TUGAS AKHIR – TF 145565

SISTEM MONITORING VARIABEL PROSES PADA RANCANG BANGUN ALAT PRODUKSI BAHAN BAKAR MINYAK (BBM) DARI LIMBAH PLASTIK MENGGUNAKAN HMI *(HUMAN MACHINE INTERFACE)*

AYU SAFITRI
NRP. 10 51 15 000 00 090

DOSEN PEMBIMBING
Arief Abdurrakhman, S.T., M.T.
NIP . 19870712 201404 1 002

Murry Raditya, S.T., M.T.
NPP . 1988201711055

PROGRAM STUDI D3 TEKNOLOGI INSTRUMENTASI
DEPARTEMEN TEKNIK INSTRUMENTASI
Fakultas Vokasi
Institut Teknologi Sepuluh Nopember Surabaya
2018

FINAL PROJECT– **TF 145565**

MONITORING SYSTEM VARIABLE PROCCES IN PROTOTYPE OF FUEL OIL PRODUCTION FROM PLASTIC WASTE WITH HUMAN MACHINE INTERFACE (HMI)

AYU SAFITRI
NRP. 10 51 15 000 00 090

ADVISOR LECTURE
Arief Abdurrakhman, S.T., M.T.
NIP . 19870712 201404 1 002

Murry Raditya, S.T., M.T.
NPP . 1988201711055

STUDY PROGRAM OF D3 INTRUMENTATION ENGINEERING
DEPARTMENT OF INSTRUMENTATION ENGINEERING
Faculty Of Vocation
Sepuluh November Institude of Technology
Surabaya
2018

LEMBAR PENGESAHAN I
"SISTEM MONITORING VARIABEL PROSES PADA RANCANG BANGUN ALAT PRODUKSI BAHAN BAKAR MINYAK (BBM) DARI LIMBAH PLASTIK DENGAN HMI *(HUMAN MACHINE INTERFACE)*"

TUGAS AKHIR
Oleh:
AYU SAFITRI
NRP. 10511500000090

Surabaya, 26 Juli 2018
Menyetujui,

Dosen Pembimbing I	Dosen Pembimbing II
Arief Abdurrakhman, S.T., M.T.	**Murry Raditya, S.T., M.T.**
NIP. 19870712 201404 1 002	**NPP .1988201711055**

Kepala Departemen
Teknik Instrumentasi FV-ITS

Dr. Ir. Purwadi Agus Darwito, M.Sc.
NIP. 19620822 198803 1 001

LEMBAR PENGESAHAN II
"SISTEM MONITORING VARIABEL PROSES PADA RANCANG BANGUN ALAT PRODUKSI BAHAN BAKAR MINYAK (BBM) DARI LIMBAH PLASTIK DENGAN HMI *(HUMAN MACHINE INTERFACE)*"

TUGAS AKHIR
Diajukan untuk memenuhi salah satu syarat memperoleh gelar Ahli Madya pada D3 Teknologi Instrumentasi, Fakultas Vokasi - Institut Teknologi Sepuluh Nopember.
Oleh:
AYU SAFITRI
NRP. 10511500000090

Disetujui oleh Tim Penguji Tugas Akhir:

1. Arief Abdurrakhman, S.T., M.T. .....(Dosen Pembimbing I)
2. Murry Raditya, S.T., M.T. .....(Dosen Pembimbing II)
3. Dwi Oktavianto W.N., S.T.,M.T .....(Dosen Penguji)
4. Ahmad Fauzan Adziimaa, S.T., M. Sc. .....(Dosen Penguji)

"SISTEM MONITORING VARIABEL PROSES PADA RANCANG BANGUN ALAT PRODUKSI BAHAN BAKAR MINYAK (BBM) DARI LIMBAH PLASTIK DENGAN HMI *(HUMAN MACHINE INTERFACE)*"

Nama : Ayu Safitri
NRP : 10511500000090
Departemen : Teknik Instrumentasi, FV-ITS
Dosen Pembimbing I : Arief Abdurrakhman, S.T., M.T.
Dosen Pembimbing II : Murry Raditya , S.T., M.T.

ABSTRAK

Negara Indonesia dengan jumlah penduduk terbanyak yang menduduki posisi empat dunia. Permasalahan dapat ditanggulangi dengan alat produksi bahan bakar minyak dari limbah plastik menggunakan metode pemanasan pirolisis. Alat ini memiliki *crusher* sebagai pencacah plastik, ruang pembakaran, 2 *condenser*, dan *4 storage* untuk proses *packaging*.Salah satu bagian terpenting pada proses produksi limbah plastik menjadi Bahan Bakar Minyak (BBM) ialah sistem monitoring yang akan menjadi alat safety maupun monitor jangka panjang pada alat produksi BBM ini dengan sistem data logger maupun sistem HMI (Human Machine Interface). Pada saat ini perkembangan zaman didunia komunikasi sangat membantu dalam proses ini, sehingga mendukung semua aspek yang terdapat didalamnya. Data yang diperoleh untuk satu kali produksi yaitu 10000 data yang menghabiskan memori sebanyak 400 Kb. Waktu yang dibutuhkan untuk memanaskan menurut data yang diterima berkisar antara 10-11 menit dimulai dengan awal suhu berkisar 20,00 ^{0}C – 30,00 ^{0}C. Masih terjadi error pada pengambilan data tetapi masih bisa secara realtime pada tampilan HMI (Human Machine Interface)

Kata Kunci : Sistem Monitoring, HMI (Human Machine Interface), Data Logger, Realtime

MONITORING SYSTEM VARIABLE PROCCES IN PROTOTYPE OF FUEL OIL PRODUCTION FROM PLASTIC WASTE WITH HUMAN MACHINE INTERFACE

Name **: Ayu Safitri**
NRP **: 10511500000090**
Department **: *Instrumentation Engineering* - ITS**
Supervisor I **: Arief Abdurrakhman, S.T., M.T.**
Supervisor II **: Murry Raditya, S.T., M.T.**

ABSTRACT

State of Indonesia with the most populations that occupy the world's fourth position. The use of plastic waste treatment methods is still not able to reduce the volume of plastic waste in Indonesia. Problems can be overcome by means of production of fuel oil from waste plastic using pyrolysis heating method. This tool has a crusher as a plastic counter, combustion chamber, 2 condenser, and 4 storage for the packaging process. One of the most important parts in the process of producing waste plastic into fuel oil (BBM) is a monitoring system that will be a tool of safety and long-term monitor on this BBM production tool with data logger system and HMI (Human Machine Interface) system. At this time the development of the world of communication is very helpful in this process, so that supports all aspects contained therein. The data obtained for a one-time production of 10000 data spent 400 Kb of memory. The time required to heat according to the received data ranges from 10-11 minutes starting with the initial temperature ranging from 20.00 ^{0}C - 30.00 ^{0}C. Still an error on the data retrieval but can still be realtime on the look of HMI (Human Machine Interface)

Keyword **:** Monitoring System, HMI (Human Machine Interface), Logger System

KATA PENGANTAR

Puji syukur kehadirat Allah SWT karena atas limpahan rahmat serta hidayah-Nya sehingga dapat menyelesaikan Laporan Tugas Akhir yang diselenggarakan oleh Departemen Teknik Instrumentasi, Fakultas Vokasi ITS dalam memenuhi mata kuliah Tugas Akhir dengan tepat waktu.

Dalam laporan ini membahas tentang **SISTEM MONITORING VARIABEL PROSES PADA RANCANG BANGUN ALAT PRODUKSI BAHAN BAKAR MINYAK (BBM) DARI LIMBAH PLASTIK DENGAN HMI *(HUMAN MACHINE INTERFACE)*.** Dalam kesempatan kali ini penulis mengucapkan terimakasih kepada:

1. Allah SWT yang telah memberikan rahmat dan hidayah-Nya sehingga laporan ini dapat terselesaikan tepat pada waktunya.
2. Kedua orangtua dan keluarga yang sudah mendukung dan selalu mendoakan penulis.
3. Bapak Dr. Ir. Purwadi Agus Darwito, M.Sc selaku Kepala Departemen Teknik Instrumentasi, FV-ITS
4. Bapak Arief Abdurrakhman, S.T., M.T. selaku dosen pembimbing pertama tugas akhir. Terimakasih banyak pak untuk waktu dan tenaganya. Salut sama bapak!
5. Bapak Murry Raditya S.T., M.T. selaku dosen pembimbing kedua tugas akhir. Terimakasih bimbingan malam dan bermanfaatnya pak. Bapak pembimbimbing terbaik!
6. Bapak dan Ibu Dosen Teknik Instrumentasi yang telah memberikan ilmu selama kuliah
7. Bapak dan Ibu Dosen Teknik Fisika yang telah memberikan ilmu selama kuliah di Teknik Fisika
8. Bapak Tri dari Madiun sebagai pelopor Tugas Akhir kami
9. Mas mbak dan teman-teman Asisten Laboratorium Instrumentasi dan Kontrol, Teknik Fisika ITS

10. Teman-teman Tim Pirolisis (Wanda, Vicky, dan Mada) yang telah melewati suka dan duka bersama selama 6 bulan ini. Kalian sangar rek!
11. Teman-teman pejuang Tugas Akhir Teknik Instrumentasi 2018. Semangat 118! Sukses setelah ini
12. Teman-teman F50, keluarga saya juga. Terimakasih
13. Madiun, kota yang beberapa kali saya kunjungin untuk sebuah perjuangan
14. Kampus SI, Lab A300, kelas A101, Lab A102, kantin, TU tempat-tempat saya untuk melakukan perjuangan ini. Kenangan yang tak terlupakan
15. Untuk teman-teman yang membantu dan tidak dapat disebutkan semuanya, terimakasih banyak sekali.
16. Untuk kamu, terimakasih pakai doa dalam hati.

Penulis menyadari bahwa banyak kekurangan dalam pembuatan laporan ini baik dari segi materi maupun penyajian. Untuk itu, penulis mengharapkan kritik dan saran yang bersifat membangun serta semoga laporan ini bermanfaat bagi penulis sendiri khususnya dan pembaca pada umumnya.

Surabaya, 31 Juli 2018

Penulis

DAFTAR ISI

BAB III. METODOLOGI

BAB IV. HASIL DAN PEMBAHASAN

BAB V. PENUTUP

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Limbah plastik menyebabkan permasalahan yang umumnya hampir terjadi di setiap kota besar di Indonesia. Tidak seperti sampah yang ada pada umumnya, limbah plastik merupakan jenis sampah yang sukar diuraikan oleh tanah. Volume sampah plastik dari tahun ke tahun semakin bertambah dan hal tersebut merupakan masalah yang harus ditanggulangi secepatnya. Indonesia berada di peringkat kedua dunia penghasil sampah plastik ke laut yang mencapai sebesar 187,2 juta ton setelah Cina yang mencapai 262,9 juta ton [1]

Plastik merupakan bahan anorganik buatan yang tersusun dari bahan-bahan kimia yang berbahaya bagi lingkungan. Limbah plastik sulit untuk diuraikan secara alami. Untuk menguraikan limbah plastik membutuhkan waktu 80 tahun agar terdegradasi secara sempurna. Penggunaan bahan plastik tidak baik bagi lingkungan apabila digunakan tanpa menggunakan batasan tertentu. Sedangkan dalam kehidupan sehari-hari, penggunaan bahan plastik dapat ditemukan hampir pada seluruh aktivitas hidup manusia. Terdapat beberapa cara penanggulangan limbah plastik selain mengubur ataupun membakarnya,antara lain meliputi mengurangi penggunaan kantong plastik dengan menggantinya dengan alat (kain) untuk membungkus barang atau dikenal dengan *furoshiki* ; pengolahan limbah plastik menggunakan metode fabrikasi; dan penggunaan plastik biodegradable yang lebih mudah terurai di alam [2]

Penggunaan metode penanggulangan limbah plastik yang telah dijelaskan diatas masih belum mampu mengurangi volume limbah plastik di Indonesia karena dinilai kurang efektif dan efisien. Oleh karena itu, alat produksibahan bakar minyak dari limbah plastik dengan menggunakan metode pemanasan pirolisis dan proses distilasi sehingga rantai karbon pada plastik dapat terurai[3]. Alat produksi bahan bakar minyak dari limbah plastik

yang sekarang ada masih menggunakan sistem manual dengan menggunakan pencacah plastik, ruang pemanas sebagai sebagai penghasil uap, memiliki 2 tahap *condenser*, dan 2 *storage*[4].

Beberapa tinjauan dan aspek yang dibutuhkan dimana variabel proses tersebut selain dikontrol, harus juga dimonitoring agar pengguna dapat menjaga sesuai dengan apa yang diinginkan dan mengantisipasi masalah yang akan muncul pada alat ini. Oleh karena itu, sistem monitoring dibutuhkan untuk alat ini dengan beberapa variabel proses yang dikontrol level, temperature, dan berat limbah dengan mengintegrasikan semua sensor dan akan menggunakan *interface display* dan HMI *(Human Machine Interface)* sebagai tampilan secara realtime dan dengan membuat data *openlog data longger* yang akan menghasilkan analisis data sebagai data jangka panjang dari proses alat ini. Maka diperlukan rancang bangun sistem monitoring variabel proses alat pengubah limbah plastik menjadi Bahan Bakar Minyak (BBM) menggunakan Atmega berbasis HMI (Human Machine Interface)

1.2 Rumusan Masalah

Adapun rumusan masalah yang dapat ditarik dari latar belakang di atas yaitu:

a. Bagaimana cara merancang sistem monitoring variabel proses alat produksi limbah plastik menjadi bahan bakar minyak (BBM)?
b. Bagaimana cara menguji sistem monitoring variabel proses dengan mengintegrasi dan menampilkannya pada interface yang diinginkan dengan berbasis HMI (Human Machine Interface)

1.3 Batasan Masalah

Untuk memfokuskan penyelesaian masalah pada penelitian tugas akhir ini maka batasan masalah yang diangkat adalah sebagai berikut :

a. Variabel yang diukur adalah temperature, level, dan beban
b. Menggunakan mikrokontroller ATMEGA 16A

c. Menggunakan modul openlog data logger shield sebagai data logger
d. LCD Character 4 x 20 sebagai display pada Control Panel
e. HMI (Human Machine Interface) menggunkan Visual Studio Basic
f. Komunikasi yang digunakan ialah komunikasi USART (TX RX)

1.4 Tujuan

Adapun tujuan yang dicapai dalam tugas akhir ini adalah merancang dan membangun sistem monitoring variabel proses pada alat produksi bahan bakar minyak dari limbah plastik dengan mikrokontroller Atmega berbasis HMI *(Human Machine Inteface)*

1.5 Manfaat

Manfaat dari Tugas Akhir ini adalah sebagai sistem monitoring variabel proses alat produksi bahan bakar minyak dari limbah plastik menggunakan *mikrokontroller* Atmega berbasis HMI (*Human Machine Interface)* yang mana diharapkan Tugas Akhir ini nantinya dapat dijadikan media pembelajaran dan pengetahuan bagi mahasiswa.

(Halaman sengaja dikosongkan)

BAB II
TINJAUAN PUSTAKA

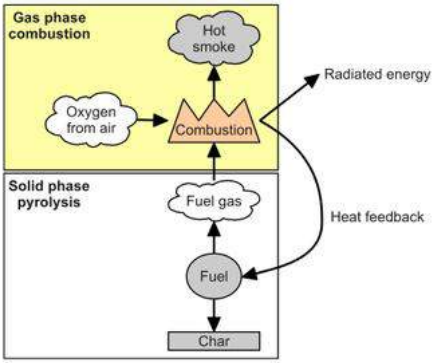
2.1 Pirolisis

Pirolisis adalah dekomposisi kimia bahan organik melalui proses pemanasan tanpa atau sedikit oksigen atau reagen lainnya, di mana material mentah akan mengalami pemecahan struktur kimia menjadi fase gas. Biasanya terdapat tiga produk dalam proses pirolisis yakni: gas, pyrolisis oil, dan arang, yang mana proporsinya tergantung dari metode pirolisis, karakteristik biomassa dan parameter reaksi. Masing masing produk pirolisis merupakan bahan bakar yang dapat di konversi menjadi listrik melalui berbagai cara yang berbeda. Proses pirolisis merupakan tahap awal dari rangkaian proses yang terjadi dalam proses gasifikasi dan melibatkan proses kimia dan fisik yang kompleks dimana suatu perubahan dalam kondisi operasi berpengaruh pada proses secara keseluruhan.

Pirolisis (juga disebut termalisis) dekomposisi termal (panas) dari bahan organik, seperti pada waktu batubara dipanaskan lebih dari 300 °C tanpa udara atmosfer. Pada reaksi kimia pirolisis biomasa, terdapat tiga faktor yang berpengaruh, yakni : 1) Bahan baku : komposisi kimia, kadar air. 2) Reaktor :vertical – shaft / batch reactor, rotating tubular / fluidized – bed reactor . 3) Kondisi operasi : suhu pirolisis, waktu pirolisis (waktu tinggal)

Produk utama dari proses pirolisis adalah arang, gas atau produk minyak yang dapat digunakan sebagai feedstocks petrokimia, dan bahan karbon untuk berbagai aplikasi. Minyak dapat dipergunakan sebagai bahan bakar untuk menghasilkn energi listrik melalui mesin pembakaran dalam atau internal combustioan engine seperti motor bensin maupun motor diesel. Char atau arang merupakan sisa pirolis yang dapat dipergunakan sebagai bahan bakar padat. Juga dapat dipergunakan sebagai bahan bakar pada proses pembakaran langsung melalui ataupun tanpa melului proses densifikasi. Sedangkan syngas dapat menghasilkan energi listrik melalui turbin gas.

Namun komposisi produk pirolisis dapat berbeda berdasarkan jenis limbah yang digunakan. Pirolisis dari limbah domestik (sampah kota) menghasilkan 35% produk arang dan kadar abu hingga 37%. Pirolisis dengan laju pemanasan yang lambat terhadap limbah ban akan menghasilkan arang hingga 50% dan kadar abu sekitar 10%



Gambar 2.1 Proses Pirolisis

2.2 Kondensasi

Kondensasi adalah proses yang mengubah zat gas dalam bentuk cair. Substansi yang mengalami kondensasi secara kimiawi sama dengan hanya keadaan fisik yang berubah. Proses yang terlibat dalam larutan sering disebut sebagai kebalikan dari penguapan dimana cairan beralih ke gas. Pada suhu dan tekanan tertentu, berbagai bahan kimia dapat berubah dari bentuk gas aslinya ke keadaan cair.

Penjelasan dasar dari proses kondensasi adalah melalui proses yang melibatkan hujan dan uap air. Sebelum hujan, uap air pada dasarnya naik ke udara untuk berkumpul dan menjadi awan. Uap air dalam bentuk gas dan pada tingkat tertentu terakumulasi di udara, air yang berupa gas ini akan berubah menjadi cairan dalam bentuk hujan. Air hujan dari awan maka akan jatuh kembali ke bumi dan menjadi bagian siklus yang sama yang menghasilkan kondensasi lagi. Proses di mana uap air berubah menjadi cair melibatkan temperatur yang lebih dingin. Dengan

penurunan suhu, uap air dalam bentuk gas akan dikompresi. Kompresi ini akan menyebabkan molekul untuk tinggal lebih dekat satu sama lain. Molekul yang berkumpul bersama-sama menjadi lebih kompak dan karena itu membentuk tetesan air yang sekarang dalam bentuk cair.

Proses kondensasi juga penting dalam industri yang menggunakan bahan kimia destilasi atau zat. Komponen kimia misalnya dapat dipisahkan atau diisolasi melalui proses distilasi yang menggunakan kondensasi sebagai salah satu cara utamanya. Percobaan kimia juga menggunakan kondensasi dalam memeriksa dan/atau mengisolasi komponen yang berbeda dari zat. Menjadi proses alami, kondensasi kadang-kadang dianggap sebagai proses yang tidak diinginkan karena efeknya pada properti. Dalam kasus bangunan dan lukisan misalnya, konversi udara lembab ke dalam cairan akan berarti kemungkinan kelembaban ekstra yang tidak diinginkan dan pembentukan noda.\

2.3 Plastik

Jenis plastik dapat dibedakan menjadi dua jenis, yaitu termoplastik dan termoset. Termoplastik merupakan jenis plastik yang menjadi lunak saat dipanaskan, dapat dicetak atau dibentuk dengan tekanan saat dalam keadaan plastik dan saat didinginkan bersifat memperkuat dan mempertahankan bentuk atau cetakannya. Beberapa termoplastik yang umum digunakan adalah *PolyEthylene Terephthalate (PET)*, *High Density PolyEthylene (HDPE)*, *PolyVinil Clorida (PVC)*, *Low Density PolyEthylene (LDPE)*, *PolyPropylene (PP)*, *PolyStyrene (PS)*, dan plastik lainnya.

PolyEthylene Terephthalate (PET) memiliki sifat umum tangguh dan jernih, kekuatan dan kekakuan yang baik, tahan kimia dan tahan panas, sifat penghalang yang baik untuk oksigen dan karbon dioksida. Digunakan dalam kemasan kemasan, minuman ringan dan botol air mineral, serat untuk pakaian, film, wadah makanan, transportasi, bangunan dan industri alat (seperti tahan api), dll.

High density polyethylene (HDPE) memiliki sifat umum kemampuan proses yang baik, keseimbangan kekuatan dan kekuatan benturan yang sangat baik, ketahanan kimia yang sangat baik, kristal, titik leleh (130-135^0 C), dan sifat penghalang uap air yang sangat baik. Digunakan untuk pembuatan produk blow molded (berbagai jenis kontainer, botol air), pipa, produk cetakan injeksi (tempat penyimpanan, tutup, ember, mug), film (tas pengangkut barang), dll.

Low Density PolyEthylene (LDPE) memiliki sifat umum kemampuan proses yang mudah, kepadatan rendah, sifat semi kristalin, kisaran leleh rendah, titik pelunakan rendah, ketahanan kimia yang baik, sifat dielektrik yang sangat baik, penghalang kelembaban rendah, abrasi yang buruk dan ketahanan peregangan. Digunakan untuk membuat tas pembawa, tas tugas berat, tas pembibitan, botol meremas kecil. Juga digunakan dalam kemasan susu, kawat dan isolasi kabel, dll.

PolyPropylene (PP) memiliki sifat umum kepadatan rendah, ketahanan kimia yang sangat baik, ketahanan terhadap tekanan lingkungan, titik lebur yang tinggi, kemampuan proses yang baik, sifat dielektrik, biaya rendah, ketahanan creep. Digunakan untuk membuat botol, wadah medis, pipa, lembaran, sedotan, film, perabotan, barang rumah, koper, mainan, pengering rambut, kipas angin, dll.

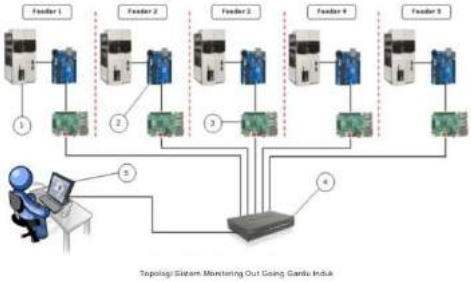
Ada banyak jenis plastik lain kecuali enam jenis ini, yang sering digunakan di sektor teknik. Contohnya meliputi polikarbonat (PC), nilon, dan butilena akrilonitril (ABS). Termoset adalah bahan yang sekali diset tidak dapat dilepas / dilunakkan dengan mengoleskan panas. Ini termasuk fenol, melamin dan urea formaldehida, poliester tak jenuh, epoksi dan poliuretan. Bahan ini tidak dapat didaur ulang.

Gambar 2.2 Jenis-jenis Plastik

2.4 Sistem Monitoring

Monitoring didefinisikan sebagai siklus kegiatan yang mencakup pengumpulan, peninjauan ulang, pelaporan, dan tindakan atas informasi suatu proses yang sedang diimplementasikan (Mercy, 2005). Umumnya, monitoring digunakan dalam checking antara kinerja dan target yang telah ditentukan. Monitoring ditinjau dari hubungan terhadap manajemen kinerja adalah proses terintegrasi untuk memastikan bahwa proses berjalan sesuai rencana (on the track). Monitoring dapat memberikan informasi keberlangsungan proses untuk menetapkan langkah menuju ke arah perbaikan yang berkesinambungan. Pada pelaksanaannya, monitoring dilakukan ketika suatu proses sedang berlangsung. Level kajian sistem monitoring mengacu pada kegiatan per kegiatan dalam suatu bagian (Wrihatnolo, 2008), misalnya kegiatan pemesanan barang pada supplier oleh bagian purchasing. Indikator yang menjadi acuan monitoring adalah output per proses / per kegiatan. Pada dasarnya, monitoring memiliki dua fungsi dasar yang berhubungan, yaitu compliance monitoring dan performance monitoring (Mercy, 2005). Compliance monitoring berfungsi untuk memastikan proses sesuai dengan harapan / rencana. Sedangkan, performance monitoring berfungsi untuk mengetahui perkembangan organisasi dalam pencapaian target yang diharapkan. Umumnya, output monitoring berupa progress report

proses. Output tersebut diukur secara deskriptif maupun non-deskriptif. Output monitoring bertujuan untuk mengetahui kesesuaian proses telah berjalan. Output monitoring berguna pada perbaikan mekanisme proses / kegiatan di mana monitoring dilakukan



Gambar 2.3 Sistem Monitoring pada suatu Industri

2.5 Komunikasi Data Serial

Komunikasi serial adalah komunikasi yang pengiriman datanya per-bit secara berurutan dan bergantian. Komunikasi ini mempunyai suatu kelebihan yaitu hanya membutuhkan satu jalur dan kabel yang sedikit dibandingkan dengan komunikasi paralel. Pada prinsipnya komunikasi serial merupakan komunikasi dimana pengiriman data dilakukan per bit sehingga lebih lambat dibandingkan komunikasi parallel, atau dengan kata lain komunikasi serial merupakan salah satu metode komunikasi data di mana hanya satu bit data yang dikirimkan melalui seuntai kabel pada suatu waktu tertentu. Pada dasarnya komunikasi serial adalah kasus khusus komunikasi paralel dengan nilai n = 1, atau dengan kata lain adalah suatu bentuk komunikasi paralel dengan jumlah kabel hanya satu dan hanya mengirimkan satu bit data secara simultan.Hal ini dapat disandingkan dengan komunikasi paralel yang sesungguhnya di mana n-bit data dikirimkan bersamaan, dengan nilai umumnya $8 \leq n \leq 128$. Komunikasi serial ada dua macam, asynchronous serial dan synchronous serial. Synchronous serial adalah komunikasi dimana hanya ada

satu pihak (pengirim atau penerima) yang menghasilkan clock dan mengirimkan clock tersebut bersama-sama dengan data. Contoh pengunaan synchronous serial terdapat pada transmisi data keyboard. Asynchronous serial adalah komunikasi dimana kedua pihak (pengirim dan penerima) masing-masing menghasilkan clock namun hanya data yang ditransmisikan, tanpa clock. Agar data yang dikirim sama dengan data yang diterima, maka kedua frekuensi clock harus sama dan harus terdapat sinkronisasi. Setelah adanya sinkronisasi, pengirim akan mengirimkan datanya sesuai dengan frekuensi clock pengirim dan penerima akan membaca data sesuai dengan frekuensi clock penerima. Contoh penggunaan asynchronous serial adalah pada Universal Asynchronous Receiver Transmitter (UART) yang digunakan pada serial port (COM) komputer.

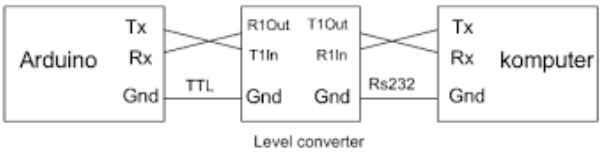
Antarmuka Kanal serial lebih kompleks/sulit dibandingkan dengan antarmuka melalui
kanal paralel, hal ini disebabkan karena:

1. Dari Segi perangkat keras: adanya proses konversi data pararel menjadi serial atau sebaliknya menggunakan piranti tambahan yang disebut UART (Universal Asynchronous Receiver/Transmitte) dan
2. Dari Segi perangkat lunak: lebih banyak register yang digunakan atau terlibat.

Namun di sisi lain antarmuka kanal serial menawarkan berapa kelebihan dibandingkan secara paralel, antara lain:

1. Kabel untuk komunikasi serial bisa lebih panjang dibandingkan dengan paralel; data-data dalam komunikasi serial dikirim-kan untuk logika ‘1’ sebagai tegangan -3 s/d -25 volt dan untuk logika ‘0’ sebagai tegangan +3 s/d +25 volt, dengan demikian tegangan dalam komunikasi serial memiliki ayunan tegangan maksimum 50 volt, sedangkan pada komunikasi paralel hanya 5 volt. Hal ini menyebabkan gangguan pada kabel-kabel panjang lebih mudah diatasi dibandingkan pada parallel.

2. Jumlah kabel serial lebih sedikit; Anda bisa menghubungkan dua perangkat komputer yang berjauhan dengan hanya 3 kabel untuk konfigurasi null modem, yaitu TXD (saluran kirim), RXD(saluran terima) dan Ground, bayangkan jika digunakan teknik paralel akan terdapat 20 – 25 kabel. Namun pada masing-masing komputer dengan komunikasi serial harus dibayar "biaya" antarmuka serial yang agak lebih mahal.
3. Banyaknya piranti saat ini (palmtop, organizer, hand-phone dan lainlain) menggunakan teknologi infra merah untuk komunikasi data, dalam hal ini pengiriman datanya dilakukan secara serial. IrDA-1 (spesifikasi infra merah pertama) mampu mengirimkan data dengan laju 115,2 kbps dan Konsep Komunikasi Serial 2 dibantu dengan piranti UART, hanya panjang pulsa berkurang menjadi 3/16 dari standar RS-232 untuk menghemat daya.
4. Untuk teknologi embedded system, banyak mikrokontroler yang dilengkapi dengan komunikasi serial (baik seri RISC maupun CISC) atau Serial Communication Interface (SCI); dengan adanya SCI yang terpadu pada 1C mikrokontroler akan mengurangi jumlah pin keluaran, sehingga hanya dibutuhkan 2 pin utama TxD dan RxD (di luar acuan ground).



Ilustrasi wiring komunikasi serial dg Level Converter TTL – Rs232

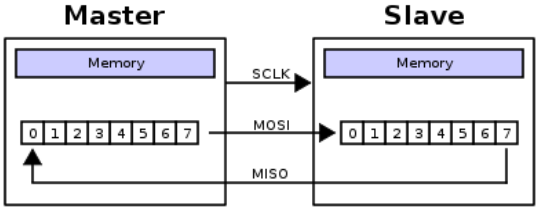
Gambar 2.4 Komunikasi Serial

2.5.1 SPI (Serial Peripheral Interface)

Serial Peripheral Interface (SPI) adalah salah satu protocol komunikasi serial shyncronous yang di-develop oleh Motorola.

Dalam koneksi SPI, device yang terhubung satu sama lain akan bersifat Full Duplex, yaitu ada device yang bertindak sebagai Master dan Slave. Master device adalah perangkat yang memulai sambungan dengan cara menginialisasi SPI address dari slave device. Lalu master dan slave dapat mengirim atau menerima data. Hal ini sudah disebutkan sebelumnya bahwa komunikasi full duplex yang artinya master dan slave dapat menerima ataupun mengirim data. Slave device dapat menerima atau mengirim data dalam waktu yang bersamaan, itulah yang disebut Full Duplex.

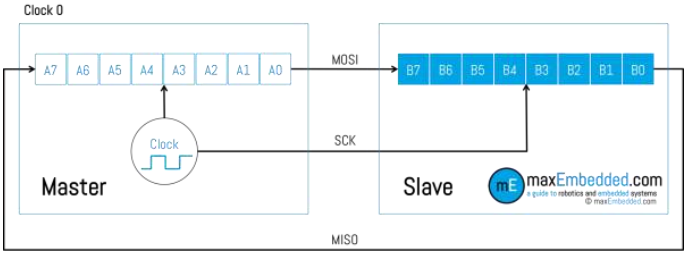
SPI beroperasi berdasarkan shift register baik master device maupun slave device, keduanya akan mempunyai 8 bit shift register. Namun tergantung dari berbagai macam arsitektur mikrokontroler, ada yang bisa memiliki 10 bit ataupun 12 bit shift register. Untuk memulai komunikasi, bus master melakukan konfigurasi clock, dengan catatan frekuensi atau kecepatan transfer data antara SPI master device dan slave device harus sama, biasanya bisa mencapai beberapa MHz. Master akan memilih perangkat slave dengan mengeluarkan logika 0, lalu master akan menunggu proses yang telah dijadwalkan di master itu sendiri seperti uratan intrupsi timer, konversi analog ke digital (ADC), dll. Lalu setelah periode itu selesai master akan mengeluarkan clock yang pertAnda akan dimulainya proses komunikasi Serial.



