UNIVERSITAS INDONESIA

MONITORING SUHU BERBASIS PROTOKOL HART

SKRIPSI

MUHAMMAD ADMULYA'S
0906602881

FAKULTAS TEKNIK UNIVERSITAS INDONESIA
DEPARTEMEN ELEKTRO
DEPOK
JULI 2012

UNIVERSITAS INDONESIA

MONITORING SUHU BERBASIS PROTOKOL HART

SKRIPSI

Diajukan Sebagai Salah Satu Syarat Untuk Kelulusan
Pada Jurusan Teknik Elektro Universitas Indonesia

MUHAMMAD ADMULYA'S
0906602881

FAKULTAS TEKNIK UNIVERSITAS INDONESIA
DEPARTEMEN ELEKTRO
DEPOK
JULI 2012

HALAMAN PERNYATAAN ORISINALITAS

Skrispi ini adalah hasil karya saya sendiri,

dan semua sumber baik yang dikutip maupun dirujuk

telah saya nyatakan dengan benar.

Nama	**: Muhammad Admulya's**
NPM	**: 0906602881**
Tanda Tangan	**: .....**
Tanggal	**: 6 Juli 2012**

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh :

Nama	:	Muhammad Admulya's
NPM	:	09902881
Program Studi	:	Teknik Elektro
Judul Skripsi	:	***MONITORING*** **SUHU BERBASIS PROTOKOL HART**

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Teknik pada Program Studi Teknik Elektro Fakultas Teknik, Universitas Indonesia

DEWAN PENGUJI

Pembimbing	:	Prof.Dr.Ir. Nji Raden Poespawati, M.T.	( )
Penguji I	:	Dr. Ir. Agus Santoso Tamsir, M.T.	( )
Penguji II	:	Taufiq Alif Kurniawan, S.T. MSc.Eng.	( )

Ditetapkan di : Depok
Tanggal : 6 Juli 2012

UCAPAN TERIMA KASIH

Puji syukur saya panjatkan kepada Allah SWT, karena atas berkat dan rahmat-Nya, saya dapat menyelesaikan skripsi ini. Penulisan skripsi ini dilakukan dalam rangka memenuhi salah satu syarat untuk mencapai gelar Sarjana Teknik Jurusan Elektro pada Fakultas Teknik Universitas Indonesia. Saya menyadari bahwa, tanpa bantuan dan bimbingan dari berbagai pihak, dari masa perkuliahan sampai pada penyusunan skripsi ini, sangatlah sulit bagi saya untuk menyelesaikan skripsi ini.

Dalam penyusunan skripsi ini, penulis banyak mendapatkan bantuan baik materil maupun moril dari berbagai pihak. Oleh karena itu, penulis mengucapkan terima kasih kepada :

(1) Prof. Dr. Ir. Nji Raden Poespawati, M.T. selaku dosen pembimbing yang telah menyediakan waktu, tenaga, dan pikiran untuk mengarahkan saya dalam penyusunan skripsi ini.

(2) Korosi Specindo atas alat bantu dan standar HART 7.2 yang berperan penting dalam membantu merumuskan penulisan skripsi ini.

Harapan penulis kiranya skripsi ini dapat memberikan pengetahuan yang bermanfaat bagi penulis khususnya dan pembaca pada umumnya. Semoga Allah SWT senantiasa melimpahkan rahmat dan hidayah pada kita semua. Amiinn.

Depok, 6 Juli 2012

Muhammad Admulya's

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI SKRIPSI UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Universitas Indonesia, saya yang bertanda tangan di bawah ini:

Nama : Muhammad Admulya's

NPM : 0906602881

Departemen : Teknik Elektro

Fakultas : Teknik

Jenis karya : Skripsi

Demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Noneksklusif (*Non-exclusive Royalty- Free Right*)** atas karya ilmiah saya yang berjudul :

***MONITORING* SUHU BERBASIS PROTOKOL HART**

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/formatkan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan mempublikasikan skripsi saya tanpa meminta izin dari saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok

Pada tanggal : 6 Juli 2012

Yang menyatakan

(Muhammad Admulya's)

ABSTRAK

Nama : Muhammad Admulya's
Program Studi : Teknik Elektro
Judul : *MONITORING* SUHU BERBASIS PROTOKOL HART

Tujuan protokol HART adalah untuk membuat instrumentasi dapat berkomunikasi secara digital satu dengan yang lainnya dengan menggunakan kabel yang sama yang digunakan untuk mengirim sinyal analog 4-20 mA. Protokol HART menggunakan teknik modulasi FSK. Sinyal digital terdiri dari dua frekuensi yaitu 1200 dan 2200 Hz yang merepresentasikan bit 1 dan 0. Pada *Monitoring* Suhu Berbasis Protokol HART di skripsi ini, tegangan *supply* dan panjang kabel dapat mempengaruhi akurasi pembacaan suhu. Minimum akurasi rata-rata pengukuran mencapai 96,62% untuk *field device address* 0 dan 95,84% untuk *address* 1.

Kata Kunci : *Loop Current* 4-20 mA, FSK, HART PDU, *Command,* Suhu

ABSTRACT

Name : Muhammad Admulya's
Study Program : Electrical Engineering
Title : TEMPERATURE MONITORING BASED ON HART PROTOCOL

The purpose of the HART protocol was to create a way for instruments to digitally communicate with one another over the same wire used to convey a 4-20 mA analog signal. The HART protocol uses the FSK modulation technique. The digital signal is made up of two frequencies 1,200 Hz and 2,200 Hz representing bits 1 and 0. On the Temperature Monitoring Based on HART Protocol in this undergraduate thesis, voltage supply and cable length can be influence measurement accuracy of temperature. The Minimum average accuracy has reached 96,62% for field device address 0 and 95,84% for address 1.

Key Words : Loop Current 4-20mA, FSK, HART PDU, Command, Temperature

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR KODE

DAFTAR LAMPIRAN

DAFTAR SINGKATAN

ADC	Analog to Digital Converter
BFSK	Binary Frequency Shift Keying
CPFSK	Continous Phase Frequency Shift Keying
DAC	Digital to Analog Converter
DTR	Data Time Request
GUI	Graphical User Interface
HART	Highway Addressable Remote Transducer
HCF	HART Communication Foundation
IDE	Integrated Development Environment
FD	Field Device
FSK	Frequency Shift Keying
PDU	Protocol Data Unit
UART	Universal Asynchronous Receiver Transmitter

DAFTAR ISTILAH

Application Layer	Lapisan teratas model OSI 7 - *layer*, meliputi HART *command* dan tipe data
Byte Count	Panjang data antara *byte count* dengan *check byte*
Check Byte	Nilai XOR dari keseluruhan data frame untuk mendeteksi *error*
Command	Instruksi HART
Data	Bagian ke-7 dari *frame* HART
Data Link Layer	Lapisan ke-2 model OSI 7-*layer*, meliputi fungsi pembentukan pesan, *error detection*.
Delimiter	Bagian ke-2 dari frame HART meliputi jenis *frame*, jenis *address*
Enum	Tipe data *integer* dari nilai numerik pada *Common Tables Specific*
Expanded Device Type	Nilai data 2 *byte* dari tipe *device* yang diberikan HCF
Field Device	Perangkat instrumentasi di lapangan
Float	Tipe data dengan mengacu IEEE 754
Frame	Satu rangkaian HART PDU
FSK	*Frequency Shift Keying*, modulasi dan demodulasi data digital
Half Duplex	Komunikasi 1 arah
Loop Current	Nilai arus dalam mA pada *field device*
Mark	Bit 1 = 1200 Hz
Master	*Host*/PC
Message	Pesan dari HART PDU
Multidrop	Tipe jaringan komunikasi digital dengan banyak *field device* yang terhubung
PDU	*Protocol Data Unit*, satu struktur *frame* HART
Physical Layer	Lapisan terbawah model OSI 7-*layer*, meliputi hubungan *electrical*
Point to Point	Tipe jaringan dari *master* langsung ke satu *field device*
Polling Address	Alamat dari *field device* dengan panjang 1 *byte*
Preamble	Informasi tanda awal suatu *message* HART PDU, dengan nilai 0xFF
Request Data Bytes	Data yang diminta *master* terhadap *slave* (*field device*)
Respon Data Bytes	Data yang dikirim kembali oleh *field device* ke *master*
Slave	*Field device*
Space	Bit 0 = 2200 Hz
Transmitter	*Field device* dengan 4-20mA *loop current*

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Pada suatu industri, jarak antara suatu instrumentasi dengan *control room* umumnya memiliki jarak yang cukup jauh. Jarak ini biasanya menjadi permasalahan pada perangkat intrumentasi tersebut, dimana sistem komunikasi/sistem transmisi antara perangkat instrumen dengan instrumen lain nya atau instrumen dengan sistem pemantau, rentan terhadap lingkungan luar.

Sinyal analog 4-20 mA merupakan transmisi sinyal yang paling populer dalam sistem industri pada aplikasi instrumentasi karena sinyal ini lebih kebal terhadap gangguan lingkungan luar. Sinyal analog yang ditransmisikan ini berupa arus yang jika dibandingkan dengan sinyal transmisi yang menggunakan tegangan, arus mampu menjangkau jarak yang jauh karena nilai arus tidak mengalami perubahan nilai, sedangkan tegangan mengalami drop tegangan seiring dengan bertambah nya resistansi kabel. Sesuai dengan hukum Kirchoff dimana arus yang masuk sama dengan arus yang keluar. Sinyal analog 4-20mA sangat mudah dikonversikan menjadi sinyal analog tegangan 1-5 Volt yang umum digunakan pada sistem digital maupun analog.

Sinyal 4-20 mA murni berisi informasi sinyal analog dimana sinyal ini hanya mampu mentransmisikan suatu nilai dari variabel yang terukur, sehingga sinyal ini tidak mampu digunakan untuk memonitor ataupun mengkalibrasi suatu perangkat instrumentasi (*field device*). Sistem *monitoring* suhu yang menggunakan protokol HART (*Highway Addressable Remote Transducer*) tidak akan mengalami kendala dalam hal kemampuan transmisi data dari *field device* ke sistem *monitoring* yang berada di *control room* atau sebaliknya, karena dengan menggunakan protokol HART permasalahan tersebut bisa diatasi tanpa memerlukan kabel/jalur komunikasi tambahan, sehingga mengurangi biaya produksi. Protokol HART memanfaatkan *loop* arus 4-20 mA yang telah ada dengan menumpangkan sinyal digital pada arus 4-20 mA.

1.2 Tujuan

Tujuan penulisan skripsi ini adalah merancang dan membangun perangkat instrumentasi *monitoring* suhu untuk dapat berkomunikasi dengan personal komputer menggunakan protokol HART.

1.3 Pembatasan Masalah

Pada skripsi ini, perancangan meliputi pembentukan model OSI 7-*Layer* yang hanya difokuskan pada *Data link layer* dan *Application Layer*. Adapun permasalahan lain yang ditentukan pada *monitoring* suhu berbasis protokol HART ini adalah sebagai berikut :

- Protokol HART yang dibangun mengikuti standar HART revisi 7 yang dikeluarkan oleh *HART Communication Foundation* (HCF).
- *Hardware* (rangkaian *Loop powered* dan HART modem) dibuat berdasarkan *Application Note* AN-534 dari *Analog Device*.
- Sensor suhu yang digunakan adalah sensor suhu yang terdapat di dalam (*built in temperature sensor*) mikrokontroler *Silabs* C8051F350.
- Panjang kabel maksimum yang digunakan dalam pengujian komunikasi data adalah 1000 *feet* atau 304,8 meter.
- *Thermometer* yang digunakan sebagai suhu pembanding dianggap sebagai suhu ideal.

1.4 Metodologi

Metodologi yang digunakan dalam skripsi ini adalah dengan meracang dan membangun perangkat sistem pemantau suhu dengan rangkaian urutan kegiatan sebagai berikut :

- Melakukan studi *literature* yang mendukung, mulai dari mencari bahan referensi hingga melakukan tanya jawab langsung kepada orang yang ahli dibidangnya.
- Perancanaan dan pembuatan sistem *hardware* seperti menentukan komponen tambahan yang akan digunakan.

- Merancang dan membuat HART PDU (*Protocol Data Unit*) pada *field device* dan program aplikasi *monitoring*.
- Menguji dan menganalisa hasil rancangan dan pembuatan.
- Mengambil kesimpulan

1.5 Sistematika Penulisan

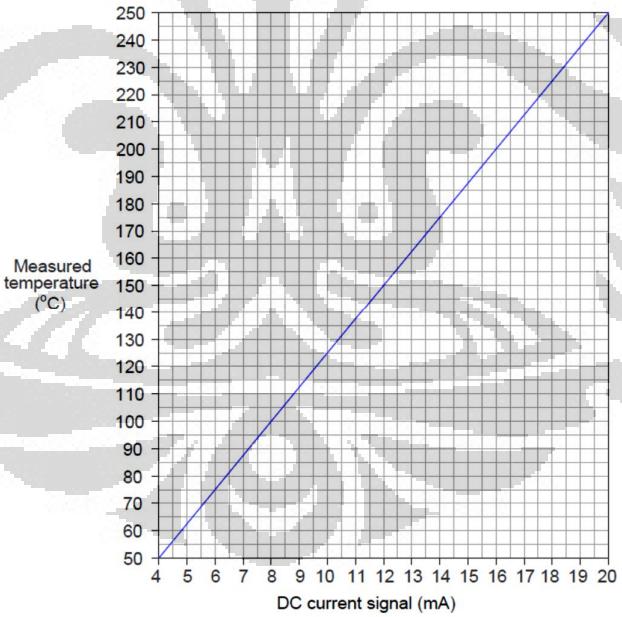
Untuk mempermudah penulisan, skripsi ini terbagi atas lima bab yang secara garis besar dapat diuraikan sebagai berikut : bab satu merupakan pendahuluan yang berisi penjelasan awal mengenai latar belakang masalah, tujuan penulisan, ruang lingkup permasalahan, batasan masalah, metodologi perancangan dan sistematika penulisan yang digunakan dalam penulisan skripsi. Bab dua berisi teori-teori dasar yang mendukung penulisan skripsi ini antara lain *loop* arus 4-20 mA , teori dasar mengenai protokol HART, FSK *physical layer* dan HART PDU. Bab tiga merupakan uraian tahapan perancangan dan implementasinya yang meliputi *software* pemantau suhu serta *hardware* meliputi *physical layer* yang sesuai dengan standar yang telah ditentukan oleh HCF (*HART Communication Foundation*). Bab empat berisi tentang ujicoba dan analisa dari hasil pengukuran *monitoring* suhu. Bab terakhir, yaitu bab lima mengambil kesimpulan dari semua hasil ujicoba dan analisa.

BAB 2
DASAR TEORI

2.1 *Loop* Arus 4-20 mA[1]

Sinyal transmisi yang paling popular digunakan pada industri moderen saat ini adalah arus standar 4-20 mA. Sinyal ini merupakan sinyal analog standar yang berarti bahwa arus listrik ini digunakan untuk mempresentasikan nilai suatu pengukuran maupun suatu sinyal perintah. Nilai arus 4 mA biasanya mempresentasikan pada nilai skala terendah 0 %, sedangkan nilai maksimum 20 mA mempresentasikan skala maksimum pada 100% dan nilai arus antara 4-20 mA mempresentasikan nilai persentase antara 0% - 100%.

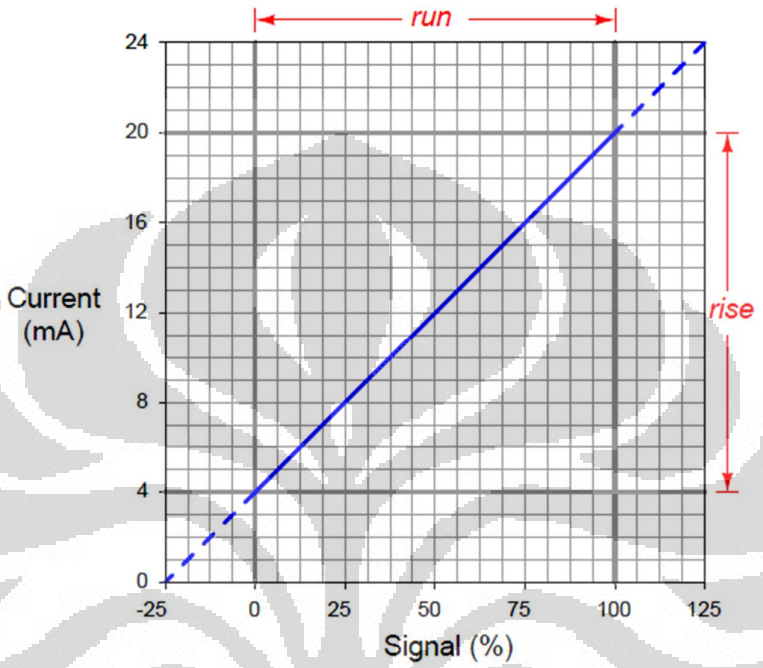
Sebagai contoh, jika diinginkan mengkalibrasi nilai arus 4-20 mA untuk mengukur nilai suhu 50^o – 250^o C, didapat hubungan nilai arus 4-20 mA dan nilai pengukuran suhu menjadi grafik seperti pada Gambar 2.1.



Gambar 2.1 Contoh hubungan arus 4-20 mA terhadap suhu[1]

2.1.1 Hubungan Sinyal 4-20 mA Terhadap Variabel Instrumentasi[1]

Sinyal arus analog 4-20 mA mempresentasikan skala nilai antara 0% hingga 100%, skala ini biasanya merupakan suatu skala yang linier seperti pada Gambar 2.2.



Gambar 2.2 Hubungan arus 4-20 mA terhadap skala persentase 0-100%[1]

Dengan menggunakan persamaan linier, bisa didapatkan suatu nilai persentase terhadap nilai arus.

