PENINGKATAN KINERJA PERANGKAT ELEKTRONIK BERBASIS MIKROKONTROLER AVR 8 BIT DENGAN MENGGUNAKAN RTOS (REAL TIME OPERATING SYSTEM)

I Wayan Sutaya
Fakultas Teknik dan Kejuruan, Universitas Pendidikan Ganesha
email: sutaya.elkt@gmail.com

Abstrak

Tujuan dari penelitian ini adalah untuk mengimplementasikan RTOS (Real Time Operating System) pada perangkat elektronik berbasis mikrokontroler AVR 8 bit sehingga didapatkan peningkatan kinerja pada perangkat tersebut. RTOS yang digunakan adalah freeRTOS dimana RTOS ini mendukung mikrokontroler 8 bit, berukuran kecil, dan bersifat open source. Hasil penelitian ini berguna bagi praktisi-praktisi elektronika dalam hal menekan biaya produksi pembuatan perangkat elektronik berbasis mikrokontroler 8 bit karena resource yang digunakan bisa dikurangi. Metode yang digunakan adalah metode penelitian pengembangan (*Research and development*) dengan cara membuat studi kasus perangkat elektronik yang berbasis mikrokontroler. Perangkat elektronik yang dibuat adalah unit kendali elektronik pada sepeda motor. Selanjutnya mikrokontroler pada perangkat ini diprogram dengan menggunakan dua skenario. Skenario pertama adalah tanpa menggunakan RTOS dan skenario kedua dengan menggunakan RTOS. Dari dua skenario ini dilakukan pengujian dan analisis untuk mengetahui besar peningkatan kinerja yang didapat.

Kata-kata kunci: mikrokontroler, freeRTOS, kinerja

Abstract

The aim of this research is to implement RTOS (Real Time Operating System) on electronic stuffs based AVR 8 bit so that the performance can be increased. The RTOS choosed is freeRTOS in which this RTOS support for microcontroller 8 bit, small size of file, and surely open source. The result of this research is useful for engineers in electronics field or hobyes because reducing resource in building an electronic application. Method used in this research was the research and development method by way of developing a study case of electronic application based on microcontroller. The electronic application made was electronic control unit for motorcyle. Then the microcontroller on the application was programmed with two scenarios. The first was without RTOS and the second was using RTOS. From the scenarios, they were examined and analyzed to knowing the increasing of performance that was gotten.

Key words: microcontroller, freeRTOS, performance

PENDAHULUAN

Mikrokontroler adalah sebuah system on chip dimana sebuah mikroprosesor dan komponen-komponen pendukung lainnya seperti peripheral I/O, memori, ADC (Analog Digital Converter) tergabung dalam satu chip (Dhananjy, 2001). Fungsi dari mikrokontroler itu sendiri adalah sebagai unit sistem kendali dalam perangkat elektronik. Mikrokontroler ini beroperasi dengan cara menjalankan program yang disimpan di memori. Program ini adalah tugas-tugas/tasks yang mikrokontroler harus kerjakan. Mikrokontroler bekerja secara sekuensial dalam artian tugas yang satu diselesaikan kemudian dilanjutkan dengan tugas yang lainnya sehingga semakin banyak tugas yang harus dikerjakan maka semakin besar kemungkinan dari sebuah tugas mengalami keterlambatan untuk dilayani. Sebagai contoh kasus, sebuah mikrokontroler dijadikan sebagai unit kendali elektronik pada sepeda motor, dimana mikrokontroler ini harus mengukur kecepatan motor, mengukur volume bensin, mengukur suhu mesin, menyalakan lampu reting saat terjadi penekanan tombol reting, dan menampilkan informasi dilayar display sehingga total kerja yang harus dilakukan sebanyak 5 tugas. Setiap tugas ini harus diselesaikan dalam waktu tertentu misalkan meng-update layar display setiap 1 ms. Apabila sebuah chip mikrokontroler tidak bisa memenuhi waktu pengerjaan dari tugas-tugas yang diberikan, maka akan terjadi kegagalan pada sistem.

