Jurnal Teknik Elektro Vol. 10 No. 2
Juli - Desember 2018

P-ISSN 1411 - 0059
E-ISSN 2549 - 1571

Implementasi Algoritma Kalkulasi Interupsi pada Rancang Bangun *Tachometer* Digital

Putut Son Maria[1] dan Elva Susianti[2]

[1]*Jurusan Teknik Elektro, Fakultas Sains dan Teknologi, Universitas Islam Negeri Sultan Syarif Kasim Riau*

[2]*Jurusan Teknik Elektronika, Politeknik Caltex Riau*

putut.son@gmail.com[1], elva@pcr.ac.id[2]

Abstract— Tachometer is an important tool for measuring the rotational speed of electro-mechanical machines and other rotating objects. In a closed-loop system, it should have a mechanism by which sensory instruments must be able to produce and deliver feedback values that can be fetched by the controller. Most commercial tachometers do not have such data communication features, making it difficult to be employed in closed loop systems. This study aims to develop a tachometer which is programmed using calculation on interrupt-based algorithm and has data communication capabilities as well. The algorithm was designed over the minimum features of a microcontroller chip, namely a timer and interrupt. The proposed algorithm that was tested on the prototype results good performance for the typical electro-mechanical machines with a measurement error range between 0.1 and 1.2%. The equation for the calculation of rotation per minute (rpm) has also been revised to provide a more valid formula.

Keywords— Digital tachometer, interrupt, data communication

Abstrak— *Tachometer* menjadi alat yang penting untuk pengukuran kecepatan rotasi mesin elektro-mekanik dan juga obyek gerak lainnya. Pada sistem lup tertutup, secara teori wajib ada mekanisme dimana instrumen sensorik harus mampu menghasilkan dan mengirimkan nilai umpan balik yang dapat dibaca oleh kontroler. Sebagian besar *tachometer* komersil tidak memiliki fitur untuk komunikasi data, sehingga sulit digunakan dalam sistem lup tertutup. Penelitian ini bertujuan untuk membangun *tachometer* yang terprogram menggunakan algoritma kalkulasi interupsi dan memiliki kemampuan komunikasi data. Algoritma dirancang atas latar belakang fitur minimum sebuah *chip* mikrokontroler yaitu *timer* dan interupsi. Algoritma baru yang diuji-cobakan pada prototipe terbukti menghasilkan kinerja yang baik untuk rentang *rotation per minute* (rpm) pada tipikal mesin elektro-mekanik dengan *error* hasil pengukuran antara 0,1 sampai 1,2 %. Perumusan kalkulasi untuk perhitungan rpm juga telah direvisi sehingga menghasilkan formula yang lebih valid.

Kata kunci— *Tachometer* digital, interupsi, komunikasi data

I. PENDAHULUAN

Tachometer adalah alat untuk mengukur kecepatan rotasi pada mesin elektrik dan mekanik. Alat ini dibutuhkan dan digunakan pada banyak bidang seperti pada industri otomotif, *plant* tenaga listrik, laboratorium sistem tenaga dan laboratorium sistem kendali [1], [2]. Pada sistem lup tertutup seperti pada praktikum motor servo kecepatan, *tachometer* menjadi alat yang wajib yang berfungsi mengukur dan memberikan nilai *feedback* kepada kontroler.

Beberapa *tachometer* komersil yang dijual di beberapa *marketplace* pada umumnya dibangun berbasis *single mounting chip* dan hanya menampilkan hasil pengukuran pada perangkat *display* sehingga *tachometer* tersebut tidak dapat diposisikan sebagai instrumen *feedback* pada sebuah sistem otomatis. Beberapa *tachometer* yang dilengkapi dengan fitur komunikasi data secara ekonomi berharga cukup mahal dan garansi yang berbatas waktu akan menyulitkan pada saat terjadi kerusakan pada alat.

Jenis *tachometer* dapat dibedakan berdasarkan sistem pengolahan sinyalnya dan sistem kopling terhadap obyek bergerak [3]. *Tachometer* digital memiliki akurasi yang lebih baik dan mudah dibangun jika dibandingkan dengan *tachometer* analog. Beberapa peneliti lain telah membangun *tachometer* digital menggunakan *platform* yang berbeda seperti pada [4]. Komponen FPGA sesuai untuk aplikasi sistem yang sifatnya *fixed* atau permanen untuk mereduksi kompleksitas *hardware* sehingga *damaged cost* dari sistem lebih murah. Namun demikian beberapa faktor seperti perangkat *programmer chip* FPGA relatif jarang, ketersediaan *chip* FPGA tidak sebanyak komponen mikrokontroler dan karakternya yang kurang fleksibel menjadikan FPGA kurang populer untuk penelitian yang memerlukan proses hapus-tulis program yang intens. Peneliti pada [5] dan [6] menggunakan Arduino Uno untuk membangun *tachometer* yang hanya menampilkan fitur *display*. Fitur tersebut terlalu minim dibandingkan kapabilitas *resource* Arduino Uno sehingga sistem yang dibangun tidak optimal dari segi teknis, padahal kapabilitas dari Arduino Uno dapat dieksplore lebih maksimum dengan menambahkan fitur lain. Peneliti [7] membangun *tachometer* menggunakan mikrokontroler ATMEGA16. Penggunaan varian tersebut telah sesuai untuk