$y = mx+b$ (2.1)

dimana :

y = nilai dari *output* instrumen

x = nilai dari *input* instrumen

m = *slope* (kemiringan)

b = y – titik perpotongan

Berdasarkan persamaan linier pada Persamaan 2.1, nilai m (nilai *slope*) didapatkan dari pembagian garis *rise* (4 – 20 mA) terhadap garis *run* (0 - 100%) sehingga didapatkan nilai m, yaitu :

$$m = \frac{rise}{run} = \frac{(20\text{-}4)}{(100\text{-}0)} = \frac{16}{100}$$

$$y = \left(\frac{16}{100}\right)x+b$$

Nilai b didapatkan dengan memasukan nilai m ke Persamaan 2.1 dengan asumsi nilai *input* (x) dan *output* (y) berada pada nilai terkecil, yang berarti x=0 dan y=4.

$$4=\left(\frac{16}{100}\right)0+b$$

$$4=0+b$$

$$b=4$$

Setelah mendapatkan nilai b dan m, bisa dihitung sinyal arusnya. Contoh, untuk mengkonversikan persentase 34,7% menjadi 4-20 mA, maka bisa didapatkan sebagai berikut :

$$y = \left(\frac{16}{100}\right)34,7+4$$

$$y = 5,552+4$$

$$y = 9,552$$

sehingga, persentase 34,7% mengintrepesentasikan arus sebesar 9,552 mA pada skala 4-20 mA. Untuk mengubah skala nilai analog 4-20 mA menjadi persentase 0- 100%, persamaannya menjadi :

$$\text{persen} = \left(\frac{x\text{-}4mA}{20mA\text{-}4mA}\right)100\% \qquad (2.2)$$

dimana :

x = Nilai arus yang akan dirubah menjadi persen

Contoh, diinginkan mengubah sinyal analog 5mA menjadi skala persentase, maka nilai persentasenya dapat dihitung sebagai berikut

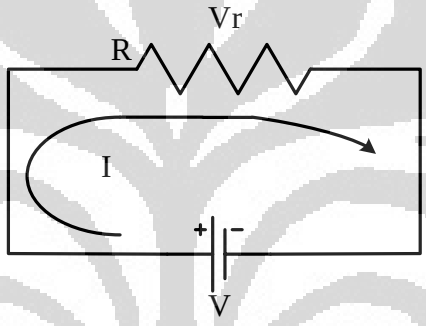
$$\text{persen} = \left(\frac{5mA\text{-}4mA}{20mA\text{-}4mA}\right)100\%$$

$$\text{persen} = \left(\frac{1mA}{16mA}\right)100\%$$

$$= 6,25\%$$

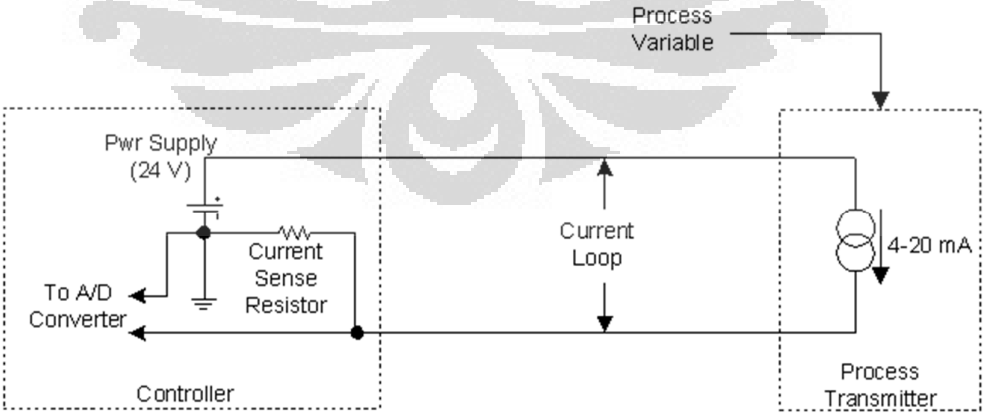
2.1.2 Arus *Loop* 4-20 mA Konvensional

Jika pada suatu *loop* tertutup terdapat hambatan yang di pasang seri seperti pada Gambar 2.3, maka tegangan jatuh (*voltage drop*) akan sebanding dengan perbandingan hambatan tersebut dan arus yang mengalir pada *loop* tersebut adalah sama. Dengan menggunakan kabel yang sangat panjang, nilai Vr akan berubah dikarenakan panjang kabel, sehingga menambah hambatan baru yang akan mempengaruhi perbandingan tegangan jatuh. Arus yang mengalir akan tetap sama atau konstan sebagaimana dengan hukum Kirchoff, arus yang masuk sama dengan arus yang keluar, sehingga panjang kabel tidak mempengaruhi nilai arus.



Gambar 2.3 Prinsip dasar *loop* arus

Gambar 2.4 merupakan rangkaian dasar *loop* arus 4-20 mA *transmitter* konvensional yang umum digunakan dengan *power supply* sebesar 24 V_{DC}.



Gambar 2.4 Rangkaian dasar *loop* arus 4-20 mA [2]

Rangkaian dasar *loop* arus 4-20 mA ini terdiri dari 4 komponen utama, yaitu :

1. Catu Daya

   Catu daya untuk rangkaian *loop* 4-20 mA umumnya dipasok dengan tegangan 36 V_{DC}, 24 V_{DC}, 15 V_{DC} dan 12 V_{DC}.

2. *Transmitter*

   Transmitter adalah bagian utama dari sistem sinyal 4-20 mA. Dalam *loop* 4-20 mA, 4 mA merupakan titik pengukuran terendah dan 20mA merupakan titik tertinggi. Beberapa *transmitter* saat ini menggunakan catu daya 15-24 V_{DC}. Tegangan rendah adalah tegangan minimum yang dibutuhkan untuk menjamin operasi yang tepat dari *transmitter*. Tegangan tinggi adalah tegangan maksimum *transmitter* untuk dapat bertahan dan beroperasi dengan spesifikasi yang ditetapkan.

3. Resistor penerima

   Resistor digunakan untuk mengubah arus menjadi tegangan dengan nilai 250Ω. Arus yang mengalir melalui resistor tersebut akan menghasilkan tegangan yang mudah diukur oleh *input* kontrol analog. Resistor yang paling umum dalam sebuah *loop* 4-20 mA adalah 250Ω, namun tergantung pada aplikasinya, resistor 100Ω sampai 750Ω dapat digunakan juga. Untuk resistor 250Ω, tegangan akan terukur 1 V_{DC} pada arus *loop* 4 mA dan 5 V_{DC} pada arus *loop* 20 mA.

4. Kabel

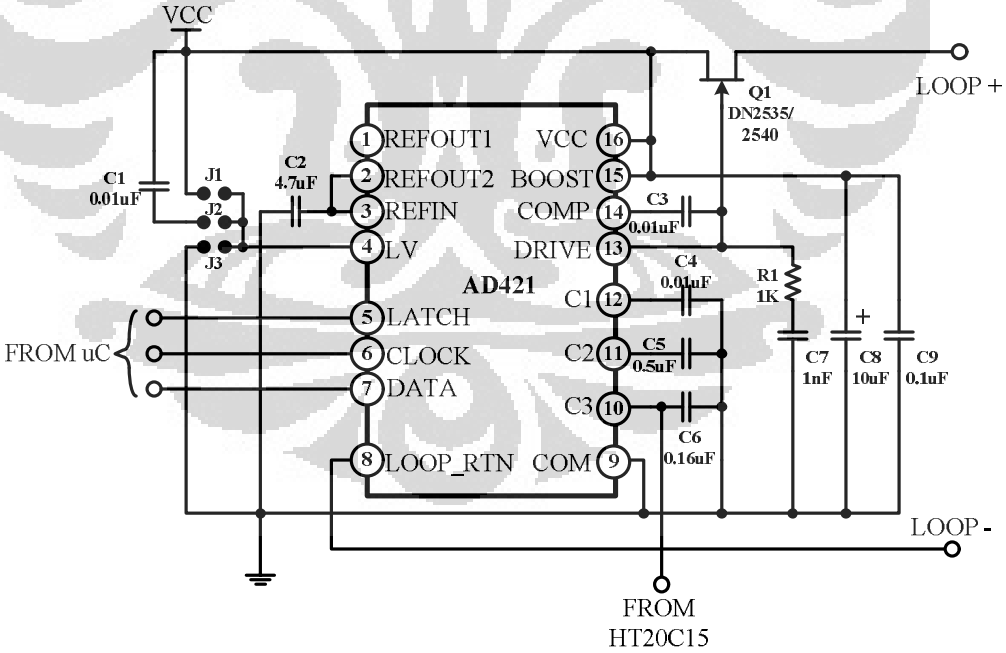
   Kabel digunakan untuk mentransmisi sinyal, namun tahanan dari kabel menghasilkan tegangan jatuh yang proporsional dengan panjang dan ukuran dari kabel tersebut, yang biasanya dinyatakan dalam ohm per 1.000 *feet*. Tegangan jatuh dapat dihitung dengan menggunakan hukum Ohm [3] :

   $$E = I.R \qquad (2.3)$$

   Dimana : E = Tegangan resistor (Volt)
   I = Arus yang melalui konduktor (Amper)
   R = Resistansi konduktor (Ohm)

2.1.3 Rangkaian *Loop Powered Digital to Analog Converter* 4-20 mA[4][5]

Rangkaian *loop powered* DAC (*Loop Powered Digital to Analog Converter)* 4-20 mA merupakan rangkaian yang berfungsi sebagai pengubah sinyal digital menjadi sinyal analog yang berupa arus 4-20 mA . DAC 4-20 mA menggunakan AD421 yang merupakan IC DAC buatan *Analog Device* dengan *performance* yang baik. IC ini memiliki beberapa keunggulan, yaitu memiliki 16 bit akurasi untuk mengolah data digital menjadi analog, kompatibel dengan sinyal FSK HART serta mampu bekerja pada tegangan 3V, 3.3V serta 5V [4]. Rangkaian *Loop Powered* DAC 4-20 mA dengan menggunakan AD421 ditunjukkan pada Gambar 2.5. Pada Gambar 2.5 terdapat *jumper* J1,J2 dan J3 yang terhubung ke LV (*regulated voltage control*) dari AD421, fungsinya adalah sebagai pilihan untuk memilih tegangan kerja IC AD421. LV terhubung ke ground/COM berarti tegangan kerja yang dipilih adalah 5V, jika dihubungkan ke V_{CC} tegangan kerjanya 3V. Apabila LV dihubungkan ke V_{CC} melalui kapasitor C1 sebesar 0,01uF berarti tegangan kerjanya adalah 3,3V.



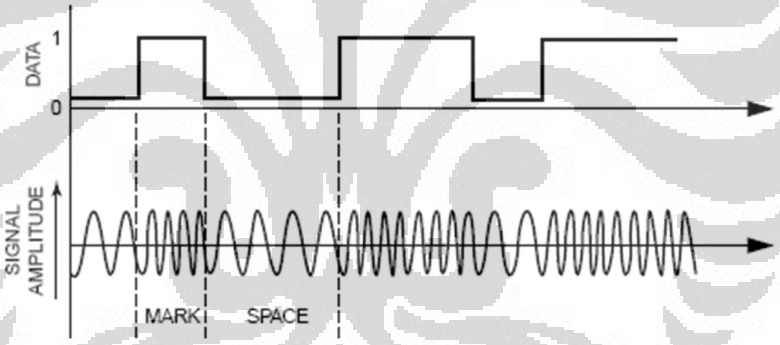
Gambar 2.5 Rangkaian *loop powered* DAC 4-20 mA dengan AD421[5]

Q1 berfungsi sebagai regulator tegangan dari *loop* + karena tegangan *loop* + disesuaikan hingga mampu bekerja pada tegangan maksimal $24V_{DC}$, sehingga Q1 berfungsi sebagai *limiter* arus maupun tegangan. Data digital yang dihasilkan mikrokontroler masuk dan dikontrol melalui pin LATCH, CLOCK dan DATA. Sinyal FSK yang berasal dari modem HART masuk ke pin C3.

2.2 HART (*Highway Addressable Remote Transducer*)

2.2.1 BFSK (*Binary Frequency Shift Keying*)

Binary Frequency Shift Keying adalah teknik modulasi digital pada spektrum frekuensi tinggi dimana informasi nilai digital direpresentasikan dalam bentuk frekuensi dengan cara sinyal digital digunakan untuk mengontrol modulasi frekuensi yang dihasilkan osilator[6]. Pada BFSK, nilai *binary* "1" disebut dengan "*mark*" dan *binary* "0" disebut dengan "*space*" [7][8], hubungan antara data digital dengan sinyal frekuensi yang ditransmisikan ditunjukkan pada Gambar 2.6.

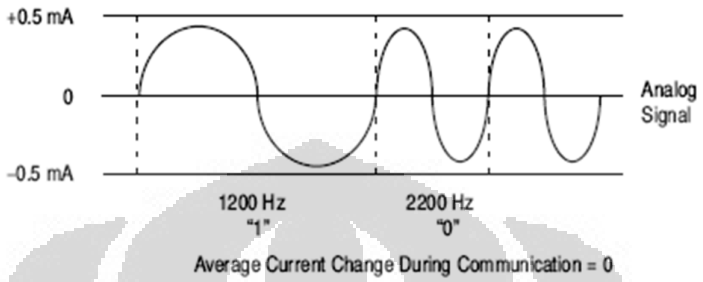


Gambar 2.6 Hubungan antara data digital dengan sinyal frekuensi [9]

2.2.2 Protokol HART

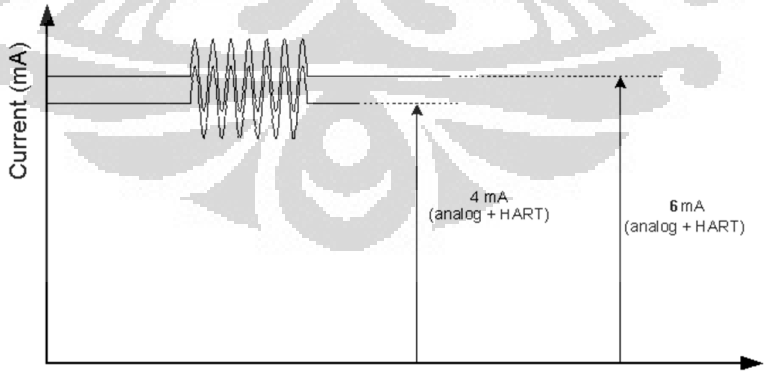
Protokol HART dibuat dengan menggunakan standar Bell 202 yang menggunakan prinsip BFSK (*Binary Frequency Shift Keying*). Pengiriman sinyal digital FSK dilakukan dengan menumpukkan pada sinyal analog 4-20 mA [1][10][11]. Sinyal digital FSK terdiri dari dua frekuensi, yaitu frekuensi 1200 Hz sebagai *binary* "1" dan frekuensi 2200 Hz sebagai *binary* "0" dengan sinyal amplitudo sebesar 0,5mA dengan nilai rata-rata sebesar 0 seperti yang ditunjukkan pada Gambar 2.7 [5][11]. Ini memungkinkan terjadinya komunikasi

dua arah antara perangkat instrument di lapangan terhadap sistem *monitoring* di daerah *control room* serta memungkinkan untuk menambahkan variabel tambahan yang ingin dimonitor.

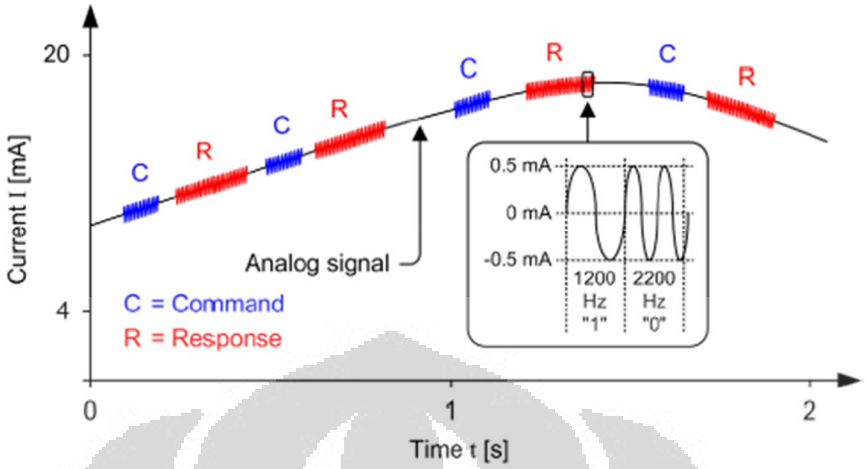


Gambar 2.7 Sinyal gelombang HART pada loop arus [5][11]

Posisi sinyal digital HART akan selalu mengikuti level besar arus analog. Jadi, misalnya ketika sinyal analog berada pada posisi 4 mA, maka sinyal HART akan berada pada posisi 4 mA atau jika secara tiba-tiba besar sinyal analog berada pada level 6 mA, maka posisi sinyal HART akan mengikuti. Untuk lebih jelasnya dapat dilihat pada ilustrasi pada Gambar 2.8. Begitu pula ketika terjadi proses pengiriman pesan yang berupa sinyal HART, sinyal digital akan mengikuti seiring dengan perubahan nilai sinyal analog yang dapat berubah sepanjang waktu seperti yang ditunjukkan pada ilustrasi Gambar 2.9.

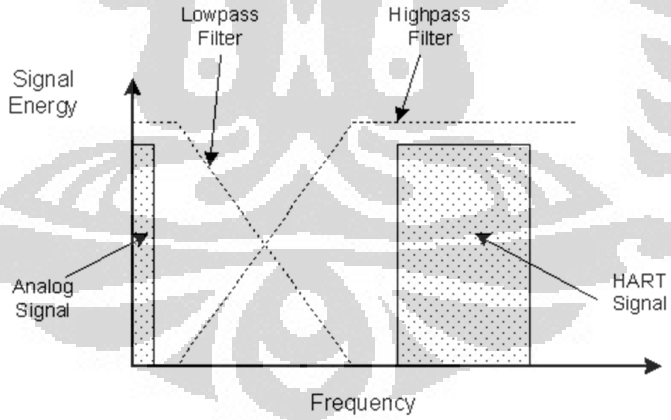


Gambar 2.8 Ilustrasi sinyal analog 4-20 mA dengan sinyal HART [2]



Gambar 2.9 Ilustrasi pengiriman pesan HART pada sinyal analog 4-20 mA [8]

Sinyal analog memiliki spektrum frekuensi 0 – 25 Hz. Sinyal ini lebih kecil dibandingkan sinyal digital HART yang menggunakan *Binary* FSK dengan spektrum frekuensi 500 Hz – 10 Khz [7]. Sehingga sinyal analog dan sinyal digital HART tidak mengganggu satu sama lain karena memiliki frekuensi yang berbeda [2], hal ini dapat diilustrasikan pada Gambar 2.10.



Gambar 2.10 Ilustrasi pemisahan spektrum frekuensi sinyal analog dan HART [2]

Protokol HART berkomunikasi dengan kecepatan 1200 bps secara *half duplex* tanpa mengganggu sinyal analog 4-20 mA yang memungkinkan *host*

(*master*) mendapatkan dua atau lebih data digital terbaru dari perangkat insrumen di lapangan (*field device*) [7][12]. Protokol HART menyediakan dua jenis komunikasi secara simultan, yaitu sinyal komunikasi analog 4-20 mA dan komunikasi digital. Sinyal analog 4-20 mA mengkomunikasikan nilai pengukuran utama dari variabel yang hendak diukur, sedangkan sinyal digital berisi informasi tambahan seperti status, diagnosa dan variabel-variabel tambahan lainnya dari suatu *field device*. Versi protokol HART saat ini adalah revisi 7.2.

2.2.3 Model OSI-7 *Layer* [13]

Lapisan protokol HART diimplementasikan berdasarkan lapisan aplikasi pada model OSI seperti yang ditunjukkan pada Gambar 2.11. Protokol HART hanya mengimplementasikan tiga lapisan saja, yaitu lapis 1, 2 dan 7. Lapis 1 merupakan *physical layer* yang menggunakan standar Bell 202 FSK yang berkomunikasi dengan kecepatan 1200 bps.

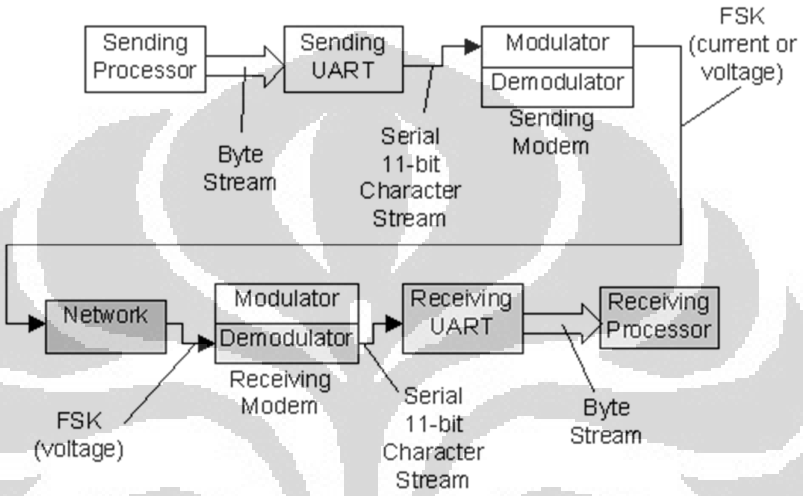
	OSI LAYER	FUNCTION	HART
7	APPLICATION	Provides the User with Network Capable Applications	Command Oriented, Predefined Data Types and Application Procedures
6	PRESENTATION	Converts Application Data Between Network and Local Machine Formats	
5	SESSION	Connections Management Services for Applications	
4	TRANSPORT	Provides Network independent, Transparent Message Transfer	
3	NETWORK	End to End Routing of Packets, Resolving Network Addresses	
2	DATA LINK	Establishes Data Packet Structure, Framing, Error detection, Bus Arbitration	A Binary, Byte Oriented, Token Passing, Master / Slave Protocol
1	PHYSICAL	Mechanical / Electrical Connection, Transmits Raw Bit Stream	Simultaneous Analog & Digital Signaling Normal 4-20mA Copper Wiring

Gambar 2.11 Model OSI 7-*Layer* protokol HART [13]

Lapisan 2, yaitu *Data link layer* yang mendefinisikan suatu *master* dan *slave* yang berkomunikasi satu sama lainnya. Pembentukan *frame* data ditentukan pada lapisan ini. Perangkat *slave* dapat dipasang hingga limabelas perangkat untuk konfigurasi jaringan *multidrop*, sedangkan *Layer* 7, yaitu A*pplication layer* terdiri dari tiga perintah-perintah HART (HART *Command*s), yaitu *Universal command*, *Common Practice* dan *Device specific*.

