DESAIN DAN IMPLEMENTASI SISTEM KOMUNIKASI PROSESOR PARALLEL PADA APLIKASI ROBOTIK BERBASIS BUS SERIAL I²C

Tito Mubarak[1], Sony Sumaryo, Ir., MT.[2], Maman Abdurohman, ST., MT.[3]
[1,2] Jurusan Teknik Elektro, [3] Jurusan Teknik Informatika, Institut Teknologi Telkom
[1]tito.mubarak@gmail.com, [2]snyry@yahoo.com, sny@stttelkom.ac.id, [3]mma@stttelkom.ac.id

ABSTRAKSI

Dewasa ini desain sistem *embedded* tidak cukup lagi hanya dengan menerapkan satu buah *microprocessor* atau *microcontroller* saja. Karena sistem yang dikontrol mempunyai peralatan dan tugas yang semakin banyak. Sehingga perlu menerapkan desentralisasi dan paralelisme pada sistem berbasis *microprocessor*.

Pada Penelitian ini, dirancang suatu sistem komunikasi antar *microcontroller* sebagai implementasi dari *microprocessor*. Yaitu berupa suatu sistem *bus* yang berbasis pada I²C *serial bus* yang dikeluarkan oleh Phillips. *Bus* ini berfungsi sebagai jalur penghubung antar *microcontroller* untuk melewatkan *data-data* kontrol atau parameter lain.

Penelitian ini menghasilkan satu set rutin *software* protokol sistem komunikasi antar *microcontroller* yang berjalan di atas I²C *serial bus* sebagai lapisan fisiknya. Rutin *software* tersebut ditulis dalam bahasa *assembly* Intel MCS-51. Kemudian diimplementasikan pada keluarga *microcontroller* Atmel AT89. Ruang ROM yang digunakan oleh *software* inti sebanyak 392 *byte* (9.57% dari 4 kB) pada *Master* dan 565 *byte* (13.79% dari 4 kB) pada *Slave*. Sedangkan *bit rate* tertinggi yang diperoleh sekitar 95057.03 bps pada *clock microcontroller* sebesar 24 MHz (2 MIPS). Sebagai bukti bahwa sistem *bus* berjalan sesuai rancangan, sistem tersebut digunakan pada sebuah model robotika *wheeled line-follower*.

Kata kunci: *I²C serial bus, MCS-51 microcontroller, parallelism, decentralized, wheeled line-follower*

1. PENDAHULUAN

Perkembangan desain sistem *embedded* pada saat ini cenderung menggunakan lebih dari satu buah *microprocessor* atau *microcontroller*. Hal ini dilakukan karena sistem yang dikontrol semakin komplek dan semakin banyak peralatan yang harus terhubung pada *microprocessor* atau *microcontroller*.

Sehingga diharapkan beban atau *processor-load* secara keseluruhan tidak terpusat pada satu buah *microprocessor* saja. Sehingga dapat dihindari *over-load* pada *microprocessor* dan sistem dapat bersifat *modular*. Selain itu sensor dan aktuator juga menjadi bersifat lebih *smart.* Di sisi lain parallelisme pada tingkat *task* dapat dicapai.

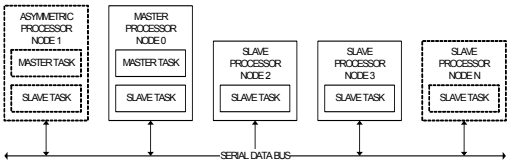
1.2 Tujuan Penelitian

Penelitian ini bertujuan untuk mendesain I²C *serial bus* sebagai *bus* penghubung antar *microcontroller* sebagai sistem *microprocessor* dan mengimplementasikannya pada suatu model robotik *wheeled line-follower*.

2. DASAR TEORI

2.1 Konsep I²C *Serial Bus*

Untuk menghubungkan antar *processor* atau *controller* dibutuhkan suatu sistem *bus* tertentu. *Bus* ini harus dapat melewatkan *data* atau parameter kontrol antar *processor* yang berfungsi sebagai jaringan atau *network*.