Gambar 2.5 Sistem SPI

Setiap satu clock SPI dilakukan, maka akan terjadi komunikasi full duplex antara master device dengan slave device. Master mengirimkan satu Bit pada line MISO, lalu slave akan membacanya. Setelah itu, pada line MISO slave device akan mengirimkan data kembali ke master device dan master akan

membacanya. Urutan atau sekuen ini akan bertahan seperti di atas meskipun kita tidak menggunakan komunikasi Full Duplex atau hanya menggunakan satu line komunikasi saja (seperti simplex). Transmisi data akan melibatkan dua shift register dari beberapa ukuran data yang diberikan seperti 8 bit, 10 bit ataupun 12 bit. Namun pada umunya digunakan 8 bit shift register. Keduanya akan terkoneksi dalam topologi ring secara virtual. Data yang dikirimkan biasanya akan bergeser satu per satu dari bit pertama hingga bit kedelapan. Setelah register bergeser keluar, berarti master dan slave sudah bertukar data. Lalu selanjutnya akan bergantian slave dan master. Jika data yang dikirim banyak, maka shift register akan diisi ulang dengan data yang baru. Lalu proses pengirimannya pun diulang. Proses pengiriman akan dihentikan jika master mengirim sinyal toggle untuk mengakhiri pemilihan slave.



Gambar 2.6 SPI Transfer Data

Master dan slave terhubung dalam 4 jalur. Setiap jalur ini mempunyai informasi dan membawa sinyal tertentu yang didefinisikan oleh protocol dari bus SPI. Keempatnya adalah:

1. MOSI (Master Output Slave Input), ini adalah sinyal output dari master device yang merupakan shift register dari master menuju input dari slave.
2. MISO (Master Input Slave Output), ini adalah input dari master device untuk menerima data shift register dari slave device menuju master.
3. SCK atau SCLK (Serial Clock), ini adalah clock yang dihasilkan master yang berguna menAndakan komuniaksi

SPI dan untuk melakukan shifting terhadap shift register dari kedua device.

4. SS' (Slave Select), ini adalah pin yang digunakan untuk memilih slave mana yang akan diajak berkomunikasi oleh master. (dengan asumsi lebih dari satu slave device)

Sinyal MOSI, SCK, dan SS berasal dari master untuk dikirim ke slave. Sedangkan MISO digunakan untuk menerima sinyal dari slave. Berikut ini adalah diagram interface antara master dan slave device.

2.5.2 USART

USART (Universal Synchronous and Asynchronous serial Receiver and Transmitter), merupakan salah satu mode komunikasi yang dimiliki oleh Mikrokontroler ATMega16. USART memiliki 2 pin (RxD dan TxD) untuk Asynchronous dan 3 bit TxD, RxD, xCK untuk Synchronous. Komunikasi serial data antara master dan slave pada SPI diatur melalui 4 buah pin yang terdiri dari SCLK, MOSI, MISO, dan SS sbb:

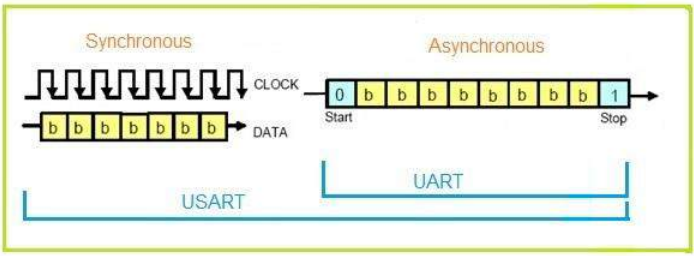
- SCLK dari master ke slave yang berfungsi sebagai clock
- MOSI jalur data dari master dan masuk ke dalam slave
- MISO jalur data keluar dari slave dan masuk ke dalam master
- SS (slave select) merupakan pin yang berfungsi untuk mengaktifkan slave

USART Pada transimi sinkron (USART) pengirim akan mengirimkan clock / timing signal sehingga device penerima tahu kapan membaca bit data berikutnya. Transimisi asinkron (UART) mengijinkan pengirim tidak memberikan clock sinyal pada penerima, sebagai gantinya untuk memulai transmisi pengirim mengirimkan start bit pada tiap byte data yang dikirimkan dan diakhiri dengan stop bit. Komunikasi dengan menggunakan USART dapat dilakukan dengan dua cara yaitu dengan mode sinkron dimana pengirim data mengeluarkan pulsa/clock untuk sinkronisasi data, dan yang kedua dengan mode asinkron, dimana pengirim data tidak mengeluarkan pulsa/clock, tetapi untuk

proses sinkronisasi memerlukan inisialisasi, agar data yang diterima sama dengan data yang dikirimkan. Pada proses inisialisasi ini setiap perangkat yang terhubung harus memiliki baud rate (laju data) yang sama. Pada mikrokontroler AVR untuk mengaktifkan dan mengeset komunikasi USART dilakukan dengan cara mengaktifkan register2 yang digunakan untuk komunikasi USART. Fungsi Penerima pada USART diaktifkan dengan menset 1 bit RXEN di register UCSRB . Ketika penerima diaktifkan, operasi normal pin i/o dirubah mejadi pin receive serial(Rx) USART . Baud rate, mode operasi dan format frame harus diatur sebelum ada penerimaan serial.
Data yang diterima serial ditampung di bufer penerima dan kita bisa mendapatkan data tsb dgn cara membaca register UDR. Berikut ini contoh pembacaan data serial USART dengan polling bit RXC . bit RxC otomatis akan bernilai 1 jika ada data di buffer penerima dan bernilai 0 jika tdk ada data di buffer penerima. bit RXC ada bit ke7 di register UCSRA.
Inisialisasi USART harus diinisialisasi sebelum komunikasi dilakukan.Proses inisialisasi biasanya terdiri dari pengaturan baud rate, pengaturan format frame dan mengaktifkan(enable) Transmitter atau Receiver/Penerima tergantung pada penggunaan. Untuk operasi USART dgn interupsi, Global Interrupt Flag harus diclearkan (dan interupsi dinonaktifkan secara global) ketika melakukan inisialisasi. Bit Flag TXC dapat digunakan untuk memeriksa bahwa Transmitter telah menyelesaikan semua transfer, dan bit flag RXC dapat digunakan untuk memeriksa bahwa tidak ada data yang belum dibaca dalam buffer penerima. Perhatikan bahwa Flag TXC harus diclearkan sebelum pengiriman (sebelum UDR ditulis) jika digunakan untuk pengiriman.

- Bit 7–RXC: USART Receive Complete
  RXC otomatis akan bernilai 1, jika ada data baru di bufer penerima. RXC otomatis akan bernilai 0, jika data sudah dibaca atau bufer penerima kosong.

- Bit 6–TXC: USART Transmit Complete
  TXC otomatis akan bernilai 1, jika data di buffer selesai dikirim.
- Bit 5–UDRE: USART Data Register Empty
  UDRE otomatis akan bernilai 1 , jika register UDR kosong transmiter siap mengirim data. UDRE=0, UDR berisi data yg belum selesai dikirim .
- Bit 4–FE: Frame Error
  FE otomatis akan bernilai 1, jika ada frame eror.
- Bit 3–DOR: Data OverRun
  DOR otomatis akan bernilai 1, jika data datang ketika bufer penuh(terjadi antrian).
- Bit 2–PE: Parity Error
  PE otomatis akan bernilai 1, jika terjadi parity eror.
  • Bit 1 – U2X: Double the USART Transmission Speed kita set U2X=0, kecepatan normal. U2X=1 kecepatan 2xbaudrate.
- Bit 0 – MPCM: Multi-processor Communication Mode kita set MCM=1 byte pertama yg diterima harus 9 bit , jika tdk data byte akan diabaikan.bit ini terjadi hanya untuk penerimaan saja pd komunikasi banyak microcontroller

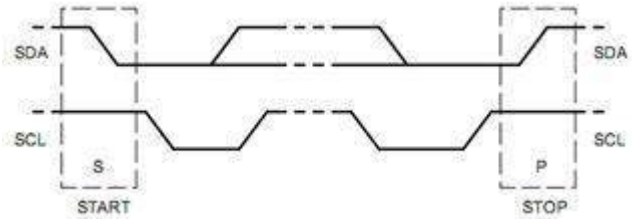


Gambar 2.7 USART dan UART

2.5.3 I2C (Inter Integrated Circuit)

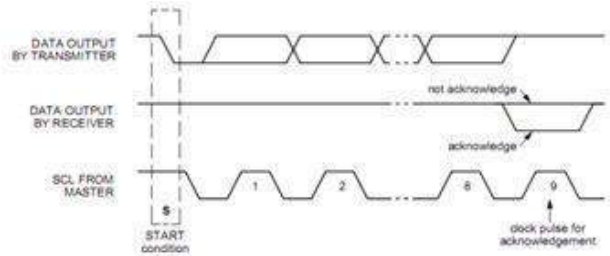
Inter Integrated Circuit atau sering disebut I2C adalah standar komunikasi serial dua arah menggunakan dua saluran

yang didisain khusus untuk mengirim maupun menerima data. Sistem I2C terdiri dari saluran SCL (Serial Clock) dan SDA (Serial Data) yang membawa informasi data antara I2C dengan pengontrolnya. Piranti yang dihubungkan dengan sistem I2C Bus dapat dioperasikan sebagai Master dan Slave. Master adalah piranti yang memulai transfer data pada I2C Bus dengan membentuk sinyal Start, mengakhiri transfer data dengan membentuk sinyal Stop, dan membangkitkan sinyal clock. Slave adalah piranti yang dialamati master. Sinyal Start merupakan sinyal untuk memulai semua perintah, didefinisikan sebagai perubahan tegangan SDA dari "1" menjadi "0" pada saat SCL "1". Sinyal Stop merupakan sinyal untuk mengakhiri semua perintah, didefinisikan sebagai perubahan tegangan SDA dari "0" menjadi "1" pada saat SCL "1". Kondisi sinyal Start dan sinyal Stop seperti tampak pada Gambar 2.8



Gambar 2.8 Kondisi sinyal start dan stop

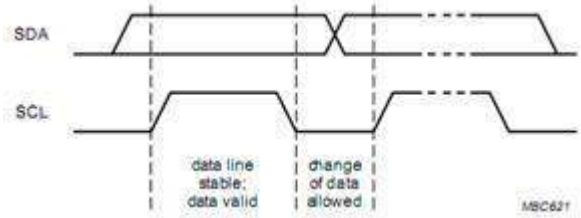
Sinyal dasar yang lain dalam I2C Bus adalah sinyal acknowledge yang disimbolkan dengan ACK Setelah transfer data oleh master berhasil diterima slave, slave akan menjawabnya dengan mengirim sinyal acknowledge, yaitu dengan membuat SDA menjadi "0" selama siklus clock ke 9. Ini menunjukkan bahwa Slave telah menerima 8 bit data dari Master. Kondisi sinyal acknowledge seperti tampak pada Gambar 2.9



Gambar 2.9 Sinyal ACK dan NACK

Dalam melakukan transfer data pada I2C Bus, kita harus mengikuti tata cara yang telah ditetapkan yaitu:

1. Transfer data hanya dapat dilakukan ketikan Bus tidak dalam keadaan sibuk.
2. Selama proses transfer data, keadaan data pada SDA harus stabil selama SCL dalam keadan tinggi. Keadaan perubahan "1" atau "0" pada SDA hanya dapat dilakukan selama SCL dalam keadaan rendah. Jika terjadi perubahan keadaan SDA pada saat SCL dalam keadaan tinggi, maka perubahan itu dianggap sebagai sinyal Start atau sinyal Stop.



Gambar 2.10 Trasfer Bit pada I2C bus

2.6 Modbus

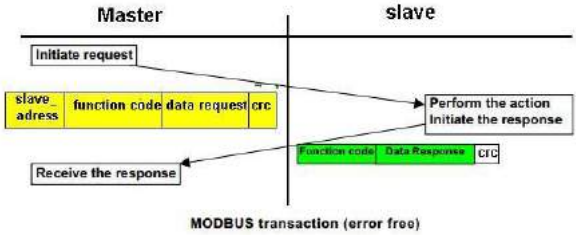
Protocol modbus dibuat oleh perusahaan PLC bernama Modicon tahun 1979 dan sampai sekarang menjadi salah satu prtotocol komunikasi standar yg dipakai dalam Automatisasi pengelolaan Gedung, Proses Industri dll.

Beberapa Jenis Type Modbus

- Modbus Serial (RTU & ASCII)
- Modbus TCP/IP
- Modbus +

Protokol komunikasi Modbus Serial mengatur cara-cara dan format komunikasi serial (rs232 atau rs485) antara master dengan Slave (master atau slave dpt berupa PLC ,microcontroller, smart device dll) .Jaringan Modbus terdiri dari Master dan beberapa Slave, Master yang berinisiatif memulai komunikasi antara lain menulis data,membaca data,dan mengetahui status SLave . Permintaan master disebut juga sebagai request atau query. Slave hanya bersifat pasif/menunggu atau dgn kata lain Slave hanya me respon jika ada permintaan/query dari Master. Jumlah Slave dalam protokol Modbus bisa sebanyak 247 slave. Slave dapat berupa PLC, peralatan elektronik, controller, sensor dll.

Penyimpanan data pada modbus



Gambar 2.11 Modbus Slave dan Master

Pada protokol modbus terdapat 4 buah jenis penyimpanan data dengan panjang masing-masing 16 bit.

1. Coil : Pada mulanya jenis data ini digunakan untuk mengaktifkan coil relay . nilai jenis data ini ON atau OFF . Coil mempunyai panjang 16 bit, sehingga untuk mengaktifkan/ON dgn cara memberi nilai FF00H dan 0000H untuk OFF. data FF00 dan 00 disimpan di register 00000 sampai 09999
2. Input Relay / input biner / input digital/input diskrit: kebalikan dengan coil, input relay digunakan untuk

mengetahui status relay apakah sedang ON atau OFF. Input relay bersifat read only bagi master dan hanya bisa dirubah oleh slave saja. Data tsb disimpan di register 10001 sampai 19999

3. Input Register : Input Regsiter digunakan untuk menyimpan data analog dgn range nilai 0 ~ 65535 . Input register bersifat read only bagi master. Data ini disimpan di register ber nomor 30001 sampai 39999
4. Holding Register : Holding register digunakan untuk menyimpan nilai dgn range 0~65535 .register ini mempunyai alamat register 40001 sampai 49999

Primary tables	Object type	Type of	Comments
Coils	Single bit	Read-Write	Master dan slave bisa merubah data coil
Discretes Input	Single bit	Read-Only	data hanya bisa di rubah oleh slave
Input Registers	16-bit word	Read-Only	data hanya bisa di rubah oleh slave
Holding Registers	16-bit word	Read-Write	Master dan slave bisa merubah data register

Gambar 2.12 Penyimpanan Data pada Modbus

2.7 Transmisi Data

Transmisi data dapat terjadi dalam dua model dasar, yaitu transmisi paralel atau transmisi serial. Data didalam sebuah sistem komputer ditransmisikan melalui model paralel yang disesuaikan dengan ukuran kata dalam sebuah sistem komputer. Data antara sebuah sistem komputer dengan sistem komputer lainnya biasanya ditransmisikan melalui model serial.

- **Transmisi Paralel**

Transmisi paralel, sejumlah bit dikirimkan per waktu. Masing-masing bit mempunyai jalurnya tersendiri. Dikarenakan oleh sifatnya yang demikian, maka data yang mengalir pada transmisi paralel jauh lebih cepat pada transmisi serial. Model transmisi paralel biasanya digunakan untuk melakukan komunikasi jarak pendek.Contohnya, transmisi ke printer atau untuk komunikasi data dua buah komputer. Pada transmisi paralel, beberapa bit (biasanya 8 bit atau satu byte / karakter) akan dikirim secara bersamaan pada saluran yang berbeda (kabel,

saluran frekuensi) dalam kabel yang sama, atau radio jalan, dan disinkronisasi untuk sebuah jam. Perangkat paralel memiliki bus data yang lebih luas daripada perangkat serial sehingga dapat mentransfer data dalam kata-kata dari satu atau lebih byte pada suatu waktu. Akibatnya, ada percepatan dalam transmisi paralel bit rate lebih dari laju bit transmisi serial. Namun, percepatan ini adalah biaya versus tradeoff sejak beberapa kabel biaya lebih dari satu kawat, dan sebagai kabel paralel mendapatkan lagi, sinkronisasi waktu antara beberapa saluran menjadi lebih sensitif terhadap jarak. Waktu untuk transmisi paralel disediakan oleh sinyal clocking konstan dikirim melalui kawat terpisah dalam kabel paralel; sehingga transmisi paralel dianggap sinkron. Suatu pengiriman data disebut paralel, jika sekelompok bit data ditransmisikan secara bersama-sama dan melewati beberapa jalur transmisi yang terpisah. Proses pengiriman data lebih cepat
Sistem ini akan lebih efektif untuk transmisi data yang memiliki jarak tidak terlalu jauh agar data yang diterima itu benar maka selang waktu yang digunakan oleh pengirim dan penerima harus sama. Untuk keperluan tersebut mka pengirim dan penerima harus menambahkan "detak" (Time Pulse).

- Data dikirimkan sekaligus, misal 8 bit bersamaan
- Kecepatan tinggi
- Karakteristik Media harus baik
- Masalah "SKEW Efek" yang terjadi pada sejumlah pengiriman bit secara serempak dan tiba pada tempat yang dituju dalam waktu yang tidak bersamaan

- **Transmisi Serial**

Pada transmisi serial, pada setiap waktu hanya 1 bit data yang dikirimkan. Dengan kata lain, bit-bit data tersebut dikirimkan secara satu per satu. Model transmisi seperti ini dijumpai pada contoh seperti seorang pengguna menghubungkan terminal ke host komputer yang berada pada bangunan yang lain. Berikut merupakan gambar pengiriman transmisi serial dari

pengirim ke penerima. Mode serial membutuhkan sinkronisasi/penyesuaian yang berfungsi untuk :

- Mengetahui bilamana sinyal yang diterimanya merupakan bit data (sinkronisasi bit)
- Mengetahui bilamana sinyal yang diterimanya membentuk sebuah karakter (sinkronisasi karakter)
- Mengetahui bilamana sinyal yang diterimanya membentuk sebuah blok data (sinkronisasi blok)
-

Selanjutnya, pada transmisi serial dapat berbentuk dua jenis, yaitu transmisi serial sinkron (synchronous) dan transmisi serial asinkron (asynchronous). Berikut ini merupakan penjelasan dari masing-masing jenis transmisi serial tersebut. Transmisi Serial Sinkron (Synchronous).

Transmisi Serial Sinkron (Synchronous)

Pada transmisi sinkron, sebelum terjadi komunikasi, diadakan sinkronisasi clock antara pengirim dan penerima.
Data dikirim dalam satu blok data (disebut Frame) yang berisi bit2 Pembuka (preamble bit), bit data itu sendiri dan bit2 penutup postamble bit. Ditambahlan juga bit2 kontrol pada blok tersebut.
Variasi ukuran frame mulai 1500 byte sampai 4096 byte
Dalam komunikasi sinkron, sbh line 56 kbps mampu membawa data sampai 7000 byte per detik

Transmisi Serial Asinkron (Asynchronous)

Pada transmisi Asinkron, sebelum terjadi komunikasi, tdk diadakan sinkronisasi clock antara pengirim dan penerima Data dikirim per karakter dan masing2 karakter memiliki bit start (biasanya 0) dan bit stop (biasanya 1) Start bit berfungsi utk menandakan adanya rangkaian bit karakter yang siap dicuplik.
Stop bit berfungsi utk melakukan proses menunggu karakter berikutnya
Setiap karakter terdiri dari 10 bit dengan rincian

- 1 bit start bit

- 1 bit stop bit
- 7 bit data

Contoh perangkat berbasis transmisi asinkron : RS-232, com #, USB, dll

Perbedaan Transmisi Paralel dan Serial

Perbedaan antara transmisi serial dengan parallel adalah transmisi serial mentransmisikan 1 bit dalam 1 waktu sedangkan transmisi parallel mentransmisikan beberapa bit dalam 1 transmisi. Hal ini menyebabkan transmisi parallel lebih cepat dibanding transmisi serial. Komunikasi serial dapat lebih cepat dibanding komunikasi parallel, yang dibutuhkan hanyalah frekuensi pengiriman data yang lebih tinggi.

Dalam komunikasi parallel, karena transmisi dilakukan pada waktu yang sama, maka dibutuhkan kabel lebih banyak. Sementara pada transmisi serial, kabel yang digunakan tetap dua. Hal ini menyebabkan kabel untuk transmisi serial lebih kompak dibanding kabel untuk transmisi parallel. Dengan semakin tingginya frekuensi, semakin tinggi juga gangguan elektromagnetik. Setiap kabel dapat diperlakukan sebagai antenna, menangkap noise yang ada di sekitarnya, dan mengganggu data yang sedang ditransmisikan. Dalam komunikasi parallel, karena banyaknya kabel yang digunakan, masalah gangguan elektromagnetik menjadi lebih serius. Di lain pihak, komunikasi serial yang hanya menggunakan dua kabel lebih mudah mengatasi masalah ini dengan melindungi kedua kabel yang digunakan. Perbedaan lain, yang juga menguntungkan komunikasi serial adalah walaupun secara teoritis komunikasi parallel mengirimkan data pada saat yg bersamaan, data tersebut tidak diterima pada saat yang bersamaan. Kelemahan komunikasi parallel adalah masalah half-duplex. Kabel yang digunakan untuk mengirim dan menerima data adalah kabel yang sama. Bandingkan dengan serial yang full-duplex, dimana masing

masing pengiriman dan penerimaan data menggunakan 2 kabel berbeda

2.8 HMI *(Human Machine Interface)*

HMI (Human Machine Interface) adalah membuat fungsi dari teknologi nyata.Dengan membuat desain HMI yang sesuai, akan membuat pekerjaan fisik lebih mudah padahampir semua solusi teknis, efektifitas dari HMI adalah dapat memprediksi penerimaan userterhadap seluruh solusi yang ada konsep HMI yang Moderen pada industri adalah sebagai media komunikasi antara operator dengan perancangan yang secara ideal mampumemberikan informasi yang diperlukan, agar perencanaan yang dilakukan dengan tingkatefisiensi maksimum. HMI merupakan sarana bagi operator untuk mengakses sistem otomasidi lapangan yang mencangkup operasional , pengembangan, perawatan troubeleshooting.

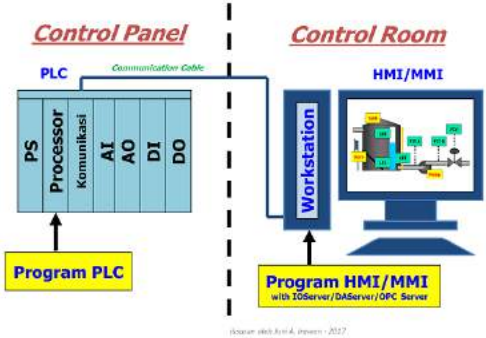
HMI biasa digunakan dalam dunia industri disebut juga sebuah tempat dimana interaksi antara manusia dan mesin terjadi. Tujuan dari interaksi antara manusia danmesin pada antarmuka pengguna adalah pengoperasian dan kontrol mesin yang efektif, danumpan balik dari mesin yang membantu operator dalam membuat keputusan operasional.Contoh-contoh dari konsep luas antarmuka pengguna ini termasuk aspek-aspek interaktifdari sistem operasi komputer, alat-alat, kontrol operator mesin berat, dan kontrol proses.Pertimbangan desain berlaku ketika membuat antarmuka pengguna yang berkaitan ataumelibatkan disiplin-disiplin ilmu seperti ergonomi dan psikologi.

Pengguna mencakup perangkat keras dan perangkat lunak. Pengguna hadir untuk berbagai sistem, dan menyediakan cara untuk:

- Input, memungkinkan pengguna untuk memanipulasi sebuah sistem
- Output, memungkinkan sistem untuk menunjukan efek dari manipulasi pengguna.

Secara umum, tujuan dari teknik interaksi manusia-mesin adalah untuk menghasilkan sebuahantarmuka pengguna yang membuatnya mudah, efisien, dan menyenangkan untukmengoperasikan sebuah mesin dengan cara yang menghasilkan hasil yang diinginkan. Ini biasanya berarti bahwa operator harus menyediakan input minimal untuk mencapai outputyang diharapkan, dan juga bahwa mesin harus meminimalkan output yang tidak diinginkan. Fungsi dari HMI yaitu:

1. Memberikan informasi plant yang up-to-date kepada operator melalui graphical userinterface.
2. Menerjemahkan instruksi operator ke mesin
3. Engineering Development Station



Gambar 2.13 HMI (*Human Machine Interface*)

2.9 Data Logger

Data Logger adalah suatu perangkat khusus yang mampu menyimpan data dalamjangka waktu tertentu. Data yang disimpan memiliki jumlah karakter tertentu untukdisimpan dalam media penyimpanan seperti pada kartu memori. Proses penyimpanan dataini biasa disebut *data logging*. Data yang disimpan dapat dari berbagai masukan, yangkemudian data masukan tersebut diperlukan dalam sebuah penelitian. Dalam merekam dataini, *data logger* memerlukan waktu yang akurat, maka dari itu

diperlukan suatu *Real TimeClock* (RTC), dan format data yang akan disimpan dalam memori, diperlukan juga sebuahmemori untuk menyimpan data.

2.10 ADC (Analog Digital Converter)

ADC (Analog To Digital Converter) adalah perangkat elektronika yang berfungsi untuk mengubah sinyal analog (sinyal kontinyu) menjadi sinyal digital. Perangkat ADC (Analog To Digital Convertion) dapat berbentuk suatu modul atau rangkaian elektronika maupun suatu chip IC. ADC (Analog To Digital Converter) berfungsi untuk menjembatani pemrosesan sinyal analog oleh sistem digital.

- Converter

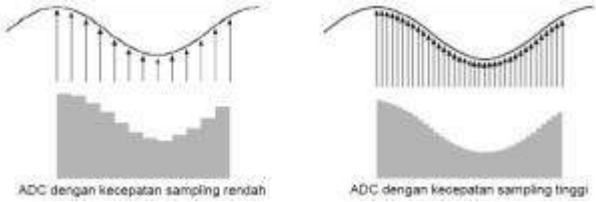
Alat bantu digital yang paling penting untuk teknologi kontrol proses adalah yang menerjemahkan informasi digital ke bentuk analog dan juga sebaliknya. Sebagian besar pengukuran variabel-variabel dinamik dilakukan oleh piranti ini yang menerjemahkan informasi mengenai vaiabel ke bentuk sinyal listrik analog. Untuk menghubungkan sinyal ini dengan sebuah komputer atau rangkaian logika digital, sangat perlu untuk terlebih dahulu melakukan konversi analog ke digital (A/D). Hal-hal mengenai konversi ini harus diketahui sehingga ada keunikan, hubungan khusus antara sinyal analog dan digital.

- ADC (Analog To Digital Convertion)

Analog To Digital Converter (ADC) adalah pengubah input analog menjadi kode – kode digital. ADC banyak digunakan sebagai Pengatur proses industri, komunikasi digital dan rangkaian pengukuran/ pengujian. Umumnya ADC digunakan sebagai perantara antara sensor yang kebanyakan analog dengan sistim komputer seperti sensor suhu, cahaya, tekanan/ berat, aliran dan sebagainya kemudian diukur dengan menggunakan sistim digital (komputer). ADC (Analog to Digital Converter) memiliki 2 karakter prinsip, yaitu kecepatan sampling dan resolusi.

- Kecepatan Sampling ADC

Kecepatan sampling suatu ADC menyatakan "seberapa sering sinyal analog dikonversikan ke bentuk sinyal digital pada selang waktu tertentu". Kecepatan sampling biasanya dinyatakan dalam sample per second (SPS).



Gambar 2.14 Ilustrasi Kecepatan Sampling

- Resolusi ADC

Resolusi ADC menentukan "ketelitian nilai hasil konversi ADC". Sebagai contoh: ADC 8 bit akan memiliki output 8 bit data digital, ini berarti sinyal input dapat dinyatakan dalam 255 ($2^n - 1$) nilai diskrit. ADC 12 bit memiliki 12 bit output data digital, ini berarti sinyal input dapat dinyatakan dalam 4096 nilai diskrit. Dari contoh diatas ADC 12 bit akan memberikan ketelitian nilai hasil konversi yang jauh lebih baik daripada ADC 8 bit.

- Prinsip Kerja ADC

Prinsip kerja ADC adalah mengkonversi sinyal analog ke dalam bentuk besaran yang merupakan rasio perbandingan sinyal input dan tegangan referensi. Sebagai contoh, bila tegangan referensi 5 volt, tegangan input 3 volt, rasio input terhadap referensi adalah 60%. Jadi, jika menggunakan ADC 8 bit dengan skala maksimum 255, akan didapatkan sinyal digital sebesar 60% x 255 = 153 (bentuk decimal) atau 10011001 (bentuk biner).

signal = (sample/max_value) * reference_voltage
= (153/255) * 5
= 3 Volts

BAB III
METODOLOGI

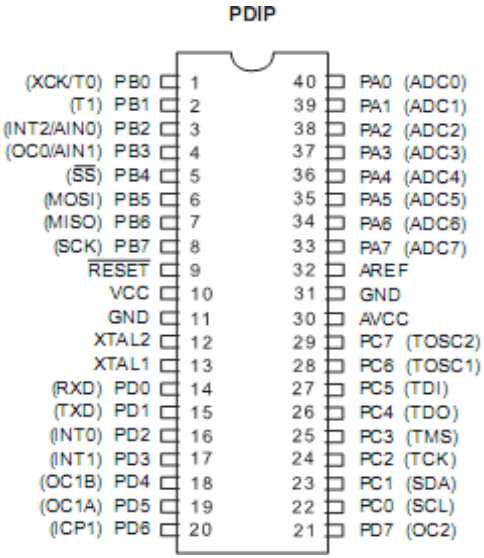
3.1 Bahan dan Peralatan yang Digunakan

Adapun komponen yang dibutuhkan pada Tugas Akhir ini, adalah sebagai berikut :

3.1.1 Mikrokontroler Atmega

Mikrokontroler ini menggunakan arsitektur harvard yang memisahkan memori program dari memori data, baik bus alamat maupun bus data, sehingga pengaksesan program dan data dapat dilakukan secara bersamaan (concurrent). secara garis besar mikrokontroler atmega16 terdiri dari :

- Arsitektur risc dengan throughput mencapai 16 mips pada frekuensi 16mhz.
- Memiliki kapasitas flash memori 16kbyte, eeprom 512 byte, dan sram 1kbyte
- Saluran i/o 32 buah, yaitu bandar a, bandar b, bandar c, dan bandar d.
- Cpu yang terdiri dari 32 buah register.
- User interupsi internal dan eksternal
- Port antarmuka spi dan bandar usart sebagai komunikasi serialfitur peripheral dua buah 8-bit timer/counter dengan prescaler terpisah dan mode compare ,satu buah 16-bit timer/counter dengan prescaler terpisah, mode compare, dan mode capture ,real time counter dengan osilator tersendiri : empat kanal pwm dan antarmuka komparator analog,8 kanal, 10 bit adckonfigurasi pena (pin) atmega16konfigurasi pena (pin) mikrokontroler atmega16 dengan kemasan 40- pena dapat dilihat pada gambar 2.2. dari gambar tersebut dapat terlihat atmega16 memiliki 8 pena untuk masing-masing bandar a (port a), bandar b (port b), bandar c (port c), dan bandar d (port d).