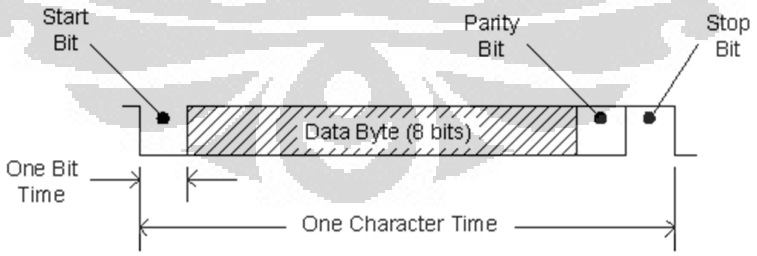
2.2.4 Pensinyalan [2]

Proses pengiriman sinyal HART dari prosesor pengirim ke prosesor penerima dapat dilihat pada Gambar 2.12. Prosesor mengirim 11 bit data UART yang kemudian dikirim ke modulator untuk dikonversi menjadi sinyal FSK. Sinyal FSK dikirim ke jaringan yang kemudian diubah kembali menjadi sinyal UART oleh *demodulator* agar bisa dibaca oleh prosesor.



Gambar 2.12 Alur diagram proses pengiriman sinyal HART [2]

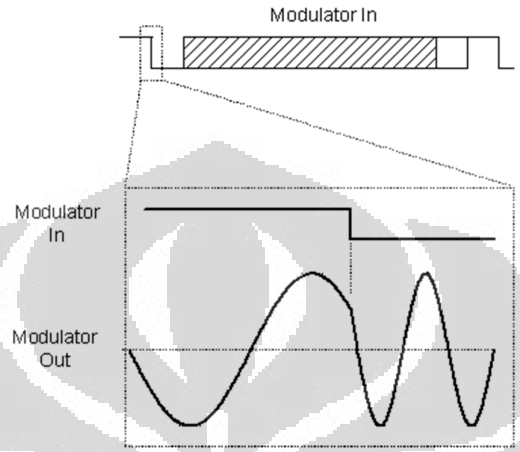
Data UART yang dikirim adalah 11 bit data, yang terdiri dari 2 bit *start/stop* sebagai sinkronisasi, 8 bit data dan 1 bit *parity* sebagai pendeteksi kesalahan. Lebih jelasnya dapat dilihat pada Gambar 2.13.



Gambar 2.13 Struktur data HART pada 1 karakter data [2]

Data UART yang dimodulasikan oleh *modulator* menjadi sinyal FSK adalah dengan mengubah data logika 1 menjadi frekuensi 1200 Hz dan logika 0

menjadi frekuensi 2200Hz. Modulasi sinyal ini terjadi terus menerus, yaitu perubahan frekuensi dari 1200 Hz ke 2200 Hz atau sebaliknya tidak mengalami jeda (berhenti). Sistem modulasi ini disebut "*Continous Phase*" *Frequency shift keying* (CPFSK). Ilustrasi ini dapat dilihat pada Gambar 2.14.



Gambar 2.14 Ilustrasi dari CPFSK [2]

2.2.5 HART PDU (*Frame* Data) [13]

Sebagai protokol, maka format data sudah ditentukan sesuai dengan ketentuan protokol tersebut. Setiap perintah (*command*) yang diterima maupun yang dikirim mengikuti aturan format data yang telah ditentukan. HART PDU terdiri dari delapan bagian utama, yaitu *Preamble, Delimiter, Address, Expansion Byte, Command, Byte Count, Data* dan *Check Byte* seperti ditunjukkan pada Gambar 2.15.

Preamble	*Delimiter*	*Address*	[*Expansion Byte*]	*Command*	*Byte Count*	[Data]	*Check Byte*

Gambar 2.15 *Frame Format* HART

2.2.2.4 HART *Command*

HART *Command* berfungsi sebagai perintah untuk melakukan beberapa macam eksekusi/mengolah data berdasarkan perintah yang dikirim. HART *Command* terdiri dari tiga jenis, yaitu *Universal Command*, *Common Practice* dan *Device Spesific* [7]. *Universal Command* merupakan perintah dasar yang umum

digunakan, setiap jenis instrumentasi yang menggunakan protokol HART, harus mampu mengenali jenis perintah ini. Semua jenis HART *command* sudah ditentukan oleh HCF (*HART Communication Foundation*). Pada skripsi ini, ada dua *command* yang digunakan, yaitu *command* 3 yang berfungsi untuk mengetahui besaran nilai *loop* arus dan nilai variabel utama, sedangkan *command* yang kedua adalah *command* 40 yang berfungsi untuk mengatur besaran *loop* arus.

Gambar 2.16 merupakan ***response data byte*** dari *command* 3, *request data byte* dikosongkan untuk *command* 3 ini [14], besarnya nilai *unit code* pada *byte* 4 ditentukan sesuai dengan jenis variabel data pada *byte* 5 sampai *byte* 8 [15]. Gambar 2.17 merupakan ***response dan request data byte*** untuk *command* 40 [16]. *Response data byte* merupakan format data respon yang dikirim *field device* ke *master*, sedangkan *request data byte* merupakan format data yang diminta *master* ke *field device*.

Byte ke-	0 - 3	4	5 - 8
	Loop Current (mA)	*Units Code* (suhu)	Suhu

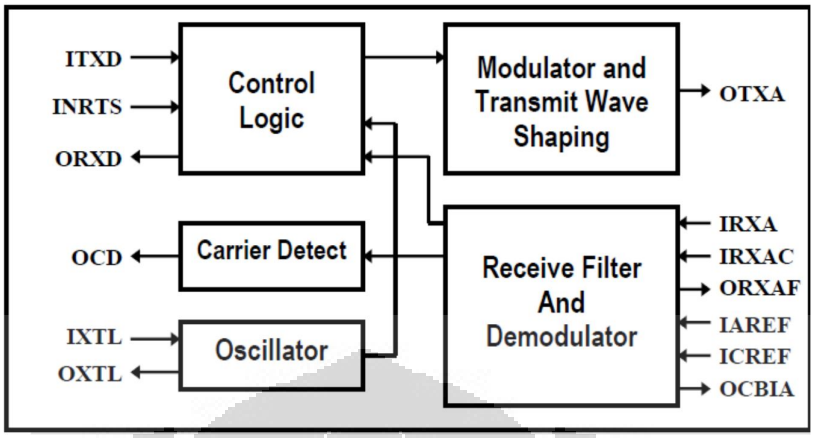
Gambar 2.16 Format ***response data byte*** dari *command* 3

Byte ke-	0 - 3
	Loop Current Level

Gambar 2.17 Format ***response data byte*** dari *command* 40

2.2.6 Rangkaian Modem HART [5][16]

Modem HART berperan sangat penting pada sistem berbasis protokol HART. Rangkaian ini difungsikan sebagai pengubah sinyal UART menjadi sinyal FSK. Salah satu modem HART yang kompatibel untuk diintegrasikan dengan AD421 adalah HT20C15. HT20C15 merupakan *single chip* CMOS buatan *SMAR Research*. Blok diagram dari HT20C15 ditunjukkan pada Gambar 2.18.



Gambar 2.18 Diagram blok HT20C15 [17]

Dengan penambahan rangkain pasif dan fungsi UART dari mikrokontroler, HT20C15 telah memenuhi fungsi kebutuhan antara lain modulasi, demodulasi, *filtering, carrier detect* dan pembentuk sinyal yang dikirim. Rangkaian modem HART dengan HT20C15 ditunjukkan pada Gambar 2.19. *Input* sinyal HART yang masuk melalui rangkaian *band pass filter*. *Band pass filter* ini tidak hanya berfungsi sebagai pengurang *noise* pada sinyal *input*, tapi bisa berfungsi sebagai penghilang frekuensi tinggi yang dapat mempengaruhi rangkaian. HT20C15 membutuhkan R_{BIAS} yang dihubungkan pada OCBIAS terhadap *ground* (V_{SS}). Besar R_{BIAS} ditentukan dengan Persamaan 2.4, dengan IAREF sebesar 1,235 V_{DC}, maka R_{BIAS} yang didapat sebesar 494 KΩ.

$$R_{BIAS} = \left(\frac{IAREF}{2{,}5\ \mu A}\right) \tag{2.4}$$

Nilai kapasitor C1 yang digunakan pada rangkaian ini adalah 6,2nF. Nilai ini didapat berdasarkan Persamaan 2.5[5].

$$Cc = \frac{20mV x C3}{V_H - 20mV} \tag{2.5}$$

dimana Cc = Kapasitor penghubung antara HT20C15 dengan AD421

V_H = Tegangan input sinyal

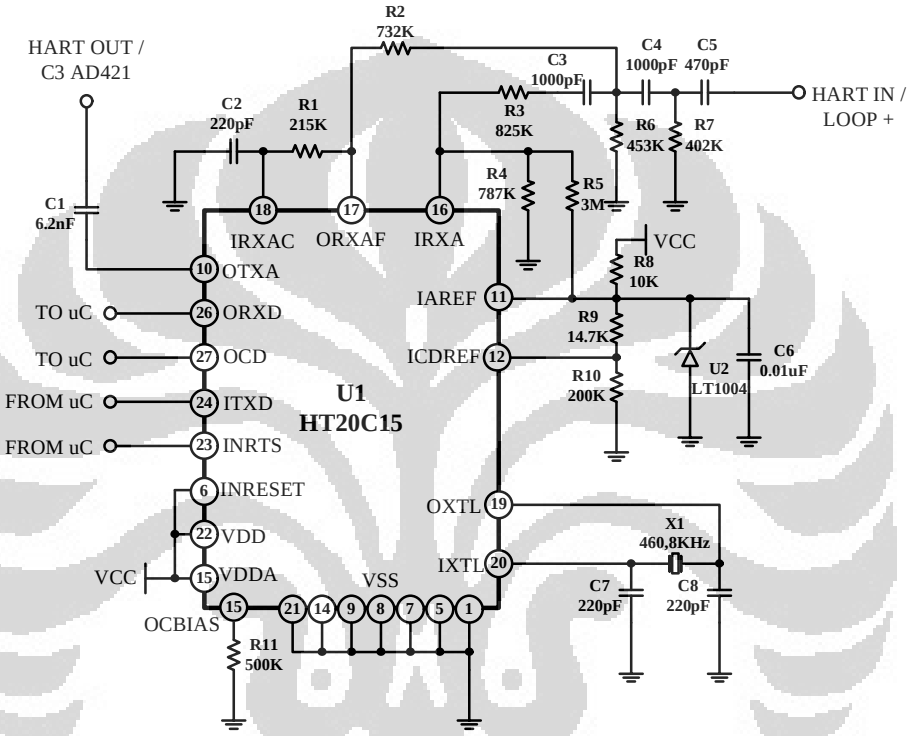
C3 = Nilai Kapasitor terpasang pada pin C3 AD421

Dari Persamaan 2.5, diketahui nilai C3 yang terpasang pada rangkaian AD421, yaitu 0,16μF. Besar V_H adalah 500 mVp-p, sehingga untuk nilai Cc bisa

didapat sebagai berikut.

$$Cc=\frac{(20x10^{-3})x(0,16x10^{-6})}{500x10^{-3}-20x10^{-3}} = 6,6x10^{-9}=6,6nF$$

Namun karena nilai kapasitor tidak tersedia di pasaran, maka dicari nilai yang mendekati, yaitu 6,2 nF.



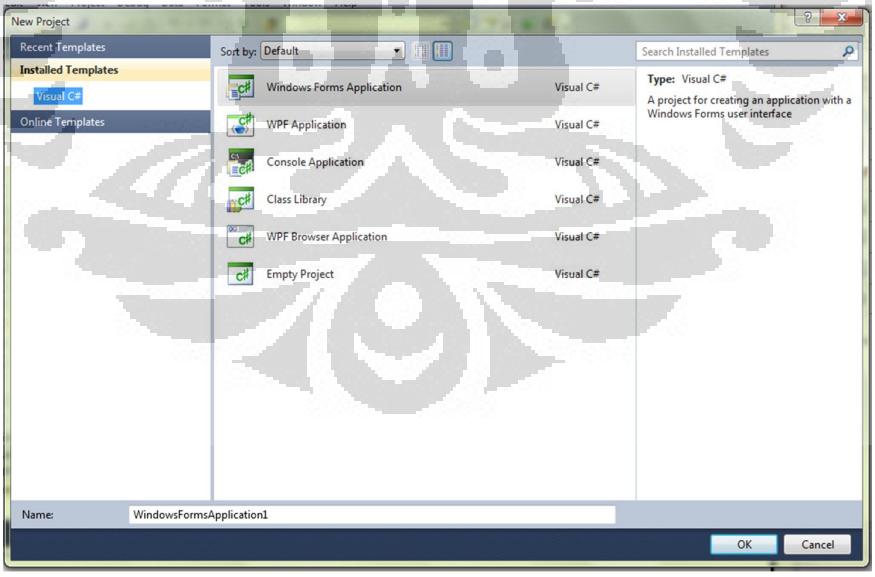
Gambar 2.19 Rangkaian modem HART dengan HT20C15 [5]

Sinyal data UART yang dikirim dari mikrokontroller masuk ke pin ITXD yang kemudian dimodulasikan menghasilkan sinyal FSK(*Frequency Shift Keying*) melalui *output* pin OTXA dengan nominal tegangan sebesar 500 mVp-p yang dikirim ke C3 AD421 melalui C1. Modulator bisa bekerja apabila INRTS di-set pada logika rendah. Demodulator menerima sinyal FSK yang telah di-*filter* melalui pin IRXA yang kemudian mendemodulasikan menjadi sinyal digital UART, OCD akan aktif tinggi apabila terdapat sinyal FSK yang masuk. *External oscillator* untuk menghasilkan *clock* yang dibutuhkan sebesar 460,8 KHz. IAREF merupakan pin tegangan referensi (V_{REF}) besar V_{REF} yang dibutuhkan adalah

1,235 V_{DC} untuk tegangan kerja (VCC) 3,3 V_{DC} dan V_{REF} 2,5 V_{DC} untuk tegangan V_{CC} sebesar 5 V_{DC}. Pada rancangan rangkain modem HART ini, besar V_{REF} yang akan digunakan adalah 1,235 V_{DC}. ICDREF sebagai tegangan referensi untuk menentukan besar tegangan nominal *detect carrier* sebesar (OCD) 100 mVp-p dengan selisih tegangan antara IAREF dan ICDREF adalah 0,08 V_{DC}.

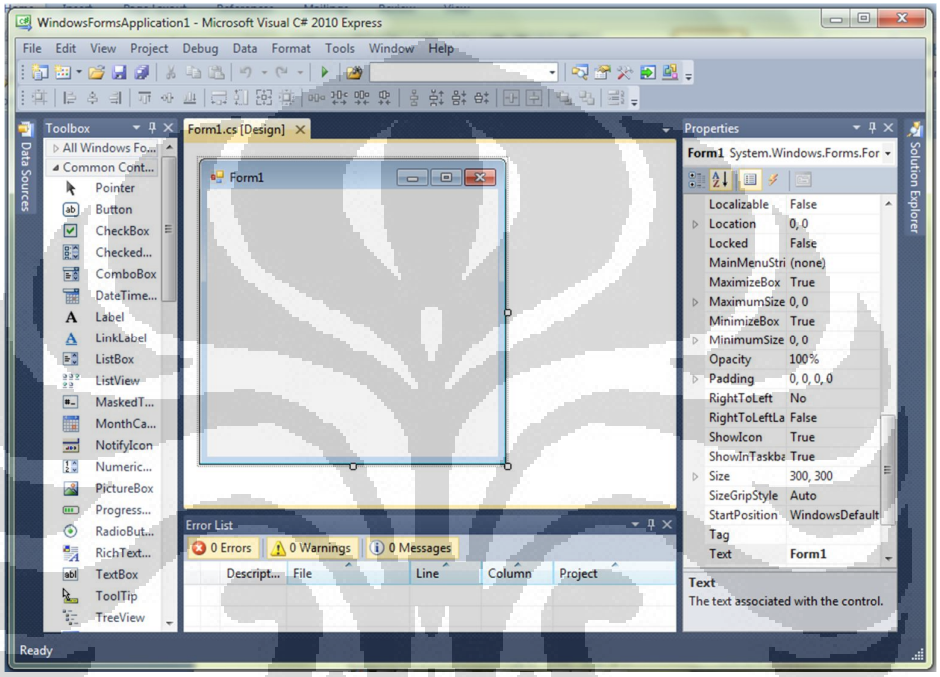
2.3 *Microsoft Visual C# Express* 2010[18]

Microsoft Visual C# merupakan perangkat lunak berdasarkan bahasa pemrograman C# dengan pengembangan aplikasi berbasis .NET *Framework*. Pemograman C# merupakan bahasa pemrograman berorientasi objek dan mempunyai kesamaan dengan C++, Java dan Visual Basic. Secara garis besar, lingkungan kerja *Visual C# Express* 2010 tidak berbeda dengan dengan *Visual C#* versi-versi sebelumnya yang menggunakan bahasa C#[18]. Pada skripsi ini, *Visual C# Express* 2010 digunakan untuk menerima data dari perangkat *hardware* (*field device*) dengan menggunakan GUI (*Graphical User Interface*) sebagai media komunikasi. Tampilan *Visual C# Express* 2010 ketika pertama kali dijalankan ditunjukkan pada Gambar 2.20.



Gambar 2.20 Tampilan awal pemilihan *project Visual C# Express* 2010

Untuk dapat berkomunikasi antara komputer dengan perangkat luar menggunakan komponen *serialport* yang telah tersedia pada *Visual C# Express* 2010. Ketika dibuat *project* baru, kita dapat mendesain tampilan program aplikasi sesuai dengan keinginan. Gambar 2.21 merupakan tampilan ketika *project* baru dijalankan.

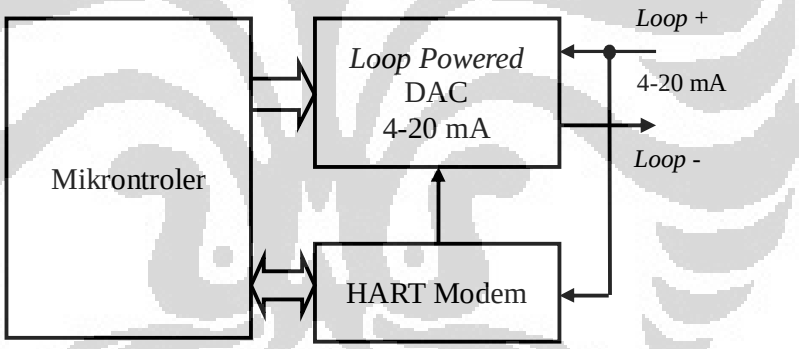


Gambar 2.21 Tampilan awal memulai *project* baru pada *Visual C# Express* 2010

BAB 3
PERANCANGAN DAN PEMBUATAN *MONITORING* SUHU BERBASIS PROTOKOL HART

3.1 *Loop Powered* DAC 4-20 mA Dengan HART Modem HT20C15[4][5]

Hardware dari perangkat *monitoring* suhu berbasis HART dibentuk berdasarkan *physical layer* FSK yang dintegrasikan dengan arus analog 4-20 mA. Rangkaian *loop powered* DAC 4-20 mA dengan HART modem yang digunakan berdasarkan *Application Note* AN-534 dari *Analog Device*[5] yang dibahas pada Bab 2, yaitu Gambar 2.5 untuk rangkaian *loop powered* DAC 4-20 mA dan Gambar 2.8 untuk rangkaian HART Modem. Kedua rangkaian dari *loop powered* DAC 4-20 mA dan HART modem HT20C15 ini diintegrasikan dengan mikrokontroler. Adapun blok diagramnya seperti pada Gambar 3.1.