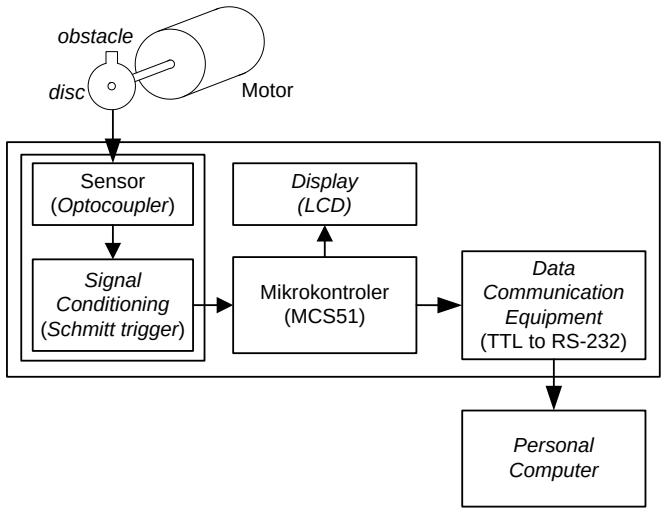
sistem yang *compact* dan *portable*, namun demikian penelitian tersebut tidak mengimplementasikannya secara *real time* sehingga faktor fungsionalitas dari *tachometer* kurang signifikan.

Tachometer digital banyak dibangun menggunakan sistem kopling secara *non-contact* pada *spindle* motor, sehingga kinerja *tachometer* dapat diandalkan karena mesin yang sedang bergerak tidak terbebani dengan adanya sistem sensorik dari *tachometer*. Penelitian ini bertujuan untuk membangun *tachometer* digital secara *non-contact* yang memiliki fitur komunikasi data sehingga data hasil pengukuran dapat direkam oleh komputer. Algoritma yang diprogramkan pada penelitian ini berlatar belakang pada tersedianya fitur *timer* dan interupsi serta komunikasi serial standar, yang mana ketiganya merupakan fitur konvensional dan pasti dimiliki oleh mikrokontroler generasi terkini.

II. METODOLOGI PENELITIAN

A. Blok Diagram Sistem

Gambar 1 menunjukkan blok diagram sistem perangkat keras pendukung pada penelitian ini. *Plant* berupa motor dimana *shaft spindle* motor dipasangi piringan dengan *obstacle* berjumlah 1 buah. Sensor *optocoupler* bertugas untuk menterjemahkan *event* menjadi sinyal logika. *Event* yang dimaksud di sini adalah kondisi dimana pada saat posisi *obstacle* tepat berada di antara celah *optocoupler* (sensor) maka terjadi perubahan *state* pada transistor *optocoupler* yang dimanifestasikan menjadi logika 0 ke 1 atau sebaliknya. Sinyal dari sensor diteruskan ke *signal conditioning* yang bertugas untuk mem-filter *bouncing* yang terjadi akibat transisi logika. Keluaran dari *signal conditioning* diteruskan ke mikrokontroler untuk memicu terjadinya interupsi. Mikrokontroler bertugas untuk melakukan perhitungan *rotation per minute* (rpm), menampilkan informasi rpm ke *liquid crystal display* (LCD) dan mengirim data rpm melalui *port serial*.



Gambar 1. Blok diagram sistem

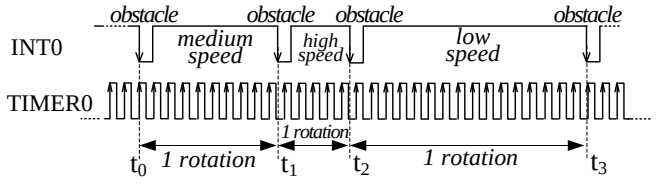
Data communication equipment bertugas untuk mengubah level tegangan TTL dari *port serial* menjadi level RS-232. Data dengan level standar RS-232 inilah yang dapat dibaca oleh komputer untuk direkam dan ditampilkan menjadi grafik.

Pemasangan *disc* dengan *obstacle* memungkinkan sensor untuk tidak perlu terhubung langsung ke *spindle* motor, hal ini karena sensor hanya cukup menunggu sampai *obstacle* melewati celah sensor. Cara tersebut digunakan agar tidak terjadi pembebanan pada motor karena massa *disc* dan *obstacle* sangat ringan sehingga dapat diabaikan.

B. Algoritma Kalkulasi Interupsi

Minimal terdapat empat algoritma yang dapat di-*coding* untuk perhitungan rpm yaitu *direct pulse counting* (DPC), *single pulse time measurement* (SPTM), *constant elapsed time* (CET), dan *pulse time measurement using a variable number of counted pulse* (PTM-UVNCP) [8]. Algoritma DPC mudah dalam pemrogramannya tetapi akurasi perhitungan sangat tergantung pada jarak spasial *encoder*, lebar *obstacle* dan jumlah *encoder*, sehingga *error* semakin besar pada pengukuran rpm tinggi. Algoritma SPTM membutuhkan penyesuaian ulang apabila *obstacle* untuk *encoder* dipasang pada *plant* berbeda karena rasio spasial *obstacle* relatif terhadap radius *shaft* obyek bergerak. Pada algoritma CET hasil pengukuran dikalkulasi dari rerata hasil *sampling* yang dilakukan pada interval waktu tertentu, kelemahannya adalah adanya waktu tunggu atau hasil baru akan didapat setelah waktu *sampling* usai. Metode ini rentan terhadap munculnya *error* pada putaran yang rendah demikian juga pada algoritma PTM-UVNCP akan sulit menentukan nilai koefisien dari *denominator* pada persamaannya. Pada penelitian ini merancang dan menguji algoritma baru yaitu algoritma kalkulasi interupsi (KI) yang dirancang berdasarkan minimnya fitur *hardware* mikrokontroler sehingga nantinya algoritma ini dapat di-*coding*-kan pada kontroler lain dengan *platform* yang berbeda.

