IMPLEMENTASI ALGORITMA MD5 PADA ARDUINO MEGA

SKRIPSI

Laboraturium Sistem Komputer Dan Robotika

Untuk Memenuhi Persyaratan Memperoleh Gelar Sarjana Komputer

Disusun Oleh:

KRESTIAN APRILIYANTO

NIM. 0910683058

KEMENTRIAN PENDIDIKAN DAN KEBUDAYAAN

UNIVERSITAS BRAWIJAYA

PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER

PROGRAM STUDI INFORMATIKA/ILMU KOMPUTER

MALANG

2014

LEMBAR PERSETUJUAN

IMPLEMENTASI ALGORITMA MD5 PADA ARDUINO MEGA

SKRIPSI

Diajukan Untuk Memenuhi Persyaratan Memperoleh Gelar Sarjana Komputer

Disusun Oleh:

KRESTIAN APRILIYANTO

NIM. 0910683058

Telah diperiksa dan disetujui :

Pembimbing I	**Pembimbing II**
Gembong Edhi Setyawan, ST.,MT.	**Adharul Muttaqin, ST.,MT.**
NIK.76120116110373	**NIP. 19760121 200501 1 001**

LEMBAR PENGESAHAN

IMPLEMENTASI ALGORITMA MD5 PADA ARDUINO MEGA

Disusun Oleh :

KRESTIAN APRILIYANTO

NIM. 0910683058

Skripsi ini telah diuji dan dinyatakan lulus pada tanggal 30 Mei 2014

Penguji I	Penguji II
Eko Setiawan,ST.,M.eng.	**Kasyful Amron, ST,M.Sc**
NIK. 870610 06 1 1 0256	**NIP. 19750803 2003 12 100 3**

Penguji III

Agung Setia Budi, ST,MT

NIK.

Mengetahui

Ketua Program Studi Teknik Informatika/Ilmu Komputer

Drs. Marji, MT.

NIP. 19670801 199203 1 001

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Tuhan Yang Mahaesa karena berkat Rahmat-Nya, penulis dapat menyelesaikan proposal skripsi yang berjudul "**IMPLEMENTASI ALGORITMA MD5 PADA ARDUINO MEGA**".

Penyusunan skripsi ini bertujuan untuk memenuhi sebagian persyaratan memperoleh gelar sarjana komputer. Tak lupa penulis mengucapkan rasa terima kasih kepada :

1. Gembong Edhi Setyawan.ST.,MT. dan Adharul Muttaqin.ST.,MT. selaku dosen pembimbing selama pelaksanaan skripsi.
2. Ir. Sutrisno, M.T, Ir. Heru Nurwasito, M.Kom., Himawat Aryadita, S.T, M.Sc., dan Edy Santoso, S.Si., M.Kom., selaku Ketua, Wakil Ketua 1, Wakil Ketua 2 dan Wakil Ketua 3 Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya.
3. Ibunda Sri Hartini, Ayahanda Suewarno, dan seluruh keluarga atas segenap dukungan dan kasih sayang yang telah diberikan.
4. Drs. Marji, M.T dan Issa Arwani, S.Kom, M.Sc selaku Ketua dan Sekretaris Program Studi Teknik Informatika Universitas Brawijaya.
5. Seluruh Dosen Teknik Informatika Universitas Brawijaya atas kesediaan membagi ilmunya kepada penulis.
6. Seluruh Civitas Akademika Teknik Informatika Universitas Brawijaya yang telah banyak memberi bantuan dan dukungan selama penulis menempuh studi di Teknik Informatika Universitas Brawijaya dan selama penyelesaian skripsi ini.
7. Teman-teman yang telah memberikan bantuan baik moril maupun spiritual.
8. Seluruh pihak yang telah membantu kelancaran penulisan skripsi yang tidak dapat saya sebutkan satu persatu.

Yang telah membantu dalam menyelesaikan pembuatan skripsi dan berkat bimbingan beliau kendala ataupun hambatan dalam penyusunan skripsi ini dapat terselesaikan.

Penulis menyadari bahwa skripsi ini jauh dari sempurna. Untuk itu penulis mengharapkan kritik dan saran agar penulis dapat melakukan perbaikan terhadap skripsi yang disusun ini. Semoga skripsi ini dapat memberikan manfaat. Amiin.

Malang, Mei 2014

Penulis

ABSTRAK

Krestian Apriliyanto 2014. : IMPLEMENTASI ALGORITMA MD5 PADA ARDUINO MEGA. Program Studi Teknik Informatika, Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya, Malang.

Dosen Pembimbing : Gembong Edhi Setyawan., S.T., M.T Dan Adharul Muttaqin.ST.,MT.

Banyak pengembang aplikasi dan perangkat komunikasi belum menyertakan sistem keamanan bagi datanya, sehingga data menjadi kurang aman saat dikirimkan karena memang tidak disertai dengan sistem keamanan, khususnya untuk menjaga integritas data. Skripsi ini menyampaikan ide bagaimana memanfaatkan Arduino sebagai perangkat tambahan yang dapat memberikan sistem keamanan pada aplikasi dan perangkat komunikasi data dengan mengunakan algoritma MD5.

Arduino yang sudah dilengkapi MD5 menerima data dari sistem lain dan mengembalikan hasil berupa hash data tersebut. Pengujian dilakukan dalam 2 tahap, yaitu waktu eksekusi proses *hash* MD5 dan validasi data. Pengujian waktu eksekusi proses *hash* memberikan hasil 0,001360 s sampai 0,001396 s dalam satu blok dan waktu eksekusi dipengaruhi oleh panjang karakter. Pengujian validasi memberikan hasil 100% benar dalam menghasilkan *hash* dari suatu data teks. Dari penelitian ini didapatkan kesimpulan bahwa Arduino dapat digunakan sebagai perangkat tambahan untuk memberikan sistem keamanan data pada alat telekomunikasi.

Kata Kunci: Data, Keamanan, *Hash*

ABATRACT

Krestian Apriliyanto 2014: IMPLEMENTATION OF MD5 ALGORITHM ON ARDUINO MEGA. ***Informatics Degree Program, Information Technology and Computer Science Program, Brawijaya University, Malang*****.**

Advisor **: Gembong Edhi Setyawan.ST.,MT and Adharul Muttaqin.ST.,MT.**

Many application and device communication developers does not yet include a security system for its data, thus the data becomes less secure when transferred because the device does not have a security system, particular for data integrity. This final project aims to utilize the Arduino as an enhancement that can provide security in application systems and telecommunications devices by using MD5 algorithm.

System test was conducted in two phases, the first was the MD5 hash process execution time and the second was data validation. Testing process execution time hash results to 0.001396 s 0.001360 s for one block and the execution time is influenced by the length of the character. Validation testing provide 100% correct result in generating a hash of a text data. From this research it was concluded that the Arduino can be used as an enhancement to provide data security systems in telecommunications equipment.

Keywords: *Data, Security, Hash*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

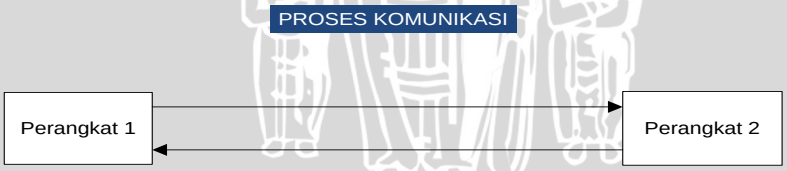
DAFTAR LAMPIRAN

DAFTAR SOURCECODE

BAB I
PENDAHULUAN

1.1 Latar Belakang

Manusia membutuhkan proses komunikasi dari satu pihak ke pihak lain untuk menyampaikan sebuah informasi. Komunikasi merupakan salah satu kebutuhan sehari-hari yang sangat penting karena semua informasi dapat dengan cepat disampaikan dengan menggunakan perangkat komunikasi. Banyak dari perangkat telekomunikasi yang menawarkan penggunaan produknya dengan media nirkabel atau kabel. Pada akhir-akhir ini orang akan lebih memilih cara komunikasi yang mudah dan aman misalnya menggunakan aplikasi seperti chat, sms, email, perlengkapan *smart home*, dan robotik karena cepat dan *realtime*. Keamanan pada komputer melingkupi empat aspek, yaitu *privacy, integrity, authentication,* dan *availability* [RAH-02]. Banyak para pengembang aplikasi yang berlomba-lomba menawarkan keunggulan aplikasinya pada pengguna karena kebutuhan komunikasi sangat dibutuhkan dimanapun. Faktor keamanan dari data yang akan dikirimkan jarang dipikirkan oleh para pengembang aplikasi bahkan banyak aplikasi yang tidak menyertakan sistem keamanan pada datanya. Proses komunikasi akan ditunjukkan pada gambar 1.1.



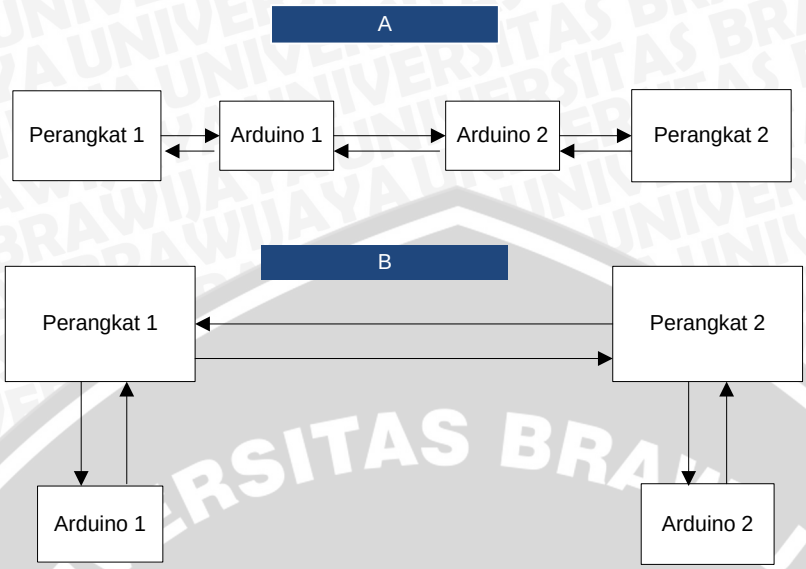
Gambar 1.1 Proses komunikasi 2 buah perangkat

Pada gambar 1.1 merupakan proses komunikasi 2 buah perangkat secara umum dengan menggunakan media kabel atau nirkabel. Perangkat tersebut ada yang menyertakan sistem keamanan pada aplikasinya dan ada yang tidak menyertakan sistem keamanan. Keamanan pada data merupakan hal yang sangat penting pada saat perangkat tersebut saling berkomunikasi dan melakukan pengiriman data. Ada diantara para pengembang aplikasi yang memberikan sistem keamanan pada datanya yang langsung diberikan pada

aplikasi tersebut dan ada banyak para pengembang aplikasi yang tidak menyertakan keamanan pada datanya.

Banyak penelitian yang telah dilakukan dalam keamanan data dan akan terus berkembang. Christof, dkk (3:2006) menerapkan keamanan *embedded* di dalam mobil, sistem ini selain bertujuan untuk mengamankan komunikasi *automotive* berdasarkan pada mekanisme kriptografi modern yang menyediakan kerahasiaan, pencegahan manipulasi, dan *autentikasi* untuk mengatasi banyak permasalahan dalam kendaraan [PAA-06]. Andre, dkk (2006) mengusulkan sebuah prosedur untuk pengiriman *software* yang aman dalam proses instalasi pada *embedded system*. Dengan solusi menggabungkan beberapa teknik kriptografi dalam hal ini menggunakan *Public Key Encryption Broadcast* (PKBE) yang memungkinkan distribusi *software* yang aman dari setiap penyedia layanan yang terkait. Pada sistem ini mengatur pengaksesan yang benar dan fleksibel untuk fungsionalitas perorangan dari *software* yang diinstal [ADE-06].

Berdasarkan dari latar belakang dari penelitian yang telah dilakukan yang berfokus pada sistem keamanan pada aplikasi, maka penulis mengusulkan ide dengan mengimplemantasikan algoritma MD5 pada *embedded system* arduino sebagai pendukung aplikasi yang tidak memiliki sistem keamanan. Arduino dipilih karena dengan spesifikasi yang sangat sederhana dibandingkan dengan Raspberry PI. *Embedded System* merupakan sistem yang terdiri dari perangkat keras dan perangkat lunak yang dirancang untuk memecahkan sebuah aplikasi yang spesifik. Banyak aplikasi yang terdiri dari komputer sederhana berupa *hardware* dan *software* [CEV-02]. Sedangkan untuk mengamankan data teks menggunakan algoritma MD5 karena dapat digunakan sebagai integritas dari sebuah data. Algoritma MD5 merupakan fungsi *hash* satu-arah yang dibuat oleh Ron Rivest. Algoritma MD5 menerima masukan berupa pesan dengan ukuran sembarang dan menghasilkan *message digest* yang panjangnya 128 bit [MUN-06]. Pada penerapan *system embedded* dapat dilakukan dengan 2 cara, proses penerapan akan ditunjukkan pada gambar 1.2.



Gambar 1.2 Proses Penerapan Arduino Sebagai Perangkat Pendukung

Pada gambar 1.2 A menunjukkan bahwa 2 buah perangkat yang akan melakukan proses komunikasi dan tidak memiliki sistem keamanan, dan menggunakan arduino yang telah diimplementasikan dengan suatu sistem keamanan. Pada proses ini aplikasi akan merubah konsep jaringan asli dari perangkat tersebut. Pada gambar 1.2 B merupakan proses penerapan perangkat yang akan melakukan proses komunikasi dan tidak merubah konsep jaringan dari perangkat tersebut. Pada proses ini menggunakan soket untuk komunikasi antar peranghkat 1 dan perangkat 2 dan pada arduino 1 dan arduino 2 menggunakan Ajax dengan tipe data JSONP.

Pada penelitian ini membahas tentang keaslian data yang dikirimkan dan menggunakan penerapan 1.2 B dengan cara membandingkan nilai *hash* dari sebuah data teks. Dengan menggunakan perangkat tambahan ini kedepannya dapat diaplikasikan ke perangkat yang tidak memiliki sistem keamanan misalnya pada *smart home* dan *gadget*. Dari penelitian ini, diharapkan perangkat yang saling berkomunikasi terjamin *validitas* pada datanya pada saat saling bertukar data dan *embedded system* sebagai perangkat pengolahan data yang dimanfaatkan sebagai sistem keamanan.

1.2 Rumusan Masalah

Berdasarkan uraian latar belakang di atas, maka dapat dirumuskan permasalah pada penelitian ini yaitu sebagai berikut:

1. Bagaimana proses implementasi *hash* data teks dengan algoritma MD5 pada arduino.
2. Apakah panjang dari sebuah data teks mempengaruhi proses *system* dan berapa tingkat persentasenya
3. Bagaimana pengimplementasian algoritma MD5 pada arduino
4. Bagaimana proses pengujian validitas dan waktu eksekusi *hash* data text dengan algoritma MD5 pada arduino

1.3 Batasan Masalah

Agar permasalahan yang dirumuskan dapat lebih terfokus, maka pada penelitian ini dibatasi oleh hal-hal berikut :

1. Proses *hash* MD5 data text diimplementasikan di arduino.
2. Penelitian ini difokuskan pada proses *hash MD5*.
3. Algoritma yang digunakan adalah MD5.
4. Aplikasi *chat* digunkan sebagai contoh implementsi dari algoritma MD5.

1.4 Tujuan

Penulisan skripsi ini mengimplementasikan *hash* data teks dengan algoritma MD5 pada arduino untuk menjaga integritas dan validitas data teks.

1.5 Manfaat

Manfaat yang bisa di dapat dari penelitian ini adalah :

1. Pembaca mendapatkan wawasan tentang Algoritma MD5.
2. Sebagai referensi dalam mengembangkan dan melanjutkan riset yang memiliki keterkaitan dengan penelitian keamanan data.

1.6 Sistematika Penulisan

Sistematika penulisan penelitian untuk memberikan gambaran dan uraian dari penyusunan tugas akhir secara garis besar yang meliputi beberapa bab, antara lain :

BAB I Pendahuluan

Bab I Pendahuluan menguraikan latar belakang skripsi, rumusan masalah, tujuan dan manfaat skripsi serta sistematika penyusunan laporan skripsi.

BAB II Tinjauan Pustaka

Bab II menguraikan teori-teori penunjang yang menjadi referensi dalam pelaksanaan skripsi.

BAB III Metode Penelitian dan Perancangan

Bab III menguraikan tentang metode dan langkah kerja yang dilakukan dalam proses perancangan dan implementasi pada pelaksanaan skripsi dan perancangan sistem yang menjadi objek studi kasus skripsi.

BAB IV Implementasi

Bab IV menguraikan proses implementasi dari dasar teori yang telah dipelajari sesuai analisis dan perancangan sistem.

BAB V Pengujian dan Analisis

Bab V memuat hasil pengujian dan analisis terhadap sistem yang telah direalisasikan.

BAB VI Penutup

Bab VI memuat kesimpulan yang diperoleh dari pembuatan dan pengujian program, serta saran-saran untuk pengembangan lebih lanjut.

BAB II
KAJIAN PUSTAKA DAN DASAR TEORI

Bab ini berisi pembahasan tentang kajian pustaka dan dasar teori yang diperlukan untuk penelitian. Kajian pustaka membahas penelitian yang telah ada dan diusulkan. Dasar teori membahas teori yang diperlukan untuk menyusun penelitian yang diusulkan.

2.1 Kajian Pustaka

Kajian pustaka membahas penelitian yang telah ada dan diusulkan. Pada penelitian ini kajian pustaka diambil dari beberapa penelitian yang relevan yang pernah dilakukan. Eko Sediyono, dkk (2013) melakukan riset tentang "*Secure Login by Using One-time Password Authentication Based on MD5 Hash Encrypted SMS"* Riset ini berisi tentang mengembangkan prosedur *login* yang aman dalam mengakses Sistem Informasi Akademikberbasis *Web*. Kesimpulan dari riset ini adalah *system* akan memberikan waktu tiga menit untuk keamanan *login* dengan berbasis SMS OTP. Kendala tersebut mempersempit waktu bagi *hacker* untuk menyadap dan menyusup. Waktu *delay* ini merupakan rata-rata yang diperoleh dari *survey* di antara beberapa penyedia layanan di Indonesia. Hasil dari riset ini mengenai perbandingan beberapa operator dalam waktu pengiriman dan waktu aktif dalam aplikasi dapat dilihat pada gambar 2.1[SED-13].

TABLE I. COMPARISON OTP DELIVER TIME AMONGS PHONECELL OPERATORS

No	times trial	average deliver time (s.ms)	Phone cell Operators
1	5	23.15	Telkomsel/As
2	5	25.80	Indosat/m3
3	5	16.52	Indosat/mentari
4	5	24.45	smartfren
5	5	24.08	Telkomsel/Simpati
6	5	19.98	XL

TABLE II. COMPARISON OTP ACTIVE TIME AMONGS APPLICATION

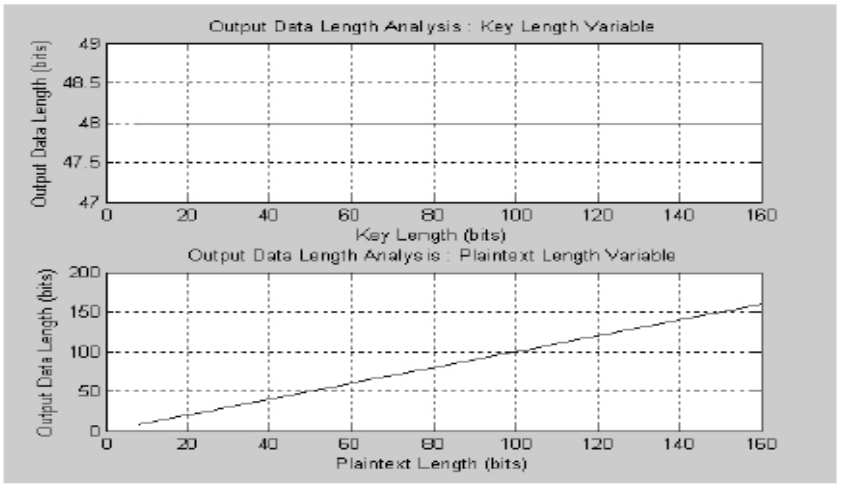
No	Application	Number of code digit	OTP active time	OTP / Token functionality
1	OTP with MD5 Hash	6	3 min	User authentication to enter the system after username and password
2	Facebook.com	6	20 min	Username and password recovery. OTP sent via e-mail to reset password
3	Google.com	6	20 min	OTP sent via e-mail to verify new account

Gambar 2.1 Hasil Riset 1

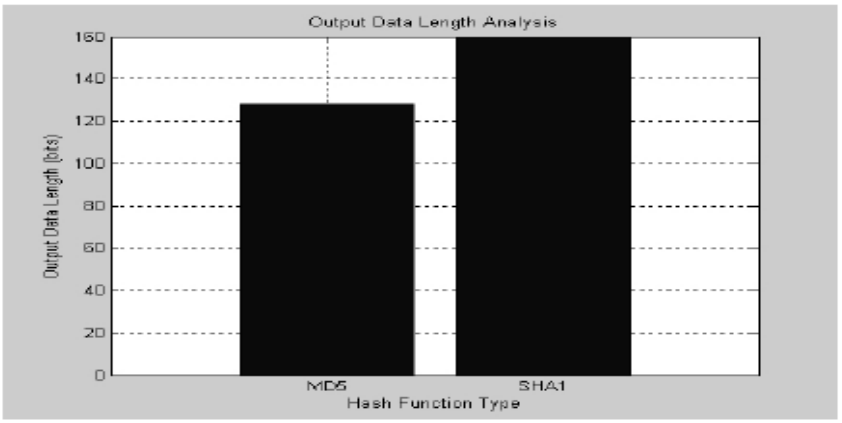
[Sumber: SED-13]

Pada Tabel 1. tabel ini menunjukkan bahwa diantara operator ponsel yang digunakan dalam percobaan, rata-rata waktu untuk mengirimkan OTP kepengguna tidak lebih dari setengah menit. Table 2 menunjukkan waktu OTP aktif antara aplikasi [SED-13]. Sri Esti, dkk (2010) melakukan riset tentang "Teknik *Digital Watermarking* untuk perlindungan *Citra Digital BITMAP IMAGE* dengan metode LSB dan MD5". Pada riset ini menerapkan *digital watermarking* untuk menyisipkan label hak cipta kedalam citra digital secara *invisible*. Hal ini mempersulit pihak lain yang tidak berwenang untuk menghapus label hak cipta sehingga hanya pihak berwenang saja yang mampu mengungkap label hak cipta. Tujuan dari

penelitian ini adalah menciptakan aplikasi *software* yang mengimplementasikan metode LSB dan MD5 untuk melindungi hak cipta dan otentikasi citra digital diam [SRI-10]. Saipul Bahri, dkk (2012) melakukan Riset tentang "IMPLEMENTASI PENGAMANAN BASIS DATA MENGGUNAKAN METODE ENKRIPSI MD5". Riset ini berisi tentang implementasi pengamanan data dari sisi kandungan data yang tersimpan pada basis data. Dengan kata lain data yang tersimpan pada data base adalah hasil *hash* dari *password* sebelum diproses dengan algoritma MD5 [BAH-12]. Angger Ardyanto, dkk (2006) melakukan Riset tentang "IMPLEMENTASI ALGORITMA SISTEM KRIPTOGRAFI MD5, SHA1, DAN RC4 PADA APLIKASI MOBILE INTERNET BERBASIS JAVA". Riset ini berisi tentang implementasi algoritma MD5, SHA1, dan RC4. Pengamanan data dari sisi kandungan data yang tersimpan pada basis data. Dengan kata lain data yang tersimpan pada data base adalah hasil *hash* dari *password* sebelum diproses dengan algoritma MD5. Semua algoritma tersebut diadaptasi ke dalam sistem yang menggunakan *platform* J2ME. Sistem yang telah diimplementasikan dapat berjalan baik pada *emulator* J2ME *Wireless Toolkit*. Analisa terhadap subsistem *security* dilakukan berdasarkan beberapa parameter, yaitu panjang data *output*, penggunaan *memory*, waktu proses, distribusi frekuensi kemunculan karakter, variansi distribusi, *avallanche effect* untuk algoritma RC4, dan waktu untuk melakukan *brute force attack* pada algoritma RC4. Dari hasil analisa terhadap parameter-parameter diatas diperoleh bahwa algoritma SHA1 memiliki keunggulan pada parameter panjang data *output*, waktu proses dengan kombinasi algoritma RC4, distribusi frekuensi, variansi, dan waktu *brute force attack* [PAM-06]. Hasil dari riset ini dilihat pada gambar dibawah ini:

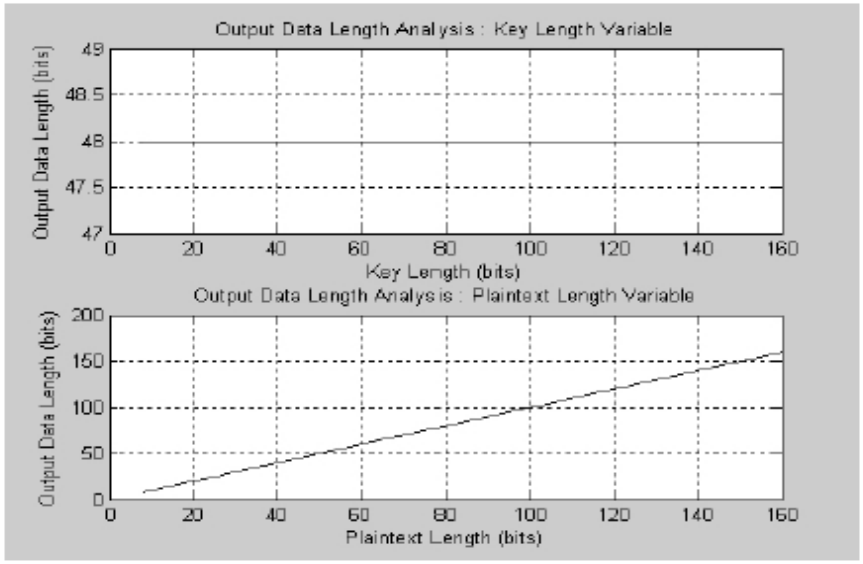


Gambar 4.a. Panjang Data Output Algoritma RC4

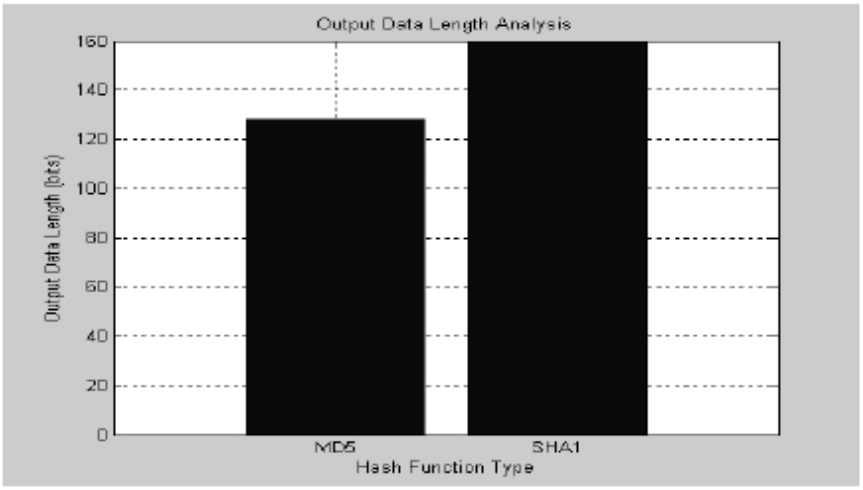


Gambar 4.b. Panjang Data Output *Hash function*

Gambar 2.2 Hasil Analisis Output Data



Gambar 4.a. Panjang Data Output Algoritma RC4

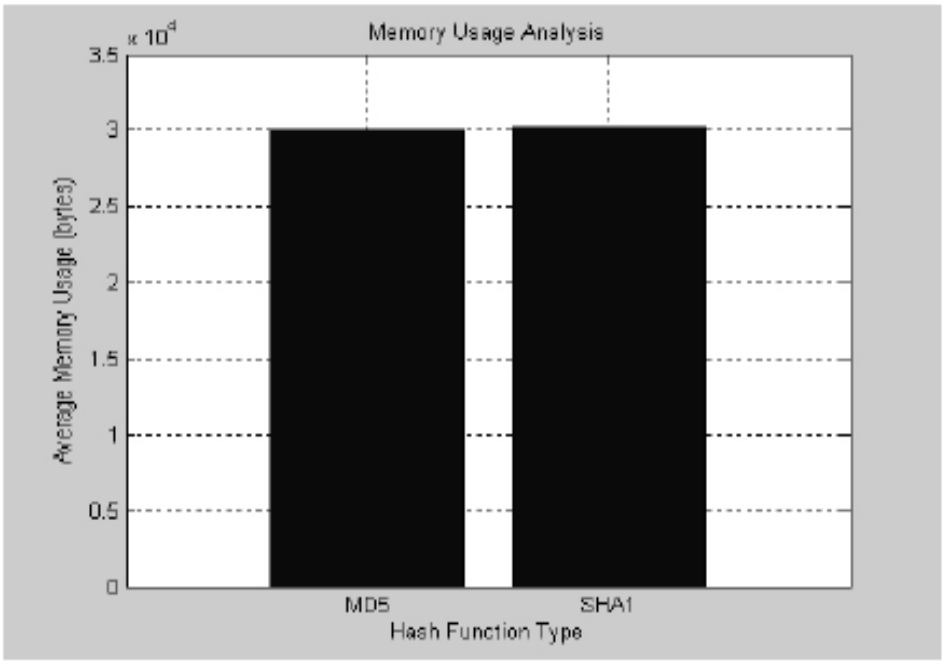


Gambar 4.b. Panjang Data Output *Hash function*

Gambar 2.3 Hasil Pengujian Penggunaan Memory

[Sumber: PAM-06]

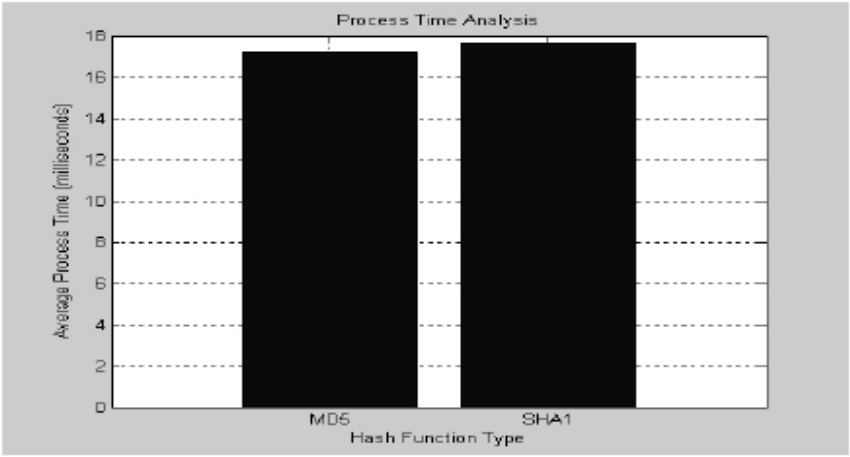
Dari Gambar 2.3 4.a Terlihat bahwa panjang data *output* dari algoritma RC4 tidak tergantung dari panjang kunci, tetapi dari panjang *plaintext sedangkan* Gambar 2.3 4.b. Terlihat bahwa panjang data *output* hasil *hash* MD5 adalah sepanjang 128 *bit*, dan hasil hash SHA1 adalah 160 *bit* [PAM-06].



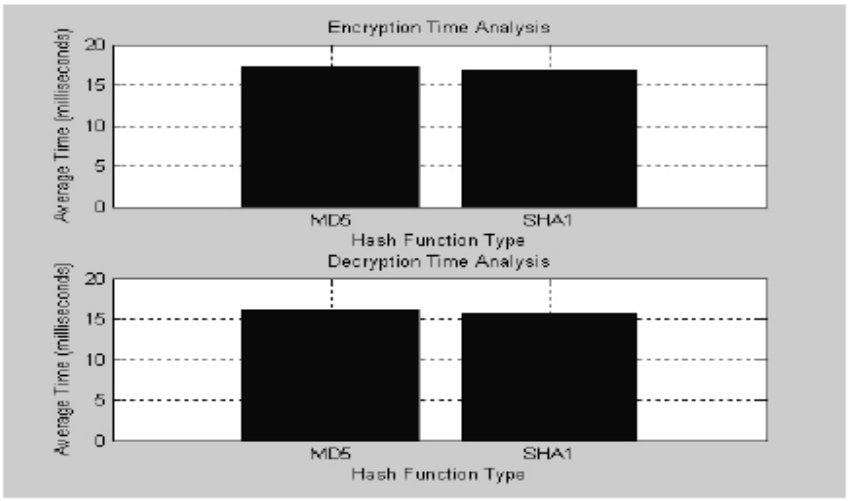
Gambar 2.4 Hasil Pengujian Penggunaan Memory

[Sumber: PAM-06]

Dari Gambar 2.4 terlihat bahwa penggunaan *memory* pada algoritma RC4 dengan *strengthened key* MD5 adalah 30060 *byte* dari total *memory* yang tersedia 500000 *byte*, atau sekitar 6,012 % dan dengan *strengthened key* SHA1 adalah 30267,6 *byte*, atau sekitar 6,053 %. Perbedaan penggunaan *memory* untuk kedua algoritma MD5 dan SHA1 adalah sekitar 0,041 %, suatu ukuran yang relatif sangat kecil [PAM-06].



Gambar 6.a. Waktu Proses *Hash function*

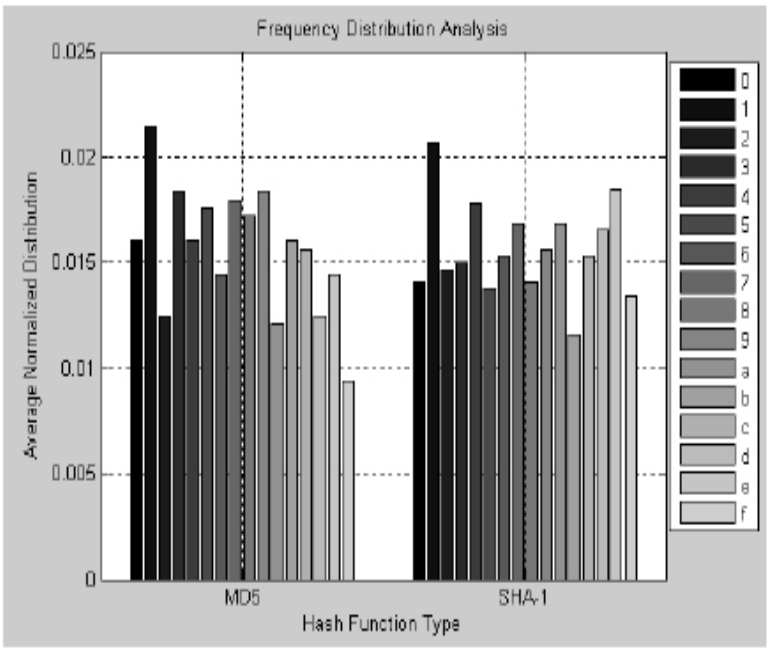


Gambar 6.b. Waktu Proses RC4

Gambar 2.5 Hasil Pengujian Waktu Eksekusi

[Sumber: PAM-06]

Gambar 2.5 6.a dan 6.b hasil pengujian waktu proses *hash function* dan waktu proses enkripsi dekripsi yang melibatkan *strenthened key*.



Gambar 2.6 Hasil Pengujian Distribusi Frekuensi
[Sumber: PAM-06]

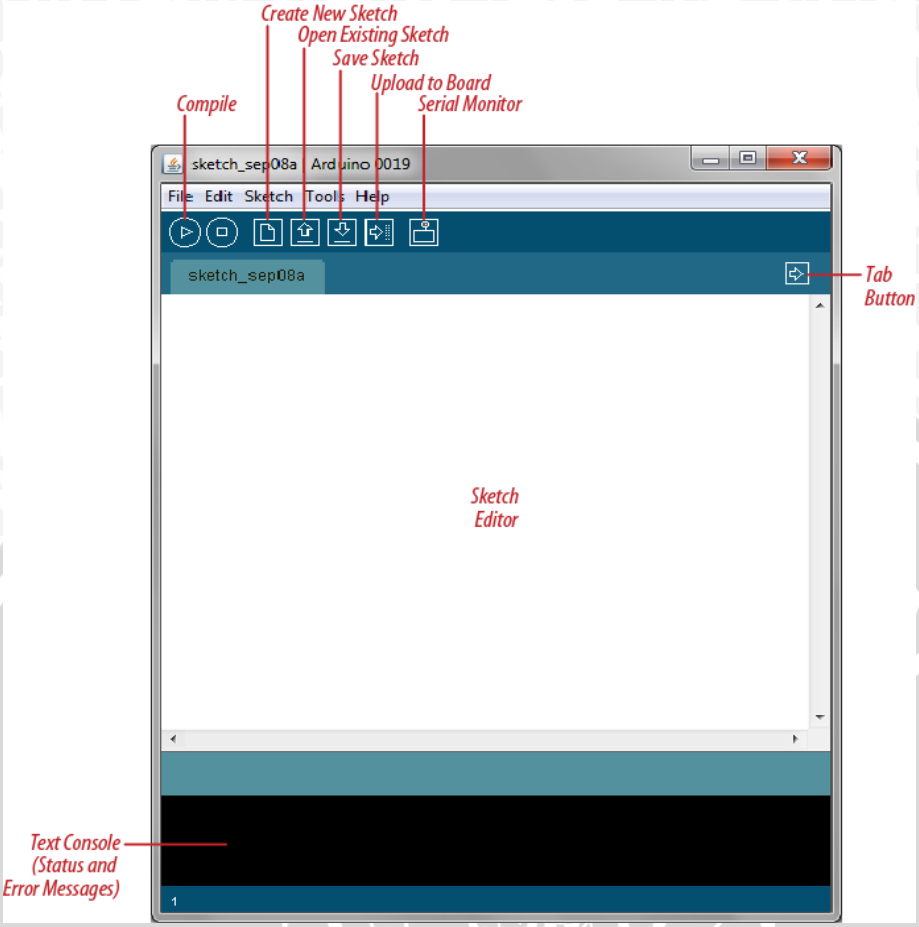
Gambar 2.6 hasil pengujian distribusi frekuensi terlihat perbedaan yang kecil antara distribusi frekuensi pada MD5 dan SHA1. Dari beberapa contoh hasil penelitian diatas, maka dapat disimpulkan beberapa persamaan dan perbedaannya. Persamaan penelitian ini dengan hasil-hasil penelitian sebelumnya adalah pada penggunaan algoritma MD5 sebagai sistem keamanan yaitu validitas dan integritas data. Pada penelitian sebelumya sistem keamanan diberikan bersamaan dengan perancangan aplikasi tersebut, jadi aplikasi tersebut telah memiliki sistem keamanan. Sedangkan, perbedaan antara penelitian ini dengan hasil-hasil penelitian sebelumnya adalah pada sisi implementasi sistem keamanan. Pada penelitian sebelumnya algoritma MD5 diimplementasikan bersamaan dengan aplikasi yang dibuat sedangkan pada penelitian ini difokuskan untuk aplikasi atau perangkat-perangkat yang tidak memiliki sistem keamanan. Dengan menggunakan perangkat *embedded* arduino yang telah diimplementasikan algoritma MD5 dengan tujuan kedepannya digunakan sebagai perangkat pendukung aplikasi atau sistem yang tidak memiliki keamanan.

2.2 Embedded System dan Arduino

Embedded system adalah sebuah sistem terdiri dari perangkat keras dan perangkat lunak yang dirancang untuk memecahkan aplikasi spesifik. Banyak aplikasi yang terdiri lebih dari komputer sederhana pada *hardware* dan *software*. Sebagai contoh, sistem pemantau yang dirancang untuk mengontrol lengan robot terdiri dari komputer, kamera, layar, dan lengan robot. Masing-masing komponen ini merupakan embedded system sendiri-sendiri. Embedded sistem telah berevolusi selama bertahun-tahun dari yang sederhana dan mandiri, sistem tunggal yang bertujuan untuk sepenuhnya terintegrasi, dan sistem web. Pesatnya perubahan teknologi dan bertambahnya persyaratan telah menyebabkan pengembang untuk mengambil pendekatan baru untuk desain dan pengembangan sistem tersebut [CEV-02].

Arduino adalah suatu alat yang dirancang dengan mudah untuk pemula yang tidak memiliki pengalaman. Arduino dapat digunakan untuk membuat berbagai hal yang menarik seperti peralatan musik, robot. Arduino didefinisikan sebagai sebuah *platform* elektronik yang *open source,* berbasis pada *software* dan *hardware* yang mudah digunakan, ditujukan untuk para seniman desainer dan setiap orang yang tertarik dalam membuat objek atau lingkungan yang instruktif [MAR-11].

Pemrograman pada arduino menggunakan bahasa C basic dengan Struktur dasar yang terdiri dari dua bagian *setup*( ) bagian untuk inisialisasi yang hanya dijalankan sekali di awal program, sedangkan *loop*() untuk mengeksekusi bagian program yang akan dijalankan berulang-ulang sampai arduino dimatikan [MAR-11]. Proses dilakukan dengan menggunakan Arduino IDE yang merupakan sebuah editor yang digunakan untuk menulis program, meng*compile*, dan mengunggah ke papan Arduino. Arduino *development environment* terdiri dari editor teks untuk menulis kode, area pesan, *console* teks, *toolbar* dengan tombol-tombol untuk fungsi umum, dan sederetan menu. Tampilan IDE Arduino dapat dilihat pada gambar 2.7 berikut



Gambar 2.7 Arduino IDE
Sumber:[MAR-11]

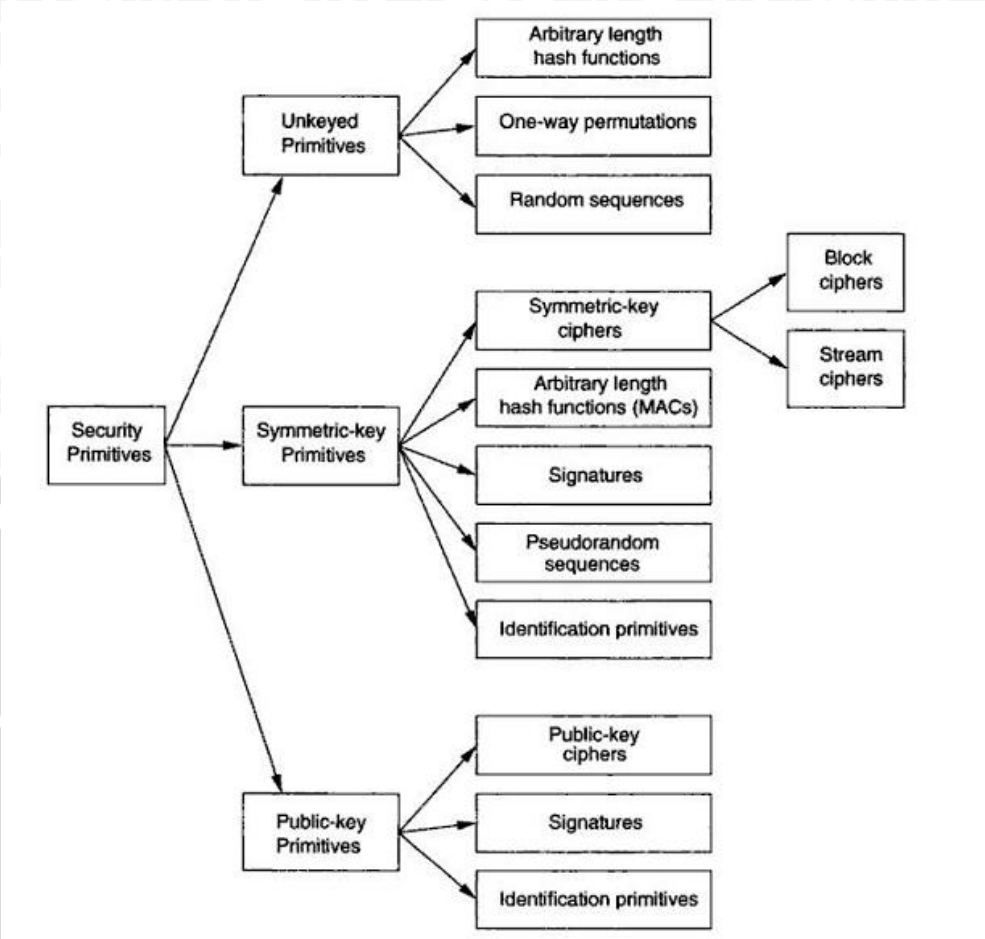
2.3 Kriptografi

Kriptografi adalah ilmu yang mempelajari teknik matematika yang berhubungan dengan aspek keamanan informasi seperti kerahasiaan data, keabsahan data, integritas data, serta autentikasi data. Kriptografi bukanlah sarana untuk memberikan informasi keamanan, melainkan sekumpulan teknik, berikut ini merupakan 4 tujuan dari kriptografi [MEN-10]:

1. Kerahasiaan merupakan layanan yang digunakan untuk teteap menjaga isi informasi dari semua pihak kecuali pihak yang memiliki otoritas, kerahasiaan adalah istilah yang identik dengan privasi dan tingkat kerahasiaan. Ada banyak pendekatan untuk kerahasiaan data, mulai dari algoritma matematika yang membuat data tidak dapat dimengerti

2. Integritas data adalah layanan yang membahas perubahan data yang tidak sah. Untuk menjamin integritas data, seseorang harus memiliki kemampuan untuk mendeteksi manipulasi data oleh pihak yang tidak berhak. Manipulasi data mencakup hal-hal seperti penyisipan, penghapusan, dan perubahan.
3. Autentikasi adalah layanan yang berhubungan dengan identifikasi. Fungsi ini berlaku untuk kedua entitas dan informasi itu sendiri. Dua pihak yang saling berkomunikasi harus mengidentifikasi setiap informasi lain yang disampaikan melalui saluran harus dikonfirmasi pengirim ke penerima, tanggal asal, isi data, waktu dikirim dan lain-lain. Untuk alasan ini aspek kriptografi ini biasanya dibagi menjadi dua kelas utama: keaslihan entitas dan keaslihan data asal (karena jika pesan dimodifikasi, sumber telah berubah).
4. *non-repudiation* (menolak penyangkalan) adalah layanan yang mencegah entitas untuk menyangkal komitmen atau tindakan sebelumnya. Ketika perselisihan timbul karena suatu entitas menyangkal bahwa tindakan tertentu diambil.

Tujuan dari kriptografi adalah untuk mengatasi empat tujuan tersebut baik dalam teori dan praktek, kriptografi merupakan teknik pencegahan dari deteksi kecurangan dan kegiatan berbahaya lainnya.



Gambar 2.8 Blok Diagram Kriptografi
Sumber: [MEN-10]

2.4 Fungsi Hash

Salah satu dasar dalam kriptografi modern adalah fungsi *hash* kriptografi, atau yang sering disebut fungsi *hash* satu arah (*one-way hash*). Fungsi *hash* adalah komputasi efisien dari fungsi pemetaan *string* biner dengan panjang sembarang untuk *string* biner dari beberapa panjang tetap yang disebut *hash-value* [MEN-10]. Fungsi *hash* dapat menerima masukan *string* yang bebas. Jika *string* menyatakan pesan (*message*), maka sembarang pesan M berukuran bebas dikompresi oleh fungsi *hash* H melalui persamaan:

$h = H(M)$ ..... (2.1)

merupakan persamaan dimana *h* adalah nilai *hash* atau *message digest* dari fungsi *H* untuk masukan *M*.

2.6.1 Fungsi Hash Satu Arah (*One-Way Hash*)

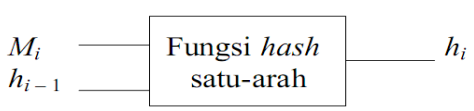
Fungsi *hash* satu-arah adalah fungsi *hash* yang bekerja dalam satu arah, pesan yang sudah diubah menjadi *message digest* tidak dapat dikembalikan lagi menjadi pesan semula. Sifat-sifat fungsi *hash* satu arah sebagai berikut:

Fungsi H dapat diterapkan pada blok data berukuran berapa saja

1. H menghasilkan nilai *(h)* dengan panjang yang tetap *(fixed-leght output*)
2. *H(x)* mudah untuk dihitung untuk nilai x yang diberikan
3. Untuk setiap nilai *h* yang dihasilkan, tidak mungkin dikembalikan kemali ke nilai *x* sedemikian sehingga *h(x)=h*. Begitulah sebabnya fungsi *H* dikatakan sebagai fungsi satu arah *(one-way hash)*
4. Untuk setiap *x* yang diberikan, tidak mungkin mencari $y \neq x$ sedemikian sehingga *H(y)=H(x)*
5. Tidak mungkin mencari pasangan x dan y sedemikian sehingga *H(x)=H(y)*
6. Masukan fungsi *hash* adalah block pesan *(M)* dan keluaran dari *hashing* blok pesan sebelumnya

$$h_i = H(M_i, h_{i-1}) \quad (2.2)$$

Fungsi *hash* adalah publik (tidak dirahasiakan), dan keamannya terletak pada sifat satu arahnya. Nilai *hash* dari masukan pertama bersama blok pesan berikutnya akan menjadi masukan berikutnya bagi fungsi pemampatan. Nilai *hash* keseluruhan adalah nilai hash dari blok paling akhir. *Pre-image* sedapatnya mengandung beberapa *binary* yang menggambarkan panjang dari masukan pesan.



Gambar 2.9 Fungsi Hash Satu Arah

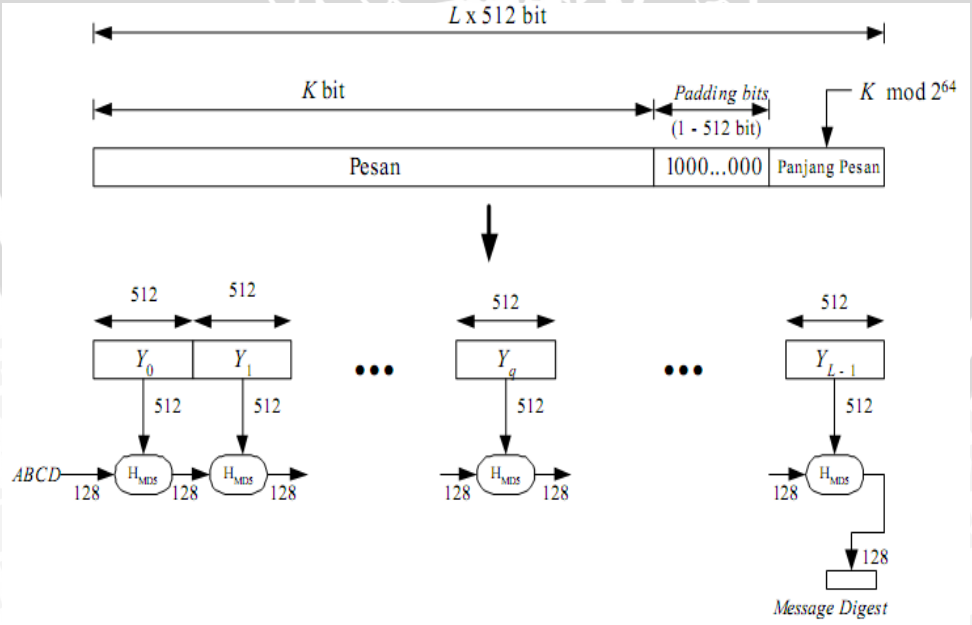
Sumber: [MUN-2006]

2.5 Algoritma MD5

Algoritma MD5 didefinisikan dalam RFC 1321, menerima masukan berupa pesan dengan ukuran sembarang dan menghasilkan *message digest* yang panjangnya 128 bit. Pesan masukan diproses dalam 512-bit yang dapat dibagi dalam 16 subblok 32-bit. Hasil dari message *digest* adalah gabungan dari empat blok 32-bit yang membentuk 128-bit [RHE-03]. MD5 ini dirancang untuk menjadi cepat dan ringkas, algoritma MD5 adalah perluasan dari algoritma *message-digest* MD4 oleh *Ronald L. Rivest* dari MIT. MD5 sedikit lebih lambat dari MD4, tetapi lebih konservatif dalam desain, mengorbankan sedikit kecepatan untuk jauh lebih besar dalam tingkat keamanan.