Pada umumnya apabila tugas-tugas/tasks yang harus dikerjakan oleh suatu perangkat elektronik jumlahnya banyak, solusi yang paling mudah dilakukan adalah dengan cara menambah jumlah chip mikrokontroler yang digunakan. Sehingga sebuah unit kendali elektronik pada sepeda motor bisa menggunakan lebih dari satu chip mikrokontroler. Tetapi menambah jumlah chip mikrokontroler berarti menaikkan biaya produksi dari perangkat elektronik yang akan dibuat dan konsumsi daya listrik yang diperlukan juga bertambah besar.

Salah satu cara untuk menangani hal ini adalah dengan meningkatkan kinerja dari sebuah chip mikrokontroler itu sendiri dengan menggunakan RTOS (Real Time Operating System). RTOS adalah sebuah sistem operasi yang diperuntukkan pada perangkat sistem embedded (bukan dekstop). Dengan RTOS sebuah mikrokontroler bisa dioperasikan secara multitasking yang berarti melakukan banyak pekerjaan dalam satu waktu secara bersamaan. Konsep multitasking ini akan mengurangi kondisi dimana suatu keadaan mikrokontroler tidak melakukan aktivitas karena sedang menunggu suatu event.

Penelitian yang dilakukan oleh B.C. Goradiya dkk (2011), "Embedded RTOS On ARM 7 Architecture", menyimpulkan penggunaan RTOS pada mikrokontroler LPC2138 (menggunakan mikroprosesor ARM) mempunyai beberapa keuntungan seperti konsumsi daya menjadi lebih rendah, biaya produksi yang rendah, lebih aman dan handal, dan lebih mudah dalam melakukan perbaikan sehingga sangat cocok sekali apabila diimplementasikan pada bidang otomotif.

Penelitian RTOS yang terkait dengan penanganan waktu kritis pada suatu event dilakukan oleh Ch. S. L. Prasanna dkk (2012), "Design of uC / Os II RTOS Based Scalable Cost Effective Monitoring System Using Arm Powered Microcontroller" menyimpulkan RTOS bisa digunakan untuk memastikan bahwa semua waktu kritis dari event-event bisa diproses. Hal ini sangat penting untuk menghindari terjadinya kegagalan dalam suatu sistem.

Penelitan yang membahas ketercocokan antara RTOS dengan mikrokontroler yang digunakan dibahas pada penelitian Jake Swart (2012) "Real Time Operating System Implemented In Hardware" menyimpulkan bahwa perpaduan RTOS dan mikrokontroler yang tepat akan menghidari terjadinya COTS(Commercial Off The Self).

Dari penelitian-penelitian yang telah dilakukan sebelumnya banyak pengimplementasian RTOS pada mikrokontroler 32 bit dan banyak RTOS yang beredar dipasaran diperuntukkan untuk mikrokontroler 32 bit karena mempunyai resource yang besar (Dolinay, 2011). Tetapi resource yang besar ini harus dibayar dengan biaya yang mahal juga. Tidak semua perangkat elektronik yang dibuat cocok dengan menggunakan mikrokontroler dengan resource besar karena terkendala dengan biaya produksi atas barang elektronik yang akan dihasilkan. Sehingga penggunaan jenis mikrokontroler tergantung dari pengaplikasiannya.

Penelitian yang terkait implementasi RTOS pada mikrokontroler 8 bit sangat penting dilakukan, karena melihat trend saat ini mikrokontroler 8 bit sudah banyak disertai ROM dengan kapasistas besar dan adanya RTOS yang mendukung mikrokontroler 8 bit. Selain itu mikrokontroler 8 bit ini juga mempunyai pangsa pasar tersediri yang tidak bisa tergantikan oleh mikrokontroler 32 bit.

