IMPLEMENTASI METODE *RELIABLE DATA TRANSFER* DAN *CYCLIC REDUDANCY CHECK* (CRC) PADA ARDUINO DENGAN KOMUNIKASI RF

SKRIPSI

KEMINATAN TEKNIK KOMPUTER

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Januar Rizky Pratama
115060900111027

PROGRAM STUDI TEKNIK INFORMATIKA
PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2016

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

PENGESAHAN

IMPLEMENTASI METODE *RELIABLE DATA TRANSFER* DAN *CYCLIC REDUDANCY CHECK* (CRC) PADA ARDUINO DENGAN KOMUNIKASI RF

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Januar Rizky Pratama
115060900111027

Skripsi ini telah diuji dan dinyatakan lulus pada
27 Januari 2016
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Sabriansyah Rizqika Akbar, S.T., M.Eng.	Aswin Suharsono, S.T., M.T.
NIP: 19820809 201212 1 004	NIK: 84091906110251

Mengetahui
Ketua Program Studi Teknik Informatika

Drs. Marji, M.T.
NIP: 19670801 199203 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 27 Januari 2016

Januar Rizky Pratama

115060900111027

KATA PENGANTAR

Puji syukur kehadirat Allah SWT yang telah melimpahkan rahmat, taufik dan hidayah-Nya sehingga laporan skripsi yang berjudul **"IMPLEMENTASI METODE RELIABLE DATA TRANSFER DAN CYCLIC REDUDANCY CHECK (CRC) PADA ARDUINO DENGAN KOMUNIKASI RF"** ini dapat terselesaikan. Skripsi ini untuk memenuhi salah satu syarat menyelesaikan studi serta dalam rangka memperoleh gelar Sarjana Pendidikan Strata Satu pada Program Studi Sistem Komputer Fakultas Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya. Penghargaan dan terima kasih yang setulus-tulusnya kepada Ayahanda tercinta Harry Suhada, S.H. dan Ibunda yang kusayangi Yuni Hardini, A.Md.Gizi serta adik tersayang Isyaqi Dwi Hardiono semoga Allah SWT melimpahkan Rahmat, Kesehatan, Karunia dan keberkahan di dunia dan di akhirat atas budi baik yang telah diberikan kepada penulis.

Penulis menyadari bahwa skripsi ini tidak akan berhasil tanpa bantuan dari beberapa pihak. Oleh karena itu, penulis ingin menyampaikan rasa hormat dan terima kasih kepada:

1. Bapak Prof. Dr. Ir. Mohammad Bisri, MS selaku Rektor Universitas Brawijaya
2. Bapak Ir Sutrisno, M.T. selaku Dekan Fakultas Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya.
3. Bapak Adharul Muttaqin, S.T., M.T. selaku Kepala Program Studi Sistem Komputer Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya.
4. Bapak Barlian Henryranu, S.T., M.T. selaku Kepala Laboratorium Sistem Komputer dan Robotika Program Teknologi Informasi dan Ilmu Komputer.
5. Bapak Sabriansyah Rizqika Akbar, S.T., M.Eng. selaku pembimbing I yang telah banyak membantu dalam pengerjaan skripsi ini.
6. Bapak Aswin Suharsono, S.T., M.T. selaku pembimbing II yang telah banyak membantu dalam pengerjaan skripsi ini.
7. Mas Didit yang selalu membantu penulis dalam hal administrasi Laboratorium Sistem Komputer dan Robotika.
8. Segenap dosen Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya atas seluruh ilmu pengetahuan dan perhatian yang diberikan.
9. Seluruh civitas akademika Program Studi Sistem Komputer Fakultas Ilmu Komputer Universitas Brawijaya yang telah banyak memberi bantuan dan dukungan selama penulis menempuh studi di Program Studi Sistem Komputer Universitas Brawijaya dan selama penyelesaian skripsi ini.
10. Teman-teman dalam satu bimbingan M. Kholil Gibran dan M. Abdul Mujib yang selalu berbagi semangat dan ilmu untu menyelesaikan skripsi.

11. Teman-teman terdekat Rizky Romadhoni P, Ryan Rizki A, Rizki Dwi AP, Bagus Priyo P, M. Dhiya' Ainul Labib, M. Agil Dwi S, Saputra Yudha W, M. Nur Arifin, M. Emirza Alam F, Rizky Budi S, Nanda Ainal Y, Alfaviega SP, Syukri Akbar, Yuni Prasetyaning M, Anake Deborah, Dini Melsye NF, Rezky Julyarti A, Donny W, Loki Sudiarta M dan semua teman Sistem Komputer 2011 yang selalu menjadi penyemangat selama menjalani perkuliahan serta penyelesaian skripsi.

Malang, 27 Januari 2016

Januar Rizky Pratama

115060900111027

repository.ub.ac.id

ABSTRAK

Komunikasi *wireless* adalah komunikasi yang memungkinkan perangkat-perangkat yang terdapat di dalamnya berkomunikasi secara wireless atau tanpa kabel. Terdapat suatu masalah dalam proses pengiriman data dengan *wireless*. Resiko kehilang sebagian atau seluruh data sangat besar dalam pengiriman menggunakan *wireless*. Kehilangan data dalam jaringan nirkabel umum dan memiliki pola khusus karena *noise*, *collision*, *unreliable link*, dan kerusakan yang tidak terduga, yang sangat mengurangi akurasi rekonstruksi.

Untuk mengatasi masalah tersebut, akan diterapkan metode *Reliable Data Transfer* untuk memastikan pengiriman data menggunakan *wireless*. Pengiriman data menggunakan modul NRF24L01. Pada pengiriman data disertakan pula CRC (*Cyclic Redudancy Check*) untuk memastikan integritas data dan mengecek kesalahan pada suatu data yang akan ditransmisikan atau disimpan. Pengiriman sendiri akan diproses dalam mikrokontroller arduino. Hasil dari penelitian ini adalah sistem yang bisa menerapkan metode yang ada pada *Reliable Data Transfer* seperti *operation with no loss*, *lost packet* dan *lost ACK.*

Kata Kunci: *Nirkabel, Reliable Data Transfer, CRC, NRF24L01, Arduino*

UNIVERSITAS BRAWIJAYA

repository.ub.ac.id

ABSTRACT

Wireless communication is communication that allows the devices contained to communicate wirelessly. There is a problem in the process of sending data by wireless. The risk of a loss of some or all of the data is huge in using wireless delivery. Loss of data in the public wireless network and have a special pattern for noise, collision, unreliable links, and unforeseen damage, which greatly reduces the accuracy of the reconstruction.

To overcome these problems, will be applied the method Reliable Data Transfer to ensure the use of wireless data transmission. Sending data using NRF24L01 module. In the data delivery also included a CRC (Cyclic Redundancy Check) to ensure data integrity and error checking on the data to be transmitted or stored. The delivery will be processed in the microcontroller arduino. Results of this research is that the system can apply the methods that exist on Reliable Data Transfer like operation with no loss, lost packet and lost ACK.

Keywords: Wireless, Reliable Data Transfer, CRC, NRF24L01, Arduino

UNIVERSITAS BRAWIJAYA

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

Bab ini menjelaskan latar belakang dari penelitian, rumusan masalah yang harus diselesaikan, batasan masalah yang ada pada penelitian, tujuan dari penelitian, manfaat dari penelitian yang dilakukan, dan sistematika penulisan dari penelitian.

1.1 Latar Belakang

Komunikasi *wireless* adalah komunikasi yang memungkinkan perangkat-perangkat yang terdapat di dalamnya berkomunikasi secara wireless atau tanpa kabel. Penelitian tentang komunikasi *wireless* sudah dilakukan sejak tahun 1960-an. Contoh dari komunikasi *wireless* adalah transfer informasi berupa apapun, secara jarak jauh tanpa menggunakan kabel.

Dalam suatu pengiriman data secara nirkabel dalam suatu sistem memiliki resiko kehilang sebagian atau seluruh data sangat besar. kehilangan data dalam jaringan sensor nirkabel umum dan memiliki pola khusus karena *noise*, *collision*, *unreliable link*, dan kerusakan yang tidak terduga, yang sangat mengurangi akurasi rekonstruksi (Kong, 2013).

Berdasarkan latar belakang tersebut, penulis melakukan penelitian untuk meminimalisir resiko kehilangan data pada pengiriman data secara nirkabel dengan menerapkan metode *Reliable Data Transfer* pada komunikasi *wireless* menggunakan modul NRF24L01. *Reliable Data Transfer* (RDT) dikenal juga sebagai protokol stop-and-wait. Setelah mengirim setiap paket, pengirim berhenti dan menunggu umpan balik dari penerima menunjukkan bahwa paket telah diterima. Dan juga dengan penambahan CRC untuk file yang akan dikirim. *Reliable Data Transfer* bertujuan untuk memastikan keberhasilan pengiriman data.

Berdasarkan latar belakang tersebut, penulis ingin menerapkan metode reliable data transfer pada komunikasi antar NRF24L01.

1.2 Rumusan Masalah

Berdasarkan pemaparan pada latar belakang, maka dapat di rumuskan permasalahan pada skripsi yaitu seberti berikut:

1. Bagaimana Implementasi Metode *Reliable Data Transfer* antar perangkat dengan media komunikasi NRF24L01.
2. Bagaimana melakukan pengecekan keutuhan data pada sisi penerima dengan metode Cyclic Redundancy Check
3. Bagaimana melakukan pengujian Reliable Data Transfer pada NRF24L01

1.3 Tujuan

Tujuan dari penelitian ini adalah untuk mencoba menerapkan konsep *Reliable Data Transfer* pada pengiriman data menggunakan NRF24L01 berbasis arduino.

Tujuan penggunaan *Reliable Data Transfer* agar memastikan data yang dikirimkan bisa terkirim dengan sempurna. Tujuan lainnya adalah pengimplementasian *checksum* pada data yang dikirim agar di sisi penerima mengetahui apakah data yang dikirimkan benar atau tidak.

1.4 Manfaat

Adapun manfaat dari sistem penentu status kebakaran adalah sebagai berikut:

1. Dapat diperoleh suatu metode pengiriman data yang Reliable.
2. Data tidak mungkin hilang atau terduplikasi.
3. Data yang dikirim bisa dipastikan kebenarannya menggunakan CRC.

1.5 Batasan Masalah

Agar permasalahan yang telah dirumuskan dapat lebih fokus, maka skripsi ini dibatasi dalam lingkup beberapa hal yaitu sebagai berikut:

1. Modul pengiriman data secara nirkabel menggunakan NRF24L01.
2. Metode *checksum* menggunakan CRC8
3. Dalam perancangan sistem ini menggunakan Board Arduino
4. Data yang dikirim berupa text
5. Pengujian *Reliable Data Transfer* hanya berfokus pada keberhasilan pengiriman

1.6 Sistematika Penulisan

Sistematika pembahasan ditunjukan untuk memberikan gambaran dan uraian dari penulisan skripsi ini secara garis besar yang meliputi beberapa bab sebagai berikut:

BAB I : PENDAHULUAN

Pada bab ini berisi tentang penguraian latar belakang masalah yang dikaji, rumusan masalah, batasan masalah pada penelitian, tujuan penelitian, manfaat penelitian, serta sistematika penulisan dari perancangan Implementasi metode Reliable Data Transfer dan CRC pada Arduino dengan Komunikasi RF.

BAB II : KAJIAN PUSTAKA DAN DASAR TEORI

Pada bab ini berisi referensi dan dasar teori yang mendasari pembuatan rancangan Implementasi metode Reliable Data Transfer dan CRC pada Arduino dengan Komunikasi RF**.**

BAB III : METODOLOGI PENELITIAN

Pada bab ini menguraikan dan membahas langkah kerja yang dilakukan dalam penulisan skripsi yang terdiri dari studi literatur, implementasi, perancangan yang

meliputi desain modul sistem secara keseluruhan, skematik rangkaian, blok diagram dan diagram alir sistem, analisis kebutuhan dan pengujian.

BAB IV : PERANCANGAN DAN IMPLEMENTASI

Pada bab ini berisi penjelasan tentang rancangan dari Implementasi metode Reliable Data Transfer dan CRC pada Arduino dengan komunikasi RF yang meliputi implementasi desain skematik rancangan tiap komponen pembentuk sistem baik hardware maupun software.

BAB V : PENGUJIAN DAN ANALISIS

Pada bab ini berisi pengujian sistem secara keseluruhan, yang meliputi pengecekan cara kerja pengiriman data dengan *Reliable Data Transfer*. Kemudian hasil dari pengujian tersebut dicatat dan dianalisis.

BAB VI : KESIMPULAN DAN SARAN

Pada bab ini berisi kesimpulan atas penelitian yang telah dilakukan, serta memberikan saran untuk pengembangan sistem lebih lanjut.

BAB 2 KAJIAN PUSTAKA DAN DASAR TEORI

Bab ini menjelaskan tinjauan pustaka yang meliputi kajian pustaka dan dasar teori yang diperlukan. Kajian pustaka membahas penelitian-penelitian yang telah dilakukan sebelumnya. Dasar teori membahas teori yang diperlukan untuk menyusun penelitian yang dilakukan.