Ide dasar dari algoritma KI ditunjukkan seperti pada Gambar 2. *Premis* yang digunakan untuk algoritma KI adalah menggunakan *optocoupler* dengan jumlah *obstacle* reflektif hanya 1 buah yang direkatkan pada *spinde* dan menggunakan fitur *timer* untuk menghasilkan durasi waktu periodik konstan setiap 1 milidetik. Setiap terjadi *overflow* oleh *timer*, maka akan memicu interupsi bagi mikrokontroler, sehingga dalam rentang waktu 1 detik akan terjadi 1000 interupsi oleh *timer* (dalam keadaan normal). Pada rentang waktu t_0 sampai t_1 adalah kecepatan menengah, jumlah interupsi oleh *timer* terjadi sebanyak ***n*** kali.



Gambar 2. Interval pulsa *obstacle* bervariasi

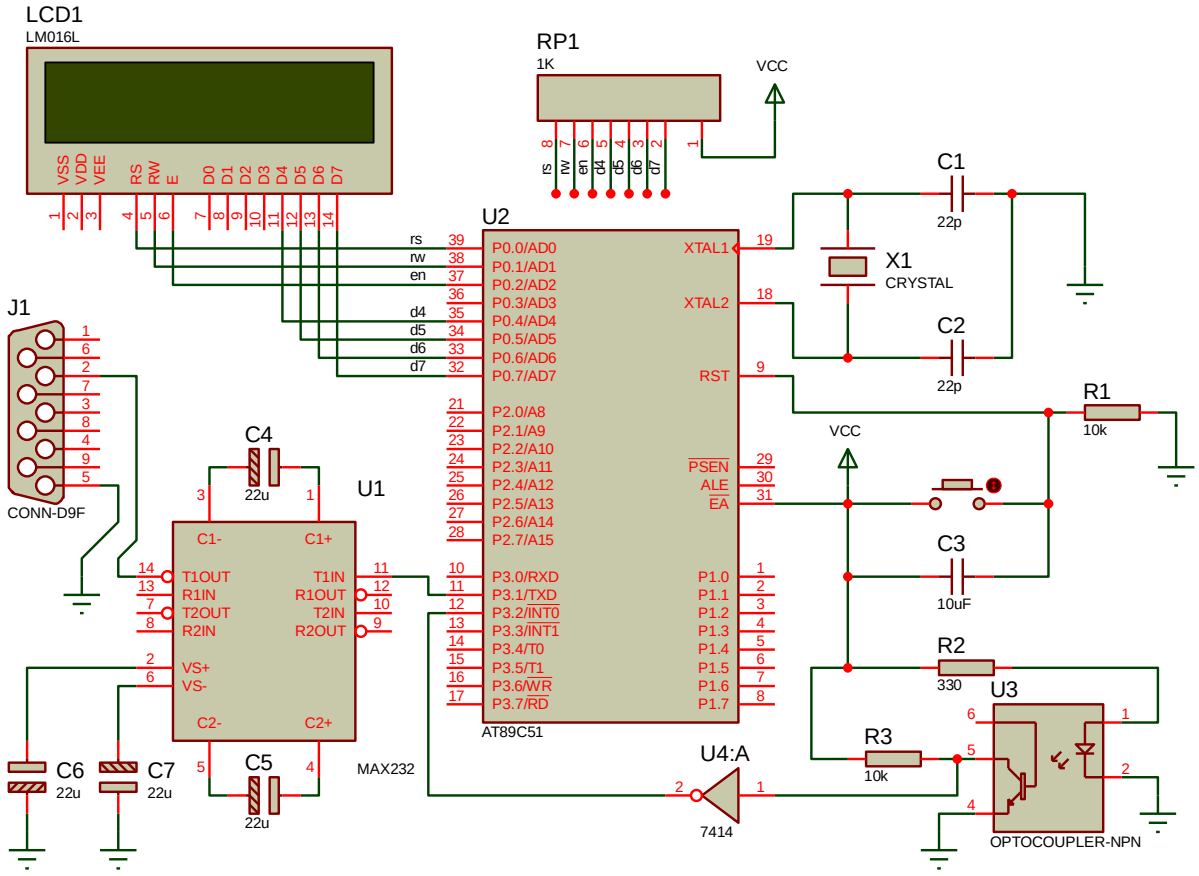
Pada saat kecepatan *spindle* bertambah (rentang waktu t_1 sampai t_2), maka interupsi eksternal akan semakin cepat sehingga jumlah interupsi *timer* yang terhitung akan berjumlah < ***n***. Sebaliknya pada saat putaran *spindle* sangat lambat(t_2 sampai t_3), maka jumlah interupsi *timer* akan semakin banyak(> ***n***). Setiap satu siklus putaran *spindle*, maka variabel penghitung interupsi *timer* harus di-nol-kan agar nilainya tidak terakumulasi dengan perhitungan dari siklus sebelumnya. Setiap satu putaran *spindle* maka mikrokontroler akan menghitung nilai rpm berdasarkan persamaan (1). Persamaan tersebut menggunakan konstanta pengali 60 sebagai konversi putaran per detik menjadi putaran per menit.