Gambar1. Blok model *prototype*

Pendekatan yang sering digunakan adalah *serial bus*. Alasan penggunaan *serial bus* adalah implementasinya yang lebih mudah dalam sisi pengkabelan dan lebih murah daripada *parallel bus*.

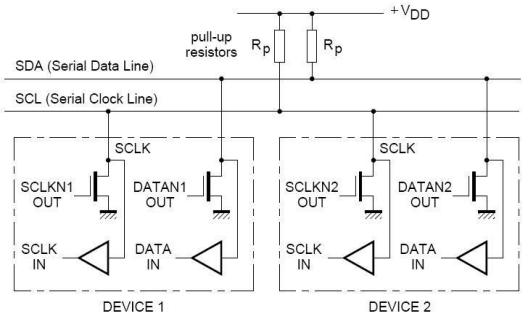
I²C *serial bus* yang dikembangkan oleh Phillips sejak 1980 ditujukan untuk aplikasi kontrol dengan *microcontroller*. Sehingga antar *microprocessor* atau *microcontroller* dapat saling berkoordinasi melalui I²C *serial bus*. Dari segi *software* dan *speed*, I²C *serial bus* cukup mudah untuk diimplementasikan dan mempunyai *speed* yang memadai untuk sistem kontrol berskala kecil seperti pada robotika.

Pada dokumen resmi yang dikeluarkan oleh Phillips *semiconductor*, terdapat tiga buah kecepatan pada I²C *serial bus*. Antara lain 100 kHz (*Standard-mode*), 400 kHz (*Fast-mode*) dan 3.4 MHz (*High-speed mode*). Untuk *microcontroller* umum dengan *troughput* sekitar 2 MIPS (*Million Instruction Per Second*), kecepatan I²C *serial bus* pada *Standard Speed* masih dapat dicapai. Namun untuk *Full Speed* atau *High Speed* diperlukan *microcontroller* yang jauh lebih cepat atau dengan menggunakan *dedicated* I²C *serial bus controller*.

Komunikasi pada I^{2}C *serial bus* merupakan komunikasi *serial* sinkron. Artinya data pada *bus* dikirim *bit* per *bit* (*serial*) dan *clock* ikut dikirimkan (sinkron). Sebagai saluran *data* berupa SDA *line* dan saluran *clock* berupa SCL *line*. Dan satu buah saluran *common ground* sebagai referensi.

I^{2}C *serial bus* menerapkan konsep *Master – Slave*. Dimana *Master* secara penuh bertugas menangani setiap *event* komunikasi pada *bus*. Setiap alur komunikasi selalu diawali oleh *Master* dan diakhiri oleh *Master*. Baik komunikasi *data* dari *Master* ke *Slave* (*write*) atau dari *Slave* ke *Master* (*read*).

Spesifikasi elektrik tegangan pada I^{2}C *serial bus* tidak dibatasi secara spesifik. Phillips hanya mendeskripsikan persyaratan tegangan minimal dan maksimal yang harus ada pada sistem *bus*. Sehingga I^{2}C *serial bus* dapat diterapkan pada sembarang sirkuit logika seperti TTL (*Transistor-Transistor Logic*), CMOS (*Complementary Metal Oxyde Semiconductor*), BiCMOS (*Bipolar* CMOS), ECL (*Emitter Coupled Logic*), 5 *Volt Logic*, 3.3 *Volt Logic*, 1.8 *Volt Logic* dan standar logika lainnya.

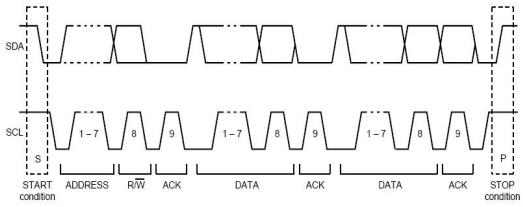


Gambar 2. Koneksi pin SDA dan SCL dengan *open-drain* pada logika CMOS

Sedangkan parameter yang dideskripsikan secara ketat adalah parameter pewaktuan sistem *bus*. Kesesuaian terhadap parameter ini menjamin sistem *bus* I^{2}C dapat berjalan sesuai rancangan. Masing-masing dijelaskan secara detail untuk setiap standar kecepatan.