2.1 Kajian Pustaka

Pada jurnal A high-speed reliable data transfer method based on FPGA hardware acknowledgement (Kaiyun Tian, 2014) dijelaskan bahwa dalam jaringan transfer data dengan kabel, Transmission Control Protocol (TCP) secara luas digunakan untuk menjamin kehandalan transmisi. Tetapi protokol membutuhkan banyak sumber daya CPU dan memiliki efisiensi transmisi rendah. TCP menggunakan retransmission saat time out dan data acknowledge untuk mencapai kemampuan transmisi yang handal. Penerima data harus menyatakan penerimaan data frame dengan acknowledgement (ACK) frame untuk menginformasikan transmitter data telah diterima dengan benar. Data pemancar hanya dapat mengirimkan frame data baru setelah menerima frame ACK data saat ini, pengiriman ulang akan dimulai ketika time out. Pembacaan dan penanganan ACK frame biasanya dilakukan dengan CPU, yang membutuhkan sumber daya CPU dan menambah waktu delay, karena itu meningkatkan rata bingkai waktu tunggu ACK dan menurunkan data rate. Makalah ini mengimplementasikan metode transfer data yang dapat diandalkan kecepatan tinggi berdasarkan FPGA hardware acknowledgement. Metode ini menggunakan acknowledgement (ACK) frame dan transmisi untuk memastikan transmisi yang handal dan berurutan seperti yang dilakukan TCP. (Tian, 2014)

Dalam jurnal yang berjudul HI : An Hybrid Adaptive Interleaved communication protocol for reliable data transfer in WSNs with mobile sinks (Antasari, 2009) membahas tentang protokol komunikasi yang efisien untuk pengumpulan data yang dapat diandalkan dalam jaringan sensor. Proses ini cukup penting karena interaksi antara *mobile sinks* mengumpulkan informasi dan sensor node menyediakan data umumnya pendek, tidak terduga dan dipengaruhi oleh *packet loss*. Oleh karena itu memerlukan transfer data yang sangat cepat dan dapat diandalkan pertukaran data dengan sinkronisasi minimal. (Antasari, 2009)

2.2 Dasar Teori

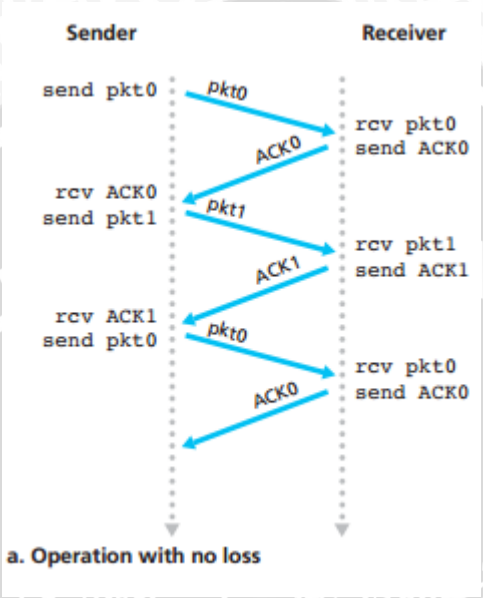
Pada bab ini akan membahas tentang dasar teori yang menunjang penelitian, yaitu mengenai Reliable Data Transfer, Checksum, Arduino Nano dan NRF24L01.

2.2.1 Reliable Data Transfer

Reliable Data Transfer (RDT) dikenal juga sebagai protokol stop-and-wait. Setelah mengirim setiap paket, pengirim berhenti dan menunggu umpan balik dari

penerima menunjukkan bahwa paket telah diterima. Ada bermacam-macam jenis RDT, contohnya RDT 1.0, 2.0, dan 3,0

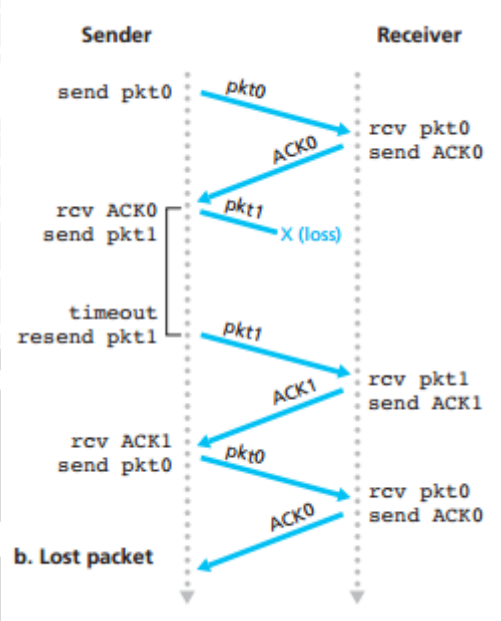
Pada penelitian ini akan menggunakan *Reliable Data Transfer (RDT) 3.0. Reliable Data Transfer (RDT)* merupakan sebuah metode pengiriman data pada protokol TCP/IP. Proses-proses Reliable Data Transfer dapat dilihat pada gambar 2.1, 2.2 dan 2.3.



Gambar 2. Operation with no loss

Sumber: (Kurose, 2012)

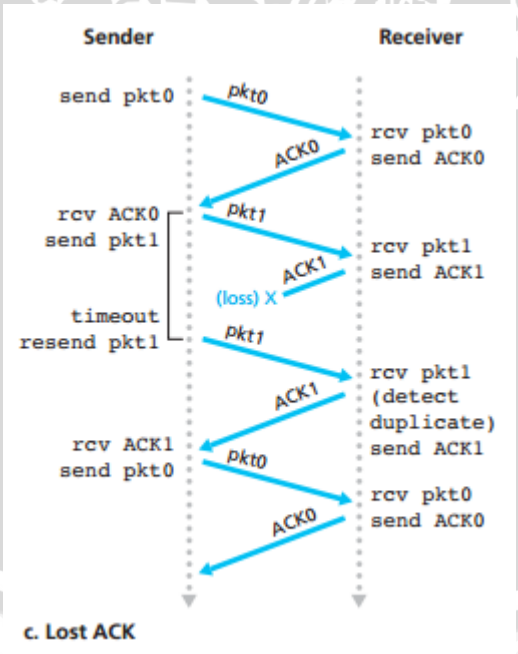
Pada keadaan ini, pengiriman paket dan ACK tidak mengalami kehilangan sehingga data terkirim dengan baik.



Gambar 2. Lost Packet

Sumber: (Kurose, 2012)

Pada kemungkinan ini, pengiriman paket mengalami loss. Dan karena paket loss maka pada sender akan mengalami timeout dan sender akan megirim paket lagi.



Gambar 2. Lost ACK

Sumber: (Kurose, 2012)

Pada kemungkinan ini, *receiver* yang akan mengirim balik ACK mengalami loss sehingga menyebabkan timeout pada *sender* dan otomatis *sender* akan mengirim paket lagi. Namun karena di sisi *receiver* terdapat paket yang sama dengan kata lain terjadi duplikasi file, maka salah satu file akan dihapus.

2.2.2 CRC (Cyclic Redundancy Check)

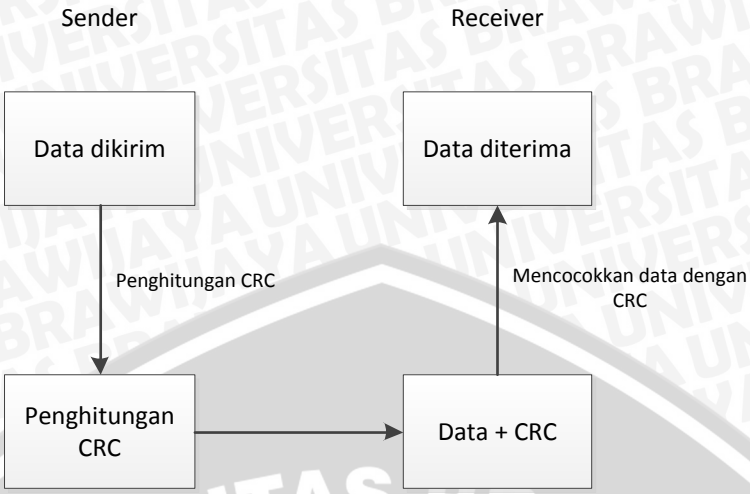
CRC (Cyclic Redundancy Check) adalah algoritma untuk memastikan integritas data dan mengecek kesalahan pada suatu data yang akan ditransmisikan atau disimpan.

Data yang hendak ditransmisikan atau disimpan ke sebuah media penyimpanan rentan sekali mengalami kesalahan, seperti halnya noise yang terjadi selama proses transmisi atau memang ada kerusakan perangkat keras. Untuk memastikan integritas data yang hendak ditransmisikan atau disimpan, CRC dapat digunakan. CRC bekerja secara sederhana, yakni dengan menggunakan perhitungan matematika terhadap sebuah bilangan yang disebut sebagai Checksum, yang dibuat berdasarkan total bit yang hendak ditransmisikan atau yang hendak disimpan.

Dalam transmisi jaringan, khususnya dalam jaringan berbasis teknologi Ethernet, checksum akan dihitung terhadap setiap frame yang hendak ditransmisikan dan ditambahkan ke dalam frame tersebut sebagai informasi dalam header atau trailer. Penerima frame tersebut akan menghitung kembali apakah frame yang ia terima benar-benar tanpa kerusakan, dengan membandingkan nilai frame yang dihitung dengan nilai frame yang terdapat dalam header frame. Jika dua nilai tersebut berbeda, maka frame tersebut telah berubah dan harus dikirimkan ulang. (hackersdelight, 2009)

CRC didesain sedemikian rupa untuk memastikan integritas data terhadap degradasi yang bersifat acak dikarenakan noise atau sumber lainnya (kerusakan media dan lain-lain). CRC tidak menjamin integritas data dari ancaman modifikasi terhadap perlakukan yang mencurigakan oleh para hacker, karena memang para penyerang dapat menghitung ulang checksum dan mengganti nilai checksum yang lama dengan yang baru untuk membodohi penerima.

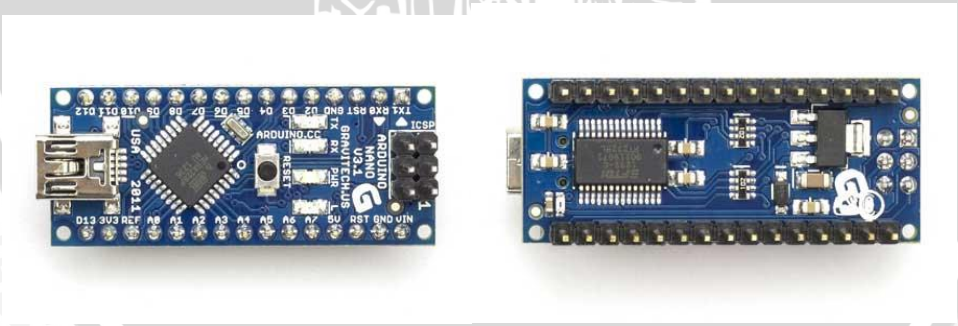
Kode pendeteksian kesalahan yang paling umum serta paling hebat adalah Cyclic Redundancy Check (CRC) yang dapat digambarkan sebagai berikut, dengan adanya blok bit k-bit, atau pesan, transmitter mengirimkan suatu deretan n-bit, disebut sebagai Frame Check Sequence (FCS), sehingga frame yang dihasilkan, terdiri dari k+n bit, dapat dibagi dengan jelas oleh beberapa nomor yang sebelumnya sudah ditetapkan. Kemudian receiver membagi frame yang datang dengan nomor tersebut dan, bila tidak ada sisa, maka diasumsikan tidak terdapat kesalahan. Rumus dari CRC 8 bit adalah x8+x2+x+1 yang dalam bit berarti 100000111. (Humphrys, 1987)



Gambar 2. Proses CRC

2.2.3 Arduino Nano

Arduino Nano adalah salah satu papan pengembangan mikrokontroler yang berukuran kecil, lengkap dan mendukung penggunaan breadboard. Arduino Nano diciptakan dengan basis mikrokontroler ATmega328 (untuk Arduino Nano versi 3.x) atau ATmega 168 (untuk Arduino versi 2.x). Arduino Nano kurang lebih memiliki fungsi yang sama dengan Arduino Duemilanove, tetapi dalam paket yang berbeda. Arduino Nano tidak menyertakan colokan DC berjenis Barrel Jack, dan dihubungkan ke komputer menggunakan port USB Mini-B. Arduino Nano dirancang dan diproduksi oleh perusahaan Gravitech. (Arduino, 2015)



Gambar 2. Arduino Nano

Sumber (Arduino, 2015)

2.2.3.1 Spesifikasi Arduino Nano

Spesifikasi dari Arduino Nano akan dijelaskan pada tabel 2.1

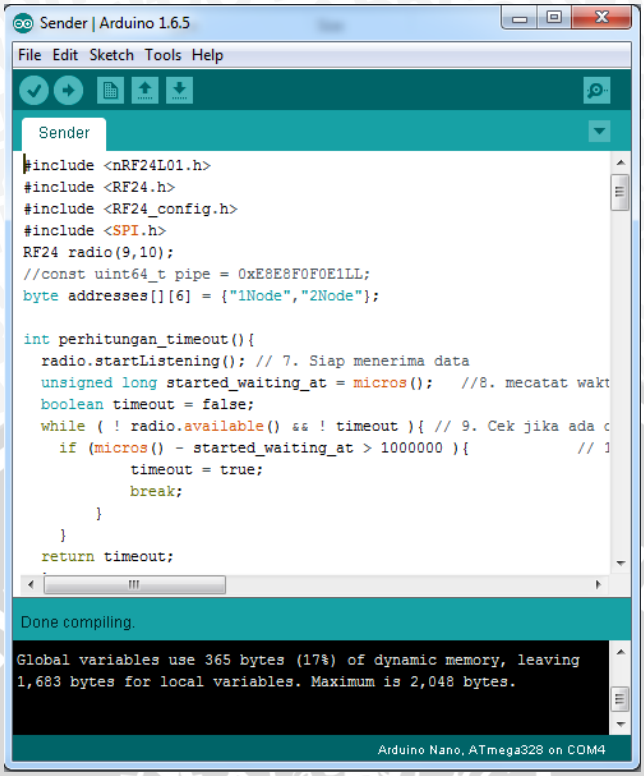
Tabel 2. Spesifikasi Arduino Nano

Microcontroller	Atmel Atmega 168 atau Atmega328
Operating Voltage	5V
Input Voltage	7 - 12 V
Digital I/O Pins	14 (*6 pin digunakan sebagai output PWM*)
Analog Input Pins	8
DC Current per I/O Pin	40 mA
Flash Memory	16 KB (*Atmega168*) atau 32 KB (Atmega328) 2KB digunakan oleh Bootloader
SRAM	1 KB (Atmega168) atau 2 KB (Atmega328)
EEPROM	512 bytes (Atmega168) atau 1 KB (Atmega328)
Clock Speed	16 MHz
Ukuran	1.85cm x 4.3cm

Sumber: (Arduino, 2015)

2.2.3.2 Perangkat lunak (Arduino IDE)

Software Arduino ini bisa dijalankan di komputer dengan berbagai macam platform karena berbasis Java. Source program yang dibuat untuk aplikasi mikrokontroler adalah bahasa C/C++ dan dapat digabungkan dengan assembly. Penulis menggunakan arduino berbasis mikrokontroler AVR dilingkungan jenis ATMEGA yaitu ATMEGA 8, 168, 328 dan 2650. Tampilan Arduino IDE ditampilkan pada Gambar 2.6



Gambar 2. Software Arduino IDE

Sumber: (Arduino, 2015)

2.2.4 NRF24L01

NRF24L01 adalah sebuah modul komunikasi jarak jauh dengan single-chip RF-transceiver yang ditujukan untuk aplikasi pada gelombang 2.4 GHz ISM band. Modul NRF24L01 menggunakan antar muka SPI (Serial Pheripheral *Interface*) untuk komunikasi dengan tegangan kerja 5V DC.