$$\text{rpm} = \frac{1000 \text{ milidetik}}{\sum \text{interupsi}} \; x \; 60 \qquad (1)$$

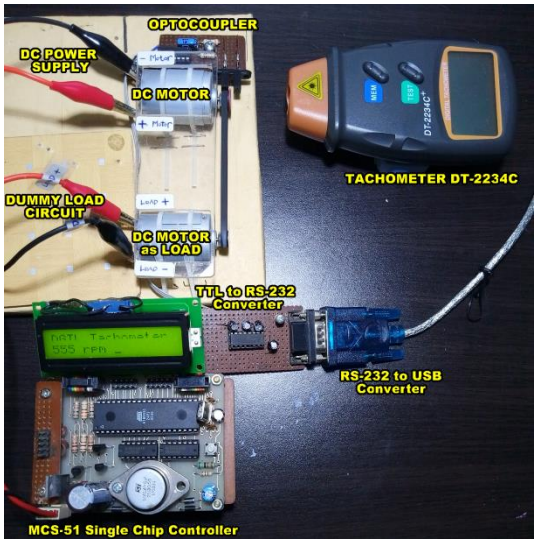
C. Skema Rangkaian dan *Set Up* Eksperimen

Gambar 3 menunjukkan *wiring diagram tachometer* pada penelitian ini. Penggunaan mikrokontroler MCS-51 sengaja dipilih karena memiliki fitur *timer*, interupsi, dan komunikasi data serial serta masih banyak *marketplace* yang menjual *chip* ini dengan harga yang murah. Selisih harga antara *chip* MCS-51 varian 28 pin dengan 40 pin adalah tidak signifikan, sehingga pemilihan varian 40 pin memberikan keuntungan apabila suatu ketika dibutuhkan penambahan fitur, maka masih tersedia pin I/O yang *reserved*. Pada bagian tranduser digunakan *optocoupler* yang *output*-nya dikaskade dengan gerbang NOT. *Chip* gerbang NOT yang digunakan adalah 74HC14 dimana IC ini memiliki mekanisme *schmitt trigger* yang membantu untuk meredam gejala *bouncing* yang muncul saat transisi *on-off* pada *optocoupler*. *Output* dari gerbang NOT terhubung ke pin INT0 dari mikrokontroler, sehingga pada saat posisi *obstacle* berada di tengah-tengah *optocoupler* maka akan memicu adanya interupsi bagi kontroler.

Perangkat komunikasi data sebagai jalur transmisi menggunakan IC MAX232 yang berfungsi mengkonversi *level* TTL menjadi *level* RS-232 agar sesuai dengan standar komunikasi RS-232. Hasil pengukuran rpm ditampilkan pada LCD dan juga dikirimkan *via* komunikasi serial. *Setting baud rate* (bps) ditetapkan sebesar 9600 bps, 8 bit, dan *no parity*. Gambar 4 menunjukkan *set up* eksperimen dimana obyek *plant* menggunakan 2 unit motor DC dimana 1 unit sebagai motor penggerak dan 1 unit difungsikan sebagai *dummy load*.



Gambar 3. Rangkaian *tachometer* berbasis mikrokontroler MCS-51

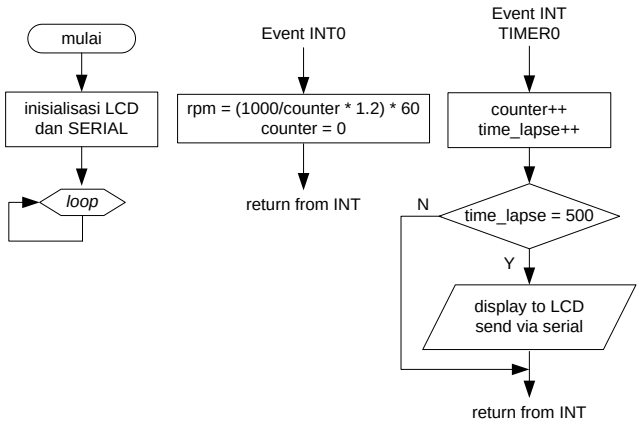


Gambar 4. *Set up* untuk eksperimen

Motor DC penggerak diberikan catu daya pada kondisi *rating*-nya dan terhubung menggunakan *belt* ke motor DC *dummy load*. Semakin kecil nilai beban resistif pada *dummy load* maka semakin besar efek pembebanan motor penggerak sehingga kecepatan putarnya menjadi lebih lambat.

D. *Flowchart* Program

Sesuai dengan ilustrasi *time chart* pada Gambar 2, sebagian besar waktu yang digunakan oleh kontroler adalah untuk menghitung banyaknya interupsi yang terjadi oleh *timer* dan melayani interupsi eksternal (*obstacle*). Pemrograman pada kontroler terdiri dari 3 bagian yaitu program utama, rutin vektor interupsi *timer* (*TIMER*0) dan rutin vektor interupsi eksternal (INT0). Gambar 5 menunjukkan *flow chart* program pada penelitian ini. Program utama berisi kode untuk inisialisasi terhadap LCD, register serial dan pengaturan untuk interupsi. Setelah itu kontroler akan *stand by* sampai terjadinya interupsi. Interupsi eksternal terjadi pada saat *obstacle* menghalangi *optocoupler*. Setiap terjadi interupsi eksternal maka kontroler akan menghitung rpm berdasarkan persamaan (1) dan mereset nilai dari *counter*. Mekanisme reset tersebut diperlukan karena hitungan harus dimulai dari nol untuk putaran spindle berikutnya atau untuk periode T_n + 1. Kontroler akan mengeksekusi rutin pada vektor interupsi eksternal ataupun internal adalah berdasarkan *event triggered*, sehingga penggambaran *flow chart* tidak dapat digabungkan dengan alur program utama, melainkan membentuk segmen program pada *range memory code* tertentu sesuai dengan nomor interupsinya. *Task* yang dikerjakan oleh kontroler jika terjadi interupsi *timer* adalah menaikkan hitungan *counter* dan mengirim data *via* serial bilamana telah mencapai batas waktu setiap 0,5 detik. Secara ideal interupsi *timer* terjadi setiap 1 milidetik, sehingga untuk mencapai 0,5 detik maka diperlukan variabel yang mencatat setiap kelipatan 500 x interupsi *timer*.