2.6 Spesifikasi Algoritma MD5

MD5 memiliki empat proses dengan menambahkan bit-bit pengganjal (*padding*), penambahan nilai panjang semula, inisialisasi penyangga (*buffer*) MD, dan pengolahan pesan dalam blok berukuran 512-bit. Untuk lebih jelasnya dapat dilihat pada gambar 2.10 secara umum proses MD5



Gambar 2.10 Proses MD5 Secara Umum

Sumber : [MUN-06]

2.6.1 Penambahan Bit-Bit Pengganjal

Pesan ditambah dengan sejumlah bit pengganjal sedemikian sehingga panjang pesan (dalam satuan bit) kongruen dengan 448 modulo 512. Berarti panjang pesan setelah ditambah bit-bit pengganjal adalah 64 bit kurang dari kelipatan 512 (448-bit). Angka 512 ini muncul karena MD5 memperoses pesan dalam blok-blok yang berukuran 512-bit. Pesan dengan panjang 448 bit akan tetap ditambah dengan bit-bit pengganjal. Jika panjang pesan 448 bit, maka pesan tersebut ditambah dengan 512 bit menjadi 960 bit. Jadi, panjang bit-bit pengganjal adalah antara 1 sampai 512. Bit-bit pengganjal terdiri dari sebuah bit 1 pertama diikuti dengan sisanya bit 0.

2.6.2 Penambahan Nilai Panjang Pesan Semula

Pesan yang telah diberi bit-bit pengganjal selanjutnya ditambah lagi dengan 64 bit yang menyatakan panjang pesan semula. Jika panjang pesan $>\mathbf{2^{64}}$ maka yang diambil adalah panjangnya dalam modulo $\mathbf{2^{64}}$. Dengan kata lain, jika panjang pesan semula adalah K bit, maka 64 bit yang ditambahkan menyatakan K modulo $\mathbf{2^{64}}$. Setelah ditambahkan dengan 64 bit yang menyatakan nilai panjang pesan maka jumlah pesan menjadi 512-bit.

2.6.3 Inisialisai Penyangga MD (*buffer)*

MD5 membutuhkan 4 buah penyangga *(buffer*) yang masing-masing panjangnya 32-bit. Total panjang penyangga adalah 4 x 32 = 128-bit, keempat penyangga ini menampung hasil antara dan hasil akhir. Penyangga ini diberi nama A, B, C, dan D dan setiap penyangga diinisialisasi dengan nilai -nilai (dalam notasi HEX) sebagai berikut:

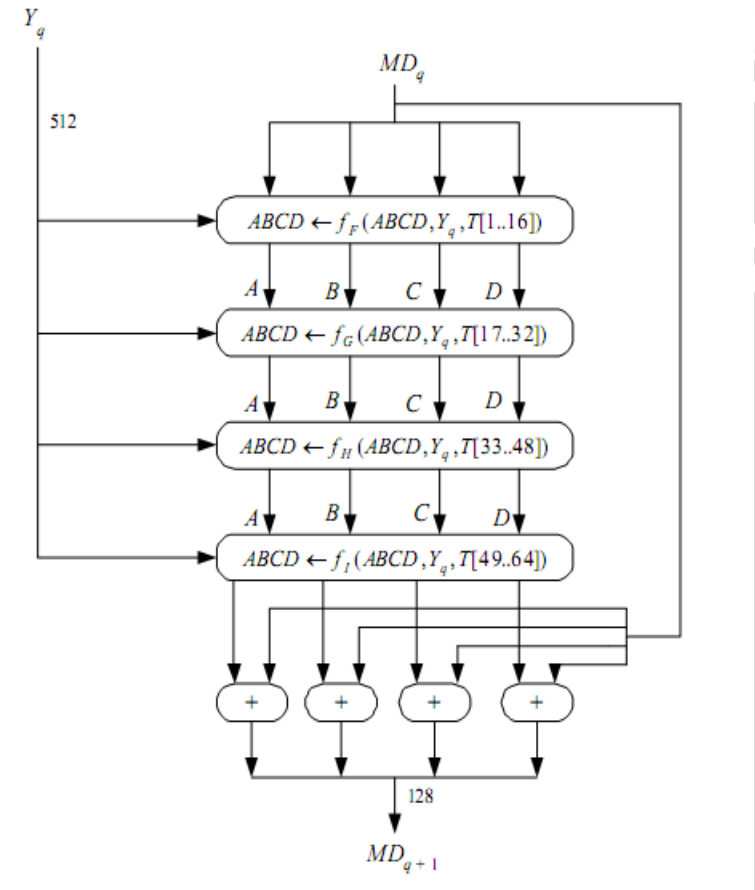
$$A = 01234567$$
$$B = 89ABCDEF$$
$$C = FEDCBA98$$
$$D = 76543210$$

Gambar 2.11 Inisilisasi Nilai Penyangga

Sumber : [MUN-06]

2.6.4 Pengolahan Pesan Dalam Blok 512-Bit

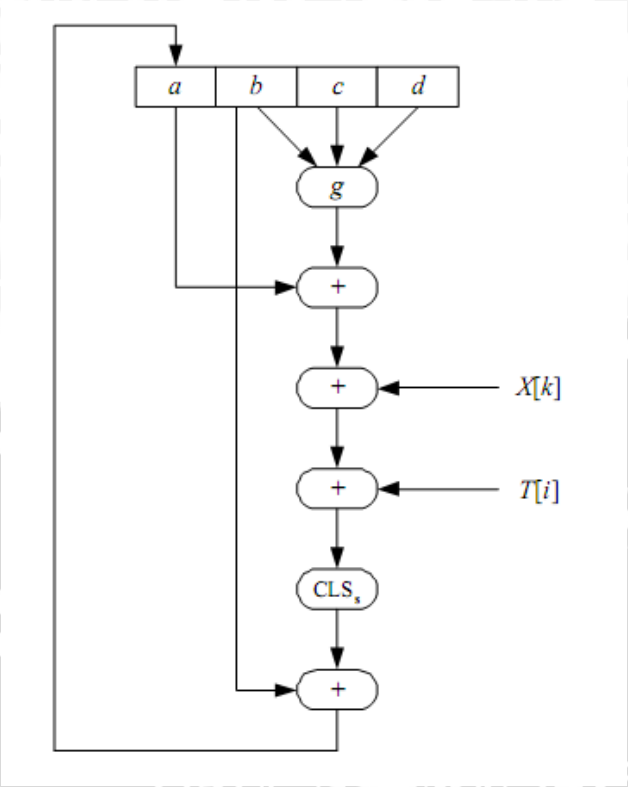
Pesan dibagi menjadi L buah blok yang masing-masing panjangnya 512 bit ($\boldsymbol{Y_0}$sampai $\boldsymbol{Y_{L-1}}$). Setiap blok 512-bit diproses bersama dengan penyangga MD menjadi keluaran 128-bit, dan ini disebut proses $\boldsymbol{H_{md5}}$.



Gambar 2.12 Pengolahan Blok Berukuran 512-bit

Sumber : [MUR-06]

Pada Gambar 2.12 Y_q menyatakan blok 512-bit ke-$_q$ dari pesan yang telah ditambah bit-bit pengganjal dan tambahan 64 bit nilai panjang pesan semula. MD_q adalah nilai message digest 128-bit dari proses $\boldsymbol{H_{md5}}$ ke-$_q$. Pada awal proses, MD_q berisi nilai inisialisasi penyangga MD Proses $\boldsymbol{H_{md5}}$ terdiri dari 4 buah putaran, dan masing-masing putaran melakukan operasi dasar MD5 sebanyak 16 kali dan setiap operasi dasar memakai sebuah elemen T. Jadi setiap putaran memakai 16 elemen.



Gambar 2.13 Operasi Dasar MD5

Sumber : [MUN-06]

Pada gambar 2.13 merupakan Operasi dasar MD5 dapat dituliskan persamaan

$a = b + (a + g(b, c, d) + X[k] + T[i]) <<<s)$ ..... (2.3).

Dimana penjelasan dari persamaan diatas ditunjukkan pada table 2.3

Tabel 2.1. Penjelasan Persamaan (2.3)

Sumber : [MUR-2006]

a, b, c, d	empat buah peubah penyangga 32-bit (berisi nilai penyangga A, B, C, D)
G	salah satu fungsi F, G, H, I
X[k]	kelompok32-bit ke-k dari blok 512 bit message ke-q. Nilai k = 0 sampai 15.
T[i]	elemen Tabel T ke-i (32 bit)
+	operasi penjumlahan modulo 2
<<<s	melambangkan operasi rotasi left shift ke-s

Dari persamaan diatas maka dapat digambarkan sebagai berikut:

1. Putaran pertama, 16 kali operasi dasar dengan g(b, c, d) = F(b, c, d)

No.	[abcd	k	s	i]
1	[ABCD	0	7	1]
2	[DABC	1	12	2]
3	[CDAB	2	17	3]
4	[BCDA	3	22	4]
5	[ABCD	4	7	5]
6	[DABC	5	12	6]
7	[CDAB	6	17	7]
8	[BCDA	7	22	8]
9	[ABCD	8	7	9]
10	[DABC	9	12	10]
11	[CDAB	10	17	11]
12	[BCDA	11	22	12]
13	[ABCD	12	7	13]
14	[DABC	13	12	14]
15	[CDAB	14	17	15]
16	[BCDA	15	22	16]

Gambar 2.14 Operasi dasar Putaran Pertama
Sumber : [MUN-06]

2. Putaran Kedua, 16 kali operasi dasar dengan g(b, c, d) = G(b, c, d)

No.	[abcd	k	s	i]
1	[ABCD	1	5	17]
2	[DABC	6	9	18]
3	[CDAB	11	14	19]
4	[BCDA	0	20	20]
5	[ABCD	5	5	21]
6	[DABC	10	9	22]
7	[CDAB	15	14	23]
8	[BCDA	4	20	24]
9	[ABCD	9	5	25]
10	[DABC	14	9	26]
11	[CDAB	3	14	27]
12	[BCDA	8	20	28]
13	[ABCD	13	5	29]
14	[DABC	2	9	30]
15	[CDAB	7	14	31]
16	[BCDA	12	20	32]

Gambar 2.15 Operasi dasar Putaran Kedua
Sumber : [MUN-06]

3. Putaran Ketiga, 16 kali operasi dasar dengan g(b, c, d) = H(b, c, d)

No.	[abcd	k	s	i]
1	[ABCD	5	4	33]
2	[DABC	8	11	34]
3	[CDAB	11	16	35]
4	[BCDA	14	23	36]
5	[ABCD	1	4	37]
6	[DABC	4	11	38]
7	[CDAB	7	16	39]
8	[BCDA	10	23	40]
9	[ABCD	13	4	41]
10	[DABC	0	11	42]
11	[CDAB	3	16	43]
12	[BCDA	6	23	44]
13	[ABCD	9	4	45]
14	[DABC	12	11	46]
15	[CDAB	15	16	47]
16	[BCDA	2	23	48]

Gambar 2.16 Operasi dasar Putaran Ketiga
Sumber : [MUN-06]

4. Putaran Keempat, 16 kali operasi dasar dengan g(b, c, d) = I(b, c, d)

No.	[abcd	k	s	i]
1	[ABCD	0	6	49]
2	[DABC	7	10	50]
3	[CDAB	14	15	51]
4	[BCDA	5	21	52]
5	[ABCD	12	6	53]
6	[DABC	3	10	54]
7	[CDAB	10	15	55]
8	[BCDA	1	21	56]
9	[ABCD	8	6	57]
10	[DABC	15	10	58]
11	[CDAB	6	15	59]
12	[BCDA	13	21	60]
13	[ABCD	4	6	61]
14	[DABC	11	10	62]
15	[CDAB	2	15	63]
16	[BCDA	9	21	64]

Gambar 2.17 Operasi dasar Putaran Keempat

Sumber : [MUN-06]

Pada MD5 juga terdapat empat buah fungsi nonlinear yang masing-masing digunakan pada tiap operasinya (satu fungsi untuk satu blok)

Nama	Notasi	$g(b, c, d)$
f_F	$F(b, c, d)$	$(b \wedge c) \vee (\sim b \wedge d)$
f_G	$G(b, c, d)$	$(b \wedge d) \vee (c \wedge \sim d)$
f_H	$H(b, c, d)$	$b \oplus c \oplus d$
f_I	$I(b, c, d)$	$c \oplus (b \wedge \sim d)$

Gambar 2.18 Fungsi Dasar MD5

Sumber : [MUN-06]

Pada MD5 terdapat fungsi Nilai *T[i]* yang disusun oleh fungsi 2^{32} *x abs(sin(i)),* i dalam radian.

T[1] = D76AA478	T[17] = F61E2562	T[33] = FFFA3942	T[49] = F4292244
T[2] = E8C7B756	T[18] = C040B340	T[34] = 8771F681	T[50] = 432AFF97
T[3] = 242070DB	T[19] = 265E5A51	T[35] = 69D96122	T[51] = AB9423A7
T[4] = C1BDCEEE	T[20] = E9B6C7AA	T[36] = FDE5380C	T[52] = FC93A039
T[5] = F57C0FAF	T[21] = D62F105D	T[37] = A4BEEA44	T[53] = 655B59C3
T[6] = 4787C62A	T[22] = 02441453	T[38] = 4BDECFA9	T[54] = 8F0CCC92
T[7] = A8304613	T[23] = D8A1E681	T[39] = F6BB4B60	T[55] = FFEFF47D
T[8] = FD469501	T[24] = E7D3FBCB	T[40] = BEBFBC70	T[56] = 85845DD1
T[9] = 698098D8	T[25] = 21E1CDE6	T[41] = 289B7EC6	T[57] = 6FA87E4F
T[10] = 8B44F7AF	T[26] = C33707D6	T[42] = EAA127FA	T[58] = FE2CE6E0
T[11] = FFFF5BB1	T[27] = F4D50D87	T[43] = D4EF3085	T[59] = A3014314
T[12] = 895CD7BE	T[28] = 455A14ED	T[44] = 04881D05	T[60] = 4E0811A1
T[13] = 6B901122	T[29] = A9E3E905	T[45] = D9D4D039	T[61] = F7537E82
T[14] = FD987193	T[30] = FCEFA3F8	T[46] = E6DB99E5	T[62] = BD3AF235
T[15] = A679438E	T[31] = 676F02D9	T[47] = 1FA27CF8	T[63] = 2AD7D2BB
T[16] = 49B40821	T[32] = 8D2A4C8A	T[48] = C4AC5665	T[64] = EB86D391

Gambar 2.19 Nilai T[i]

Sumber : [MUN-06]

Setelah putaran keempat, a, b, c, dan d ditambahkan ke A, B, C, dan D, dan selanjutnya algoritma memproses untuk blok data berikutnya *(Y_{q+1})*. Keluaran akhir dari algoritma MD5 adalah hasil penyambungan bit-bit di A, B, C, dan D.

2.7 PHP dan HTML

PHP adalah akronim dari *hypertext preprocessor*, yaitu suatu bahasa pemrograman berbasiskan kode-kode *(scrip)* yang digunakan untuk mengolah suatu data dan mengirimkannya kembali ke web browser menjadi kode HTML. Kode PHP mempunyai ciri-ciri khusus yaitu [OKT-10]:

1. Hanya dapat dijalankan menggunakan web server misalnya: Apache.
2. Kode PHP diletakkan dan dijalankan pada web browser
3. Kode PHP dapat digunakan untuk mengakses database, seperti: MySQL, PostgreSQL, Oracel, dan lain-lain
4. Merupakan *software* yang bersifat *open source* memiliki sifat *multiplatfom*, artinya dapat dijalankan menggunakan sistem Operasi apapun, seperti: linux, unix, windows, dan lain-lain.

HTML (*Hyper Text markup Language)* adalah bahasa pemrograman web yang memiliki sekumpulan simbol-simbol atau tag-tag yang dituliskan dalam sebuah file yang digunakan untuk menampilkan halaman pada web browser. Tag-tag HTML selalu diawali dengan <x> dan diakiri dengan </x> dimana x tag HTML seperti b, i, u dan lain-lain [ANH-10].

2.8 JSON

JSON (*JavaScript Object Notation*) adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (*generate*) oleh komputer. Format ini dibuat berdasarkan bagian dari Bahasa Pemprograman *JavaScript*, Standar ECMA-262 Edisi ke-3 - Desember 1999. JSON merupakan format teks yang tidak bergantung pada bahasa pemprograman apapun karena menggunakan gaya bahasa yang umum digunakan oleh programmer keluarga C termasuk C, C++, C#, Java, JavaScript, Perl, Python dll. Oleh karena sifat-sifat tersebut, menjadikan JSON ideal sebagai bahasa pertukaran-data.

JSON terbuat dari dua struktur [ORG-11]:

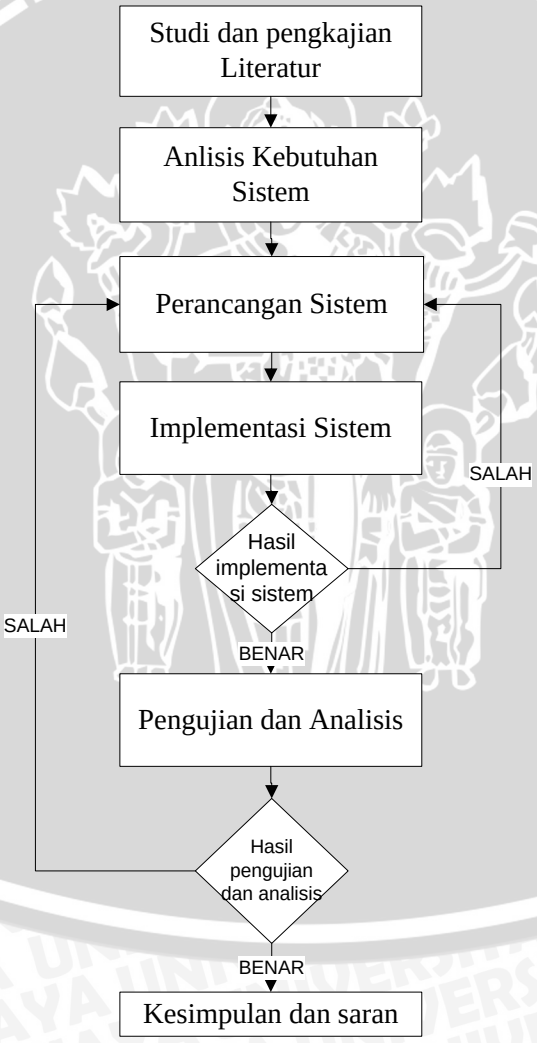
1. Kumpulan pasangan nama/nilai. Pada beberapa bahasa, hal ini dinyatakan sebagai objek (*object*), rekaman (*record*), struktur (*struct*), kamus (*dictionary*), tabel *hash* (*hash table*), daftar berkunci (*keyed list*), atau *associative array*.
2. Daftar nilai terurutkan (*an ordered list of values*). Pada kebanyakan bahasa, hal ini dinyatakan sebagai larik (*array*), vektor (*vector*), daftar (*list*), atau urutan (*sequence*).
3. Struktur-struktur data ini disebut sebagai struktur data *universal*. Pada dasarnya, semua bahasa pemprograman moderen mendukung struktur data ini dalam bentuk yang sama maupun berlainan. Hal ini pantas disebut demikian karena format data mudah dipertukarkan dengan bahasa-bahasa pemrograman yang juga berdasarkan pada struktur data ini.

2.9 Library

Library merupakan tempat dimana kita menyimpan dan mengatur simbol yang diciptakan di pemrograman C, dan juga file lainnya yang diimport, seperti algoritma, perintah-perintah.

BAB III
METODOLOGI PENELITIAN DAN PERANCANGAN

Bab ini menjelaskan langkah-langkah yang digunakan dalam penyusunan skripsi Implementasi Algoritma Md5 pada Arduino Mega dan menggunakan beberapa library yang ada pada *website* resmi arduino, meliputi library MD5 dan *Ethernet Shield*. Pada penelitian ini berisi studi dan pengkajian literatur, analisis kebutuhan sistem, perancangan sistem, implementasi sistem, pengujian dan analisis, kesimpulan dan saran.



Gambar 3.1 Flowchart Metodologi Penelitian

3.1 Studi Litelatur

Studi literatur menjelaskan dasar teori yang digunakan untuk menunjang penulisan skripsi teori pendukung tersebut di dapat dari buku, jurnal, paper dan internet. Literatur yang digunakan meliputi :

1. Kriptografi
   a. Fungsi hash
   b. Fungsi hash satu arah
2. Algoritma *MD5*
   a. Penambahan bit-bit pengganjal
   b. Penambahan nilai panjang pesan
   c. Inisialisasi penyangga MD (*buffer*)
   d. Pengolahan pesan dalam blok 512-bit
3. *Embedded System*
   a. *Arduino mega 2560*
   b. *Ethernet shield*
   c. *Keyboard input*

3.2 Analisa Kebutuhan

Analisis kebutuhan bertujuan untuk mendapatkan semua kebutuhan yang diperlukan oleh sistem yang akan dibangun dan di uji. Analisis kebutuhan dilakukan dengan mengidentifikasi kebutuhan sistem. Sesuai dengan tujuan penulisan ini yang difokuskan pada *embedded system* yaitu suatu sistem berbasis *mikroprosessor* yang dibangun untuk mengendalikan beberapa fungsi yang tidak dikerjakan oleh perangkat utama. Kebutuhan perangkat pada penelitian ini dibutuhkan *embedded system* yang dapat bekerja secara cepat dan efisien dalam memproses algoritma MD5, dalam penelitian ini menggunakan arduino sebagai perangkat *embedded system* karena arduino memiliki beberapa keunggulan yaitu:

1. *Open source* pada hardware dan softwarenya
2. Tidak memerlukan chip *programmer*, chip yang ada pada arduino sudah dilengkapi dengan *bootloader* yang akan menangani proses *upload* dari computer

3. Fasilitas *chip* yang cukup lengkap, pada arduino menggunakan *chip* AVR ATmega 168/328 yang memiliki fasilitas PWM, komunikasi serial, ADC, *timer*, *interrupt*, SPI dan I2C. Sehingga Arduino bisa digabungkan bersama modul atau alat lain dengan protokol yang berbeda-beda.
4. Ukuran kecil dan mudah dibawa, ukuran board Arduino cukup kecil, sehingga sangat mudah untuk dibawa.
5. Bahasa pemrograman yang relative, walaupun bahasa pemrograman pada arduino adalah bahasa C/C++, tetapi dengan penambahan library dan fungsi-fungsi standar membuat pemrogramman arduino lebih mudah dipelajari

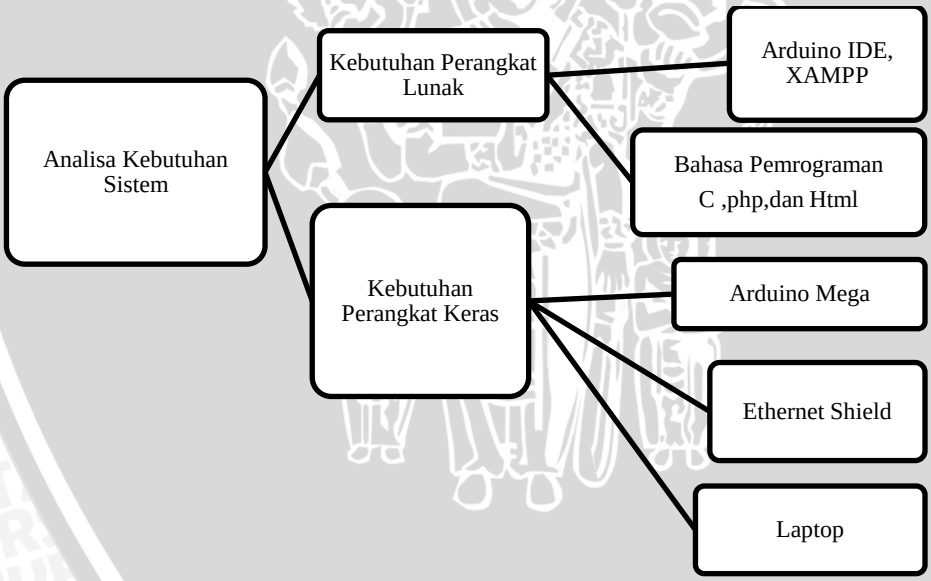
Pada penelitian ini, implementasi pada sistemnya dibutuhkan sebuah PC atau laptop sebagai contoh media pengimplemantasian aplikasi yang tidak memiliki *system* keamanan. Kebutuhan perangkat lunak digunakan sebagai implementasi program *hash* data text dengan algoritma MD5 kareana sifat satu arahnya dimanfaatkan sebagai validitas dari suatu pesan. Untuk desain dan perancangan *system* dibutuhkan *compiler* yang cocok dengan *hardware* yang digunakan yaitu IDE. Arduino IDE merupakan *software* bawaan dari Arduino dan nantinya digunakan untuk mengimportkan program kedalam Arduino. Pada kebutuhan perangkat lunak yang lainnya adalah bahasa pemrograman. Pada penelitian ini dibutuhkan *Web Browser* untuk menampilkan hasil dari *hash* data teks. Pada aplikasi chat, juga dibutuhkan *port* yang menghubungkan *Keyboard* yang nantinya berfungsi sebagai nilai *input*. Bahasa pemrograman yang digunakan sebagai *interface* dari implementasi algoritma MD5 pada arduino adalah PHP dan HTML, *system* yang dibuat nantinya memiliki *multiple* behavior dan *multipleplatform* dalam sebuah *interface*.