Gambar 3.1 Pin Atmega16

Deskripsi pin mikrokontroler AVR ATMega16, antara lain:

a. VCC (Power Supply) dan GND (Ground).
b. Port A (PA7-PA0)

Port A berfungsi sebagai input analog pada konverter A/D. Port A juga sebagai suatu port I/O 8-bit dua arah, jika A/D konverter tidak digunakan. Pin-pin Port dapat menyediakan resistor internal *pull-up* (yang dipilih untuk masing-masing bit). Ketika pin PA0 sampai PA7 digunakan sebagai input dan secara eksternal diset rendah ketika arus sumber resistor pull-up diaktifkan. Pin Port A dapat dalam keadaan *tri-stated*, yaitu suatu suatu kondisi reset menjadi aktif sekalipun waktu sudah habis. Dalam Port A ini juga dapat digunakan sebagai ADC 8 *channel* berukuran 10 bit.

c. Port B (PB7-PB0)

Port B adalah suatu port I/O 8-bit dua arah dengan resistor internal *pull-up*. Sebagai input, pin-pin Port B secara eksternal dapat diset rendah ketika arus sumber resistor *pull-up* diaktifkan. Pin Port B dapat dalam keadaan *tri-stated*, yaitu suatu suatu kondisi reset menjadi aktif sekalipun waktu sudah habis.

d. Port C (PC7-PC0)

Port C adalah suatu port I/O 8-bit dua arah dengan resistor internal *pull-up*. Sebagai input, pin-pin Port C secara eksternal dapat diset rendah ketika arus sumber resistor *pull-up* diaktifkan. Pin Port C dapat dalam keadaan *tri-stated*, yaitu suatu suatu kondisi reset menjadi aktif sekalipun waktu sudah habis.

e. Port D (PD7-PD0)

Port D adalah suatu port I/O 8-bit dua arah dengan resistor internal pull-up. Sebagai input, pin-pin Port D secara eksternal dapat diset rendah ketika arus sumber resistor *pull-up* diaktifkan. Pin Port D dapat dalam keadaan *tri-stated*, yaitu suatu suatu kondisi reset menjadi aktif sekalipun waktu sudah habis. Port D ini juga bisa digunakan untuk jalur komunikasi serial dengan perangkat luar.

f. RESET (Reset input).

Memori Program, Arsitektur ATMega16 mempunyai dua memori utama, yaitu memori data dan memori program. Selain itu, ATMega16 memiliki memori EEPROM untuk menyimpan data. ATMega16 memiliki 16K byte *On-chip In-System Reprogrammable Flash Memory* untuk menyimpan program. Instruksi ATMega16 semuanya memiliki format 16 atau 32 bit, maka memori *flash* diatur dalam 8K x 16 bit. Memori *flash* dibagi kedalam dua bagian, yaitu bagian program *boot flash section dan aplication flash section.*

3.1.2 LCD Character

Liquid Crystal Display (LCD) adalah sebuahperalatan elektronik yang berfungsi untuk menampilkanoutput sebuah

sistem dengan cara membentuk suatu citra atau gambaran pada sebuah layar. Secara garis besar komponen penyusun LCD terdiri dari kristal cair (liquid crystal) yang diapit oleh 2 buah elektroda transparan dan 2 buah filter polarisasi (polarizing filter). LCD yang ada dipasaran dikategorikan menurut jumlah baris yang dapat digunakan pada LCD yaitu 1 baris , 2 baris , dan 4 baris yang dapat digunakan hingga 80 karakter. Umumnya LCD yang digunakan adalah LCD dengan 1 controller yang memiliki 14 pin.

Gambar 3.2 LCD

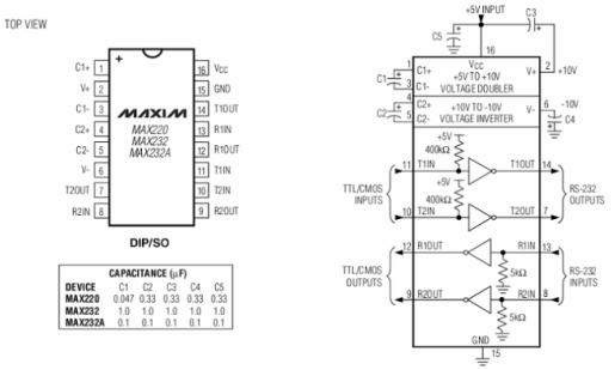
3.1.3 FT232RL USB

Komunikasi serial dalam hal ini mengirimkan data secara bergantian sesuai dengan urutan menggunakan sebuah kabel yaitu USB FTDI. Pada komunikasi serial data dikirim dari satu titik menuju satu titik yang lain tiap bit dalam satu waktu.USB FTDI Memiliki 2 port USB yang dapat dihubungkan dengan 1 buah USB flash disk dan 1 peralatan USB FTDI lainnya. USB FTDI tersedia antarmuka UART, paralel FIFO, dan SPI. Level tegangan antarmuka adalah 3,3 volt dan kompatibel dengan level tegangan TTL (5 volt tolerant), dimana membutuhkan sumber tegangan 5Volt DC

Gambar 3.3 USB FTDI

3.1.4 IC MAX 232

MAX232 merupakan salah satu jenis IC rangkaian antar muka dual RS-232 transmitter / receiver yang memenuhi semua spesifikasi standar EIA-232-E. IC MAX232 hanya membutuhkan power supply 5V (single power supply) sebagai catu. IC MAX232 di sini berfungsi untuk merubah level tegangan pada COM1 menjadi level tegangan TTL / CMOS. IC MAX232 terdiri atas tiga bagian yaitu dual charge-pump voltage converter, driver RS232, dan receiver RS232. Pada RS232, 1s (high) direpresentasikan dengan tegangan -3 s/d -25V, dan 0s (low) direpresentasikan sebagai +3 s/d +25V. Sedang diantara -3 dan +3V dianggap sebagai status mengambang dan tidak dianggap. Atas alasan ini, untuk menghubungkan 8051 yang ber-standar TTL dengan komputer (atau alat lain) yang menggunakan RS232, kita harus menggunakan peralatan tambahan misalnya dengan chip MAX232 untuk mengkonversi level TTL ke RS232 dan level RS232 ke level TTL. IC MAX232 adalah komponen untuk mengubah sinyal dari RS232 ke sinyal TTL yang bisa diolah oleh mikrontroler. IC ini berguna kalau anda mau membuat komunikasi data antara komputer (atau alat lain yang menggunakan RS232) dengan mikrokontroler.



Gambar 3.4 Arsitektur IC MAX 232

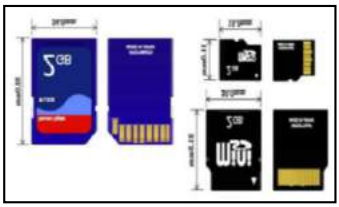
3.1.5 SD Card Modul

Modul SD Card adalah sebuah modul yang berfungsi untuk membaca dan menulis data ke/ dari SD Card. Modul ini memiliki interfacing menggunakan komunikasi SPI. Tegangan kerja dari modul ini dapat menggunakan level tegangan 3.3 V DC atau 5V DC, yang dapat digunakan salah satunya.Modul ini ccocok digunakan untuk membuat piranti-piranti yang membutuhkan suatu penyimpanan bersifat non-volatile (data akan tetap tersimpan walaupun tidak mendapatkan supply tegangan) dengan kapasitas besar, hingga mencapai Gigabyte. Modul ini banyak digunakan untuk pembuatan perekaman medis, perekam dan playback musik, data logger dan juga untuk pembuatan basis data.Modul ini memiliki 8 buah pin, diantaranya :GND, VCC 3.3V, VCC 5V, CS, MOSI, SCKMISO, GND

Gambar 3.5 SD CARD module

3.1.6 Media Penyimpanan

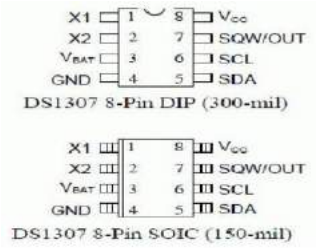
Micro SD seringkali digunakan sebagai sarana penyimpan data pada *Personal Digital Assistant* (PDA), kamera digital, dan telepon seluler (ponsel). SD card memiliki dimensi 32 mm x 24 mm x 2,1 mm (panjang x lebar x tebal). Pengembangan lebih lanjut dari media penyimpanan ini menghasilkan dimensiyang lebih kecil dan kompak seiring dengan perkembangan zaman yang berupa Mini SD dan Micro SD seperti yang ditunjukan Gambar 3.6



Gambar 3.6 Bentuk fisik dan dimensi SD Card, Mini SD, dan MicroSD

3.1.7Real Time Clock (RTC)

Real Time Clock (RTC) merupakan IC yang dibuat oleh perusahaan *Dallas* Semikonduktor. *Real Time Clock* (RTC) merupakan suatu *chip* (IC) yang memiliki fungsi sebagai penyimpan waktu dan tanggal.DS1307 merupakan *Real Time Clock* (RTC) yang menggunakan jalur data parallel yang dapat menyimpan data-data detik, menit, jam, tanggal, bulan, hari dalam seminggu, da tahun valid hingga 2100. 56 *byte,battery-backed*, RAM *nonvolatile* (NV) RAM untuk penyimpanan.DS1307 merupakan *Real Time Clock* (RTC) dengan jalur data parallel yang memiliki *interface* serial *Two-wire* (12C), sinyal luaran gelombang-kotak terprogram *(Programmable Squarewave)*, deteksi otomatis kegagalan-daya *(power-fail)* dan rangkaian *switch*, konsumsi daya kurang dari 500nA menggunakan modebaterai cadangan dengan operasional osilator. Tersedia fitur industri dengan ketahanan suhu : -40˚C hingga +85 ˚C. Tersedia dalam kemasan 8-pin DIP atau SOIC.



Gambar 3.7 Diagram Pin

Gambar 3.12 merupakan gambar pin pada modul RTC, Berikut ini merupakan daftar pin untuk RTC Parallel DS1307 :

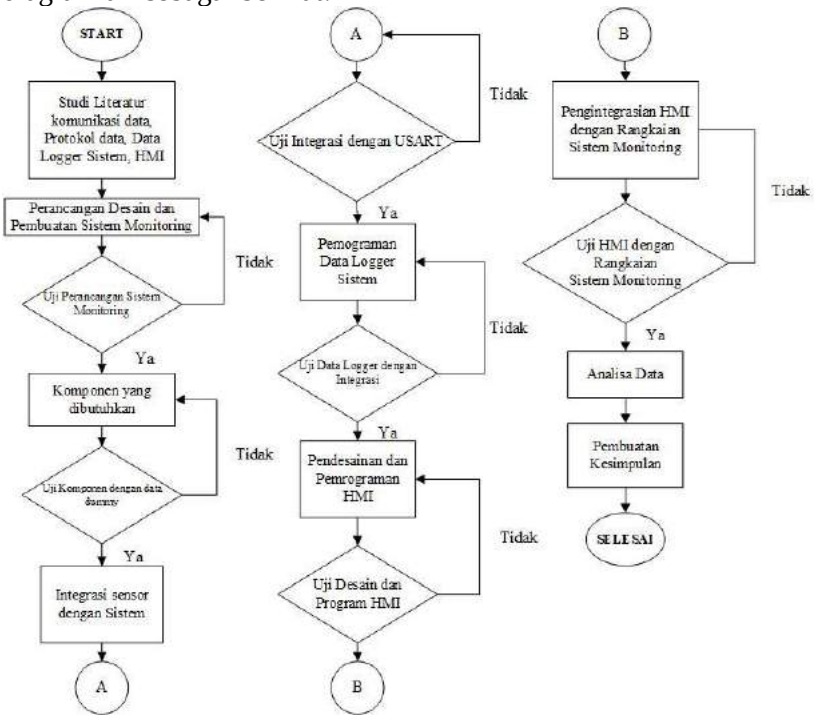
1. X1, merupakan pin yang digunakan untuk dihubungkan dengan X2
2. X2, berfungsi sebagai keluaran / output dari *crystal* yang digunakan. Terhubung juga dengan X1
3. V_{BAT} , merupakan *backup supply* untuk serial RTC dalam menjalankan fungsi waktu dan tanggal. Besarnya adalah 3 V dengan menggunakan jenis *Lithium Cell* atau sumber energi lain. Jika pin ini tidak digunakan maka harus terhubung dengan *Ground*. Sumber tegangan dengan 48mAH atau lebih besar dapat digunakan sebagaicadangan energi sampai lebih besar dari 10 tahun, namun dengan persyaratan untuk pengoperasian dalam suhu 25 ˚C.
4. GND, berfungsi sebagai Ground.
5. SDA – Serial Data, berfungsi sebagai masukan/ keluaran (I/O) untuk I2C serial *interface*. Pin ini bersifat open drain, oleh sebab itu membutuhkan eksternal *pull up resistor*.
6. SCL – Serial Data, berfungsi sebagai *clock* untuk input ke I2C dan digunakan untuk mensinkronisasi pergerakan data dalam serial *interface*. Bersifat *open drain*, oleh sebab itu membutuhkan eksternal *pull up resistor*.
7. SWQ/OUT, sebagai *square wafe/ Output Driver*. Jika diaktifkan, maka akan menjadi 4 frekuensi gelombang kotak yaitu 1 Hz, 4 kHz, 8 kHz, 32 kHz sifat dari pin ini sama dengan sifatpin SDA dan SCL sehingga membutuhkan eksternal *pull up resistor*. Dapat dioperasikan dengan VCC maupun dengan V_{BAT} .
8. VCC, merupakan sumber tegangan utama. Jika sumber tegangan terhubung dengan baik, maka peengaksesan data dan pembacaan data dapat dilakukan dengan baik.

Namun jika *backup supply* terhubung juga dengan VCC, namun besar VCC di bawah V_{TP} , maka pengaksesan data tidak dapat dilakukan

3.2 Proedur Pelaksanaan

3.2.1 *Flowchart*

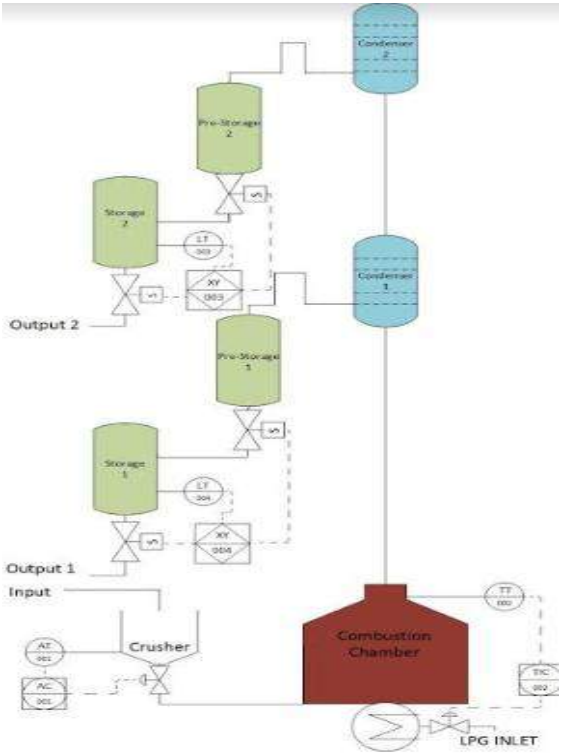
Langkah-langkahdalamtugasakhirinidigambarkandalam diagram alirsebagai berikut:



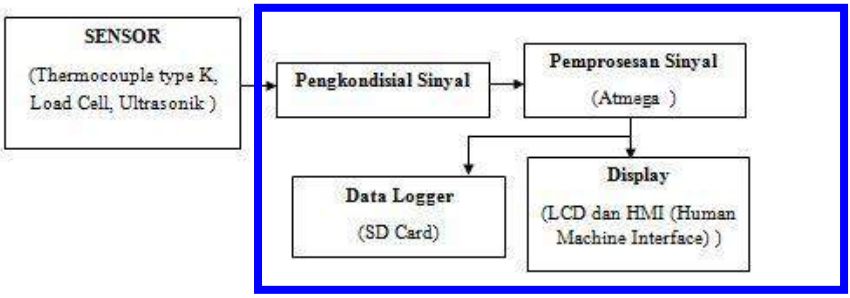
Gambar 3.8 Diagram Alir Sistem Monitoring Variabel Proses

3.2.2 Perancangan dan Pembuatan Sistem Monironing

Perancangan dan pembuatan*Piping and Instrumentation Diagram* digunakan sebagai alat bantu berupa skema untuk menerangkan konsep desain dari sistem monitoring, meliputi: jalur perpipaan, peralatan yang diperlukan, serta sistem kontrol dari proses yang berjalan.Pada diagram ini semua peralatan proses dan sistem instrumentasi digambarkan dalam bentuk simbol-simbol standar "Instrument Society of America" yang biasa disebut ISA *Standart.*

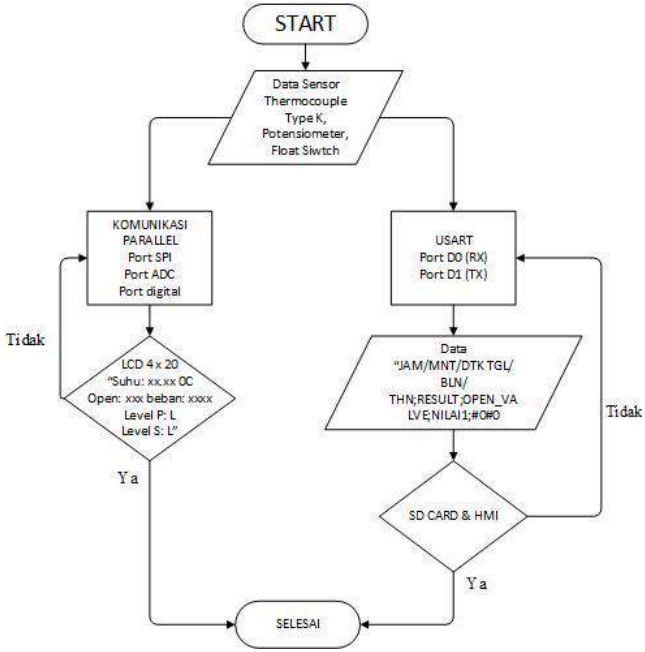


Gambar 3.9 P&ID Sistem



Gambar 3.10 Blok Diagram Sistem Monitoring Variabel Proses

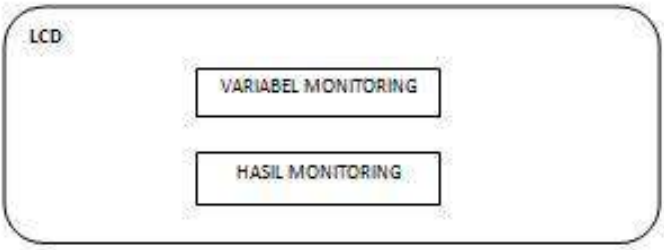
Pada gambar diatas dapat dilihat saat sensor mendeteksi variabel yang dimonitoring menghasilkan output berupa tegangan listrik yang kemudian masuk ke pengkondisian sinyal dan diubah ke data digital agar dapat dikirimkan ke mikrokontroler. Mikrokontroler berfungsi memproses data untuk ditampilkan pada layar LCD, selain itu mikrokontoler juga berfungsi mengirimkan data untuk interface berbasis object yang akan ditampilkan pada HMI (Human Machine Interface), serta mikrokontroler Atmega juga mengirimkan data ke data logger yang sudah terpasang SD card sebagai media penyimpanan. Sensor yang digunakan adalah thermocouple type k, potensiometer serta float switch. Dimana thermocouple type k menghasilkan tegangan dengan menggunakan mode SPI yang akan langsung di*convert* sesuai kebutuhkan dari Atmega16. Sensor potensiometer akan menghasilkan data analog terlebih dahulu dan akan di*convert* digital menggunakan rumus yang ada dan langsung akan diterima oleh Atmega16 hasil yang akan didapat berupa angka bulat. Sensor float switch merupakan sensor dengan menghasilkan hasil digital "0" atau "1" dimana pada tampilan akan memberitahu kondisi Low atau High, menghasilkan tegangan dalam sinyal elektriknya. Terdapat 2 sensor float switch yang berada pada 2 tangki *stora*ge, dimana akan menghasilkan 4 kondisi pada cara mengkonversinya.



Gambar 3.11 Flow Chart Sistem

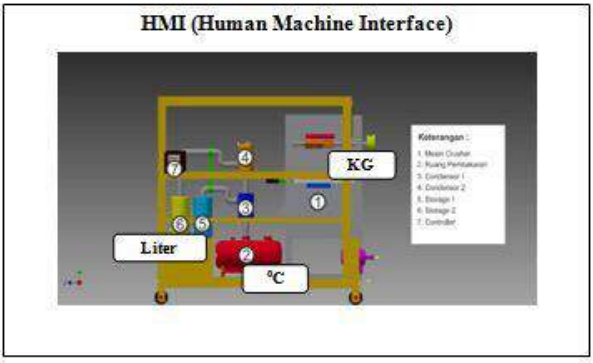
3.2.3 Integrasi Sistem Monitoring

Sistem monitoring pada alat ini akan menggunakan Atmega sebagai mikrokontrollernya dan menggunakan beberapa *interface* dalam menampilkan hasil monitoringnya yaitu LCD dan *interface* berbasis *display object* yaitu HMI *(Human Machine Interface)*serta untuk penyimpanan jangka panjang akan menggunakan openlog data logger dengan menggunakan SD Card.



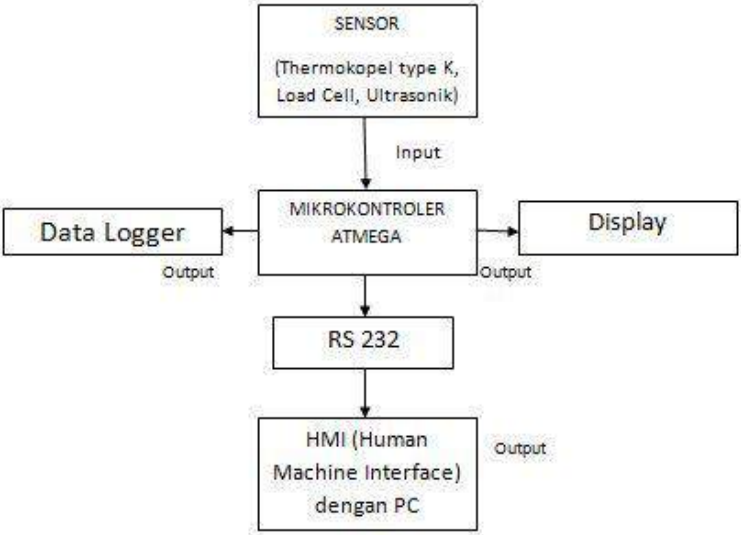
Gambar 3.12 Desain LCD (awal)

Menggunakan LCD Character 4x20 berlatar dengan warna biru. Dimana terdapat nama atau judul pembacaan sensor dan hasil monitoring pembacaan sensor. LCD ini dipasang pada control panel bagian depan dan terdapat juga lampu indikator serta button pemantik. Pada desain ini nama pembacaan sensor yang diperintahkan adalah Suhu, Open, Beban serta Level S dan Level P yang akan menghasilkan angka maupun kondisi High dan Low. Pin yang digunakan pada LCD terdapat 20 Pin, yang digunakan sebanyak 16. Pin yang digunakan yaitu VDD, GND, RW, RS, E, D4, D5, D6, D7. D8 serta A dan K.



Gambar 3.13 Design Display Human Machine Interface (HMI) (awal)

Pada desain HMI *(Human Machine Interface)* menampilkan plant dengan aslinya, menggunakan software wonderware intouch 10.0 dengan pengambaran nyata apa saja yang digunakan, seperti tangki pemanas, sensor, *crusher*, *storage*, aktuator yang sama persis yang digunakan pada plant. Dengan menambahkan tempat untuk menampilkan sistem monitoring yang dijalankan serta tombol untuk mengconnectkan antara HMI *(Human Machine Interface)* pada PC dengan Atmega16 untuk pembacaan sensor.



Gambar 3.14 Diagram I/O

Dimana, sistem untuk data menggunakan komunikasi data serial dengan USARTyang akan diproses pada Atmega16 setelah itu data akan masuk ke LCD dan akan diconverter untuk bisa ditampilkan pada HMI (Human Machine Interface) menggunakan IC MAX232 dan FT232RL USBserta disimpan pada data logger pada SD Card

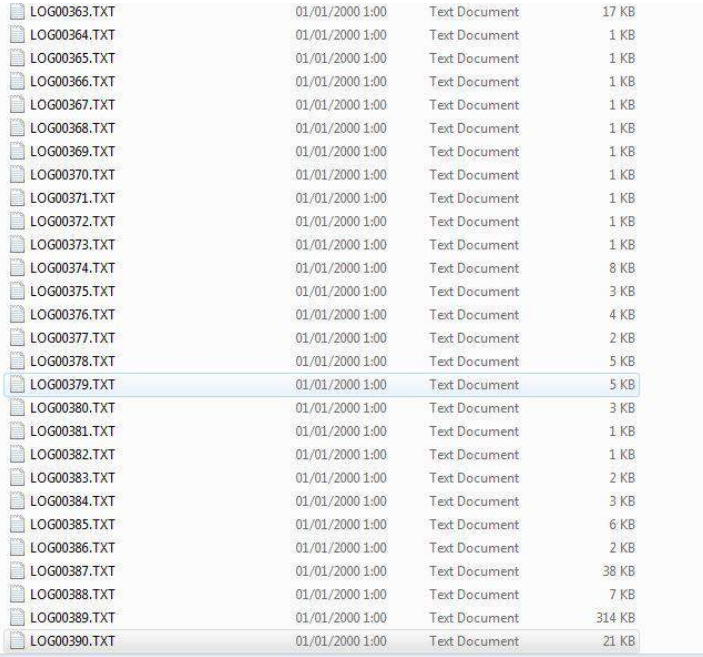
3.2.4 Pengujian Integrasi Sistem Monitoring

Pengujian dimaksudkan untuk memastikan integrasi sistem monitoring variabel proses pada alat ini berjalan dengan seharusnya dan dapat menghasilkan data yang valid, pengujian ini berfungsi sebagai penentu apakah hasil yang diharapkan sesuai atau tidak. Jika hasil tidak sesuai dengan yang diharapkan makan akan dilakukan proses perancangan integrasi sistem monitoring kembali. Adapun tolak ukur dalam keberhasilan ini adalah :

a. Dapat menampilkan pada LCD semua variabel proses dengan menampilkan suhu, open valve, beban dan level dengan Low High
b. Dapat menampilkan pada interface berbasis object pada PC untuk melihat data dan pengaruh dalam perubahan-perubahan yang terjadi berbasis HMI (Human Machine Interface)
c. Dapat menyimpan data pada SD card sebagai media penyimpanan jangka panjang

3.2.5 PembuatanData Logger

Data akan tersimpan pada memori secara terus-menerus. Format data yang digunakan adalah berekstensi .txt serta akan bisa berganti menjadi .csv. Karakter-karakter yang disimpan dalam kartu memori adalah jam, menit, detik, tanggal, bulan, tahun, suhu, oepn valve, beban dan kode level. Dalam data logger sistem dapat menyimpan dalam bentuk .txt maupun .csv dimana bernama “LOG00XX.TXT” atau “LOG00XX.csv”



Gambar 3.15 File data logger dalam penyimpanan SD CARD

Format paket data yang akan disimpan memiliki 40 karakter dan disimpan secara terus menerus didalam kartu memori ketika plant bekerja. Format data yang digunakan adalah berekstensi .txt. Karakter-karakter yang disimpan dalam kartu memori adalah tanggal, jam,suhu,open valve, beban dan kode level. Berikut merupakan format data di dalam sdcard yang dijelaskan pada tabel 3.5 dan tabel 3.6.

Tabel 3.1 Format Data Perwaktuan

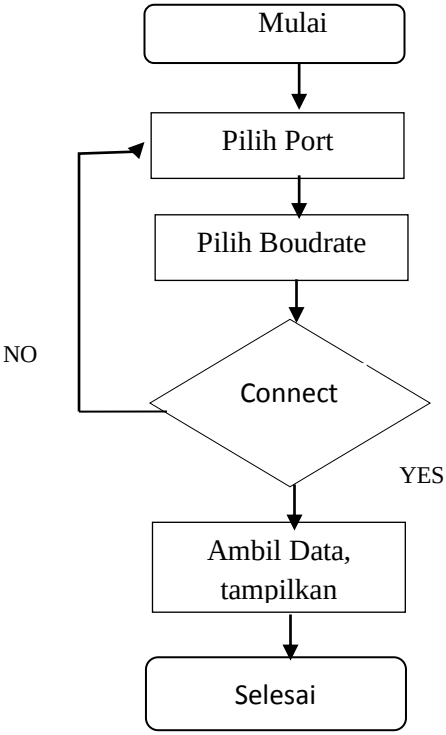
	JAM			TGL		
	hh	mm	ss	dd	mm	yy

Jumlah Karakter	2	2	4	2	2	2

Tabel 3.2 Format Data Pengukuran

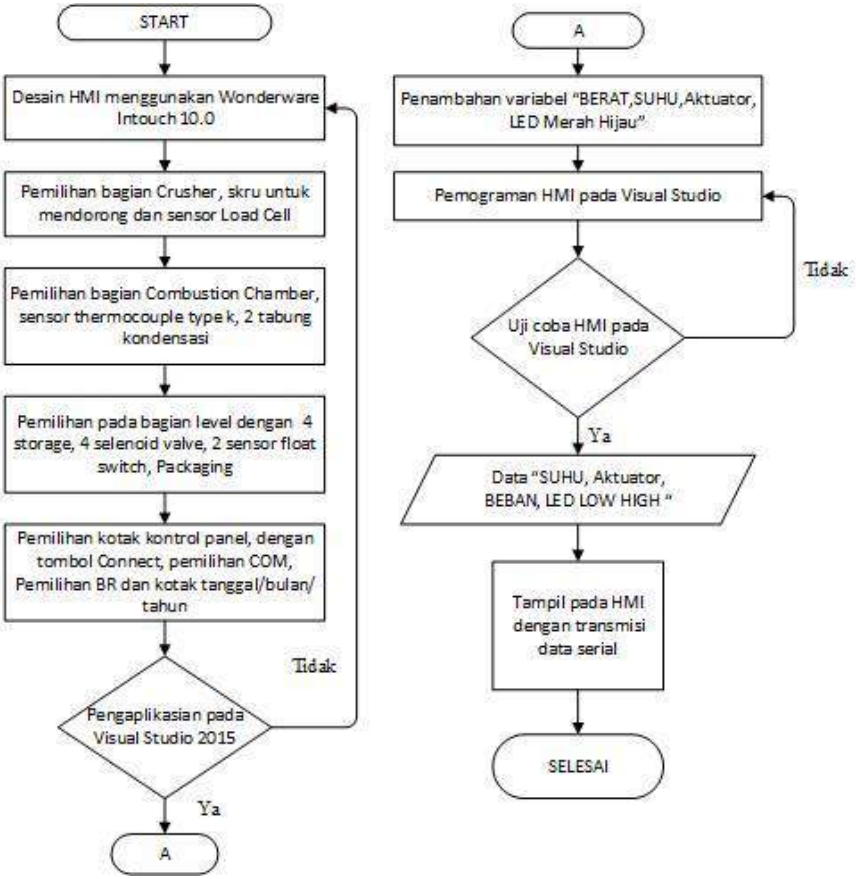
	Suhu ^{0}C	Valve	Beban	Kode level (#x#x)
Jumlah Karakter	5	3	5	4

3.2.6 Pembuatan HMI (human machine interface)



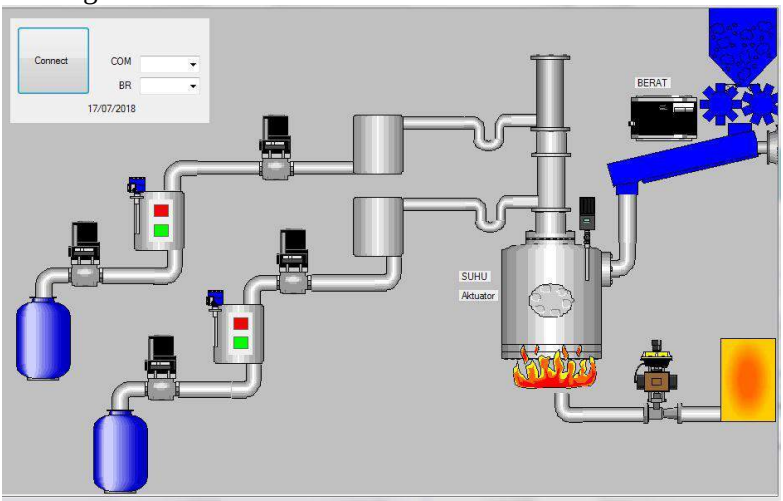
Gambar 3.16 Flowchart Connect HMI

Perancangan Human Machine Interface pada plant perancangan Bahan Bakar Minyak dari plastik mempunyai fungsi untuk memonitoring keadaan 4 variabel yaitu suhu, open valve, beban dan kode level ketika plant berjalan. Pengerjaan HMI menggunakan software visual studio 2015, dengan bahasa Visual Basic.



Gambar 3.17 Flowchart Desain HMI

Pada gambar 3.17 merupakan flowchart sistem HMI dengan hal pertama yang dilakukan yaitu mendesain layout HMI dengan aplikasi wonderware intouch 10.0 dengan gambar instrument yang terdapat pada aplikasi tersebut. Setelah itu membuat pada visual studio 2015 dengan menambahkan kotak variabel suhu, aktuator, berat dan LED sebagai petanda level Low atau High.