Gambar 2. Modul NRF24L01

Sumber: (Charles's Blog, 2013)

NRF24L01 memiliki spesifikasi sebagai berikut:

- Worldwide 2.4GHz ISM band operation
- 250kbps, 1Mbps and 2Mbps on air data rates
- Ultra low power operation
- 11.3Ma TX at 0dBm output power
- 13.5Ma RX at 2Mbps air data rate
- 900Na in power down
- 26Ma in standby-I
- On chip voltage regulator
- 1.9 to 3.6V supply range
- Enhanced ShockBurst™
- Automatic packet handling
- Auto packet transaction handling
- 6 data pipe MultiCeiver™
- Drop-in compatibility with Nrf24L01
- On-air compatible in 250kbps and 1Mbps with Nrf2401A, Nrf2402, Nrf24E1 and Nrf24E2
- Low cost BOM
- ±60ppm 16MHz crystal
- 5V tolerant inputs
- Compact 20-pin 4x4mm QFN package

NRF24L01 bisa diimplementasikan pada:

- *Wireless* PC Peripherals
- Mouse, key*board*s and remotes
- 3-in-1 desktop bundles
- Advanced Media center remote controls
- VoIP headsets
- Game controllers
- Sports watches and sensors
- RF remote controls for consumer electronics
- Home and commercial automation
- Ultra low power sensor networks
- Active RFID
- Asset tracking sistems
- Toys

NRF24L01 memiliki beberapa fitur, antara lain:

- Radio:

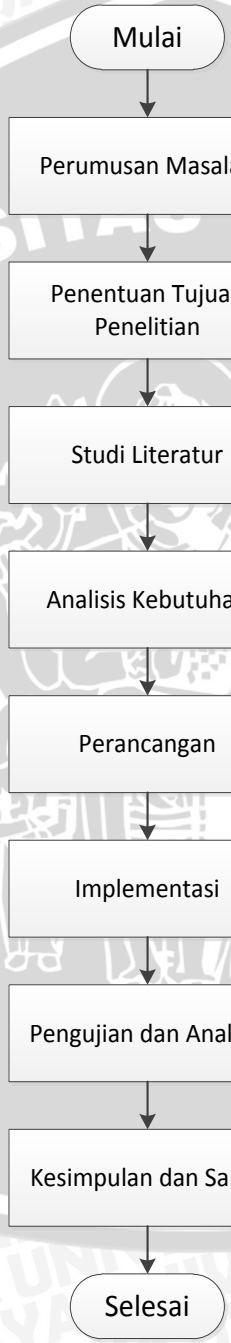
Worldwide 2.4GHz ISM band operation
126 RF channels
Common RX and TX *interface*
GFSK modulation
250kbps, 1 and 2Mbps air data rate
1MHz non-overlapping channel spacing at 1Mbps

2MHz non-overlapping channel spacing at 2Mbps
• Transmitter:
Programmable output power: 0, -6, -12 or -18dBm
11.3Ma at 0dBm output power
• Receiver:
Fast AGC for improved dynamic range
Integrated channel filters
13.5Ma at 2Mbps
-82dBm sensitivity at 2Mbps
-85dBm sensitivity at 1Mbps
-94dBm sensitivity at 250kbps
• RF Synthesizer:
Fully integrated synthesizer
No external loop filer, VCO varactor diode or resonator
Accepts low cost ±60ppm 16MHz crystal
• Enhanced ShockBurst™:
1 to 32 bytes dynamic payload length
Automatic packet handling
Auto packet transaction handling
6 data pipe MultiCeiver™ for 1:6 star networks
• Power Management:
Integrated voltage regulator
1.9 to 3.6V supply range
Idle modes with fast start-up times for advanced power management
26Ma Standby-I mode, 900Na power down mode
Max 1.5ms start-up from power down mode
Max 130us start-up from standby-I mode
• Host *Interface*:
4-pin hardware SPI
Max 10Mbps
separate 32 bytes TX and RX FIFOs
5V tolerant inputs]
• Compact 20-pin 4x4mm QFN package

BAB 3 METODOLOGI PENELITIAN

3.1 Diagram Alir

Pada bab ini akan menjelaskan langkah-langkah yang dilakukan dalam penyusunan skripsi. Metode penelitian yang digunakan pada skripsi ini ditunjukan pada diagram alir pada Gambar 3.1.



Gambar 3. Alur pelaksanaan

3.2 Perumusan masalah

Perumusan masalah merupakan tahap menentukan masalah yang dibahas dalam penelitian. Masalah yang terbentuk meliputi:

1. Bagaimana cara pengiriman data dari NRF24L01 dengan menerapkan *Reliable Data Transfer*?
2. Bagaimana cara menerapkan CRC pada data yang akan dikirim?

3.3 Penentuan Tujuan Penelitian

Dalam penelitian ini ditetapkan beberapa tujuan untuk memfokuskan permasalahan dengan hasil akhir berupa laporan. Adapun tujuan dari penelitian ini adalah yang sudah dijelaskan pada bab 1, yaitu untuk mengimplementasikan metode *Reliable Data Transfer* pada pengiriman data menggunakan NRF24L01 dan penerapan CRC pada data yang dikirim.

3.4 Studi literatur

Studi literatur digunakan sebagai teori penguat dan landasan dasar dalam penelitian. Teori pendukung tersebut diperoleh dari buku, jurnal, paper dan sumber lain yang dapat dipertanggung jawabkan. Literatur yang digunakan meliputi:

1. *Reliable Data Transfer*
   a. *Knowledge Base* (Basis Pengetahuan)
   b. Jenis-jenis *Reliable Data Transfer*
2. *Checksum CRC*
   a. Pengenalan *Checksum* CRC
   b. Cara kerja *Checksum* CRC
3. Arduino
   a. Pengenalan Arduino
   b. Arduino IDE
   c. Pemrograman Arduino
4. NRF24L01
   a. Pengenalan NRF24L01

3.5 Analisis Kebutuhan

Analisis kebutuhan bertujuan untuk mengetahui semua kebutuhan yang diperlukan ketika perancangan sampai pembuatan sistem. Analisis dari kebutuhan dilakukan dengan mengidentifikasi jenis dan fungsi kebutuhan.

Secara keseluruhan, kebutuhan yang digunakan dalam pembuatan sistem ini dikelompokkan menjadi 2 (dua) bagian yaitu kebutuhan Perangkat Keras (*hardware*) dan kebutuhan Perangkat Lunak (*software*).

3.5.1 Perangkat Keras (*Hardware*)

Perangkat keras pada sistem ini digunakan sebagai pengolah data dari *input* sehingga menghasilkan *output*. Beberapa perangkat keras alat yang dibutuhkan oleh sistem adalah sebagai berikut :

1. 1 buah PC untuk memonitor hasil pengiriman data
2. 2 buah Arduino Nano sebagai board mikrokontroler
3. NRF24L01 sebagai modul pengiriman
4. Kabel Jumper sebagai penghubung antara Arduino Nano dan NRF24L01
5. Kabel Penghubung Arduino ke PC

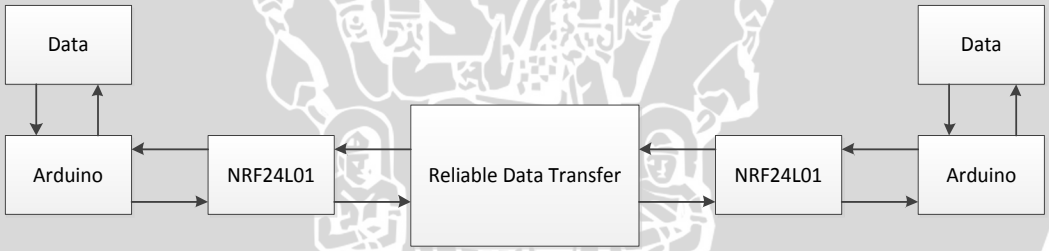
3.5.2 Perangkat Lunak (*Software*)

Perangkat lunak yang dibutuhkan untuk mendukung kerja pada sistem dan penulisan skripsi adalah sebagai berikut :

1. Windows 7 Ultimate *System type* 32-bit, Sebagai sistem operasi yang digunakan untuk menjalan kan perangkat lunak.
2. Arduino 1.6.5, sebuah perangkat lunak yang digunakan untuk membuat kode program yang disimpan pada mikrokontroler.
3. Microsoft Visio, digunakan untuk membuat diagram blok dan flowchart.

3.6 Perancangan Sistem

Perancangan sistem dapat dilihat pada gambar berikut:



Gambar 3. Diagram Blok Sistem

Pada gambar terdapat 2 NRF24L01 yang masing-masing tersambung dengan Arduino Nano sebagai mikrokontroler untuk pengolahan data. Dan komunikasi antara 2 NRF24L01 ini menggunakan *Reliable Data Transfer* yang berarti terjadi komunikasi 2 arah yang memungkinkan 2 NRF24L01 ini saling menerima data.

3.7 Implementasi

Implementasi sistem dilakukan dengan mengacu pada perancangan sistem seperti yang terdapat pada gambar 3.2 di atas. Pada implementasi dicantumkan gambar sistem yang sudah dibuat serta potongan-potongan bahasa pemrograman yang digunakan.

3.8 Pengujian dan analisis

Pengujian sistem pada penelitian ini dilakukan agar dapat menunjukkan bahwa sistem telah mampu bekerja sesuai dengan spesifikasi yang sudah dirancang. Pengujian yang akan dilakukan meliputi:

1. Pengujian konvert data yang dikirim menjadi bit oleh proses CRC.
2. Pengujian 3 kemungkinan yang ada pada *Reliable Data Transfer,* yaitu Operation with No Loss, Lost Packet dan Lost ACK.

3.9 Kesimpulan dan Saran

Kesimpulan didapatkan setelah melakukan perancangan, implementasi, pengujian dan analisis kepada sistem. Kesimpulan diambil berdasarkan hasil pengujian dan analisis sistem yang telah dibuat.

BAB 4 PERANCANGAN DAN IMPLEMENTASI

Pada bab ini akan dijelaskan perancangan dan implementasi dari sistem yang dibuat. Bab ini akan memudahkan proses pengujian dan analisis karena didasarkan pada perancangan dan implementasi.

4.1 Perancangan

Tahap perancangan ini memfokuskan pembahasan tentang perancangan atau desain dari sistem yang nantinya dapat menjadi sebuah sistem dalam satu kesatuan. Dalam perancangan sistem terdapat beberapa tahap yang harus dilakukan diantaranya:

4.1.1 Analisis Kebutuhan

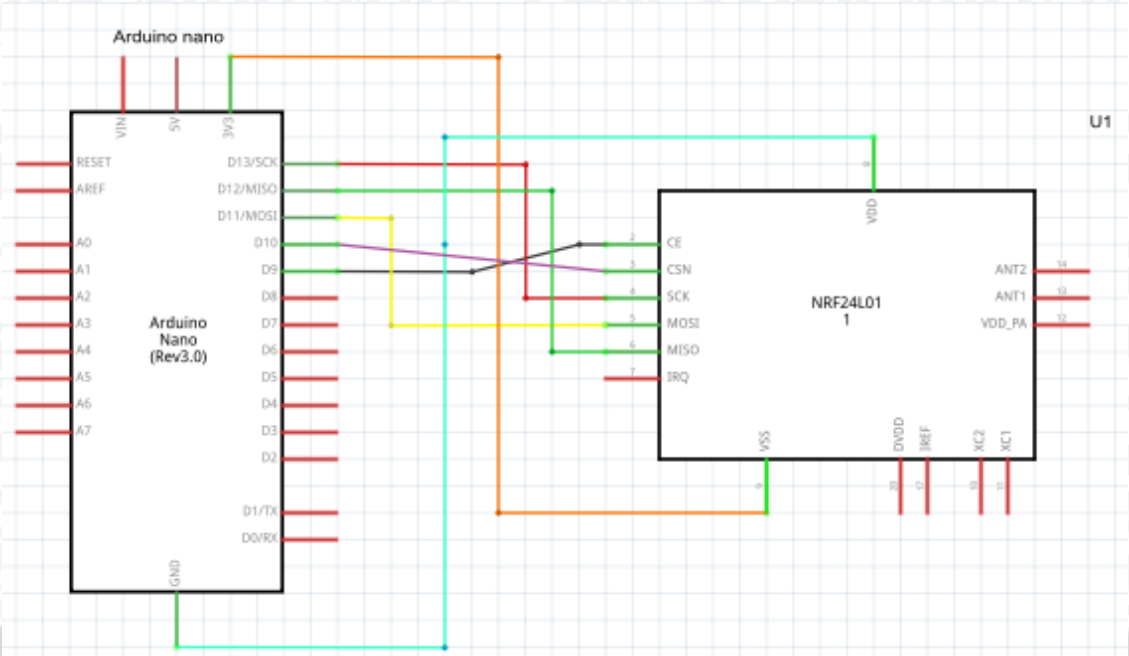
Analisis kebutuhan dalam perancangan bertujuan untuk menggambarkan kebutuhan-kebutuhan dari sistem. Perangkat keras dan perangkat lunak akan dianalisis sesuai dengan perannya dalam membangun sistem sehingga akan mempermudah dalam mendesain dan mengimplementasikan sistem.

4.1.2 Pendefinisian Sistem

Sistem yang akan dibuat adalah sistem untuk implementasi pengiriman data menggunakan *Reliable Data Transfer* pada NRF24L01 di mana data dikirimkan dari NRF24L01 yang satu ke NRF24L01 lain. Dalam proses pengiriman ini, di dalamnya terdapat metode *Reliable Data Transfer* yang bertujuan untuk memastikan data yang dikirim valid.

4.1.3 Rangkaian Sistem

Pada rangkaian sistem ini akan ditampilkan rangkaian Arduino Nano dengan NRF24L01. Rangkaian sistem secara keseluruhan dapat dilihat pada Gambar 4.1



Gambar 4. Rangkaian Arduino Nano dengan NRF24L01

Gambar di atas menggambarkan rangkaian yang akan dibuat dengan menghubungkan pin antara Arduino Nano dengan NRF24L01.

4.1.4 Perancangan Perangkat Keras (*Hardware*)

Perangkat keras pada sistem ini menggunakan dua Arduino Nano dan dua modul NRF24L01. Modul NRF24L01 digunakan untuk mengirim dan menerima data lalu diolah di Arduino Nano. Untuk penghubung antara Arduino Nano dan NRF24L01 digunakan kabel jumper seperti pada gambar 4.2.