Sistem komunikasi pada I^{2}C *serial bus* selalu diawali dengan *Start Condition* dan diakhiri dengan *Stop Condition*. Setelah pengiriman *Start Condition* diikuti dengan pengiriman *Address* atau alamat dari *Slave* yang akan terlibat dalam komunikasi. Terdapat dua buah versi alamat yaitu 7 *bit* dan 10 *bit*. Baik pengalamatan 7 *bit* maupun 10 *bit* mempunyai satu buah *bit* sifat dari komunikasi (*read* atau *write*).

Pengiriman *data* berorientasi *byte* dengan satu buah bit tambahan yang berfungsi sebagai *acknowledgement bit*. Sehingga setiap terjadi komunikasi merupakan rangkaian dari *Start Condition – Address – Read / Write Bit – Acknowledge – Data – Acknowledge – Stop Condition*. Banyaknya *data* yang dapat dikirim tidak dibatasi secara spesifik.

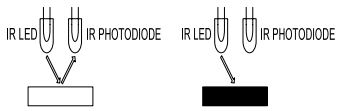


Gambar 3. *Data transfer* pada I^{2}C *serial bus*

2.2 Sensor Infra Merah sebagai Pendeteksi Garis

Sensor yang digunakan pada model robotik adalah sensor cahaya infra merah. Ada dua buah sensor pada bagian depan model robotik untuk mendeteksi gelap-terang dari permukaan jalan yang dilalui. Masing-masing sensor mempunyai pemancar infra merahnya. Panjang gelombang yang dipakai pada pasangan pemancar infra merah (IR LED) dan sensor infra merah (IR PhotoDiode) merupakan pasangan yang sesuai.

Misalkan permukaan yang sedang dilewati berwarna terang (putih). Sehingga sebagian besar cahaya infra merah dipantulkan dan diterima oleh sensor. Maka sensor akan menghantar dan rangkaian penguat sensor akan memberikan logika 0 (*low*). Begitu pula sebaliknya apabila permukaan yang sedang dilewati berwarna gelap (hitam). Maka sebagian besar cahaya infra merah akan diserap oleh permukaan gelap. Sehingga sensor akan berhenti menghantar dan penguat sensor akan mengeluarkan logika 1 (*high*). Selanjutnya kondisi *low* atau *high* dari sensor akan digunakan oleh *microcontroller* untuk mengambil keputusan dalam menggerakkan motor.



Gambar 4. Sensor infra merah

3. PERANCANGAN DAN IMPLEMENTASI

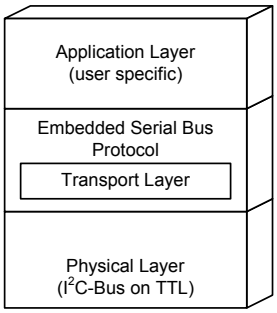
Perancangan sistem dilakukan melalui tiga buah tahap. Yang pertama adalah perancangan protokol berikut *layer stack* dengan bahasa pemrograman *assembly* dan simulasi pada *simulator*. Berikutnya adalah implementasi protokol pada *hardware*. Dan yang terakhir adalah implementasi sistem *bus* pada model robotik *wheeled line-follower*.

3.1 Perancangan Protokol dengan Bahasa *Assembly* dan *Simulator*

Layer stack yang diimplementasikan terdapat tiga buah. Pada *layer* paling bawah adalah *Physical Layer* yang terdiri dari level logika itu sendiri (TTL / CMOS) dan I^{2}C *serial bus*. *Layer* ini berfungsi menyediakan jalur fisik bagi sistem *bus*.

Di atas *Physical Layer* adalah *Transport Layer*. *Transport Layer* bertugas mengatur penanganan *Frame* yang dikirim dan yang diterima. Fungsi yang terdapat pada *layer* ini antara lain *Slave addressing*, *byte counting* dan *error detecting*.