Gambar 5. *Flow chart* program

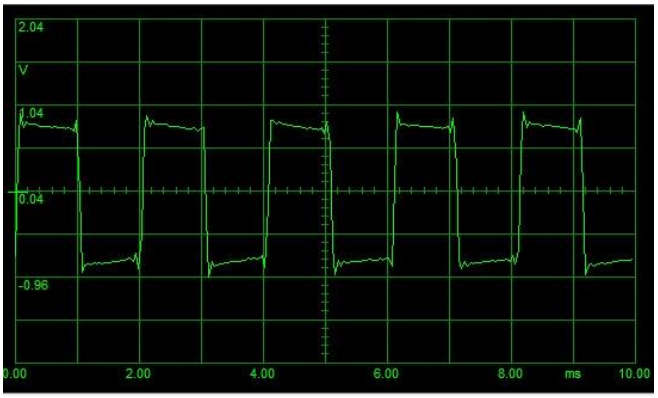
III. HASIL DAN PEMBAHASAN

A. Konsistensi *Timer*

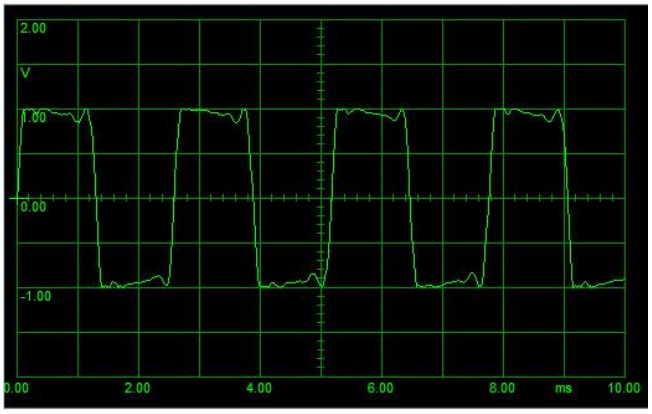
Konsistensi *timer* untuk menghasilkan durasi waktu 1 milidetik adalah krusial, karena jika interval dan durasi *timer* tidak konsisten maka persamaan (1) menjadi tidak valid. Dari perancangan *hardware* sebelumnya diketahui bahwa mikrokontroler harus menangani interupsi dari 2 sumber interupsi yaitu INT0 dan *TIMER*0. Oleh karena itu pengujian *timer* perlu dilakukan untuk 2 kondisi yaitu *timer-code only* dan *full-code program*. *Timer-code* adalah mikrokontroler yang diprogram hanya berisi rutin untuk vektor interupsi *timer* dan *full-code* adalah mikrokontroler yang diprogram dengan melibatkan semua rutin yang diperlukan untuk membentuk fungsi *tachometer*.

Gambar 6 menunjukkan pulsa periodik yang dirancang untuk menghasilkan durasi dan interval 1 milidetik. Pulsa tersebut dimanifestasikan via port 3 bit 0 dari mikrokontroler. Dengan frekuensi osilator 11,059200 Mhz dan frekuensi *timer* adalah 1/12 dari frekuensi osilator dan mode *timer* 16 bit maka isi register TH0 dan TL0 adalah FC hex dan 64 hex. Hasilnya *timer* dapat bekerja secara konsisten.

Pada saat mikrokontroler diprogram secara *full-code*, maka terjadi anomali seperti ditunjukkan pada Gambar 7. Durasi pulsa menjadi lebih lebar sekitar 1,2 milidetik dengan interval yang masih konsisten. Hal ini terjadi karena *program counter* dari mikrokontroler sangat sibuk menangani rutin interupsi dari *timer* dan INT0 sekaligus. Pada rutin INT0 terdapat perintah untuk melakukan perhitungan dan mereset counter. *Task-task* tersebut membutuhkan waktu, sehingga tambahan *task* pada rutin INT0 mengakibatkan eksekusi terhadap rutin *timer* menjadi lebih lambat seperti terlihat pada Gambar 7.



Gambar 6. Pulsa periodik *on-off* oleh *timer* (*timer-code only*)



Gambar 7. Pulsa periodik *on-off* oleh *timer* (*full-code*)

Dengan adanya hasil pengujian tersebut, maka formula pada persamaan (1) harus direvisi menjadi seperti persamaan (2). Konstanta 1,2 pada persamaan (2) adalah sebagai konsekuensi bahwa ternyata setelah kontroler diprogram secara *full-code* efeknya adalah terjadi pelebaran durasi waktu dari 1 milidetik menjadi 1,2 milidetik.