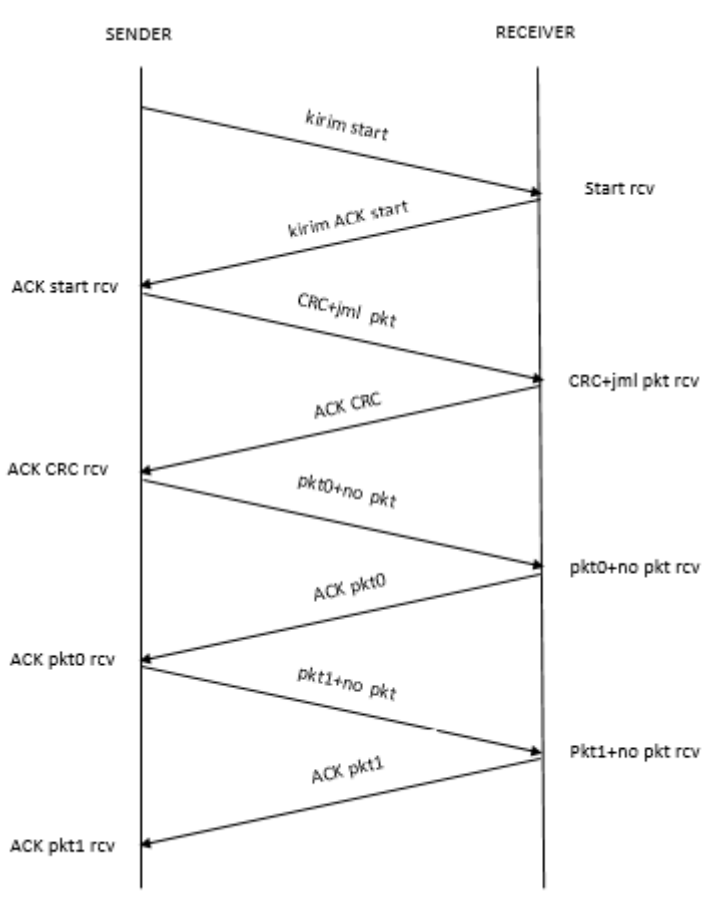
Gambar 4. Arduino Nano dan NRF24L01

4.1.5 Perancangan *Reliable Data Transfer*

Pada perancangan sistem *Reliable Data Transfer* ini akan dibuat mekanisme alur pengiriman data dari ketiga kondisi, yaitu *Operation with No Loss, Lost Packet* dan *Lost ACK*.

4.1.5.1 Perancangan *Operation with No Loss*

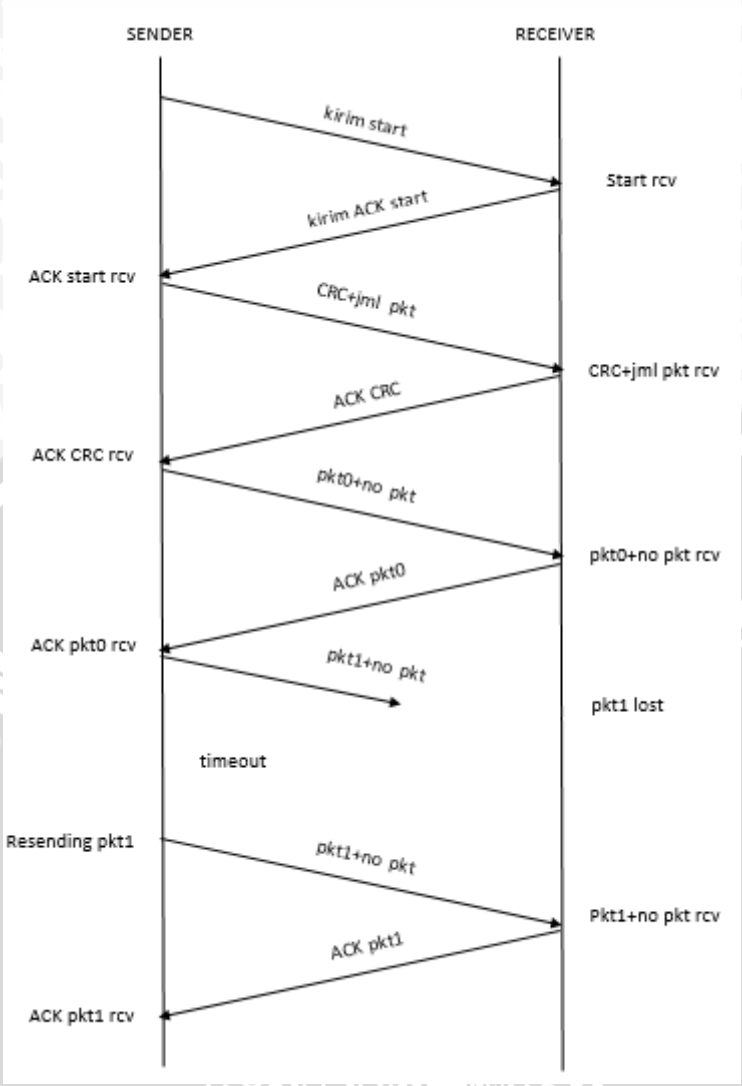
Pada perancangan ini semua data dikirim dengan baik, mulai dari start, kemudian CRC+jumlah paket dan kemudian data paket yang sudah dipecah-pecah yang disertakan dengan nomor paket dan keterangannya ditunjukkan pada gambar 4.3.



Gambar 4. Perancangan *Operation with No Loss*

4.1.5.2 Perancangan *Lost Packet*

Pada perancangan *Lost Packet*, ada beberapa paket yang tidak sampai ke *receiver*. Pertama pengiriman start dan CRC berjalan dengan baik, tetapi pada saat pengiriman paket, terjadi kehilangan paket di tengah jalan. Sehingga *sender* tidak menerima ACK dan akan mengirim ulang paket yang hilang seperti ditunjukkan pada gambar 4.4.

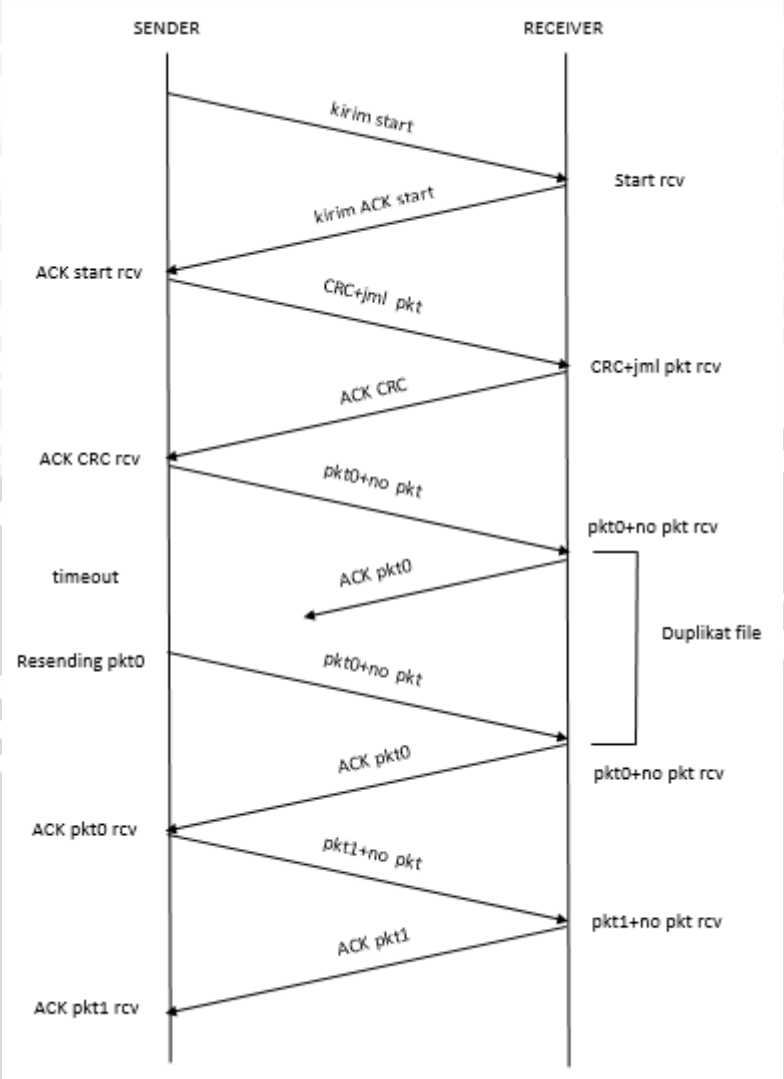


Gambar 4. Perancangan *Lost Packet*

4.1.5.3 Perancangan *Lost ACK*

Pada Perancangan *lost ACK* sama dengan perancangan *Lost Packet*, bedanya hanya terdapat duplikasi file pada *receiver.* Duplikasi file terjadi karena saat *sender* mengirim paket dan receiver menerima paket tersebut tetapi ACK dari *receiver* tidak sampai pada *sender* sehingga *sender* mengirim paket yang sama kepada receiver.

Di *receiver* terdapat dua paket yang sama dan cara mengatasinya adalah me*replace* paket yang lama dengan paket yang baru sehingga paket yang diterima tidak ada yang sama. Selengkapnya ditunjukkan pada gambar 4.5.



Gambar 4. Perancangan *Lost ACK*

4.1.6 Perancangan *Cyclic Redudancy Check* (CRC)

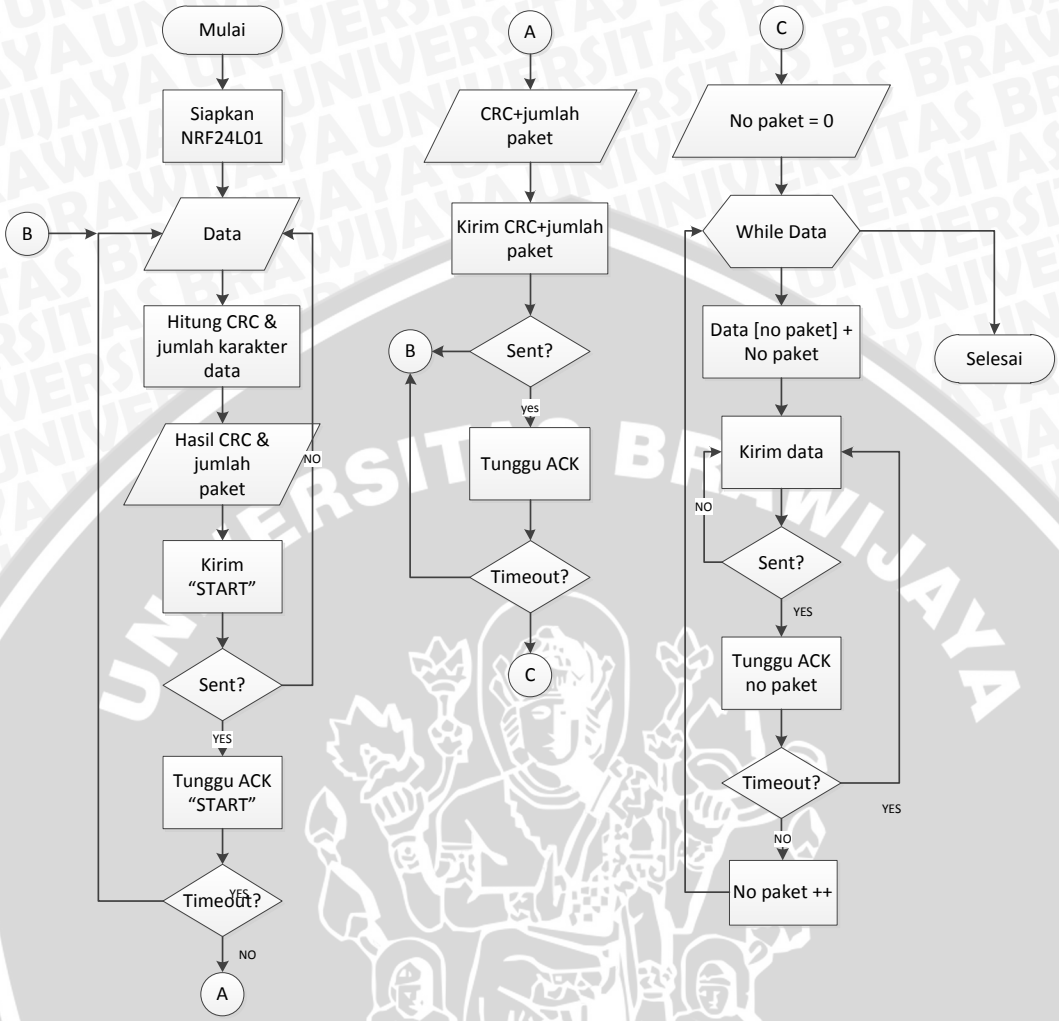
Dalam sistem ini, Cyclic Redudancy Check (CRC) digunakan untuk menguji kebenaran dari data yang dikirim. Di sisi penerima, *Cyclic Redudancy Check* akan menghhitung seluruh data yang ada dan dikirim terlebih dahulu ke *receiver* dalam bentuk hexa.

Kemudian di sisi penerima, *Cyclic Redudancy Check* akan menghitung kembali data yang diterima dari *sender* dan membandingkan dengan data CRC yang sudah diterima sebelumnya.

4.1.7 Perancangan Algoritma

Di bagian ini akan dijelaskan flowchart tentang *Sender* dan *Receiver* yang akan digunakan pada pengiriman data pada NRF24L01.