Pada penelitian ini dibutuhkan juga aplikasi *chat* dipilih karena banyak pengguna yang menggunakanya. Proses komunikasi dapat dilakukan secara *realtime*, sebagai contoh pengimplementasian dan pengujian dari algoritma MD5 pada arduino untuk menampilkan hasil proses dari *system*. Analisis

kebutuhan tersebut meliputi, kebutuhan perangkat lunak dan kebutuhan perangkat keras yang memiliki spesifikasi sebagai berikut:

a. Laptop compaq Cq41 Performa: Intel (R) Core (TM) i3 M350 @2,27 GHz
b. Sistem operasi : Windows 7 Ultimate 32-bit (6.1, Build 7600)
c. VGA card : ATI Mobility Radeon HD 5470
d. Arduino Mega 2560
e. *Ethernet Shield*
f. Impementasi algoritma MD5 dibangun dengan menggunakan compailer Arduino IDE
g. Aplikasi web server yang digunakan adalah *Apache* dan PHP yang ada dalam XAMPP-win32-1.8.3-2-VC11

dengan performa kebutuhan–kebutuhan tersebut maka dirancanglah diagram kebutuhan sistem seperti yang ditunjukkan pada Gambar 3.2

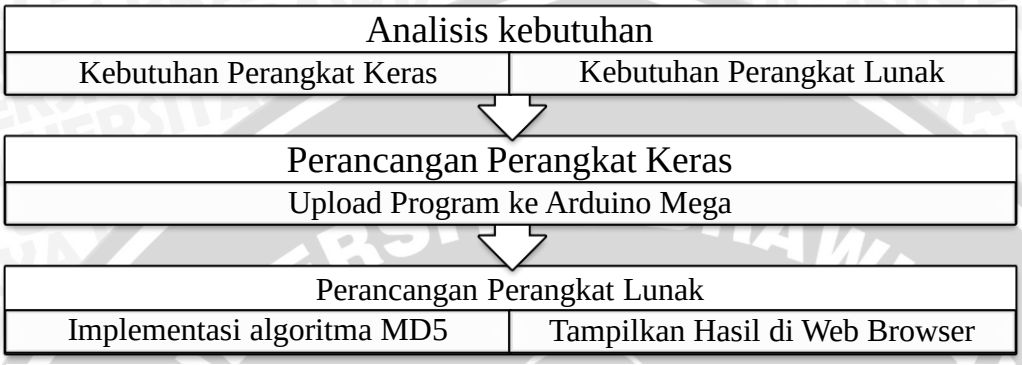


Gambar 3.2 Diagram Kebutuhan Sistem

3.3 Perancangan Sistem

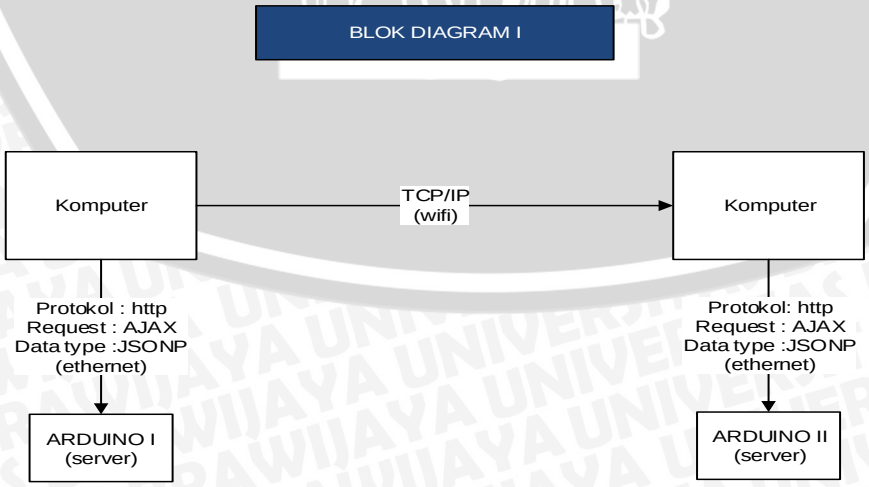
Perancangan sistem dilakukan setelah semua kebutuhan sistem didapatkan melalui tahap analisis kebutuhan yang meliputi kebutuhan perangkat lunak dan kebutuhan perangkat keras. Untuk mengembangkan sistem, tahap perancangan dilakukan dengan cara menganalisis kebutuhan

perangkat lunak dan merancang sistem yang sesuai dengan kebutuhan. Pada bagian ini akan dipaparkan mengenai langkah-langkah yang akan dijalankan untuk melakukan implementasi algoritma MD5 pada arduino dan aplikasi *chat* sebagai implementasi dan pengujian dari algoritma MD5 pada arduino, seperti yang di gambarkan dalam Gambar 3.3.



Gambar 3.3 Alur sistem secara umum

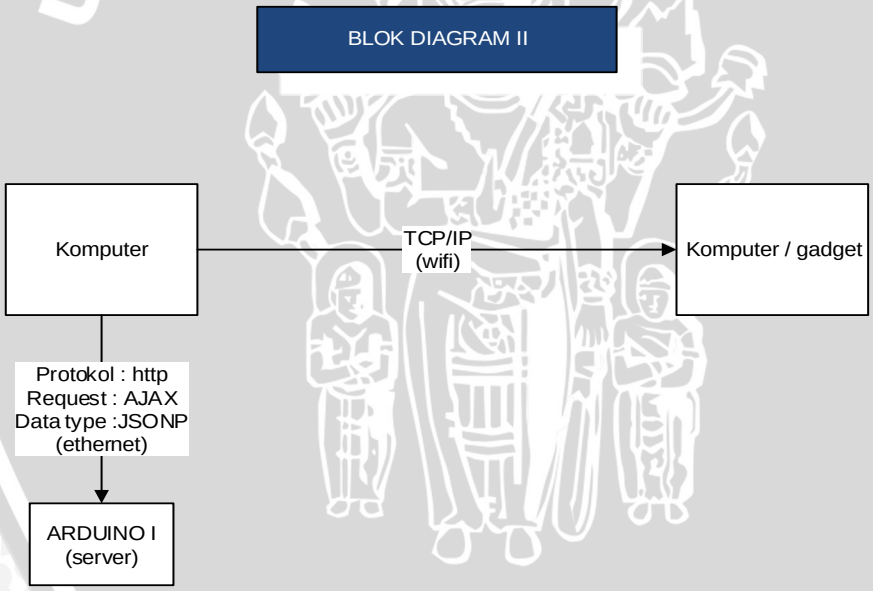
Pada implementasi *hash* dilakukan dengan algoritma *Message Digest Algorithm 5* (MD5). MD5 memproses teks masukan ke dalam blok-blok bit sebanyak 512-bit. Kemudian dibagi ke dalam 32-bit sub-blok sebanyak 16 buah, keluaran dari MD5 berupa 4 buah blok yang masing-masing 32 bit yang mana akan menjadi 128-bit yang bisa disebut nilai *hash* pesan dan akan ditampilkan pada *web browser*. Pada perancangan perangkat keras terdapat proses *upload* program ke arduino dan akan memproses data yang akan dikirimkan dengan algoritma MD5. Untuk lebih jelas dapat dilihat pada gambar blok diagram 3.4.



Gambar 3.4 Blok Diagram Perancangan I

Gambar 3.4 merupakan perancangan dengan menggunakan 2 buah arduino, pada sisi komputer 1 ke komputer 2 menggunakan TCP/IP dengan *wifi* sebagai proses komunikasi, sedangkan komputer 1 juga bertindak sebagai *server* 1 dan arduino 1 juga bertindak sebagai *server* A dengan menggunakan AJAX dan tipe data JSONP karena *cross domain*, *Ethernet* digunakan sebagai proses komunikasi. Sedangkan pada gambar 3.5 merupakan perancangan II dengan menggunakan 1 buah arduino.

Pada Arduino ditambahkan sekumpulan intruksi yang merupakan proses MD5 dan perangkat komunikasi. Dari hasil perancangan diharapkan kedepannya arduino yang sudah diimplementasikan dengan algoritma MD5 dapat diterapkan pada perangkat-perangkat *smart home* atau *robotik* yang khususnya tidak memiliki sistem keamanan.

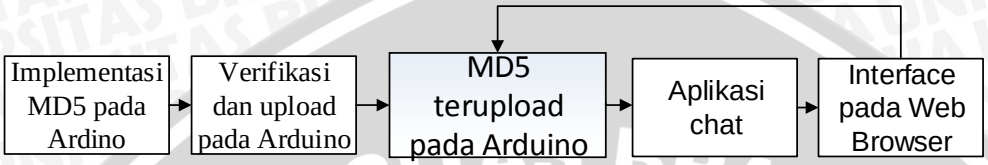


Gambar 3.5 Blok Diagram Perancangan II

3.4 Implementasi

Merupakan penjelasan mulai dari implementasi program sampai hasil akhir yang di tampilkan dalam *Web Browser*. Seperti yang di gambarkan dalam Gambar 3.6.

Untuk menguji apakah program yang ditanamkan sudah benar, selanjutnya diaktifkan aplikasi chat yang memanfaatkan Arduino untuk proses *hash* MD5. Hasi operasi MD5 dari Arduino selanjutnya dapat ditampilkan pada *Web Browser*. Pengujian dilakukan berulangkali sampai diperoleh hasil yang valid.



Gambar 3.6 Gambaran Secara Umum Implementasi *Hash* Data Teks Dengan Algoritma MD5

Untuk mengimplementasikan MD5 pada arduino terlebih dahulu dilakukan, *verifikasi dan upload* dengan menggunakan perangkat lunak pendukung Arduino IDE. Jika sukses, hasil kompilasi program dapat di *upload* ke Arduino.

3.5 Perancangan

3.5.1 Perancangan Proses Utama Chat

Langkah-langkah yang dilakukan untuk membentuk proses utama chat adalah:

1. Data teks sebagai nilai input
2. Dari data teks diubah menjadi bentuk *hash* data, proses ini dinamakan proses hash *chat*
3. Setelah proses *hash* selesai maka data asli, hasil proses hash, dan waktu dikirim ke penerima (*client* 2)
4. Pengirim melakukan pengecekan pada *hash* yang sudah diterima
5. Setelah proses pengecekan maka didapatkan data dan nilai *hash*nya
6. Proses utama *chat* selesai

Diagram alir proses utama chat dapat digambarkan pada Gambar 3.7.

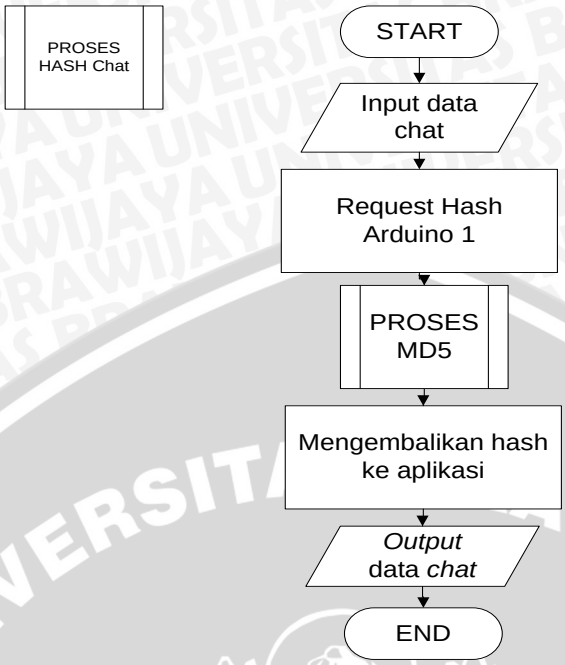
Gambar 3.7 Diagram Alir Proses Utama Chat

3.5.2 Perancangan Proses *Hash Chat*

Pada tahap proses *hash chat* langkah-langkahnya antara lain:

1. Data teks sebagai nilai input
2. Data teks dikirim ke Arduino 1 dari *client* 1
3. Data teks yang diterima arduino akan diproses menggunakan algoritma MD5 untuk menghasilkan *hash*
4. Setelah melalui proses MD5 akan meghasilkan *hash* dan waktu eksekusinya kemudian dikembalikan ke aplikasi *chat*
5. Hasil keluarannya berupa data teks, *hash*, dan waktu eksekusinya
6. Proses hash chat selesai

Diagram alir proses *hash chat* dapat digambarkan pada Gambar 3.8.



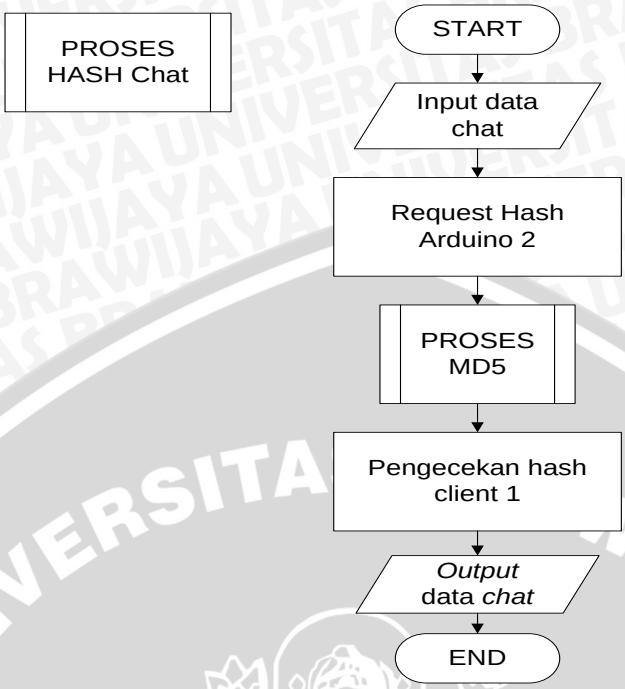
Gambar 3.8 Diagram Alir Proses *Hash Chat*

3.5.3 Perancangan Proses Pengecekan *Hash* Client 1

Pada tahap proses pengecekan hash langkah-langkahnya antara lain:

1. Data teks sebagai nilai input
2. Data teks dikirim ke Arduino 2 dari client
3. Data teks yang diterima arduino akan diproses menggunakan algoritma MD5 untuk menghasilkan *hash*
4. Setelah melalui proses MD5 akan meghasilkan *hash* dan waktu eksekusinya kemudian dikembalikan ke aplikasi *chat*
5. Membandingkan nilai *hash* dari pengirim (client 1) dengan nilai *hash* di penerima (client 2)
6. Hasil keluarannya berupa data teks, *hash*, dan waktu eksekusi
7. Proses hash chat selesai

Diagram alir proses pengecekan *hash chat* dapat digambarkan pada Gambar 3.9.



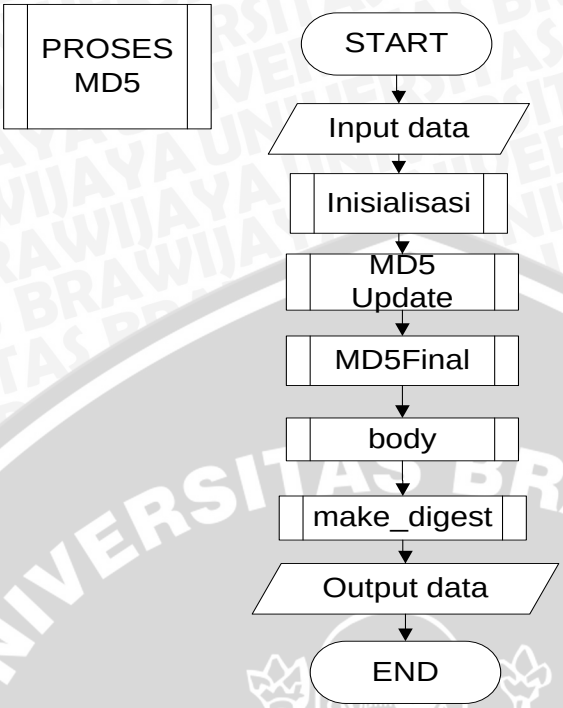
Gambar 3.9 Diagram Alir Proses Pengecekan Hash Chat

3.5.4 Perancangan Proses MD5

Langkah – langkah yang ada dalam proses MD5 sebagai berikut :

1. Data teks sebagai nilai input
2. Melakukan proses inisialisasi *buffer* algoritma MD5
3. Setelah proses inisialisai buffer kemudian melakukan proses *update* MD5
4. Proses selanjutnya adalah MD5 *final*
5. Melakukan proses *body*
6. Setelah proses *body* kemudian melakukan proses *make_digest*
7. Hasil keluaran beruapa nilai *hash*
8. Proses MD5 selesai

Diagram alir proses MD5 dapat digambarkan pada Gambar 3.10.



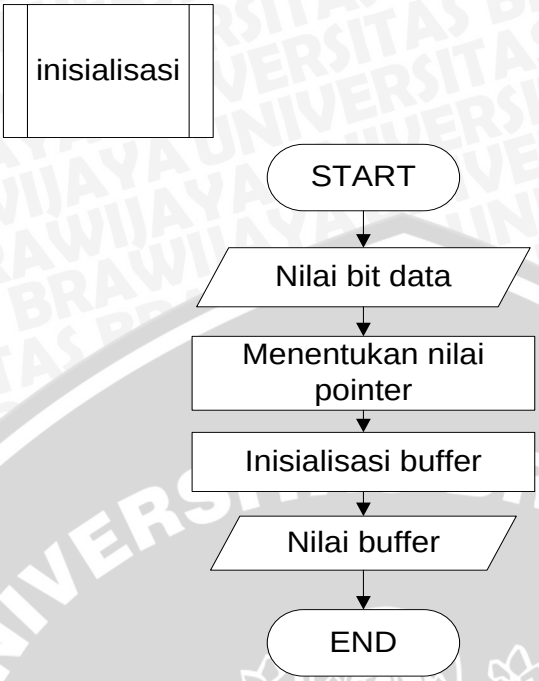
Gambar 3.10 Diagram Alir Proses MD5

3.5.4.1 Perancangan Proses Inisialisai

Langkah-langkah yang ada dalam proses inisialisasi adalah sebagai berikut :

1. Data bit sebagai nilai input
2. Menentukan nilai pointer dari *buffer*
3. Proses selanjutnya inisialisai nilai *buffer* yang ditunjukkan pada Gambar 2.10, *buffer* ini digunakan untuk menghitung proses pengolahan pesan dalam blok 512-bit
4. Hasil keluarannya berupa nilai *buffer*
5. Proses inisialisasi selesai

Diagram alir proses inisialisai dapat digambarkan pada Gambar 3.11.



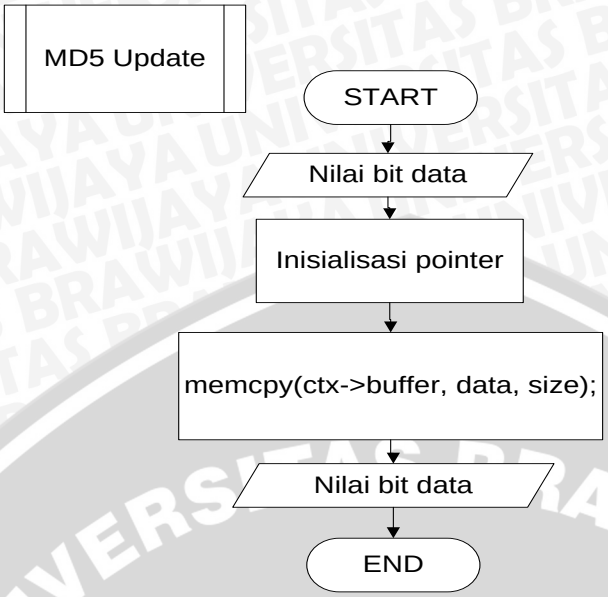
Gambar 3.11 Diagram Alir Proses Inisialisai

3.5.4.2 Perancangan Proses MD5 Update

Langkah-langkah yang ada dalam proses inisialisasi adalah sebagai berikut :

1. Data bit sebagai nilai input
2. Menetapkan nilai pointer yang akan digunakan pada proses MD5 *update*
3. Menyalin nilai pointer data dan *size* yang didapat dari proses update
4. Hasil keluarannya berupa nilai bit
5. Proses MD5 *update* selesai

Diagram alir proses MD5 Update dapat digambarkan pada Gambar 3.12.



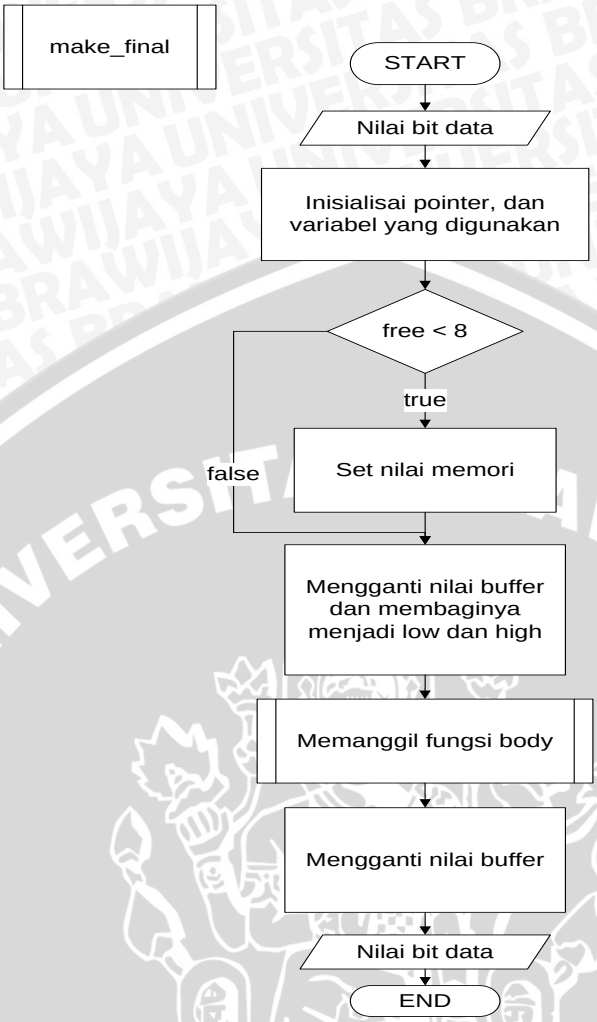
Gambar 3.12 Diagram Alir Proses MD5 Update

3.5.4.3 Perancangan Proses MD5 Final

Langkah-langkah yang ada dalam proses MD5 final adalah sebagai berikut :

1. Bit data sebagai nilai input
2. Menginisialisasikan pointer yang digunakan dan variable yang digunakan
3. Jika nilai free kurang dari 8 maka set memori jika salah ke langkah no 4
4. Mengganti nilai *buffer* dan membaginya menjadi *high* dan *low*
5. Setelah langkah ke 4 kemudian memanggil fungsi *body*
6. Mengganti nilai *buffer* dengan cara menggeser
7. Hasil keluarannya berupa nilai bit
8. Proses MD5 final selesai

Diagram alir proses MD5 final digambarkan pada Gambar 3.13.



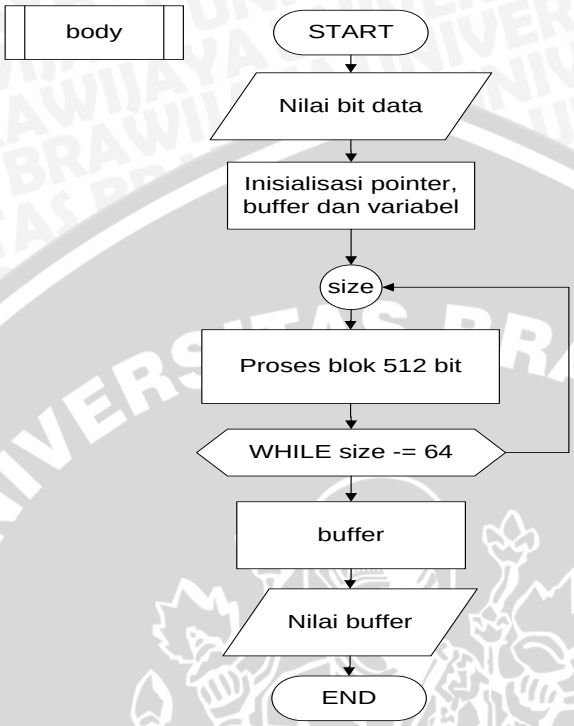
Gambar 3.13 Flowchart Proses MD5 Final

3.5.4.4 Perancangan Proses Body

Langkah-langkah yang ada dalam proses MD5 final adalah sebagai berikut :

1. Data bit sebagai nilai input
2. Menginisialisasikan pointer, buffer, dan variabel yang digunakan
3. Melakukan proses pengolahan pesan dalam blok 512-bit
4. Ketika size dikurangi dengan 64 proses berlanjut ke langkah 5, jika tidak makan akan melakukan proses iterasi
5. Nilai akir dari proses ke 4 akan menjadi nilai buffer
6. Hasil keluarannya berupa nilai *buffer*
7. Proses body selesai

Diagram alir proses MD5 final digambarkan pada Gambar 3.14.



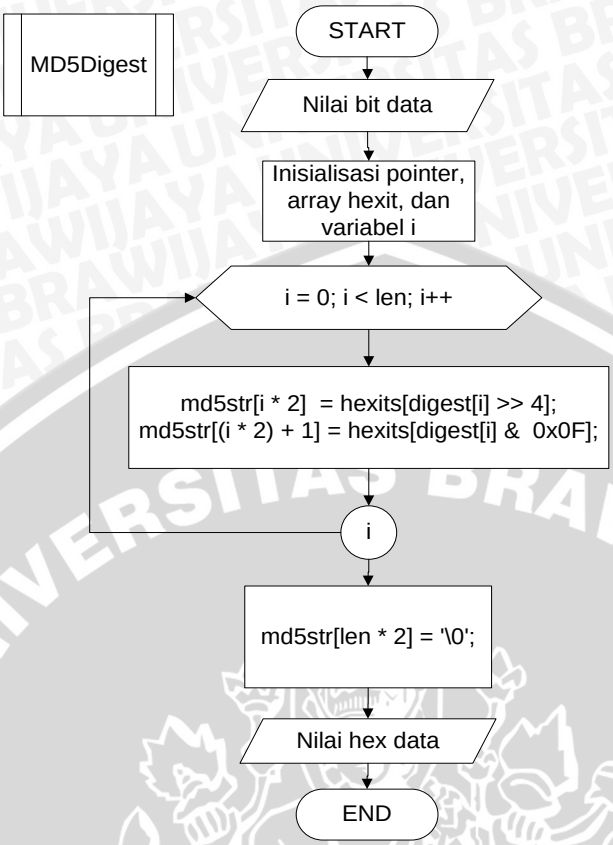
Gambar 3.14 Flowchart Proses Body

3.5.4.5 Perancangan Proses Make Digest

Langkah-langkah yang ada dalam proses MD5 final adalah sebagai berikut :

1. Data bit sebagai nilai input
2. Menginisialisasikan pointer, array dan variable i yang digunakan
3. Melakukan perulangan untuk panjang bit
4. Menentukan nilai dari md5str [I * 2] dan nilai md5str [(I * 2) + 1]
5. Melakukan iterasi hingga selesai
6. Menentukan nilai [len * 2]

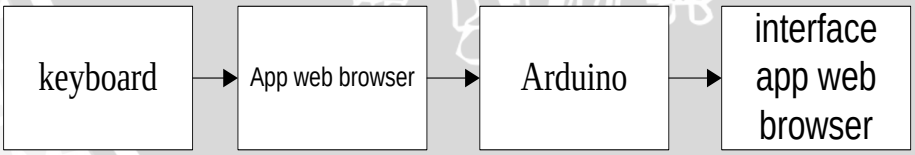
Diagram alir proses *make digest* dapat digambarkan pada Gambar 3.15.