Dan *layer* paling atas adalah *Application Layer*. *Layer* ini adalah *task* dari sistem itu sendiri. Dimana *layer* ini adalah *layer* yang memanfaatkan fungsi dari sistem *bus*.

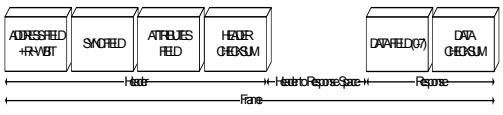


Gambar 5. Model *layer stack*

Perancangan protokol dimulai dengan mendeskripsikan *event* pada sistem *bus*. Antara lain *Start Condition*, *Stop Condition*, *Shift-In Bit*, *Shift-Out Bit*, dan *Acknowledgement*.

Pada *Slave* terdapat fungsi rutin *dummy* seperti *Dummy Shift-In* dan *Dummy Shift-Out*. Fungsi ini berguna untuk mensinkronkan *Slave* dengan *bus* ketika *Slave* tidak dialamati oleh *Master* dalam suatu *Frame*.

Pada *Transport Layer* terdapat kesatuan data yang disebut dengan *Frame*. *Frame* sendiri terdiri dari *Header* dan *Response*. Pembentukan *Frame* yang terdiri dari *Header* dan *Response* diilhami dari LIN-*subbus*. *Header* selalu bersifat *write* artinya *Header* selalu berasal dari *Master* melalui *time scheduling*. Sedangkan *Response* selalu muncul setelah *Header*. Baik *Response* yang bersifat *write* (dari *Master* ke *Slave*) maupun yang bersifat *read* (dari *Slave* ke *Master*).



Gambar 6. Format *Frame*

Dalam suatu *Header* terdapat empat buah informasi yang masing-masing sepanjang satu *byte*. Yang pertama adalah *Address Field* dan *Read* / *Write Bit*. *Address Field* dan *Read* / *Write Bit* ini mengikuti standar I^2C dan bersifat tetap. Yaitu 0x04 untuk fungsi *write* dan 0x05 untuk fungsi *read*. Alamat ini dipilih karena alamat inilah yang dicadangkan oleh Phillips untuk aplikasi sistem *bus* berbasis I^2C dengan format lain.

Berikutnya adalah *Sync Field*. *Sync Field* berfungsi sebagai konstanta penanda pada *Header*. Konstanta yang digunakan yaitu 0xAA.

Informasi ketiga adalah *Attributes Field*. *Attributes Field* sendiri berisi dua buah informasi yaitu *Slave Address* dan *Data Length*. Masing-masing selebar tiga *bit*. Sehingga mampu mengalamati sebanyak delapan buah *Slave* dan tujuh *byte data* dalam satu buah *Frame*.

Sedangkan informasi terakhir pada *Header* adalah *Header Checksum*. Informasi ini berfungsi untuk mendeteksi adanya kesalahan pada *Header*. Algoritma yang digunakan adalah fungsi XOR (*Exclusive* OR) pada *Address Field*, *Sync Field* dan *Attributes Field*.

Response yang mengikuti *Header*, berisi *Data Field* yang sepanjang 0 hingga 7 *byte* dan satu buah *Data Checksum*. Panjang *Data Field* ini ditentukan oleh nilai pada *Attributes Field*. Panjang minimal dari *Response* yaitu 1 *byte* (*Data Checksum*) dan panjang maksimal yaitu 8 *byte* (7 *Data Field* + *Data Checksum*).

3.2 Implementasi pada *Hardware*

Tahap berikutnya adalah pengujian sistem *bus* pada *hardware* secara nyata. Program yang diaplikasikan terdapat dua macam. Yaitu pengiriman data antar dua buah *Slave* melalui satu buah *Master*. *Data* akan terkirim akan ditampilkan pada *Slave* yang berlawanan.