Gambar 3.18 Desain HMI pada Visual Studio 2015

Pada gambar 3.17 mengambarkan tentang tampilan HMI (Human Machine Interface) pada PC dengan banyak instrument yang ada. Terdapat crusher, skru dan sensor berat serta kotak variabel berat untuk menampilkan data. setelah proses crusher, masuk pada tank combustion chamber dengan pemanasan gas yang diatur oleh MOV stepper. Menggunakan thermocouple type k dan 2 tabung kondesasi dengan yang memisahkan hasil kondensasi premium dan solar, hasil fluida akan ditampung pada storage dan jika sudah HIGH maka fluida akan menuju packaging dengan LED merah yang akan menyala dan sebaliknya jika sudah LOW maka fluida akan mengisi lagi serta menyalakan LED hijau. Data akan masuk pertama kali jika kita menentukan COM

dan BR yang diinginkan setelah itu tekan tombol Connect. Maka data akan masuk sesuai plant yang berjalan, secara realtime antara LCD dan data logger sistem menggunakan SD Card.

3.2.7 Menganalisis Data

Analisis data digunakan untuk menjawab dan menjelaskan permasalahan yang telah ditemui. Dalam sistem monitoring variabel proses ini perlu ada pembanding antara data yang diambil dari Atmegadengan data pada data logger serta HMI *(Human Machine Interface*

3.2.8 Pengambilan Kesimpulan

Pengambilan kesimpulan dilakukan setelah menganalisis data yang telah didapat. Kesimpulan tersebut akan diolah dan mendapat saran kedepannya agar sistem monitoring variabel proses ini dapat berkembang dan lebih baik lagi.

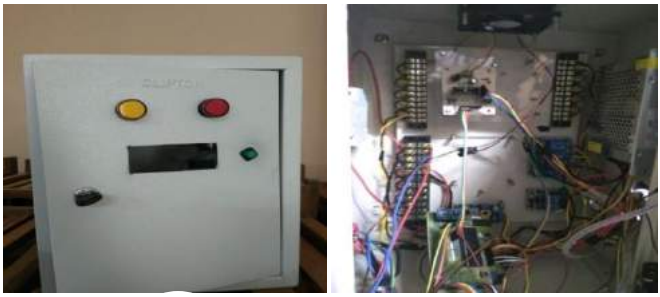
BAB IV
HASIL DAN ANALISA DATA

4.1 Hasil Sistem Monitoring

4.1.1 Rancang Bangun Sistem Monitoring

Setelah perancangan system monitoring variabel proses pada alat produksi Bahan Bakar Minyak dari limbah plastik dibuat maka pengujian baik dari *hardware*, *controlling*, dan *software* perlu dilakukan. Pengujian dilakukan untuk mendapatkan data-data dari alat yang dirancang guna mengetahui spesifikasi serta performansi dari alat secara keseluruhan dan seberapa besar *error* atau kesalahan yang terjadi pada alat berdasarkan respon system dari nilai *set poin* yang diberikan. Secara mekanisme kerja dari perancangan system monitoring variabel proses ada pada alat produksi Bahan Bakar Minyak dari limbah plastik ini untuk mengetahui variabel proses yang terdapat pada plant..

Pada sistemini juga dilengkapidengan RTC (*real time clock*) yang berfungsi untuk menampilkan waktu yang sebenarnya, openlog datalogger yang diggunakan untuk penyimpanan data ke memori SD Card dan LCD sebagai penampil data serta tampilan real time pada HMI (Human Machine Interface)



(a) (b)

Gambar 4.1 (a)control panel tampak depan, (b) tampak dalam

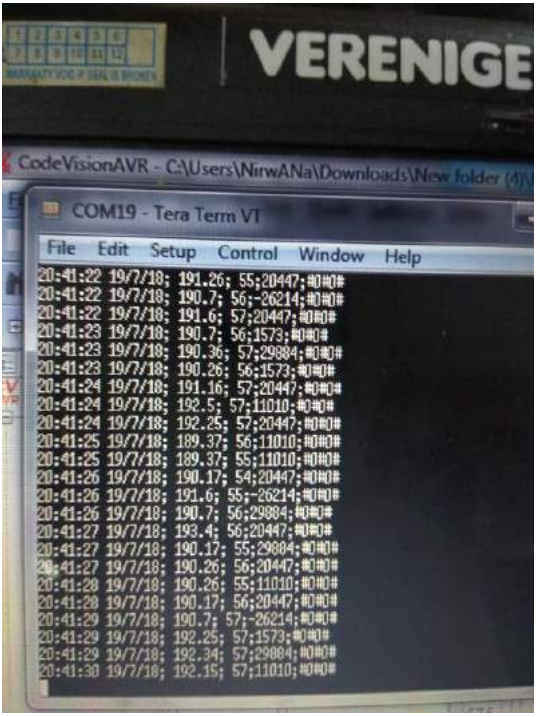
4.1.2 Pengujian *Hardware* Sistem Monitoring Variabel Proses

Pengujian hardware adalah untuk memastikan ATMega 16 dengan open logger, RTC dan LCD terpasang dengan benar. Dimana terdapat Atmega16 yang terpasang sebagai mikrokontroler dan terdapat wiring yang menuju Atmega 16 dari sensor maupun aktuator. Tampilan dari data yang akan diterima oleh mikrokontroller dengan komunikasi serial USART akan menuju LCD 4 x 20 dan HMI (Human Machine Interface) pada PC serta akan tersimpan pada media penyimpanan SD CARD.

Gambar 4.2 Penggabungan ATMega dengan peralatan pendukung

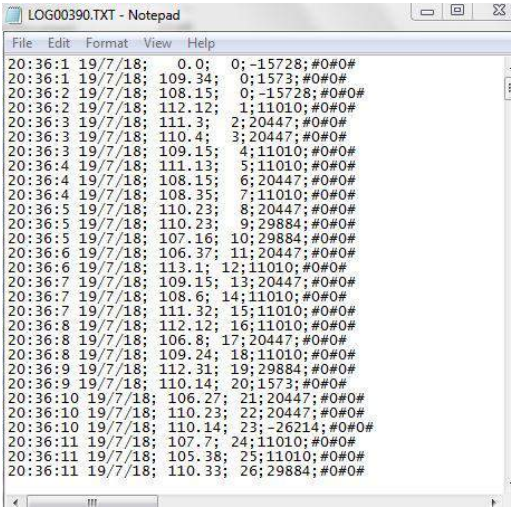
4.1.3 Pengujian *Software* Sistem Monitoring Variabel Proses

Pengujian software bertujuan untuk mengetahui apakah software yang telah dibuat dapat bekerja sesuai dengan yang direncanakan. Dalam hal ini akan program di uji dengan cara menyambungkan ATMega16 dengan Open Logger, RTC dan LCD dengan menggunakan komunikasi serial (USART). Pengujian dilakukan dengan membuka software terminal bernama *Tera Term.*



Gambar 4.3 Tampilan terminal tera term pada PC

Software *Tera Term* berfungsi untuk melihat komunikasi serialnya berjalan atau tidak. Di dalam Atmega terdapat program yang berfungsi sebagai penginput data dari pembacaan mikrokontroller ke terminal serta penyimpanan pada sd card. Data hasil dari pembacaan akan masuk ke dalam data logger pada sd card dan muncul pada LCD character serta akan tampil pada *HMI (Human Machine Interface)*. Selain bisa menggunakan *Tera Term*, untuk mengecek USART berjalan dengan baik, bisa menggunkan *Hyperterminal* atau *Hercules 6.02*. Pengiriman pada *Tera Term* atau USART akan sama dengan pengiriman pada penyimpanan data logger menggunakan Open Log.



(a)

19/07/2018 20:36	0.0	0	-15728	#0#0#
19/07/2018 20:36	109.34	0	1573	#0#0#
19/07/2018 20:36	108.15	0	-15728	#0#0#
19/07/2018 20:36	112.12	1	11010	#0#0#
19/07/2018 20:36	111.3	2	20447	#0#0#
19/07/2018 20:36	110.4	3	20447	#0#0#
19/07/2018 20:36	109.15	4	11010	#0#0#
19/07/2018 20:36	111.13	5	11010	#0#0#
19/07/2018 20:36	108.15	6	20447	#0#0#
19/07/2018 20:36	108.35	7	11010	#0#0#
19/07/2018 20:36	110.23	8	20447	#0#0#
19/07/2018 20:36	110.23	9	29884	#0#0#
19/07/2018 20:36	107.16	10	29884	#0#0#
19/07/2018 20:36	106.37	11	20447	#0#0#
19/07/2018 20:36	113.1	12	11010	#0#0#
19/07/2018 20:36	109.15	13	20447	#0#0#
19/07/2018 20:36	108.6	14	11010	#0#0#
19/07/2018 20:36	111.32	15	11010	#0#0#
19/07/2018 20:36	112.12	16	11010	#0#0#
19/07/2018 20:36	106.8	17	20447	#0#0#
19/07/2018 20:36	109.24	18	11010	#0#0#
19/07/2018 20:36	112.31	19	29884	#0#0#
19/07/2018 20:36	110.14	20	1573	#0#0#
19/07/2018 20:36	106.27	21	20447	#0#0#
19/07/2018 20:36	110.23	22	20447	#0#0#
19/07/2018 20:36	110.14	23	-26214	#0#0#
19/07/2018 20:36	107.7	24	11010	#0#0#

(b)

Gambar 4.4 Data penyimpanan dengan .txt dan .csv

Penyimpanan data menggunkan openlog datalogger berhasil disimpan pada Microsoft excel. Pada modul in menggunakan pin

TX yang akan disambungkan pada RX mikrokontroler Atmega16 yaitu pada PIND.0 dan pin RX modulakandisambungkan pada pin TX mikrokontroler Atmega16 yaitu pada PIND.1. Pengambilan data pada produksi bertujuan untuk mengetahui nilai variabel proses pada plant. Pengambilan data ini diambil data dari suhu, beban dan level pada sdcard dengan pemrosesan menggunakan Atmega16. Penyimpanan data menggunakan micro sdcard berhasil disimpan pada file.txt yang kemudian akan diubah menjadi file.csv jika dibutuhkan. Untuk mengetahui waktu menggunakan Modul real Time Clock (RTC) DS1307, modul ini menggunakan pin Vcc Gnd, pin SDA disambungkan pada pin SDA mikrokontroler Atmega16 pada pin PC1 dan pin SCL pada modul RTC disambungkan pada pin SCL mikrokontroler Atmega16 pada pin PC0.

Komunikasi data yang digunakan untuk modul Open log ini adalah USART, karena membutuhkan port RX untuk jalur perpindahan data dan TX mengirim data Modul open log ini tidak menggunakan protocol selain komunikasi serial USART untuk dapat menyimpan data pada SDCard, mikrokontroler hanya perlu mengirim data secara serial ke open log sehingga bisa terbaca oleh SDCard.

```
// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: On
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud Rate: 9600
UCSRA=0x00;
UCSRB=0x98;
UCSRC=0x86;
UBRRH=0x00;
UBRRL=0x33;
```

Sebelum membuat code untuk bagaimana cara mnegirimkannya, terlebih dulu menginisialisasi USART dengan

pengaturan awal pada CV AVR. Proses inisialisasi mengatur *baudrate* yang digunakan, pengaturan komunikasi parameter, *usart reciever* dan *transmitter* diaktifkan atau tidak serta mode yang digunakan, berikut juga dengan *code-code hexadesimal* yang digunakan dapat disesuaikan dengan yang dibutuhkan.

```
// Timer2 output compare interrupt service routine
interrupt [TIM2_COMP] void timer2_comp_isr(void)
{
// Place your code here
kali=kali+1;
x=0;
if(kali==29000)//100000
{
   x=1;
   kali = 0;
}}
void kirim_data( unsigned char data )
{
   /* Wait for empty transmit buffer */
   while ( !( UCSRA & (1<<UDRE)) )
   ;
   /* Put data into buffer, sends the data */
   UDR = data;
}
void kirim_string(char *s)
{
        while (*s)
        {
                kirim_data(*s++);
        }}
```

Code diatas menggunakan Timer2 Interrupt dengan penjelasan setiap interrupt jalan maka nilai akan selalu ditambah 1. Kode x=0 adalah deklarasi nilai yang berfungsi untuk menghentikan pembacaan data sensor untuk dikirim menuju

USART. Code if(kali=2900) jika nilai kali mencapai 2900 maka if akan melakukan perintah. Code X=1 berfungsi sebagai menjalankan nilai pembacaan sensor untuk dikirim menuju USART dan code kali=0 berfungsi untuk mereset kondisi if agar kembali pada keadaan awal. Pada code diatas menjelaskan tentang code yang digunakan dengan mengunakan "void kirim_data" dan "void kirim_string" dengan ditempatkan diluar while, dengan tujuan "kirim_data" untuk pengiriman data char dan "kirim_string" untuk pengiriman data string. Dimana pada penjelasan code dibawah "kirim_data(59)" merupakan kirim char urutan ke 59, dengan arti pada tabel ASCII yaitu (;) digunakan sebagai pemisah setiap pembacaan sensor serta waktu. Untuk "kirim_string(buff)" sebagai mengirimkan isi dari buff dimana buff ialah bentuk array char yang terdapat pada sprintf() (semacam memory), semuanya hanya sebagai bentuk data yang dikirimkan.

Adapun urutan protokol pada proses penyimpanan data adalah sebagai berikut:

PAKET PENGIRIMAN DATA

Time & Date	(59)	Result	(59)	Valve_open	(59)	Nilai1	(59)	Level Kode	(13)	(10)
A	(59)	B	(59)	C	(59)	D	(59)	E	(13)	(10)

Gambar 4.5 Paket Pengiriman Data pada SDCard

Gambar 4.5 menjelaskan tentang protokol paket pengiriman data sekali pengiriman dengan urutan berikut dari paket A hingga paket E secara terus menerus selama proses produksi berlangsung dan mengirimkan data pembacaan sensor. Sekitar 1 detik terdapat 2 sampai 3 data yang dapat masuk dalam sekali pengiriman.

```
if (x==0){
        sprintf(buff,"%d:%d:%d %d/%d/%d", jam, menit,
detik, tanggal, bulan, tahun);
```

```
        kirim_string(buff);
        kirim_data(59);
         sprintf(buff,"%4u.%u",result/40,(result%40),0xDF);
        kirim_string(buff);
        kirim_data(59);
        sprintf(buff,"%3d",valve_open);
        kirim_string(buff);
        kirim_data(59);
         sprintf(temps,"%3d",nilai1);
        kirim_string(temps);
        kirim_data(59);
        if (k==0 && l == 0){
        kirim_string("#0#0#"); //level 1 low;level 2 low
        }
        else if (k==0 && l == 1){
        kirim_string("#0#1#"); //level 1 low;level 2 high
        }
        else if (k==1 && l == 0){
        kirim_string("#1#0#"); //level 1 high;level 2 low
        }
        else if (k==1 && l == 1){
        kirim_string("#1#1#"); //level 1 high;level 2 high
        }
        kirim_data(13);
        kirim_data(10);
}
```

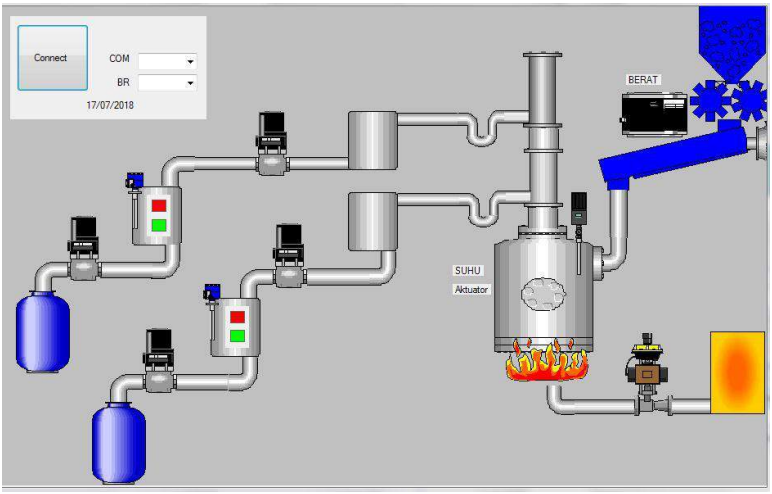
Pengiriman data hasil pembacaan sensor digunakan perintah sprintf(); yang ada pada library stdio.h. Hasil pembacaan sensor lansung dikirimkan menurut protokol pengiriman paket data diatas. Setiap pembacaan sensor terpisahkan oleh (;) dengan menggunakan kode ASCII control character dengan perintah “kirim_data(59)” . Ketika sudah pada pembacaan sensor terakhir yaitu kode level, kemudian menggunkan perintah “kirim_data(10)” untuk memerintah data selanjutnya agar berganti line bawahnya. Kemudian menuliskan perintah

"kirim_data(13)" artinya Carriage return. Untuk fungsi void dipanggil diluar while. Dengan protokol pengiriman data seperti gambar diatas yang berupa pembacaan sensor sert awaktu dan tanggal dapat tersimpan pada SDCard. Format yang telah tersimpan pada SDCard adalah.*TXT*. Untuk dapat melihat hasil dari penyimpanan file .*TXT* tersebut di save as dengan diubah menjadi.*CSV*. Maka data dapat dilihat dari Microsoft excel.

Tabel 4.1 Data Logger Sistem

23/07/2018 13:34	201.16	3	20447	#0#0#
23/07/2018 13:34	201.26	3	-15728	#0#0#
23/07/2018 13:34	201.6	3	-26214	#0#0#
23/07/2018 13:34	201.26	3	-26214	#0#0#
23/07/2018 13:34	201.16	3	20447	#0#0#
23/07/2018 13:34	201.26	3	29884	#0#0#
23/07/2018 13:34	201.16	3	29884	#0#0#
23/07/2018 13:34	201.6	3	-16777	#0#0#
23/07/2018 13:34	201.6	3	29884	#0#0#
23/07/2018 13:34	200.37	3	29884	#0#0#
23/07/2018 13:34	201.6	3	-16777	#0#0#
23/07/2018 13:34	201.6	3	29884	#0#0#
23/07/2018 13:34	200.27	3	11010	#0#0#
23/07/2018 13:34	200.17	3	20447	#0#0#
23/07/2018 13:34	200.27	3	11010	#0#0#
23/07/2018 13:34	200.17	3	-26214	#0#0#
23/07/2018 13:34	200.7	3	20447	#0#0#
23/07/2018 13:34	200.17	3	-16777	#0#0#
23/07/2018 13:34	200.17	3	29884	#0#0#
23/07/2018 13:34	200.27	3	20447	#0#0#
23/07/2018 13:34	200.17	3	20447	#0#0#
23/07/2018 13:34	200.17	3	20447	#0#0#

23/07/2018 13:34	200.17	3	20447	#0#0#
23/07/2018 13:34	200.7	3	29884	#0#0#
23/07/2018 13:34	200.7	3	29884	#0#0#
23/07/2018 13:34	200.7	3	11534	#0#0#
23/07/2018 13:34	200.7	3	-26214	#0#0#
23/07/2018 13:34	200.7	3	20447	#0#0#
23/07/2018 13:34	199.38	3	-26214	#0#0#
23/07/2018 13:34	200.7	3	20447	#0#0#
23/07/2018 13:34	199.38	3	11010	#0#0#
23/07/2018 13:34	199.28	3	-26214	#0#0#
23/07/2018 13:34	199.28	3	29884	#0#0#
23/07/2018 13:34	199.38	3	-26214	#0#0#
23/07/2018 13:34	199.28	3	20447	#0#0#
23/07/2018 13:34	199.28	3	-26214	#0#0#
23/07/2018 13:34	199.28	3	-15728	#0#0#
23/07/2018 13:34	199.28	3	-7340	#0#0#
23/07/2018 13:34	199.18	3	20447	#0#0#
23/07/2018 13:34	199.28	3	-26214	#0#0#
23/07/2018 13:34	199.18	3	-26214	#0#0#
23/07/2018 13:34	199.8	3	20447	#0#0#
23/07/2018 13:34	199.8	3	20447	#0#0#
23/07/2018 13:34	199.18	3	-15728	#0#0#
23/07/2018 13:34	198.38	3	29884	#0#0#
23/07/2018 13:34	199.8	3	20447	#0#0#
23/07/2018 13:34	198.38	3	20447	#0#0#
23/07/2018 13:34	199.8	3	29884	#0#0#
23/07/2018 13:34	199.8	3	11010	#0#0#
23/07/2018 13:34	199.8	3	11010	#0#0#
23/07/2018 13:35	198.29	3	29884	#0#0#
23/07/2018 13:35	198.29	3	29884	#0#0#



Gambar 4.6 HMI (Human Machine Interface) pada PC

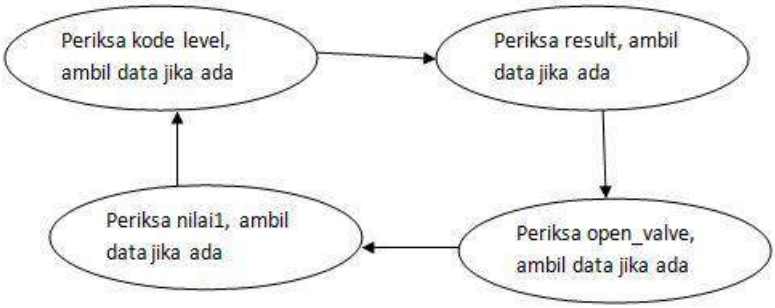
HMI (Human Machine Interface) berfungsi sebagai interface pada pc yang akan menampilkan secara real time data dari sensor yang diambil. Pengiriman data pada HMI (Human Machine Interface) menggunakan komunikasi data serial dengan mode USART, karena membutuhkan port RX untuk jalur perpindahan data dan TX mengirim data mikrokontroler hanya perlu mengirim data secara serial ke PC sehingga bisa terbaca oleh HMI (Human Machine Interface), menggunakan cara pengiriman yang hampir sama dengan pengiriman Open Log.

PAKET PENGIRIMAN DATA

Time & Date	(59)	Result	(59)	Valve_open	(59)	Nilai1	(59)	Level Kode	(13)	(10)
A	(59)	B	(59)	C	(59)	D	(59)	E	(13)	(10)

Gambar 4.7 Paket pengiriman data

Dengan paket pengiriman data sebagai berikut, HMI akan jalan ketika komunikasi serial USART berjalan dengan baik. Dengan komunikasi serial USART yang digunakan pengiriman akan berjalan bergantian. Tetapi pada HMI (Human Machine Interface) harus mengatur pada code untuk visual studio berbasis basic pada Form1.Vb. Pengunaan USART yang dilakukan menggunakan mode polling dimana, memeriksa semua yang ada dan mengambil data jika tersedia. Tetapi tetap untuk memilah-milah setiap sensor diatur pada code visual studio, menggunakan "splitData1" dan "splitData2"



Gambar 4.8 Metode Polling untuk HMI

Pada gambar diatas menjelaskan bahwa, sistem akan terus berjalan secara berkala bergiliran memeriksa data sensor sudah tersedia. Dimana data sensor akan dimasukan pada kotak label yang sudah disediakan untuk bisa ditampilkan.

```
Private Sub Timer1_Tick(sender As Object, e As
EventArgs) Handles Timer1.Tick
'Label2.Text = Convert.ToString(random.Next(0, 1000))
 Dim tag1 As Integer, tag2 As Integer
 Dim led1 As Integer, led2 As Integer
 ' Dim s As String = "1262666; 99; 0.0; 12122f; hghjg;
gfg;"
        tag1 = 0
```

```
        tag2 = 0

receivedData = ReceiveSerialData()
 'Label1.Text &= receivedData
Dim splitData1 As String() = receivedData.Split(New
Char() {";"c})
Dim splitData2 As String() = receivedData.Split(New
Char() {"#"c})

Dim datadiv1 As String
For Each datadiv1 In splitData1
Select Case tag1
 Case 1
 Label1.Text = datadiv1 ' suhu
 Case 2
 Label4.Text = datadiv1 ' aktuator
 Case 3
 Label2.Text = datadiv1 'beban
 End Select

      tag1 += 1
 Next
Dim datadiv2 As String
For Each datadiv2 In splitData2
Select Case tag2
   Case 1
   led1 = CInt(datadiv2) ' level1
   Case 2
   led2 = CInt(datadiv2) ' level2

End Select
        tag2 += 1
   Next

If led1 = 0 And led2 = 0 Then
   Label5.BackColor = Color.Lime
   Label6.BackColor = Color.Gray
   Label7.BackColor = Color.Lime
   Label8.BackColor = Color.Gray
ElseIf led1 = 0 And led2 = 1 Then
   Label5.BackColor = Color.Lime
   Label6.BackColor = Color.Gray
```

```
   Label7.BackColor = Color.Gray
   Label8.BackColor = Color.Red
ElseIf led1 = 1 And led2 = 0 Then
   Label5.BackColor = Color.Gray
       Label6.BackColor = Color.Red
       Label7.BackColor = Color.Lime
       Label8.BackColor = Color.Gray
ElseIf led1 = 1 And led2 = 1 Then
       Label5.BackColor = Color.Gray
       Label6.BackColor = Color.Red
       Label7.BackColor = Color.Gray
       Label8.BackColor = Color.Red
    End If

        tag1 = 0
        tag2 = 0

    End Sub
```

Code diatas menjelaskan tentang "splitData1" untuk mereceived data dengan tanda (;), sedangkan "splitData2" untuk mereceived data dengan tanda (#). Setiap sensor dibedakan dengan case per sensornya, dimana "datadiv1" merupakan inisialisasi dari data yang diterima yang pada "splitData1". Data akan masuk sesuai dengan kotak label yang dituju. Begitu juga dengan "datadiv2" akan menerima data dari "splitData2" berupa kode level dengan tanda "#" dengan case yaitu menyala LED dengan jika sesuai dengan code diatas.

4.1.4 Pengujian Integrasi Monitoring Variabel Proses

Pengujian integrasi Monitoring akan ditampilkan pada LCD Character 4 x 20 serta HMI (Human Machine Interface) dengan menggunakan Visual Studio dan komunikasi data paralel yang akan menghasilkan data real time serta mengetahui data loss pada saat pengiriman data serial. Dimana data akan serentaj untuk masuk pada LCD 4x20 dengan perintah sprintf(); dengan penempatan tata letak pembacaan sensor menggunakan perintah "lcd_gotoxy()" dan perintah memasukan pembacaan sensor pada

kotak dengan perintah “lcd_puts()”. Data akan masuk secara bersamaan dengan tata letak yang berbeda-beda serta dari kotak penyimpanan yang berbeda-beda juga.

Gambar 4.9 Tampilan LCD pada control panel pada saat plant berjalan

4.1.5 Perhitungan Memori pada Penyimpanan Logger

Penyimpanan data logger pada sistem monitoring variabel proses ini menggunakan SD CARD dengan kapasitas 4 gb. Dengan sistem penyimpanan 1 kali proses produksi dimana bisa 30 menit hingga 60 menit, maka diperlukan perhitungan memori pada penyimpanan SD Card dapat menampung berapa kali proses produksi. Pencatatan data yang dilakukan dalam waktu 1 uji coba *running* membutuhkan kapasitas memory sebanyak 314 kilo byte atau 0,314 MB. Micro Sdcard dengan kapasitas 4 GB memiliki nilai kapasitas maksimal yang bias digunakan adalah 3710 MB, Sehingga jumlah pencatatan yang dapat dilakukan dengan menggunakan SD Card yang bekapasitas 4GB adalah sebagai berikut :

Lama Waktu = $\frac{Kapasitas\ SD\ Card}{Ukuran\ file\ per\ produksi}$.....(4.1)

Dari persamaan 4.1 maka penggunaan memory dapat digunakan selama 12.738,8535 produksi per 30 menit atau 6.369,42675 produksi per 60 menit

4.2 Pembahasan

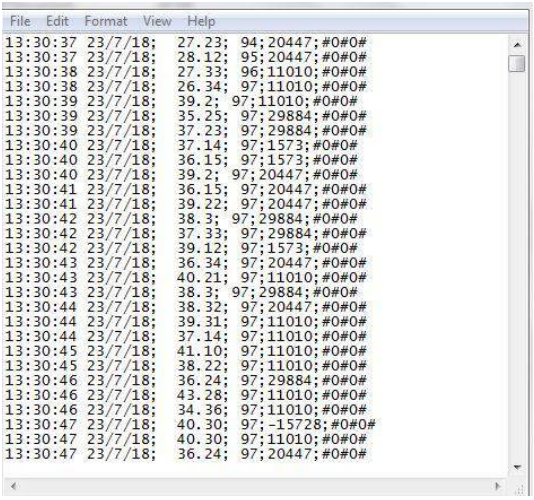
4.2.1 Hasil Perancangan Sistem Monitoring

Pada perancangan sistem monitoring variabel proses pada plant BBM menggunakan Logger Sistem serta HMI (Human Machine Interface) dimana logger sistem serta HMI dibentuk secara real time antara keduanya. Produksi dari plant ini dijalankan selama 30 – 45 menit. Dengan bahan yang digunakan yaitu PP (Polypropylen) sebanyak 3 kg. Awal dari produksi ini adalah mencacah plastik menjadi bahan yang lebih kecil dan sebanyak 3 kg, dimana perhitungannya setiap load cell sensor menimbang 300 gram maka akan menumpahkan bahan plastik menuju skru yang akan mendorong menuju combustion chamber. Setelah masuk sebanyak 3 kg pada combustion chamber, proses pemanasan akan dimulai, dengan menyalakan pemantik kompor secara otomatis. Kompor akan menyala dan proses pemanasan akan dimulai. Awal pertama suhu akan dimulai dengan suhu ruangan berkisar kurang lebih 25.00 ^{0}C. Hingga mencapai setpoint yang ditentukan sesuai dengan titik lebur dari bahan yang digunakan yaitu diatas 300.00 ^{0}C. Tetapi dikarenakan keadaan dari plant yang belum memungkinkan maka dari itu set point diturunkan menjadi 200.00^{0}C. setelah dari combustion chamber yang akan menghasilkan uap, uap akan dikondensasi pada tahap kolom destilasi yang akan bersifat sesuai dengan rantai karbonnya. Jika rantai karbon pendek akan terkondensasi pada kolom yang pertama dan akan menjadi solar, tapi jika memiliki rantai karbon panjang akan naik keatas dan terkondensasi di tabung kedua menjadi premium. Setelah sudah menjadi bahan liquid maka akan di packaging pada botol-botol. Dimana akan ada sensor level yang mendeteksi kapan waktu aktuator membuka untuk mengisi dari botol pada setiap bahan bakar minyak. Semua variabel akan masuk pada tampilan LCD di control panel, serta

data akan masuk pada data logger sistem dan tampilan HMI (Human Machine Interface) secara real time didalam PC. Semua variabel proses terintegrasi pada mikrokontroler Atmega 16 dengan RTC sebagai penanda waktu pada data logger sistem serta FT232RL sebagai USB serial antara mikrokontroler Atmega 16 dengan PC.

