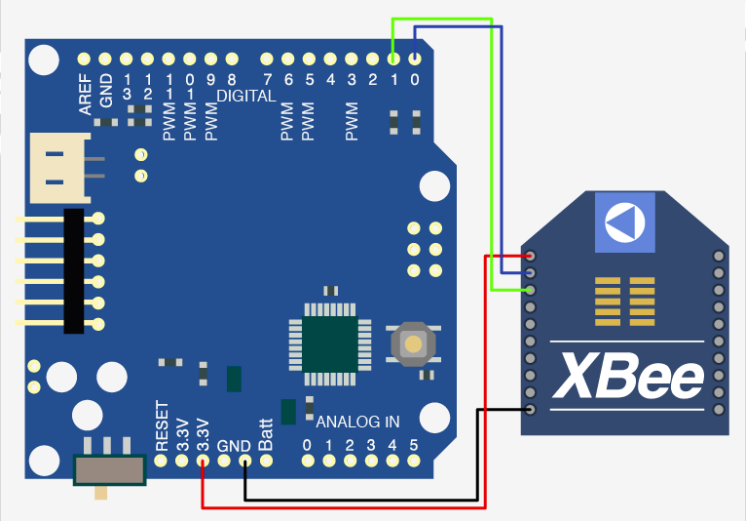
Pengkomunikasian Arduino dengan XBee dengan cara menyamakan kabel jumper untuk menghubungkan pin-pin. Pin yang dihubungkan antara lain:

Tabel 2.1 Tabel Koneksi Pin Arduino-XBee

XBee	Arduino
VCC atau 3.3 V	3V3
TX atau DOUT	RX atau 0
RX atau DIN	TX atau 1
GND	GND

Sumber:[QIS-12]

Penyusunan pin dapat dilihat pada gambar 2.24



Gambar 2.24 Pemasangan Pin XBee dengan Arduino Uno
Sumber:[Perancangan]

Selain dengan menggunakan kabel jumper, arduino dapat berkomunikasi dengan XBee melalui shield yang disebut XBee shield. XBee shield berupa papan yang memiliki kaki-kaki yang sama dengan arduino yang fungsinya sebagai jalur penghubung XBee dengan Arduino. Shield ini dihubungkan langsung dengan papan Arduino atau USB, dan melengkapinya dengan kapabilitas komunikasi nirkabel menggunakan modul XBee.