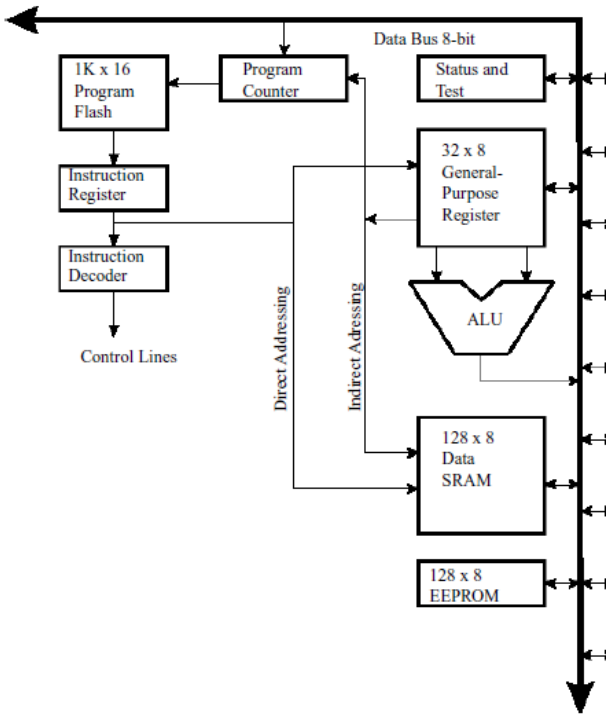
Mikrokontroler AVR

AVR adalah sebuah prosesor RISC dengan arsitektur Harvard. Arsitektur Harvard menggunakan sistem bahwa CPU mempunyai sebuah memori program dan sebuah memori data yang terpisah. Mikrokontroler AVR mempunyai fitur-fitur sebagai berikut:

1. Memori yang digunakan sebagai program memori adalah On-chip dan In System Programmable Flash.
2. Terdapat sebanyak 32 register 8 bit yang bekerja untuk tujuan umum.
3. Memori data EEPROM dan RAM berada dalam satu chip mikrokontroler.
4. Operasi kecepatan clock 0 sampai 10 MHz. Sebagai operasi bekerja dalam satu siklus clock.

5. Terdapat Power On RESET.
6. Terdapat Timer yang bisa diprogram.
7. Terdapat sumber interupt internal dan ekternal.
8. Terapat Timer watchdog yang bisa deprogram dengan osilator yang mandiri.
9. Mempunyai mode operasi SLEEP dan POWER DOWN .
10. Terdapat osilator clock RC.

AVR menggunakan arsitektur Harvard. Ini menekankan pemisahan bus data dan bus memori. Gambar 1 menunjukkan tampilan arsitektur AVR. Bus data adalah sebuah bus 8 bit yang menghubungkan komponen peripheral ke file register.



Gambar 1 Contoh chip mikrokontroler AVR

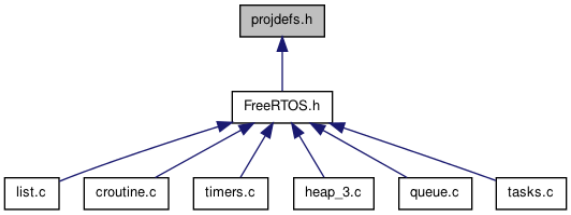
FreeRTOS

FreeRTOS adalah sistem operasi *real time* yang bersifat open source untuk sistem embedded. FreeRTOS mendukung banyak jenis arsitektur mikrokontroler dan kompiler. Seperti sistem operasi pada umumnya, tugas utama FreeRTOS adalah untuk menjalankan tugas-tugas/*tasks*.

Inti utama dari FreeRTOS hanya terdiri dari tiga file *source .c* dan satu file *header*, dengan total 9000 baris kode, termasuk komen dan baris yang kosong. Kode-kode FreeRTOS dipecah menjadi tiga area utama yaitu: tugas-tugas/*tasks*, komunikasi, antarmuka hardware (Richard, 2009).

- Tugas-tugas : Hampir setengah dari kode-kode freeRTOS adalah untuk proses tugas /tasks. Sebuah tugas adalah fungsi dalam bahasa C yang didefinisikan oleh user dan fungsi ini juga diberikan prioritas oleh user. Pada *tasks.c* dan *task.h* dilakukan proses pembuatan, penjadwalan, dan perbaikan *tasks*.
- Komunikasi : Mengkomunikasikan antar tugas/*task* adalah perkerjaan utama kedua dari freeRTOS. Sekitar 40% dari kode inti freeRTOS digunakan untuk proses komunikasi ini. File *queue.c* dan *queue.h* menangani komunikasi pada freeRTOS. *Tasks* dan *interrupt* menggunakan *queue* ini untuk proses mengiriman data dan pemberian sinyal pada saat terjadinya *resource* yang kritis.
- Antarmuka hardware : Kode-kode pada freeRTOS adalah hardware-independent. Kode-kode ini bisa jalan pada mikrokontroler 8051, ARM, dan sebagainya.