4.2.2 Data Logger

Penyimpanan data logger menggunakan komunikasi serail dengan mode USART, pengiriman berupa paket-paket data yang dimana dalam satu paket pengiriman data berisi paket A hingga pake E dari pembacaan sensor yang ada. Dalam akses komunikasi data menggunakn modul Open Logger Spark Fun dengan kapasitas memori 4 GB SDHC, dapat mengjhasilkan beberapa file .TXT sesuai data pengirimannya, tetapi dalam satu file bisa mencukupi satu kali produksi dengan memori yang dihabiskan sekitar 300-400 kb. Adapun hasil data logger dari produksi plastik menjadi bahan bakar minyak, dengan waktu 30 menit dan menggunakan bahan PP (*Poly Prophilen*)


File Edit Format View Help

```
13:30:37 23/7/18;  27.23; 94;20447;#0#0#
13:30:37 23/7/18;  28.12; 95;20447;#0#0#
13:30:38 23/7/18;  27.33; 96;11010;#0#0#
13:30:38 23/7/18;  26.34; 97;11010;#0#0#
13:30:39 23/7/18;  39.2; 97;11010;#0#0#
13:30:39 23/7/18;  35.25; 97;29884;#0#0#
13:30:39 23/7/18;  37.23; 97;29884;#0#0#
13:30:40 23/7/18;  37.14; 97;1573;#0#0#
13:30:40 23/7/18;  36.15; 97;1573;#0#0#
13:30:40 23/7/18;  39.2; 97;20447;#0#0#
13:30:41 23/7/18;  36.15; 97;20447;#0#0#
13:30:41 23/7/18;  39.22; 97;20447;#0#0#
13:30:42 23/7/18;  38.3; 97;29884;#0#0#
13:30:42 23/7/18;  37.33; 97;29884;#0#0#
13:30:42 23/7/18;  39.12; 97;1573;#0#0#
13:30:43 23/7/18;  36.34; 97;20447;#0#0#
13:30:43 23/7/18;  40.21; 97;11010;#0#0#
13:30:43 23/7/18;  38.3; 97;29884;#0#0#
13:30:44 23/7/18;  38.32; 97;20447;#0#0#
13:30:44 23/7/18;  39.31; 97;11010;#0#0#
13:30:44 23/7/18;  37.14; 97;11010;#0#0#
13:30:45 23/7/18;  41.10; 97;11010;#0#0#
13:30:45 23/7/18;  38.22; 97;11010;#0#0#
13:30:46 23/7/18;  36.24; 97;29884;#0#0#
13:30:46 23/7/18;  43.28; 97;11010;#0#0#
13:30:46 23/7/18;  34.36; 97;11010;#0#0#
13:30:47 23/7/18;  40.30; 97;-15728;#0#0#
13:30:47 23/7/18;  40.30; 97;11010;#0#0#
13:30:47 23/7/18;  36.24; 97;20447;#0#0#
```

Gambar 4.10 Data Logger awal produksi

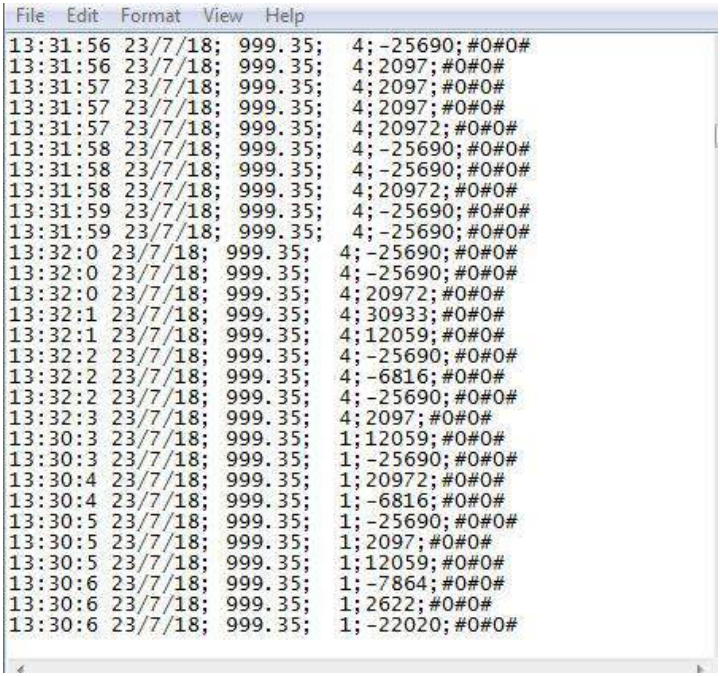
Gambar diatas menjelaskan awal dari temperatur suhu yaitu 27.23 ^{0}C, pada waktu 13.30.37 dengan membuka valve 94 dengan level low – low.

File Edit Format View Help

```
13:34:43 23/7/18; 201.16;  3;29884;#0#0#
13:34:44 23/7/18; 201.6;  3;-16777;#0#0#
13:34:44 23/7/18; 201.6;  3;29884;#0#0#
13:34:44 23/7/18; 200.37;  3;29884;#0#0#
13:34:45 23/7/18; 201.6;  3;-16777;#0#0#
13:34:45 23/7/18; 201.6;  3;29884;#0#0#
13:34:46 23/7/18; 200.27;  3;11010;#0#0#
13:34:46 23/7/18; 200.17;  3;20447;#0#0#
13:34:46 23/7/18; 200.27;  3;11010;#0#0#
13:34:47 23/7/18; 200.17;  3;-26214;#0#0#
13:34:47 23/7/18; 200.7;  3;20447;#0#0#
13:34:47 23/7/18; 200.17;  3;-16777;#0#0#
13:34:48 23/7/18; 200.17;  3;29884;#0#0#
13:34:48 23/7/18; 200.27;  3;20447;#0#0#
13:34:49 23/7/18; 200.17;  3;20447;#0#0#
13:34:49 23/7/18; 200.17;  3;20447;#0#0#
13:34:49 23/7/18; 200.17;  3;20447;#0#0#
13:34:50 23/7/18; 200.7;  3;29884;#0#0#
13:34:50 23/7/18; 200.7;  3;29884;#0#0#
13:34:50 23/7/18; 200.7;  3;11534;#0#0#
13:34:51 23/7/18; 200.7;  3;-26214;#0#0#
13:34:51 23/7/18; 200.7;  3;20447;#0#0#
13:34:52 23/7/18; 199.38;  3;-26214;#0#0#
13:34:52 23/7/18; 200.7;  3;20447;#0#0#
13:34:52 23/7/18; 199.38;  3;11010;#0#0#
13:34:53 23/7/18; 199.28;  3;-26214;#0#0#
13:34:53 23/7/18; 199.28;  3;29884;#0#0#
13:34:53 23/7/18; 199.38;  3;-26214;#0#0#
13:34:54 23/7/18; 199.28;  3;20447;#0#0#
```

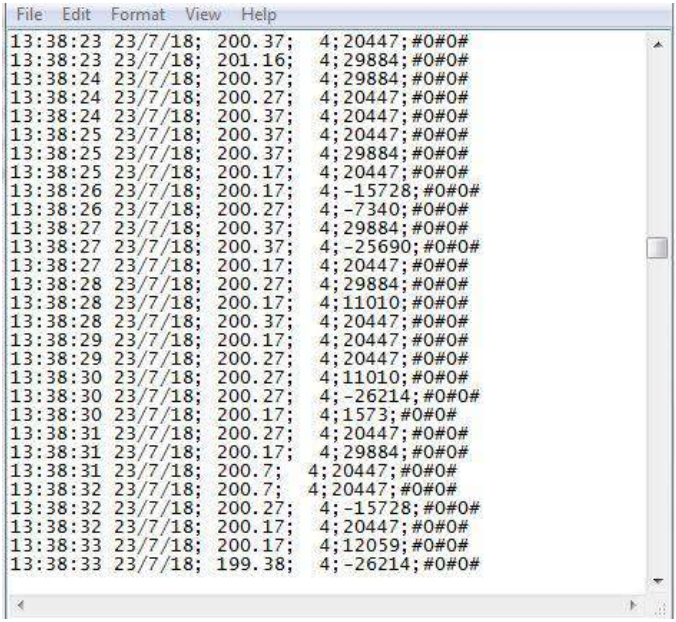
Gambar 4.11 Data Logger Sistem pada saat 200 ^{0}C

Gambar diatas menjelaskan bahwa untuk mendapat suhu 200 ^{0}C dibutuhkan waktu sekitar 4 – 5 menit dengan suhu yang steady state serta open valve 3 dan level low – low.



Gambar 4.12 Error pada Sistem

Gambar diatas menjelaskan tentang, pada saat mencapai suhu 200 ^{0}C tenyata terdapat error sehingga mengulang lagi pada waktu 13:30:2. Error dikarenakan sensor thermocouple type K mengalam grounding, tetapi penangannya memberi kabel grounding pada sensor sehingga kembali seperti semula.

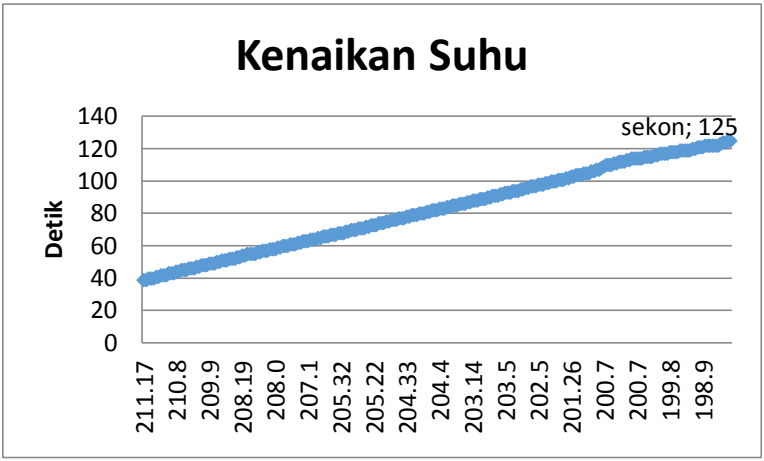


File Edit Format View Help

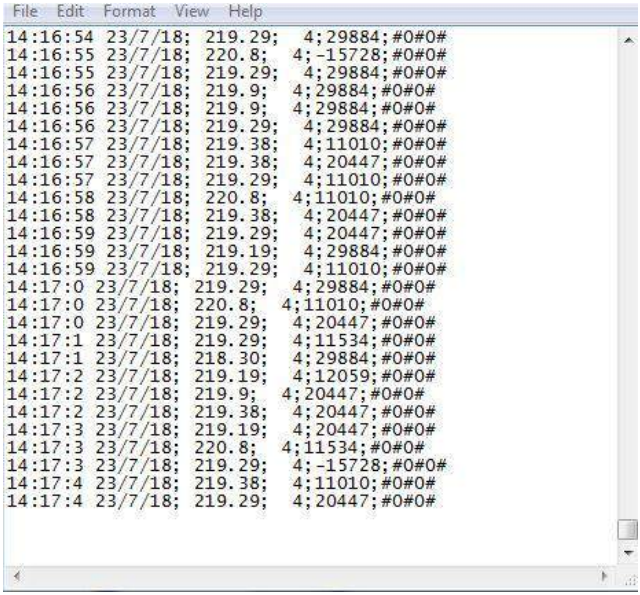
```
13:38:23 23/7/18;  200.37;   4;20447;#0#0#
13:38:23 23/7/18;  201.16;   4;29884;#0#0#
13:38:24 23/7/18;  200.37;   4;29884;#0#0#
13:38:24 23/7/18;  200.27;   4;20447;#0#0#
13:38:24 23/7/18;  200.37;   4;20447;#0#0#
13:38:25 23/7/18;  200.37;   4;20447;#0#0#
13:38:25 23/7/18;  200.37;   4;29884;#0#0#
13:38:25 23/7/18;  200.17;   4;20447;#0#0#
13:38:26 23/7/18;  200.17;   4;-15728;#0#0#
13:38:26 23/7/18;  200.27;   4;-7340;#0#0#
13:38:27 23/7/18;  200.37;   4;29884;#0#0#
13:38:27 23/7/18;  200.37;   4;-25690;#0#0#
13:38:27 23/7/18;  200.17;   4;20447;#0#0#
13:38:28 23/7/18;  200.27;   4;29884;#0#0#
13:38:28 23/7/18;  200.17;   4;11010;#0#0#
13:38:28 23/7/18;  200.37;   4;20447;#0#0#
13:38:29 23/7/18;  200.17;   4;20447;#0#0#
13:38:29 23/7/18;  200.27;   4;20447;#0#0#
13:38:30 23/7/18;  200.27;   4;11010;#0#0#
13:38:30 23/7/18;  200.27;   4;-26214;#0#0#
13:38:30 23/7/18;  200.17;   4;1573;#0#0#
13:38:31 23/7/18;  200.27;   4;20447;#0#0#
13:38:31 23/7/18;  200.17;   4;29884;#0#0#
13:38:31 23/7/18;  200.7;   4;20447;#0#0#
13:38:32 23/7/18;  200.7;   4;20447;#0#0#
13:38:32 23/7/18;  200.27;   4;-15728;#0#0#
13:38:32 23/7/18;  200.17;   4;20447;#0#0#
13:38:33 23/7/18;  200.17;   4;12059;#0#0#
13:38:33 23/7/18;  199.38;   4;-26214;#0#0#
```

Gambar 4.13 Data Logger mencapai suhu 200^0C

Gambar diatas menjelaskan setelah error yang terjadi dan mengulang waktu kembali. Waktu yang dibutuhkan untuk mencapai suhu setpoint yaitu sekitar 7-8 menit. Sehingga total keseluruhan waktu yang dibutuhkan sekitar 10-11 menit untuk mencapai suhu setpoint yaitu 200^{o}C. Dengan peningkatan suhu yang terus menerus terjadi maka grafik pada gambar 4.12 menjelaskan tentang kenaikan suhu yang terjadi setelah terjadi error dimana membutuhkan waktu 3-4 menit untuk mencapai setpoint 200^0C hingga akan terjadi perubahan suhu yang tetap naik dikarenakan data yang diambil tetap setiap 1 detik bisa terdapat 2-3 data suhu yang diambil. Sehingga menghasilkan grafik sebagai berikut:



Gambar 4.14 Grafik Kenaikan Suhu pada menit ke 9 setelah terjadi error.


File Edit Format View Help

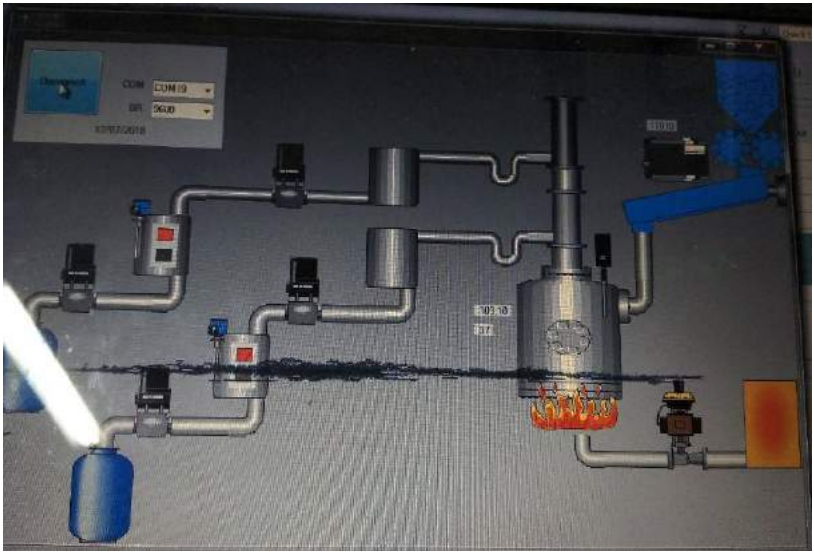
```
14:16:54 23/7/18; 219.29;   4;29884;#0#0#
14:16:55 23/7/18; 220.8;   4;-15728;#0#0#
14:16:55 23/7/18; 219.29;   4;29884;#0#0#
14:16:56 23/7/18; 219.9;   4;29884;#0#0#
14:16:56 23/7/18; 219.9;   4;29884;#0#0#
14:16:56 23/7/18; 219.29;   4;29884;#0#0#
14:16:57 23/7/18; 219.38;   4;11010;#0#0#
14:16:57 23/7/18; 219.38;   4;20447;#0#0#
14:16:57 23/7/18; 219.29;   4;11010;#0#0#
14:16:58 23/7/18; 220.8;   4;11010;#0#0#
14:16:58 23/7/18; 219.38;   4;20447;#0#0#
14:16:59 23/7/18; 219.29;   4;20447;#0#0#
14:16:59 23/7/18; 219.19;   4;29884;#0#0#
14:16:59 23/7/18; 219.29;   4;11010;#0#0#
14:17:0 23/7/18; 219.29;   4;29884;#0#0#
14:17:0 23/7/18; 220.8;   4;11010;#0#0#
14:17:0 23/7/18; 219.29;   4;20447;#0#0#
14:17:1 23/7/18; 219.29;   4;11534;#0#0#
14:17:1 23/7/18; 218.30;   4;29884;#0#0#
14:17:2 23/7/18; 219.19;   4;12059;#0#0#
14:17:2 23/7/18; 219.9;   4;20447;#0#0#
14:17:2 23/7/18; 219.38;   4;20447;#0#0#
14:17:3 23/7/18; 219.19;   4;20447;#0#0#
14:17:3 23/7/18; 220.8;   4;11534;#0#0#
14:17:3 23/7/18; 219.29;   4;-15728;#0#0#
14:17:4 23/7/18; 219.38;   4;11010;#0#0#
14:17:4 23/7/18; 219.29;   4;20447;#0#0#
```

Gambar 4.15 Data Logger Produksi Akhir

Gambar diatas menjelaskan bahwa waktu terakhir untuk produksi yaitu 14:17:4, dimana waktu yang dibutuhkan sekitar 45-47 menit dalam satu kali produksi dengan error yang ada serta menggunakan bahan plastik PP.

4.2.3 HMI (Human Machine Interface)

Adapun tampilan pada HMI (Human Machine Interface) menggunakn Visual Studio 2015 dengan bahasa pemograman basic. Dengan menggunakan layout dari gambar aplikasi wonderware intouch 10.0 dan dilengkapi dengan gambar instrument dari aplikasi. HMI (Human Machine Interface) akan dapat digunakan jika serial usb dipasang antara mikrokontroler dengan PC. HMI akan berjalan real time sesuai dengan kondisi plant berjalan sesuai juga dengan tampilan LCD pada control panel.



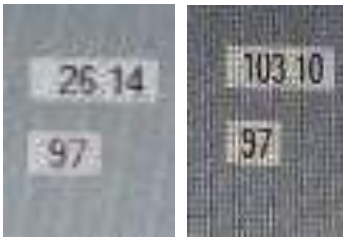
Gambar 4.16 HMI (*Human Machine Interface*) pada saat dijalan

Gambar 4.13 menjelaskan tentang desain HMI yang digunakan untuk tampilan pada PC. Terdapat kolom COM sebagai com berapa yang akan kita pilih sebagai serial usb yang masuk, BR sebagai berapa baud rate yang kita pilih serta connect untuk mengkoneksikan antara data yang masuk dengan tampilan HMI yang sudah dibuat. Jika kita mengklik tombol connect maka data akan otomatis masuk pada kotak sensor (SUHU dan BERAT) , aktuator akan masuk pada kotak Aktuator serta tampilan lampu LED yang akan menandakan LOW atau HIGH dari sensor level, dimana hijau untuk LOW dan merah untuk HIGH. Berikut tampilan jika HMI (*Human Machine Interface*) dijalan pada PC dengan plant berjalan serta pembacaan sensor yang masuk pada Atmga16.

Gambar 4.17 Conncet HMI

Pada awal untuk bisa memonitoring plant yang sedang berproduksi men*connect*kan antara PC dengan USB FT232RL yang terpasang pada Atmega16 yang dipasang pada tx rx pada Atmega 16 pada PD 0 dan PD 1. Setelah itu mengisi COM yang dipasang untuk meng*connect*kan, pada PC yang saya gunakan saya meng*connect*kan pada COM19 dengan USB FT232RL yang saya gunakan. Setelah itu mengatur BR (baudrate) yang digunakan agar data tidak bergeser dan sesuai dengan pembacaan sensor yang dibaca oleh SDCard maupun LCD 4x20. Pada tampilan HMI juga terdapat tanggal/bulan/tahun pada saat digunakan. Setelah mengatur COM maupun BR, tekan tombol

Connect secara 2 kali untuk memulai memonitoring dan akan berjalan secara real time.

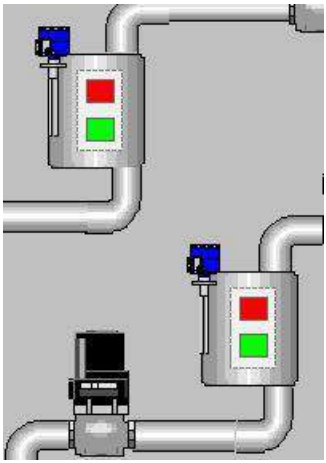


Gambar 4.18 Monitoring Suhu

Pembacaan sensor yang akan dibaca adalah salah satunya suhu dengan menampilkan suhu derajat dengan koma serta berapa besar aktuator MOV stepper membuka untuk menyalurkan gas agar kompor menyala. Jika suhu sudah mencapai sekitar *setpoint* maka MOV Stepper akan berubah menjadi angka yang lebih kecil seperti "3 atau 4" sehingga api yang muncul pada kompor akan meredup dan pemanasan akan menurun hingga stabil dengan *setpoint*.

Gambar 4.19 Sensor beban menggunakan potensiometer

Sensor beban akan berupa angka bulat tanpa satuan, dimana menggunakan potensiometer. Jika potensiometer diputar makan akan berubah nilainya. Potensiometer akan masuk pada Port ADC pada Atmega16 dengan menggunakn modul ads1115.



Gambar 4.20 Sensor level akan dikondisikan berupa LED

Sensor yang terakhir yang digunakan pada plant BBM adalah sensor Float Switch untuk mengsensor level pada storage untuk dipacking pada botol bensin yang pas, dimana menggunakan 4 selenoid valve yang berjalan bergantian sesuai perintah sensor yang aktif. Sensor yang digunakan terdapat pada high dan low storage, dimana pengaktifan sensor sesuai dengan kondisi "0" dan "1" dimana kondisi "1" sensor *on* atau yang akan menandakan level pada *storage high* dan kondisi "0" sensor *off* atau yang akan menandakan level pada *storage low.* Untuk menampilkan kondisi tersebut dibuatlah lampu LED dengan warna yang berbeda bertujuan untuk menandakan sensor itu *Low* atau *High*, dimana warna merah akan menandakan sensor akan *high* dan membuang bensin pada storage serta warna hijau akan menandakan sensor akan *low* dan menutup aktuator dan mengisi storage dengan membuka aktuator atas.

Gambar 4.21 Sensor level keadan High

Sensor level akan kondisi seperti gambar diatas jika level sudah mencapai *setpoint* dan akan membuka selenoid bawah serta mengisikan bensinnya pada botol yang sudah disiapkan. Kondisi diatas membuat sensor *high* dan membuka aktuator bawah serta menutup aktuator pengisi atau atas dari storage pertama. Jika sudah mencapai keadaan dimana bensin sudah mencapai titik *low*, maka akan berubah pada kondisi LED dengan nyala hijau.

BAB V
PENUTUP

5.1 Kesimpulan

Adapun kesimpulan yang dapat disimpulkan dalam pengerjaan Tugas Akhir "Sistem Monitoring Variabel Proses pada Rancang Bangun Alat Produksi Bahan Bakar Minyak (BBM) dari Limbah Plastik Menggunkan HMI (*Human Machine Interface*) adalah sebagai berikut :

a. Kesimpulan yang akan diambil nanti dari hasil serta analisa data dari data hasil pengujian *software* dapat dilakukan pada sebuah sistem monitoring variabel proses sangat memungkinkan untuk dilakukan karena berdasarkan hasil pengujian *software* komunikasi data serial antara data sensor dengan *interface* berhasil secara real time dengan menampilkan variabel proses yang dimonitoring pada penyimpanan data serta HMI (human machine interface) tetapi masih ditemukan *data* yang tidak dapat terbaca sehingga data yang ditampilkan pada *HMI (Human Machine Interface)* tidak sama persis dengan yang ditampilkan pada data hasil dari pembacaan ATMega16 pada data logger. Dari data yang didapat pada penyimpanan yang menyimpan sebanyak 10000 data, dengan error yang ada, didapatkan data yang masuk sebanyak kurang lebih 80% dari data yang masuk.
b. Penggunaan memory pada satu kali produksi dengan menggunakan Modul *Open Log* serta memory SDHC 4 GB menghabiskan sekitar 300-500 Kb. Dengan perhitungan yang sudah dilakukan *memory* dapat digunakan selama 12.738,8535 produksi per 30 menit atau 6.369,42675 produksi per 60 menit.
c. Dalam satu kali produksi dengan memperhatikan sistem monitoring jangka panjang menggunakan SD card, waktu yang dibutuhkan untuk mencapai setpoint yang ditentukan yaitu sekitar 10-11 menit dari awal proses

pemanasan pada *combustion chamb*er dimana menggunakan bahan bakar plastik jenis PP (*Polypropilen*) dengan *setpoint* 200^0C

5.2 Saran

Penelitian ini masih jauh dari sempurna, masih butuh pengembangan-pengembangan ke depan agar mendapatkan hasil yang lebih baik. Saran untuk pengembangan penelitian ini antara lain:

a. Metode pemrograman sangat diperlukan untuk mencapai hasil yang baik sesuai dengan parameter yang telah dirancang dan diinginkan sebelumnya.
b. Dalam perancangan dan pengerjaan *plant* oleh sebuah kelompok diperlukan komunikasi yang baik antar anggota kelompok untuk memperkecil kesalahan atau kegagalna dalam pengerjaan alat atau *plant* sudah dirancang. Karena jika ada kekurangan dalam pembangunan suatu *plant*, dapat mengurangi keterandalan suatu *plant* dan *plant* tidak akan berjalan dengan sepenuhnya sesuai dengan fungsi-fungsi yang diinginkan atau bahkan *plant* tidak akan berfungsi sama sekali.
c. *Troubleshooting* harus dilakukan setelah *plant* hampir mendekati *final* atau terselesaikan. Karena jika *troubleshooting* tidak dilakukan pada saat proses pembangunan *plant* akan lebih mempersulit perancang dan pembangun *plant* ketika pada saat proses atau sistem suatu *plant* tersebut sedang beroperasi.

DAFTAR PUSTAKA

[1] Kurniawan, D. (2015, Juli 20). Fungsi Port Port Pada ATmega 8535, ATmega 8, AT89s51. Diambil kembali dari Ignatius: http://ignatius.ilearning.me

[2] Riyadi, G. (2016, November 24). Cara Menggunakan PuTTY, Panduan Simpel untuk Pemula! Diambil kembali dari Gege Riyadi: https://gegeriyadi.com

[3] Sibro, M. (2016). Pengertian Fungsi Manfaat SSH (Secure Shell). Diambil kembali dari Si Bro 21: http://www.sibro21.org

[4] Suyadi. (2012). Komunikasi Serial dan Port Serial. Surakarta: Teknik Informatika Universitas Muhammadiyah Surakarta.

[5] Triasanti, D. (2017, Juni 6). Konsep Dasar Python. Diambil kembali dari http://andriyani.staff.gunadarma.ac.id

[6] UBAYA, U. S. (2010, September 2). Android: Sistem Operasi Pada Smartphone. Diambil kembali dari UBAYA Universitas Surabaya: http://www.ubaya.ac.id

[7] Ulinnuha, M. A. (2016, September 5). Mengirim dan Menerima Data Melalui Serial UART (TX RX) Raspberry Pi. Diambil kembali dari Blog Ulindev: http://blog.ulindev.com

[8] Lamsami, M (2014). Transmisi Data. Diambil kembali dari Jurnal komdat2.pdf

[9] Susana, Ratna (2016, Desember). Penerapan Metoda Serial Peripheral Interface (SPI) pada Rancang Bangun Data Logger berbasis SD card. Diambil kembali dari Jurnal ELKOMIKA Vol. 4 No. 2 Halaman 208 – 227

[10] Atmel. (2011). Atmega16. San Jose, USA.

[15] Innovativeelectronics. (2012). Manual DT-Sense Gas Sensor. Surabaya, Indonesia.

[16] https://proyekarduino.wordpress.com/2015/04/01/pengetahua n-dasar-rtc-ds1307/ diakses pada tanggal 01 April 2015

[17] https://www.sparkfun.com/products/13712 SparkFun OpenLog, diakses pada tahun 2003

[18]Bentley, John P. 2005. Principles of Measurement Systems. Prentice Hall, London

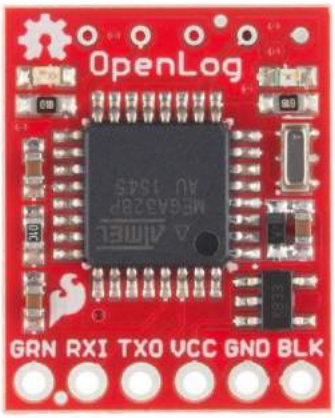
[11] https://purnomosejati.wordpress.com/2011/08/25/mengenal-komunikasi-i2cinter-integrated-circuit/, diakses pada tahun 2018

[12]http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b.pdf, diakses pada Januari 2018

[13] https://boimcontrol.wordpress.com/2011/09/22/belajar-wonderware-intouch-hmi-untuk-koneksi-modbus/ , diakses pada Januari 2018

LAMPIRAN A

OPEN LOGGER SPARK FUN



OpenLog is a simple serial logger based on the ATmega328 running at 16MHz. OpenLog is able to talk to very large capacity (tested up to 64GB) SD cards. The whole purpose of this logger was to create a logger that just powered up and worked. OpenLog ships with a standard serial bootloader so you can load new firmware with a simple serial connection.

###The Basics###

- OpenLog runs at 3.3-5V at 9600bps by default. The baud rate is configurable from 300bps to 1000000bps. We recommend you modify the config file to work at a different serial speed, but you can also reconfigure OpenLog via software commands. See the command example sketchfor more information.
- The microSD card can be any size from 64MB to 16GB. Before using OpenLog be sure to format the card either FAT16 or FAT32. We recommend using Windows to format your card. If using Linux, be sure to create a DOS filesystem after formatting the card.
- During power up, you will see **12**> or **12**<. **1** indicates the serial connection is established. **2**indicates the SD card has been successfully initialized.
- < indicates OpenLog is ready and will log any serial data received (this is the default mode)
- > indicates OpenLog is ready to receive commands.
- Type **?** at the > prompt to bring up a list of supported commands.
- If you are actively logging in NewLog or SeqLog mode, sending Ctrl+z (ASCII 26) three times will exit logging mode and enter command mode.
- For a full list of commands, see the Command Set page.

###Connections###

- **GRN**: Reset pin and connects to the **GRN** pin on the Arduino Pro Mini. Pulling this line low will reset the ATmega328. Because there is a capacitor on this line, holding this line low will *not*keep OpenLog in reset.
- **RXI**: Serial input into OpenLog.
- **TXO**: Serial output from OpenLog.
- **VCC**: 3.3V to 12V input. We recommend 3.3V to 5V.
- **GND**: Ground
- **BLK**: This pin is connected to GND. Connect this pin to **BLK** on the Arduino Pro Mini.

The four pins shown at the top of the board are connected to the SPI pins for programming. We use a special pogo-pin jig to program the serial bootloader onto each board. You are welcome to connect to them but realize the SPI pins are shared with the interface to the SD socket so you might not want to use them as GPIOs. From left to right, the pins are RST, SCK, MOSI, and MISO respectively. You can find more information regarding this in the Eagle files.

###Status LEDs###

- STAT1 LED is the LED shown above, right of the word OpenLog and is sitting on PD5 (Arduino D5) This LED toggles on/off every time a new character is received. This LED helps troubleshoot and indicate serial communication is working.
- STAT2 LED is the LED shown above, left of the word OpenLog and is sitting on PB5 (Arduino D13) This LED is attached to the SPI Serial Clock line. You will see this LED flash rarely. It only turns on when the SPI interface is active and this is rare as the OpenLog buffers 512 bytes at a time before recording to the SD card. Recording 512 bytes is very fast so the LED is on for very little.

###Features###

- Supports automatic log generation and recording. Turn on OpenLog, wait ~2 seconds and start throwing text at it!
- Supports any baudrate from *300* to *1,000,000* at 8-N-1.
- Supports 8.3 file names. "12345678.123" is the longest name.
- All file names are alpha-numeric. "MYLOG1.SZZ" is ok, "Hi !e_.txt" may not work.
- Recording constant 9600bps datastreams are supported. Throw it everything you've got! Higher datarates are supported with some considerations.
- The change directory command is a bit weird. Normally it's 'cd..' but to change to a lower dir, use 'cd ..' (space between cd and ..)
- If you get OpenLog stuck into an unknown baudrate, there is a safety mechanism built-in. Tie the RX pin to ground and power up OpenLog. You should see the LEDs blink back and forth for 2 seconds, then blink in unison. Now power down OpenLog and remove the RX/GND jumper. OpenLog is now reset to 9600bps with an escape character of ctrl+z sent three consecutive times. **Note:** This feature can be overridden if needed. See configuration file for more information.
- Pre-programmed STK500 (Arduino Uno compatible) serial bootloader running at 57600bps @ 16MHz with Optiboot improvements.

Bootloader Note: The preloaded Optiboot serial bootloader uses the upper 500 bytes of flash. If you are modifying the stock OpenLog firmware and the new code is larger than 32,256, you will get verification errors during serial bootloading. **Warning:** some early units (sold in December of 2009) of OpenLog did not have a bootblock protection lock bit set and will overwrite the bootloader. To check if you have one of these versions, drop to command mode and type **?**. The OpenLog firmware version number is shown at the top of the menu. Everything after v1.0 (v1.1 and above) is good and does not have this problem.

###Power### Input voltage on VCC can be 3.3 to 12V. Input voltage on **RXI** pin must not exceed 6V. Output voltage on **TXO** pin will not be greater than 3.3V. This may cause problems with some systems - for example if your attached microcontroller requires 4V minimum for serial communication (this is rare).

OpenLog has reverse power protection. The Micrel voltage regulator can take some serious abuse (reverse power applied, over current shut down, over voltage protection). All parts are static sensitive, but the ATmega328 and Micrel regulator have built-in static protection.