Gambar 3.1 Blok diagram rancangan *hardware monitoring* suhu berbasis HART

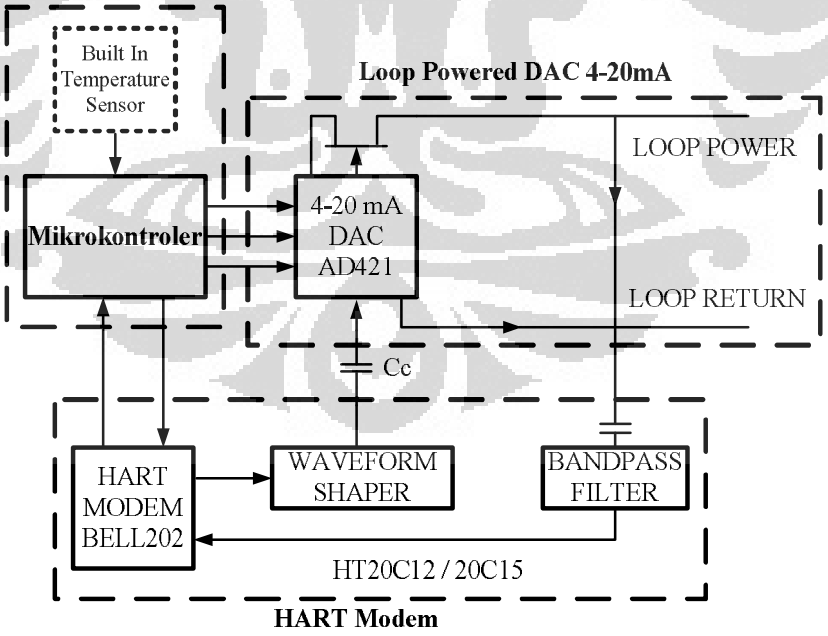
Mikrokontroler yang digunakan adalah C8051F350 buatan *Silicon Laboratories*. Mikrokontroler ini berfungsi sebagai pengolah data UART untuk mengirim sekaligus menerima data dari/ke HART modem. Pada mikrokontroler ini juga dimanfaatkan fitur *temperature sensor* yang terintegrasi di dalam mikrokontroler tersebut, sehingga hanya tinggal mengaktifkan *temperature sensor* tersebut melalui *register* ADC. HART modem berfungsi sebagai modulator dan demodulator sinyal FSK menjadi data UART dan sebaliknya UART menjadi sinyal FSK. *Loop powered* DAC 4-20 mA berfungsi sebagai pengubah data

digital yang dikirim mikrokontroler menjadi sinyal analog 4-20 mA, dengan sinyal analog 4-20 mA, sinyal FSK dari HART modem dapat diintegrasikan. Spesifikasi komponen yang digunakan dari *hardware* sistem berbasis protokol HART ditunjukkan pada Tabel 3.1.

Tabel 3.1 Spesifikasi *hardware* masing-masing blok instrumentasi berbasis protokol HART

Bagian	**Komponen**
Mikrokontroler	Silabs C8051F350 dengan *built in temperature sensor*
Loop Powered DAC 4-20 mA	Analog Device AD421
HART Modem	Smar Research HT20C15

Blok diagram lengkap dari sistem pemantau suhu yang akan dirancang dan dibuat adalah seperti pada Gambar 3.2, dimana terdapat 3 bagian yang menjadi satu kesatuan dari sebuah rangkaian modem HART, sedangkan Cc adalah kapasitor penghubung antara modem HART HT20C15 dengan AD421.



Gambar 3.2 Blok diagram rangkaian *hardware monitoring* suhu berbasis HART

Fitur dari mikrokontroler yang berperan penting dalam sistem *monitoring* suhu berbasis protokol HART adalah UART dan *Built in temperature sensor*, jadi setidaknya ada dua buah pin dari mikrokontroler yang difungsikan sebagai penerima dan pengirim data UART. Selain itu dibutuhkan tiga buah pin lagi sebagai *data, clock* dan *latch* ke AD421.

3.2 Perancangan dan Pembuatan *Software*

Perancangan *software* dari pemantau suhu yang akan dirancang mengikuti baku standar protokol HART revisi 7.2 yang dikeluarkan oleh HCF. Mengacu protokol HART revisi 7.2 tersebut, ada beberapa aturan yang perlu diperhatikan, yaitu :

1. Ada atau tidak adanya suatu pesan yang datang ditandai dengan pengiriman *preamble* 0xFF dua kali secara berturut.
2. Jika terjadi kesalahan komunikasi, maka *frame* pesan yang dikirim kembali. Hal ini terus dilakukan hingga tiga kali.

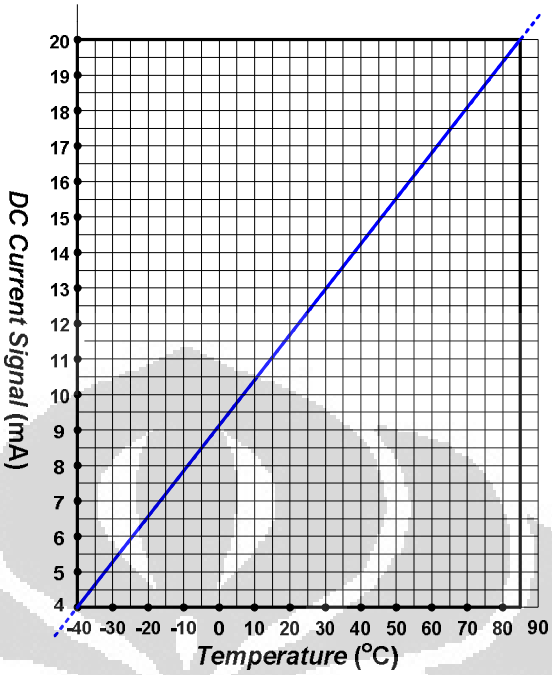
Perencanaan dalam pembuatan program pada mikrokontroler C8051F350 dengan menggunakan *Silicon Laboratories* IDE ver 4.20 yang menggunakan bahasa pemograman C dan membuat program aplikasi *monitoring* berbasis *Windows* dengan menggunakan *Microsoft Visual C# Express 2010*.

Pada Sistem *monitoring* suhu berbasis protokol HART, *command* yang digunakan ada dua, yaitu:

1. *Command* 3[14], yang digunakan untuk mengetahui besar nilai arus dan nilai suhu terukur. Besar data nilai arus yang terbaca merupakan nilai yang mewakili besarnya nilai suhu yang terukur. Besar suhu yang akan diukur disesuaikan dengan kemampuan pembacaan sensor suhu pada mikrokontroler *Silabs* C8051F350 yaitu -40 - 85^{o}C. Hubungan grafik besarnya arus dengan suhu ditunjukkan pada Gambar 3.3. Dengan menggunakan persamaan linier pada Persamaan 2.1, dapat dikonversikan nilai suhu terukur menjadi nilai arus atau sebaliknya, sehingga :

$$\text{Arus} = (\textit{Slope} \times \text{Suhu}) + \textit{Offset}$$

$$\text{Suhu} = \frac{\text{Arus} - \textit{Offset}}{\textit{Slope}}$$



Gambar 3.3 Hubungan arus 4-20 mA terhadap suhu -40 - 85°C

Nilai *Offset* dan *Slope* didapat dengan perhitungan sebagai berikut.

$$\text{Slope} = \frac{20\text{-}4}{85\text{-}(-40)} = \frac{16}{125} = 0{,}128$$

$$\text{Arus} = (0.128)\ \text{Suhu} + \textit{Offset}$$

$$4 = (0{,}128)(-40) + \textit{Offset}$$

$$\textit{Offset} = 4 + 5{,}12$$

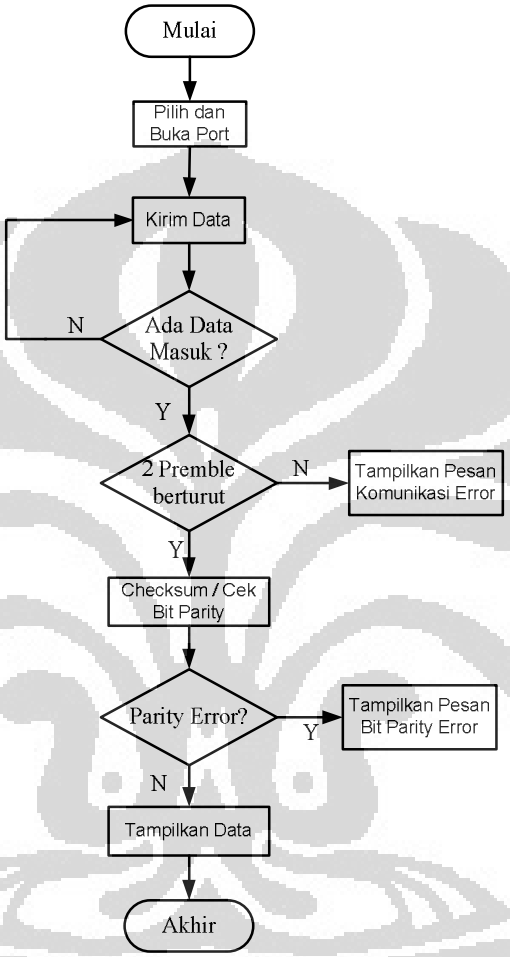
$$\textit{Offset} = 9{,}12$$

2. *Command* 40[16], yang digunakan untuk mengatur besar nilai arus pada *field device*.

3.2.1 Aplikasi *Monitoring* Menggunakan *Visual C# Express* 2010

Visual C# Express 2010 digunakan untuk membuat aplikasi *monitoring* suhu yang berfungsi sebagai indikator nilai suhu melalui GUI (*Graphical Universal Interface*) yang dibaca oleh *temperature sensor* yang terdapat pada mikrokontroler *Silabs* C8051F350. Pada Gambar 3.4 diperlihatkan diagram alir

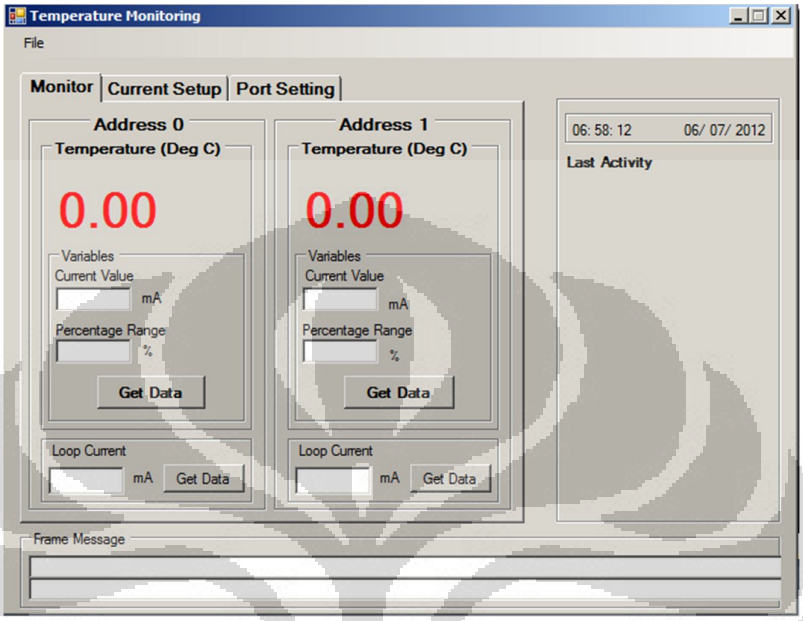
algoritma pemrograman aplikasi *monitoring* suhu menggunakan *Visual C# Express* 2010. Tujuan pemrograman ini adalah untuk menerima data pesan yang dikirim oleh *field device* melalui *serialport*.



Gambar 3.4 Diagram alir algoritma untuk program aplikasi *monitoring* suhu

Pada tahap awal pembuatan aplikasi ini, *project type* yang dipilih adalah *Windows Form Application*. Dengan pemilihan *project type* ini, nantinya aplikasi akan menampilkan suatu *form* seperti aplikasi *Windows*. Komponen utama dalam pembuatan program aplikasi ini adalah *SerialPort* yang berfungsi sebagai pengiriman data dan komponen *Timer* sebagai *event* untuk menerima data dari *serialport*. Gambar 3.5 adalah gambar tampilan utama dari aplikasi *monitoring* suhu. Untuk mendapatkan nilai suhu yang diukur, dilakukan dengan meng-klik

tombol “Get Data”. Pada *Tab Control* utama ini terdapat informasi berupa besar *loop current* dan besar persentasenya.



Gambar 3.5 Tampilan utama dari aplikasi *monitoring* suhu

Algoritma *pseudocode* program untuk “Get Data” untuk *long address* A1 A8 00 00 00 ditunjukkan pada Kode 3.1. Pada *pseudocode* program Kode 3.1 ini, dilakukan pengiriman data ke *field device* yang mana proses pertama kali adalah dengan membuka *port* terlebih dahulu, lalu meng-*enable*-kan RTS *port*. Setelah data (*frame message*) terkirim, RTS *port* dibuat dalam keadaan *disable* dan DTR *port* dalam keadaan *enable*, ini dimaksudkan agar *port* siap menerima respon data dari *field device* melalui *serialport*.

```
function GetData()
       OpenPort()
       RtsEnable=true
       KirimData(0x82,longaddress 0,0x03,0x00,data pesan)
       RtsEnable=false
       DtrEnable=true
efunction
```

Kode 3.1 *Pseudocode* program untuk mengambil data suhu yang terukur dengan long *address* A1 A8 00 00 00

Data pesan yang dikirim dibentuk berdasarkan format HART PDU. Nilai dari pesan yang dikirim pada bagian "Get Data" ditunjukkan pada Tabel 3.2. Pada Tabel 3.2, nilai data dikosongkan, ini karena jumlah *byte* dari data adalah 0 yang ditandai oleh nilai dari *byte count* sebesar 0.

Tabel 3.2 Nilai data pesan untuk "Get Data"

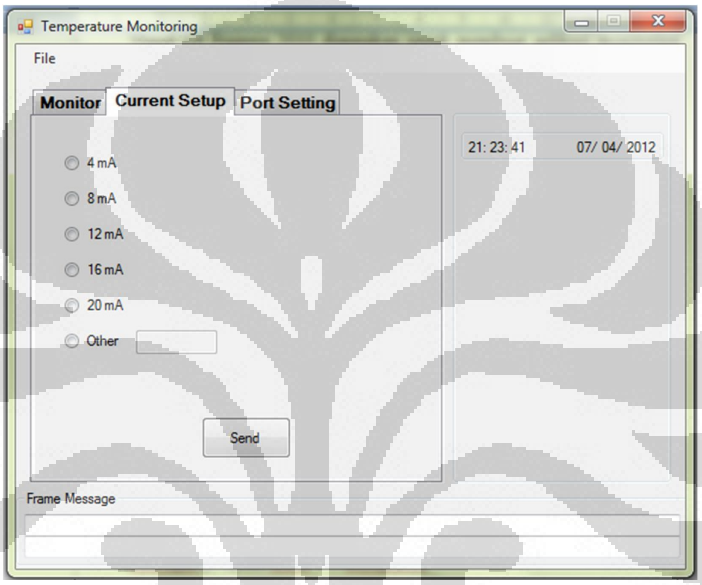
Delimiter	*Address*	*Command*	*Byte Count*	[Data]	*Check Byte*
82	A1 A8 00 00 00	02	00	-	89

Untuk "Get Data" dengan *long address* A1 A8 00 00 01 kurang lebih sama dengan *long address* A1 A8 00 00 01 hanya tinggal mengganti *address*-nya saja, Proses pengiriman data dilakukan dengan memanggil *sub rutin* "KirimData" yang ditunjukkan pada Kode 3.2.

```
function KirimData()
      for(i=0,i<=4,i++)
            message[i]=delimiter
      efor
      for(j=0,j<=panjang address,j++)
            Message[i++]=address
      efor
      message[i++]=command
      message[i++]=bytecount
      If (bytecount!=0)
            for(j=0,j<=panjang data pesan,j++)
                  message[i++]=data pesan
            efor
      eif
      checksum=0
      for(j=5,j<(bytecount+13),j++)
            checksum=checksum^message[j]
      efor
      message[i++]=checksum
      framelenght=bycount+14
      writePort(message,0,framelenght)
      delay(180ms)
      displaydata()
efunction
```

Kode 3.2 *Pseudocode* program untuk mengirim data ke *field device* melalui *serialport*

Gambar 3.6 merupakan tampilan dari *Tab Control* "Current Setup" yang digunakan untuk mengatur besar nilai arus *loop* yang diinginkan. Kode 3.3 merupakan *pseudocode* program untuk mengatur besar nilai arus *loop* dari *field device*. Proses kerja dari program ini hampir sama dengan proses pengambilan data pada Kode 3.1 dengan membuka *port* terlebih dahulu lalu meng-*disable*-kan RTS dan meng-*enable*-kan DTR.



Gambar 3.6 Tampilan *Tab Control* "Current Setup"

```
function SendCurrent()
      if(RB4mA checked)
            RtsEnable=true
            CurrentFloat(4)
            KirimData(0x82,longaddress 0,0x28,0x04,data pesan)
            RtsEnable=false
            DtrEnable=true
      eif
      if(RBother checked)
            read(x)
            RtsEnable=true
            CurrentFloat(x)
            KirimData(0x82,longaddress,0x28,0x04,data pesan)
            RtsEnable=false
            DtrEnable=true
      eif
efunction
```

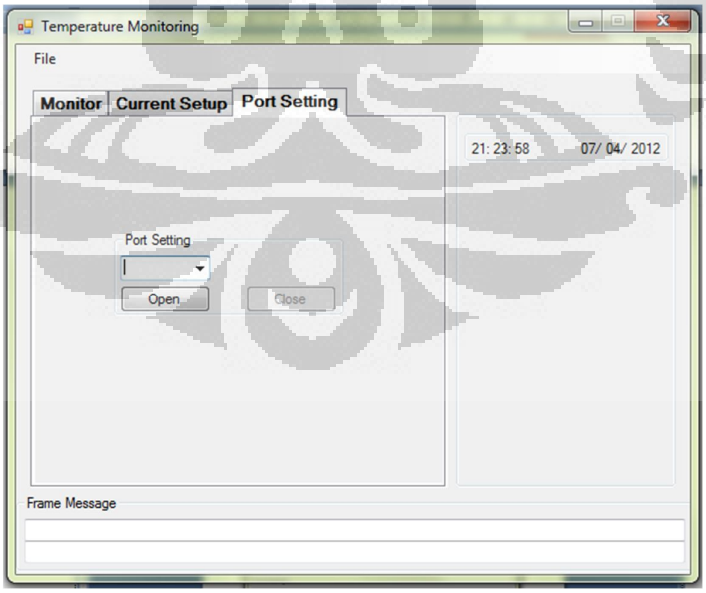
Kode 3.3 *Pseudocode* program untuk mengatur nilai arus *loop* pada *field device*

Nilai dari pesan yang dikirim pada bagian “Current Setup” ditunjukkan pada Tabel 3.3. Besarnya nilai data, *address* dan *check byte* yang dikirim tergantung dari nilai arus yang di-*set* dan *address* yang dituju. Nilai data merupakan nilai hexa 4 *byte* dari nilai arus yang bertipe data *float.*

Tabel 3.3 Nilai data pesan untuk “Current Setup”

Delimiter	***Address***	***Command***	***Byte Count***	**[Data]**	***Check Byte***	
82	A1 A8 00 00 00	28	04	40 80 00 00	67	4 mA
82	A1 A8 00 00 00	28	04	41 00 00 00	E6	8 mA
82	A1 A8 00 00 00	28	04	41 40 00 00	A6	12 mA
82	A1 A8 00 00 00	28	04	41 80 00 00	66	16 mA
82	A1 A8 00 00 00	28	04	41 A0 00 00	46	20 mA

Untuk memilih dan membuka COM *port* yang tersedia, dilakukan pada *Tab Control* “Port Setting” yang ditunjukkan seperti Gambar 3.7. *Pseudocode* program untuk memilih dan membuka COM *port* pada *form* ini ditunjukkan pada Kode 3.4.