4.1.7.1 Perancangan Algoritma *Sender*

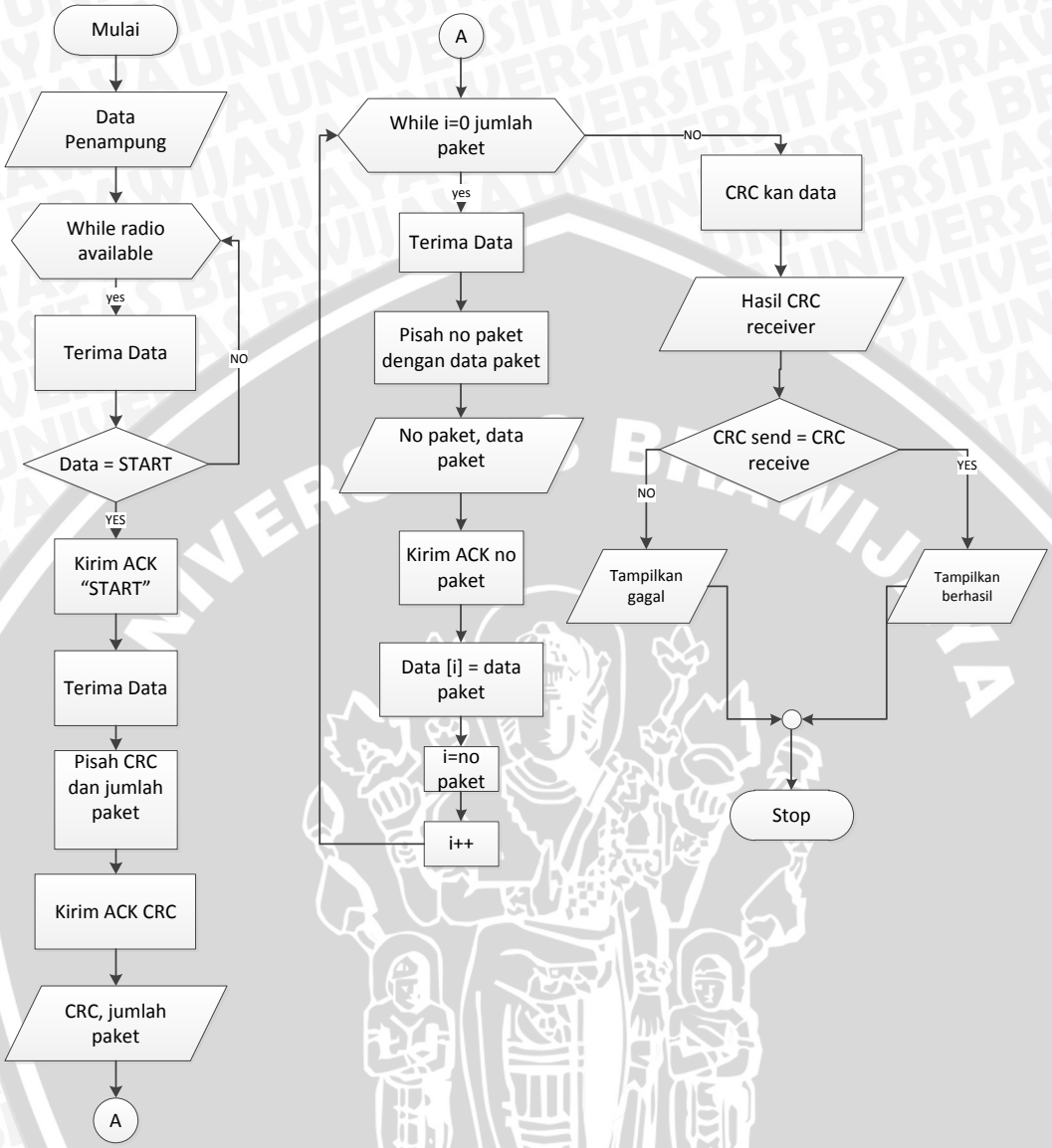


Gambar 4. Perancangan Algoritma *Sender*

Dari flowchart di atas, dijelaskan bahwa yang dilakukan pertama adalah menyiapkan NRF24L01, kemudian penyiapan data yang akan dikirim dalam bentuk string. Data variabel dihitung CRCnya dan jumlah karakter dalam variabel data tersebut. Kemudian hasilnya adalah CRC dan jumlah paket. Setelah itu Kirim "start" untuk menandakan bahwa ada paket yang akan dikirim. Kemudian apakah paket terkirim atau tidak, jika terkirim maka siap untuk menunggu ACK, dan jika tidak maka ulangi mengirim "start".

Jika ACK start diterima kurang dari batas waktu yang ditentukan maka tidak terjadi timeout, dan jika lebih maka akan terjadi timeout. Kemudian inisialisasi nomor paket. Selanjutnya ada looping sebanyak karakter yang dikirim. Karena data diakses per karakter, a=index ke 0, a+0 kemudian dikirim. Kemudian cek apakah terkirim atau tidak, jika terkirim maka tunggu ACK berupa nomor paket. Kemudian jika tidak ada data lagi, maka proses berhenti.

4.1.7.2 Perancangan Algoritma Receiver



Gambar 4. Perancangan Algoritma Receiver

Pada sisi *receiver,* pertama dilakukan penyiapan data penampung untuk menampung semua data yang diterima, setelah itu selama ada paket yang dikirm, *receiver* siap menunggu data. Setelah data diterima, kemudian data dicek, apakah kode "start"? Jika bukan berarti belum ada yang menandakan ada yang mengirim data. Jika iya, terima ACK "start". Kemudian *receiver* terima data CRC dan jumlah paket, pisah lalu dikirim. Kemudian looping sebanyak jumlah paket. Pada looping ini, *receiver* akan menerima data paket dan nomor paket. Dalam proses ini, ada kemungkinan terjadi duplikasi paket karena ACK yang dikirimkan *receiver* tidak sampai ke *sender* dan paket yang sama akan dikirimkan ulang. Untuk mengatasi duplikasi paket, maka setiap paket yang diterima akan dimasukkan ke dalam index array berdasarkan nomor paket. Jadi jika *receiver* menerima paket yang sama, maka otomatis melakukan *replace* pada paket sebelumnya. Setelah looping selesai maka data akan dikumpulkan dan masuk dalam proses CRC. Setelah CRC selesai

dihitung, maka CRC di recever akan dibandingkan dengan CRC yang sudah dikirimkan oleh *sender* di awal.

4.2 Implementasi

Implementasi sistem akan dibatasi dengan batasan implementasi. Selain itu terdapat implementasi berupa implementasi perangkat keras dan perangkat lunak berdasarkan perancangan sistem yang telah dibuat sebelumnya.

4.2.1 Batasan Implemantasi

Beberapa batasan dalam implementasi dalam sistem Implementasi Metode *Reliable Data Transfer* dan CRC Pada Arduino Dengan Komunikasi RF adalah:

1. Penerapan sistem hanya pada arduino nano
2. Menggunakan modul *wireless* NRF24L01
3. Data yang dikirim berupa text
4. Pengujian *Reliable Data Transfer* hanya berfokus pada keberhasilan pengiriman
5. Metode *checksum* menggunakan CRC8

4.2.2 Implementasi Arduino dengan NRF24L01

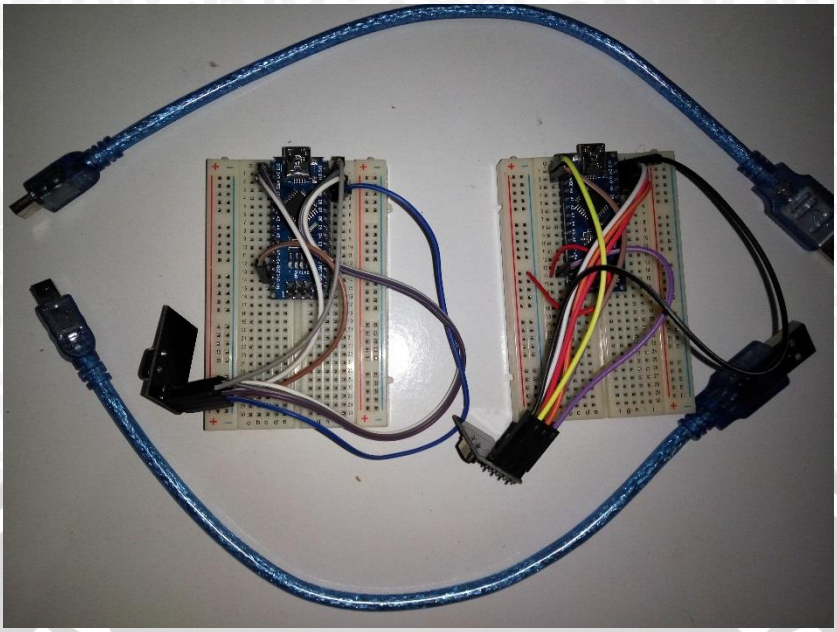
Penghubungan antara Arduino Nano dengan NRF 24L01 bisa dilihat pada tabel 4.1.

Tabel 4. Penghubungan pin Arduino Nano dan NRF24L01

Arduino Nano	NRF24L01
3,3V	VCC
GND	GND
D12	MISO
D11	MOSI
D13	SCK
D9	CE
D10	CSN

Sumber: (Lenotta, 2013)

Berikut adalah gambar rangkaian setelah dihubungkan sesuai tabel di atas.



Gambar 4. Rangkaian Arduino Nano dan NRF24L01

4.2.3 Implementasi Program

Pada sub bab ini akan dijelaskan implementasi program pada sisi *sender* dan sisi *receiver.*

4.2.3.1 Implementasi Sender

Pada implementasi *sender,* akan dijelaskan potongan program yang akan digunakan untuk mengimplementasikan program untuk mengirim data pada NRF24L01.

Pesan yg akan dikirim dalam format String

```
msg1=”isi pesan”
```

Setelah itu untuk memanggil fungsi CRC dan hasilnya disimpan di variable hasil, digunakan source code

```
Hasil=CRC8(msg1);
```

Cari CRC dari msg1, dengan source code:

```
String crc8(String msg1){
  byte a[8];
  int Panjang_Pesan=msg1.length();
  Serial.println("");  Serial.print("Jumlah Karakter : ");
  Serial.print(Panjang_Pesan);
  Serial.println("");
  //char msg[Panjang_Pesan];
```

```
/*for(int i=0;i<Panjang_Pesan;i++){
   msg[i]=msg1.charAt(i);
  }  */

/*Konversi Pesan Ke dalam Bit untuk di CRC kan*/
byte data[Panjang_Pesan*8];
int k=0;
for(int i=0; i<Panjang_Pesan; i++){
   char v = msg1.charAt(i);
    for(int j=7; j>=0; j--){
      data[k]=bitRead(v,j);
      k++;
    }
}
Serial.print("BINARY : ");
for(int i=0;i<Panjang_Pesan*8;i++){
  Serial.print(data[i]);
  }
byte peng_xor[9]={1,0,0,0,0,0,1,1,1};
//byte coba;
int rr;
int panjang_data=sizeof(data);
for(int i=0;i<panjang_data;i++){
  if(data[i]==B1){
    rr=(sizeof(data))-i;
    if(rr>=sizeof(peng_xor)){
        for(int j=0;j<9;j++){
          data[i+j] = peng_xor[j]^data[i+j];

        }
    }
    else{
      Serial.println("");
      Serial.print("CRC8 : ");
      int g=0;
      for(int u=i;    u<panjang_data;      u++){
```

```
            Serial.print(data[u]);
            a[g]=data[u];
            g++;
          }
          break;
        }
      }
    }

Serial.println("");

k=0;
int hasil=0;
int total=1;
byte hex[2];
int index_hex=1;
for(int i=sizeof(a)-1;i>=0;i--){
  if(a[i]==1){
    if(k==0){total=1;}
    else{
      total=1;
      for(int b=1;b<=k;b++){
        total=total*2;
        }
      }

    }
  else{
    total=0;
    }

  hasil=hasil+total;
 if(k==3){
    k=-1;
    hex[index_hex]=hasil;
    index_hex--;
```

```
      hasil=0;
      }
    k++;

  }
  String datanya;
  datanya+=(String(hex[0],HEX));
  datanya+=(String(hex[1],HEX));
  return datanya;
}
```

Kemudian dilakukan pengiriman tanda awal untuk memulai melakukan pengiriman data yaitu dengan mengirimkan “start”, dengan source code sebagai berikut

```
radio.stopListening();
  if (!radio.write( "start", 6 )){
      Serial.println("failed");
      Serial.println("Retrying");
     }
  else{
    Serial.println("");
    Serial.print("1. Kirim : ");
    Serial.print("start");
    Serial.println("");
    boolean timeout = false;
    timeout = perhitungan_timeout();
```

Siap terima ACK start dengan batas timeout. Setelah berhasil mengirim “start”, kemudian program memanggil fungsi timeout yang hasil perhitungannya disimpan di variable boolean dengan source code:

```
boolean perhitungan_timeout(){
  delay(40);
  radio.startListening();
  unsigned long started_waiting_at = millis();
  boolean timeout = false;
  Serial.print("Waktu Terkirim : ");
  Serial.print(started_waiting_at);
```

```
 Serial.print("  ms");
  while ( !radio.available() && !timeout ){
    if (millis() - started_waiting_at > 1000){

            timeout = true;
            break;
        }
    }
```

Didapatkan apakah ack yang datang atau diterima timeout atau tidak, JIKA tidak maka menerima data ack dengan source code:

```
char data[6]={0};
      radio.read(&data,6);
      String data1;
      for(int i=0;i<=4;i++){
        data1+=data[i];
       }
```

Data yang diterima dimasukkan ke dalam variable string data1, kemudian data1 dibandingkan apakah ack yang diterima tadi berupa ack start, dengan source code

```
if(data1=="start")
```

jika ack start kemudian menyiapkan paket crc dan jumlah paket yg akan dikirimkan, source code sebagai berikut

```
hasil+="+";
        hasil+=Panjang_Pesan;;
        delay(40);
        char hasil1[hasil.length()+1];
        for(int
i=0;i<sizeof(hasil1);i++){hasil1[i]=hasil.charAt(i);}
```

Maka akan didapatkan format data hex+jumlah data, setelah itu siap menerima ACK CRC, apakah CRC atau tidak.

```
boolean timeout = false;
          timeout = perhitungan_timeout();
          if(timeout){Serial.println("ACK                TIMEOUT
!!!!");Serial.println("Resending ACK");}
else{
            char data[4]={0};
            radio.read(&data,4);
```

```
            String data1;
            for(int i=0;i<=2;i++){
               data1+=data[i];
             }
```

Jika ACK benar ACK CRC, maka menyiapkan looping sebanyak jumlah paket dengan kode

```
if(data1=="CRC"){
               Serial.println("TERKIRIM");
               String datatosend;
               int retry;
               for(int i=0;i<Panjang_Pesan;i++)
```

Di dalam loping akan dilakukan pengiriman pesan per karakter yang sudah ditambahkan dengan no paket, berikut source code penomoran paket

```
datatosend="";
                 datatosend+=msg1.charAt(i);
                 datatosend+="+";
                 datatosend+=i;
```

Maka paket yg siap dikirim memiliki format “paket+nomor paket”. Kemudian dilakukan pengiriman tiap paket dan penerimaan ACKtiap paket, dengan source code

```
for(int i=0;i<Panjang_Pesan;i++){
                 radio.stopListening();
                 datatosend="";
                 datatosend+=msg1.charAt(i);
                 datatosend+="+";
                 datatosend+=i;
                 char hasil2[datatosend.length()+1];
                 for(int
n=0;n<sizeof(hasil2);n++){hasil2[n]=datatosend.charAt(n);Ser
ial.print(datatosend.charAt(n));}

                 if (!radio.write(hasil2,sizeof(hasil2))){
                   Serial.println("failed");
                   i--;
                   Serial.println("Retrying");
                  }
                 else{
```

```
                    Serial.print(i);
                    Serial.print(". Kirim Paket : ");
                    Serial.print(hasil2);
                    Serial.println("");
                    boolean timeout = false;
                    timeout = perhitungan_timeout();
                    if(timeout){i--;Serial.println("ACK  PAKET
TIMEOUT !!!!");Serial.println("Resending Paket");i--;
                       retry++;
                       if (retry>3){i=Panjang_Pesan;}
                       Serial.print(retry);}
                    else{
                      char data2[32]={0};
                      radio.read(&data2,32);
                      Serial.println(data2);
                      String ii;
                      for(int
n=0;n<sizeof(data2);n++){ii+=data2[n];}


                      if(ii.toInt()==i){Serial.println("ACK
OK");}
                      else{
                        Serial.println("RESEND PAKET");
                        i--;
                        retry++;
                        if (retry>3){i=Panjang_Pesan;}
                        Serial.print(retry);
                        }
                      }
                    }
                  }
```