File-file kode freeRTOS bisa didownload secara gratis di alamat http://www.freertos.org/. Dimana susunan file utama freeRTOS ini seperti ditunjukkan pada Gambar 2 dibawah.



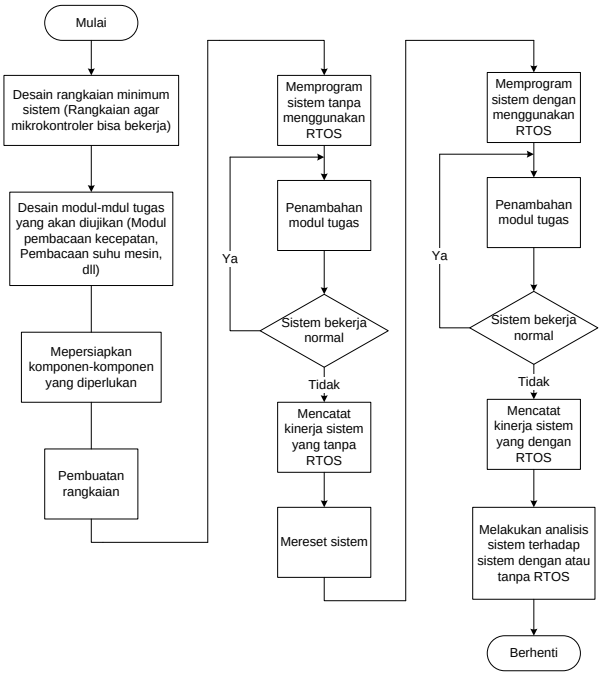
Gambar 2. Susunan file-file utama kode FreeRTOS (Dikutip dari: http://asf.atmel.com/docs/2.5.1/thirdparty/freertos/doxygen/avr32.services.os.freertos/html/a00028.html)

METODE

Pada penelitian ini metode yang digunakan adalah metode penelitian pengembangan (*Research and*

development) dengan cara membuat perangkat elektronik berbasis mikrokontroler sebagai studi kasus dimana perangkat elektronik yang dibuat adalah unit kendali elektronik pada sepeda motor.

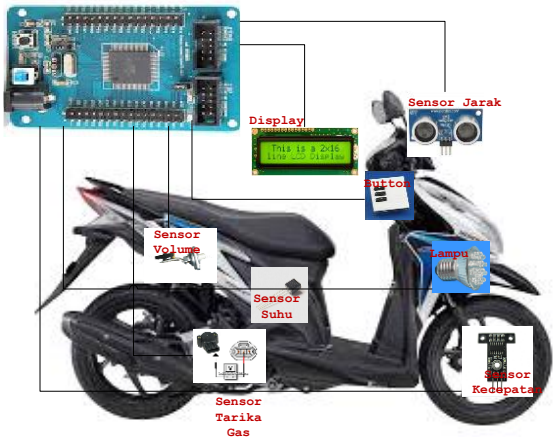
Kondisi pertama perangkat ini diprogram tanpa menggunakan RTOS dan kondisi kedua perangkat ini diprogram dengan menggunakan RTOS. Kemudian dilakukan pengujian kinerja pada masing-masing kondisi berupa penambahan tugas-tugas yang harus dikerjakan oleh perangkat ini sampai tidak bisa melayani semua tugas-tugas yang diberikan. Bagan alir dari proses penelitian ini ditunjukkan pada Gambar 3 dibawah.



Gambar 3. Bagan alir proses penelitian

Studi Kasus

Studi kasus yang dibuat pada penelitian ini adalah perangkat unit kendali elektronik pada sepeda motor. Perangkat unit kontrol elektronik ini bertugas untuk menampilkan informasi kecepatan sepeda motor, informasi volume bensin, tarikan gas, suhu mesin, informasi jarak, dan smart reting.