Gambar 3.7 Tampilan *Tab Control* “Port Setting”

```
function PortSetting()
      read(port)
      function open()
            read(portname)
            OpenPort()
            if(port open)
                  openEnabled=false
                  closeEnabled=true
            eif
      efunction
      function close()
            if(port open)
                  openEnabled=true
                  closeEnabled=false
            eif
      efunction
efunction
```

Kode 3.4 *Pseudocode* program memilih dan membuka COM *port* yang tersedia

Untuk menerima data dari *serialport*, dibuat dengan menggunakan komponen *Timer*. Dengan mengaktifkan *event* pada *Timer* ini, data yang terdapat pada *serialport* bisa langsung dibaca dengan menggunakan perintah "port.Read(message_in, 0, bytes);". "message_in" merupakan variabel tempat disimpannya data yang dibaca dari *serialport* dalam bentuk data *array*, "0" adalah nilai indeks dari *array* yang akan dibaca pertama kali dan sedangkan "bytes" merupakan panjang dari data yang terdapat pada *serialport*. *Pseudocode* dari program membaca data pada *serialport* ditunjukkan pada Kode 3.5.

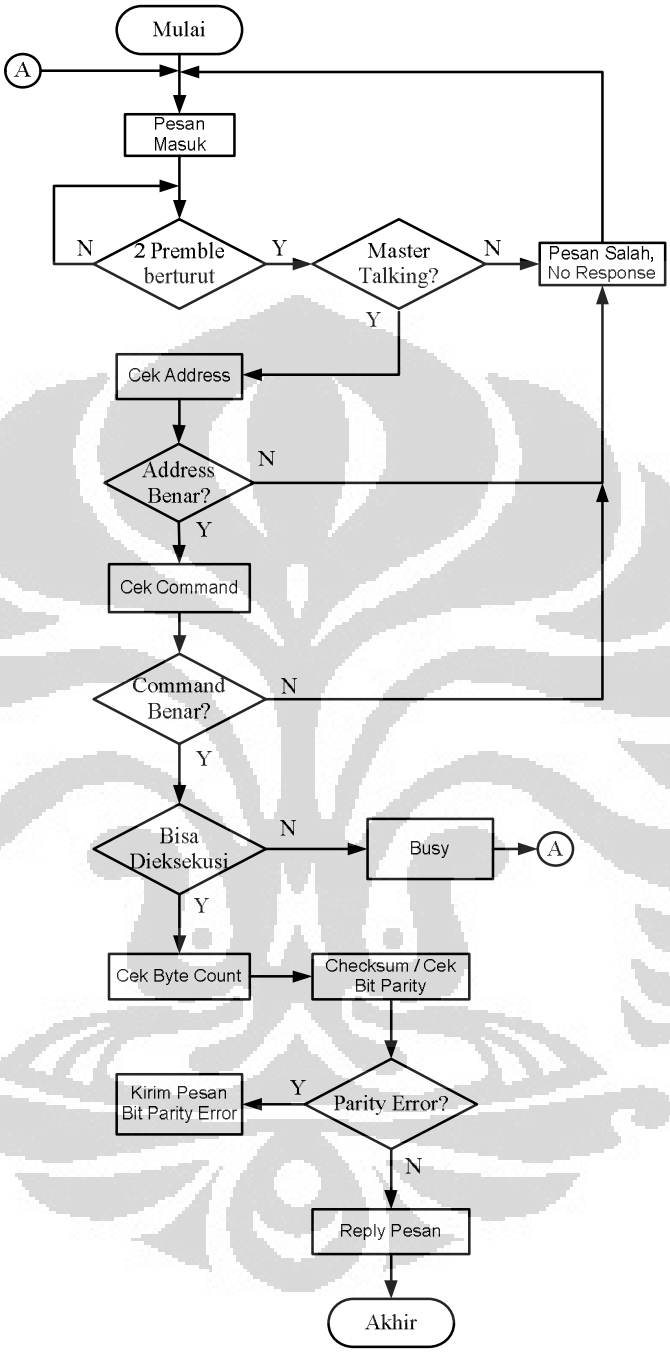
```
function Data Received()
      read(panjang data)
      if(panjang data!=0)
            readPort(message in,0,panjang data)
      eif
efunction
```

Kode 3.5 *Pseudocode* program menerima data *serialport*

3.2.2 Pemrograman *Field Device* Dengan *Silicon Laboratories* IDE

Pemrograman *field device* yang dilakukan adalah dengan memprogram mikrokontroler *Silabs* C8051F350. Hal utama dalam pemrograman mikrokontroler ini adalah mengaktifkan fitur *built in temperature sensor* dengan

memberikan nilai register ADCMUX sebesar 0xFF.



Gambar 3.8 Diagram alir untuk respon dari suatu *field device*

Gambar 3.8 merupakan diagram alir dari algoritma respon *field device*. Jika ada pesan yang datang, maka akan ditandai dengan diterimanya minimal dua

buah *preamble* secara berturut-turut. Kemudian sistem mengidentifikasi apakah *master* yang mengirim pesan atau bukan. Jika bukan, maka *field device* tidak merespon dan pengiriman pesan diulang kembali. Setelah diidentifikasi bahwa *master* yang mengirim pesan, maka sistem memeriksa alamat (*address*) yang dikirim sesuai dengan *address field device* tersebut. Jika *address* tidak sesuai, maka sistem tidak akan merespon dan jika *address*-nya sesuai langkah selanjutnya adalah mengetahui jenis perintah (*command*) yang diminta oleh *master*.

Apabila *field device* sibuk, maka *command* yang diminta *master* tidak akan di-eksekusi dan *field device* akan memberitahu bahwa sistem sibuk, jika *field device* siap meng-eksekusi *command* langkah selanjutnya adalah memeriksa *parity* bit keseluruhan pesan yang diterima, pesan yang diterima tidak boleh mengalami kesalahan. Apabila terdapat kesalahan, maka *field device* akan mengirim pesan bahwa bit *parity check* terdapat kesalahan. Jika bit *parity check* sukses, maka *field device* merespon pesan yang berisi informasi sesuai dengan jenis *command* yang dieksekusi.

HART PDU untuk *field device* merupakan pesan balasan (*return message*) yang disampaikan oleh *master*. Pesan yang dibalas sesuai dengan *command* yang diminta oleh *master*. Karena pada sistem *monitoring* suhu ini hanya dua *command* yang dibuat, maka *field device* juga hanya membalas kedua *command* tersebut.

Untuk *command* 3, pesan balasan ditunjukkan pada Tabel 3.4. Untuk *command* ini, besarnya nilai data dan *check byte* tergantung dari besarnya nilai suhu yang terukur. Data terdiri dari dua tipe data, yaitu data arus dan data suhu.

Tabel 3.4 Pesan balasan untuk *command* 3

Delimiter	***Address***	***Command***	***Byte Count***	**[Data]**	***Check Byte***
86	A1 A8 00 00 00	03	0B	[arus] [20] [suhu]	*Check byte*

Data arus merupakan representasi data dari nilai pengukuran suhu yang didapat dengan nilai *slope* dan *offset* yang telah ditentukan sebelumnya berdasarkan hubungan persamaan garis linier. Untuk mendapatkan nilai suhu maupun arus dilakukan dengan mengambil data ADC yang berada pada *register*

ADC0, dimana panjang nilainya sebesar 24 bit. *Pseudocode* program untuk mengambil data suhu ditunjukkan pada Kode 3.6.

```
function Get Suhu()
      for(;;)
            temp = ADC0H
            temp = (temp<<8) + ADC0M
            temp = (temp<<8) + ADC0L
            voltage = temp*3000.0/16777215
            voltage=voltage*1000
            suhu=(voltage-104934.52)/360.18
            arus=(suhu*0.128)+9.12;
      efor
efunction
```

Kode 3.6 *Pseudocode* program mengambil data suhu

Untuk *setting* arus atau *command* 40, balasan pesan ditunjukkan pada Tabel 3.5. Sama halnya seperti Tabel 3.3, besarnya nilai data yang dikirim tergantung dari nilai arus yang di-*set* dimana nilai data merupakan nilai hexa 4 *byte* dari nilai arus yang bertipe data *float*. Besarnya nilai *address* tergantung dengan address yang dituju. Nilai *check byte* merupakan nilai *X-OR* dari keseluruhan nilai data.

Tabel 3.5 Pesan balasan untuk *setting* arus

Delimiter	***Address***	***Command***	***Byte Count***	**[Data]**	***Check Byte***	
86	A1 A8 00 00 00	28	06	*00 40* 40 80 00 00	21	4 mA
86	A1 A8 00 00 00	28	06	*00 40* 41 00 00 00	A0	8 mA
86	A1 A8 00 00 00	28	06	*00 40* 41 40 00 00	E0	12 mA
86	A1 A8 00 00 00	28	06	*00 40* 41 80 00 00	20	16 mA
86	A1 A8 00 00 00	28	06	*00 40* 41 A0 00 00	00	20 mA

BAB 4
PENGUJIAN DAN ANALISIS

Pengujian dilakukan untuk mengetahui kinerja dari sistem dengan tujuan utama untuk mengetahui apakah *field device* mampu mengenali data dari *master* lalu merespon data sesuai dengan *request master* tanpa ada data yang cacat/hilang. Pengujian ini meliputi pengujian pengiriman dan penerimaan HART PDU. Adapun untuk melakukan pengujian diperlukan alat sebagai berikut :

1. HART Modem

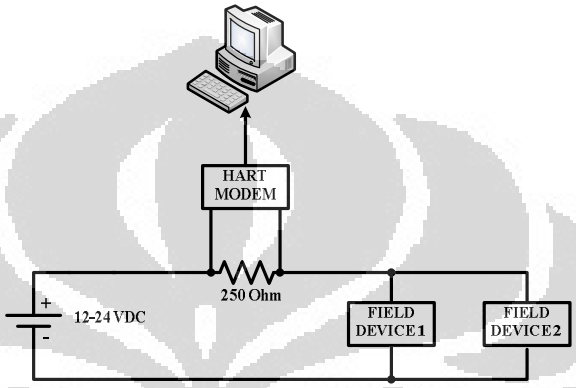
   HART modem difungsikan sebagai perangkat antar muka HART antara sistem yang akan diuji (*field device*) dengan komputer sebagai *master*. HART modem yang digunakan adalah HART modem buatan *microcyber*, HART modem ini mengubah sinyal HART menjadi UART agar dapat berkomunikasi dengan komputer melalui komunikasi serial.

Gambar 4.1 HART modem buatan *Microcyber*

2. Resistor 250 Ω.
3. 4-20 mA Display.
4. Display 4-20 mA ini prinsip nya sama dengan ampermeter yang digunakan untuk memonitor *loop* arus pada sistem.
5. Kabel data dengan panjang 100 meter dan 304,8 meter (1000 *feet*).
6. Program aplikasi *monitoring* suhu.

7. *Thermometer* Digital.

Dalam pengujian ini, sistem dirangkai dengan konfigurasi sistem *multidrop* seperti yang diperlihatkan pada Gambar 4.2. Besarnya tegangan *input* yang diberikan sebesar maksimal 12 V_{DC}. Sistem yang diuji menggunakan kabel dengan panjang 100 meter dan 304,8 meter (1000 *feet*).



Gambar 4.2 Konfigurasi sistem *multidrop*

4.1 Pengujian dan Analisis "Current Setup"

Pengujian "Current Setup" adalah dengan mengirim data arus sesuai dengan tombol *radio button* yang dipilih. Data arus yang diterima harus sama dengan besarnya data arus yang dipilih. Tabel 4.1 dan Tabel 4.2 menunjukkan data hasil pengukuran/pengujian "Current Setup". Tabel 4.1 dan Tabel 4.2 ini merupakan ringkasan dari data yang terdapat pada Lampiran 1.

Tabel 4.1 Data hasil pengujian/pengukuran "Current Setup" FD 0

Data Terkirim		**Respon Data**		
Nilai Float	Nilai Hexa	Nilai Float	Nilai Hexa	Tampilan Display 4-20mA
4 mA	40 80 00 00	4 mA	40 80 00 00	4,00 mA
8 mA	41 00 00 00	8 mA	41 00 00 00	8,00 mA
12 mA	41 40 00 00	12 mA	41 40 00 00	12,00 mA
16 mA	41 80 00 00	16 mA	41 80 00 00	16,00 mA
20 mA	41 A0 00 00	20 mA	41 A0 00 00	20,00 mA
10,56 mA	41 28 F5 C3	10,56 mA	41 28 F5 C3	10,56 mA

Tabel 4.2 Data hasil pengujian/pengukuran “Current Setup” FD 1

Data Terkirim		**Respon Data**		
Nilai Float	Nilai Hexa	Nilai Float	Nilai Hexa	Tampilan Display 4-20mA
4 mA	40 80 00 00	4 mA	40 80 00 00	4,00 mA
8 mA	41 00 00 00	8 mA	41 00 00 00	8,00 mA
12 mA	41 40 00 00	12 mA	41 40 00 00	12,00 mA
16 mA	41 80 00 00	16 mA	41 80 00 00	16,00 mA
20 mA	41 A0 00 00	20 mA	41 A0 00 00	20,00 mA
10,56 mA	41 28 F5 C3	10,56 mA	41 28 F5 C3	10,56 mA

Berdasarkan data pada Tabel 4.1 dan Tabel 4.2, *Field Device* mengirim respon data yang sama persis sesuai dengan data yang dikirim oleh *master*.

4.2 Hasil Pengukuran Suhu

Pengukuran suhu yang dilakukan adalah dengan mengambil data yang diterima oleh komputer yang dikirimkan oleh *field device* (FD). Nilai akurasi merupakan perhitungan selisih pengukuran suhu terhadap nilai suhu yang dianggap ideal, yaitu suhu pembacaan *thermometer*, dimana keakurasian merupakan derajat kedekatan dari pengukuran kuantitas terhadap nilai sebenarnya/yang dianggap benar.

Tabel 4.3 hingga Tabel 4.8 merupakan data hasil pengukuran suhu, untuk data yang lebih lengkap diperlihatkan pada Lampiran 2. Data hasil pengukuran *field device address* 0 dengan menggunakan panjang kabel 100 meter untuk V=12V_{DC} ditunjukkan pada Tabel 4.3. Nilai *error* yang tercatat pada data tabel merupakan nilai *error* mutlak.

Tabel 4.3 Data hasil pengukuran FD 0 panjang kabel = 100 meter, V_{DC}=12 Volt

No.	Suhu (^oC)		*Error* terhadap pembacaan *thermometer*		
	Pengukuran	Pembacaan *Thermometer*	*Error*	% *Error*	% Akurasi
1	25,74	25,60	0,14	0,55	99,45
2	25,70	25,60	0,10	0,39	99,61
3	25,52	25,60	0,08	0,31	99,69
4	25,56	25,60	0,04	0,16	99,84
5	25,48	25,60	0,12	0,47	99,53
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
27	25,58	25,60	0,02	0,08	99,92
28	25,57	25,60	0,03	0,12	99,88
29	25,60	25,60	0,00	0,00	100,00
30	25,62	25,60	0,02	0,08	99,92
	Rata-rata	**Rata-rata**		**Rata-rata % *error***	**Rata-rata % Akurasi**
	25,57	25,60		0,21	99,79

Tabel 4.4 menunjukkan data hasil pengukuran *field device address* 0 dengan menggunakan panjang kabel 100 meter untuk V=10$V_{DC.}$

Tabel 4.4 Data hasil pengukuran FD 0 panjang kabel = 100 meter, V_{DC}=10 Volt

No.	Suhu (^oC)		*Error* terhadap pembacaan *thermometer*		
	Pengukuran	Pembacaan *Thermometer*	*Error*	% *Error*	% Akurasi
1	29,60	30,00	0,40	1,33	98,67
2	29,53	30,00	0,47	1,57	98,43
3	29,65	30,00	0,35	1,17	98,83
4	29,84	30,00	0,16	0,53	99,47
5	29,89	30,00	0,11	0,37	99,63
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
27	30,50	30,50	0,00	0,00	100,00
28	30,27	30,30	0,03	0,10	99,90
29	30,36	30,30	0,06	0,20	99,80
30	30,33	30,30	0,03	0,10	99,90
	Rata-rata	**Rata-rata**		**Rata-rata % *error***	**Rata-rata % Akurasi**
	30,06	30,10		0,44	99,56

Data hasil pengukuran *field device address* 0 dengan menggunakan panjang kabel 304,8 meter/1000 *feet* ditunjukkan pada Tabel 4.5 untuk V=12V_{DC} dan Tabel 4.6 untuk V=10V_{DC}.

Tabel 4.5 Data hasil pengukuran FD 0 panjang kabel = 1000 *feet*, V_{DC}=12 Volt

No.	Suhu (^oC)		*Error* terhadap pembacaan *thermometer*		
	Pengukuran	Pembacaan *Thermometer*	*Error*	% *Error*	% Akurasi
1	27,76	28,00	0,24	0,86	99,14
2	27,90	28,00	0,10	0,36	99,64
3	27,95	28,00	0,05	0,18	99,82
4	27,97	28,00	0,03	0,11	99,89
5	28,00	28,00	0,00	0,00	100,00
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
27	28,43	28,40	0,03	0,11	99,89
28	28,5	28,40	0,10	0,35	99,65
29	28,54	28,40	0,14	0,49	99,51
30	28,58	28,40	0,18	0,63	99,37
	Rata-rata	**Rata-rata**		**Rata-rata % *error***	**Rata-rata % Akurasi**
	28,03	28,03		0,31	99,69

Tabel 4.6 Data hasil pengukuran FD 0 panjang kabel = 1000 *feet*,V_{DC}=10 Volt

No.	Suhu (^oC)		*Error* terhadap pembacaan *thermometer*		
	Pengukuran	Pembacaan *Thermometer*	*Error*	% *Error*	% AKurasi
1	30,89	31,1	0,21	0,68	99,32
2	30,96	31,4	0,44	1,42	98,58
3	31,01	31,5	0,49	1,58	98,42
4	31,01	31,5	0,49	1,58	98,42
5	31,01	31,5	0,49	1,58	98,42
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
27	32,00	32,0	0,00	0,00	100,00
28	32,01	32,0	0,01	0,03	99,97
29	32,01	32,0	0,01	0,03	99,97
30	32,04	32,0	0,04	0,12	99,88
	Rata-rata	**Rata-rata**		**Rata-rata % *error***	**Rata-rata % Akurasi**
	31,46	31,94		1,59	98,41

Data hasil pengukuran *field device address* 1 dengan menggunakan panjang kabel 100 meter ditunjukkan pada Tabel 4.7 untuk $V=12V_{DC}$ dan Tabel 4.8 untuk $V=10V_{DC}$.