Gambar 3.15 Flowchart Proses Make Digest

3.6 Perancangan Input dan Output Program

Perancangan Input dan Output Program akan ditunjukan pada Gambar 3.16.



Gambar 3.16 Perancangan Input Dan Output Program

Pada Gambar 3.16 menunjukkan bahwa keyboard sebagai input data teks kemudian ditampilkan pada aplikasi *Web Browser*. *Web Browser* bertindak sebagai *interface* untuk menghubungkan antara pengguna dan *system*, data teks yang diterima pada aplikasi *chat* kemudian dikirimkan pada Arduino dan diolahn menjadi *hash* data dengan algoritma MD5.

Setelah diolah di Arduino hasil proses *hash*, waktu, dan data asli dikirimkan kembali dan ditampilkan pada aplikasi web browser.

3.4 Perancangan Konfigurasi Perangkat Keras

Berdasarkan pada diagram perancangan pada Gambar 3.2. Perancangan perangkat keras dilakukan setelah analisis kebutuhan dan perancangan perangkat lunak selesai dilakukan. Konfigurasi perangkat keras ditunjukan pada Gambar 3.17.

Gambar 3. 17 Perancangan perangkat keras

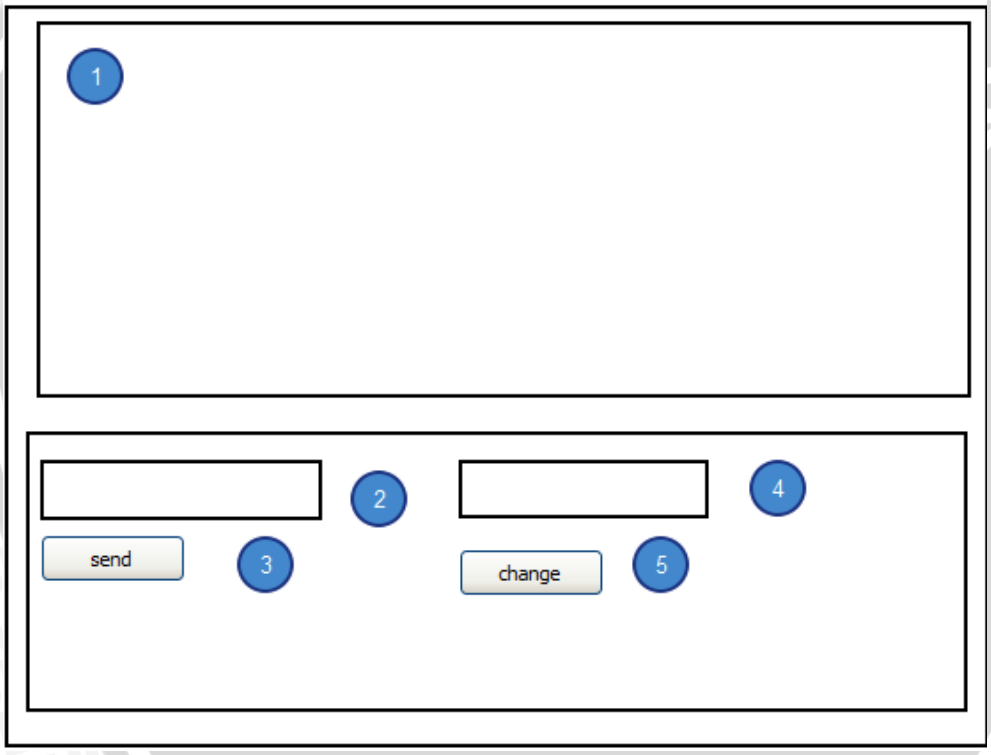
Penyusunan konfigurasi perangkat keras dilakukan setelah perancangan perangkat lunak selesai dilakukan. Seperti yang sudah disebutkan didalam analisis kebutuhan, pada perangkat keras dibutuhkan laptop/PC, arduino mega, *Ethernet shield* dan kabel USB.

Setelah implementasi program selesai dan program sukses langkah selanjutnya adalah mengaktifkan Apache dan MySQL yang diperlukan untuk menjalankan aplikasi chat. Kabel USB difungsikan sebagai penghubung antara arduino ke laptop dan arduino berfungsi sebagai suatu alat tambahan yang memberikan *integritas* pada data chat. Setelah Apache dan MySQL aktif, selanjutnya aplikasi *chat* dapat diaktifkan untuk membuka jalur koneksi antara *web server* dengan Arduino. Langkah ini

dapat dilakukan melalui *command prompt* ataupun dipanggil melalui *browser*. Aplikasi chat selanjutnya dapat dipanggil melalui *browser*.

3.7 Perancangan *Interface*

Perancangan *user interface* dari implementasi algoritma MD5 pada *embedded system* yang telah diimplementasikan pada aplikasi chat ditunjukkan pada gambar 3.18. Tahapan ini menghasilkan suatu program aplikasi sebagai media yang mewakili hasil dari impelentasi yang dilakukan. Pembuatan perancangan *interface* menggunakan aplikasi yang bernama pencil.



Gambar 3.18 Tampilan *User Interface*

Pada Gambar 3.18 terdapat 2 tombol yaitu tombol *send* (3) dan tombol *change* (5). Fungsi tombol *send* adalah untuk mengirimkan pesan sedangkan tombol *change* untuk mengganti nama dari pengguna aplikasi *chat*. Box (1) untuk menampilkan *chat*, box (2) digunakan untuk menuliskan pesan yang akan dikirimkan, pada proses ini pesan chat akan dikirimkan terlebih dahulu pada Arduino untuk mengubah menjadi nilai

hash (proses MD5) kemudian akan mengembalikan pada aplikasi *chat*. Box (4) digunakan untuk mengganti nama pengguna apilkasi *chat*.

3.8 Skenario Pengujian dan Analisis

Pengujian dimaksudkan untuk mengetahui apakah penelitian yang dilakukan telah dapat memenuhi tujuan penelitian yang telah direncanakan. Sebagaimana disebutkan diatas, tujuan dari penelitian ini adalah untuk mengimplementasikan algoritma MD5 pada *Embedded system Arduino*. Analisis dilakukan untuk melihat apakah implementasi MD5 pada Arduino memberikan hasil yang sesuai. Analisis dilakukan berdasarkan pengamatan waktu eksekusi program dan validitas data. Hasilnya dikatakan baik jika waktu yang dibutuhkan untuk memproses data teks cukup cepat dan hasil dari MD5 valid. Adapun analisis yang dilakukan meliputi:

a. Analisis pengaruh jumlah karakter terhadap waktu eksekusi

Tabel hasil analisis pengaruh jumlah karakter terhadap waktu eksekusi berisi karatker yang diuji, jumlah, hasil waktu untuk lebih jelasnya dapat dilihat pada lampiran 1.

b. Analisis pengujian validitas hasil *hash* pada system dan server

Tabel hasil analisis pengujian validitas hasil *hash* pada *system* dan *server* berisi karakter yang diuji, hasil *system*, hasil *server*, benar atau salah untuk lebih jelasnya dapat dilihat pada lampiran 8.

BAB IV
IMPLEMENTASI

Bab ini membahas tentang implementasi sistem, batasan-batasan implementasi, implementasi algoritma MD5, implementasi chat, implementasi konfigurasi aplikasi, dan implementasi *interface*.

4.1 Implementasi Sistem

Lingkungan implementasi dari *hash* teks dengan algoritma MD5 pada Arduino meliputi lingkungan perangkat lunak (*Software*) dan lingkungan perangkat keras (*Hardware*).

4.1.1 Implementasi Perangkat Keras

Lingkungan perangkat keras yang digunakan dalam mengimplementasikan *hash* teks dengan algoritma MD5 pada Arduino adalah sebagai berikut:

1. Arduino Mega
2. *Ethernet Shield*
3. Kabel LAN
4. Kabel USB
5. Laptop prosesor intel® core™ i3 CPU M330
6. RAM 2GB

4.1.2 Implementasi Perangkat Lunak

Lingkungan perangkat lunak yang digunakan dalam mengimplementasikan *hash* teks dengan algoritma MD5 adalah sebagai berikut:

1. Sistem operasi windows 7 ultimate 64-bit sebagai proses implementasi.

2. Arduino IDE 1.0.5 for windows merupakan *software development* bawaan dari arduino yang digunakan untuk pemrograman.
3. *Web server* yang digunakan pada aplikasi chat adalah *Apache* dan PHP yang ada dalam XAMPP-win32-1.8.3-2-VC11

4.2 Batasan-Batasan Implementasi

Beberapa batasan implementasi dalam implementasi teks dengan algoritma MD5 pada Arduino sebagai berikut:

1. *Hash* data text di implementasikan pada Arduino
2. Implementasi dititikberatkan pada proses hash data.
3. Aplikasi *chat* digunakan sebagai contoh suatu proses yang memerlukan implementasi algoritma MD5 yang diambil dari arduino.
4. Panjang karakter yang diuji 111 karakter untuk plain text dan menghasilkan 32 karakter nilai hash.

4.3 Implementasi Algoritma MD5

Pada bagian ini dijelaskan bagaimana implementasi MD5 dalam bahasa C untuk dapat ditanamkan ke dalam Arduino. Kode program yang digunakan merupakan milik Scott MacVicar yang diambil dari http://playground.arduino.cc/Main/LibraryList. Penjelasan disesuaikan dengan tahapan-tahapan yang terjadi pada proses *hash* dengan MD5.

4.3.1 Inisialisasi

Pada proses implementasi inisialisai array ABCD berisi empat 4-byte "*register*" yang dimanipulasi untuk menghasilkan MD5 *digest*. Input bertindak atas *register*, bukan sebaliknya. Nilai-nilai yang diberikan sebagai nilai 32-bit dalam format bahasa C. Proses inisialisai ditunjukkan pada *Sourcecode* 4.1.

Sourcecode 4.1 Inisialisasi

```
1  void MD5::MD5Init(void *ctxBuf)
```

```
2   {
3         MD5_CTX *ctx = (MD5_CTX*)ctxBuf;
4
5         ctx->a = 0x67452301;
6         ctx->b = 0xefcdab89;
7         ctx->c = 0x98badcfe;
8         ctx->d = 0x10325476;
9
10        ctx->lo = 0;
11        ctx->hi = 0;
12  }
```

4.3.2 MD5 Update

MD5 *update* merupakan proses melanjutkan sebuah Operasi MD5 dan memproses blok pesan lain kemudian meng*update* isi dari blok sebelumnya. Proses MD5 *update* ditunjukkan pada *Sourcecode* 4.2.

Sourcecode 4.2 MD5 Update

```
1   void MD5::MD5Update(void *ctxBuf, const void *data,
2   size_t size)
3   {
4         MD5_CTX *ctx = (MD5_CTX*)ctxBuf;
5         MD5_u32plus saved_lo;
6         MD5_u32plus used, free;
7         saved_lo = ctx->lo;
8         if ((ctx->lo = (saved_lo + size) & 0x1fffffff) <
9   saved_lo) {
10              ctx->hi++;
11        }
12        ctx->hi += size >> 29;
13
14        used = saved_lo & 0x3f;
15
16        if (used) {
17              free = 64 - used;
18
19              if (size < free) {
```

```
20              memcpy(&ctx->buffer[used], data, size);
21              return;
22          }
23
24          memcpy(&ctx->buffer[used], data, free);
25          data = (unsigned char *)data + free;
26          size -= free;
27          body(ctx, ctx->buffer, 64);
28      }
29
30      if (size >= 64) {
31          data = body(ctx, data, size & ~(size_t)0x3f);
32          size &= 0x3f;
33      }
34
35      memcpy(ctx->buffer, data, size);
36
```

4.3.3 MD5 *Final*

Pada proses MD5 *final* dilakukan proses konversi blok masukan ke dalam array, menjaga untuk membaca blok dalam urutan endian kecil (bit signifikan dari banyaknya tempat data multibyte yang disimpan pada alamat memory terendah). *Loop* ini melakukan empat putaran permutasi. Putaran masing-masing sangat mirip, perbedaannya pada fungsi (F, G, H, dan I) digunakan untuk melakukan permutasi *bitwise* pada *register*. Implementasi lengkap MD5 *final* ditunjukkan pada *Sourcecode* 4.3.

Sourcecode 4.3 MD5 Final

```
1   void MD5::MD5Final(unsigned char *result, void *ctxBuf)
2   {
3       MD5_CTX *ctx = (MD5_CTX*)ctxBuf;
4       MD5_u32plus used, free;
5
6       used = ctx->lo & 0x3f;
7
8       ctx->buffer[used++] = 0x80;
```

```
9
10      free = 64 - used;
11
12      if (free < 8) {
13          memset(&ctx->buffer[used], 0, free);
14          body(ctx, ctx->buffer, 64);
15          used = 0;
16          free = 64;
17      }
18
19      memset(&ctx->buffer[used], 0, free - 8);
20
21      ctx->lo <<= 3;
22      ctx->buffer[56] = ctx->lo;
23      ctx->buffer[57] = ctx->lo >> 8;
24      ctx->buffer[58] = ctx->lo >> 16;
25      ctx->buffer[59] = ctx->lo >> 24;
26      ctx->buffer[60] = ctx->hi;
27      ctx->buffer[61] = ctx->hi >> 8;
28      ctx->buffer[62] = ctx->hi >> 16;
29      ctx->buffer[63] = ctx->hi >> 24;
30
31      body(ctx, ctx->buffer, 64);
32
33      result[0] = ctx->a;
34      result[1] = ctx->a >> 8;
35      result[2] = ctx->a >> 16;
36      result[3] = ctx->a >> 24;
37      result[4] = ctx->b;
38      result[5] = ctx->b >> 8;
39      result[6] = ctx->b >> 16;
40      result[7] = ctx->b >> 24;
41      result[8] = ctx->c;
42      result[9] = ctx->c >> 8;
43      result[10] = ctx->c >> 16;
44      result[11] = ctx->c >> 24;
45      result[12] = ctx->d;
46      result[13] = ctx->d >> 8;
```

```
47          result[14] = ctx->d >> 16;
48          result[15] = ctx->d >> 24;
49
50          memset(ctx, 0, sizeof(*ctx));
51  }
```

4.3.4 *Body*

Fungsi *body* yang ditunjukkan pada *Sourcecode* 4.4 melakukan proses putaran pertama. Dalam putaran berikutnya, dilakukan proses sedikit lebih bervariasi, dalam setiap putaran terdapat operasi memutar ke kiri dilakukan sebagai bagian dari 16 permutasi. Ada empat putaran dari 16 permutasi untuk total 64, dalam setiap 64 operasi permutasi ini nilai konstanta yang di tambahkan berbeda.

Sourcecode 4.4 *body*:

```
1   const void *MD5::body(void *ctxBuf, const void *data,
2   size_t size)
3   {
4           MD5_CTX *ctx = (MD5_CTX*)ctxBuf;
5           const unsigned char *ptr;
6           MD5_u32plus a, b, c, d;
7           MD5_u32plus saved_a, saved_b, saved_c, saved_d;
8
9           ptr = (unsigned char*)data;
10
11          a = ctx->a;
12          b = ctx->b;
13          c = ctx->c;
14          d = ctx->d;
15
16          do {
17              saved_a = a;
18              saved_b = b;
19              saved_c = c;
20              saved_d = d;
21
22  /* Round 1 */
```

```
23          STEP(F, a, b, c, d, SET(0), 0xd76aa478, 7)
24          STEP(F, d, a, b, c, SET(1), 0xe8c7b756, 12)
25          STEP(F, c, d, a, b, SET(2), 0x242070db, 17)
26          STEP(F, b, c, d, a, SET(3), 0xc1bdceee, 22)
27          STEP(F, a, b, c, d, SET(4), 0xf57c0faf, 7)
28          STEP(F, d, a, b, c, SET(5), 0x4787c62a, 12)
29          STEP(F, c, d, a, b, SET(6), 0xa8304613, 17)
30          STEP(F, b, c, d, a, SET(7), 0xfd469501, 22)
31          STEP(F, a, b, c, d, SET(8), 0x698098d8, 7)
32          STEP(F, d, a, b, c, SET(9), 0x8b44f7af, 12)
33          STEP(F, c, d, a, b, SET(10), 0xffff5bb1, 17)
34          STEP(F, b, c, d, a, SET(11), 0x895cd7be, 22)
35          STEP(F, a, b, c, d, SET(12), 0x6b901122, 7)
36          STEP(F, d, a, b, c, SET(13), 0xfd987193, 12)
37          STEP(F, c, d, a, b, SET(14), 0xa679438e, 17)
38          STEP(F, b, c, d, a, SET(15), 0x49b40821, 22)
39
40  /* Round 2 */
41          STEP(G, a, b, c, d, GET(1), 0xf61e2562, 5)
42          STEP(G, d, a, b, c, GET(6), 0xc040b340, 9)
43          STEP(G, c, d, a, b, GET(11), 0x265e5a51, 14)
44          STEP(G, b, c, d, a, GET(0), 0xe9b6c7aa, 20)
45          STEP(G, a, b, c, d, GET(5), 0xd62f105d, 5)
46          STEP(G, d, a, b, c, GET(10), 0x02441453, 9)
47          STEP(G, c, d, a, b, GET(15), 0xd8a1e681, 14)
48          STEP(G, b, c, d, a, GET(4), 0xe7d3fbc8, 20)
49          STEP(G, a, b, c, d, GET(9), 0x21e1cde6, 5)
50          STEP(G, d, a, b, c, GET(14), 0xc33707d6, 9)
51          STEP(G, c, d, a, b, GET(3), 0xf4d50d87, 14)
52          STEP(G, b, c, d, a, GET(8), 0x455a14ed, 20)
53          STEP(G, a, b, c, d, GET(13), 0xa9e3e905, 5)
54          STEP(G, d, a, b, c, GET(2), 0xfcefa3f8, 9)
55          STEP(G, c, d, a, b, GET(7), 0x676f02d9, 14)
56          STEP(G, b, c, d, a, GET(12), 0x8d2a4c8a, 20)
57
58  /* Round 3 */
59          STEP(H, a, b, c, d, GET(5), 0xfffa3942, 4)
60          STEP(H, d, a, b, c, GET(8), 0x8771f681, 11)
```

```
61              STEP(H, c, d, a, b, GET(11), 0x6d9d6122, 16)
62              STEP(H, b, c, d, a, GET(14), 0xfde5380c, 23)
63              STEP(H, a, b, c, d, GET(1), 0xa4beea44, 4)
64              STEP(H, d, a, b, c, GET(4), 0x4bdecfa9, 11)
65              STEP(H, c, d, a, b, GET(7), 0xf6bb4b60, 16)
66              STEP(H, b, c, d, a, GET(10), 0xbebfbc70, 23)
67              STEP(H, a, b, c, d, GET(13), 0x289b7ec6, 4)
68              STEP(H, d, a, b, c, GET(0), 0xeaa127fa, 11)
69              STEP(H, c, d, a, b, GET(3), 0xd4ef3085, 16)
70              STEP(H, b, c, d, a, GET(6), 0x04881d05, 23)
71              STEP(H, a, b, c, d, GET(9), 0xd9d4d039, 4)
72              STEP(H, d, a, b, c, GET(12), 0xe6db99e5, 11)
73              STEP(H, c, d, a, b, GET(15), 0x1fa27cf8, 16)
74              STEP(H, b, c, d, a, GET(2), 0xc4ac5665, 23)
75
76      /* Round 4 */
77              STEP(I, a, b, c, d, GET(0), 0xf4292244, 6)
78              STEP(I, d, a, b, c, GET(7), 0x432aff97, 10)
79              STEP(I, c, d, a, b, GET(14), 0xab9423a7, 15)
80              STEP(I, b, c, d, a, GET(5), 0xfc93a039, 21)
81              STEP(I, a, b, c, d, GET(12), 0x655b59c3, 6)
82              STEP(I, d, a, b, c, GET(3), 0x8f0ccc92, 10)
83              STEP(I, c, d, a, b, GET(10), 0xffeff47d, 15)
84              STEP(I, b, c, d, a, GET(1), 0x85845dd1, 21)
85              STEP(I, a, b, c, d, GET(8), 0x6fa87e4f, 6)
86              STEP(I, d, a, b, c, GET(15), 0xfe2ce6e0, 10)
87              STEP(I, c, d, a, b, GET(6), 0xa3014314, 15)
88              STEP(I, b, c, d, a, GET(13), 0x4e0811a1, 21)
89              STEP(I, a, b, c, d, GET(4), 0xf7537e82, 6)
90              STEP(I, d, a, b, c, GET(11), 0xbd3af235, 10)
91              STEP(I, c, d, a, b, GET(2), 0x2ad7d2bb, 15)
92              STEP(I, b, c, d, a, GET(9), 0xeb86d391, 21)
93
94              a += saved_a;
95              b += saved_b;
96              c += saved_c;
97              d += saved_d;
98
```

```
99              ptr += 64;
100         } while (size -= 64);
101
102         ctx->a = a;
103         ctx->b = b;
104         ctx->c = c;
105         ctx->d = d;
106
107         return ptr;
108 }
```

4.3.5 *Make Digest*

Fungsi *make digest* digunakan untuk mengubah bit-bit biner menjadi nilai hex. Menambah panjang blok baru dengan total panjang, dan menyalin data blok baru ke dalam blok konteks (ctx). Kemudian memanggil fungsi permutasi setiap kali blok penuh, Proses *Make Digest* ditunjukkan pada *Sourcecode* 4.5.

Sourcecode 4.5 Make Digest

```
1   char* MD5::make_digest(const unsigned char *digest, int
2   len) /* {{{ */
3   {
4         char * md5str = (char*)
5   malloc(sizeof(char)*(len*2+1));
6         static const char hexits[17] = "0123456789abcdef";
7         int i;
8   //Add the new block's length to the total length.
9   //Copy the new block's data into the context block.
10     *// Call the Permute() function whenever the context
    //block is full.
11
12        for (i = 0; i < len; i++) {
13              md5str[i * 2]       = hexits[digest[i] >> 4];
14              md5str[(i * 2) + 1] = hexits[digest[i] &
15  0x0F];
16        }
17        md5str[len * 2] = '\0';
```

```
18          return md5str;
19  }
```

4.4 Pembuatan Aplikasi Chat

Kode program dan komponen dalam aplikasi chat yaitu:

1. Kode program pengiriman data *chat* ke computer lain.
2. *User interface* yang digunakana untuk tampilan pada chat
3. Kode program chat_2 yang digunakan untuk pengiriman data *chat* ke Arduino yang akan memproses MD5
4. Kode untuk konfigurasi *chat* agar aplikasi dapat digunakan untuk konfigurasi aplikasi *chat.*
5. Kode program utama adalah digunakan untuk perhitungan MD5 pada Arduino.

4.4.1 Kode Program Chat

Kode program chat bertujuan untuk mengambil dan menerima data-data chat yang dikirimkan ke komputer lain berupa pesan, waktu, dan hasil *hash*. Kode ini disimpan pada *file* Post.php ditunjukkan pada *Sourcecode* 4.6.

Sourcecode 4.6 Kode Program Chat

```
1   require_once('conf.php');
2   session_start();
3
4   if(isset($_POST['msg'])){
5           $msg = $_POST['msg'];
6           $waktu = $_POST['waktu'];
7           $hash = $_POST['hash'];
8           $st = "valid";
9           if(md5($msg)!==$hash){
10              $st= "invalid";
11          }
12          if(preg_match('%^[A-Za-z0-9\s\?\!\'\-
13  \_\;]{1,255}$%', trim($msg))){
14
```

```
15              $fp =
16  fsockopen(CON_IP,CON_PORT,$errors,$errdesc,1);
17              fputs($fp,$_SESSION['name'].' says:
18  '.$msg.'<br>hash : '.$hash.'</br> (time :  '.$waktu.'
19  ms)');
20              fclose($fp);
21
22          } else {
23              echo 'Wrong input';
24          }
25
26  }
27
28  if(isset($_POST['name'])){
29      $_SESSION['name'] = strip_tags($_POST['name']);
30  }
31  if(!isset($_SESSION['name'])){
32        $_SESSION['name'] = 'Anonymous';
33  }
```

4.4.2 Kode Program User Interface

Kode program *user interface* digunakan untuk sarana penghubung antara user dan aplikasi untuk berinteraksi. Pada proses *user interface* berisikan form yang digunakan oleh user untuk proses komunikasi. Kode ini berada dalam *file* post.php yang ditunjukkan pada *Sourcecode* 4.7.