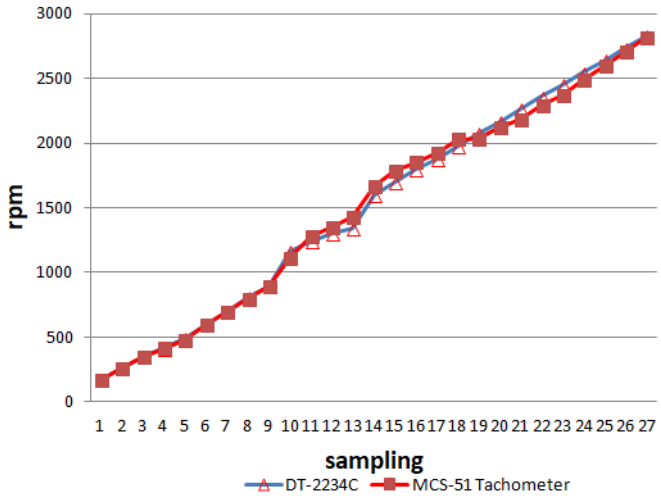
$$\text{rpm} = \frac{1000 \text{ milidetik}}{\sum \text{interupsi } x\ 1{,}2}\ x\ 60 \qquad (2)$$

B. Akurasi

Pengujian akurasi dilakukan dengan cara mengukur putaran *spindle* motor penggerak menggunakan *tachometer* DT-2234C dan *tachometer* prototipe (MCS-51 *tachometer*) pada waktu yang bersamaan. Eksperimen dilaksanakan terbatas pada rentang rpm dimana sebagai acuannya adalah data yang tertera pada *name-plate* mesin-mesin listrik di laboratorium mesin listrik, sistem tenaga dan sistem kendali yang mana mesin-mesin tersebut bekerja pada kecepatan maksimal 3000 rpm (*nominal rating speed*).

DT-2234C adalah *tachometer* komersil dengan spesifikasi akurasi 0,1 % dan resolusi 1 rpm. Dari Gambar 8 terlihat bahwa *trend* kurva hasil pengukuran DT-2234C dan prototipe menunjukkan karakter yang serupa. Pada kecepatan rendah prototipe mampu bekerja dengan baik dengan *error* yang kecil sampai kecepatan sekitar 1300 rpm. *Error* membesar pada rentang kecepatan 1700 rpm sampai 1850 rpm kemudian kembali mengecil mulai kecepatan di atas 1850 rpm. Hasil pengamatan menunjukkan gejala tersebut terjadi karena beberapa faktor diantaranya: kecepatan motor yang tidak *steady* pada rentang kecepatan tersebut dan ditambah dengan waktu *sampling* prototipe yang lebih cepat. *Sampling time* DT-2234C adalah 0,8 detik sedangkan prototipe melakukan *sampling* setiap 1 putaran *spindle* motor. Karena *sampling time* prototipe adalah berdasarkan *event*, maka pada saat terjadi gejala *transient* akan dianggap sebagai data baru dan langsung dihitung oleh *controller*, akibatnya adalah angka hasil hitungan lebih besar dibandingkan pada saat kecepatan motor telah *steady*. Perhitungan pada saat terjadi gejala *transient* adalah normal menghasilkan angka yang berbeda jika dibandingkan dengan 0,8 detik setelah terjadinya *transient*.

Sampling time yang berdasarkan *event* juga berarti bahwa kemampuan prototipe untuk melakukan *update* data sangat fleksibel atau cepat, namun demikian pada penelitian ini informasi rpm dan pengiriman data ditetapkan setiap 0,5 detik. Data menunjukkan bahwa *error* maksimum dari prototipe adalah 1,2 %. Dengan asumsi bahwa DT-2234C adalah sudah terkalibrasi dan akurat, maka dapat diklaim bahwa prototipe memiliki akurasi yang cukup baik dengan rentang *error* antara 0,1% sampai 1,2%.



Gambar 8. Grafik akurasi perbandingan *tachometer* DT-2234C dan prototipe (MCS-51 *tachometer*)