Tabel 4.7 Data hasil pengukuran FD 1 panjang kabel = 100 meter,V_{DC}=12 Volt

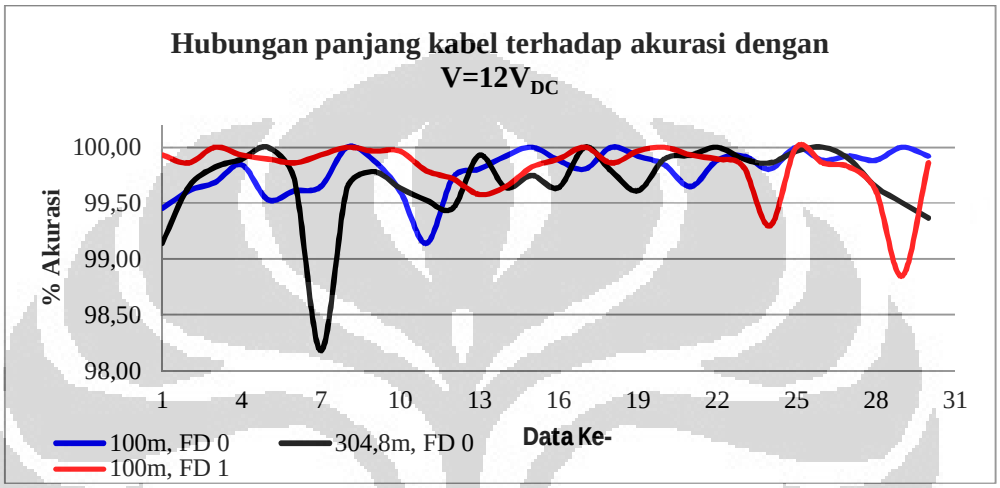
No.	**Suhu (°C)**		***Error* terhadap pembacaan *thermometer***		
	Pengukuran	Pembacaan *Thermometer*	*Error*	% *Error*	% Akurasi
1	28,62	28,60	0,02	0,07	99,93
2	28,64	28,60	0,04	0,14	99,86
3	28,60	28,60	0,00	0,00	100,00
4	28,58	28,60	0,02	0,07	99,93
5	28,57	28,60	0,03	0,10	99,90
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
27	28,55	28,60	0,05	0,17	99,83
28	28,49	28,60	0,11	0,38	99,62
29	28,33	28,66	0,33	1,15	98,85
30	28,60	28,64	0,04	0,14	99,86
	Rata-rata	**Rata-rata**		**Rata-rata % *error***	**Rata-rata % Akurasi**
	28,52	28,51		0,18	99,82

Tabel 4.8 Data hasil pengukuran FD 1 panjang kabel = 100 meter,V_{DC}=10 Volt

No.	**Suhu (°C)**		***Error* terhadap pembacaan *thermometer***		
	Pengukuran	Pembacaan *Thermometer*	*Error*	% *Error*	% Akurasi
1	32,14	32,5	0,36	1,11	98,89
2	32,20	32,5	0,30	0,92	99,08
3	32,26	32,5	0,24	0,74	99,26
4	32,27	32,7	0,43	1,31	98,69
5	32,33	32,7	0,37	1,13	98,87
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
27	29,78	29,8	0,02	0,07	99,93
28	29,66	29,6	0,06	0,20	99,80
29	29,62	29,6	0,02	0,07	99,93
30	30,18	30,0	0,18	0,60	99,40
	Rata-rata	**Rata-rata**		**Rata-rata % *error***	**Rata-rata % Akurasi**
	31,69	32,22		1,67	98,33

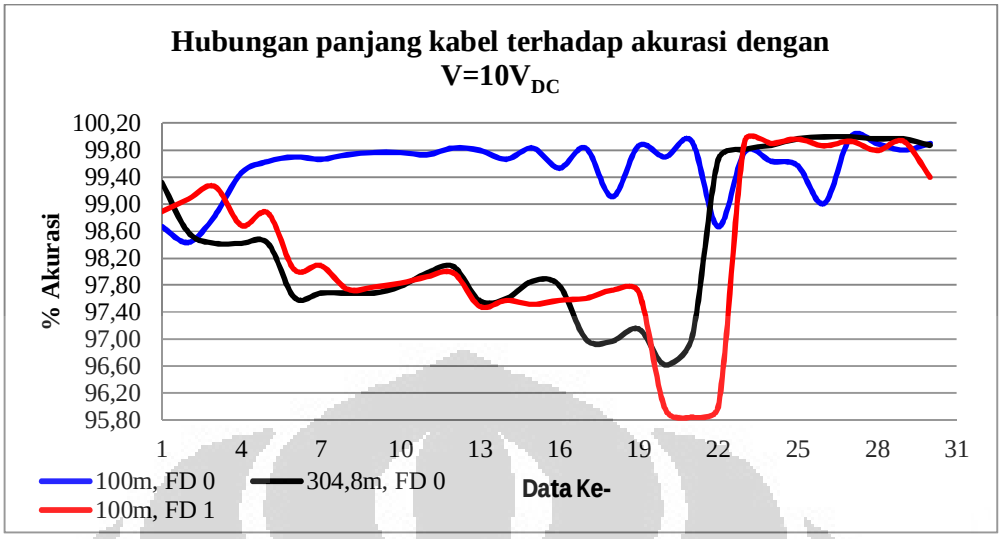
4.3 Analisa Pengaruh Panjang Kabel Terhadap Akurasi

Gambar 4.3 dan Gambar 4.4 merupakan analisa data berdasarkan data hasil pengukuran Tabel 4.3 hingga Tabel 4.8 (data lebih lengkap diperlihatkan pada Lampiran 2). Pada Gambar 4.3 dan Gambar 4.4 ini memperlihatkan pengaruh panjang kabel terhadap persen akurasi, dimana panjang kabel 100 meter memiliki persen akurasi yang lebih baik ketimbang kabel 304,8 meter/1000 *feet*.



Gambar 4.3 Grafik analisa pengaruh panjang kabel terhadap akurasi dengan tegangan *supply* 12 V_{DC}

Gambar 4.3 menunjukkan pengaruh panjang kabel terhadap akurasi dengan menggunakan tergangan *supply* sebesar 12V_{DC}. Berdasarkan Gambar 4.3, menunjukkan bahwa performa panjang kabel 100 meter cenderung lebih baik ketimbang kabel 304,8 meter/1000 *feet*, dimana untuk kabel 100 meter, FD 0 memperoleh minimum % akurasi sebesar 99,14% , sedangkan FD 1 sebesar 98,85%. Panjang Kabel 304,8 meter/1000 *feet* menghasilkan 98,18% akurasi yang diperoleh oleh FD 0.

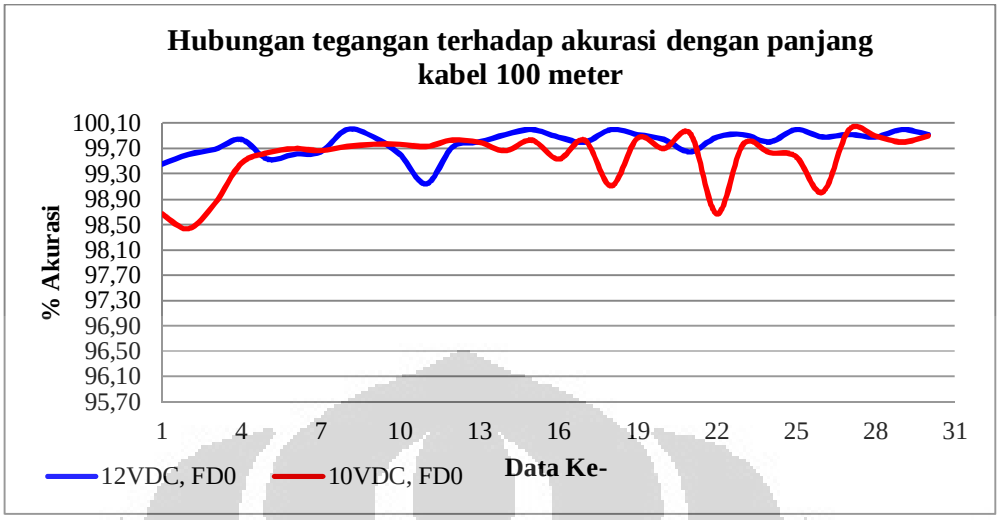


Gambar 4.4 Grafik analisa pengaruh panjang kabel terhadap akurasi dengan tegangan *supply* 10 V_{DC}

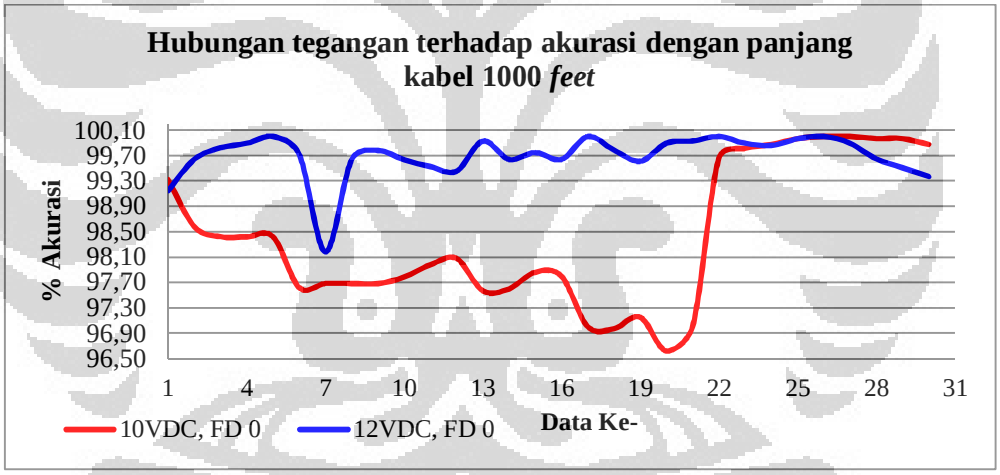
Gambar 4.4 menunjukkan pengaruh panjang kabel terhadap akurasi dengan menggunakan tergangan *supply* sebesar 10V_{DC}. Pada Gambar 4.4 ini menunjukkan bahwa performa FD 0 dengan panjang kabel 100 meter lebih baik ketimbang 304,8 meter/1000 *feet* dan FD 1 dengan panjang kabel 100 meter dengan minimum akurasi mencapai 98,43%. FD 0 dengan kabel 304,8 meter /1000 *feet* memilki performa yang relatif sama dengan FD 1 dengan panjang kabel 100 meter. FD 0 dengan kabel 304,8 meter/1000 *feet* memiliki minimum akurasi sebesar 96,62% sedangkan FD 1 dengan panjang kabel 100 meter sebesar 95,84%.

4.4 Analisa Pengaruh Tegangan *Supply* Terhadap Akurasi

Untuk memudahkan analisa pengaruh tegangan terhadap akurasi, dibuat grafik secara terpisah untuk masing-masing panjang kabel sesuai dengan *field address* masing-masing. Pada grafik Gambar 4.5 menujukkan performa FD 0 dengan panjang kabel 100 meter yang menunjukkan bahwa tegangan *supply* 12V_{DC} memiliki performa yang lebih baik ketimbang 10V_{DC}, begitu pula dengan panjang kabel 1000 *feet*, tegangan *supply* 12V_{DC} jauh lebih baik persen akurasinya ketimbang tegangan *supply* 10V_{DC} seperti yang diperlihatkan pada grafik Gambar 4.6.

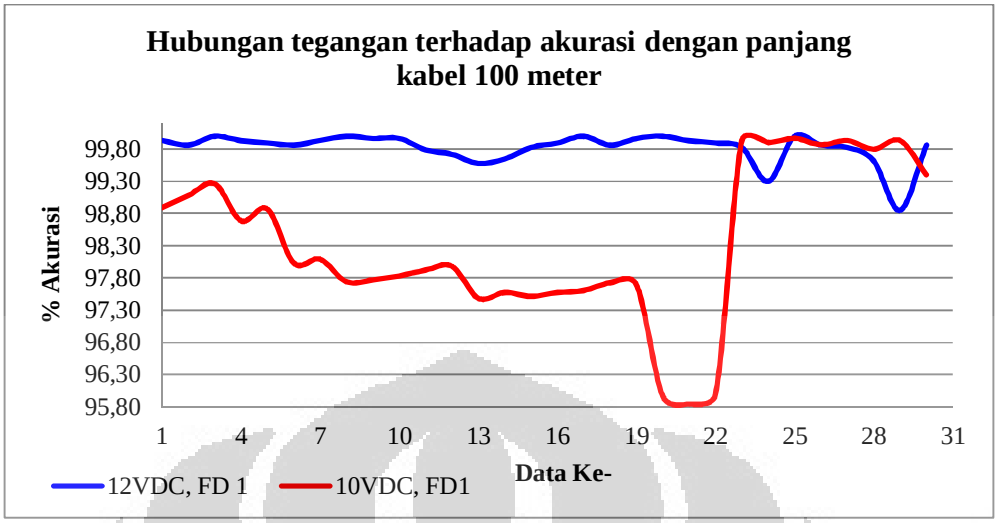


Gambar 4.5 Grafik analisa pengaruh tegangan *supply* terhadap akurasi dengan panjang kabel 100 meter pada FD 0



Gambar 4.6 Grafik analisa pengaruh tegangan *supply* terhadap akurasi dengan panjang kabel 1000 *feet* pada FD 0

Gambar 4.7 menunjukkan persen akurasi FD 1 dengan panjang kabel 100 meter. Dari Gambar 4.7 ini, bisa dilihat bahwa persen akurasi dengan tegangan *supply* $12V_{DC}$ lebih baik ketimbang $10V_{DC}$.



Gambar 4.7. Grafik analisa pengaruh tegangan *supply* terhadap akurasi dengan panjang kabel 100 meter pada FD 1

Tabel 4.9 merupakan rangkuman dari data Tabel 4.3 hingga Tabel 4.8. Berdasarkan Tabel 4.9 ini, akurasi rata-rata terbaik sebesar 99,79% dimiliki oleh FD 0 dengan panjang kabel 100 meter dengan tegangan *supply* sebesar $12V_{DC}$. Akurasi rata-rata terburuk diperoleh FD 1 dengan kabel 100 meter dengan tegangan *supply* sebesar $10V_{DC}$ dengan akurasi rata-rata sebesar 98,33%.

Tabel 4.9. Data rangkuman hasil pengukuran suhu

	Rata-rata *thermometer*	**Rata-rata pengukuran**	**Rata-rata % *error***	**Rata-rata % Akurasi**
$12V_{DC}$, 100 m, *Field device* 0	25,60(°C)	25,57(°C)	0,21	99,79
$12V_{DC}$, 1000 *ft*, *Field device* 0	28,03(°C)	28,03(°C)	0,31	99,69
$10V_{DC}$, 100 m, *Field device* 0	30,10(°C)	30,06(°C)	0,44	99,56
10 V_{DC}, 1000 *ft*, *Field device* 0	31,94(°C)	31,46(°C)	1,59	98,41
$12V_{DC}$, 100 m *Field device* 0	28,51(°C)	28,52(°C)	0,18	99,82
$10V_{DC}$,100 m *Field device* 1	32,22(°C)	31,69(°C)	1,67	98,33

BAB 5
KESIMPULAN

Berdasarkan hasil pengujian dan analisis data yang dilakukan pada *monitoring* suhu berbasis protokol HART pada konfigurasi jaringan *multidrop*, dapat diambil beberapa kesimpulan sebagai berikut:

1. Besar tegangan *supply* memberikan pengaruh terhadap akurasi pembacaan suhu, dimana tegangan ***supply*** $12V_{DC}$ memiliki performa yang lebih baik ketimbang tegangan ***supply*** $10V_{DC}$.
2. Panjang kabel 100 meter memiliki persen akurasi yang lebih baik ketimbang panjang kabel 304,8 meter/1000 *feet*.
3. Dengan kabel panjang 100 meter, *field device address* 0 memiliki performa akurasi yang lebik baik ketimbang *field device address* 1 baik dengan tegangan *supply* $12V_{DC}$ maupun $10V_{DC}$.
4. Minimum akurasi pengukuran mencapai 96,62% untuk *field device address* 0 dan 95,84 untuk *address* 1.
5. Hasil Akurasi rata-rata pembacaan terbaik yang diperoleh *field device address* 0 sebesar 99,79% dengan panjang kabel 100 meter dan tegangan *supply* sebesar $12V_{DC}$. Sedangkan akurasi rata-rata pembacaan terbaik yang diperoleh *field device address* 1 sebesar 98,33% dengan panjang kabel 100 meter dan tegangan *supply* sebesar $10V_{DC}$.
6. Pengujian dengan memberikan *setup current* ke *field device* memberikan respon pesan data yang sama/sesuai dengan pesan yang dikirim oleh *master*.

DAFTAR REFERENSI

[1] Tony R. Kuphaldt. *Lesson In Industrial Instrumentation* (Ver.1.14). January 11. 2011.

[2] *About HART* (Revised 2-1-01). Analog Services,Inc. 31 Januari. 2011. <www.analogservice.com/about_part0.htm>.

[3] DMS-AN-20. *Application Note: 4- 20 mA Current Loop Primer*. Murata Power Solutions. 2009.

[4] AD421 Datasheet. *Loop Powered 4mA to 20mA AD421* (Rev. C). Analog Device. 2000.

[5] Albert O'Grady. *Application Note:Adding HART® Capability to the AD421, Loop-Powered 4 mA–20 mA DAC Using the 20C15 HART Modem*. AN-534. Analog Device. 1999.

[6] B. Carlson, Paul B. Crilly & Janet C. Rutledge. *Communication Systems : An Introduction to Signals and Noise in Electrical Communication* (4th Edition). McGraw-Hill. 2002.

[7] HCF_SPEC-054. *FSK Physical Layer Specification* (Revision 8.1). HART Communication Foundation. 24 August.1999.

[8] Milos P., Vladislav J. & Buldimir L.J. *Intelligent Transmitters of Pressure and Other Process Parameters*. Telfor Journal. 2009.

[9] Bob Watson. *FSK:Signal and Demodulation* (Vol.7, No.5, Rev.2001). Watkins-Jhonson Company. 1980.

[10] HCF_LIT-039. *HART Communication Application Guide* (Revision 1.0 Preliminary). HART Communication Foundation. March 25. 2010.

[11] Z. Mingru, Y. Wentang & Q. Xin. *The Development of Intelligent Pressure Transmitter Based on HART Protocol*. Nancang University. 2010.

[12] *How HART Works*. HCF. 28 Januari. 2011. <www.hartcomm.org/protocol/about/aboutprotocol_how.html>.

[13] HCF_SPEC-81. *Token Passing Data Link Layer* (Revision 8.2). HART Communication Foundation. 23 July. 2007

[14] HCF_SPEC-127. *Universal Command Specification* (Revision 127.2). HART Communication Foundation. 10 May. 2008.

[15] HCF_SPEC-183. *Common Tables Specification* (Revision 20.0). HART Communication Foundation. 3 March. 2009

[16] HCF_SPEC-151. *Common Practice Command Specification* (Revision 9.1). HART Communication Foundation. 21 May. 2008.

[17] *HART Modem HT20C15 Datasheet*. Smar Research.

[18] Wahana Komputer. *Microsoft Visual C# 2010*. Andi Offset. Yogyakarta. 2011.