Sourcecode 4.7 Kode Program *User Interface*

```
1   <head>
2         <script type="text/javascript"
3   src="http://code.jquery.com/jquery-
4   1.10.2.min.js"></script>
5         <title>Aplikasi Chat</title>
6   </head>
7
8         <body>
9             <label>IP Arduino:</label><br><input
10  type="text" value="169.254.41.82" id="iparduino" />
```

```
11          <br><br>
12          <form method="post" id="chat">
13            <input name="hash" value="" type="hidden" />
14            <input name="waktu" value="" type="hidden"
15  />
16            <label>Pesan :</label><br>
17            <input name="msg" type="text" /><br />
18            <input type="submit" value="Send" />
19
20          </form>
21
22          <form method="post">
23
24          <input name="name" type="text" value="<?php
25  echo $_SESSION['name']; ?>" /><br />
26          <input type="submit" value="change name" />
27          </form>
28        </body>
29  </html>
```

4.4.3 Kode Program chat_2

Masih dalam file post.php, *Sourcecode* 4.8 merupakan kode program chat_2 yang digunakan untuk mengirim isi chat pada Arduino menggunakan AJAX dengan pengembalian tipe data JSONP. Data yang diterima pada Arduino kemudian diproses menjadi *Hash.*

Sourcecode 4.8 Kode Program Chat_2

```
1   $(document).ready(function(){
2
3       var iparduino = $('#iparduino').val();
4
5       $('form#chat').submit(function(e){
6
7           $('input[name=hash]').val('');
8           $('input[name=waktu]').val('');
9
10          $.ajax({
11            dataType: 'jsonp',
```

```
12              // async: false,
13              data: {'msg': $('input[name=msg]').val()},
14              jsonp: 'cb',
15              url: 'http://'+iparduino+'/input',
16              success:function(a){
17
18              }
19
20          });
21
22          window.cb = function(o){
23                  $('input[name=hash]').val(o.hash);
24                  $('input[name=waktu]').val(o.waktu);
25                  console.log(o);
26
27          }
28
29
30          alert('ok');
31          return true;
32
33      })
34
35  })
```

4.4.4 Kode Program Konfigurasi Chat

Kode program konfigurasi *chat* digunakan untuk proses konfigurasi aplikasi *chat* yang berisi IP *server*, alamat *port*, maksimal pengguna yang dapat berkomunikasi dengan *server*. Kode ini disimpan pada file conf.php yang ditujukkan pada *Sourcecode* 4.9.

Sourcecode 4.9 Kode Program Konfigurasi *Chat*

```
1   <?php
2
3   define('CON_IP','172.21.10.175');
4   define('CON_PORT','8080');
5   define('MAX_CLIENTS','10');
```

```
6  ?>
```

4.4.5 Kode Program Utama

Proses implementasi program utama digunakan untuk proses komunikasi antara *server* dan *Ethernet Shield* dan menerima data dari aplikasi *chat* kemudian data diproses dengan algoritma MD5 untuk diubah menjadi data *hash*. Kode ini disimpan pada file tes.ino yang ditunjukkan *Sourcecode* 4.10.

Sourcecode 4.10. Kode Program Utama

```
1  #include <SPI.h>
2  #include <Ethernet.h>
3  #include <MD5.h>
4  #include <time.h>
5
6  byte mac[] = {
7    0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
8  IPAddress ip(169,254,41,82);
9  EthernetServer server(80);
10
11 String HTTP_req;
12 char *inputchat = "";
13 char *md5str = "";
14 unsigned long int waktu = 0;
15
16 void setup() {
17      Serial.begin(9600);
18    while (!Serial) {
19      ;
20   }
21 Ethernet.begin(mac, ip);
22   server.begin();
23
24   Serial.print("server is at ");
25   Serial.println(Ethernet.localIP());
26 }
27
```

```
28  void loop()
29  {
30      EthernetClient client = server.available();
31      if (client) {
32          boolean currentLineIsBlank = true;
33          while (client.connected()) {
34
35              if (client.available())
36  char c = client.read();
37
38                  HTTP_req += c;
39                  if (c == '\n' && currentLineIsBlank) {
40
41  if (HTTP_req.indexOf("input") > -1) {
42  // ambil data dari aplikasi chatint startsplit=
43                      HTTP_req.indexOf("&msg=")+5;
44                        int endsplit=
45                      HTTP_req.indexOf("&_=")+1;
46  String msg = HTTP_req.substring(startsplit, endsplit);
47  char
48                      myCharArray[msg.length()];msg.toCh
49                      arArray(myCharArray,msg.length());
50  inputchat = myCharArray;
51
52
53  unsigned long int t1 = micros();
54
55                      unsigned char* hash=
56  MD5::make_hash(inputchat);
57  md5str = MD5::make_digest(hash, 16);
58  unsigned long int t2 = micros();
59  waktu = t2-t1;
60
61  client.print("cb({'hash':'");
62  client.print(md5str);
63  client.print("','chat':'");
64  client.print(inputchat);
65  client.print("','waktu':'");
```

```
66  client.print(waktu);
67   client.print("'})");
68                      }
69
70   HTTP_req = "";
71                      break;
72                  }
73                  if (c == '\n') {
74                      currentLineIsBlank = true;
75                  }
76                  else if (c != '\r') {
77  currentLineIsBlank = false;
78                  }
79              }
80          }
81   delay(1);              client.stop();
82      }
83  }
```

4.5 Konfigurasi Aplikasi Chat

Konfigurasi Aplikasi *Chat* meliputi *upload* program ke Arduino, mengaktifkan aplikasi *chat*, dan setelah itu menampilkan *interface* pada aplikasi *Web Browser*.

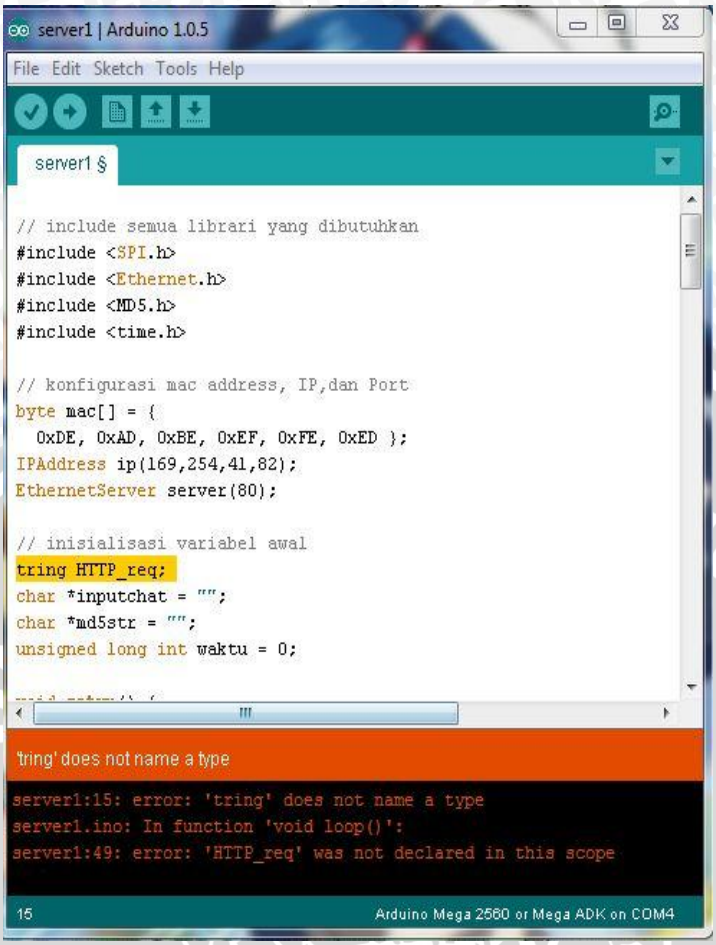
1. Menghubungkan Laptop dengan Arduino

Tahap untuk *upload* program ke Arduino dibutuhkan kabel USB dan LAN untuk menghubungkan antara laptop dan Arduino. Susunan hubungan antara laptop dan Arduino ditunjukkan pada gambar 4.1.

Gambar 4.1 Menghubungkan Laptop dengan Arduino

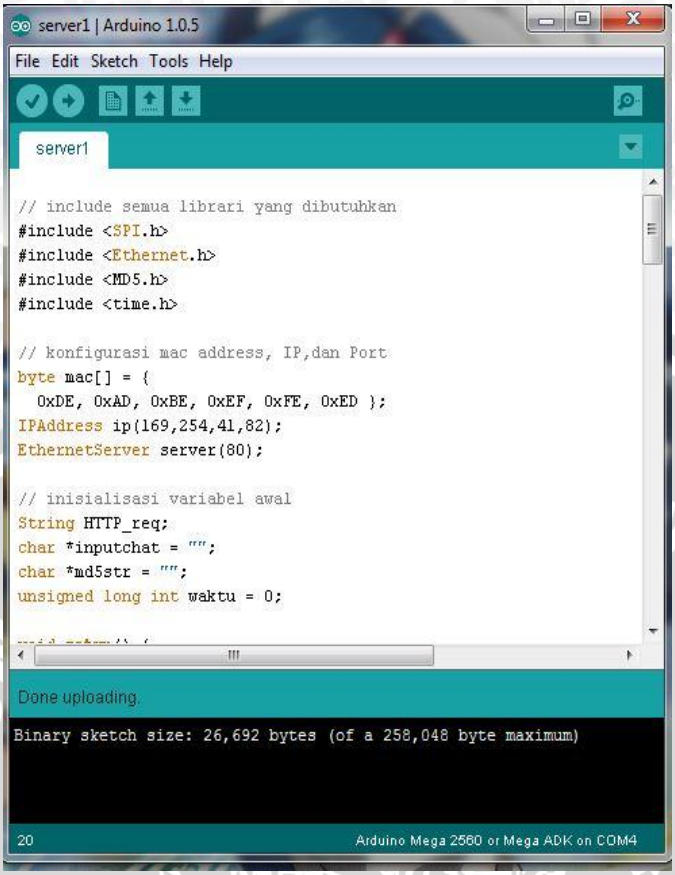
2. *Upload* program ke Arduino

Tahap awal yang dilakukan sebelum *upload* program kedalam Arduino adalah memastikan bahwa program telah selesai, berhasil dijalankan dan tidak ada *error* dalam program. Proses compile program seperti yang ditunjukan pada Gambar 4.2.



Gambar 4.2 Compile Kemudian Upload Program Pada Arduino

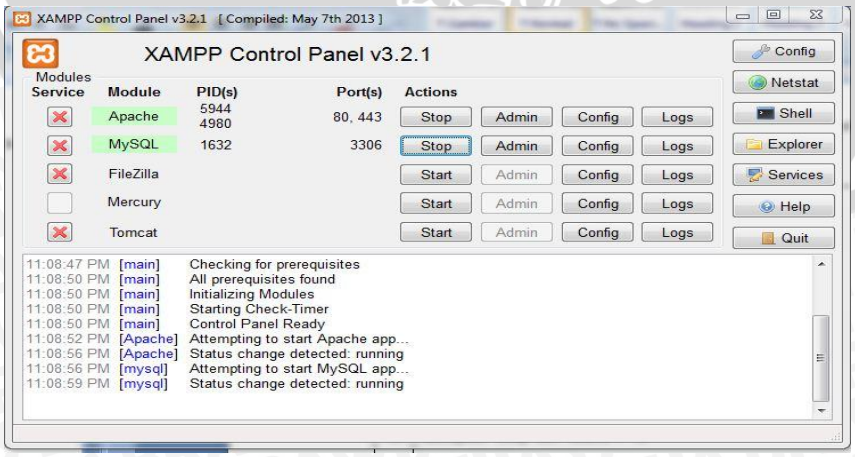
Pada kolom *comment* program harus berhasil dan berwarna putih. Warna kuning menunjukan pada *program* masih ada kesalahan. Tanda sukses dan tidak ada eror pada program ditunjukan dalam Gambar 4.3.



Gambar 4.3 Program sukses

3. Mengaktifkan Apache dan MySQL pada XAMPP

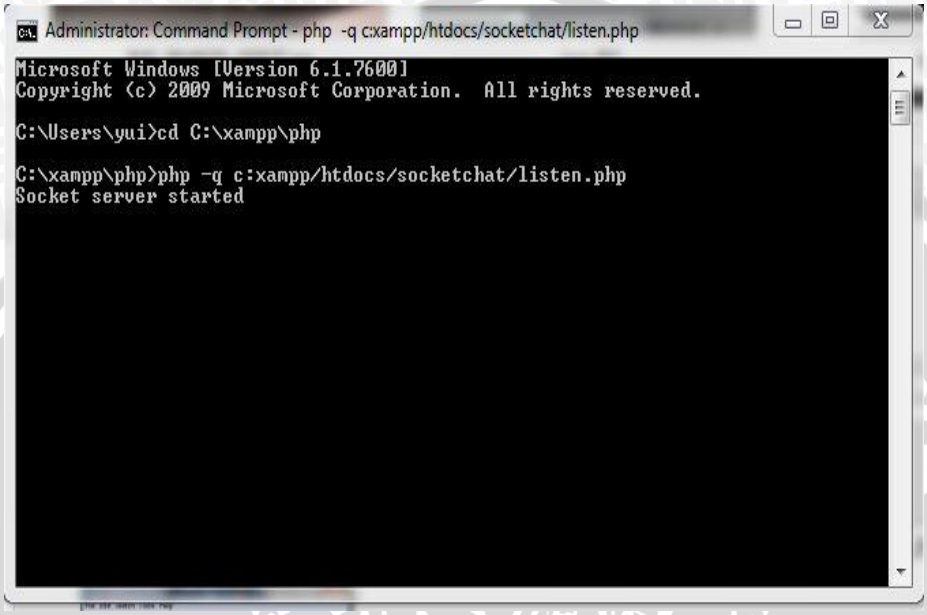
Proses selanjutnya setelah *upload* program pada Arduino yaitu mengaktifkan Apache dan MySQL pada XAMPP agar aplikasi chat bisa dijalankan, proses ditunjukkan pada gambar 4.4.



Gambar 4.4 Mengaktifkan Apache Dan Mysql

3. Mengaktifkan Aplikasi Chat

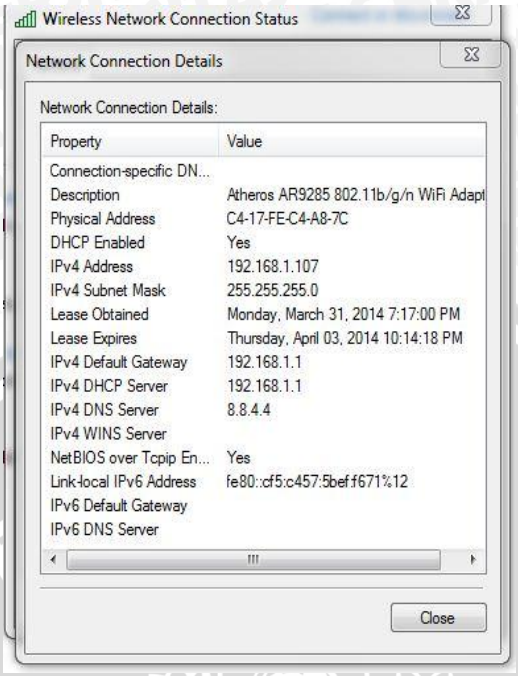
Pada proses mengaktifkan aplikasi chat dilakukan dengan cara menuliskan perintah pada *command prompt* adapun proses tampilannya ditunjukkan pada gambar 4.5.



Gambar 4.5 Mengaktifkan Aplikasi Chat

4. Proses Menampilkan Program

Setelah langkah proses 1– 4 dilakukan kemudian menampilkan *interface* program dengan membuka aplikasi *web browser*, kemudian menuliskan alamat IP *server*, untuk mengecek IP pada *Server* dapat dilihat pada gambar 4.6



Gambar 4.6 Cek IP Server

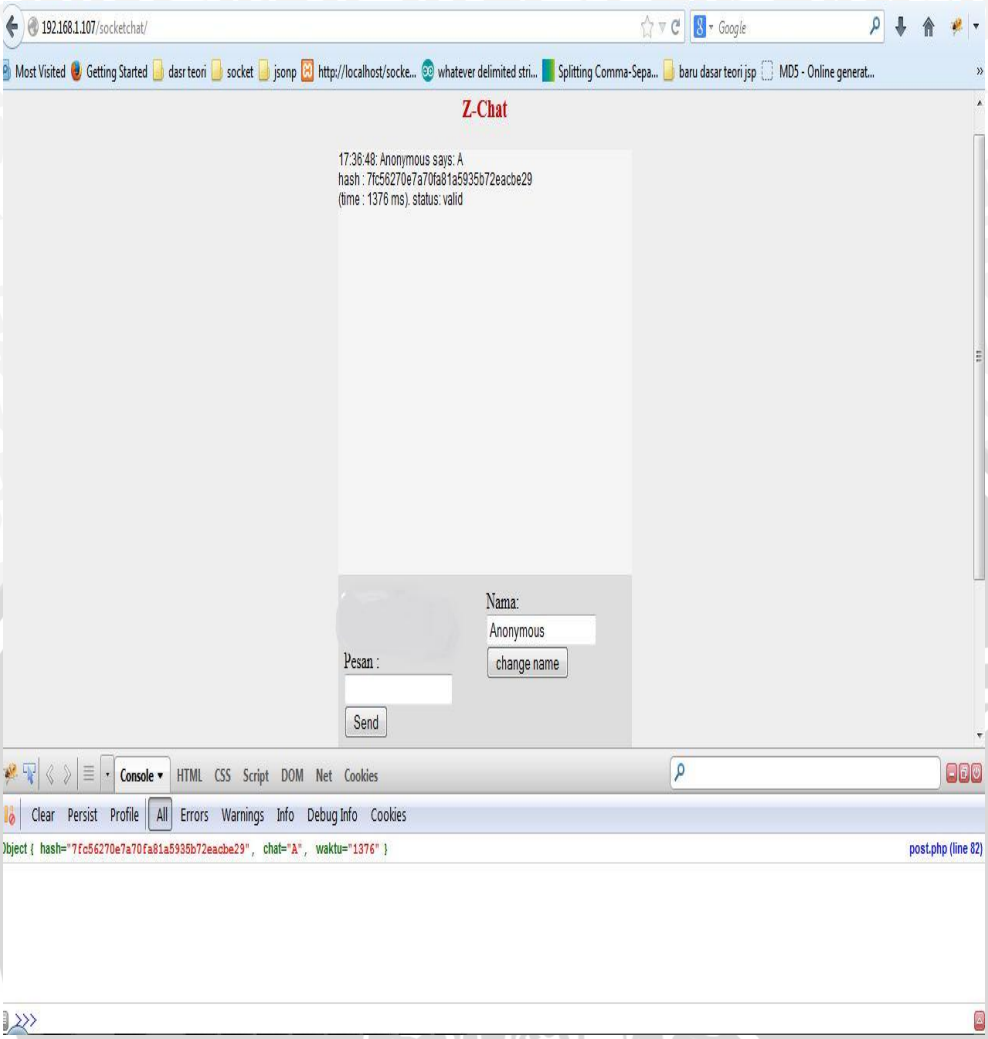
Gambar 4.6 menunjukkan alamat IP pada komputer, karena komputer bertindak sebagai *server* maka alamat IP tersebut menjadi alamat IP *server* yang diakses *client* lain yang ingin menggunakan aplikasi *chat*

4.6 User Interface

Implementasi *interface* terdiri dari 2 bagian, yaitu:

a. Form Utama

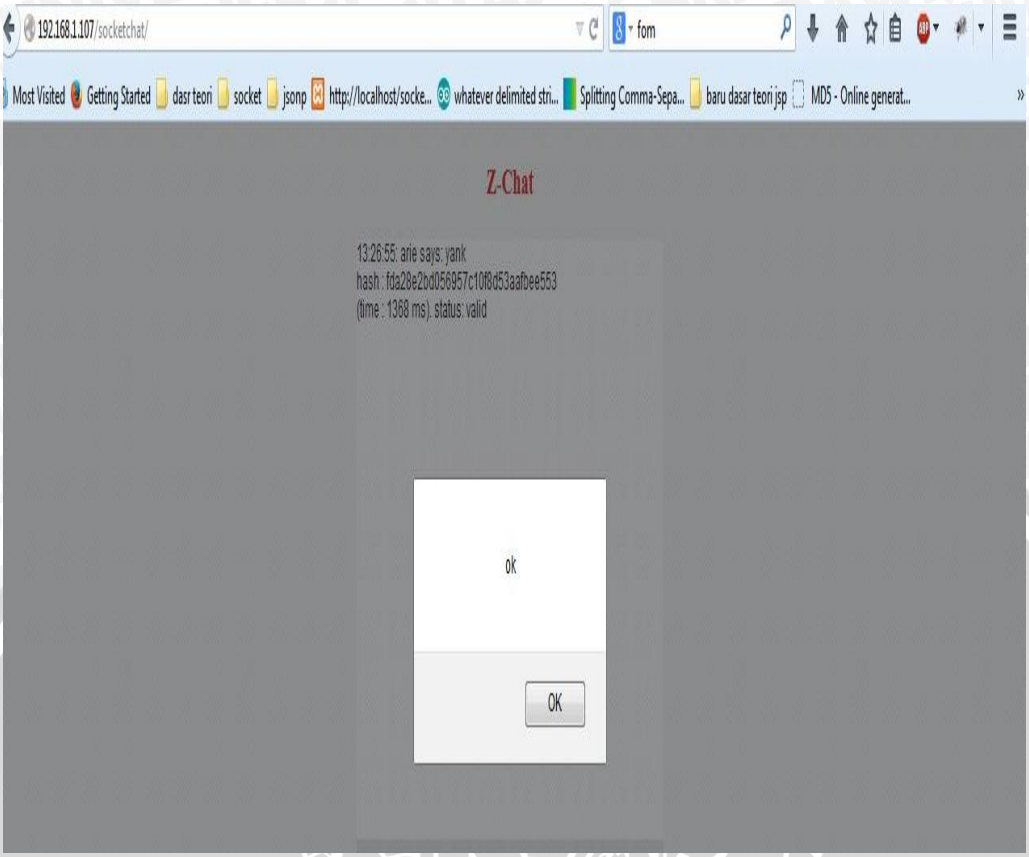
Form ini menampilkan seluruh *menu* dan hasil yang ditampilkan pada aplikasi *chat*. Adapun *menu* dalam form ini antara lain: form chat dan tombol *send* yang digunakan untuk menuliskan pesan yang dikirimkan, form nama dan *change* yang digunakan untuk mengubah nama pengguna dan menyimpan nama yang diubah oleh pengguna. Adapun tampilannya ditunjukkan pada Gambar 4.7 berikut ini



Gambar 4.7 Form Utama

b. Tombol OK

Form ini menampilkan seluruh *menu* OK yang ditampilkan yang ada pada aplikasi *chat* ketika data berhasil di *hash*kan. Adapun tampilannya ditunjukkan pada Gambar 4.8 berikut ini:



Gambar 4.8 Tombol Ok

BAB V
PENGUJIAN DAN ANALISIS

5.1 Pengujian

Pengujian dilakukan dengan dua tahap yaitu pengujian *validitas* dan pengujian waktu. Pengujian pertama adalah pengujian *validitas* hasil *hash system*. Pengujian dilakukan dengan membandingkan hasil *hash* dari *system* yang diimplementasikan dengan hasil *hash* yang dihasilkan pada *server*. Jika sistem memberikan hasil yang valid maka, hasil yang dikeluarkan sama dengan hasil yang dikeluarkan oleh *server*. Pengujian ini diambil 111 Data yang di uji.

Pengujian kedua adalah pengujian waktu, pengujian ini dilakukan untuk mengetahui waktu eksekusi yang dibutuhkan sistem untuk menghasilkan *hash* data teks. Pengujian ini dilakukan dengan cara memasukkan beberapa *sample* data yang berbeda panjang karakternya. Hasil dari setiap panjang karakter dibandingkan dan di analisa.

5.1.1 Pengujian Waktu Eksekusi

Pada Tabel 5.1 adalah hasil pengujian waktu eksekusi terhadap 111 karakter adapun hasil pengujiannya akan dituliskan 10 data acak, dan hasil rata-rata waktu yang diperlukan untuk memproses satu blok yaitu antara 1 file karakter sampai dengan 55 karakter adalah kurang dari 1 *second* yaitu antara 0.00136 s sampai 0.001396 s yang dapat dilihat pada lampiran 1, pengujian 10 data acak dapat ditunjukkan pada table 5.1 sebagai berikut:

Tabel 5.1 Pengujian Waktu *Eksekusi System*

No	Karakter yang diuji	Jumlah karakter	Blok	Hasil waktu
1.	A	1	1	0.00136 s
2.	Brawijaya	10	1	0.001372 s
5.	Hasil dari setiap panjang karakter	35	1	0.001384 s

6.	Variable dideklarasikan Didalam proses	40	1	0.001388 s
7.	dari Hasil Pengujian didapatkan KesimpulaN bahwa m	50	1	0.001396 s
8.	dari Hasil Pengujian didapatkan Kesimpulan bahwa md5 dapat diimplemen	70	2	0.002576 s
9.	dari Hasil pengujian didapatkan Kesimpulan bahwa md5 dapat diimplementasikan padaembe	85	2	0.0026 s
10.	Dari hasil pengujian didapatkan kesimpulan, bahwa MD5 dapat diimplementasikan pada embedded system..	100	2	0.002608s

5.1.2 Pengujian Validitas

Pada Tabel 5.2 adalah hasil pengujian validitas terhadap 111 karakter yang diuji dan dibandingkan dengan *server*. Adapun hasil pengujiannya dituliskan 10 data acak dan sisanya dituliskan pada lampiran sebagai berikut:

Tabel 5.2 Pengujian Validitas

No	Karakter yang diuji	Hasil dari system	Hasil dari server	Benar atau salah
1.	A	7fc56270e7a70fa81a5935b72eacbe29	7fc56270e7a70fa81a5935b72eacbe29	Benar
2.	Brawijaya	7ba0cd4c4b7b4c80ccb9164b053455b4	7ba0cd4c4b7b4c80ccb9164b053455b4	Benar

3.	Perancangan sistem !	74de3641514b5d008ae433e422fdeef8	74de3641514b5d008ae433e422fdeef8	Benar
4.	APA SAJA YANG BISA dipakai !!	7a03eb4265a14c45678477d9e397c0ef	7a03eb4265a14c45678477d9e397c0ef	Benar
5.	Hasil dari setiap panjang karakter	dbf706fecdac23923dd2c58a4570a88b	dbf706fecdac23923dd2c58a4570a88b	Benar
6.	Pengambilan kesimpulan dilakukan setelah	b3cec0df5ca45fc2183f15cdd71535d7	b3cec0df5ca45fc2183f15cdd71535d7	Benar
7.	dari Hasil Pengujian didapatkan KesimpulaN bahwa m	98675de4ce3f9faa55f1e16ae2c9e744	98675de4ce3f9faa55f1e16ae2c9e744	Benar
8.	dari Hasil Pengujian didapatkan Kesimpulan bahwa md5 dapat diimplemen	5001631313c8b48618cc0772c8b18953	5001631313c8b48618cc0772c8b18953	Benar
9.	dari Hasil pengujian didapatkan Kesimpulan bahwa md5 dapat diimplementasikan padaembe	abea7a1c103a4f4f38013082eb346659	abea7a1c103a4f4f38013082eb346659	Benar
10.	Dari hasil pengujian didapatkan kesimpulan, bahwa MD5 dapat	32362862af1f3cb722917af4b2e79e22	32362862af1f3cb722917af4b2e79e22	Benar

	diimplementasikan pada embedded system..			