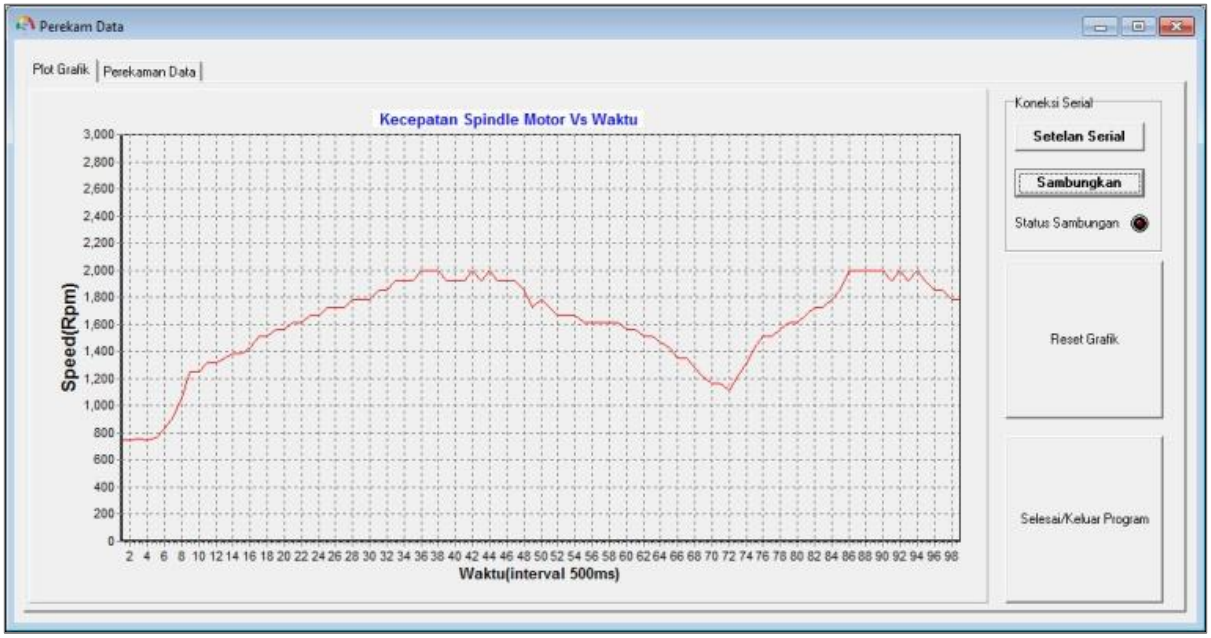
C. Komunikasi Data

Fitur komunikasi data adalah parameter penting yang ingin dicapai dalam penelitian ini untuk mengkonter aksi kekurangan pada *tachometer komersil*. Untuk itu perlu instrumen untuk melihat keberhasilan komunikasi data seperti yang diharapkan.

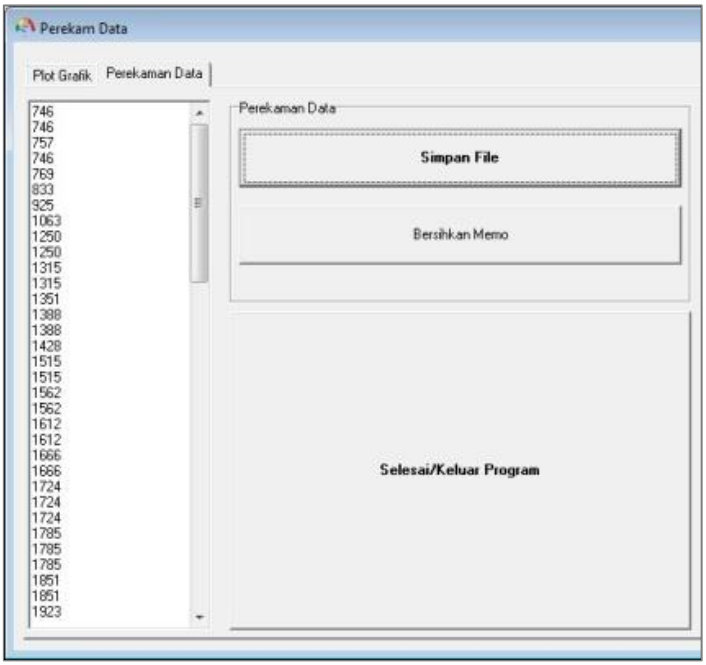
Pada penelitian ini juga dibangun program aplikasi untuk *desktop* menggunakan bahasa program pascal dengan *compiler* lazarus ver 1.2 dan *library* comport ver 4.11. Data yang dikirimkan oleh *tachometer* prototipe adalah standar serial konvensional, dengan demikian perlu perangkat *RS-232*

to USB converter agar data serial dapat dibaca oleh komputer/laptop yang tidak memiliki terminal serial DB-9 konvensional. Gambar 9 menunjukkan *plot* grafik yang diterima oleh komputer dari hasil pembacaan *tachometer* prototipe terhadap kecepatan motor yang bervariasi. Data dari *tachometer* prototipe dikirim *via* terminal serial setiap 0,5 detik dan interval pengiriman ini dapat diubah sesuai kebutuhan. Kemudahan untuk memodifikasi program tersebut adalah poin positif dimana hal tersebut tidak dapat dilakukan pada *tachometer* komersil. Untuk menambah nilai fungsionalitas dari program aplikasi, disediakan juga fitur tampilan numerik dan penyimpanan data dalam bentuk file berformat TXT.

Gambar 10 menunjukkan *interface* untuk penyimpanan file yang dimaksud. Untuk aplikasi *tachometer* dengan durasi waktu yang lama seperti pada sistem *monitoring*, maka penyimpanan file akan sangat membantu *user* tanpa harus mengamati setiap waktu dan setiap perubahan yang terjadi. Jika *user* membutuhkan grafik untuk analisis data, maka dapat dilakukan dengan merekonstruksi file TXT menjadi grafik.



Gambar 9. Grafik respon pengukuran terhadap kecepatan variabel



Gambar 10. Fitur perekaman data dan penyimpanan dalam bentuk file

TABEL I. PERBANDINGAN INSTRUMEN *TACHOMETER* DARI BEBERAPA PENELITIAN

Researchers	***Controller***	***Sensor***	***Error***	***Reference Tachometer***	***Data Samples***	***Communication Data***
David Tisaj [1]	Arduino MEGA	Hall Effect	N/A	N/A	N/A	No
Md. Masud Rana, *et al* [2]	Arduino UNO	IR-LED	1,07%	Analogue	4	No
M. Ehikhamenie, *et al* [3]	AT89C52	IR-LED	1,02%	Analogue	4	No
F. Barrero, *et al* [4]	FPGA	N/A	N/A	N/A	N/A	No
Prateek Mishra, *et al* [5]	Arduino UNO	IR-LED	N/A	N/A	N/A	No
P. K. Cariappa, *et al* [6]	Arduino UNO	IR-LED	N/A	Analogue	3	No
Salice Peter, *et al* [7]	ATMEGA16	IR-LED	N/A	N/A	N/A	No
Nitin Singh, *et al* [8]	AT89C2051	IR-LED	N/A	N/A	N/A	No
Penelitian ini	AT89S51	IR-LED	0,1 - 1,2%	Digital	27	Yes