Current consumption:

- 2mA Idle
- 6mA Actively writing to a file

6mA is rare. The vast majority of the time OpenLog is idle. Writing to the SD card (6mA) happens once a 512 byte buffer fills up. Recording that buffer completes in a fraction of a second so the average consumption is very near 5mA unless you are pounding the serial port at 115200bps with a constant data stream.

Note: OpenLog may lose characters if power is removed. During an append, OpenLog will buffer 512 characters at a time. That means that if the system loses power while reading in characters, you may loose up to, but no more than, 511 characters. This is important for low power systems where you may not know when the battery or power will die. OpenLog should record each buffer as it receives each 512 byte chunk. The only way to exit an append is with Ctrl+z (ASCII 26). In firmware v1.3 and above, OpenLog has an auto-store feature. If OpenLog is idle for more than 2 seconds, it will auto-save any characters in the buffer. This is very helpful for systems that store a few characters every few seconds. This feature also significantly saves on power.

###Dimensions###

###Troubleshooting###

The easiest way to get OpenLog working is with a serial connection to a computer. Power up OpenLog and you should see **12<**. If you don't, make sure your **TXO** and **RXI** pins are connected correctly. **TXO** is an output pin from OpenLog and will need to be connected to a input pin on your serial conversion board.

I don’t know what baud rate I put it into! Help! Emergency reset - aka factory defaults. If you get OpenLog stuck into an unknown baud rate, there is a safety mechanism built-in. Tie the RX pin to ground and power up OpenLog. You should see the LEDs blink back and forth for 2 seconds, then blink in unison. Now power down OpenLog and remove the RX/GND jumper. OpenLog is now reset to 9600bps. After a power up you should see **12<**. To get OpenLog into command mode, press ctrl+z three times.

OpenLog communicates with TTL, not RS232, because it is meant to be connected with a microcontroller or an embedded project. If you are connecting OpenLog to a computer, you will need a TTL-to-RS232 converter board such as the RS232 Shifter board, FTDI Basic, or the FT232 Breakout.

OpenLog has two onboard LEDs. STAT1 will blink with an error code if something is wrong. Currently there are two error codes:

- 3 Blinks: The SD card failed to initialize. You may need to format the card with FAT/FAT16 on a computer.
- 5 Blinks: OpenLog has changed to a new baud rate and needs to be power cycled.

The *Card Detect* feature of the microSD socket is connected to the ATmega328 but we are not currently checking for physical presence. This is because there are some SD sockets that have this feature available, and some that do not. When we began production of OpenLog we were not sure which socket would be available so we skipped this check in the firmware.

What is the limit on the number of files I can create in the root directory? Currently the log number limit is 65,534. You can load hundreds of files into the root directory, but OpenLog will perform more and more slowly as more files are introduced. We recommend logging to a freshly formatted FAT16 microSD card. 100 or 200 files/logs is fine. As you approach thousands of files OpenLog can take multiple seconds to create a new file and start to log.

What is the limit of sub-directories I can create? OpenLog currently supports two sub directories but can be increased by changing the code, recompiling, and loading the new firmware onto OpenLog. Change the definition of FOLDER_TRACK_DEPTH from 2 to the number of sub-directories you need to support.

Example Arduino Code: The easiest way to use OpenLog with an Arduino is to simply attach the RXI pin on the OpenLog to the TX pin on the Arduino. Anything that the Arduino outputs (sensor readings, GPS coordinates, etc) will be recorded.

```
Serial.begin(9600); //9600bps is default for OpenLog
Serial.println("123");
```

will cause the Arduino to output **123** at 9600bps and will be logged by OpenLog. There are multiple example sketches available in this repository. If you don't know how to use Github, download this entire repo to get the example Arduino sketches and then checkout this tutorial on how to use Github.

Example C Code: You will need to setup your microcontroller to output serial streams at 9600bps 8N1. Almost all microcontrollers now have a UART and are configurable for this setup.

How do I attach a FTDI Basic board to OpenLog for configuring and bootloading? You *can not*attach an FTDI Basic or FTDI Cable directly. This is because you have to swap TX and RX.

It's easiest to use a breadboard to make the connection from an FTDI to an OpenLog.

But once you've swapped TX/RX, you can easily use an FTDI Basic to talk to, configure, and quickly bootload new firmware onto OpenLog.

Why in the world did you do mess up TX and RX like that? Most of the data logging projects (such as logging the temperature of your compost pile over 3 months) take place away from a computer. Therefore, OpenLog will probably not be connected to a computer - instead, it will likely be connected to a microcontroller. We made OpenLog so that it can plug directly onto an Arduino Mini Pro

LAMPIRAN B

ATMEGA16

Datasheet Atmega 16

Features

- High Performance, Low Power Atmel®AVR® 8-bit Microcontroller
- Advanced RISC Architecture
  - 131 Powerful Instructions - Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 1 MIPS throughput per MHz
  - On-chip 2-cycle Multiplier
- Data and Non-Volatile Program Memory
  - 16/32/64K Bytes Flash of In-System Programmable Program Memory
  - 512B/1K/2K Bytes of In-System Programmable EEPROM
  - 1/2/4K Bytes Internal SRAM
  - Write/Erase Cycles: 10,000 Flash/ 100,000 EEPROM
  - Data Retention: 20 years at 85°C/ 100 years at 25°C(1)
  - Optional Boot Code Section with Independent Lock Bits
    - In-System Programming by On-chip Boot Program
    - True Read-While-Write Operation
  - Programming Lock for Flash Program and EEPROM Data Security
- On Chip Debug Interface (debugWIRE)
- CAN 2.0A/B with 6 Message Objects - ISO 16845 Certified
- LIN 2.1 and 1.3 Controller or 8-Bit UART
- One 12-bit High Speed PSC (Power Stage Controller)
  - Non Overlapping Inverted PWM Output Pins With Flexible Dead-Time
  - Variable PWM duty Cycle and Frequency
  - Synchronous Update of all PWM Registers
  - Auto Stop Function for Emergency Event
- Peripheral Features
  - One 8-bit General purpose Timer/Counter with Separate Prescaler, Compare Mode and Capture Mode
  - One 16-bit General purpose Timer/Counter with Separate Prescaler, Compare Mode and Capture Mode
  - One Master/Slave SPI Serial Interface
  - 10-bit ADC
    - Up To 11 Single Ended Channels and 3 Fully Differential ADC Channel Pairs
    - Programmable Gain (5x, 10x, 20x, 40x) on Differential Channels
    - Internal Reference Voltage
    - Direct Power Supply Voltage Measurement
  - 10-bit DAC for Variable Voltage Reference (Comparators, ADC)
  - Four Analog Comparators with Variable Threshold Detection
  - 100µA ±2% Current Source (LIN Node Identification)
  - Interrupt and Wake-up on Pin Change
  - Programmable Watchdog Timer with Separate On-Chip Oscillator
  - On-chipTemperature Sensor
- Special Microcontroller Features
  - Low Power Idle, Noise Reduction, and Power Down Modes
  - Power On Reset and Programmable Brown Out Detection
  - In-System Programmable via SPI Port
  - High Precision Crystal Oscillator for CAN Operations (16MHz)
  - Internal Calibrated RC Oscillator (8MHz)
  - On-chip PLL for fast PWM (32MHz, 64MHz) and CPU (16MHz)
- Operating Voltage: 2.7V - 5.5V
- Extended Operating Temperature:
  - -40°C to +85°C
- Core Speed Grade:
  - 0 - 8MHz @ 2.7 - 4.5V
  - 0 - 16MHz @ 4.5 - 5.5V

8-bit AVR® Microcontroller with 16/32/64K Bytes In-System Programmable Flash

ATmega16M1
ATmega32M1
ATmega64M1

Preliminary Summary

8209DS–AVR–11/10

1.1 Pin Descriptions

Table 1-1. Pinout description

QFN32 Pin Number	Mnemonic	Type	Name, Function & Alternate Function
5	GND	Power	**Ground:** 0V reference
20	AGND	Power	**Analog Ground:** 0V reference for analog part
4	VCC	Power	**Power Supply**
19	AVCC	Power	**Analog Power Supply:** This is the power supply voltage for analog part For a normal use this pin must be connected
21	AREF	Power	**Analog Reference:** reference for analog converter . This is the reference voltage of the A/D converter. As output, can be used by external analog ISRC (Current Source Output)
8	PB0	I/O	MISO (SPI Master In Slave Out) PSCOUT2A[1] (PSC Module 2 Output A) PCINT0 (Pin Change Interrupt 0)
9	PB1	I/O	MOSI (SPI Master Out Slave In) PSCOUT2B[1] (PSC Module 2 Output B) PCINT1 (Pin Change Interrupt 1)
16	PB2	I/O	ADC5 (Analog Input Channel 5) INT1 (External Interrupt 1 Input) ACMPN0 (Analog Comparator 0 Negative Input) PCINT2 (Pin Change Interrupt 2)
23	PB3	I/O	AMP0- (Analog Differential Amplifier 0 Negative Input) PCINT3 (Pin Change Interrupt 3)
24	PB4	I/O	AMP0+ (Analog Differential Amplifier 0 Positive Input) PCINT4 (Pin Change Interrupt 4)
26	PB5	I/O	ADC6 (Analog Input Channel 6) INT2 (External Interrupt 2 Input) ACMPN1 (Analog Comparator 1 Negative Input) AMP2- (Analog Differential Amplifier 2 Negative Input) PCINT5 (Pin Change Interrupt 5)
27	PB6	I/O	ADC7 (Analog Input Channel 7) PSCOUT1B[1] (PSC Module 1 Output A) PCINT6 (Pin Change Interrupt 6)
28	PB7	I/O	ADC4 (Analog Input Channel 4) PSCOUT0B[1] (PSC Module 0 Output B) SCK (SPI Clock) PCINT7 (Pin Change Interrupt 7)
30	PC0	I/O	PSCOUT1A[1] (PSC Module 1 Output A) INT3 (External Interrupt 3 Input) PCINT8 (Pin Change Interrupt 8)

Table 1-1. Pinout description (Continued)

QFN32 Pin Number	Mnemonic	Type	Name, Function & Alternate Function
3	PC1	I/O	PSCIN1 (PSC Digital Input 1) OC1B (Timer 1 Output Compare B) SS_A (Alternate SPI Slave Select) PCINT9 (Pin Change Interrupt 9)
6	PC2	I/O	T0 (Timer 0 clock input) TXCAN (CAN Transmit Output) PCINT10 (Pin Change Interrupt 10)
7	PC3	I/O	T1 (Timer 1 clock input) RXCAN (CAN Receive Input) ICP1B (Timer 1 input capture alternate B input) PCINT11 (Pin Change Interrupt 11)
17	PC4	I/O	ADC8 (Analog Input Channel 8) AMP1- (Analog Differential Amplifier 1 Negative Input) ACMPN3 (Analog Comparator 3 Negative Input) PCINT12 (Pin Change Interrupt 12)
18	PC5	I/O	ADC9 (Analog Input Channel 9) AMP1+ (Analog Differential Amplifier 1 Positive Input) ACMP3 (Analog Comparator 3 Positive Input) PCINT13 (Pin Change Interrupt 13)
22	PC6	I/O	ADC10 (Analog Input Channel 10) ACMP1 (Analog Comparator 1 Positive Input) PCINT14 (Pin Change Interrupt 14)
25	PC7	I/O	D2A (DAC output) AMP2+ (Analog Differential Amplifier 2 Positive Input) PCINT15 (Pin Change Interrupt 15)
29	PD0	I/O	PSCOUT0A[1] (PSC Module 0 Output A) PCINT16 (Pin Change Interrupt 16)
32	PD1	I/O	PSCIN0 (PSC Digital Input 0) CLKO (System Clock Output) PCINT17 (Pin Change Interrupt 17)
1	PD2	I/O	OC1A (Timer 1 Output Compare A) PSCIN2 (PSC Digital Input 2) MISO_A (Programming & alternate SPI Master In Slave Out) PCINT18 (Pin Change Interrupt 18)
2	PD3	I/O	TXD (UART Tx data) TXLIN (LIN Transmit Output) OC0A (Timer 0 Output Compare A) SS (SPI Slave Select) MOSI_A (Programming & alternate Master Out SPI Slave In) PCINT19 (Pin Change Interrupt 19)

Table 1-1. Pinout description (Continued)

QFN32 Pin Number	Mnemonic	Type	Name, Function & Alternate Function
12	PD4	I/O	ADC1 (Analog Input Channel 1) RXD (UART Rx data) RXLIN (LIN Receive Input) ICP1A (Timer 1 input capture alternate A input) SCK_A (Programming & alternate SPI Clock) PCINT20 (Pin Change Interrupt 20)
13	PD5	I/O	ADC2 (Analog Input Channel 2) ACMP2 (Analog Comparator 2 Positive Input) PCINT21 (Pin Change Interrupt 21)
14	PD6	I/O	ADC3 (Analog Input Channel 3) ACMPN2 (Analog Comparator 2 Negative Input) INT0 (External Interrupt 0 Input) PCINT22 (Pin Change Interrupt 22)
15	PD7	I/O	ACMP0 (Analog Comparator 0 Positive Input) PCINT23 (Pin Change Interrupt 23)
31	PE0	I/O or I	RESET (Reset Input) OCD (On Chip Debug I/O) PCINT24 (Pin Change Interrupt 24)
10	PE1	I/O	XTAL1 (XTAL Input) OC0B (Timer 0 Output Compare B) PCINT25 (Pin Change Interrupt 25)
11	PE2	I/O	XTAL2 (XTAL Output) ADC0 (Analog Input Channel 0) PCINT26 (Pin Change Interrupt 26)

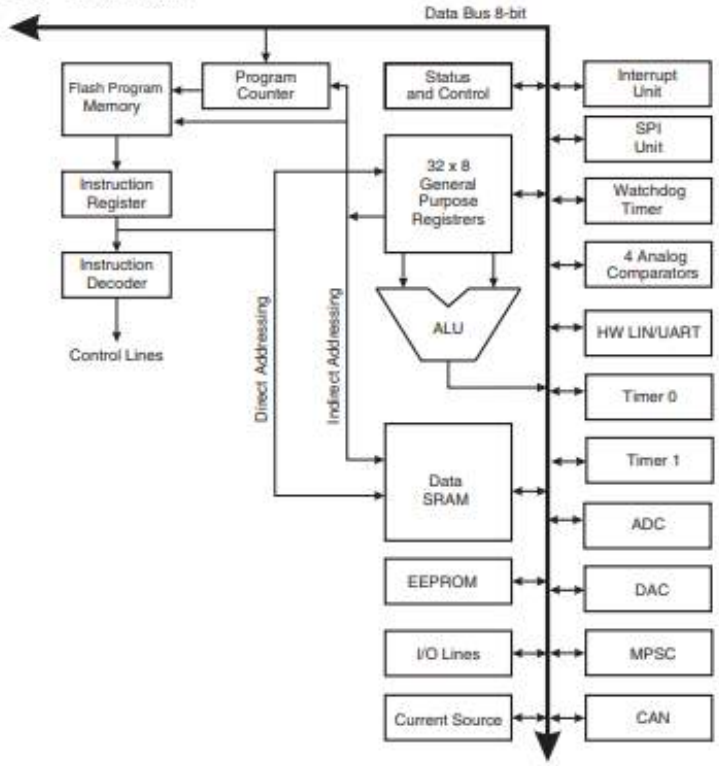
Note: 1. Only for Atmega32M1/64M1
2. On the engineering samples, the ACMPN3 alternate function is not located on PC4. It is located on PE2

2. Overview

The ATmega16M1/32M1/64M1 is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega16M1/32M1/64M1 achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

2.1 Block Diagram

Figure 2-1. Block Diagram



The AVR core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

The ATmega16M1/32M1/64M1 provides the following features: 16/32/64K bytes of In-System Programmable Flash with Read-While-Write capabilities, 512B/1K/2K bytes EEPROM, 1/2/4K bytes SRAM, 27 general purpose I/O lines, 32 general purpose working registers, one Motor Power Stage Controller, two flexible Timer/Counters with compare modes and PWM, one UART with HW LIN, an 11-channel 10-bit ADC with two differential input stages with programmable gain, a 10-bit DAC, a programmable Watchdog Timer with Internal Individual Oscillator, an SPI serial port, an On-chip Debug system and four software selectable power saving modes.

The Idle mode stops the CPU while allowing the SRAM, Timer/Counters, SPI ports, CAN, LIN/UART and interrupt system to continue functioning. The Power-down mode saves the register contents but freezes the Oscillator, disabling all other chip functions until the next interrupt or Hardware Reset. The ADC Noise Reduction mode stops the CPU and all I/O modules except ADC, to minimize switching noise during ADC conversions. In Standby mode, the Crystal/Resonator Oscillator is running while the rest of the device is sleeping. This allows very fast start-up combined with low power consumption.

The device is manufactured using Atmel's high-density nonvolatile memory technology. The On-chip ISP Flash allows the program memory to be reprogrammed in-system through an SPI serial interface, by a conventional nonvolatile memory programmer, or by an On-chip Boot program running on the AVR core. The boot program can use any interface to download the application program in the application Flash memory. Software in the Boot Flash section will continue to run while the Application Flash section is updated, providing true Read-While-Write operation. By combining an 8-bit RISC CPU with In-System Self-Programmable Flash on a monolithic chip, the Atmel ATmega16M1/32M1/64M1 is a powerful microcontroller that provides a highly flexible and cost effective solution to many embedded control applications.

The ATmega16M1/32M1/64M1 AVR is supported with a full suite of program and system development tools including: C compilers, macro assemblers, program debugger/simulators, in-circuit emulators, and evaluation kits.

2.2 Pin Descriptions

2.2.1 VCC

Digital supply voltage.

2.2.2 GND

Ground.

2.2.3 Port B (PB7..PB0)

Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port B also serves the functions of various special features of the ATmega16M1/32M1/64M1 as listed on page 70.

2.2.4 Port C (PC7..PC0)

Port C is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port C output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated. The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port C also serves the functions of special features of the ATmega16M1/32M1/64M1 as listed on page 74.

2.2.5 Port D (PD7..PD0)

Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port D also serves the functions of various special features of the ATmega16M1/32M1/64M1 as listed on page 78.

2.2.6 Port E (PE2..0) $\overline{\text{RESET}}$/ XTAL1/XTAL2

Port E is an 3-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port E output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port E pins that are externally pulled low will source current if the pull-up resistors are activated. The Port E pins are tri-stated when a reset condition becomes active, even if the clock is not running.

If the RSTDISBL Fuse is programmed, PE0 is used as an I/O pin. Note that the electrical characteristics of PE0 differ from those of the other pins of Port E.

If the RSTDISBL Fuse is unprogrammed, PE0 is used as a Reset input. A low level on this pin for longer than the minimum pulse length will generate a Reset, even if the clock is not running. The minimum pulse length is given in "System and Reset Characteristics" on page 311. Shorter pulses are not guaranteed to generate a Reset.

Depending on the clock selection fuse settings, PE1 can be used as input to the inverting Oscillator amplifier and input to the internal clock operating circuit.

Depending on the clock selection fuse settings, PE2 can be used as output from the inverting Oscillator amplifier.

The various special features of Port E are elaborated in "Alternate Functions of Port E" on page 81 and "Clock Systems and their Distribution" on page 27.

2.2.7 AVCC

AVCC is the supply voltage pin for the A/D Converter, D/A Converter, Current source. It should be externally connected to V_{CC}, even if the ADC, DAC are not used. If the ADC is used, it should be connected to V_{CC} through a low-pass filter.

2.2.8 AREF

This is the analog reference pin for the A/D Converter.

3. Resources

A comprehensive set of development tools, application notes and datasheets are available for download on http://www.atmel.com/avr.

4. About Code Examples

This documentation contains simple code examples that briefly show how to use various parts of the device. Be aware that not all C compiler vendors include bit definitions in the header files and interrupt handling in C is compiler dependent. Please confirm with the C compiler documentation for more details.

These code examples assume that the part specific header file is included before compilation. For I/O registers located in extended I/O map, "IN", "OUT", "SBIS", "SBIC", "CBI", and "SBI" instructions must be replaced with instructions that allow access to extended I/O. Typically "LDS" and "STS" combined with "SBRS", "SBRC", "SBR", and "CBR".

5. Data Retention

Reliability Qualification results show that the projected data retention failure rate is much less than 1 PPM over 20 years at 85°C or 100 years at 25°C.

6. Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0xFF)	Reserved	–	–	–	–	–	–	–	–	
(0xFE)	Reserved	–	–	–	–	–	–	–	–	
(0xFD)	Reserved	–	–	–	–	–	–	–	–	
(0xFC)	Reserved	–	–	–	–	–	–	–	–	
(0xFB)	Reserved	–	–	–	–	–	–	–	–	
(0xFA)	CANMSG	MSG 7	MSG 6	MSG 5	MSG 4	MSG 3	MSG 2	MSG 1	MSG 0	195
(0xF9)	CANSTMPH	TIMSTM15	TIMSTM14	TIMSTM13	TIMSTM12	TIMSTM11	TIMSTM10	TIMSTM9	TIMSTM8	195
(0xF8)	CANSTMPL	TIMSTM7	TIMSTM6	TIMSTM5	TIMSTM4	TIMSTM3	TIMSTM2	TIMSTM1	TIMSTM0	195
(0xF7)	CANIDM1	IDMSK28	IDMSK27	IDMSK26	IDMSK25	IDMSK24	IDMSK23	IDMSK22	IDMSK21	194
(0xF6)	CANIDM2	IDMSK20	IDMSK19	IDMSK18	IDMSK17	IDMSK16	IDMSK15	IDMSK14	IDMSK13	194
(0xF5)	CANIDM3	IDMSK12	IDMSK11	IDMSK10	IDMSK9	IDMSK8	IDMSK7	IDMSK6	IDMSK5	194
(0xF4)	CANIDM4	IDMSK4	IDMSK3	IDMSK2	IDMSK1	IDMSK0	RTRMSK	–	IDEMSK	194
(0xF3)	CANIDT1	IDT28	IDT27	IDT26	IDT25	IDT24	IDT23	IDT22	IDT21	192
(0xF2)	CANIDT2	IDT20	IDT19	IDT18	IDT17	IDT16	IDT15	IDT14	IDT13	192
(0xF1)	CANIDT3	IDT12	IDT11	IDT10	IDT9	IDT8	IDT7	IDT6	IDT5	192
(0xF0)	CANIDT4	IDT4	IDT3	IDT2	IDT1	IDT0	RTRTAG	RB1TAG	RB0TAG	192
(0xEF)	CANCDMOB	CONMOB1	CONMOB0	RPLV	IDE	DLC3	DLC2	DLC1	DLC0	191
(0xEE)	CANSTMOB	DLCW	TXOK	RXOK	BERR	SERR	CERR	FERR	AERR	190
(0xED)	CANPAGE	MOBNB3	MOBNB2	MOBNB1	MOBNB0	AINC	INDX2	INDX1	INDX0	190
(0xEC)	CANHPMOB	HPMOB3	HPMOB2	HPMOB1	HPMOB0	CGP3	CGP2	CGP1	CGP0	189
(0xEB)	CANREC	REC7	REC6	REC5	REC4	REC3	REC2	REC1	REC0	189
(0xEA)	CANTEC	TEC7	TEC6	TEC5	TEC4	TEC3	TEC2	TEC1	TEC0	189
(0xE9)	CANTTCH	TIMTTC15	TIMTTC14	TIMTTC13	TIMTTC12	TIMTTC11	TIMTTC10	TIMTTC9	TIMTTC8	189
(0xE8)	CANTTCL	TIMTTC7	TIMTTC6	TIMTTC5	TIMTTC4	TIMTTC3	TIMTTC2	TIMTTC1	TIMTTC0	189
(0xE7)	CANTIMH	CANTIM15	CANTIM14	CANTIM13	CANTIM12	CANTIM11	CANTIM10	CANTIM9	CANTIM8	189
(0xE6)	CANTIML	CANTIM7	CANTIM6	CANTIM5	CANTIM4	CANTIM3	CANTIM2	CANTIM1	CANTIM0	189
(0xE5)	CANTCON	TPRSC7	TPRSC6	TPRSC5	TPRSC4	TPRSC3	TPRSC2	TPRSC1	TPRSC0	188
(0xE4)	CANBT3	–	PHS22	PHS21	PHS20	PHS12	PHS11	PHS10	SMP	188
(0xE3)	CANBT2	–	SJW1	SJW0	–	PRS2	PRS1	PRS0	–	187
(0xE2)	CANBT1	–	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0	–	186
(0xE1)	CANSIT1	–	–	–	–	–	–	–	–	186
(0xE0)	CANSIT2	–	–	SIT5	SIT4	SIT3	SIT2	SIT1	SIT0	186
(0xDF)	CANIE1	–	–	–	–	–	–	–	–	186
(0xDE)	CANIE2	–	–	IEMOB5	IEMOB4	IEMOB3	IEMOB2	IEMOB1	IEMOB0	186
(0xDD)	CANEN1	–	–	–	–	–	–	–	–	185
(0xDC)	CANEN2	–	–	ENMOB5	ENMOB4	ENMOB3	ENMOB2	ENMOB1	ENMOB0	185
(0xDB)	CANGIE	ENIT	ENBOFF	ENRX	ENTX	ENERR	ENBX	ENERG	ENOVRT	184
(0xDA)	CANGIT	CANIT	BOFFIT	OVRTIM	BXOK	SERG	CERG	FERG	AERG	183
(0xD9)	CANGSTA	–	OVRG	–	TXBSY	RXBSY	ENFG	BOFF	ERRP	182
(0xD8)	CANGCON	ABRQ	OVRQ	TTC	SYNTTC	LISTEN	TEST	ENA/STB	SWRES	181
(0xD7)	Reserved	–	–	–	–	–	–	–	–	
(0xD6)	Reserved	–	–	–	–	–	–	–	–	
(0xD5)	Reserved	–	–	–	–	–	–	–	–	
(0xD4)	Reserved	–	–	–	–	–	–	–	–	
(0xD3)	Reserved	–	–	–	–	–	–	–	–	
(0xD2)	LINDAT	LDATA7	LDATA6	LDATA5	LDATA4	LDATA3	LDATA2	LDATA1	LDATA0	222
(0xD1)	LINSIL	–	–	–	–	/LAINC	LINDX2	LINDX1	LINDX0	222
(0xD0)	LINIDR	LP1	LP0	LID5 / LDL1	LID4 / LDL0	LID3	LID2	LID1	LID0	221
(0xCF)	LINDLR	LTXDL3	LTXDL2	LTXDL1	LTXDL0	LRXDL3	LRXDL2	LRXDL1	LRXDL0	221
(0xCE)	LINBRRH	–	–	–	–	LDIV11	LDIV10	LDIV9	LDIV8	220
(0xCD)	LINBRRL	LDIV7	LDIV6	LDIV5	LDIV4	LDIV3	LDIV2	LDIV1	LDIV0	220
(0xCC)	LINBTR	LDISR	–	LBT5	LBT4	LBT3	LBT2	LBT1	LBT0	220
(0xCB)	LINERR	LABORT	LTOERR	LOVERR	LFERR	LSERR	LPERR	LCERR	LBERR	219
(0xCA)	LINENIR	–	–	–	–	LENERR	LENIDOK	LENTXOK	LENRXOK	218
(0xC9)	LINSIR	LIDST2	LIDST1	LIDST0	LBUSY	LERR	LIDOK	LTXOK	LRXOK	217
(0xC8)	LINCR	LSWRES	LIN13	LCONF1	LCONF0	LENA	LCMD2	LCMD1	LCMD0	216
(0xC7)	Reserved	–	–	–	–	–	–	–	–	
(0xC6)	Reserved	–	–	–	–	–	–	–	–	
(0xC5)	Reserved	–	–	–	–	–	–	–	–	
(0xC4)	Reserved	–	–	–	–	–	–	–	–	
(0xC3)	Reserved	–	–	–	–	–	–	–	–	
(0xC2)	Reserved	–	–	–	–	–	–	–	–	
(0xC1)	Reserved	–	–	–	–	–	–	–	–	
(0xC0)	Reserved	–	–	–	–	–	–	–	–	
(0xBF)	Reserved	–	–	–	–	–	–	–	–	

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0x0E)	Reserved	–	–	–	–	–	–	–	–	
(0x0D)	Reserved	–	–	–	–	–	–	–	–	
(0x0C)[1]	PIFR	–	–	–	–	PEV2	PEV1	PEV0	PEOP	152
(0x0B)[1]	PIM	–	–	–	–	PEVE2	PEVE1	PEVE0	PEOPE	151
(0x0A)[1]	PMIC2	POVEN2	PISEL2	PELEV2	PFLTE2	PAOC2	PRFM22	PRFM21	PRFM20	150
(0x09)[1]	PMIC1	POVEN1	PISEL1	PELEV1	PFLTE1	PAOC1	PRFM12	PRFM11	PRFM10	150
(0x08)[1]	PMIC0	POVEN0	PISEL0	PELEV0	PFLTE0	PAOC0	PRFM02	PRFM01	PRFM00	150
(0x07)[1]	PCTL	PPRE1	PPRE0	PCLKSEL	–	–	–	PCCYC	PRUN	150
(0x06)[1]	POC	–	–	POEN2B	POEN2A	POEN1B	POEN1A	POEN0B	POEN0A	146
(0x05)[1]	PCNF	–	–	PULOCK	PMODE	POPB	POPA	–	–	149
(0x04)[1]	PSYNC	–	–	PSYNC21	PSYNC20	PSYNC11	PSYNC10	PSYNC01	PSYNC00	147
(0x03)[1]	POCR_RBH	–	–	–	–	POCR_RB11	POCR_RB10	POCR_RB9	POCR_RB8	149
(0x02)[1]	POCR_RBL	POCR_RB7	POCR_RB6	POCR_RB5	POCR_RB4	POCR_RB3	POCR_RB2	POCR_RB1	POCR_RB0	149
(0x01)[1]	POCR2SBH	–	–	–	–	POCR2SB11	POCR2SB10	POCR2SB9	POCR2SB8	148
(0x00)[1]	POCR2SBL	POCR2SB7	POCR2SB6	POCR2SB5	POCR2SB4	POCR2SB3	POCR2SB2	POCR2SB1	POCR2SB0	148
(0xAF)[1]	POCR2RAH	–	–	–	–	POCR2RA11	POCR2RA10	POCR2RA9	POCR2RA8	148
(0xAE)[1]	POCR2RAL	POCR2RA7	POCR2RA6	POCR2RA5	POCR2RA4	POCR2RA3	POCR2RA2	POCR2RA1	POCR2RA0	148
(0xAD)[1]	POCR2SAH	–	–	–	–	POCR2SA11	POCR2SA10	POCR2SA9	POCR2SA8	148
(0xAC)[1]	POCR2SAL	POCR2SA7	POCR2SA6	POCR2SA5	POCR2SA4	POCR2SA3	POCR2SA2	POCR2SA1	POCR2SA0	148
(0xAB)[1]	POCR1SBH	–	–	–	–	POCR1SB11	POCR1SB10	POCR1SB9	POCR1SB8	148
(0xAA)[1]	POCR1SBL	POCR1SB7	POCR1SB6	POCR1SB5	POCR1SB4	POCR1SB3	POCR1SB2	POCR1SB1	POCR1SB0	148
(0xA9)[1]	POCR1RAH	–	–	–	–	POCR1RA11	POCR1RA10	POCR1RA9	POCR1RA8	148
(0xA8)[1]	POCR1RAL	POCR1RA7	POCR1RA6	POCR1RA5	POCR1RA4	POCR1RA3	POCR1RA2	POCR1RA1	POCR1RA0	148
(0xA7)[1]	POCR1SAH	–	–	–	–	POCR1SA11	POCR1SA10	POCR1SA9	POCR1SA8	148
(0xA6)[1]	POCR1SAL	POCR1SA7	POCR1SA6	POCR1SA5	POCR1SA4	POCR1SA3	POCR1SA2	POCR1SA1	POCR1SA0	148
(0xA5)[1]	POCR0SBH	–	–	–	–	POCR0SB11	POCR0SB10	POCR0SB9	POCR0SB8	148
(0xA4)[1]	POCR0SBL	POCR0SB7	POCR0SB6	POCR0SB5	POCR0SB4	POCR0SB3	POCR0SB2	POCR0SB1	POCR0SB0	148
(0xA3)[1]	POCR0RAH	–	–	–	–	POCR0RA11	POCR0RA10	POCR0RA9	POCR0RA8	148
(0xA2)[1]	POCR0RAL	POCR0RA7	POCR0RA6	POCR0RA5	POCR0RA4	POCR0RA3	POCR0RA2	POCR0RA1	POCR0RA0	148
(0xA1)[1]	POCR0SAH	–	–	–	–	POCR0SA11	POCR0SA10	POCR0SA9	POCR0SA8	148
(0xA0)[1]	POCR0SAL	POCR0SA7	POCR0SA6	POCR0SA5	POCR0SA4	POCR0SA3	POCR0SA2	POCR0SA1	POCR0SA0	148
(0x9F)	Reserved	–	–	–	–	–	–	–	–	
(0x9E)	Reserved	–	–	–	–	–	–	–	–	
(0x9D)	Reserved	–	–	–	–	–	–	–	–	
(0x9C)	Reserved	–	–	–	–	–	–	–	–	
(0x9B)	Reserved	–	–	–	–	–	–	–	–	
(0x9A)	Reserved	–	–	–	–	–	–	–	–	
(0x99)	Reserved	–	–	–	–	–	–	–	–	
(0x98)	Reserved	–	–	–	–	–	–	–	–	
(0x97)	AC3CON	AC3EN	AC3IE	AC3IS1	AC3IS0	–	AC3M2	AC3M1	AC3M0	258
(0x96)	AC2CON	AC2EN	AC2IE	AC2IS1	AC2IS0	–	AC2M2	AC2M1	AC2M0	258
(0x95)	AC1CON	AC1EN	AC1IE	AC1IS1	AC1IS0	AC1ICE	AC1M2	AC1M1	AC1M0	257
(0x94)	AC0CON	AC0EN	AC0IE	AC0IS1	AC0IS0	ACCKSEL	AC0M2	AC0M1	AC0M0	256
(0x93)	Reserved	–	–	–	–	–	–	–	–	
(0x92)	DACH	- / DAC9	- / DAC8	- / DAC7	- / DAC6	- / DAC5	- / DAC4	DAC9 / DAC3	DAC8 / DAC2	266
(0x91)	DACL	DAC7 / DAC1	DAC6 /DAC0	DAC5 / -	DAC4 / -	DAC3 / -	DAC2 / -	DAC1 / -	DAC0 /	266
(0x90)	DACON	DAATE	DATS2	DATS1	DATS0	–	DALA	DAOE	DAEN	265
(0x8F)	Reserved	–	–	–	–	–	–	–	–	
(0x8E)	Reserved	–	–	–	–	–	–	–	–	
(0x8D)	Reserved	–	–	–	–	–	–	–	–	
(0x8C)	Reserved	–	–	–	–	–	–	–	–	
(0x8B)	OCR1BH	OCR1B15	OCR1B14	OCR1B13	OCR1B12	OCR1B11	OCR1B10	OCR1B9	OCR1B8	127
(0x8A)	OCR1BL	OCR1B7	OCR1B6	OCR1B5	OCR1B4	OCR1B3	OCR1B2	OCR1B1	OCR1B0	127
(0x89)	OCR1AH	OCR1A15	OCR1A14	OCR1A13	OCR1A12	OCR1A11	OCR1A10	OCR1A9	OCR1A8	127
(0x88)	OCR1AL	OCR1A7	OCR1A6	OCR1A5	OCR1A4	OCR1A3	OCR1A2	OCR1A1	OCR1A0	127
(0x87)	ICR1H	ICR115	ICR114	ICR113	ICR112	ICR111	ICR110	ICR19	ICR18	128
(0x86)	ICR1L	ICR17	ICR16	ICR15	ICR14	ICR13	ICR12	ICR11	ICR10	128
(0x85)	TCNT1H	TCNT115	TCNT114	TCNT113	TCNT112	TCNT111	TCNT110	TCNT19	TCNT18	127
(0x84)	TCNT1L	TCNT17	TCNT16	TCNT15	TCNT14	TCNT13	TCNT12	TCNT11	TCNT10	127
(0x83)	Reserved	–	–	–	–	–	–	–	–	
(0x82)	TCCR1C	FOC1A	FOC1B	–	–	–	–	–	–	126
(0x81)	TCCR1B	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	125
(0x80)	TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	–	–	WGM11	WGM10	123
(0x7F)	DIDR1	–	AMP2PD	ACMP0D	AMP0PD	AMP0ND	ADC10D	ADC9D	ADC8D	246
(0x7E)	DIDR0	ADC7D	ADC6D	ADC5D	ADC4D	ADC3D	ADC2D	ADC1D	ADC0D	246
(0x7D)	Reserved	–	–	–	–	–	–	–	–	