4.2.3.2 Implementasi *Receiver*

Pada implementasi receiver, yang dilakukan pertama adalah menunggu paket "start" dengan source code

```
while(!radio.available()){
    waktu=millis();
  }
```

```
  char text1[32]={0};
  radio.read(&text1, sizeof(text1));
  Serial.println(text1);
  String data1;
     for(int i=0;i<=4;i++){
         data1+=text1[i];
       }
```

Kemudian data yang diterima ditest apakah tanda “start” atau bukan, berikut *source code*nya

```
if(data1=="start")
```

Jika benar, maka akan dijalankan *source code* sebagai berikut

```
radio.stopListening();
    radio.write("start", 6);
kemudian menunggu data crc dan jumlah paket, kode
radio.startListening();
    while(!radio.available()){
      waktu=millis();
    }

    radio.read(&text1, sizeof(text1));
    Serial.println(text1);
    String data1;
    for(int i=0;i<5;i++){
           data1+=text1[i];
         }
```

Data yang diterima dicek di fungsi pemisah, apakah paket bisa dipecah atau tidak dengan memanggil fungsi pemisah dengan *source code*

```
String crc8;
    crc8=pemisah(data1,"crc");
```

Dan berikut adalah fungsi pemisah

```
String pemisah(String msg1, String tandanya){
  int Panjang_Pesan=msg1.length();
  char msg[Panjang_Pesan];
```

```
  for(int i=0;i<Panjang_Pesan;i++){
     msg[i]=msg1.charAt(i);
    }
  char pesan[3];
  char no_paket[2];
  int pembatas=0;
  for(int i=0;i<Panjang_Pesan;i++){
    if(msg[i]=='+'){
       pembatas=i;
      }

    }
  if(pembatas==0){return "no";}
  int n=0;

  for(int i=pembatas+1;i<Panjang_Pesan;i++){
      no_paket[n]=msg[i];
      n++;
    }
  for(int i=0;i<pembatas;i++){
      pesan[i]=msg[i];
    }
  if (tandanya=="crc"){
        String data1;
        for(int i=0;i<pembatas;i++){
          data1+=pesan[i];
         }
    return data1;
  }
  else{
        String data1;
        for(int i=0;i<sizeof(no_paket);i++){
          data1+=no_paket[i];
         }
  return data1;}
}
```

Dari fungsi pemisah itu membutuhkan input pesan yang diterima oleh NRF24L01 dan sebuah variable tanda untuk membedakan apakah fungsi pemisah akan memisahkan paket untuk crc dan jumlah paket atau data paket dan nomer paket.

Hasil pemanggilan fungsi pemisah dimasukkan ke dalam variable CRC8 untuk dibandingkn apakah hasil keluaran adalah "NO". Jika "NO" berarti data tidak bisa dipecah dan program kembali ke awal. Jika tidak maka program akan mendapatkan data CRC yang telah diterima. Kemudaian program akan menyiapkan looping sebanyak jumlah paket. Jumlah paket didapatkan dengan memanggil fungsi pemisah

```
String jml_pkt;
        jml_pkt=pemisah(data1,"cr");
```

setelah itu receiver akan mengirimkan ACK CRC dengan kode

```
radio.stopListening();
        radio.write("CRC", 6);
```

setelah itu, *receiver* siap menerima data sebanyak jumlah paket. Menyiapkan variabel penampung pesan sebanyak jumlah paket

```
char datanya_terakhir[jml_pkt.toInt()];
for(int i=0;i<jml_pkt.toInt();i++){
          radio.startListening();
          while(!radio.available()){
            waktu=millis();
          }

          radio.read(&text1, sizeof(text1));
          String no_pkt;
          String data_pkt;
          no_pkt=pemisah(text1,"cr");
          if(no_pkt=="no"){break;}
          data_pkt=pemisah(text1,"crc");

datanya_terakhir[no_pkt.toInt()]=data_pkt.charAt(0);
          Serial.println(text1);
          delay(40);
          radio.stopListening();
          char da[no_pkt.length()+1];
          for(int
n=0;n<sizeof(da);n++){da[n]=no_pkt.charAt(n);}
```

```
        radio.write(da, sizeof(da));
        delay(40);
      }
```

Dalam looping sebanyak jumlah paket, program menunggu paket kemudaian memisah paket menjadi 2 bagian taitu data paket dan nomor paket, jika tidak bidsa dipecah maka akan keluar dari looping. Jika bisa maka data paket akan dimasukkan ke dalam variabel datanya_terakhir pada index array ke nomer paket kemudian akan mengirimkan ack no paket.

Setelah keluar dari looping, maka didapatkan variabel datanya terakhir yang sudah terisi oleh hasil pengiriman sender. Kemudian dilakukan CRC8 terhadap variabel datanya_terakhir

```
String data_final;
       for(int i=0;i<sizeof(datanya_terakhir);i++){
          data_final+=datanya_terakhir[i];
        }
       Serial.println(data_final);
       Serial.println(data_final.length());
       String crc8_rcv;
       crc8_rcv=crc88(data_final);
```

Setelah itu *receiver* memiliki data CRC yang bisa dibandingkan yaitu crc8 dan crc8_rcv, jika sama maka data yang diterima benar. Kemudian program akan memulai menerima data dari awal lagi.

BAB 5 PENGUJIAN DAN ANALISIS

Bab ini membahas mengenai tahapan pengujian dan analisis dari sistem yang sudah dibuat.

5.1 Pengujian

Pada sub bab pengujian akan dilakukan 2 pengujian, yaitu terhadap operasi CRC dan *Reliable Data Transfer.*

5.1.1 Pengujian CRC

Tujuan pengujian CRC ini adalah untuk memastikan kebenaran data yang dikirim. Data yang akan dikirim berupa data string. Data bisa dilihat dalam tabel 5.1

Tabel 5. Hasil uji CRC

No.	Data	Jumlah	Konvert ke bit	Jumlah bit	Hasil CRC	Hex
1.	AS	2	0100000101010011	16	10010011	93
2.	AA	2	0100000101000001	16	10000001	81
3.	SS	2	0101001101010011	16	11101101	dc
4.	QA	2	0101000101000001	16	11110001	f1
5.	ZZ	2	0101101001011010	16	11011011	ca

Contoh data yang dikirimkan adalah “AS”. Pertama data akan dikonvert menjadi bit oleh program dan menghasilkan data 0100000101010011. Kemudian akan dilakukan proses CRC dengan melakukan proses XOR dengan bit 10010011 sebagai pengXOR. Tabel kebenaran XOR adalah sebagai berikut:

Tabel 5. Tabel XOR

Masukan		**Keluaran**
A	**B**	**XOR**
0	0	0
0	1	1
1	0	1
1	1	0

Bit 0100000101010011 di XOR dengan 100000111 dan akan mendapatkan hasil 000000010010011. Karena bit 0 yang terdapat di depan tidak digunakan maka hasil CRC menjadi 10010011. Hasil pengujian manual dengan hasil uji program adalah sama.

Dilihat dari hasil CRC, setiap 1 perbedaan huruf maka akan didapatkan hasil konvert bit yang berbeda. Dengan demikian CRC bisa digunakan sebagai *checksum* dari data yang akan dikirim. Lalu pada sisi *receiver* juga akan melakukan proses CRC pada data yang diterima dan akan dilakukan pembandingan dengan CRC yang diterima di awal.

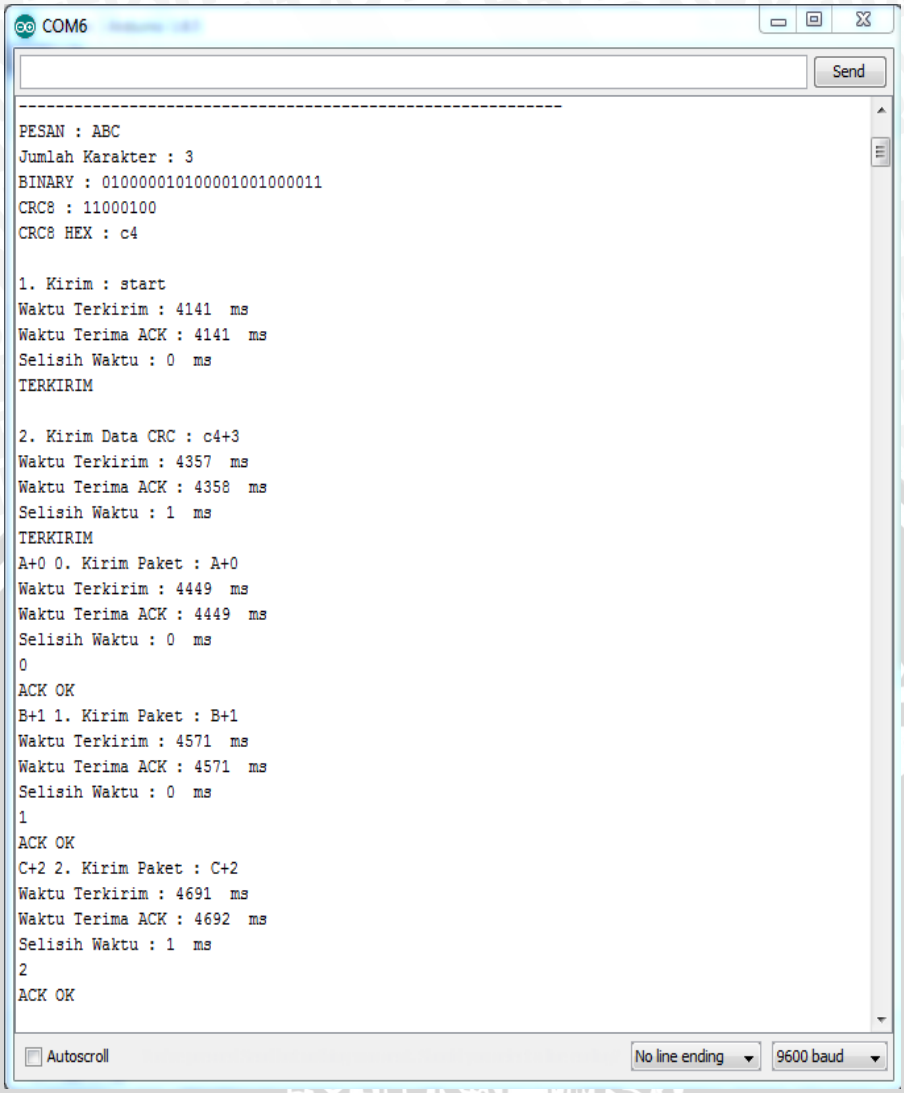
5.1.2 Pengujian Reliable Data Transfer

Pengujian pada sistem ini dimulai dengan mengirim file berbentuk string. Dalam pengujian pengiriman tersebut terdapat 3 skenario yaitu *operation with no loss, lost packet* dan *lost ack.*

5.1.2.1 Operation With No Loss

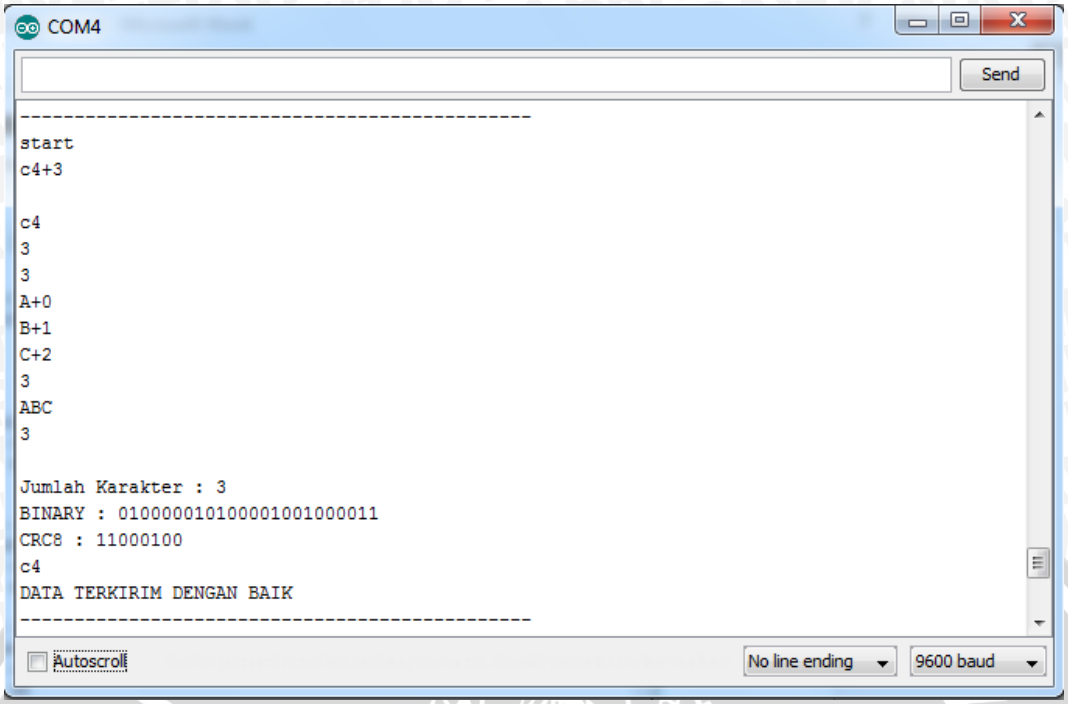
Pada skenario ini, pengiriman paket beserta file CRC berjalan tanpa gangguan serta paket diterima dengan baik, dengan kata lain tidak ada *lost packet* atau *lost ACK.*

Contoh pertama adalah pengiriman data dari *sender* dengan nama data "ABC". Pertama akan ditampilkan jumlah paket yang akan dikirim dan kode dalam binernya. Kemudian dilakukan proses CRC untuk mendapatkan bit hasil CRC yang kemudian akan dikonvert menjadi hexa dan kemudian dikirimkan pada receiver. Pengiriman dimulai dengan mengirim penanda awal bernama "start". Setelah "start" terkirim, maka paket akan dikirim satu per satu ke receiver. Format pengiriman paket adalah paket + nomor paket. Hasil pengujian o*peration with no loss* pada *sender* bisa dilihat pada gambar 5.1