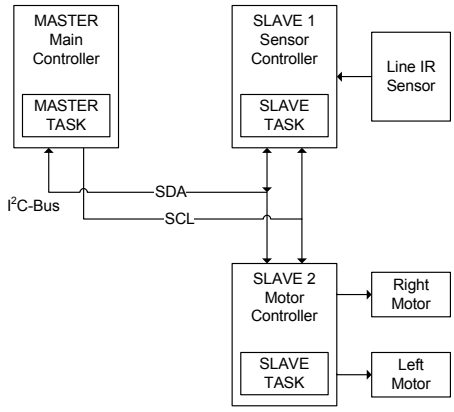
Yang kedua adalah pengujian dengan modul *Slave* yang telah dilengkapi dengan *push button* dan *motor driver*. *Slave* dengan *push button* mewakili *Slave* yang terhubung dengan *sensor* dan *Slave* dengan *motor driver* mewakili *Slave* yang terhubung dengan aktuator.

3.3 Implementasi pada Model Robotik *Wheeled Line-Follower*

Setelah pengimplementasian pada *hardware* sesuai dengan rancangan, tahap terakhir adalah implementasi protokol pada model robotik *Wheeled Line-Follower*.

Behaviour dan *task* robot ditulis pada *software* di *Application Layer*. *Task* tersebut terbagi menjadi tiga buah. Yaitu *sensory*, *decision making* dan *actuating*. Ketiga buah *task* ini akan membentuk suatu *behaviour* robot yaitu bergerak mengikuti garis.

Diagram blok model robotik terdiri dari tiga buah modul utama yaitu *Master* sebagai *Main Controller*, *Slave* 1 sebagai *Sensor Controller* dan *Slave* 2 sebagai *Motor Controller*.



Gambar 7. Diagram Blok Model Robotik

Pada sistem di atas, pertama kali *Master* meminta *data* dari *Slave* 1 (*request*). Kemudian *Slave* 1 mengirimkan *data* posisi *line* yang dibaca dari *sensor*. Setelah *Master* memperoleh posisi *line* dari *Slave* 1, *Master* menentukan apa yang harus dilakukan oleh *Slave* 2 sebagai *Motor Controller*. Berikutnya adalah, *Slave* 2 menggerakkan motor kanan dan motor kiri sesuai dengan perintah yang dikirimkan oleh *Master*. Hal yang penting pada sistem ini adalah timbulnya *delay* antara pembacaan pada *sensor* hingga penggerakan motor yang disebut dengan *process delay*.

4 PENGUJIAN DAN ANALISA SISTEM

Pengujian dan analisa dilakukan dalam tiga tahap. Yang pertama adalah pengukuran sistem *bus* pada *simulator* dan alat ukur *osiloskop* / *logic analyzer*. Berikutnya adalah pengamatan *behaviour* pada model robotika *Wheeled Line-Follower*. Dan yang terakhir adalah analisa dari I^2C serial *bus*.

4.1 Pengukuran Tegangan dan Pewaktuan I^2C *serial bus* dari Simulasi dan Alat Ukur

Simulasi dan pengukuran dilakukan dengan *osiloskop* / *logic analyzer*.