LAMPIRAN 1

Data Hasil Pengujian "Current Setup"

Data hasil pengujian/pengukuran "Current Setup" FD 0

Data Terkirim		Respon Data		
Nilai Float	Nilai Hexa	Nilai Float	Nilai Hexa	Tampilan Display 4-20mA
4 mA	40 80 00 00	4 mA	40 80 00 00	4,00 mA
5 mA	40 A0 00 00	5 mA	40 A0 00 00	5,00 mA
6 mA	40 C0 00 00	6 mA	40 C0 00 00	6,00 mA
7 mA	40 E0 00 00	7 mA	40 E0 00 00	7,00 mA
8 mA	41 00 00 00	8 mA	41 00 00 00	8,00 mA
9 mA	41 10 00 00	9 mA	41 10 00 00	9,00 mA
10 mA	41 20 00 00	10 mA	41 20 00 00	10,00 mA
11 mA	41 30 00 00	11 mA	41 30 00 00	11,00 mA
12 mA	41 40 00 00	12 mA	41 40 00 00	12,00 mA
13 mA	41 50 00 00	13 mA	41 50 00 00	13,00 mA
14 mA	41 60 00 00	14 mA	41 60 00 00	14,00 mA
15 mA	41 70 00 00	15 mA	41 70 00 00	15,00 mA
16 mA	41 80 00 00	16 mA	41 80 00 00	16,00 mA
17 mA	41 88 00 00	17 mA	41 88 00 00	17,00 mA
18 mA	41 90 00 00	18 mA	41 90 00 00	18,00 mA
19 mA	41 98 00 00	19 mA	41 98 00 00	19,00 mA
20 mA	41 A0 00 00	20 mA	41 A0 00 00	20,00 mA
6,77 mA	40 D8 A3 D7	6,77 mA	40 D8 A3 D7	6,77 mA
10,56 mA	41 28 F5 C3	10,56 mA	41 28 F5 C3	10,56 mA
15,45 mA	41 77 33 33	15,45 mA	41 77 33	15,45 mA

(LANJUTAN)

Data hasil pengujian/pengukuran "Current Setup" FD 1

Data Terkirim		Respon Data		
Nilai Float	Nilai Hexa	Nilai Float	Nilai Hexa	Tampilan Display 4-20mA
4 mA	40 80 00 00	4 mA	40 80 00 00	4,00 mA
5 mA	40 A0 00 00	5 mA	40 A0 00 00	5,00 mA
6 mA	40 C0 00 00	6 mA	40 C0 00 00	6,00 mA
7 mA	40 E0 00 00	7 mA	40 E0 00 00	7,00 mA
8 mA	41 00 00 00	8 mA	41 00 00 00	8,00 mA
9 mA	41 10 00 00	9 mA	41 10 00 00	9,00 mA
10 mA	41 20 00 00	10 mA	41 20 00 00	10,00 mA
11 mA	41 30 00 00	11 mA	41 30 00 00	11,00 mA
12 mA	41 40 00 00	12 mA	41 40 00 00	12,00 mA
13 mA	41 50 00 00	13 mA	41 50 00 00	13,00 mA
14 mA	41 60 00 00	14 mA	41 60 00 00	14,00 mA
15 mA	41 70 00 00	15 mA	41 70 00 00	15,00 mA
16 mA	41 80 00 00	16 mA	41 80 00 00	16,00 mA
17 mA	41 88 00 00	17 mA	41 88 00 00	17,00 mA
18 mA	41 90 00 00	18 mA	41 90 00 00	18,00 mA
19 mA	41 98 00 00	19 mA	41 98 00 00	19,00 mA
20 mA	41 A0 00 00	20 mA	41 A0 00 00	20,00 mA
6,77 mA	40 D8 A3 D7	6,77 mA	40 D8 A3 D7	6,77 mA
10,56 mA	41 28 F5 C3	10,56 mA	41 28 F5 C3	10,56 mA
15,45 mA	41 77 33 33	15,45 mA	41 77 33	15,45 mA

LAMPIRAN 2

Data Hasil Pengukuran Suhu

Data hasil pengukuran FD 0 panjang kabel = 100 meter,V_{DC}=12 Volt

No.	Suhu (^oC)		*Error* terhadap pembacaan *thermometer*		
	Pengukuran	Pembacaan *Thermometer*	*Error*	% *Error*	% Akurasi
1	25,74	25,60	0,14	0,55	99,45
2	25,70	25,60	0,10	0,39	99,61
3	25,52	25,60	0,08	0,31	99,69
4	25,56	25,60	0,04	0,16	99,84
5	25,48	25,60	0,12	0,47	99,53
6	25,50	25,60	0,10	0,39	99,61
7	25,51	25,60	0,09	0,35	99,65
8	25,60	25,60	0,00	0,00	100,00
9	25,57	25,60	0,03	0,12	99,88
10	25,50	25,60	0,10	0,39	99,61
11	25,38	25,60	0,22	0,86	99,14
12	25,53	25,60	0,07	0,27	99,73
13	25,55	25,60	0,05	0,20	99,80
14	25,58	25,60	0,02	0,08	99,92
15	25,60	25,60	0,00	0,00	100,00
16	25,63	25,60	0,03	0,12	99,88
17	25,65	25,60	0,05	0,20	99,80
18	25,60	25,60	0,00	0,00	100,00
19	25,58	25,60	0,02	0,08	99,92
20	25,56	25,60	0,04	0,16	99,84
21	25,51	25,60	0,09	0,35	99,65
22	25,57	25,60	0,03	0,12	99,88
23	25,58	25,60	0,02	0,08	99,92
24	25,55	25,60	0,05	0,20	99,80
25	25,60	25,60	0,00	0,00	100,00
26	25,63	25,60	0,03	0,12	99,88
27	25,58	25,60	0,02	0,08	99,92
28	25,57	25,60	0,03	0,12	99,88
29	25,60	25,60	0,00	0,00	100,00
30	25,62	25,60	0,02	0,08	99,92
	Rata-rata	**Rata-rata**		**Rata-rata % error**	**Rata-rata % Akurasi**
	25,57	25,60		0,21	99,79

(LANJUTAN)

Data hasil pengukuran FD 0 panjang kabel = 100 meter,V_{DC}=10 Volt

No.	Suhu (^oC)		*Error* terhadap pembacaan *thermometer*		
	Pengukuran	Pembacaan *Thermometer*	*Error*	% *Error*	% Akurasi
1	29,60	30,00	0,40	1,33	98,67
2	29,53	30,00	0,47	1,57	98,43
3	29,65	30,00	0,35	1,17	98,83
4	29,84	30,00	0,16	0,53	99,47
5	29,89	30,00	0,11	0,37	99,63
6	29,91	30,00	0,09	0,30	99,70
7	29,90	30,00	0,10	0,33	99,67
8	29,92	30,00	0,08	0,27	99,73
9	29,93	30,00	0,07	0,23	99,77
10	29,93	30,00	0,07	0,23	99,77
11	29,92	30,00	0,08	0,27	99,73
12	29,95	30,00	0,05	0,17	99,83
13	30,06	30,00	0,06	0,20	99,80
14	30,10	30,00	0,10	0,33	99,67
15	30,05	30,00	0,05	0,17	99,83
16	30,14	30,00	0,14	0,47	99,53
17	29,95	30,00	0,05	0,17	99,83
18	30,03	30,30	0,27	0,89	99,11
19	30,04	30,00	0,04	0,13	99,87
20	29,91	30,00	0,09	0,30	99,70
21	30,02	30,00	0,02	0,07	99,93
22	30,40	30,00	0,40	1,33	98,67
23	30,23	30,30	0,07	0,23	99,77
24	30,41	30,30	0,11	0,36	99,64
25	30,43	30,30	0,13	0,43	99,57
26	30,60	30,30	0,30	0,99	99,01
27	30,50	30,50	0,00	0,00	100,00
28	30,27	30,30	0,03	0,10	99,90
29	30,36	30,30	0,06	0,20	99,80
30	30,33	30,30	0,03	0,10	99,90
	Rata-rata	**Rata-rata**		**Rata-rata % error**	**Rata-rata % Akurasi**
	30,06	30,10		0,44	99,56

(LANJUTAN)

Data hasil pengukuran FD 0 panjang kabel = 1000 *feet*, V_{DC}=12 Volt

No.	Suhu (°C)		*Error* terhadap pembacaan *thermometer*		
	Pengukuran	Pembacaan *Thermometer*	*Error*	% *Error*	% Akurasi
1	27,76	28,00	0,24	0,86	99,14
2	27,90	28,00	0,10	0,36	99,64
3	27,95	28,00	0,05	0,18	99,82
4	27,97	28,00	0,03	0,11	99,89
5	28,00	28,00	0,00	0,00	100,00
6	27,92	28,00	0,08	0,29	99,71
7	27,00	27,50	0,50	1,82	98,18
8	27,40	27,50	0,10	0,36	99,64
9	27,56	27,50	0,06	0,22	99,78
10	27,60	27,50	0,10	0,36	99,64
11	27,63	27,50	0,13	0,47	99,53
12	27,65	27,50	0,15	0,55	99,45
13	27,72	27,70	0,02	0,07	99,93
14	27,80	27,70	0,10	0,36	99,64
15	27,77	27,70	0,07	0,25	99,75
16	27,80	27,70	0,10	0,36	99,64
17	28,20	28,20	0,00	0,00	100,00
18	28,26	28,20	0,06	0,21	99,79
19	28,31	28,20	0,11	0,39	99,61
20	28,37	28,40	0,03	0,11	99,89
21	28,38	28,40	0,02	0,07	99,93
22	28,40	28,40	0,00	0,00	100,00
23	28,37	28,40	0,03	0,11	99,89
24	28,36	28,40	0,04	0,14	99,86
25	28,39	28,40	0,01	0,04	99,96
26	28,40	28,40	0,00	0,00	100,00
27	28,43	28,40	0,03	0,11	99,89
28	28,5	28,40	0,10	0,35	99,65
29	28,54	28,40	0,14	0,49	99,51
30	28,58	28,40	0,18	0,63	99,37
	Rata-rata	**Rata-rata**		**Rata-rata % error**	**Rata-rata % Akurasi**
	28,03	28,03		0,31	99,69

(LANJUTAN)

Data hasil pengukuran FD 0 panjang kabel = 1000 *feet*, V_{DC}=10 Volt

No.	Suhu (°C)		*Error* terhadap pembacaan *thermometer*		
	Pengukuran	Pembacaan *Thermometer*	*Error*	% *Error*	% AKurasi
1	30,89	31,1	0,21	0,68	99,32
2	30,96	31,4	0,44	1,42	98,58
3	31,01	31,5	0,49	1,58	98,42
4	31,01	31,5	0,49	1,58	98,42
5	31,01	31,5	0,49	1,58	98,42
6	31,06	31,8	0,74	2,38	97,62
7	31,08	31,8	0,72	2,32	97,68
8	31,08	31,8	0,72	2,32	97,68
9	31,08	31,8	0,72	2,32	97,68
10	31,11	31,8	0,69	2,22	97,78
11	31,17	31,8	0,63	2,02	97,98
12	31,20	31,8	0,60	1,92	98,08
13	31,24	32,0	0,76	2,43	97,57
14	31,25	32,0	0,75	2,40	97,60
15	31,33	32,0	0,67	2,14	97,86
16	31,31	32,0	0,69	2,20	97,80
17	31,36	32,3	0,94	3,00	97,00
18	31,35	32,3	0,95	3,03	96,97
19	31,60	32,5	0,90	2,85	97,15
20	31,63	32,7	1,07	3,38	96,62
21	31,75	32,7	0,95	2,99	97,01
22	32,11	32,0	0,11	0,34	99,66
23	32,06	32,0	0,06	0,19	99,81
24	32,04	32,0	0,04	0,12	99,88
25	32,01	32,0	0,01	0,03	99,97
26	32,00	32,0	0,00	0,00	100,00
27	32,00	32,0	0,00	0,00	100,00
28	32,01	32,0	0,01	0,03	99,97
29	32,01	32,0	0,01	0,03	99,97
30	32,04	32,0	0,04	0,12	99,88
	Rata-rata	**Rata-rata**		**Rata-rata % error**	**Rata-rata % Akurasi**
	31,46	31,94		1,59	98,41

(LANJUTAN)

Data hasil pengukuran FD 1 panjang kabel = 100 meter,V_{DC}=12 Volt

No.	Suhu (^oC)		*Error* terhadap pembacaan *thermometer*		
	Pengukuran	Pembacaan *Thermometer*	*Error*	% *Error*	% Akurasi
1	28,62	28,60	0,02	0,07	99,93
2	28,64	28,60	0,04	0,14	99,86
3	28,60	28,60	0,00	0,00	100,00
4	28,58	28,60	0,02	0,07	99,93
5	28,57	28,60	0,03	0,10	99,90
6	28,56	28,60	0,04	0,14	99,86
7	28,58	28,60	0,02	0,07	99,93
8	28,60	28,60	0,00	0,00	100,00
9	28,59	28,60	0,01	0,03	99,97
10	28,61	28,60	0,01	0,03	99,97
11	28,46	28,40	0,06	0,21	99,79
12	28,48	28,40	0,08	0,28	99,72
13	28,52	28,40	0,12	0,42	99,58
14	28,50	28,40	0,10	0,35	99,65
15	28,45	28,40	0,05	0,18	99,82
16	28,43	28,40	0,03	0,11	99,89
17	28,40	28,40	0,00	0,00	100,00
18	28,44	28,40	0,04	0,14	99,86
19	28,41	28,40	0,01	0,04	99,96
20	28,40	28,40	0,00	0,00	100,00
21	28,42	28,40	0,02	0,07	99,93
22	28,43	28,40	0,03	0,11	99,89
23	28,45	28,40	0,05	0,18	99,82
24	28,60	28,40	0,20	0,70	99,30
25	28,60	28,60	0,00	0,00	100,00
26	28,56	28,60	0,04	0,14	99,86
27	28,55	28,60	0,05	0,17	99,83
28	28,49	28,60	0,11	0,38	99,62
29	28,33	28,66	0,33	1,15	98,85
30	28,60	28,64	0,04	0,14	99,86
	Rata-rata	**Rata-rata**		**Rata-rata % error**	**Rata-rata % Akurasi**
	28,52	28,51		0,18	99,82

(LANJUTAN)

Data hasil pengukuran FD 1 panjang kabel = 100 meter,V_{DC}=10 Volt

No.	Suhu (^oC)		*Error* terhadap pembacaan *thermometer*		
	Pengukuran	Pembacaan *Thermometer*	*Error*	% *Error*	% Akurasi
1	32,14	32,5	0,36	1,11	98,89
2	32,20	32,5	0,30	0,92	99,08
3	32,26	32,5	0,24	0,74	99,26
4	32,27	32,7	0,43	1,31	98,69
5	32,33	32,7	0,37	1,13	98,87
6	32,35	33,0	0,65	1,97	98,03
7	32,37	33,0	0,63	1,91	98,09
8	32,45	33,2	0,75	2,26	97,74
9	32,46	33,2	0,74	2,23	97,77
10	32,48	33,2	0,72	2,17	97,83
11	32,51	33,2	0,69	2,08	97,92
12	32,53	33,2	0,67	2,02	97,98
13	32,56	33,4	0,84	2,51	97,49
14	32,59	33,4	0,81	2,43	97,57
15	32,57	33,4	0,83	2,49	97,51
16	32,59	33,4	0,81	2,43	97,57
17	32,60	33,4	0,80	2,40	97,60
18	32,64	33,4	0,76	2,28	97,72
19	32,63	33,4	0,77	2,31	97,69
20	31,86	33,2	1,34	4,04	95,96
21	31,82	33,2	1,38	4,16	95,84
22	31,87	33,2	1,33	4,01	95,99
23	30,02	30,0	0,02	0,07	99,93
24	29,83	29,8	0,03	0,10	99,90
25	29,79	29,8	0,01	0,03	99,97
26	29,76	29,8	0,04	0,13	99,87
27	29,78	29,8	0,02	0,07	99,93
28	29,66	29,6	0,06	0,20	99,80
29	29,62	29,6	0,02	0,07	99,93
30	30,18	30,0	0,18	0,60	99,40
	Rata-rata	**Rata-rata**		**Rata-rata % error**	**Rata-rata % Akurasi**
	31,69	32,22		1,67	98,33

LAMPIRAN 3

LIST PROGRAM FIELD DEVICE

```
#include <c8051f350.h>
#include <stdio.h>
#include <string.h>
#include <math.h>
#include "resp_data.h"

#define uchar unsigned char
#define uint unsigned int
#define ulong unsigned long

extern void send_resp_data(uchar delimit,uchar *address, command, byte_count, comm_or_resp_code, field_dev_stat,
uchar *resp_data);
extern void com_send(uchar *send, uchar send_bytes);
extern uchar checksum(uchar *z, uchar nbytes);
extern void msec(uint x);
extern int analog_value(float current);
extern void set_AD421(uint datanya);
extern float current_value;

sfr16 TMR2RL   = 0xca;                  // Timer2 reload value
sfr16 TMR2     = 0xcc;                  // Timer2 counter

#define SYSCLK      24500000
#define L1 00
#define L2 64
#define L3 20
#define L4 84
void delimiter_check(void);
void checksum_rx(void);
void hart_execute(void);
void hart_comm_resp(void);
void check_address(void);
void main_exe(void);

void lcddatwr(uchar dbyte);
void lcdcmdwr(uchar dbyte);
void Clear_Display (void);
void display(uchar startloc,uchar*s);
void numdsp(uchar startloc, ulong number, bit dpt);
void initialize(void);
```

(LANJUTAN)

```
xdata uchar uniq_addrs_prmry[5]={0xA1,0xA8,0x00,0x00,0x01};
xdata uchar uniq_addrs_scndry[5]={0x21,0xA8,0x00,0x00,0x01};
uchar angkax, char_rx;
sbit INRTS=P1^1;
xdata uchar send[50];
xdata uchar receive[50];
uchar send_bytes,comm,xx,yy,frame,delimiter, x1,rec_ptr,rec_bytes,received,not_resp,status1,status2;
uint ss,count, limit_counter;
bit error, ena_counter,sudahterima;
bit data_valid=0;
bit data_valid2=0;
float PV_loop_current, PV_percent_current,volt,value;;
double voltage,suhu;

void SYSCLK_Init (void)
{           PCA0MD &= ~0x40;
            OSCICN = 0x83;
}
void UART0_Init (void)
{           SCON0 = 0x10;          CKCON = 0x02;
            TMOD  = 0x20;          TH1   = 0x2B;
            TR1   = 1;   TI0   = 0;
            RI0   = 0;   ES0 = 1;                  // Enable UART0 interrupts
}
void PORT_Init (void)
{
            P0MDOUT   = 0x9D;    P1MDOUT   = 0xFF;
            XBR0      = 0x01; //UART ENABLE
            XBR1      = 0x40;        P1=0;
}
void Timer2_ISR (void) interrupt 5{
            TF2H = 0;
            if(ena_counter)limit_counter++;
}
void Timer2_Init (int counts)
{
            TMR2CN  = 0x00;      CKCON  &= ~0x60;      TMR2RL  = -counts;
            TMR2    = 0xffff;    ET2    = 1;           TR2     = 1;
}
void UART0int(void)interrupt 4 using 1{
            if(RI0){
                        RI0=0;                  ena_counter=1;
                        char_rx=SBUF0;          receive[rec_ptr]=char_rx;
```

(LANJUTAN)

```
                rec_ptr++;              limit_counter=0;
        }
}
void calibrate_ADC(void){
        AD0CALC = 0x10;     ADC0MD = 0x81;
        while(AD0CBSY);     msec(0xFFFF);
        while(ADC0MD & 0x07);
}
void ADC_Init()
{
        REF0CN |= 0x03;     ADC0CN = 0x01;      ADC0CF = 0x00;
        ADC0CLK  = 0xff;    ADC0BUF = 0x00;     ADC0MUX = 0xff;
        ADC0MD  = 0x80;             // Enable the ADC0 (IDLE Mode)
}
//============= MAIN OF PROGRAM ============================
void main (void)
{       SYSCLK_Init ();     PORT_Init();        initialize();       Timer2_Init(SYSCLK/12 /10);
        UART0_Init();       ADC_Init();         ADC0MD=0x80;
        EA = 1; received=0; error=0;  count=1;  char_rx=0;
        analog_value(4);
for(;;){
        temper=0;   suhu=0;   volt=0;           value=0;
        ADC0MD = 0x83;              // Start continuous conversions
        EA = 1;
        while(!AD0INT);
        AD0INT=0;
        temper = ADC0H;     temper = (temper<<8) + ADC0M;       temper = (temper<<8) + ADC0L;
        voltage = temper * 3000.0/16777215;     voltage=voltage*1000;
        suhu=(voltage-104500.00)/360.00; // good 2 for address 1
        value=(0.128*suhu)+9.12;
        INRTS=1;
                if(limit_counter>200)
                {       numdsp(0,rec_ptr,0);
                        ena_counter=0;
                        limit_counter=0;
                        for(xx=0;xx<rec_ptr;xx++)
                        {       if (receive[xx]==255 && receive[xx-1]==255)
                                { data_valid=1;     }
                if(data_valid){  if (receive[xx]!=255)      { data_valid2=1;    } }
                                if(data_valid2) { check_address(); }
                        }       rec_ptr=0;
                }
}
```