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0x7C)	**ADMUX**	REFS1	REFS0	ADLAR	–	MUX3	MUX2	MUX1	MUX0	242
(0x7B)	**ADCSRB**	ADHSM	ISRCEN	AREFEN	–	ADTS3	ADTS2	ADTS1	ADTS0	244
(0x7A)	**ADCSRA**	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	243
(0x79)	**ADCH**	- / ADC9	- / ADC8	- / ADC7	- / ADC6	- / ADC5	- / ADC4	ADC9 / ADC3	ADC8 / ADC2	245
(0x78)	**ADCL**	ADC7 / ADC1	ADC6 / ADC0	ADC5 / -	ADC4 / -	ADC3 / -	ADC2 / -	ADC1 / -	ADC0 /	245
(0x77)	**AMP2CSR**	AMP2EN	AMP2IS	AMP2G1	AMP2G0	AMP2MP2	AMP2TS2	AMP2TS1	AMP2TS0	248
(0x76)	**AMP1CSR**	AMP1EN	AMP1IS	AMP1G1	AMP1G0	AMP1MP1	AMP1TS2	AMP1TS1	AMP1TS0	248
(0x75)	**AMP0CSR**	AMP0EN	AMP0IS	AMP0G1	AMP0G0	AMP0MP0	AMP0TS2	AMP0TS1	AMP0TS0	247
(0x74)	**Reserved**	–	–	–	–	–	–	–	–	
(0x73)	**Reserved**	–	–	–	–	–	–	–	–	
(0x72)	**Reserved**	–	–	–	–	–	–	–	–	
(0x71)	**Reserved**	–	–	–	–	–	–	–	–	
(0x70)	**Reserved**	–	–	–	–	–	–	–	–	
(0x6F)	**TIMSK1**	–	–	ICIE1	–	–	OCIE1B	OCIE1A	TOIE1	128
(0x6E)	**TIMSK0**	–	–	–	–	–	OCIE0B	OCIE0A	TOIE0	100
(0x6D)	**PCMSK3**	–	–	–	–	–	PCINT26	PCINT25	PCINT24	62
(0x6C)	**PCMSK2**	PCINT23	PCINT22	PCINT21	PCINT20	PCINT19	PCINT18	PCINT17	PCINT16	63
(0x6B)	**PCMSK1**	PCINT15	PCINT14	PCINT13	PCINT12	PCINT11	PCINT10	PCINT9	PCINT8	63
(0x6A)	**PCMSK0**	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0	63
(0x69)	**EICRA**	ISC31	ISC30	ISC21	ISC20	ISC11	ISC10	ISC01	ISC00	60
(0x68)	**PCICR**	–	–	–	–	PCIE3	PCIE2	PCIE1	PCIE0	61
(0x68)	**Reserved**	–	–	–	–	–	–	–	–	
(0x67)	**Reserved**	–	–	–	–	–	–	–	–	
(0x66)	**OSCCAL**	–	CAL6	CAL5	CAL4	CAL3	CAL2	CAL1	CAL0	34
(0x65)	**Reserved**	–	–	–	–	–	–	–	–	
(0x64)	**PRR**	–	PRCAN	PRPSC	PRTIM1	PRTIM0	PRSPI	PRLIN	PRADC	41
(0x63)	**Reserved**	–	–	–	–	–	–	–	–	
(0x62)	**Reserved**	–	–	–	–	–	–	–	–	
(0x61)	**CLKPR**	CLKPCE	–	–	–	CLKPS3	CLKPS2	CLKPS1	CLKPS0	35
(0x60)	**WDTCSR**	WDIF	WDIE	WDP3	WDCE	WDE	WDP2	WDP1	WDP0	50
0x3F (0x5F)	**SREG**	I	T	H	S	V	N	Z	C	11
0x3E (0x5E)	**SPH**	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	14
0x3D (0x5D)	**SPL**	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	14
0x3C (0x5C)	**Reserved**	–	–	–	–	–	–	–	–	
0x3B (0x5B)	**Reserved**	–	–	–	–	–	–	–	–	
0x3A (0x5A)	**Reserved**	–	–	–	–	–	–	–	–	
0x39 (0x59)	**Reserved**	–	–	–	–	–	–	–	–	
0x38 (0x58)	**Reserved**	–	–	–	–	–	–	–	–	
0x37 (0x57)	**SPMCSR**	SPMIE	RWWSB	–	RWWSRE	BLBSET	PGWRT	PGERS	SPMEN	275
0x36 (0x56)	**Reserved**	–	–	–	–	–	–	–	–	
0x35 (0x55)	**MCUCR**	SPIPS	–	–	PUD	–	–	IVSEL	IVCE	57 & 63
0x34 (0x54)	**MCUSR**	–	–	–	–	WDRF	BORF	EXTRF	PORF	50
0x33 (0x53)	**SMCR**	–	–	–	–	SM2	SM1	SM0	SE	37
0x32 (0x52)	**MSMCR**	Monitor Stop Mode Control Register								reserved
0x31 (0x51)	**MONDR**	Monitor Data Register								reserved
0x30 (0x50)	**ACSR**	AC3IF	AC2IF	AC1IF	AC0IF	AC3O	AC2O	AC1O	AC0O	260
0x2F (0x4F)	**Reserved**	–	–	–	–	–	–	–	–	
0x2E (0x4E)	**SPDR**	SPD7	SPD6	SPD5	SPD4	SPD3	SPD2	SPD1	SPD0	162
0x2D (0x4D)	**SPSR**	SPIF	WCOL	–	–	–	–	–	SPI2X	161
0x2C (0x4C)	**SPCR**	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	160
0x2B (0x4B)	**Reserved**	–	–	–	–	–	–	–	–	
0x2A (0x4A)	**Reserved**	–	–	–	–	–	–	–	–	
0x29 (0x49)	**PLLCSR**	-	-	-	-	-	PLLF	PLLE	PLOCK	35
0x28 (0x48)	**OCR0B**	OCR0B7	OCR0B6	OCR0B5	OCR0B4	OCR0B3	OCR0B2	OCR0B1	OCR0B0	100
0x27 (0x47)	**OCR0A**	OCR0A7	OCR0A6	OCR0A5	OCR0A4	OCR0A3	OCR0A2	OCR0A1	OCR0A0	100
0x26 (0x46)	**TCNT0**	TCNT07	TCNT06	TCNT05	TCNT04	TCNT03	TCNT02	TCNT01	TCNT00	100
0x25 (0x45)	**TCCR0B**	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00	99
0x24 (0x44)	**TCCR0A**	COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00	96
0x23 (0x43)	**GTCCR**	TSM	ICPSEL1	–	–	–	–	–	PSRSYNC	132
0x22 (0x42)	**EEARH**	–	–	–	–	–	–	EEAR9	EEAR8	22
0x21 (0x41)	**EEARL**	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	22
0x20 (0x40)	**EEDR**	EEDR7	EEDR6	EEDR5	EEDR4	EEDR3	EEDR2	EEDR1	EEDR0	22
0x1F (0x3F)	**EECR**	–	–	–	–	EERIE	EEMWE	EEWE	EERE	22
0x1E (0x3E)	**GPIOR0**	GPIOR07	GPIOR06	GPIOR05	GPIOR04	GPIOR03	GPIOR02	GPIOR01	GPIOR00	26
0x1D (0x3D)	**EIMSK**	–	–	–	–	INT3	INT2	INT1	INT0	60
0x1C (0x3C)	**EIFR**	–	–	–	–	INTF3	INTF2	INTF1	INTF0	61

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
0x1B (0x3B)	**PCIFR**	–	–	–	–	PCIF3	PCIF2	PCIF1	PCIF0	62
0x1A (0x3A)	**GPIOR2**	GPIOR27	GPIOR26	GPIOR25	GPIOR24	GPIOR23	GPIOR22	GPIOR21	GPIOR20	26
0x19 (0x39)	**GPIOR1**	GPIOR17	GPIOR16	GPIOR15	GPIOR14	GPIOR13	GPIOR12	GPIOR11	GPIOR10	26
0x18 (0x38)	**Reserved**	–	–	–	–	–	–	–	–	
0x17 (0x37)	**Reserved**	–	–	–	–	–	–	–	–	
0x16 (0x36)	**TIFR1**	–	–	ICF1	–	–	OCF1B	OCF1A	TOV1	129
0x15 (0x35)	**TIFR0**	–	–	–	–	–	OCF0B	OCF0A	TOV0	101
0x14 (0x34)	**Reserved**	–	–	–	–	–	–	–	–	
0x13 (0x33)	**Reserved**	–	–	–	–	–	–	–	–	
0x12 (0x32)	**Reserved**	–	–	–	–	–	–	–	–	
0x11 (0x31)	**Reserved**	–	–	–	–	–	–	–	–	
0x10 (0x30)	**Reserved**	–	–	–	–	–	–	–	–	
0x0F (0x2F)	**Reserved**	–	–	–	–	–	–	–	–	
0x0E (0x2E)	**PORTE**	–	–	–	–	–	PORTE2	PORTE1	PORTE0	84
0x0D (0x2D)	**DDRE**	–	–	–	–	–	DDE2	DDE1	DDE0	84
0x0C (0x2C)	**PINE**	–	–	–	–	–	PINE2	PINE1	PINE0	84
0x0B (0x2B)	**PORTD**	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	84
0x0A (0x2A)	**DDRD**	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	84
0x09 (0x29)	**PIND**	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	84
0x08 (0x28)	**PORTC**	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	83
0x07 (0x27)	**DDRC**	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	83
0x06 (0x26)	**PINC**	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	83
0x05 (0x25)	**PORTB**	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	83
0x04 (0x24)	**DDRB**	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	83
0x03 (0x23)	**PINB**	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	83
0x02 (0x22)	**Reserved**	–	–	–	–	–	–	–	–	
0x01 (0x21)	**Reserved**	–	–	–	–	–	–	–	–	
0x00 (0x20)	**Reserved**	–	–	–	–	–	–	–	–	

Note:
1. For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written
2. I/O Registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions
3. Some of the status flags are cleared by writing a logical one to them. Note that, unlike most other AVRs, the CBI and SBI instructions will only operate on the specified bit, and can therefore be used on registers containing such status flags. The CBI and SBI instructions work with registers 0x00 to 0x1F only
4. When using the I/O specific commands IN and OUT, the I/O addresses 0x00 - 0x3F must be used. When addressing I/O Registers as data space using LD and ST instructions, 0x20 must be added to these addresses. The ATmega16M1/32M1/64M1 is a complex microcontroller with more peripheral units than can be supported within the 64 location reserved in Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60 - 0xFF in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used
5. These registers are only available on ATmega32/64M1. For other products described in this datasheet, these locations are reserved

8. Ordering Information

8.1 ATmega16M1

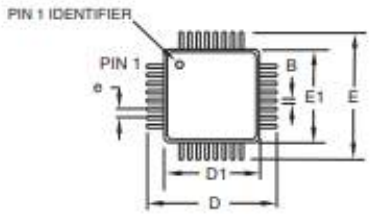
Speed	Power Supply	Ordering Code	Package	Operation Range
16MHz	2.7V - 5.5V	ATmega16M1 - AU	32A	Industrial (-40°C to 85°C)
		ATmega16M1 - MU	PV	

Note: All packages are Pb free, fully LHF

Package Type	
32A	32-lead, Thin (1.0mm) Plastic Quad Flat Package (TQFP)
PV	PV, 32-Lead, 7.0mm × 7.0mm Body, 0.65mm Pitch Quad Flat No Lead Package (QFN)

9. Packaging Information

9.1 32A



COMMON DIMENSIONS
(Unit of Measure = mm)

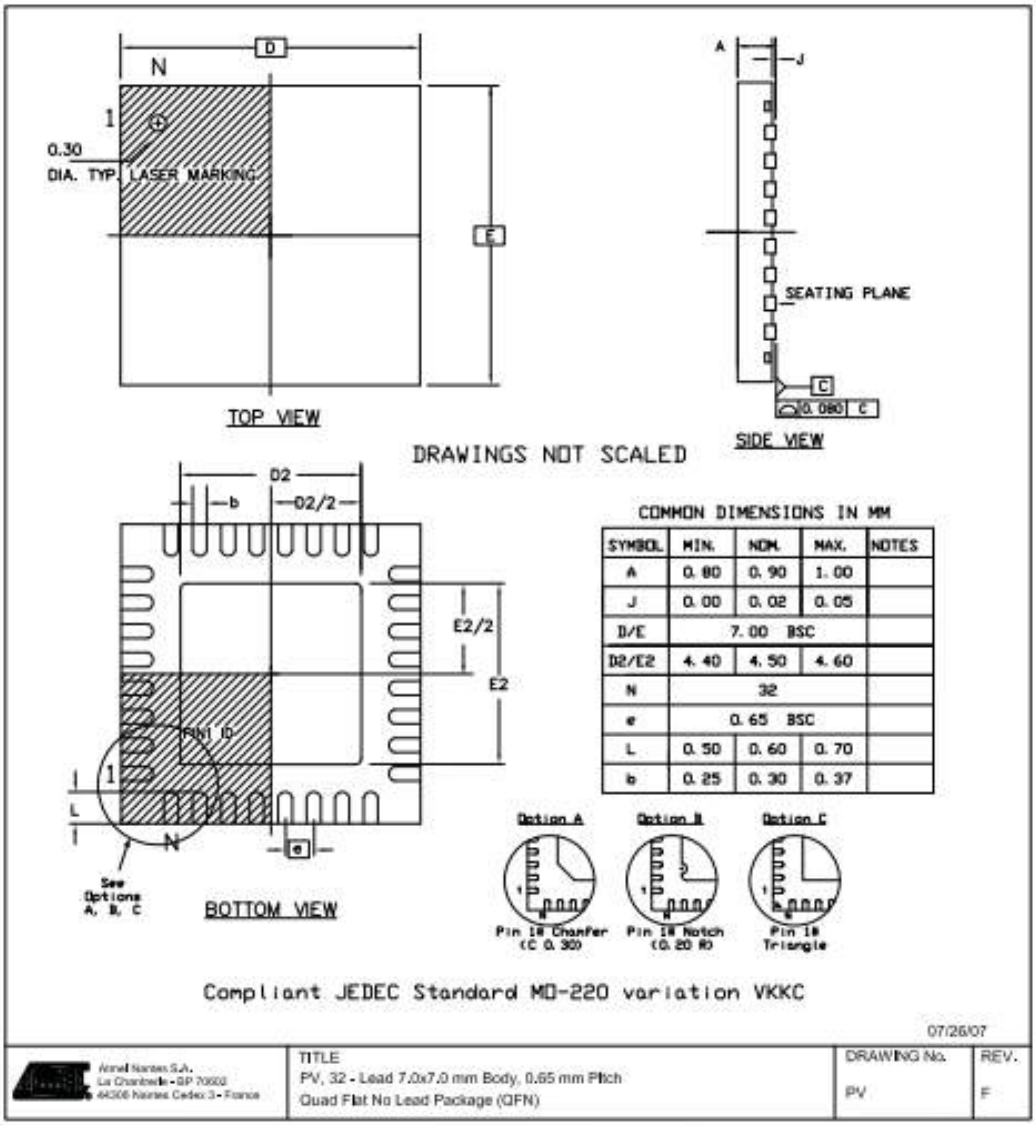
SYMBOL	MIN	NOM	MAX	NOTE
A	–	–	1.20	
A1	0.05	–	0.15	
A2	0.95	1.00	1.05	
D	8.75	9.00	9.25	
D1	6.90	7.00	7.10	Note 2
E	8.75	9.00	9.25	
E1	6.90	7.00	7.10	Note 2
B	0.30	–	0.45	
C	0.09	–	0.20	
L	0.45	–	0.75	
e	0.80 TYP			

Notes:
1. This package conforms to JEDEC reference MS-026, Variation ABA.
2. Dimensions D1 and E1 do not include mold protrusion. Allowable protrusion is 0.25 mm per side. Dimensions D1 and E1 are maximum plastic body size dimensions including mold mismatch.
3. Lead coplanarity is 0.10 mm maximum.

2010-10-20

	TITLE	DRAWING NO.	REV.
2325 Orchard Parkway San Jose, CA 95131	**32A**, 32-lead, 7 x 7 mm Body Size, 1.0 mm Body Thickness, 0.8 mm Lead Pitch, Thin Profile Plastic Quad Flat Package (TQFP)	32A	C

9.2 PV



SYMBOL	MIN.	NOM.	MAX.	NOTES
A	0.80	0.90	1.00	
J	0.00	0.02	0.05	
D/E		7.00 BSC		
D2/E2	4.40	4.50	4.60	
N		32		
e		0.65 BSC		
L	0.50	0.60	0.70	
b	0.25	0.30	0.37	

LAMPIRAN C

RTC (Real Time Clock)

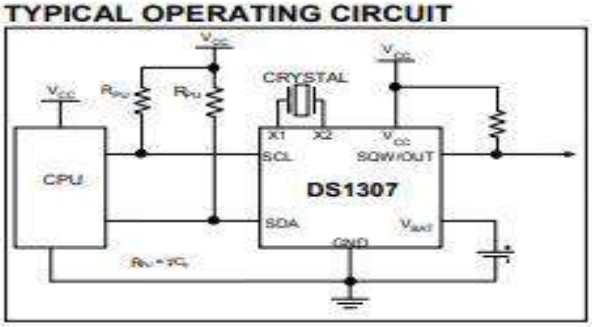
maxim integrated

DS1307
64 x 8, Serial, I²C Real-Time Clock

GENERAL DESCRIPTION

The DS1307 serial real-time clock (RTC) is a low-power, full binary-coded decimal (BCD) clock/calendar plus 56 bytes of NV SRAM. Address and data are transferred serially through an I²C, bidirectional bus. The clock/calendar provides seconds, minutes, hours, day, date, month, and year information. The end of the month date is automatically adjusted for months with fewer than 31 days, including corrections for leap year. The clock operates in either the 24-hour or 12-hour format with AM/PM indicator. The DS1307 has a built-in power-sense circuit that detects power failures and automatically switches to the backup supply. Timekeeping operation continues while the part operates from the backup supply.

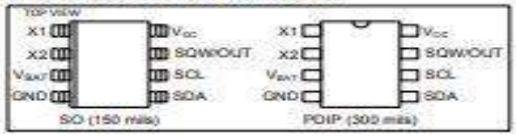
TYPICAL OPERATING CIRCUIT



BENEFITS AND FEATURES

- Completely Manages All Timekeeping Functions
  - Real-Time Clock Counts Seconds, Minutes, Hours, Date of the Month, Month, Day of the Week, and Year with Leap-Year Compensation Valid Up to 2100
  - 56-Byte, Battery-Backed, General-Purpose RAM with Unlimited Writes
  - Programmable Square-Wave Output Signal
- Simple Serial Port Interfaces to Most Microcontrollers
  - I²C Serial Interface
- Low Power Operation Extends Battery Backup Run Time
  - Consumes Less than 500nA in Battery-Backup Mode with Oscillator Running
  - Automatic Power-Fail Detect and Switch Circuitry
- 8-Pin DIP and 8-Pin SO Minimizes Required Space
- Optional Industrial Temperature Range: -40°C to +85°C Supports Operation in a Wide Range of Applications
- Underwriters Laboratories® (UL) Recognized

PIN CONFIGURATIONS



ORDERING INFORMATION

PART	TEMP RANGE	VOLTAGE (V)	PIN-PACKAGE	TOP MARK*
DS1307+	0°C to +70°C	5.0	8 PDIP (300 mils)	DS1307
DS1307N+	-40°C to +85°C	5.0	8 PDIP (300 mils)	DS1307N
DS1307Z+	0°C to +70°C	5.0	8 SO (150 mils)	DS1307
DS1307ZN+	-40°C to +85°C	5.0	8 SO (150 mils)	DS1307N
DS1307Z+T&R	0°C to +70°C	5.0	8 SO (150 mils) Tape and Reel	DS1307
DS1307ZN+T&R	-40°C to +85°C	5.0	8 SO (150 mils) Tape and Reel	DS1307N

+Denotes a lead-free/RoHS-compliant package.
"A "+" anywhere on the top mark indicates a lead-free package. An "N" anywhere on the top mark indicates an industrial temperature range device.
Underwriters Laboratories, Inc. is a registered certification mark of Underwriters Laboratories, Inc.

TIMING DIAGRAM

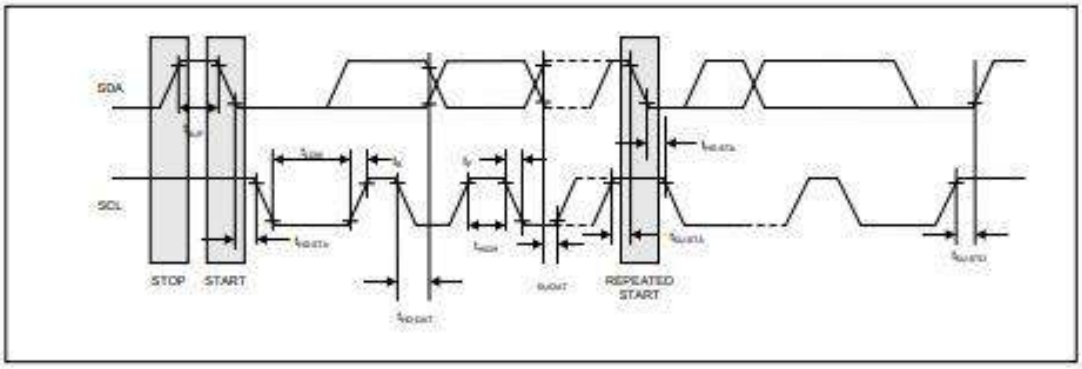
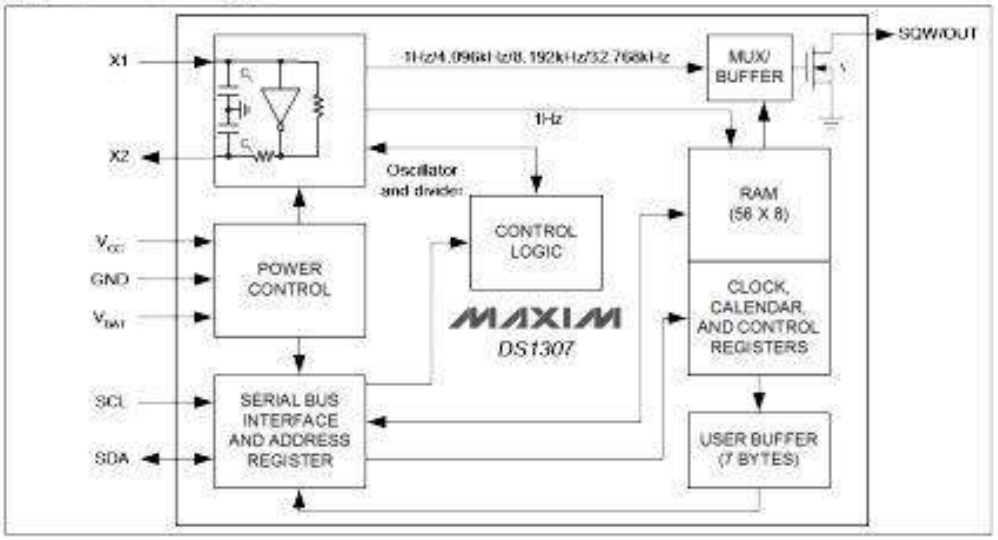


Figure 1. Block Diagram



LAMPIRAN D

FT232RL USB

FTDI Chip

Document No.: FT_000053
FT232R USB UART IC Datasheet Version 2.11

Future Technology Devices International Ltd.

FT232R USB UART IC

The FT232R is a USB to serial UART interface with the following advanced features:

- Single chip USB to asynchronous serial data transfer interface.
- Entire USB protocol handled on the chip. No USB specific firmware programming required.
- Fully integrated 1024 bit EEPROM storing device descriptors and CBUS I/O configuration.
- Fully integrated USB termination resistors.
- Fully integrated clock generation with no external crystal required plus optional clock output selection enabling a glue-less interface to external MCU or FPGA.
- Data transfer rates from 300 baud to 3 Mbaud (RS422, RS485, RS232) at TTL levels.
- 128 byte receive buffer and 256 byte transmit buffer utilising buffer smoothing technology to allow for high data throughput.
- FTDI's royalty-free Virtual Com Port (VCP) and Direct (D2XX) drivers eliminate the requirement for USB driver development in most cases.
- Unique USB FTDIChip-ID™ feature.
- Configurable CBUS I/O pins.
- Transmit and receive LED drive signals.
- UART interface support for 7 or 8 data bits, 1 or 2 stop bits and odd / even / mark / space / no parity
- FIFO receive and transmit buffers for high data throughput.
- Synchronous and asynchronous bit bang interface options with RD# and WR# strobes.
- Device supplied pre-programmed with unique USB serial number.
- Supports bus powered, self powered and high-power bus powered USB configurations.
- Integrated +3.3V level converter for USB I/O.
- Integrated level converter on UART and CBUS for interfacing to between +1.8V and +5V logic.
- True 5V/3.3V/2.8V/1.8V CMOS drive output and TTL input.
- Configurable I/O pin output drive strength.
- Integrated power-on-reset circuit.
- Fully integrated AVCC supply filtering - no external filtering required.
- UART signal inversion option.
- +3.3V (using external oscillator) to +5.25V (internal oscillator) Single Supply Operation.
- Low operating and USB suspend current.
- Low USB bandwidth consumption.
- UHCI/OHCI/EHCI host controller compatible.
- USB 2.0 Full Speed compatible.
- -40°C to 85°C extended operating temperature range.
- Available in compact Pb-free 28 Pin SSOP and QFN-32 packages (both RoHS compliant).

1 Typical Applications

- USB to RS232/RS422/RS485 Converters
- Upgrading Legacy Peripherals to USB
- Cellular and Cordless Phone USB data transfer cables and interfaces
- Interfacing MCU/PLD/FPGA based designs to USB
- USB Audio and Low Bandwidth Video data transfer
- PDA to USB data transfer
- USB Smart Card Readers
- USB Instrumentation
- USB Industrial Control
- USB MP3 Player Interface
- USB FLASH Card Reader and Writers
- Set Top Box PC - USB interface
- USB Digital Camera Interface
- USB Hardware Modems
- USB Wireless Modems
- USB Bar Code Readers
- USB Software and Hardware Encryption Dongles

1.1 Driver Support

Royalty free VIRTUAL COM PORT (VCP) DRIVERS for...

- Windows 98, 98SE, ME, 2000, Server 2003, XP and Server 2008
- Windows 7 32,64-bit
- Windows XP and XP 64-bit
- Windows Vista and Vista 64-bit
- Windows XP Embedded
- Windows CE 4.2, 5.0 and 6.0
- Mac OS 8/9, OS-X
- Linux 2.4 and greater

Royalty free D2XX *Direct* Drivers (USB Drivers + DLL S/W Interface)

- Windows 98, 98SE, ME, 2000, Server 2003, XP and Server 2008
- Windows 7 32,64-bit
- Windows XP and XP 64-bit
- Windows Vista and Vista 64-bit
- Windows XP Embedded
- Windows CE 4.2, 5.0 and 6.0
- Linux 2.4 and greater

The drivers listed above are all available to download for free from FTDI website (www.ftdichip.com). Various 3rd party drivers are also available for other operating systems - see FTDI website (www.ftdichip.com) for details.