Gambar 5. Contoh Operation with No Loss pada *sender*

Selanjutnya di sisi *receiver* akan ditampilkan penanda yang pertama kali dikirim oleh *sender.* Setelah itu *receiver* menerima paket hasil CRC dan jumlah paket yang akan dikirim. *Receiver* kemudian menerima paket yang dikirim satu per satu oleh *sender.* Setelah semua paket diterima, maka akan dihitung jumlah paket yang diterima dan menampilkan semua paket yang dikirim. Kemudian paket akan dihitung CRCnya dan dibandingkan dengan CRC pengirim. Hasil pengujian o*peration with no loss* pada *receiver* bisa dilihat pada gambar 5.2

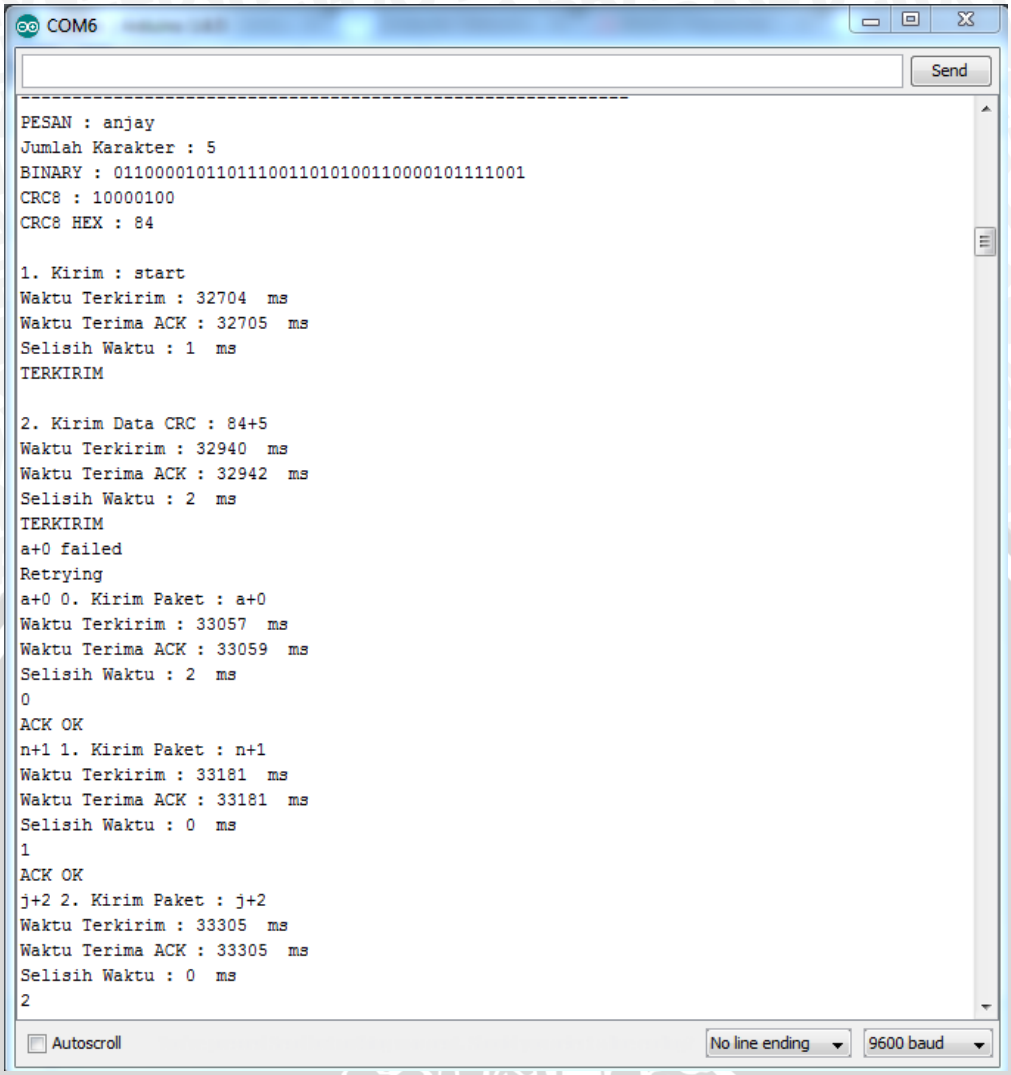


Gambar 5. Contoh Operation with No Loss pada *receiver*

5.1.2.2 Lost Packet

Pada skenario ini, pengiriman paket mengalami gangguan dan tidak sampai kepada *receiver* dan pada sender terjadi timeout karena ACK tidak datang sehingga sender mengirim paket ulang.

Pada gambar 5.3 dapat dilihat bahwa pengiriman paket "A" terjadi kegagalan. Dalam kasus *packet loss* ini kehilangan paket akan ditanggulangi RDT dengan mengirim ulang paket yang gagal. Dapat dilihat pada gambar 5.3 saat terjadi *failed,* maka pada sisi *sender* akan mengalami timeout dan *sender* akan melakukan *retrying* dan paket akan dikirim ulang.



Gambar 5. Contoh Lost Packet

5.1.2.3 Lost ACK

Pada skenario ini pengiriman paket berhasil, tapi *Sender* tidak menerima ACK kembali dari *Receiver* sehingga *Sender* mengirim paket kembali. Karena sebelumnya di sisi *Receiver* telah terdapat paket, maka terjadi duplikasi file. Maka selanjutnya *Receiver* akan menghapus salah satu file yang terduplikat.

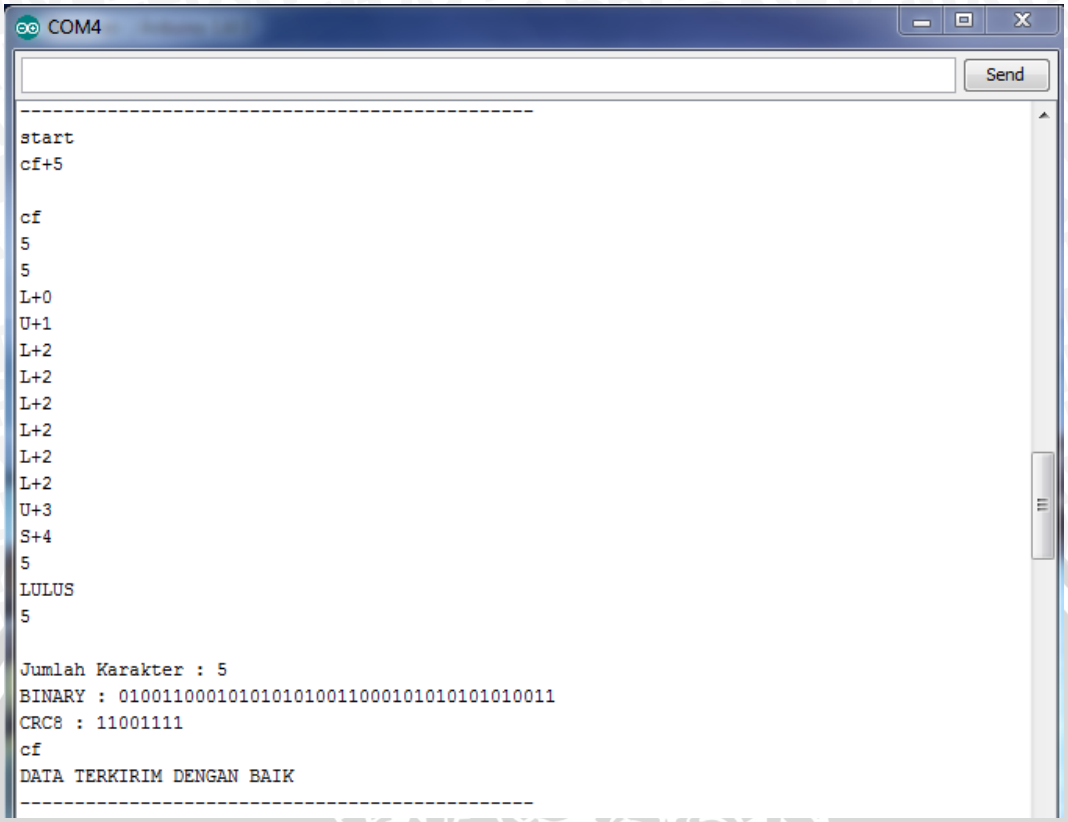
Dapat dilihat pada gambar 5.4 bahwa paket "A" sudah berhasil dikirim tapi tidak mendapatkan ACK dari *receiver.* Jika ini terjadi, maka *sender* akan mengirim ulang paket yang sama sampai mendapat ACK OK dari receiver. Dan pada *receiver* otomatis terdapat duplikat file. Hasil pengujian *lost packet* dapat dilihat pada gambar 5.4

```
a+3 3. Kirim Paket : a+3
Waktu Terkirim : 62979  ms
Waktu Terima ACK : 62980  ms
Selisih Waktu : 1  ms
2
RESEND PAKET
a+3 3. Kirim Paket : a+3
Waktu Terkirim : 63109  ms
Waktu Terima ACK : 63110  ms
Selisih Waktu : 1  ms
3
ACK OK
y+4 4. Kirim Paket : y+4
Waktu Terkirim : 63233  ms
Waktu Terima ACK : 63234  ms
Selisih Waktu : 1  ms
3
RESEND PAKET
y+4 failed
Retrying
y+4 4. Kirim Paket : y+4
Waktu Terkirim : 63393  ms
Waktu Terima ACK : 63393  ms
Selisih Waktu : 0  ms
4
ACK OK
```

Gambar 5. Contoh Lost ACK pada *sender*

Pada sisi *receiver,* dilihat bahwa terjadi duplikat file pada paket "L". Ini terjadi karena ACK dari paket "L" hilang dan tidak diterima oleh *sender*. Dalam kasus ini, *receiver* akan menghapus duplikat data yang ada. Dan bisa dilihat pada paket yang sudah digabungkan lagi bahwa paket utuh seperti yang dikirim *sender*. Duplikat data bisa diketahui karena data yang dikirim oleh *sender* disertai nomor paket. Seperti pada gambar 5.5 dapat dilihat bahwa beberapa data yang diterima terdiri dari dua paket bernama "L", tetapi karena diberi penomoran paket sehingga *receiver* dapat mengetahui jika terjadi duplikasi paket.

Ketika receiver menerima setiap paket data, receiver langsung memasukkan data tersebut ke dalam array berdasarkan nomor paket. Berarti jika L adalah paket nomor 2 dan diterima berulang-ulang, maka array index ke-2 akan diisi L yang terakhir kali diterima.



Gambar 5. Contoh Lost ACK pada *receiver*

5.1.3 Pengujian Sistem Keseluruhan

Dalam bagian ini akan diuji pengiriman data sebanyak 10 kali oleh masing-masing data. Kemudian akan dicatat hasil berapa kali data tersebut mengalami *no loss, lost packet* dan *lost ACK*.

Tabel 5. Tabel Pengujian Keseluruhan

Data	Pengujian	CRC	Hex	No Loss	Lost Packet	Lost ACK
KAMPUS	10 kali	10010000	f0	7 kali	3 kali	1 kali
KULIAH	10 kali	11010111	d7	4 kali	4 kali	2 kali
SKRIPSI	10 kali	11100110	e6	5 kali	4 kali	1 kali
PAMERAN	10 kali	11001001	c9	4 kali	6 kali	0 kali
SIDANG	10 kali	10101000	a8	8 kali	1 kali	1 kali
YUDISIUM	10 kali	10001000	88	5 kali	5 kali	0 kali
WISUDA	10 kali	10100000	a0	3 kali	4 kali	3 kali
LULUS	10 kali	11001111	cf	5 kali	3 kali	2 kali

5.2 Analisis

Dalam pengujian *Reliable Data Transfer* ini, sistem hanya difokuskan pada keberhasilan pengiriman data tanpa ada faktor lain seperti faktor jarak antara *sender* dan *receiver.* Jadi, untuk menjalankan skenario *lost packet* dan *lost ACK,* bisa dengan mengurangi waktu timeout pada sender.

Sistem Reliable Data Transfer yang dibuat memiliki kelemahan yaitu proses eksekusinya lebih lama dibanding sistem Reliable Data Transfer yang sebenarnya karena dalam sistem ini program dimulai dengan mengirimkan inisialisasi terlebih dahulu dan proses CRC dilakukan pada seluruh data, bukan pada setiap paket yang dikirimkan.

BAB 6 KESIMPULAN DAN SARAN

Bab ini berisi kesimpulan dari hasil perancangan, implementasi, dan analisis pengujian yang telah dilakukan, beserta saran yang dapat meningkatkan dan mengembangkan sistem yang telah ada.

6.1 Kesimpulan

Kesimpulan yang dapat diambil setelah melakukan perancangan, implementasi dan analisis pengujian adalah sebagai berikut:

1. *Reliable Data Transfer* bertujuan untuk memastikan pengiriman data dari *sender* ke receiver bisa terkirim dengan baik sekaligus ada pengecekan nilai dari data yang dikirim dalam proses CRC.
2. Pada pengujian *lost packet* dapat diketahui bahwa jika terdapat paket yang *lost* saat pengiriman, maka pada *sender* akan tampil bahwa paket yang dikirim mengalami *loss* dan dengan metode *Reliable Data Transfer* maka *lost packet* akan ditanggulangi dengan mengirim ulang paket yang sama.
3. Pada pengujian *lost ACK*, hilangnya ACK dapat diketahui jika pada sisi *receiver* terdapat duplikasi paket. Hal ini terjadi karena saat *receiver* mengirimkan kembali ACK, ACK tersebut tidak sampai kepada *sender* dan otomatis karena tidak mendapatkan ACK, *sender* akan mengirim ulang paket yang sama. Di sisi *receiver* akan mendeteksi paket yang sama, tapi hal ini dapat diatasi karena setiap paket yang diterima *receiver* akan dimasukkan ke dalam index sesuai nomor paket. Jadi data akhir pada *receiver* tidak ada yang terduplikasi.
4. Penggunaan *Cyclic Redudancy Check* (CRC) dalam sistem ini dapat bekerja dengan baik sesuai fungsinya, yaitu untuk memeriksa kebenaran dari nilai data yang dikirim dan diterima dan membandingkannya di akhir.

6.2 Saran

1. Untuk penerapan *Reliable Data Transfer* kedepannya bisa menggunakan sistem *Reliable Data Transfer* yang asli agar meminimalisir waktu eksekusi data karena sistem *Reliable Data Transfer* yang asli langsung mengirimkan data tampa melakukan inisialisasi di awal dan pengiriman langsung disertakan dengan *checksum*.
2. Pada penelitian selanjutnya, pengiriman data bukan berupa data biasa atau inputan manual, melainkan membaca data dari sensor dengan kata lain data yang dikirimkan bersifat dinamik.
3. Untuk pengembangan lebih lanjut bisa diterapkan pada sistem nyata, seperti sistem rumah cerdas, sistem transportasi dan lain-lain.