(LANJUTAN)

```
}
//=======================END OF MAIN PROGRAM=================
void check_address(void){
        uchar address,address1,address2,address3,address4;
        delimiter=receive[xx+0];
        if(delimiter==0x02){
                address=(receive[xx+1]);
                if(address==poll_addrs_prmry || address==poll_addrs_scndry)
                { delimiter_check();    }
        }
        if(delimiter==0x82){
                address=(receive[xx+1]);                address1=(receive[xx+2]);
                address2=(receive[xx+3]);               address3=(receive[xx+4]);
                address4=(receive[xx+5]);
                if(address==(uniq_addrs_prmry[0]) && (address1==uniq_addrs_prmry[1]) &&
(address2==uniq_addrs_prmry[2]) && (address3==uniq_addrs_prmry[3]) && (address4==uniq_addrs_prmry[4]))
                { delimiter_check();    }
                else if (address==(uniq_addrs_scndry[0]) && (address1==uniq_addrs_scndry[1]) &&
(address2==uniq_addrs_scndry[2]) && (address3==uniq_addrs_scndry[3]) && (address4==uniq_addrs_scndry[4]))
                { checksum_rx();}
        }
}
void main_exe(void)
{       if(error)
        {       data_valid=0;       data_valid2=0;      INRTS=0;   msec(5000);          }
        else
        {       data_valid=0;       data_valid2=0;      hart_comm_resp();   count=count+1;      }
}
void delimiter_check(void){
uchar byte_count;
        delimiter=receive[xx+0];
        if(delimiter==0x02){
                comm=(receive[xx+2]);           byte_count=(receive[xx+3]);
                frame=byte_count+5;             checksum_rx();
                main_exe();
        }
        if(delimiter==0x82){
                comm=(receive[xx+6]);           byte_count=(receive[xx+7]);
                frame=byte_count+9;             checksum_rx();
                main_exe();
        }
}
void checksum_rx(void){
```

(LANJUTAN)

```
        uchar last_data,sum,n,checksum;
        last_data=frame-1;    sum=(receive[xx+last_data]);      checksum=(receive[xx]);
        for(n=1;n<last_data;n++){          checksum=checksum^(receive[xx+n]);         }
        error=checksum^sum;
}
void hart_comm_resp(void){
        if(comm==2){
                INRTS=0;msec(5000);
                PV_loop_current=current_value;
                PV_percent_current=((PV_loop_current-4)/16)*100;
                PV_loop_current=(16*(PV_percent_current/100))+4;
                resp_data_2[0]=((uchar *)&PV_loop_current)[0];
                resp_data_2[1]=((uchar *)&PV_loop_current)[1];
                resp_data_2[2]=((uchar *)&PV_loop_current)[2];
                resp_data_2[3]=((uchar *)&PV_loop_current)[3];
                resp_data_2[4]=((uchar *)&PV_percent_current)[0];
                resp_data_2[5]=((uchar *)&PV_percent_current)[1];
                resp_data_2[6]=((uchar *)&PV_percent_current)[2];
                resp_data_2[7]=((uchar *)&PV_percent_current)[3];
                send_resp_data(0x86,uniq_addrs_prmry, 0x02, 0x0A, 0x00, field_device_status_2, resp_data_2);
        }
        if(comm==3){
                INRTS=0;msec(5000);
                PV_loop_current=value;
                resp_data_3[0]=((uchar *)&PV_loop_current)[0];resp_data_3[1]=((uchar *)&PV_loop_current)[1];
                resp_data_3[2]=((uchar *)&PV_loop_current)[2];resp_data_3[3]=((uchar *)&PV_loop_current)[3];
                resp_data_3[4]=0x20; // Temperature unit code (degree celcius
                resp_data_3[5]=((uchar *)&suhu)[0];          resp_data_3[6]=((uchar *)&suhu)[1];
                resp_data_3[7]=((uchar *)&suhu)[2];          resp_data_3[8]=((uchar *)&suhu)[3];
                send_resp_data(0x86,uniq_addrs_prmry,0x03, 0x0B, 0x00, field_device_status_2, resp_data_3);
        }
        if(comm==40){
                resp_data_40_45_46[0]=receive[xx+8];      resp_data_40_45_46[1]=receive[xx+9];
                resp_data_40_45_46[2]=receive[xx+10];     resp_data_40_45_46[3]=receive[xx+11];
                PV_loop_current= *(float *)&resp_data_40_45_46;
                analog_value(PV_loop_current);
                send_resp_data(0x86,uniq_addrs_prmry, 0x28, 0x06, 0x00, field_device_status_5,
resp_data_40_45_46);
        msec(0xffff);msec(0xffff);msec(0xffff);msec(0xffff);msec(0xffff);msec(0xffff);msec(0xffff);msec(0xffff);
        }
}
```

LAMPIRAN 4

LIST PROGRAM APLIKASI *MONITORING* SUHU

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using WindowsFormsApplication1;
using System.Threading;
using System.IO;
using System.IO.Ports;


namespace WindowsFormsApplication1
{
   public partial class TempMonitor : Form
   {
      int i;
      int framelenght;
      byte _checksum;
      byte[] message = new byte[50];
      byte[] long_address = new byte[5] ;
      byte[] long_address0 = new byte[] { 0xA1, 0xA8, 0x00, 0x00, 0x00 };
      byte[] long_address1 = new byte[] { 0xA1, 0xA8, 0x00, 0x00, 0x01 };
      byte[] data_pesan = new byte[20];
      DateTime jamku;
      public TempMonitor()
      {
         InitializeComponent();
         TBother.Enabled = false;
         LblAdd.Text = "";          LblCmd.Text = "";
         timer_auto_mode.Enabled = false;

#region Get Data
      private void btnGetData_Click(object sender, EventArgs e)
      {
         timer1.Enabled = true;       timer_interval(1500);
         TBframesend.Clear();       TBframereceived.Clear();
         LblAdd.Text = "";          LblCmd.Text = "";
         try
         {
            port.RtsEnable = true;
            frame(0x82, long_address0, 0x03, 0x00, data_pesan);
            port.RtsEnable = false;             port.DtrEnable = true;
         }
         catch (Exception) { MessageBox.Show("Open Port First!"); }
      }
#endregion
#region Select Fix Current
      private void btnSendCurrent_Click(object sender, EventArgs e)
      {
         timer1.Enabled = true;          timer_interval(1500);
         TBframesend.Clear();          TBframereceived.Clear();
         LblAdd.Text = "";          LblCmd.Text = "";
         try
         {
            if (RB4mA.Checked)
               {
               port.RtsEnable = true;
               current_float(4);
               frame(0x82, long_address, 0x28, 0x04, data_pesan);
```

(LANJUTAN)

```
            port.RtsEnable = false;                port.DtrEnable = true;
        }
        if (RB8mA.Checked)
        {
            port.RtsEnable = true;
            current_float(8);
            frame(0x82, long_address, 0x28, 0x04, data_pesan);
            port.RtsEnable = false;                port.DtrEnable = true;
        }
        if (RB12mA.Checked)
        {
            port.RtsEnable = true;
            current_float(12);
            frame(0x82, long_address, 0x28, 0x04, data_pesan);
            port.RtsEnable = false;                port.DtrEnable = true;
        }
        if (RB16mA.Checked)
        {
            port.RtsEnable = true;
            current_float(16);
            frame(0x82, long_address, 0x28, 0x04, data_pesan);
            port.RtsEnable = false;                port.DtrEnable = true;
        }
        if (RB20mA.Checked)
        {
            port.RtsEnable = true;
            current_float(20);
            frame(0x82, long_address, 0x28, 0x04, data_pesan);
            port.RtsEnable = false;                port.DtrEnable = true;
        }
        if (RBother.Checked)
        {
            double x;
            x = Convert.ToSingle(TBother.Text);
            port.RtsEnable = true;
            current_float(x);
            frame(0x82, long_address, 0x28, 0x04, data_pesan);
            port.RtsEnable = false;                port.DtrEnable = true;
        }
      }
      catch (Exception) { MessageBox.Show("Open Port First!"); }
    }
    private void RBother_CheckedChanged(object sender, EventArgs e)
    {
      TBother.Enabled = (RBother.Checked == true) ? true : false;
    }
#endregion
    #region send message
    public void frame(byte delimiter, byte[] addresss, byte command, byte bytecount, byte[] pesan)
    {
      for (i = 0; i <= 4; i++) { message[i] = 0xFF; }
      message[i++] = delimiter;//Delimiter
      foreach (byte item in addresss)
      {          message[i++] = item;       }
      message[i++] = command;        message[i++] = bytecount;
      if (bytecount!=0)
      {
        foreach (byte item in pesan)
        {
          message[i++] = item;
        }
      }
      _checksum = 0;
      for (i = 5; i < (bytecount + 13); i++)
```

(LANJUTAN)

```
    {
        _checksum = Convert.ToByte(_checksum ^ message[i]);
    }
    message[i++] = _checksum;
    framelenght = bytecount + 14;
    port.Write(message, 0, framelenght);
    Thread.Sleep(500);          displaydata();          displaystatusdata();
}
#endregion
#region displaydata
public void displaydata()
{
    string hex = BitConverter.ToString(message).Replace("-", " ");
    var arrays = hex.Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);
    for (i = 0; i < framelenght; i++)
    {
        TBframesend.Text += arrays[i] + " ";
    }
}
#endregion
#region display status data
public void displaystatusdata()
{
    string hex = BitConverter.ToString(message).Replace("-", " ");
    var arrays = hex.Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);
    if (message[5] == 0x82)
    {
        LblAdd.Text = "Long Address : ";
        for (int i = 6; i <= 10; i++) { LblAdd.Text += arrays[i] + " "; }
        LblCmd.Text += "Command Out: " + message[11];
    }
}
#endregion
#region Float to Hex
private void current_float(double current)
{
    float float_value = (float)current;
    byte[] bytes = BitConverter.GetBytes(float_value);
    data_pesan[0] = bytes[3];          data_pesan[1] = bytes[2];
    data_pesan[2] = bytes[1];          data_pesan[3] = bytes[0];
}
#endregion
#region Hex to Float
private void current_hex(byte[] data)
{
    byte[] newdata = new byte[4];
    newdata[3] = data[0];          newdata[2] = data[1];
    newdata[1] = data[2];          newdata[0] = data[3];
    float f = BitConverter.ToSingle(newdata, 0);
    TBCurrent.Text = f.ToString("0.00");          float percent = ((f - 4) / 16) * 100;
    TBPercent.Text = percent.ToString("0.00");
}
private void current_hex1(byte[] data)
{
    byte[] newdata = new byte[4];
    newdata[3] = data[0];          newdata[2] = data[1];
    newdata[1] = data[2];          newdata[0] = data[3];
    float f = BitConverter.ToSingle(newdata, 0);
    TBCurrent1.Text = f.ToString("0.00");          float percent = ((f - 4) / 16) * 100;
    TBPercent1.Text = percent.ToString("0.00");
}
private void loopcurrent_hex(byte[] data)
{
    byte[] newdata = new byte[4];
```

(LANJUTAN)

```
        newdata[3] = data[0];          newdata[2] = data[1];
        newdata[1] = data[2];          newdata[0] = data[3];
        float f = BitConverter.ToSingle(newdata, 0);          LoopCurrent0.Text = f.ToString("0.00");
    }
    private void loopcurrent_hex1(byte[] data)
    {
        byte[] newdata = new byte[4];
        newdata[3] = data[0];          newdata[2] = data[1];
        newdata[1] = data[2];          newdata[0] = data[3];
        float f = BitConverter.ToSingle(newdata, 0);          LoopCurrent1.Text = f.ToString("0.00");
    }
    #endregion
    #region Hex to Float temperature
    private void temper_hex(byte[] data)
    {
        byte[] newdata = new byte[4];
        newdata[3] = data[0];          newdata[2] = data[1];
        newdata[1] = data[2];          newdata[0] = data[3];
        float f = BitConverter.ToSingle(newdata, 0);          DerajatCelciusView.Text = f.ToString("0.00");
    }
    private void temper_hex1(byte[] data)
    {
        byte[] newdata = new byte[4];
        newdata[3] = data[0];          newdata[2] = data[1];
        newdata[1] = data[2];          newdata[0] = data[3];
        float f = BitConverter.ToSingle(newdata, 0);          Derajat1.Text = f.ToString("0.00");
    }
    #endregion
    #region Hex to Float(percent)
    private void percent_hex(byte[] datat)
    {
        byte[] newdata = new byte[4];
        newdata[3] = datat[0];          newdata[2] = datat[1];
        newdata[1] = datat[2];          newdata[0] = datat[3];
        float f = BitConverter.ToSingle(newdata, 0);          TBPercent.Text = f.ToString("0.00");
    }
    private void percent_hex1(byte[] datat)
    {
        byte[] newdata = new byte[4];
        newdata[3] = datat[0];          newdata[2] = datat[1];
        newdata[1] = datat[2];          newdata[0] = datat[3];
        float f = BitConverter.ToSingle(newdata, 0);          TBPercent1.Text = f.ToString("0.00");
    }
    #endregion

    private void TempMonitor_Load(object sender, EventArgs e)
    {
        jamku = DateTime.Now;
        timer2.Enabled = true;          timer_auto_mode.Enabled = false;
        port.BaudRate = 1200;          cboPort.Items.Clear();
        foreach (string s in SerialPort.GetPortNames())
        {
            cboPort.Items.Add(s);
        }
        btnClose.Enabled = false;
    }
#region Port Setting
    private void btnOpen_Click(object sender, EventArgs e)
    {
        try
        {
            port.PortName = cboPort.Text;
            port.Open();
            if (port.IsOpen)
```

(LANJUTAN)

```
            {
                btnOpen.Enabled = false;                btnClose.Enabled = true;
            }
        }
        catch (Exception)
        {
            MessageBox.Show("Select Port Please!");
        }
    }
    private void btnClose_Click(object sender, EventArgs e)
    {
        if (port.IsOpen)
        {
            port.Close();            btnOpen.Enabled = true;            btnClose.Enabled = false;
        }
    }
#endregion
#region Timer Data Received
    private void timer1_Tick(object sender, EventArgs e)
    {
        try
        {
            byte[] data_pesan_in = new byte[4];            int bytes = port.BytesToRead;
            byte[] message_in = new byte[bytes];            byte[] address_in = new byte[5];
            byte[] command_in=new byte[1];
            if (port.BytesToRead != 0)
            {
                TBframereceived.Clear();
                port.Read(message_in, 0, bytes);
                string hexa = BitConverter.ToString(message_in).Replace("-", " ");
                _checksum = 0;
                byte bytecount_in=message_in[12];
                for (i = 5; i < (message_in[12] + 13); i++)
                {
                    _checksum = Convert.ToByte(_checksum ^ message_in[i]);
                }
                byte checkerror = Convert.ToByte(_checksum ^ (message_in[bytecount_in + 13]));
                address_in[0] = message_in[6];                address_in[1] = message_in[7];
                address_in[2] = message_in[8];                address_in[3] = message_in[9];
                address_in[4] = message_in[10];                command_in[0] = message_in[11];
                if (message_in[0] == 0xFF && message_in[1] == 0xFF && message_in[5] == 0x86 || message_in[5] == 0x81 )
                {
                    if (checkerror == 0)
                    {
                        TBframereceived.Invoke(new EventHandler(delegate
                        {
                            TBframereceived.Text += hexa + " ";//display received data;
                        }));
                        if (address_in[4] == long_address0[4])
                        {
                            if (message_in[11] == 3||message_in[11] == 40)
                            {
                                try
                                {
                                    for (int x = 0; x < 5; x++)
                                    {
                                        data_pesan_in[x] = message_in[x + 15];                    current_hex(data_pesan_in);
                                    }
                                }
                                catch (Exception) { }
                                    for (int y = 0; y < 5; y++)
                                    {
                                        data_pesan_in[y] = message_in[y + 20];                    temper_hex(data_pesan_in);
                                    }
```

(LANJUTAN)

```
                    }
                    if (message_in[11] == 2)
                    {
                        try
                        {
                            for (int x = 0; x < 5; x++)
                            {
                                data_pesan_in[x] = message_in[x + 15];                    loopcurrent_hex(data_pesan_in);
                            }
                        }
                        catch (Exception) { }
                    }
                }
                if (address_in[4]== long_address1[4])
                {
                    if (message_in[11] == 3 || message_in[11] == 40)
                    {
                        try
                        {
                            for (int x = 0; x < 5; x++)
                            {
                                data_pesan_in[x] = message_in[x + 15];                     current_hex1(data_pesan_in);
                            }
                        }
                        catch (Exception) { }
                        try
                        {
                            for (int y = 0; y < 5; y++)
                            {
                                data_pesan_in[y] = message_in[y + 20];                    temper_hex1(data_pesan_in);
                            }
                        }
                        catch (Exception) { }
                    }
                    if (message_in[11] == 2)
                    {
                            for (int x = 0; x < 5; x++)
                            {
                                data_pesan_in[x] = message_in[x + 15];
                                loopcurrent_hex1(data_pesan_in);
                            }
                    }

                }
            }
        }
    }
    catch (Exception) { }
}

    private void timer_interval(int x)
    {
        timer1.Interval = x;
    }
#endregion

    private void timer2_Tick(object sender, EventArgs e)
    {
        jamku = DateTime.Now;
        Hour.Text = jamku.Hour.ToString("00") + ":";          Minute.Text = jamku.Minute.ToString("00") + ":";
        Second.Text = jamku.Second.ToString("00");
        Day.Text = jamku.Day.ToString("00") + "/";          Month.Text = jamku.Month.ToString("00") + "/";
        Year.Text = jamku.Year.ToString("0000");
```

(LANJUTAN)

```
    }
#endregion
    private void btnGetData1_Click(object sender, EventArgs e)
    {
      timer1.Enabled = true;          timer_interval(1500);
      TBframesend.Clear();          TBframereceived.Clear();
      LblAdd.Text = "";          LblCmd.Text = "";
      try
      {
        port.RtsEnable = true;
        frame(0x82, long_address1, 0x03, 0x00, data_pesan);
        port.RtsEnable = false;              port.DtrEnable = true;
      }
      catch (Exception) { MessageBox.Show("Open Port First!"); }
    }
    private void CB0_CheckedChanged(object sender, EventArgs e)
    {
      long_address=new byte[]{ 0xA1, 0xA8, 0x00, 0x00, 0x00 };
    }
    private void CB1_CheckedChanged(object sender, EventArgs e)
    {
      long_address =new byte[]{ 0xA1, 0xA8, 0x00, 0x00, 0x01 };
    }

    private void timer_auto_mode_Tick(object sender, EventArgs e)
    {
      frame(0x82, long_address0, 0x02, 0x00, data_pesan);
      port.RtsEnable = false;          port.DtrEnable = true;
      TBframesend.Text = "";        TBframesend.Text = "";
    }
    private void GetData_Arus0_Click(object sender, EventArgs e)
    {
      timer1.Enabled = true;          timer_interval(2000);
      TBframesend.Clear();          TBframereceived.Clear();
      LblAdd.Text = "";          LblCmd.Text = "";
      try
      {
        port.RtsEnable = true;
        frame(0x82, long_address0, 0x02, 0x00, data_pesan);
        port.RtsEnable = false;              port.DtrEnable = true;
      }
      catch { }
    }
    private void GetData_Arus1_Click(object sender, EventArgs e)
    {
      timer1.Enabled = true;          timer_interval(2000);
      TBframesend.Clear();          TBframereceived.Clear();
      LblAdd.Text = "";          LblCmd.Text = "";
      try
      {
        port.RtsEnable = true;
        frame(0x82, long_address1, 0x02, 0x00, data_pesan);
        port.RtsEnable = false;              port.DtrEnable = true;
      }
      catch { }
    }
  }
}
```