Tabel 1. Hasil Pengukuran Pewaktuan

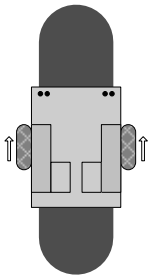
Parameter	Simbol	Standar resmi I2C Serial Bus		Hasil Pengukuran	
		Minimal	Maksimal	Simulator	Digital Oscilloscope
Low level input voltages V_{DD}-related input levels	VIL	-0.5	$0.3V_{DD}$	48.24 mV	
High level input voltages V_{DD}-related input levels	VIH	$0.7V_{DD}$	$V_{DD max}$ + 0.5 V	5.05 V	
Low level output voltage (open-drain or open-collector) at 3 mA sink current	VOL	0	0.4 V	4.71 V	4.71 V
Output fall time from V_{IHmin} to V_{ILmax} with a bus capacitance from 10pF to 400pF	tof	-	250 nS	<250 nS	<80 nS
Input current for each I/O pin with an input voltage between 0.1V_{DD} and 0.9V_{DDmax}	Ii	-10 uA	10 uA	± 10 uA	± 10 uA
Capacitance for each I/O pin	Ci	-	10 pF	10 pF	10 pF
SCL clock frequency	fSCL	0	100 kHz	95238,80 Hz	95057,93 Hz
Hold time START condition	tHD:STA	4 uS	-	7,50 uS	7,50 uS
SCL LOW period	tLOW	4,7 uS	-	5,50 uS	5,52 uS
SCL HIGH period	tHIGH	4,0 uS	-	4,00 uS	4,00 uS
Data hold time	tHD:DAT	0	3,45 uS	2,50 uS	2,48 uS
Data set-up time	tSU:DAT	250 ns	-	4,00 uS	4,00 uS
Rise time of both SDA and SCL signals	tr	-	1 uS	<1 uS	<80 nS
Fall time of both SDA and SCL signals	tf	-	300 nS	<300 nS	<80 nS
Set-up time for STOP condition	tSU:STO	4 uS	-	7,50 uS	7,50 uS
Bus free time between a STOP and START condition	tBUF	4,7 uS	-	16,50 uS	16,56 uS
Capacitive load for each bus line		-	400 pF	30 – 99 pF	30 – 99 pF

4.2 Pengamatan Model Robotika *Wheeled Line-Follower*

Model robotika *Wheeled Line-Follower* mempunyai dua buah sensor infra merah dan pemancar infra merah serta dua buah motor DC. Sensor berfungsi sebagai pendeteksi jalur atau garis. Apakah berwarna gelap atau terang. Jika garis berwarna terang, maka cahaya akan dipantulkan ke sensor. Dan jika garis berwarna gelap, maka hanya sedikit cahaya yang akan dipantulkan ke sensor.

Robot akan bergerak maju atau mundur apabila kedua motor bergerak searah. Dan jika kedua motor bergerak berlawanan arah, maka robot akan berbelok dengan cara berputar dengan pusat putaran pada tengah robot.

Dari hasil pengamatan, motor robot akan bergerak sesuai dengan kondisi dari kedua sensor. Jika kedua sensor memperoleh garis berwarna gelap, maka kedua motor akan bergerak maju. Namun jika kedua sensor memperoleh garis berwarna terang, maka kedua motor akan berhenti.

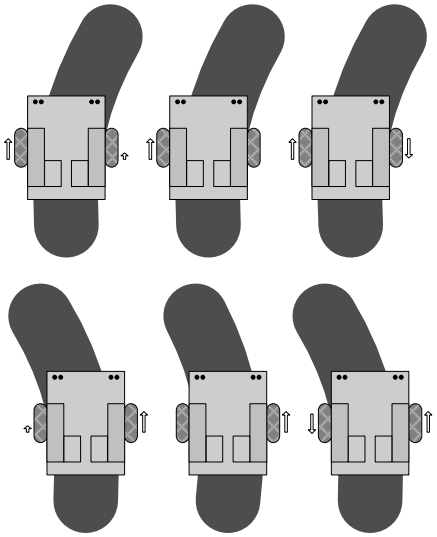


Gambar 8. Gerakan motor ketika kedua sensor mendeteksi garis hitam

Sedangkan jika sensor kanan memperoleh warna terang dan sensor kiri memperoleh warna gelap, maka motor kanan akan bergerak maju dan motor kiri akan bergerak mundur. Sehingga robot akan berputar ke kiri.

Namun jika sensor kanan memperoleh warna gelap dan sensor kiri memperoleh warna terang, maka motor kanan akan bergerak mundur dan motor kiri akan bergeran maju. Sehingga robot akan berputar ke kanan.

 ISSN: 1907-5022



Gambar 9. Gerakan motor ketika sensor mendeteksi belokan

Ketika diuji dengan jalur garis hitam di atas dasar warna putih, robot dapat bergerak sesuai dengan garis. Namun ketika berbelok terkadang robot gagal mengikuti lekukan garis. Hal ini disebabkan oleh kecepatan motor yang cukup tinggi namun tidak diimbangi dengan kecepatan respon sensor-aktuator yang sama cepat karena adanya *delay* sensor-aktuator.