DAFTAR PUSTAKA

(2009). Retrieved from hackersdelight: http://www.hackersdelight.org/crc.pdf

(2011). Retrieved from Forum Arduino: www.forum.arduino.cc/

Antasari, G. (2009). HI : An Hybrid Adaptive Interleaved communication protocol for reliable data transfer in WSNs with mobile sinks.

Arduino. (2015). Retrieved from www.arduino.cc: https://www.arduino.cc/en/Main/ArduinoBoardNano

Charles's Blog. (2013). Retrieved from https://hallard.me/nrf24l01-real-life-range-test/

Humphrys, D. M. (1987). Retrieved from http://www.computing.dcu.ie/~humphrys/Notes/Networks/data.polynomial.html.

Kong, L. (2013). Data Loss and Reconstruction in Sensor Networks.

Kurose, J. F. (2012). *Computing Networking: A Top Down Approach.*

Lenotta, H. (2013). Retrieved from http://hack.lenotta.com/arduino-raspberry-pi-switching-light-with-nrf24l01/

Tian, K. (2014). A high-speed reliable data transfer method based on FPGA hardware acknowledgement .

LAMPIRAN

Lampiran Program *Sender*

```
#include <nRF24L01.h>
#include <RF24.h>
#include <RF24_config.h>
#include <SPI.h>
RF24 radio(9,10);

byte addresses[][6] = {"1Node","2Node"};

String crc8(String msg1){
  byte a[8];
  int Panjang_Pesan=msg1.length();
  Serial.println("");  Serial.print("Jumlah Karakter : ");
  Serial.print(Panjang_Pesan);
  Serial.println("");
  //char msg[Panjang_Pesan];
  /*for(int i=0;i<Panjang_Pesan;i++){
     msg[i]=msg1.charAt(i);
    }  */

  /*Konversi Pesan Ke dalam Bit untuk di CRC kan*/
  byte data[Panjang_Pesan*8];
  int k=0;
  for(int i=0; i<Panjang_Pesan; i++){
     char v = msg1.charAt(i);
      for(int j=7; j>=0; j--){
        data[k]=bitRead(v,j);
        k++;
      }
  }
  Serial.print("BINARY : ");
  for(int i=0;i<Panjang_Pesan*8;i++){
    Serial.print(data[i]);
    }
```

```
byte peng_xor[9]={1,0,0,0,0,0,1,1,1};
//byte coba;
int rr;
int panjang_data=sizeof(data);
for(int i=0;i<panjang_data;i++){
  if(data[i]==B1){
    rr=(sizeof(data))-i;
    if(rr>=sizeof(peng_xor)){
        for(int j=0;j<9;j++){
          data[i+j] = peng_xor[j]^data[i+j];

        }
    }
    else{
      Serial.println("");
      Serial.print("CRC8 : ");
      int g=0;
      for(int u=i;    u<panjang_data;      u++){
          Serial.print(data[u]);
          a[g]=data[u];
          g++;
        }
        break;
      }
    }
  }

Serial.println("");

k=0;
int hasil=0;
int total=1;
byte hex[2];
```

```
  int index_hex=1;
  for(int i=sizeof(a)-1;i>=0;i--){
    if(a[i]==1){
      if(k==0){total=1;}
      else{
        total=1;
        for(int b=1;b<=k;b++){
          total=total*2;
          }
        }

      }
    else{
      total=0;
      }

     hasil=hasil+total;
    if(k==3){
      k=-1;
      hex[index_hex]=hasil;
      index_hex--;
      hasil=0;
      }
     k++;


  }
  String datanya;
  datanya+=(String(hex[0],HEX));
  datanya+=(String(hex[1],HEX));
  return datanya;
}

boolean perhitungan_timeout(){
  delay(40);
  radio.startListening();
  unsigned long started_waiting_at = millis();
```

```
  boolean timeout = false;
  Serial.print("Waktu Terkirim : ");
  Serial.print(started_waiting_at);
  Serial.print("  ms");
  while ( !radio.available() && !timeout ){
    if (millis() - started_waiting_at > 1000){

          timeout = true;
          break;
      }
    }
  long sekarang= millis();
  Serial.println("");
  Serial.print("Waktu Terima ACK : ");
  Serial.print(sekarang);
  Serial.print("  ms");
  Serial.println("");
  Serial.print("Selisih Waktu : ");
  Serial.print(sekarang-started_waiting_at);
  Serial.print("  ms");
  Serial.println("");
  return timeout;
}

void setup() {
  Serial.begin(9600);
  radio.begin();
  radio.openWritingPipe(addresses[1]);
  radio.openReadingPipe(1,addresses[0]);

}

void loop() {
  Serial.println("");
  Serial.println("----------------------------------------
-----------------");
```

```
  String msg1="AKU";
  Serial.print("PESAN : ");
  Serial.print(msg1);
  int Panjang_Pesan=msg1.length();
  String hasil;
  hasil = crc8(msg1);
  Serial.print("CRC8 HEX : ");
  Serial.print(hasil);
  Serial.println("");
  radio.stopListening();
  if (!radio.write( "start", 6 )){
       Serial.println("failed");
       Serial.println("Retrying");
      }
  else{
    Serial.println("");
    Serial.print("1. Kirim : ");
    Serial.print("start");
    Serial.println("");
    boolean timeout = false;
    timeout = perhitungan_timeout();

    if(timeout){Serial.println("Start                    TIMEOUT
!!!!");Serial.println("Resending Start");}
    else{
      char data[6]={0};
      radio.read(&data,6);
      String data1;
      for(int i=0;i<=4;i++){
          data1+=data[i];
       }
      if(data1=="start"){
        Serial.println("TERKIRIM");
        Serial.println("");
        delay(40);
        radio.stopListening();
```

```
        delay(40);
        hasil+="+";
        hasil+=Panjang_Pesan;;
        delay(40);
        char hasil1[hasil.length()+1];
        for(int i=0;i<sizeof(hasil1);i++){
          hasil1[i]=hasil.charAt(i);}
        if (!radio.write(hasil1,sizeof(hasil1))){
           Serial.println("failed");
           Serial.println("Retrying");
           }
        else{
          Serial.print("2. Kirim Data CRC : ");
          Serial.print(hasil1);
          Serial.println("");
          boolean timeout = false;
          timeout = perhitungan_timeout();
          if(timeout){Serial.println("ACK                TIMEOUT
!!!!");Serial.println("Resending ACK");}
          else{
            char data[4]={0};
            radio.read(&data,4);
            String data1;
            for(int i=0;i<=2;i++){
                data1+=data[i];
              }
            if(data1=="CRC"){
              Serial.println("TERKIRIM");
              String datatosend;
              for(int i=0;i<Panjang_Pesan;i++){
                radio.stopListening();
                datatosend="";
                datatosend+=msg1.charAt(i);
                datatosend+="+";
                datatosend+=i;
                char hasil2[datatosend.length()+1];
```

```
                for(int
n=0;n<sizeof(hasil2);n++){hasil2[n]=datatosend.charAt(n);Ser
ial.print(datatosend.charAt(n));}


                if (!radio.write(hasil2,sizeof(hasil2))){
                  Serial.println("failed");
                  i--;
                  Serial.println("Retrying");
                 }
                else{
                  Serial.print(i);
                  Serial.print(". Kirim Paket : ");
                  Serial.print(hasil2);
                  Serial.println("");
                  boolean timeout = false;
                  timeout = perhitungan_timeout();
                  if(timeout){i--;Serial.println("ACK  PAKET
TIMEOUT !!!!");Serial.println("Resending Paket");}
                  else{
                    char data2[32]={0};
                    radio.read(&data2,32);
                    Serial.println(data2);
                    String ack;
                    for(int n=0;n<sizeof(data2);n++)
                        {ack+=data2[n];}


                    if(ack.toInt()==i){Serial.println("ACK
OK");}
                    else{
                      Serial.println("RESEND PAKET");
                      i--;
                      }
                    }
                 }
              }
            else{Serial.println("BUKAN ACK CRC");}
```

```
            }
          }
        }
        else{Serial.println("Bukan ACK Start");}
      }
    }
    delay(1000);
}
```

Lampiran Program *Reveiver*

```
#include <nRF24L01.h>
#include <RF24.h>
#include <RF24_config.h>
#include <SPI.h>

RF24 radio(9,10);
byte addresses[][6] = {"1Node","2Node"};

String crc88(String msg1){
  byte a[8];

  int Panjang_Pesan=msg1.length();
  Serial.println("");  Serial.print("Jumlah Karakter : ");
  Serial.print(Panjang_Pesan);
  Serial.println("");
  char msg[Panjang_Pesan];
  for(int i=0;i<Panjang_Pesan;i++){
     msg[i]=msg1.charAt(i);
    }

  /*Konversi Pesan Ke dalam Bit untuk di CRC kan*/
  byte data[Panjang_Pesan*8];
  int k=0;
  for(int i=0; i<Panjang_Pesan; i++){
     char a = msg[i];
      for(int j=7; j>=0; j--){
        data[k]=bitRead(a,j);
        k++;
      }
  }
  Serial.print("BINARY : ");
  for(int i=0;i<Panjang_Pesan*8;i++){
    Serial.print(data[i]);
    }
  byte peng_xor[9]={1,0,0,0,0,0,1,1,1};
```

```
byte coba;
int r=0,rr;
int panjang_data=sizeof(data);
for(int i=0;i<panjang_data;i++){
  if(data[i]==B1){
    r=0;
    for(int u=sizeof(data)-1;u<sizeof(data);u--){
      r++;
      if(data[u]==B1){
        rr = r;
        }
      }
    if(rr>=sizeof(peng_xor)){
      for(int j=0;j<9;j++){
        coba = peng_xor[j]^data[i+j];
        data[i+j]=coba;
        }
      }
    else{
      Serial.println("");
      Serial.print("CRC8 : ");
      int uu=0;
      for(int u=i;u<panjang_data;u++){
          Serial.print(data[u]);
          a[uu]=data[u];
          uu++;
        }
        break;
      }
    }
  }


Serial.println("");

k=0;
```

```
  int hasil=0;
  int total=1;
  byte hex[2];
  int index_hex=1;
  for(int i=sizeof(a)-1;i>=0;i--){
    if(a[i]==1){
      if(k==0){total=1;}
      else{
        total=1;
        for(int b=1;b<=k;b++){
          total=total*2;
          }
        }

      }
    else{
      total=0;
      }

    hasil=hasil+total;
   if(k==3){
      k=-1;
      hex[index_hex]=hasil;
      index_hex--;
      hasil=0;
      }
    k++;

  }
  String datanya;
  datanya+=(String(hex[0],HEX));
  datanya+=(String(hex[1],HEX));
  return datanya;
}
```

```
String pemisah(String msg1, String tandanya){
  int Panjang_Pesan=msg1.length();
  char msg[Panjang_Pesan];
  for(int i=0;i<Panjang_Pesan;i++){
     msg[i]=msg1.charAt(i);
    }
  char pesan[3];
  char no_paket[2];
  int pembatas=0;
  for(int i=0;i<Panjang_Pesan;i++){
    if(msg[i]=='+'){
        pembatas=i;
      }

    }
  if(pembatas==0){return "no";}
  int n=0;

  for(int i=pembatas+1;i<Panjang_Pesan;i++){
      no_paket[n]=msg[i];
      n++;
    }
  for(int i=0;i<pembatas;i++){
      pesan[i]=msg[i];
    }

  if (tandanya=="crc"){
        String data1;
        for(int i=0;i<pembatas;i++){
          data1+=pesan[i];
        }

    return data1;

  }
```

```
  else{
        String data1;
        for(int i=0;i<sizeof(no_paket);i++){
          data1+=no_paket[i];
         }
  return data1;}
}


void setup(void){
  Serial.begin(9600);
  radio.begin();
  radio.openWritingPipe(addresses[0]);
  radio.openReadingPipe(1,addresses[1]);
  radio.startListening();
  }



void loop() {
  Serial.println("-----------------------------------------
-----");
  long waktu;
  while(!radio.available()){
    waktu=millis();
  }

  char text1[32]={0};
  radio.read(&text1, sizeof(text1));
  Serial.println(text1);
  String data1;
      for(int i=0;i<=4;i++){
          data1+=text1[i];
        }
  if(data1=="start"){
    delay(40);
    radio.stopListening();
```

```
radio.write("start", 6);

radio.startListening();
while(!radio.available()){
  waktu=millis();
}

radio.read(&text1, sizeof(text1));
Serial.println(text1);
String data1;
for(int i=0;i<5;i++){
        data1+=text1[i];
      }
//Serial.println(data1);
String crc8;
crc8=pemisah(data1,"crc");
if(crc8!="no"){
    Serial.println("");
    Serial.print(crc8);
    String jml_pkt;
    jml_pkt=pemisah(data1,"cr");
    Serial.println("");
    Serial.print(jml_pkt);
    delay(40);
    radio.stopListening();
    radio.write("CRC", 6);
    delay(40);
    Serial.println("");
    char datanya_terakhir[jml_pkt.toInt()];
    Serial.println(sizeof(datanya_terakhir));
    for(int i=0;i<jml_pkt.toInt();i++){
      radio.startListening();
      while(!radio.available()){
        waktu=millis();
      }
```

```
          radio.read(&text1, sizeof(text1));
          String no_pkt;
          String data_pkt;
          no_pkt=pemisah(text1,"cr");
          if(no_pkt=="no"){break;}
          data_pkt=pemisah(text1,"crc");

datanya_terakhir[no_pkt.toInt()]=data_pkt.charAt(0);
          Serial.println(text1);
          delay(40);
          radio.stopListening();
          char da[no_pkt.length()+1];
          for(int
n=0;n<sizeof(da);n++){da[n]=no_pkt.charAt(n);}
          radio.write(da, sizeof(da));
          i=no_pkt.toInt();
          delay(40);
        }
        Serial.println(sizeof(datanya_terakhir));
        String data_final;
        for(int i=0;i<sizeof(datanya_terakhir);i++){
            data_final+=datanya_terakhir[i];
          }
        Serial.println(data_final);
        Serial.println(data_final.length());
        String crc8_rcv;
        crc8_rcv=crc88(data_final);
        Serial.println(crc8_rcv);
        if(crc8_rcv==crc8){Serial.println("DATA     TERKIRIM
DENGAN BAIK");}
        else{Serial.println("MAAF, DATA ANDA TIDAK TERKIRIM
DENGAN BAIK ");}

        radio.startListening();
        }
      }
}
```