Gambar 4. Studi kasus perangkat unit control elektronik pada sepeda motor

Desain Task

Berdasarkan fungsi dari unit kendali elektronik pada sepeda motor yang dibuat maka jumlah task yang diperlukan adalah sebagai berikut:

1. *TaskDisplay*
   Task ini berfungsi untuk menampilkan data-data yang dibaca dari sensor-sensor. Informasi yang ditampilkan yaitu: jarak sepeda motor dengan benda yang di depannya, informasi temperature mesin, informasi volume minyak, dan informasi tarikan gas. Data yang ditampilkan diupdate setiap 2 ms. Sehingga task ini akan dieksekusi atau dikerjakan kembali setiap 2 ms.
2. *TaskSensorSpeed*
   Task ini berfungsi untuk memproses data yang didapat dari sensor kecepatan. Tugas dari task ini adalah menkonversikan tegangan rendah yang didapat dari sensor kecepatan menjadi satuan kecepatan Km/jam. Task ini dikerjakan secara periodik kembali setiap 1 ms.
3. *TaskSensorTempMachine*
   Task berfungsi untuk mengkonversikan tegangan yang didapat dari sensor suhu LM35 kedalam satuan suhu ^{0}C. Task ini dikerjakan secara periodik setiap 1 ms.
4. *TaskSensorPullGas*

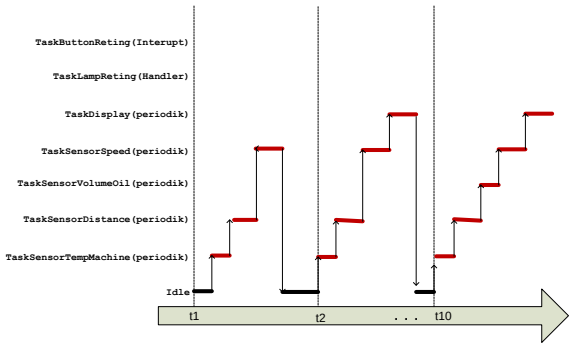
Task ini berfungsi untuk mengkonversikan tegangan dari potensiometer menjadi satuan tarikan gas. Task ini dikerjakan secara periodik setiap 1 ms.

5. *TaskButtonReting*
   Task ini berfungsi untuk memproses inputan yang didapat dari tombol reting. Task ini akan dikerjakan secara interupt saat terjadi penekanan tombol. Task ini akan menyela atau menghentikan task-task lainnya yang sedang melakukan proses.
6. *TaskLampReting*
   Task ini berfungsi sebagai handler saat *TaskButtonReting* memberikan isyarat bahwa lampu reting harus dihidupkan. Task ini dikerjakan berdasarkan even yang didapat dari *TaskButtonReting*.
7. *TaskSensorDistance*
   Modul ini berfungsi untuk memproses data yang didapat dari sensor jarak. Pemrosesan data dilakukan setiap 1 ms.
8. *TaskSensorVolumeOil*
   Task ini berfungsi untuk memproses data yang didapat dari sensor volume. Pemrosesan data dilakukan setiap 10 ms.

Proses penjadwalan

Proses penjadwalan dari task-task yang telah dibuat untuk perangkat unit control elektronik pada sepeda motor dapat digambarkan sebagai berikut:

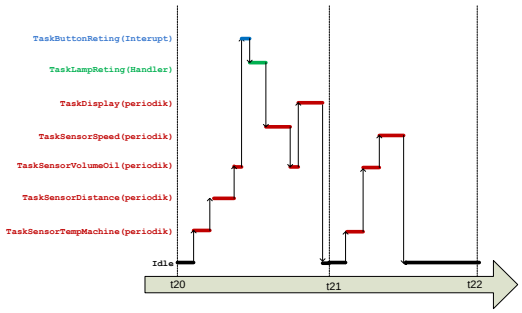
1. Proses tidak terjadi penekanan tombol
   Pada saat tidak ada penekanan tombol reting maka *TaskButtonReting* dan *TaskLampuReting* tidak dieksekusi. Rentang waktu antara t_1 dengan t_2 sebesar 1 ms.