For driver installation, please refer to http://www.ftdichip.com/Documents/InstallGuides.htm

1.2 Part Numbers

Part Number	Package
FT232RQ-xxxx	32 Pin QFN
FT232RL-xxxx	28 Pin SSOP

Note: Packing codes for xxxx is:

- Reel: Taped and Reel, (SSOP is 2,000pcs per reel, QFN is 6,000pcs per reel).
- Tube: Tube packing, 47pcs per tube (SSOP only)
- Tray: Tray packing, 490pcs per tray (QFN only)

2 FT232R Block Diagram

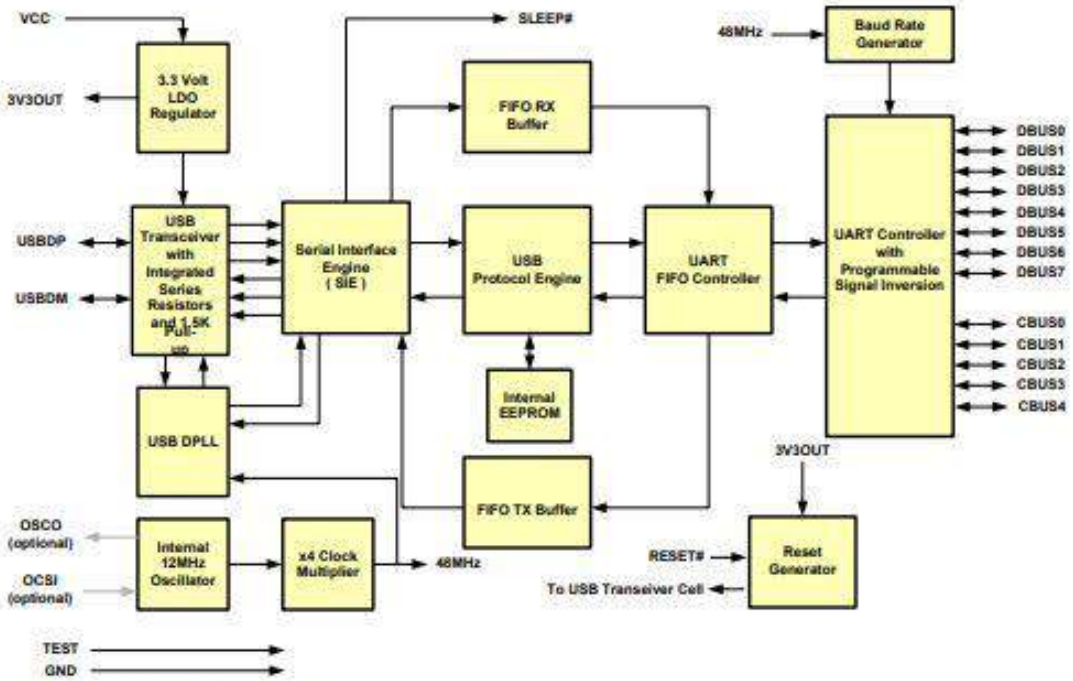


Figure 2.1 FT232R Block Diagram

For a description of each function please refer to Section 4.

LAMPIRAN E

```
/*****************************************
*****************
This program was created by the
CodeWizardAVR V3.12 Advanced
Automatic Program Generator
© Copyright 1998-2014 Pavel Haiduc, HP
InfoTech s.r.l.
http://www.hpinfotech.com

Project :
Version :
Date    : 13/07/2018
Author  :
Company :
Comments:
Chip type               : ATmega16
Program type            : Application
AVR Core Clock frequency: 8,000000 MHz
Memory model            : Small
External RAM size       : 0
Data Stack size         : 256
*****************************************
****************/

#include <mega16.h>
#include <delay.h>

// Declare your global variables here
char buff_tx[100];
char jam,menit, detik, tanggal, bulan, tahun,
hari, buff[17];
unsigned result;
unsigned long sekon;
unsigned char temp[16],temps[16];

#define maxstep 100
#define Kp 1.2
#define Kd 3
int direct,stepcount,respon;

unsigned result;
int
valve_open,derivative,awalerror,target_valve,de
ltaH;

//float Weight = 0;
//unsigned int overflow,motorservo;

int p;
unsigned long int kali;
int state = 0;
int kondisi = 0;
int step = 0;
int x = 0;
int y = 0;
int k = 0;
int l = 0;
int adc1;
float nilai1;

//unsigned int adc;
//float nilai;

#define timerx      40

// Premium
#define SW_Premium_Atas     PINB.1
#define SW_Premium_Bawah    PINB.0
#define SV_Premium_Atas     PORTD.5 //in3
relay
#define SV_Premium_Bawah    PORTD.4 //
in4 relay

// Solar
#define SW_Solar_Atas       PINB.3
#define SW_Solar_Bawah      PINB.2
#define SV_Solar_Atas       PORTD.7  // in1
relay
#define SV_Solar_Bawah      PORTD.6  //in2
relay
int cycle,perhitungan_liter;
float liter=0,liter;

// I2C Bus functions
#asm
   .equ __i2c_port=0x15 ;PORTC
   .equ __sda_bit=1
   .equ __scl_bit=0
#endasm
#include <i2c.h>

// DS1307 Real Time Clock functions
#include <ds1307.h>

// Alphanumeric LCD Module functions
#include <alcd.h>

// External Interrupt 0 service routine
interrupt [EXT_INT0] void ext_int0_isr(void)
{
// Place your code here
state = 0;
kondisi = 0;
step = 0;
```

```
}

// Standard Input/Output functions
#include <stdio.h>

// SPI functions
#include <spi.h>

// Timer 0 overflow interrupt service routine
interrupt [TIM0_OVF] void
timer0_ovf_isr(void)
{
// Place your code here
   //respon++;
   //overflow++;
   cycle++;

//   if (respon==3)
//   {
//      y = 1;
//      respon=0;
//   }

//   if (overflow==625){
//      overflow=0;
//      PINC.2=1;
//   }
//   if(overflow==motorservo){
//      PINC.2=0;
//   }
}

// Timer1 overflow interrupt service routine
interrupt [TIM1_OVF] void
timer1_ovf_isr(void)
{
// Reinitialize Timer1 value
TCNT1H=0x85EE >> 8;
TCNT1L=0x85EE & 0xff;
// Place your code here
sekon++;
}

// Timer2 output compare interrupt service
routine
interrupt [TIM2_COMP] void
timer2_comp_isr(void)
{
// Place your code here
kali=kali+1;
x=0;
if(kali==29000)//100000
{
   x=1;
   kali = 0;
}
}

void kirim_data( unsigned char data )
{
   /* Wait for empty transmit buffer */
   while ( !( UCSRA & (1<<UDRE)) )
   ;
   /* Put data into buffer, sends the data */
   UDR = data;
}

void kirim_string(char *s)
{
        while (*s)
        {
                kirim_data(*s++);
        }
}

////////////////////////// INTERRUPT USART RX
////////////////////////////////

char buf_tx[30];
int penanda = 0;
int data_ke = 0;
char buffer_rx[10];
float data_load_cell = 0;
int data;
void olah_data()
{
   data_load_cell = (float)buffer_rx[0]*0.1 +
(float)buffer_rx[1]*10;
}

interrupt [USART_RXC] void
usart_rx_isr(void)
{
   data=UDR;
   if(penanda == 0 && data == 'a') penanda =
1;
   else if(penanda ==1 && data == 'b')
penanda = 2;
   else if(penanda ==2 && data == 'c')
penanda = 3;
   else if(penanda == 3)
   {
      switch(data_ke)
      {
         case 0: buffer_rx[data_ke] = data;
data_ke++; break;
         case 1: buffer_rx[data_ke] = data;
data_ke++; olah_data(); data_ke = 0; penanda
= 0;break;
      }
```

```
    }
    else penanda  = 0;
}

////////////////////////////////////////////////////////////////
////////////////////////////////

int ads1115_0(){
    int a = 0;
    int b = 0;
    int data;
    //adc init
    i2c_start();
    i2c_write(0b10010000);//addr
    i2c_write(0b00000001);//config
    i2c_write(0b11000001);//msb
    i2c_write(0b10000011);//lsb
    i2c_stop();

    while((a & 0b10000000)==0){
      i2c_start();
      i2c_write(0b10010001);
      a=i2c_read(1);
      b=i2c_read(0);
      i2c_stop();
      }

    i2c_start();
    i2c_write(0b10010000);
    i2c_write(0b00000000);
    i2c_stop();

    i2c_start();
    i2c_write(0b10010001);
    a=i2c_read(1);
    b=i2c_read(0);
    i2c_stop();

    data = ((a<<8)|(b));
    return data;

    /*if(data>32768){//if data is greater than
32768,then it is false because we are using
single ended mode here
    return 0;}
    else{
    return data;}*/
}

int ads1115_1(){
    int a = 0;
    int b = 0;
    int data;
    //adc init
    i2c_start();
    i2c_write(0b10010000);//addr
    i2c_write(0b00000001);//config
    i2c_write(0b11010001);//msb
    i2c_write(0b10000011);//lsb
    i2c_stop();

    while((a & 0b10000000)==0){
      i2c_start();
      i2c_write(0b10010001);
      a=i2c_read(1);
      b=i2c_read(0);
      i2c_stop();
      }

    i2c_start();
    i2c_write(0b10010000);
    i2c_write(0b00000000);
    i2c_stop();

    i2c_start();
    i2c_write(0b10010001);
    a=i2c_read(1);
    b=i2c_read(0);
    i2c_stop();

    data = ((a<<8)|(b));
    return data;

    /*if(data>32768){//if data is greater than
32768,then it is false because we are using
single ended mode here
    return 0;}
    else{
    return data;}*/
}

int baca_adc(int ch) {
    ch &= 0b00000111;
    ADMUX = (ADMUX & 0xF8)|ch;
    ADCSRA |= (1<<ADSC);
    while(ADCSRA & (1<<ADSC));

    return(ch);
}

int ads1115_2(){
    int a = 0;
    int b = 0;
    int data;
    //adc init
    i2c_start();
    i2c_write(0b10010000);//addr
    i2c_write(0b00000001);//config
    i2c_write(0b11100001);//msb
    i2c_write(0b10000011);//lsb
```

```
   i2c_stop();

   while((a & 0b10000000)==0){
     i2c_start();
     i2c_write(0b10010001);
     a=i2c_read(1);
     b=i2c_read(0);
     i2c_stop();
     }
   i2c_start();
   i2c_write(0b10010000);
   i2c_write(0b00000000);
   i2c_stop();
   i2c_start();
   i2c_write(0b10010001);
   a=i2c_read(1);
   b=i2c_read(0);
   i2c_stop();
   data = ((a<<8)|(b));
   return data;
   /*if(data>32768){//if data is greater than
32768,then it is false because we are using
single ended mode here
   return 0;}
   else{
   return data;}*/
}
int ads1115_3(){
   int a = 0;
   int b = 0;
   int data;
   //adc init
   i2c_start();
   i2c_write(0b10010000);//addr
   i2c_write(0b00000001);//config
   i2c_write(0b11110001);//msb
   i2c_write(0b10000011);//lsb
   i2c_stop();
   while((a & 0b10000000)==0){
     i2c_start();
     i2c_write(0b10010001);
     a=i2c_read(1);
     b=i2c_read(0);
     i2c_stop();
     }
   i2c_start();
   i2c_write(0b10010000);
   i2c_write(0b00000000);
   i2c_stop();
   i2c_start();
   i2c_write(0b10010001);
   a=i2c_read(1);
   b=i2c_read(0);
   i2c_stop();
   data = ((a<<8)|(b));
   return data;
   /*if(data>32768){//if data is greater than
32768,then it is false because we are using
single ended mode here
   return 0;}
   else{
   return data;}*/
}
moving_stepper(int i)
{
   if (direct==1)    //valve buka
      { switch (i)
       {
       PORTC = 0x80;
       delay_ms(10);
       PORTC = 0x40;
       delay_ms(10);
       PORTC = 0x30;
       delay_ms(10);
       PORTC = 0x10;
       delay_ms(10);
       }
       stepcount--;
      if(stepcount<=0)
      {stepcount=0;
       direct=0;
       PORTC = 0x00;
       delay_ms(10);
       PORTC = 0x00;
       delay_ms(10);
       PORTC = 0x00;
       delay_ms(10);
       PORTC = 0x00;
       delay_ms(10);}
      }
      else if(direct==2)    //valve tutup
      {
      switch (i)
       {
       PORTC = 0x10;
       delay_ms(10);
       PORTC = 0x20;
       delay_ms(10);
       PORTC = 0x40;
       delay_ms(10);
       PORTC = 0x80;
       delay_ms(10);
       }
       stepcount++;
       if (stepcount>=maxstep)
       {direct=0;stepcount=maxstep;
       PORTC = 0x00;
       delay_ms(10);
       PORTC = 0x00;
       delay_ms(10);
```

```
      PORTC = 0x00;
      delay_ms(10);
      PORTC = 0x00;
      delay_ms(10);
      }
      }
     else if (direct==0)
      {
      PORTC = 0x00;
      delay_ms(10);
      PORTC = 0x00;
      delay_ms(10);
      PORTC = 0x00;
      delay_ms(10);
      PORTC = 0x00;
      delay_ms(10);
      }      //valve mati
}
/*void putar_kiri (void)
{
PORTC = 0x08;
       delay_ms(10);
       PORTC = 0x04;
       delay_ms(10);
       PORTC = 0x02;
       delay_ms(10);
       PORTC = 0x01;
       delay_ms(10);
}
void putar_kanan (void)
{
PORTC = 0x01;
       delay_ms(10);
       PORTC = 0x02;
       delay_ms(10);
       PORTC = 0x04;
       delay_ms(10);
       PORTC = 0x08;
       delay_ms(10);
}
void berhenti (void)
{
PORTC = 0x00;
delay_ms(10);
PORTC = 0x00;
delay_ms(10);
PORTC = 0x00;
delay_ms(10);
PORTC = 0x00;
delay_ms(10);
}*/
void sensor(){
   // baca RTC
   rtc_get_time(&jam, &menit, &detik);
   rtc_get_date(&hari, &tanggal, &bulan,
&tahun);
   // baca SPI (Termokopel)
   PORTB.4=0;
   result=(unsigned)spi(0)<<8;
   // read the LSB using SPI and combine with
MSB
   result|=spi(0);
   PORTB.4=1;
   result=(unsigned) (((unsigned long)
result*5000)/4096L);

         //baca adc
         //adc1= ads1115_1();0.1875/1000;
   //adc1 = baca_adc(1);
         adc1 = ads1115_1();//*0.1875/1000;
         nilai1 = ((float)adc1*0.0001875);

int pengisian_premium = 0;
int pengisian_solar = 0;
int liter1 = 0, time1 = 0, liter2 = 0, time2 = 0;
unsigned char display[20];
void main(void)
{
int p;
// Declare your local variables here
// Input/Output Ports initialization
// Port A initialization
// Func7=Out Func6=Out Func5=Out
Func4=Out Func3=Out Func2=Out
Func1=Out Func0=Out
// State7=0 State6=0 State5=0 State4=0
State3=0 State2=0 State1=0 State0=0
PORTA=0x00;
DDRA=0xFF;
// Port B initialization
// Func7=Out Func6=In Func5=Out
Func4=Out Func3=In Func2=In Func1=In
Func0=In
// State7=0 State6=T State5=0 State4=0
State3=P State2=P State1=P State0=P
PORTB=0x0F;
DDRB=0xB0;
// Port C initialization
// Func7=Out Func6=Out Func5=Out
Func4=Out Func3=Out Func2=Out Func1=In
Func0=In
// State7=0 State6=0 State5=0 State4=0
State3=0 State2=0 State1=T State0=T
PORTC=0x00;
DDRC=0xFC;
// Port D initialization
// Func7=Out Func6=Out Func5=Out
Func4=Out Func3=Out Func2=In Func1=In
Func0=In
```

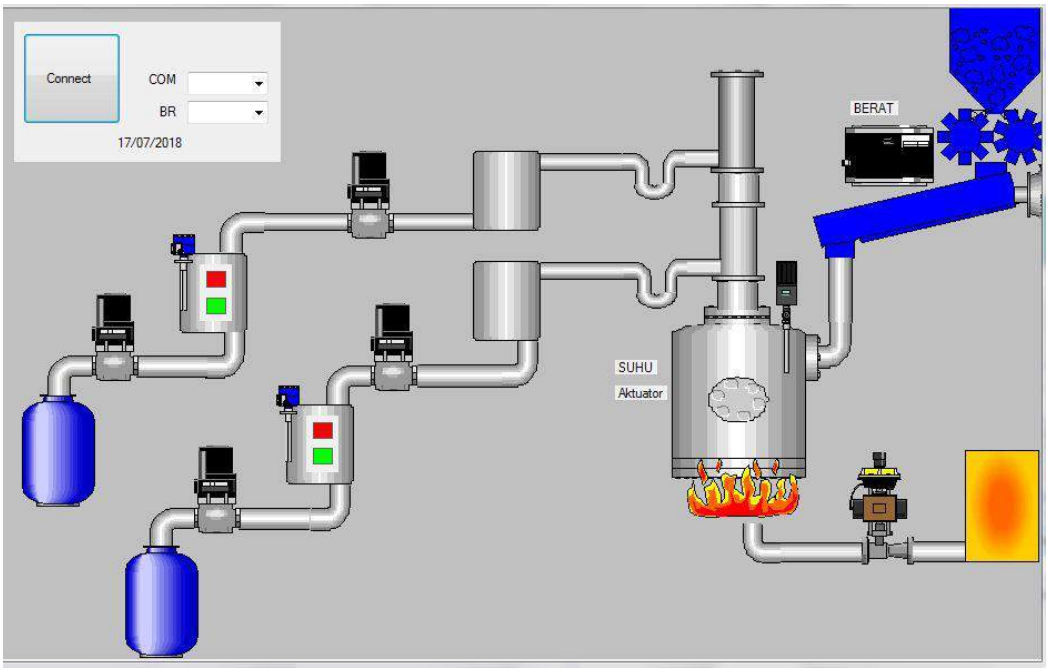
```
// State7=0 State6=0 State5=0 State4=0
State3=0 State2=P State1=T State0=T
PORTD=0x04;
DDRD=0xF8;
// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: 8000,000 kHz
// Mode: Normal top=0xFF
// OC0 output: Disconnected
TCCR0=0x01;
TCNT0=0x00;
OCR0=0x00;
// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 31,250 kHz
// Mode: Normal top=0xFFFF
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer1 Overflow Interrupt: On
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x04;
TCNT1H=0x85;
TCNT1L=0xEE;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;
// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: 125,000 kHz
// Mode: Normal top=0xFF
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x04;
TCNT2=0x4B;
OCR2=0x00;
// External Interrupt(s) initialization
// INT0: On
// INT0 Mode: Low level
// INT1: Off
// INT2: Off
GICR|=0x40;
MCUCR=0x00;
MCUCSR=0x00;
GIFR=0x40;
// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x85;
// USART initialization
// Communication Parameters: 8 Data, 1 Stop,
No Parity
// USART Receiver: On
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud Rate: 9600
UCSRA=0x00;
UCSRB=0x98;
UCSRC=0x86;
UBRRH=0x00;
UBRRL=0x33;;
/// dengan interupt
//UCSRA=0x00;
//UCSRB=0x98;
//UCSRC=0x86;
//UBRRH=0x00;
//UBRRL=0x33;
// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by
Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;
// ADC initialization
// ADC disabled
ADCSRA=0x00;
// SPI initialization
// SPI Type: Master
// SPI Clock Rate: 2000,000 kHz
// SPI Clock Phase: Cycle Start
// SPI Clock Polarity: Low
// SPI Data Order: MSB First
SPCR=0x50;
SPSR=0x00;
// TWI initialization
// TWI disabled
TWCR=0x00;
// I2C Bus initialization
i2c_init();
// DS1307 Real Time Clock initialization
// Square wave output on pin SQW/OUT: Off
// SQW/OUT pin state: 0
rtc_init(0,0,0);
// Alphanumeric LCD initialization
// Connections specified in the
// Project|Configure|C
Compiler|Libraries|Alphanumeric LCD menu:
// RS - PORTA Bit 0
// RD - PORTA Bit 1
// EN - PORTA Bit 2
// D4 - PORTA Bit 4 q
// D5 - PORTA Bit 5
// D6 - PORTA Bit 6
// D7 - PORTA Bit 7
// Characters/line: 16
```

```
lcd_init(20);
stepcount=0;
target_valve=0;
// Kondisi awal
SV_Premium_Bawah = 0;
SV_Premium_Atas = 1;
SV_Solar_Bawah = 0;
SV_Solar_Atas = 1;
perhitungan_liter=0;
liter=0;
kondisi=0;
rtc_set_time(13,30,00);
rtc_set_date(1,23,07,18);
// Global enable interrupts
#asm("sei")
delay_ms(1000)
while (1)
    {
      sensor();

      if
((perhitungan_liter==1)&&(cycle>=66)){liter=l
iter+0.08;cycle=0;}
      /*if (result >= 400)
      {PORTA.3 = 1;
           // led emergency
      }
      else
      {
         PORTA.3 = 0;
      }*/
      sprintf(buff,"Suhu=%4u.%u
%cC",result/40,(result%40),0xDF);
      //sprintf(buff,"Suhu=%3d",target_valve);
      lcd_gotoxy(0,0);
      lcd_puts(buff);
      valve_open=stepcount/(maxstep/100);
      lcd_gotoxy(0,1);
      lcd_putsf("open=");
      sprintf(temp,"%3d",valve_open);
      lcd_gotoxy(5,1);
      lcd_puts(temp);
      sprintf(temps,"%3d",nilai1);
      lcd_gotoxy(12,1);
      lcd_puts(temps);
      delay_ms(10);
         p++;
         if (p>=9) p=1;
         moving_stepper(p);
         //Kendali Arah Stepper
         if
(target_valve>=valve_open+5)direct=2;
         else if (target_valve<=valve_open-
5)direct=1;
         else if
((target_valve<=valve_open+4)&&(target_valv
e>=valve_open-4))direct=0;
         //progam setpoint valve
         awalerror=deltaH;   // 15
         deltaH=200-(result/40); //  14
         derivative= deltaH-awalerror;
         target_valve=(Kp * deltaH)+(Kd*
derivative);
         if
(target_valve>=99){target_valve=100;}
         else
if(target_valve<=1){target_valve=0;}

    if(data_load_cell>=300 && state != 9){
      state = state + 1;
    }
    if(state == 9){
      kondisi = 1;
      }
    if (kondisi == 1)    //proses berjalan
      {
         PORTD.3 = 1; //led proses
         /*p++;
         if (p>=2) p=1;
         moving_stepper(p);*/
         //Kendali Arah Stepper
         if (target_valve<=valve_open-
5){direct=2;}
         else if
(target_valve>=valve_open+5){direct=1;}
         else if ((target_valve<=valve_open-
4)&&(target_valve>=valve_open+4)){direct=0;}
         //progam setpoint valve
         awalerror=deltaH;   // 15
         deltaH=60-(result/40); //  14
         derivative= deltaH-awalerror;
         target_valve=(Kp * deltaH)+(Kd*
derivative);
         if
(target_valve>=99){target_valve=100;}
         else
if(target_valve<=1){target_valve=0;}
      }

    else if (kondisi == 0)       // proses mati
      {
        PORTD.3 = 1; //led proses
        if (sekon<=15)PORTA.3=0; //pc3 relay
        else if (sekon >=14)PORTA.3=1;
      }

      if (x==0){
         sprintf(buff,"%d:%d:%d %d/%d/%d",
jam, menit, detik, tanggal, bulan, tahun);
```

```
        kirim_string(buff);
        kirim_data(59);

sprintf(buff,"%4u.%u",result/40,(result%40),0
xDF);
        kirim_string(buff);
        kirim_data(59);
        sprintf(buff,"%3d",valve_open);
        kirim_string(buff);
        kirim_data(59);
        //sprintf(buff,"%0.01f",data_load_cell);

         sprintf(temps,"%3d",nilai1);
        kirim_string(temps);
        kirim_data(59);
        if (k==0 && l == 0){
        kirim_string("#0#0#"); //level 1
low;level 2 low
        }
        else if (k==0 && l == 1){
        kirim_string("#0#1#"); //level 1
low;level 2 high
        }
        else if (k==1 && l == 0){
        kirim_string("#1#0#"); //level 1
high;level 2 low
        }
        else if (k==1 && l == 1){
        kirim_string("#1#1#"); //level 1
high;level 2 high
        }
        kirim_data(13);
        kirim_data(10);
        /*putchar(13);
        putstring(buff);
        putchar(59);
        sprintf(buff, "%0.0001f C", result);
        putstring(buff);
        putchar(10); // baris baru
        putchar(13); // kolom 'home'*/
        x = 0;
        //delay (100);
        }

        if(SW_Premium_Atas == 0){
        pengisian_premium = 0;
       }
       if(SW_Premium_Bawah == 1){
        pengisian_premium = 1;
       }

        if(SW_Solar_Atas == 0){
        pengisian_solar = 0;
       }
       if(SW_Solar_Bawah == 1){
        pengisian_solar = 1;
       }

    if(pengisian_premium == 1){
       lcd_gotoxy(0,2);
       lcd_putsf("Level P : Low   ");
       SV_Premium_Atas = 1; // Selenoid atas
       SV_Premium_Bawah = 0; // Selenoid
bawah
       liter1 = 0; time1=0;
       k=0;
       }else if (pengisian_premium == 0){
          if(SW_Premium_Bawah == 0 &&
liter1 != 1){
             if(time1 >= timerx){
               liter1 = 1;
               time1 = 0;
               pengisian_premium = 2;
             }else{
               time1++;
             }
             sprintf(display,"Level P : High
%2d",time1);
             lcd_gotoxy(0,2);  lcd_puts(display);
             SV_Premium_Atas = 0; // Selenoid
atas
             SV_Premium_Bawah = 1; //
Selenoid bawah
             l=1;
          }
       }else if(pengisian_premium == 2){
       lcd_gotoxy(0,2);
       if (SW_Premium_Atas == 0 &&
SW_Premium_Bawah == 0){
pengisian_premium = 0;}
       lcd_putsf("    STOP-WAITING    ");
       SV_Premium_Atas = 1; // Selenoid atas
       SV_Premium_Bawah = 0; // Selenoid
bawah
       }

       if(pengisian_solar == 1){
       lcd_gotoxy(0,3);
       lcd_putsf("Level S : Low   ");
       SV_Solar_Atas = 1; // Selenoid atas
       SV_Solar_Bawah = 0; // Selenoid bawah
       perhitungan_liter=0;
       liter2=0; time2 = 0;
       k=0;
       }else if(pengisian_solar == 0){
          if(SW_Solar_Bawah == 0 && liter2 !=
1){
             if(time2 >= timerx){
               liter2 = 1;
               time2 = 0;
```

```
            pengisian_solar = 2;
          }else{
            time2++;
          }
          sprintf(display,"Level S : High
%2d",time2);
          lcd_gotoxy(0,3);  lcd_puts(display);
          SV_Solar_Atas = 0; // Selenoid atas
          SV_Solar_Bawah = 1; // Selenoid
bawah
          perhitungan_liter=1;
          cycle=0;
          l=1;
        }
      }else if(pengisian_solar == 2){
        lcd_gotoxy(0,3);
        if (SW_Solar_Atas == 0 &&
SW_Solar_Bawah == 0){    pengisian_solar =
0;}
        lcd_putsf("PENGISIAN STOP-
WAITING");
        SV_Solar_Atas = 1; // Selenoid atas
        SV_Solar_Bawah = 0; // Selenoid
bawah
      }
    delay_ms(200);
   }
}
```

LAMPIRAN F



Code :

```
Imports System
Imports System.IO
Imports System.IO.Ports
Imports System.Threading
Imports System.ComponentModel


Public Class Form1
    Dim datasensor As String
    Dim myPORT As String
    Dim myBR_temp As String, myBR As Integer
    Dim dtt As Date = Today
    Dim receivedData As String = ""
    Shared random As New Random()

    Private Sub Button2_Click(sender As Object, e As EventArgs)
        ' do nothing
    End Sub

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click

        If (Button1.Text = "Connect") Then
            If (myPORT <> "" Or myBR_temp <> "") Then
                Serial1.Close()
                Serial1.PortName = myPORT
                Serial1.BaudRate = myBR
                Serial1.DataBits = 8
                Serial1.Parity = Parity.None
                Serial1.StopBits = StopBits.One
                Serial1.Handshake = Handshake.None
                Serial1.Encoding = System.Text.Encoding.Default
```

```
            Serial1.ReadTimeout = 10000

            Serial1.Open()
            Button1.Text = "Disconnect"
            Timer1.Enabled = True
            ' Timer_LBL.Text = "Timer: ON"
        Else
            MsgBox("COM and Baudrate cant Blank")
            ComboBox1.Focus()
            ComboBox2.Focus()
        End If
    Else
        Serial1.Close()
        Button1.Text = "Connect"
        Timer1.Enabled = False
        ' Timer_LBL.Text = "Timer: OFF"
    End If

    Label5.BackColor = Color.Gray
    Label6.BackColor = Color.Gray
    Label7.BackColor = Color.Gray
    Label8.BackColor = Color.Gray

End Sub

Private Sub Timer1_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick
    'Label2.Text = Convert.ToString(random.Next(0, 1000))
    Dim tag1 As Integer, tag2 As Integer
    Dim led1 As Integer, led2 As Integer
    ' Dim s As String = "1262666; 99; 0.0; 12122f; hghjg; gfg;"

    tag1 = 0
    tag2 = 0
    receivedData = ReceiveSerialData()
    'Label1.Text &= receivedData
    Dim splitData1 As String() = receivedData.Split(New Char() {";"c})
    Dim splitData2 As String() = receivedData.Split(New Char() {"#"c})

    Dim datadiv1 As String
    For Each datadiv1 In splitData1
        Select Case tag1
            Case 1
                Label1.Text = datadiv1 ' suhu
            Case 2
                Label4.Text = datadiv1 ' aktuator
            Case 3
                Label2.Text = datadiv1 'beban
        End Select
        tag1 += 1
    Next

    Dim datadiv2 As String
    For Each datadiv2 In splitData2
        Select Case tag2
            Case 1
```

```
                led1 = CInt(datadiv2) ' level1
            Case 2
                led2 = CInt(datadiv2) ' level2
        End Select
        tag2 += 1
    Next

    If led1 = 0 And led2 = 0 Then
        Label5.BackColor = Color.Lime
        Label6.BackColor = Color.Gray
        Label7.BackColor = Color.Lime
        Label8.BackColor = Color.Gray
    ElseIf led1 = 0 And led2 = 1 Then
        Label5.BackColor = Color.Lime
        Label6.BackColor = Color.Gray
        Label7.BackColor = Color.Gray
        Label8.BackColor = Color.Red
    ElseIf led1 = 1 And led2 = 0 Then
        Label5.BackColor = Color.Gray
        Label6.BackColor = Color.Red
        Label7.BackColor = Color.Lime
        Label8.BackColor = Color.Gray
    ElseIf led1 = 1 And led2 = 1 Then
        Label5.BackColor = Color.Gray
        Label6.BackColor = Color.Red
        Label7.BackColor = Color.Gray
        Label8.BackColor = Color.Red
    End If

    tag1 = 0
    tag2 = 0

End Sub

Private Sub DateTimePicker1_ValueChanged(sender As Object, e As EventArgs)
    '
End Sub

Private Sub ComboBox1_SelectedIndexChanged(sender As Object, e As EventArgs) Handles
ComboBox1.SelectedIndexChanged
    If (ComboBox1.SelectedItem <> "") Then
        myPORT = ComboBox1.SelectedItem
    End If
End Sub

Private Sub ComboBox2_SelectedIndexChanged(sender As Object, e As EventArgs) Handles
ComboBox2.SelectedIndexChanged
    If (ComboBox2.SelectedItem <> "") Then
        myBR_temp = ComboBox2.SelectedItem
        myBR = CInt(myBR_temp)
    End If
End Sub

Private Sub ProgressBar1_Click(sender As Object, e As EventArgs)
    '
```

```
    End Sub

    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles Me.Load
        Timer1.Enabled = False
        date_Time.Text = dtt.Date

        myPORT = ""
        For Each sp As String In My.Computer.Ports.SerialPortNames
            ComboBox1.Items.Add(sp)
        Next

        '''''''
        ComboBox2.Items.Add("2400")
        ComboBox2.Items.Add("9600")
        ComboBox2.Items.Add("115200")
        ComboBox2.Items.Add("345000")

    End Sub

    Function ReceiveSerialData() As String
        Dim Incoming As String

        Try
            Incoming = Serial1.ReadLine()
            If Incoming Is Nothing Then
                Return "nothing" & vbCrLf
            Else
                Return Incoming
            End If
        Catch ex As TimeoutException
            Return "Error: Serial Port read timed out."
        End Try

    End Function

End Class
```

BIODATA PENULIS

Penulis dilahirkan di Kota Surabaya, 3 Februari 1998. Penulis merupakan anak ketiga dari tiga bersaudara. Saat ini penulis tinggal di Jalan Abdulrahman No.103 pabean sedati, Sidoarjo. Pada tahun 2004 penulis menyelesaikan pendidikan di TK Al-Huda, Sidoarjo. Tahun 2010 lulus dari SD HANG TUAH 10 JUANDA. Tahun 2013 lulus dari SMPN 1 Sedati dan tahun 2015 dari MA AMANATUL UMMAH . Penulis diterima di Departemen Teknik Instrumentasi Fakultas Vokasi ITS. Penulis aktif sebagai asisten laboratorium Rekayasa Instrumentasi dan Kontrol Divisi *Maintenance and Equipment*. Penulis berhasil menyelesaikan tugas akhir dengan judul **"SISTEM MONITORING VARIABEL PROSES PADA RANCANG BANGUN ALAT PRODUKSI BAHAN BAKAR MINYAK (BBM) DARI LIMBAH PLASTIK MENGGUNAKAN HMI (HUMAN MACHINE INTERFACE)".** Bagi pembaca yang memiliki kritik, saran , atau pertanyaan mengenai tugas akhir ini dapat menghubungi penulis melalaui email ayusafitri3298@gmail.com. Sekian dan terimakasih.