4.3 Analisa Protokol I^{2}C *serial bus*

Secara keseluruhan protokol yang telah diimplementasikan berbasis I^{2}C *serial bus* sudah sesuai dengan spesifikasi pada dokumen resmi I^{2}C *serial bus*. Meski ada beberapa parameter yang cukup kritikal. Namun dapat diatasi dengan pengaturan atau *tuning* pada *software* inti I^{2}C *serial bus*.

Model robotika *Wheeled Line-Follower* berjalan sesuai dengan perancangan dan ekpektasi awal dimana ketiga *controller* dapat saling berkoordinasi melaksanakan *behaviour* robot yang telah ditentukan. Hambatan dan kekurangan pada model robotika muncul dari sisi mekanika dan elektronik yang belum betul-betul sinkron.

5 KESIMPULAN

5.1 Kesimpulan

Berdasarkan hasil pengujian dan analisa dapat diambil beberapa kesimpulan.

- I^{2}C *serial bus* yang dikeluarkan oleh Phillips dapat digunakan sebagai sistem *bus* penghubung antar *processor* atau *controller* dalam sebuah sistem kontrol.
- Sistem *bus* tersebut mempunyai *delay Frame* sekitar 13 – 15 uS untuk sekali operasi baca – tulis dan dapat digunakan dalam aplikasi kontrol yang tidak membutuhkan kecepatan tinggi dan dalam skala kecil.
- Hambatan pada model robotika yang muncul adalah sisi mekanik terutama motor yang cukup cepat yang tidak diimbangi dengan kecepatan respon dari sistem.
- Implementasi *software* inti pada *Physical* dan *Transport Layer* membutuhkan ruang program sebesar 392 *byte* (9.57% dari 4 kB) pada *Master* dan 565 *byte* (13.79% dari 4 kB) pada *Slave*. Sedangkan *bit rate* tertinggi yang diperoleh sebesar 97938.14 bps pada *clock microcontroller* sebesar 24 MHz atau 2 MIPS.

5.2 Saran

Berdasarkan hasil yang dicapai dalam Penelitian ini, penulis menyarankan beberapa hal untuk pengembangan lebih lanjut.

- *Microcontroller* keluarga MCS-51 dapat diganti dengan *microcontroller* yang lebih cepat seperti Atmel AVR. Di mana beberapa Atmel AVR sudah mempunyai I^{2}C *serial bus controller on-chip*. Sehingga dapat menghemat penggunaan ruang program pada ROM.
- Pengimplementasian *Asymmetric Processor* sebagai *watchdog* atau pengawas pada sistem *bus* dan *Master*. Hal ini perlu dilakukan karena jika terjadi kegagalan pada *Master* maka seluruh sistem akan gagal.
- Disarankan penggunaan *gear box* yang mempunyai perbandingan yang lebih besar. Sehingga putaran motor akan lebih kecil. Hal ini dilakukan untuk mengimbangi *delay* pada sistem.

PUSTAKA

[1] Corp., Atmel. (1999). *Flash Microcontroller Architectural Overview*. San Jose: Atmel Corporation.

[2] Corp., Atmel. (1998). *MCS-51 Instruction Set*. San Jose: Atmel Corporation.

[3] Dominik Henrich and Thomas Honiger. (1997). *Parallel Processing Approaches In Robotics*. Institute of Real-Time Computer Systems and Robotics, University of Karlsruhe, Karlsruhe, Germany.

[4] Eko Putra, Agfianto. (2003). *Belajar Mikrokontroller AT89C51/52/55 Teori dan Aplikasi*. Yogyakarta: Gava Media.

[5] Electronics, Koninklijke Philips. (2000). The *I^{2}C Bus Specification Version 2.1 January 2000*. Netherland: Koninklijke Philips Electronics.

[6] Irazabal, Jean-Marc and Blozis, Steve. (2003). *Application Note AN10216-01 I^{2}C Manual*. Netherland: Koninklijke Philips Electronics.