Gambar 5. Contoh proses penjadwalan dengan tidak menekan tombol

Saat pertama kali program dijalankan maka yang dieksekusi adalah task idle, task idle adalah task yang tidak melakukan proses apapun. Selanjutnya *TaskSensorTempMachine*, *TaskSensorDistance*, *TaskSensorSpeed*, dan *TaskPullGas* akan diproses. Apabila semua *task-task* ini sudah selesai dikerjakan dalam waktu 1 ms maka task *Idle* akan dieksekusi sampai t_2. Selanjutnya saat memasuki t2 maka *TaskSensorTempMachine*, *TaskSensorDistance*, *TaskSensorSpeed*, dan *TaskDisplay* akan dieksekusi. Jadi *TaskDisplay* ini dieksekusi setiap 2 ms. Proses ini terus berulang sampai pada t_{10} dimana *TaskSensorTempMachine*, *TaskSensorDistance*, *TaskSensorSpeed*, *TaskSensorPullGas*, *TaskSensorVolumeOil*, dan *TaskDisplay* akan dieksekusi. Jadi *TaskSensorVolumeOil* dieksekusi setiap 10 ms.

2. Proses penekanan tombol
   Pada saat terjadi penekanan tombol reting maka *TaskTombolReting* akan langsung dieksekusi. Apabila ada *task* yang sedang diproses maka task tersebut akan dihentikan. Seperti yang ditunjukkan pada Gambar 6 dimana proses penekanan tombol reting terjadi pada waktu 20 ms.

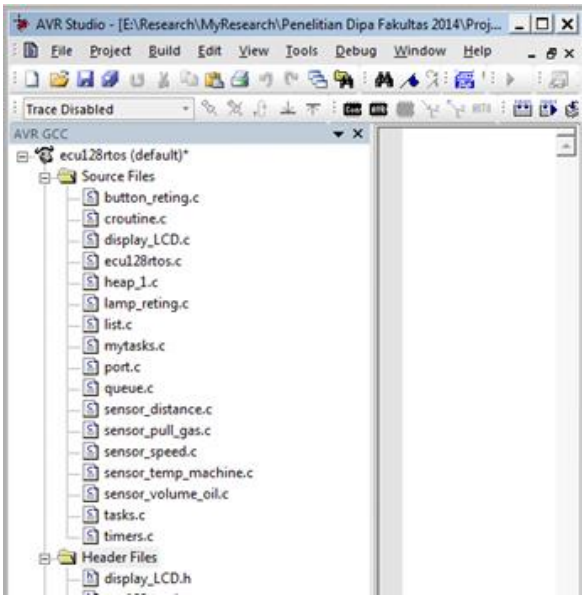


Gambar 6. Contoh proses penjadwalan dengan penekanan tombol

Tombol reting ditekan saat *TaskSensorVolumeOil* sedang ditekan sehingga proses yang terjadi pada task ini dihentikan sementara yang selanjutnya *TaskButtonReting* akan dieksekusi. *TaskButtonReting* akan memberikan event ke *TaskLampReting* sehingga memaksa proses menuju ke *TaskLampReting*. Apabila proses pada *TaskLampReting* sudah selesai maka proses akan dilanjutkan kembali ke *TaskSensorVolumeOil* yang sebelumnya tertunda. Proses yang terjadi pada *TaskSensorVolumeOil* akan dilanjutkan bukan mengulang dari awal. Misalkan apabila proses yang terjadi sudah 40% maka akan dilanjutkan untuk memproses 60% yang tersisa. Selanjutnya pada t_{21} task-task yang akan dikerjakan yaitu: *TaskSensorTempMachine*, *TaskSensorDistance*, *TaskSensorSpeed*, dan *TaskSensorPullGas*.

Program

Untuk memudah dalam memanajemen proses pada setiap task maka setiap task dibuatkan file tersendiri. Sebagai contoh modul-modul program yang diproses pada *TaskSensorDistance* disimpan pada file *sensor_distance.c* sedangkan variabel atau konstanta yang terkait disimpan pad *sensor_distance.h.* Karena terdapat 8 task sehingga diperlukan file-file untuk program dari proses sebanyak 8 file .c dan 8 file .h. File-file yang telah dibuat ditunjukkan pada Gambar 7.