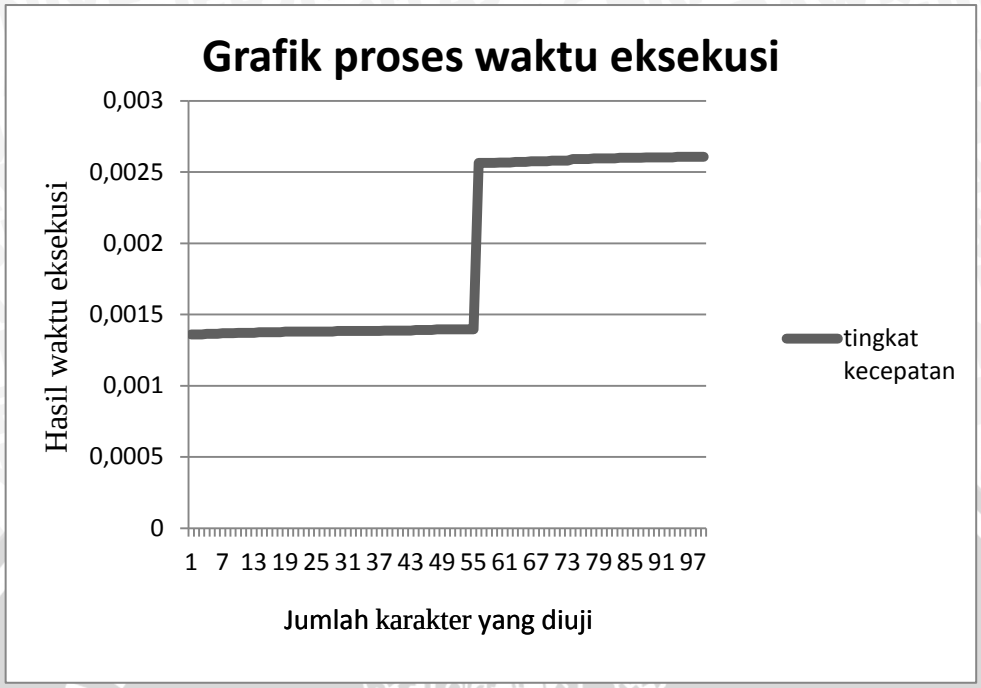
5.2 Pengolahan Hasil pengujian

Dari hasil yang didapatkan dari pengujian dapat dilihat bahwa perbedaan waktu yang dibutuhkna Arduino untuk mengeksekusi data yang diolah menjadi nilai *hash*. Perbedaan tersebut dapat dilihat dengan cara membandingkan berapa jumlah karakter yang diproses oleh *system*. Hubungan jumlah karakter dengan waktu eksekusi ditunjukkan pada tabel 5.3 berikut:

Tabel 5.3 Kenaikan Waktu Eksekusi

Jumlah karakter ke	Blok	Kenaikan
1 – 55 karakter	1	0.000036 s
55 – 56 karakter	2	0.001168 s
56 - 111 karakter	3	0.001220 s

Pada table 5.3 merupakan hasil pengujian waktu eksekusi, jumlah kenaikan semakin tinggi ketika penambahan satu putaran pada proses perhitungan MD5 yaitu ketika pada 55 ke 56 karakter bertambah 1 blok dan kenaikan waktu menjadi 0.001168 s. Hubungan jumlah data teks dengan waktu eksekusi ditunjukkan pada gambar grafik berikut ini:



Gambar 5.1 Grafik Kecepatan Waktu Eksekusi Terhadap Jumlah Karakter

5.3 Analisis

Proses analisis bertujuan untuk mendapatkan kesimpulan dari hasil pengujian yang telah dilakukan. Analisis mengacu pada pengujian yang dilakukan dan berikut ini hasil analisis yang diperoleh dari pengujian yang dilakukan.

5.3.1 Pengaruh Jumlah Karakter Terhadap Waktu Eksekusi

Berdasarkan Gambar 5.1 menunjukkan peningkatan jumlah karakter mengakibatkan meningkatnya besar waktu eksekusi yang diproses oleh *system* pada Arduino. Pada proses ini karakter yang diuji sebagai pembanding tingkat proses eksekusi dari data teks yang diubah menjadi nilai *hash*. Semakin banyak data teks yang diproses, maka jumlah blok dan waktu eksekusi proses data teks algoritma MD5 pada Arduino semakin bertambah. Dari hasil pengujian ini mendapatkan hasil bahwa panjang pendeknya karakter pada proses *hash* MD5 mempengaruhi waktu yang dibutuhkan, semakin bertambah satu karakter maka akan mempengaruhi waktu eksekusi, dan setiap bertambah satu blok maka waktu yag diperlukan juga semakin bertambah.

BAB VI
PENUTUP

6.1 Kesimpulan

Dari hasil penelitian didapatkan kesimpulan sebagai berikut:

1. Implementasi *hash* data teks pada *embedded system* dengan algoritma MD5 dapat menjadikan Arduino sebagai perangkat tambahan pada aplikasi atau perangkat lain yang memerlukan/data *integrity*
2. Waktu yang diperlukan pada proses *hash* sebanding dengan panjang teks yang diproses, selisih waktu antara satu blok dan blok lain adalah 0.001168 s.
3. Waktu yang diperlukan untuk *hash* data teks berukuran 1 sampai dengan 55 karakter (satu blok) memerlukan waktu antara 0.00136 s sampai 0.001396 s.
4. Untuk teks yang sama, proses *hash* yang dilakukan pada Arduino dan *server* memberikan hasil yang sama.
5. Pengujian validitas *hash* dari suatu data teks memberikan hasil 100% benar

6.2 Saran

Untuk meningkatkan hasil yang telah dicapai dari penelitian ini dapat dilakukan perbaikan sebagai berikut:

1. Hasil dari pengujian validitas dan waktu eksekusi dapat digunakan sebagai standar acuan kedepan untuk dikembangkan lebih lanjut
2. Untuk menjalankan perangkat tambahan ini dibutuhkan sebuah aplikasi atau fungsi tambahan untuk memangil dan memproses MD5 pada Arduino.
3. Proses hash data teks mengkonsumsi flash memory sebesar 26 KB dari 256 KB yang tersedia sehingga sisa dari flash memory dapat digunakan untuk proses lainnya.
4. Untuk pengembangan lebih lanjut, sistem keamanan data pada *embedded* dapat dikembangkan dengan penambahan algoritma kriptografi algoriotma asimetri, dan fungsi *hash* lainnya.

LAMPIRAN

Lampiran 1 Hasil pengujian waktu

No	Karakter yang diuji	Jumlah karakter	Hasil waktu
1.	A	1	0.00136 s
2.	Aa	2	0.00136 s
3.	Apa	3	0.00136 s
4.	Diaz	4	0.001364 s
5.	KaMus	5	0.001364 s
6.	Pelaku	6	0.001364 s
7.	Siap ya	7	0.001368 s
8.	Askop ya	8	0.001368 s
9.	Embedded	9	0.001368 s
10.	Brawijaya	10	0.001372 s
11.	Katulistiwa	11	0.001372 s
12.	katuListiwa.	12	0.001372 s
13.	Penjelasan ya	13	0.001372 s
14.	Informatika UB	14	0.001376 s
15.	Pesawat yerbang	15	0.001376 s
16.	Pasnya 42ab762ci	16	0.001376 s
17.	Penjelasan dapat.	17	0.001376 s
18.	Inisialisasi Nilai	18	0.001376 s
19.	Perancangan sistem	19	0.001376 s
20.	Perancangan sistem !	20	0.001380 s
21.	Universitas brawijaya	21	0.001380 s
22.	dari Hasil Pengujian..	22	0.001380 s
23.	09182374856970ytejkA	23	0.001380 s
24.	Penjelasan dapat dilihat	24	0.001380 s

25.	penJelasan Dapat dilihat.	25	0.001380 s
26.	langkah-langkah yang akan	26	0.001380 s
27.	pengujian validitas hasil !!	27	0.001384 s
28.	Perancangan sistem dilakukan	28	0.001384 s
29.	Segera hubungi 085259322811	29	0.001384 s
30.	Apa Saja Yang Bisa Dipakai !!	30	0.001384 s
31.	perancangan sistem, implementasi	31	0.001384 s
32.	hampir sama dengan fungsi, namun	32	0.001384 s
33.	Arsitektur dapat didesain sesuai	33	0.001384 s
34.	Bahasa pemrograman yang digunakan	34	0.001384 s
35.	Hasil dari setiap panjang karakter	35	0.001384 s
36.	Hasil dari setiap panjang karakter !	36	0.001384 s
37.	sebagai kabel yang menjadi penghubung	37	0.001384 s
38.	Diantara kedua kode tersebut memiliki	38	0.001388 s
39.	Diantara keDua kode tersebut memiliki	39	0.001388 s
40.	Variable dideklarasikan Didalam proses	40	0.001388 s
41.	Variable dideklarasikan Didalam proses	41	0.001388 s
42.	dari Hasil Pengujian didapatkan KesimpulaN	42	0.001388 s
43.	dari Hasil Pengujian didapatkan KesimpulaN.	43	0.001392 s
44.	dari Hasil Pengujian didapatkan KesimpulaN b	44	0.001392 s
45.	dari Hasil Pengujian didapatkan KesimpulaN ba	45	0.001392 s
46.	dari Hasil Pengujian didapatkan KesimpulaN bah	46	0.001392 s
47.	dari Hasil Pengujian didapatkan KesimpulaN bahw	47	0.001392 s

48.	dari Hasil Pengujian didapatkan KesimpulaN bahwa	48	0.001396 s
49.	dari Hasil Pengujian didapatkan KesimpulaN bahwa.	49	0.001396 s
50.	dari Hasil Pengujian didapatkan KesimpulaN bahwa m	50	0.001396 s
51.	dari Hasil Pengujian didapatkan KesimpulaN bahwa md	51	0.001396 s
52.	dari Hasil Pengujian didapatkan KesimpulaN bahwa md5	52	0.001396 s
53.	dari Hasil Pengujian didapatkan KesimpulaN bahwa md5.	53	0.001396 s
54.	dari Hasil Pengujian didapatkan KesimpulaN bahwa md5 .	54	0.001396 s
55.	dari Hasil Pengujian didapatkan KesimpulaN bahwa md5 d	55	0.001396 s
56.	dari Hasil Pengujian didapatkan KesimpulaN bahwa md5 da	56	0.002564 s
57.	dari Hasil Pengujian didapatkan KesimpulaN bahwa md5 dap	57	0.002564 s
58.	dari Hasil Pengujian didapatkan KesimpulaN bahwa md5 dapa	58	0.002564 s
59.	dari Hasil Pengujian didapatkan KesimpulaN bahwa md5 dapat	59	0.002564 s
60.	dari Hasil Pengujian didapatkan KesimpulaN bahwa md5 dapat.	60	0.002568 s
61.	dari Hasil Pengujian didapatkan KesimpulaN bahwa md5 dapat d	61	0.002568 s
62.	dari Hasil Pengujian didapatkan KesimpulaN bahwa md5 dapat di	62	0.002568 s

63.	dari Hasil Pengujian didapatkan KesimpulaN bahwa md5 dapat dii	63	0.002572 s
64.	dari Hasil Pengujian didapatkan KesimpulaN bahwa md5 dapat diim	64	0.002572 s
65.	dari Hasil Pengujian didapatkan KesimpulaN bahwa md5 dapat diimp	65	0.002572 s
66.	dari Hasil Pengujian didapatkan KesimpulaN bahwa md5 dapat diimpl	66	0.002572 s
67.	dari Hasil Pengujian didapatkan KesimpulaN bahwa md5 dapat diimple	67	0.002576 s
68.	dari Hasil Pengujian didapatkan KesimpulaN bahwa md5 dapat diimplem	68	0.002576 s
69.	dari Hasil Pengujian didapatkan Kesimpulan bahwa md5 dapat diimpleme	69	0.002576 s
70.	dari Hasil Pengujian didapatkan Kesimpulan bahwa md5 dapat diimplemen	70	0.002576 s
71.	dari Hasil Pengujian didapatkan Kesimpulan bahwa md5 dapat diimplement	71	0.002580 s
72.	dari Hasil Pengujian didapatkan Kesimpulan bahwa md5 dapat diimplementa	72	0.002580 s
73.	dari Hasil Pengujian didapatkan Kesimpulan bahwa md5 dapat diimplementas	73	0.002580 s

74.	dari Hasil Pengujian didapatkan Kesimpulan bahwa md5 dapat diimplementasi	74	0.002580 s
75.	dari Hasil Pengujian didapatkan Kesimpulan bahwa md5 dapat diimplementasik	75	0.002592 s
76.	dari Hasil Pengujian didapatkan Kesimpulan bahwa md5 dapat diimplementasika	76	0.002592 s
77.	dari Hasil pengujian didapatkan Kesimpulan bahwa md5 dapat diimplementasikan	77	0.002592 s
78.	dari Hasil pengujian didapatkan Kesimpulan bahwa md5 dapat diimplementasikan	78	0.002592 s
79.	dari Hasil pengujian didapatkan Kesimpulan bahwa md5 dapat diimplementasikan.	79	0.002596 s
80.	dari Hasil pengujian didapatkan Kesimpulan bahwa md5 dapat diimplementasikan p	80	0.002596 s
81.	dari Hasil pengujian didapatkan Kesimpulan bahwa md5 dapat diimplementasikan pada	81	0.002596 s
82.	dari Hasil pengujian didapatkan Kesimpulan bahwa md5 dapat diimplementasikan padae	82	0.002596 s
83.	dari Hasil pengujian didapatkan Kesimpulan bahwa md5 dapat diimplementasikan padaem	83	0.002596 s

84.	dari Hasil pengujian didapatkan Kesimpulan bahwa md5 dapat diimplementasikan padaemb	84	0.002600 s
85.	dari Hasil pengujian didapatkan Kesimpulan bahwa md5 dapat diimplementasikan padaembe	85	0.002600 s
86.	dari Hasil pengujian didapatkan Kesimpulan bahwa md5 dapat diimplementasikan padaembed	86	0.002600 s
87.	dari Hasil pengujian didapatkan Kesimpulan bahwa md5 dapat diimplementasikan padaembedd	87	0.002600 s
88.	dari Hasil pengujian didapatkan Kesimpulan bahwa md5 dapat diimplementasikan pada embedd	88	0.002600 s
89.	dari Hasil pengujian didapatkan Kesimpulan bahwa md5 dapat diimplementasikan pada embedde	89	0.002604 s
90.	dari hasil pengujian didapatkan Kesimpulan bahwa md5 dapat diimplementasikan pada embedded	90	0.002604 s
91.	Dari hasil pengujian didapatkan Kesimpulan bahwa md5 dapat diimplementasikan pada embedded	91	0.002604 s
92.	Dari hasil pengujian didapatkan Kesimpulan bahwa md5 dapat diimplementasikan pada embedded s	92	0.002604 s
93.	Dari hasil pengujian didapatkan Kesimpulan bahwa md5 dapat diimplementasikan pada embedded sy	93	0.002604 s

94.	Dari hasil pengujian didapatkan kesimpulan bahwa md5 dapat diimplementasikan pada embedded sys	94	0.002604 s
95.	Dari hasil pengujian didapatkan kesimpulan bahwa mD5 dapat diimplementasikan pada embedded syst	95	0.002608 s
96.	Dari hasil pengujian didapatkan kesimpulan bahwa MD5 dapat diimplementasikan pada embedded syste	96	0.002608 s
97.	DarI hasil pengujian didapatkan kesimpulan, bahwa MD5 dapat diimplementasikan pada embedded syste	97	0.002608 s
98.	Dari hasil pengujian didapatkan kesimpulan, bahwa MD5 dapat diimplementasikan pada embedded system	98	0.002608 s
99.	Dari hasil pengujian didapatkan kesimpulan, bahwa MD5 dapat diimplementasikan pada embedded system.	99	0.002608 s
100.	Dari hasil pengujian didapatkan kesimpulan, bahwa MD5 dapat diimplementasikan pada embedded system..	100	0.002608 s

Lampiran 2 Hasil pengujian validitas

No	Karakter yang diuji	Hasil dari system	Hasil dari server	Benar atau salah
1.	A	7fc56270e7a70fa81a5935b72eacbe29	7fc56270e7a70fa81a5935b72eacbe29	Benar
2.	Aa	98568d540134639be4655198a36614a4	98568d540134639be4655198a36614a4	Benar
3.	Apa	2822f4f8d9643b3f109a8eb24b1fedba	2822f4f8d9643b3f109a8eb24b1fedba	Benar
4.	Diaz	325ba0be16b1db8711e860e8d0c91542	325ba0be16b1db8711e860e8d0c91542	Benar
5.	KaMus	be8201c753e63a81bfa5d2a166fe048a	be8201c753e63a81bfa5d2a166fe048a	Benar
6.	Pelaku	e599835ae0a58856d5b2c4eac10cfd7a	e599835ae0a58856d5b2c4eac10cfd7a	Benar
7.	Siap ya	0f43c0d9225b2f241710f04adf9261c0	0f43c0d9225b2f241710f04adf9261c0	Benar
8.	Askop ya	919094adf38fe2dfdc696502e6951c80	919094adf38fe2dfdc696502e6951c80	Benar

9.	Embedded	069e7821d3b4c0bbf437147263eb4d0e	069e7821d3b4c0bbf437147263eb4d0e	Benar
10.	Brawijaya	7ba0cd4c4b7b4c80ccb9164b053455b4	7ba0cd4c4b7b4c80ccb9164b053455b4	Benar
11.	Katulistiwa	beedb676d6a9837bcfeaf1de3e498cb2	beedb676d6a9837bcfeaf1de3e498cb2	Benar
12.	KatuListiwa	1c3d2c4b41d2d0c12acae0305c410305	1c3d2c4b41d2d0c12acae0305c410305	Benar
13.	Penjelasan ya	baf398717e052d3f4edf0bb2217792b9	baf398717e052d3f4edf0bb2217792b9	Benar
14.	Informatika UB	3bde233576388bf3fa0d165e5b36e61b	3bde233576388bf3fa0d165e5b36e61b	Benar
15.	Pesawat terbang	cee9ef7f388702331a3596d2a92d5e14	cee9ef7f388702331a3596d2a92d5e14	Benar
16.	Pasnya 42ab762ci	6c82f29d9a5ee9ebe318d3a5f7c4bff8	6c82f29d9a5ee9ebe318d3a5f7c4bff8	Benar
17.	Penjelasan dapat.	bdf427384b77bb2ca82e60f093954c6b	bdf427384b77bb2ca82e60f093954c6b	Benar
18.	Inisialisasi Nilai	2e92d2de11b2c9303829536ab21ac00c	2e92d2de11b2c9303829536ab21ac00c	Benar

19.	Perancangan sistem	5b4cc3a4abb899fcd008e8fd478e4fbb	5b4cc3a4abb899fcd008e8fd478e4fbb	Benar
20.	Perancangan sistem !	74de3641514b5d008ae433e422fdeef8	74de3641514b5d008ae433e422fdeef8	Benar
21.	Universitas brawijaya	381dfe60a45ffc4137e7712bca90558e	381dfe60a45ffc4137e7712bca90558e	Benar
22.	dari Hasil Pengujian..	6270f8cc1d6c2bc356b1885929f52fd0	6270f8cc1d6c2bc356b1885929f52fd0	Benar
23.	09182374856970ytejkA	6ef3a5a7a0760ded1a1518b4851226c9	6ef3a5a7a0760ded1a1518b4851226c9	Benar
24.	Penjelasan dapat dilihat	0f9beef8f03ddc3d5747385a74ba9340	0f9beef8f03ddc3d5747385a74ba9340	Benar
25.	penJelasan Dapat dilihat.	ab043006a124fa394e031dd6ac51cd06	ab043006a124fa394e031dd6ac51cd06	Benar
26.	langkah-langkah yang akan	a5c9971d43ba02874474969575 9276f1	a5c9971d43ba02874474969575 9276f1	Benar
27.	pengujian validitas hasil !!	a5c9971d43ba02874474969575 9276f1	a5c9971d43ba02874474969575 9276f1	Benar
28.	Perancangan sistem dilakukan	5d7bdb94b3f241366227789086 6a22d4	5d7bdb94b3f241366227789086 6a22d4	Benar

29.	Segera hubungi 085259322811	16f67903d04f77f4bf6a64c40d96668b	16f67903d04f77f4bf6a64c40d96668b	Benar
30.	APA SAJA YANG BISA dipakai !!	7a03eb4265a14c45678477d9e397c0ef	7a03eb4265a14c45678477d9e397c0ef	Benar
31.	perancangan sistem, implementasi	52c3a4fedb9353378417cb72501d0d0b	52c3a4fedb9353378417cb72501d0d0b	Benar
32.	hampir sama dengan fungsi, namun	d63749f609ba3e01d3f8c8d36fa0acf4	d63749f609ba3e01d3f8c8d36fa0acf4	Benar
33.	Arsitektur dapat didesain sesuai	a9d79a3f0b250eb3096ecd2d241e6f33	a9d79a3f0b250eb3096ecd2d241e6f33	Benar
34.	Bahasa pemrograman yang digunakan	1aa180931071d1120a104acc7c1afc06	1aa180931071d1120a104acc7c1afc06	Benar
35.	Hasil dari setiap panjang karakter	dbf706fecdac23923dd2c58a4570a88b	dbf706fecdac23923dd2c58a4570a88b	Benar
36.	Hasil dari setiap panjang karakter !	fa581a74f49061802342197d06c7e63e	fa581a74f49061802342197d06c7e63e	Benar
37.	sebagai kabel yang menjadi penghubung	a63779fab236de21e8fe2d20271e8cdb	a63779fab236de21e8fe2d20271e8cdb	Benar
38.	Diantara kedua kode tersebut memiliki	2ec0e4d43bd6b9881da364794a6a957b	2ec0e4d43bd6b9881da364794a6a957b	Benar

39.	Diantara keDua kode tersebut memiliki	cc26170dee1513312a0534aad2389104	cc26170dee1513312a0534aad2389104	Benar
40.	Pengambilan kesimpulan dilakukan setelah	b3cec0df5ca45fc2183f15cdd71535d7	b3cec0df5ca45fc2183f15cdd71535d7	Benar
41.	Pengambilan kesimpulan dilakukan setelah.	c4e957245121645a93ed299e207a28a9	c4e957245121645a93ed299e207a28a9	Benar
42.	dari Hasil Pengujian didapatkan KesimpulaN	a3a7497c9691be3733cb9e588271b861	a3a7497c9691be3733cb9e588271b861	Benar
43.	dari Hasil Pengujian didapatkan KesimpulaN.	c4ab45108281ae007028473f7f1a09ea	c4ab45108281ae007028473f7f1a09ea	Benar
44.	dari Hasil Pengujian didapatkan KesimpulaN b	e74ad2bf538a1ff703c8e6b7e03a92b1	e74ad2bf538a1ff703c8e6b7e03a92b1	Benar
45.	dari Hasil Pengujian didapatkan KesimpulaN ba	5828f391a60e818c8c64de8ace32f362	5828f391a60e818c8c64de8ace32f362	Benar
46.	dari Hasil Pengujian didapatkan KesimpulaN bah	75a77844ea45b36d9f57962b822f7147	75a77844ea45b36d9f57962b822f7147	Benar
47.	dari Hasil Pengujian didapatkan KesimpulaN bahw	96396de5c6be97a4aea7fad902c4bb50	96396de5c6be97a4aea7fad902c4bb50	Benar
48.	dari Hasil Pengujian didapatkan KesimpulaN bahwa	3da9743cce30ef29750f3137b6d4f825	3da9743cce30ef29750f3137b6d4f825	Benar

49.	dari Hasil Pengujian didapatkan KesimpulaN bahwa.	67587bd7d4d4d59065471fd8f91fe905	67587bd7d4d4d59065471fd8f91fe905	Benar
50.	dari Hasil Pengujian didapatkan KesimpulaN bahwa m	98675de4ce3f9faa55f1e16ae2c9e744	98675de4ce3f9faa55f1e16ae2c9e744	Benar
51.	dari Hasil Pengujian didapatkan KesimpulaN bahwa md	a51fac3b8ec4b336a02acf2ee637112d	a51fac3b8ec4b336a02acf2ee637112d	Benar
52.	dari Hasil Pengujian didapatkan KesimpulaN bahwa md5	db7341c974f9a22921a1dfbcd83d4f2c	db7341c974f9a22921a1dfbcd83d4f2c	Benar
53.	dari Hasil Pengujian didapatkan KesimpulaN bahwa md5.	1547d668601af66ec1f4b0f3b4737512	1547d668601af66ec1f4b0f3b4737512	Benar
54.	dari Hasil Pengujian didapatkan KesimpulaN bahwa md5 .	dd376e8aa7c2dbee2abb945140b610f9	dd376e8aa7c2dbee2abb945140b610f9	Benar
55.	dari Hasil Pengujian didapatkan KesimpulaN bahwa md5 d	32db6fe997486c7a32302afd0d6c11c2	32db6fe997486c7a32302afd0d6c11c2	Benar