D. Pembahasan

Tabel I menunjukkan capaian hasil yang telah dilaksanakan oleh beberapa penelitian. Sebagian diantaranya kurang dapat dijadikan sebagai pembanding terhadap penelitian ini dikarenakan minimnya informasi terkait dengan fitur dan spesifikasi teknis dari alat yang dibuat, sehingga tersisa tiga penelitian yang cukup layak untuk dijadikan pembanding yaitu Md. Masud Rana *et al* [2], M. Ehikhamenie, *et al* [3], dan P. K. Cariappa, *et al* [6].

Kesamaan dari tiga penelitian tersebut terletak pada terlalu sedikitnya sampel data yang diambil dan tidak adanya fitur komunikasi data. Sampel pengujian yang terlalu sedikit kurang dapat merepresentasikan secara total tentang karakter pengukuran *tachometer* dan tidak menjamin bahwa karakteristik akurasi dari *tachometer* akan bersifat linier. Akurasi dapat divalidasi jika *tachometer* digunakan untuk pengukuran pada rentang rpm rendah sampai tinggi dengan *interval* yang teratur. Pada penelitian ini *interval* pengukuran rpm dilakukan pada kisaran setiap selisih 100 rpm, sehingga jumlah sampel tercatat sebanyak 27 sampel data, dan batasan maksimum rpm yang diukur sama dengan tiga penelitian yang telah disebutkan sebelumnya.

Dengan kondisi tersebut di atas, *error* yang timbul dari penelitian ini tidak terpaut jauh dengan penelitian lainnya. Hal ini membuktikan bahwa reliabilitas dari prototipe yang dibangun pada penelitian ini cukup dapat diandalkan. Munculnya fenomena anomali pada saat *full-code program* menunjukkan bahwa jumlah *byte* pada *source code* perlu diperhatikan karena mempengaruhi akurasi kinerja.

Fitur komunikasi data yang dibangun pada penelitian ini juga menjadi pembeda dengan penelitian lainnya. Selain faktor akurasi, ada nilai tambah yang penting yang sangat bermanfaat pada ranah aplikasi, misalnya pada sistem lup tertutup. Dibandingkan dengan beberapa penelitian yang telah dilaksanakan oleh peneliti lain, maka hasil yang dicapai pada penelitian ini telah dapat dipertimbangkan sebagai satu langkah maju dalam bidang rancang bangun instrumen *tachometer*.

IV. SIMPULAN

Perancangan dan pembuatan *tachometer* berbasis *single chip controller* telah berhasil dikerjakan dengan hasil yang cukup baik. Akurasi *tachometer* prototipe yang dibangun pada penelitian ini masih dalam batas toleransi yang memadai untuk penggunaan skala laboratorium kampus. *Error* maksimum sebesar 1,2% dan adanya fitur komunikasi data menjadi poin positif untuk kelengkapan peralatan laboratorium yang memerlukan *tachometer* dengan fitur komunikasi data. Algoritma kalkulasi interupsi yang diujikan pada penelitian ini terbukti bekerja secara baik, dan diharapkan dapat diadopsi oleh peneliti lain untuk diimplementasikan pada *platform* mikrokontroler berbeda atau pengembangan sistem seperti misalnya fitur *wireless* atau aplikasi berbasis *gadget*.

REFERENSI

[1] D. Tisaj, “Design and Construction of a Tachometer”, Thesis, Murdoch University, 2014.

[2] M. M. Rana, M. Sahabuddin, dan S. Mondol, “Design and Implementation of a Digital Tachometer”, International Journal of Scientific Engineering and Technology”, Vol. 5, Issue 1, pp: 85-88, January 2016.

[3] M. Ehikhamenle dan B.O. Omijeh, “Design and Development of a Smart Digital Tachometer Using AT89C52 Microcontroller”, American Journal of Electrical and Electronic Engineering, Vol. 5, No. 1, pp. 1-9, 2017.

[4] F. Barrero, E. Galvan, J. V. Torrellas, dan L.G. Franquelo, ”Implementation of a tachometer by using two simple FPGAs”, www.researchgate.net/publication/275000668, July, 2000.

[5] P. Mishra, S. Pradhan, S. Sethiya, dan V. Chaudhary, “Contactless Tachometer With Auto Cut Off”, International Research Journal Of Engineering And Technology, Vol. 4, Issue : 04, April, 2017.

[6] P. K. Cariappa, A. Shweta, D. Pooja, B. T. Sudharani, dan M. N. Geetha, “Contactless Tachometer”, International Journal of Engineering Research and Technology, Vol. 6, Issue 13, 2018.

[7] S. Peter, N. M. Naveen, M. N. Nidheesh, S. A. James, dan S. Joseph, “Design of a Contactless Tachometer”, Internaational Journal of advanced Research in Electrical, Electronics and Instrumentation Engineering, Vol. 3, Issue 2, February 2014.

[8] N. Singh dan R. S. Toma, “Design of a Low-Cost Contact-Less Digital Tachometer with Added Wireless Feature”, International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075, Volume-3, Issue-7, December 2013.