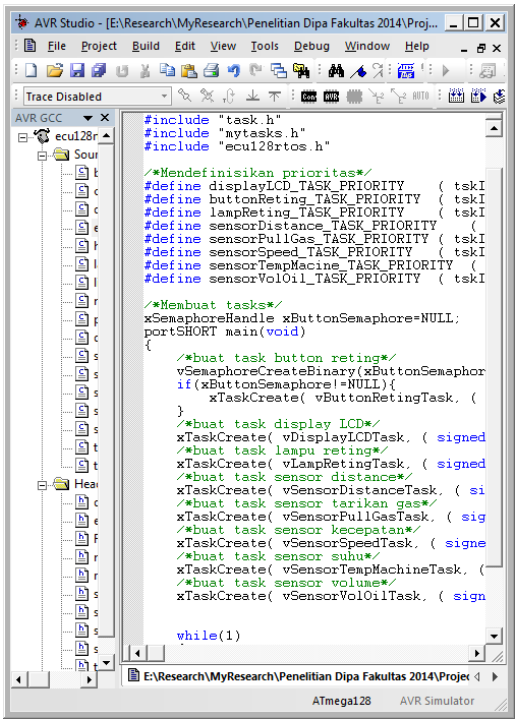


Gambar 7. File-file untuk membuat program untuk masing-masing task.

Sebagai contoh program yang dibuat dalam *sensor_temp_mesin.c* adalah sebagai berikut:

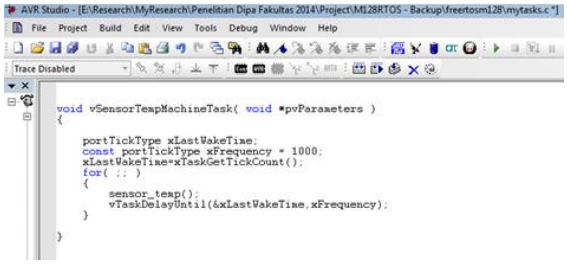
```
void sensor_temp(){
  DDRD = 0xFF;
  DDRA = 0;
  ADCSRA = 0x87;
  ADMUX = 0xE0;
  char tempC, tempF, display;
  float tempff;
  ADCSRA |= (1<<ADSC);
  while((ADCSRA & (1<<ADIF)) ==
0);
  tempC = ADCH;

  tempff = (float)tempC;
  tempff = (tempff*9)/5 + 32;
  tempF = tempff;
 PORTD = tempF;
}
```

Program ini akan dipanggil jika *TaskSensorTempMesin* dieksekusi. Langkah selanjutnya adalah membuat membuat task itu sendiri. Semua task dibuat di file *mytasks.c*. Proses pembuatan task ditunjukkan pada Gambar 8.



Gambar 8. Membuat task

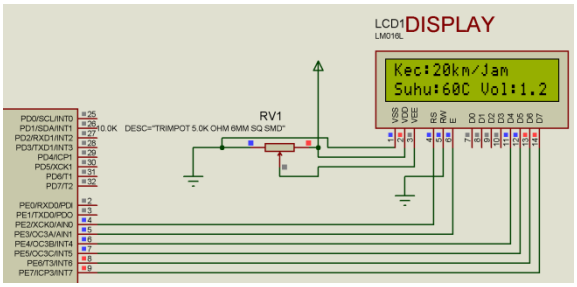
Selanjutnya Mendefinisikan properti masing-masing task. Pendefinisian property untuk masing-masing task dibuat pada file mytasks.c. Salah satu contoh pendefinisian task yaitu *SensorTempMachineTask* ditunjukkan pada Gambar 9. Task ini akan dieksekusi secara periodic setiap 1 ms dengan perintah *cost portTickType xFrequency = 1000*. Task ini akan memanggil fungsi sensor_temp() yang berfungsi membaca sensor suhu LM35.



Gambar 9. Contoh *TaskSensorTempMachine*

Simulasi

Implemetasi dalam bentuk bentuk hardware dari perangkat unit kontrol elektronik pada sepeda motor ini bisa disimulasikan dengan menggunakan software *proteus*. Software ini bisa digunakan untuk mensimulasikan perangkat elektronik sebelum dibuat prototipe atau diproduksi secara masal.



Gambar 10. Simulasi display

HASIL DAN PEMBAHASAN

Dari projek penelitian yang telah dibuat maka kinerja sistem antara yang tidak menggunakan RTOS dibandingkan dengan menggunakan RTOS bisa ditabelkan seperti yang ditunjukkan pada Tabel 1.