56.	dari Hasil Pengujian didapatkan KesimpulaN bahwa md5 da	fd542c96412a8fb7a894de0a47286063	fd542c96412a8fb7a894de0a47286063	Benar
57.	dari Hasil Pengujian didapatkan KesimpulaN bahwa md5 dap	93d83975f0af584564d804a08a859cb5	93d83975f0af584564d804a08a859cb5	Benar
58.	dari Hasil Pengujian didapatkan KesimpulaN bahwa md5 dapa	42cca7ca4657b4e642bddd651375545b	42cca7ca4657b4e642bddd651375545b	Benar
59.	dari Hasil Pengujian didapatkan KesimpulaN bahwa md5 dapat	32f32b87114b75d96039c7d3e1c52221	32f32b87114b75d96039c7d3e1c52221	Benar
60.	dari Hasil Pengujian didapatkan KesimpulaN bahwa md5 dapat.	c4c84e65c4e4bb93fd8c9e0d0bff7b7c	c4c84e65c4e4bb93fd8c9e0d0bff7b7c	Benar
61.	dari Hasil Pengujian didapatkan KesimpulaN bahwa md5 dapat d	0e76f902a99971315c26dd0e2a3d125d	0e76f902a99971315c26dd0e2a3d125d	Benar
62.	dari Hasil Pengujian didapatkan KesimpulaN bahwa md5 dapat di	16e2aff1d5ae37439fee8a0ad3015e49	16e2aff1d5ae37439fee8a0ad3015e49	Benar

63.	dari Hasil Pengujian didapatkan KesimpulaN bahwa md5 dapat dii	8d5ebe327afdc8bd6080346a6884a84b	8d5ebe327afdc8bd6080346a6884a84b	Benar
64.	dari Hasil Pengujian didapatkan KesimpulaN bahwa md5 dapat diim	286bdf33b93103d06db02a7e5eaf7c1e	286bdf33b93103d06db02a7e5eaf7c1e	Benar
65.	dari Hasil Pengujian didapatkan KesimpulaN bahwa md5 dapat diimp	65ac4c5f6b6745303ff1e1410d9eb664	65ac4c5f6b6745303ff1e1410d9eb664	Benar
66.	dari Hasil Pengujian didapatkan KesimpulaN bahwa md5 dapat diimpl	c17c2f0b66fd5ad3791d5dfca3115bf2	c17c2f0b66fd5ad3791d5dfca3115bf2	Benar
67.	dari Hasil Pengujian didapatkan KesimpulaN bahwa md5 dapat diimple	d0f2cfb2e9dec85f7a5697d7c619b66b	d0f2cfb2e9dec85f7a5697d7c619b66b	Benar
68.	dari Hasil Pengujian didapatkan KesimpulaN bahwa md5 dapat diimplem	6f519e2b3bfb4192f8a2c1d26df0bd95	6f519e2b3bfb4192f8a2c1d26df0bd95	Benar
69.	dari Hasil Pengujian didapatkan Kesimpulan bahwa md5 dapat diimpleme	c227487c38b8f1e50dec4fd867366cf8	c227487c38b8f1e50dec4fd867366cf8	Benar

70.	dari Hasil Pengujian didapatkan Kesimpulan bahwa md5 dapat diimplemen	5001631313c8b48618cc0772c8b18953	5001631313c8b48618cc0772c8b18953	Benar
71.	dari Hasil Pengujian didapatkan Kesimpulan bahwa md5 dapat diimplement	bbb265c9b240dc6e7f791c8c31012f3c	bbb265c9b240dc6e7f791c8c31012f3c	Benar
72.	dari Hasil Pengujian didapatkan Kesimpulan bahwa md5 dapat diimplementa	1f0738e8d4c355ddfb10a79ffff8ea51	1f0738e8d4c355ddfb10a79ffff8ea51	Benar
73.	dari Hasil Pengujian didapatkan Kesimpulan bahwa md5 dapat diimplementas	912553a7f2c981a2938a168f4561c1ac	912553a7f2c981a2938a168f4561c1ac	Benar
74.	dari Hasil Pengujian didapatkan Kesimpulan bahwa md5 dapat diimplementasi	81dc5417858397aea8405aa19a07fe0d	81dc5417858397aea8405aa19a07fe0d	Benar

75.	dari Hasil Pengujian didapatkan Kesimpulan bahwa md5 dapat diimplementasik	6fa5d650ba3114b53440b6c0c3ef4192	6fa5d650ba3114b53440b6c0c3ef4192	Benar
76.	dari Hasil Pengujian didapatkan Kesimpulan bahwa md5 dapat diimplementasika	f441998307fdd16ad3975e82dd6024b8	f441998307fdd16ad3975e82dd6024b8	Benar
77.	dari Hasil pengujian didapatkan Kesimpulan bahwa md5 dapat diimplementasikan	6bbdc1e1840e8f07c3e883e86f462bc9	6bbdc1e1840e8f07c3e883e86f462bc9	Benar
78.	dari Hasil pengujian didapatkan Kesimpulan bahwa md5 dapat diimplementasikan	747331bb6501b58c2710a09695ed32bf	747331bb6501b58c2710a09695ed32bf	Benar
79.	dari Hasil pengujian didapatkan Kesimpulan bahwa md5 dapat diimplementasikan.	7237330c400f1cfeabf631eaacbef231	7237330c400f1cfeabf631eaacbef231	Benar

80.	dari Hasil pengujian didapatkan Kesimpulan bahwa md5 dapat diimplementasikan p	75840a2e5eb4fd38201c521dd4b7e739	75840a2e5eb4fd38201c521dd4b7e739	Benar
81.	dari Hasil pengujian didapatkan Kesimpulan bahwa md5 dapat diimplementasikan pada	5135fea2d0625aaa96081b794dd9aaa6	5135fea2d0625aaa96081b794dd9aaa6	Benar
82.	dari Hasil pengujian didapatkan Kesimpulan bahwa md5 dapat diimplementasikan padae	4a312556004d8250a8e92f5eeb3d2ebf	4a312556004d8250a8e92f5eeb3d2ebf	Benar
83.	dari Hasil pengujian didapatkan Kesimpulan bahwa md5 dapat diimplementasikan padaem	d0f56e1949953f7362cbe3ca3782aa39	d0f56e1949953f7362cbe3ca3782aa39	Benar

84.	dari Hasil pengujian didapatkan Kesimpulan bahwa md5 dapat diimplementasikan padaemb	a2b137a8bac84a349d016e09cd7ac838	a2b137a8bac84a349d016e09cd7ac838	Benar
85.	dari Hasil pengujian didapatkan Kesimpulan bahwa md5 dapat diimplementasikan padaembe	abea7a1c103a4f4f38013082eb346659	abea7a1c103a4f4f38013082eb346659	Benar
86.	dari Hasil pengujian didapatkan Kesimpulan bahwa md5 dapat diimplementasikan padaembed	f78ea6cb31d4bca3e1507524ed260207	f78ea6cb31d4bca3e1507524ed260207	Benar
87.	dari Hasil pengujian didapatkan Kesimpulan bahwa md5 dapat diimplementasikan padaembedd	84ee766575bab9b00dad5d62c93a3aaa	84ee766575bab9b00dad5d62c93a3aaa	Benar

88.	dari Hasil pengujian didapatkan Kesimpulan bahwa md5 dapat diimplementasikan pada embed	9f13ae66a3d1ced6f0708bedfa04c160	9f13ae66a3d1ced6f0708bedfa04c160	Benar
89.	dari Hasil pengujian didapatkan Kesimpulan bahwa md5 dapat diimplementasikan pada embedded	8a97d13702070e002a8961016bbc4d2d	8a97d13702070e002a8961016bbc4d2d	Benar
90.	dari hasil pengujian didapatkan Kesimpulan bahwa md5 dapat diimplementasikan pada embedded	ca794a63a3dbe25015b09549a5cfa22e	ca794a63a3dbe25015b09549a5cfa22e	Benar
91.	Dari hasil pengujian didapatkan Kesimpulan bahwa md5 dapat diimplementasikan pada embedded	ca794a63a3dbe25015b09549a5cfa22e	ca794a63a3dbe25015b09549a5cfa22e	Benar
92.	Dari hasil pengujian didapatkan Kesimpulan bahwa md5 dapat diimplementasikan pada embedded s	e36522a0f92b2285a1bb73fbc9fe7047	e36522a0f92b2285a1bb73fbc9fe7047	Benar

93.	Dari hasil pengujian didapatkan Kesimpulan bahwa md5 dapat diimplementasikan pada embedded sy	e3c9995518d997672ae27f5dafd5e029	e3c9995518d997672ae27f5dafd5e029	Benar
94.	Dari hasil pengujian didapatkan kesimpulan bahwa md5 dapat diimplementasikan pada embedded sys	556f3d2bcb3f14c821327d294e3225d4	556f3d2bcb3f14c821327d294e3225d4	Benar
95.	Dari hasil pengujian didapatkan kesimpulan bahwa mD5 dapat diimplementasikan pada embedded syst	84411d5d39acf13c1ae355dec30c891b	84411d5d39acf13c1ae355dec30c891b	Benar
96.	Dari hasil pengujian didapatkan kesimpulan bahwa MD5 dapat diimplementasikan pada embedded syste	51ecf31f1f394d4d83cac3c987cff616	51ecf31f1f394d4d83cac3c987cff616	Benar
97.	DarI hasil pengujian didapatkan kesimpulan, bahwa MD5 dapat diimplementasikan pada embedded syste	8adaa0abc8fa479d439fde92915febe2	8adaa0abc8fa479d439fde92915febe2	Benar

98.	Dari hasil pengujian didapatkan kesimpulan, bahwa MD5 dapat diimplementasikan pada embedded system	43da1e08f4a52efc2a558fcd3e742004	43da1e08f4a52efc2a558fcd3e742004	Benar
99.	Dari hasil pengujian didapatkan kesimpulan, bahwa MD5 dapat diimplementasikan pada embedded system.	de74bb67ae9de4e881f0d973282f5d87	de74bb67ae9de4e881f0d973282f5d87	Benar
100.	Dari hasil pengujian didapatkan kesimpulan, bahwa MD5 dapat diimplementasikan pada embedded system..	32362862af1f3cb722917af4b2e79e22	32362862af1f3cb722917af4b2e79e22	Benar

Lampiran 3 MD5.CPP

```
#include "MD5.h"

MD5::MD5()
{
      //nothing
      return;
}
char* MD5::make_digest(const unsigned char *digest, int
len) /* {{{ */
{
      char * md5str = (char*)
malloc(sizeof(char)*(len*2+1));
      static const char hexits[17] =
"0123456789abcdef";
      int i;

      for (i = 0; i < len; i++) {
          md5str[i * 2]       = hexits[digest[i] >> 4];
          md5str[(i * 2) + 1] = hexits[digest[i] &
0x0F];
      }
      md5str[len * 2] = '\0';
      return md5str;
}


#define F(x, y, z)                  ((z) ^ ((x) & ((y)
^ (z))))
#define G(x, y, z)                  ((y) ^ ((z) & ((x)
^ (y))))
#define H(x, y, z)                  ((x) ^ (y) ^ (z))
#define I(x, y, z)                  ((y) ^ ((x) |
~(z)))

 * The MD5 transformation for all four rounds.
#define STEP(f, a, b, c, d, x, t, s) \
```

```
37        (a) += f((b), (c), (d)) + (x) + (t); \
38        (a) = (((a) << (s)) | (((a) & 0xffffffff) >> (32
39  - (s)))); \
40        (a) += (b);
41
42  #if defined(__i386__) || defined(__x86_64__) ||
43  defined(__vax__)
44  # define SET(n) \
45        (*(MD5_u32plus *)&ptr[(n) * 4])
46  # define GET(n) \
47        SET(n)
48  #else
49  # define SET(n) \
50        (ctx->block[(n)] = \
51        (MD5_u32plus)ptr[(n) * 4] | \
52        ((MD5_u32plus)ptr[(n) * 4 + 1] << 8) | \
53        ((MD5_u32plus)ptr[(n) * 4 + 2] << 16) | \
54        ((MD5_u32plus)ptr[(n) * 4 + 3] << 24))
55  # define GET(n) \
56        (ctx->block[(n)])
57  #endif
58
59  const void *MD5::body(void *ctxBuf, const void *data,
60  size_t size)
61  {
62        MD5_CTX *ctx = (MD5_CTX*)ctxBuf;
63        const unsigned char *ptr;
64        MD5_u32plus a, b, c, d;
65        MD5_u32plus saved_a, saved_b, saved_c, saved_d;
66
67        ptr = (unsigned char*)data;
68
69        a = ctx->a;
70        b = ctx->b;
71        c = ctx->c;
72        d = ctx->d;
73
74        do {
```

```
 75              saved_a = a;
 76              saved_b = b;
 77              saved_c = c;
 78              saved_d = d;
 79
 80      /* Round 1 */
 81              STEP(F, a, b, c, d, SET(0), 0xd76aa478, 7)
 82              STEP(F, d, a, b, c, SET(1), 0xe8c7b756, 12)
 83              STEP(F, c, d, a, b, SET(2), 0x242070db, 17)
 84              STEP(F, b, c, d, a, SET(3), 0xc1bdceee, 22)
 85              STEP(F, a, b, c, d, SET(4), 0xf57c0faf, 7)
 86              STEP(F, d, a, b, c, SET(5), 0x4787c62a, 12)
 87              STEP(F, c, d, a, b, SET(6), 0xa8304613, 17)
 88              STEP(F, b, c, d, a, SET(7), 0xfd469501, 22)
 89              STEP(F, a, b, c, d, SET(8), 0x698098d8, 7)
 90              STEP(F, d, a, b, c, SET(9), 0x8b44f7af, 12)
 91              STEP(F, c, d, a, b, SET(10), 0xffff5bb1, 17)
 92              STEP(F, b, c, d, a, SET(11), 0x895cd7be, 22)
 93              STEP(F, a, b, c, d, SET(12), 0x6b901122, 7)
 94              STEP(F, d, a, b, c, SET(13), 0xfd987193, 12)
 95              STEP(F, c, d, a, b, SET(14), 0xa679438e, 17)
 96              STEP(F, b, c, d, a, SET(15), 0x49b40821, 22)
 97
 98      /* Round 2 */
 99              STEP(G, a, b, c, d, GET(1), 0xf61e2562, 5)
100              STEP(G, d, a, b, c, GET(6), 0xc040b340, 9)
101              STEP(G, c, d, a, b, GET(11), 0x265e5a51, 14)
102              STEP(G, b, c, d, a, GET(0), 0xe9b6c7aa, 20)
103              STEP(G, a, b, c, d, GET(5), 0xd62f105d, 5)
104              STEP(G, d, a, b, c, GET(10), 0x02441453, 9)
105              STEP(G, c, d, a, b, GET(15), 0xd8a1e681, 14)
106              STEP(G, b, c, d, a, GET(4), 0xe7d3fbc8, 20)
107              STEP(G, a, b, c, d, GET(9), 0x21e1cde6, 5)
108              STEP(G, d, a, b, c, GET(14), 0xc33707d6, 9)
109              STEP(G, c, d, a, b, GET(3), 0xf4d50d87, 14)
110              STEP(G, b, c, d, a, GET(8), 0x455a14ed, 20)
111              STEP(G, a, b, c, d, GET(13), 0xa9e3e905, 5)
112              STEP(G, d, a, b, c, GET(2), 0xfcefa3f8, 9)
```

```
113             STEP(G, c, d, a, b, GET(7), 0x676f02d9, 14)
114             STEP(G, b, c, d, a, GET(12), 0x8d2a4c8a, 20)
115
116     /* Round 3 */
117             STEP(H, a, b, c, d, GET(5), 0xfffa3942, 4)
118             STEP(H, d, a, b, c, GET(8), 0x8771f681, 11)
119             STEP(H, c, d, a, b, GET(11), 0x6d9d6122, 16)
120             STEP(H, b, c, d, a, GET(14), 0xfde5380c, 23)
121             STEP(H, a, b, c, d, GET(1), 0xa4beea44, 4)
122             STEP(H, d, a, b, c, GET(4), 0x4bdecfa9, 11)
123             STEP(H, c, d, a, b, GET(7), 0xf6bb4b60, 16)
124             STEP(H, b, c, d, a, GET(10), 0xbebfbc70, 23)
125             STEP(H, a, b, c, d, GET(13), 0x289b7ec6, 4)
126             STEP(H, d, a, b, c, GET(0), 0xeaa127fa, 11)
127             STEP(H, c, d, a, b, GET(3), 0xd4ef3085, 16)
128             STEP(H, b, c, d, a, GET(6), 0x04881d05, 23)
129             STEP(H, a, b, c, d, GET(9), 0xd9d4d039, 4)
130             STEP(H, d, a, b, c, GET(12), 0xe6db99e5, 11)
131             STEP(H, c, d, a, b, GET(15), 0x1fa27cf8, 16)
132             STEP(H, b, c, d, a, GET(2), 0xc4ac5665, 23)
133
134     /* Round 4 */
135             STEP(I, a, b, c, d, GET(0), 0xf4292244, 6)
136             STEP(I, d, a, b, c, GET(7), 0x432aff97, 10)
137             STEP(I, c, d, a, b, GET(14), 0xab9423a7, 15)
138             STEP(I, b, c, d, a, GET(5), 0xfc93a039, 21)
139             STEP(I, a, b, c, d, GET(12), 0x655b59c3, 6)
140             STEP(I, d, a, b, c, GET(3), 0x8f0ccc92, 10)
141             STEP(I, c, d, a, b, GET(10), 0xffeff47d, 15)
142             STEP(I, b, c, d, a, GET(1), 0x85845dd1, 21)
143             STEP(I, a, b, c, d, GET(8), 0x6fa87e4f, 6)
144             STEP(I, d, a, b, c, GET(15), 0xfe2ce6e0, 10)
145             STEP(I, c, d, a, b, GET(6), 0xa3014314, 15)
146             STEP(I, b, c, d, a, GET(13), 0x4e0811a1, 21)
147             STEP(I, a, b, c, d, GET(4), 0xf7537e82, 6)
148             STEP(I, d, a, b, c, GET(11), 0xbd3af235, 10)
149             STEP(I, c, d, a, b, GET(2), 0x2ad7d2bb, 15)
150             STEP(I, b, c, d, a, GET(9), 0xeb86d391, 21)
```

```
151
152             a += saved_a;
153             b += saved_b;
154             c += saved_c;
155             d += saved_d;
156
157             ptr += 64;
158         } while (size -= 64);
159
160         ctx->a = a;
161         ctx->b = b;
162         ctx->c = c;
163         ctx->d = d;
164
165         return ptr;
166 }
167
168 void MD5::MD5Init(void *ctxBuf)
169 {
170         MD5_CTX *ctx = (MD5_CTX*)ctxBuf;
171         ctx->a = 0x67452301;
172         ctx->b = 0xefcdab89;
173         ctx->c = 0x98badcfe;
174.        ctx->d = 0x10325476;
175
176         ctx->lo = 0;
177         ctx->hi = 0;
178 }
179
180 void MD5::MD5Update(void *ctxBuf, const void *data,
181 size_t size)
182 {
183         MD5_CTX *ctx = (MD5_CTX*)ctxBuf;
184         MD5_u32plus saved_lo;
185         MD5_u32plus used, free;
186
187         saved_lo = ctx->lo;
188         if ((ctx->lo = (saved_lo + size) & 0x1fffffff) <
```

```
189 saved_lo) {
190         ctx->hi++;
191     }
192     ctx->hi += size >> 29;
193
194     used = saved_lo & 0x3f;
195
196     if (used) {
197         free = 64 - used;
198
199         if (size < free) {
200           memcpy(&ctx->buffer[used], data, size);
201           return;
202         }
203
204         memcpy(&ctx->buffer[used], data, free);
205         data = (unsigned char *)data + free;
206         size -= free;
207         body(ctx, ctx->buffer, 64);
208     }
209
210     if (size >= 64) {
211         data = body(ctx, data, size & ~(size_t)0x3f);
212         size &= 0x3f;
213     }
214
215     memcpy(ctx->buffer, data, size);
216 }
217
218 void MD5::MD5Final(unsigned char *result, void *ctxBuf)
219 {
220     MD5_CTX *ctx = (MD5_CTX*)ctxBuf;
221     MD5_u32plus used, free;
222
223     used = ctx->lo & 0x3f;
224
225     ctx->buffer[used++] = 0x80;
226
```

```
227         free = 64 - used;
228
229         if (free < 8) {
230             memset(&ctx->buffer[used], 0, free);
231             body(ctx, ctx->buffer, 64);
232             used = 0;
233             free = 64;
234         }
235
236         memset(&ctx->buffer[used], 0, free - 8);
237
238         ctx->lo <<= 3;
239         ctx->buffer[56] = ctx->lo;
240         ctx->buffer[57] = ctx->lo >> 8;
241         ctx->buffer[58] = ctx->lo >> 16;
242         ctx->buffer[59] = ctx->lo >> 24;
243         ctx->buffer[60] = ctx->hi;
244         ctx->buffer[61] = ctx->hi >> 8;
245         ctx->buffer[62] = ctx->hi >> 16;
246         ctx->buffer[63] = ctx->hi >> 24;
247
248         body(ctx, ctx->buffer, 64);
249
250         result[0] = ctx->a;
251         result[1] = ctx->a >> 8;
252         result[2] = ctx->a >> 16;
253         result[3] = ctx->a >> 24;
254         result[4] = ctx->b;
255         result[5] = ctx->b >> 8;
256         result[6] = ctx->b >> 16;
257         result[7] = ctx->b >> 24;
258         result[8] = ctx->c;
259         result[9] = ctx->c >> 8;
260         result[10] = ctx->c >> 16;
261         result[11] = ctx->c >> 24;
262         result[12] = ctx->d;
263         result[13] = ctx->d >> 8;
264         result[14] = ctx->d >> 16;
```

```
265         result[15] = ctx->d >> 24;
266
267         memset(ctx, 0, sizeof(*ctx));
268 }
269 unsigned char* MD5::make_hash(char *arg)
270 {
271         MD5_CTX context;
272         unsigned char digest[16];
273         MD5Init(&context);
274         MD5Update(&context, arg, strlen(arg));
275         MD5Final(digest, &context);
276         return digest;
277 }
278
279
280
```

Lampiran 5 MD5.H

```
#ifndef MD5_h
#define MD5_h

#include "Arduino.h"
#include <string.h>

typedef unsigned long MD5_u32plus;

typedef struct {
      MD5_u32plus lo, hi;
      MD5_u32plus a, b, c, d;
      unsigned char buffer[64];
      MD5_u32plus block[16];
} MD5_CTX;

class MD5
{
public:
      MD5();
      static unsigned char* make_hash(char *arg);
      static  char*  make_digest(const  unsigned  char
*digest, int len);
      static const void *body(void *ctxBuf, const void
*data, size_t size);
      static void MD5Init(void *ctxBuf);
      static void MD5Final(unsigned char *result, void
*ctxBuf);
      static  void  MD5Update(void  *ctxBuf,  const  void
*data, size_t size);
};

#endif
```

DAFTAR PUSTAKA

[ADE-06] ADELSBACH, André; HUBER, Ulrich; SADEGHI, Ahmad-Reza. Secure software delivery and installation in embedded systems.In: Embedded Security in Cars. Springer Berlin Heidelberg, 2006. p. 27-49.

[ANH-10] ANHAR, S. T. Panduan Menguasai PHP dan MySQL Secara Otodidak. Mediakita: Jakarta, 2010.

[BAH-12] Bahri, Saipul, Diana Diana, and P. S. Dian."STUDI DAN IMPLEMENTASI PENGAMANAN BASIS DATA MENGGUNAKAN METODE ENKRIPSI MD5." SKRIPSI MAHASISWA TI S1 (2012).

[CEV-02] CEVOLI, Paul. Embedded freeBSD cookbook. Access Online via Elsevier, 2002.

[GEO-12] Georgitzikis, Vasilis. Arduino MD5 Library. Retrieved from Arduino Playground: http://playground.arduino.cc/Main/GeneralCodeLibrary,2012. (Diakses pada tanggal 12 juni 2013)

[MAR-11] Margolis, Michael. Arduino cookbook. O'Reilly Media, 2011.

[MEN-10] MENEZES, Alfred J.; VAN OORSCHOT, Paul C.;VANSTONE, Scott A. Handbook of applied cryptography. CRC press, 2010

[MUN-06] MUNIR, Rinaldi. Kriptografi. Informatika, Bandung, 2006.

[OKT-10] OKTAVIAN, Diar Puji. Menjadi Programmer Jempolan Menggunakan PHP. Penerbit Mediakom, 2010.

[ORG-11] ORG, JSON. JSON IN JAVASCRIPT. 2011. 2011.

[PAA-06] PAAR, Christof. Embedded IT Security in Automotive Application—An Emerging Area.In: Embedded Security in Cars. Springer Berlin Heidelberg, 2006. p. 3-13. (3-4 SATU BUKU)

[PAM-06] Pamungkas, Angger Ardyanto, M. Ary Murti, and M.

Ramdhani. "IMPLEMENTASI ALGORITMA SISTEM KRIPTOGRAFI MD5, SHA1, DAN RC4 PADA APLIKASI MOBILE INTERNET BERBASIS JAVA." Jurnal Penelitian dan Pengembangan TELEKOMUNIKASI 11, no. 1 (2006).

[RAH-02] RAHARDJO, Budi. Keamanan Sistem Informasi Berbasis Internet. *PT Insan Komunikasi Indonesia, Bandung*, 2002.

[RHE-03] RHEE, Man Young. Internet security: cryptographic principles, algorithms and protocols. Wiley. com, 2003.

[RIV-92] RIVEST, Ronald. The MD5 message-digest algorithm. 1992.

[SED-13] Sediyono, Eko, and Kartika Imam Santoso. Secure login by using One-time Password authentication based on MD5 Hash encrypted SMS. In Advances in Computing, Communications and Informatics (ICACCI), 2013 International Conference on, pp.

[SRI-10] Sri Esti, T. S., Nikmatus Sholihah, and STMIK Pradnya Paramita STIMATA Malang. "Penerapan Teknik Digital Watermarking untuk perlindungan Citra Digital BITMAP IMAGE dengan metode LSB dan MDS." Jurnal DINAMIKA DOTCOM Vol 1, no. 1 (2010).