Tabel 1 Perbandingan kinerja sistem dengan RTOS dan tanpa RTOS

No	Kinerja Sistem	Tanpa RTOS	Dengan RTOS
1.	Waktu eksekusi sebuah proses bisa ditentukan secara mandiri	Tidak	Ya
2.	Antar proses bisa bekerja secara independent	Tidak	Ya
3.	Proses tidak dilakukan secara kontinu	Tidak	Ya
4.	Mampu memberikan prioritas pada proses yang kritikal	Tidak	Ya
5.	Kebutuhan ROM untuk menyimpan program meningkat	Tidak	Ya

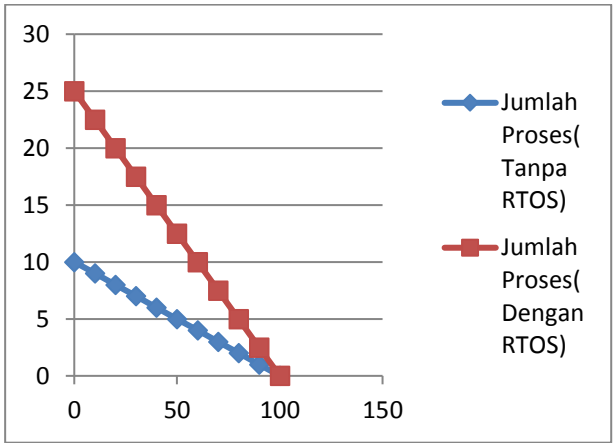
Dengan menggunakan RTOS maka waktu eksekusi suatu proses bisa ditentukan secara mandiri sedangkan tanpa RTOS hal tersebut tidak bisa dilakukan. Keuntungan yang didapat dari ini adalah sebuah proses

tidak harus dialokasikan waktu yang sama. Sebagai contoh pada projek yang dibuat, *TaskSensorDistance* dialokasikan waktu secara periodik setiap 1 ms sedangkan *TaskSensorVolumeOil* dialokasikan waktu secara periodik setiap 10 ms. Informasi terhadap sensor jarak perlu diupdate terus menerus karena pengendara sepeda motor memerlukan informasi tersebut setiap saat. Sedangkan informasi tentang volume bensin tidak memerlukan update dalam periode yang singkat. Hal ini meningkatkan kinerja sistem karena proses yang memerlukan urgenitas yang rendah bisa diberikan alokasi waktu yang lebih sedikit. Sistem yang tidak menggunakan RTOS tidak bisa melakukan hal ini karena eksekusi bersifat sekuensial sehingga tidak bisa melakukan alokasi waktu untuk setiap proses.

Setiap proses pada sistem yang menggunakan RTOS bisa bekerja secara independen. Hal ini ditunjukkan setiap *task* pada sistem mempunyai waktu eksekusi secara periodik masing-masing. Proses dilakukan secara periodik bukan kontinyu seperti pada sistem yang tidak menggunakan RTOS. Proses yang kontinyu berarti sistem tidak pernah berhenti untuk mengeksekusi proses-proses yang ada melainkan melakukan secara berulang terus menerus sehingga akan mengkonsumsi daya listrik yang lebih besar. Sedangkan proses pada sistem dengan RTOS dilakukan secara periodik, sehingga sistem akan sempat mengalami kondisi idle/tidak bekerja apabila belum ada proses/task mencapai waktu eksekusi. Hal ini memungkinkan terjadi penghematan daya listrik. Pada sistem yang menggunakan RTOS mampu memberikan prioritas pada proses-proses yang bersifat kritis. Pada projek penelitian yang dibuat proses yang kritikal adalah tombol reting. Jadi apabila tombol reting ditekan maka sistem akan menghentikan proses-proses lainnya untuk melakukan proses tombol reting. Sehingga *TaskTombolReting* tidak bersifat periodik tetapi bersifat interrupt. Salah satu kelemahan dari penggunaan RTOS adalah membutuhkan ROM penyimpanan program yang lebih besar. Hal ini disebabkan terdapat tambahan program yaitu freeRTOS yang dimasukkan ke dalam projek.

Analisa Peningkatan Kinerja

Dengan menggunakan RTOS maka jumlah pekerjaan/proses yang bisa dikerjakan semakin meningkat. Sebagai perbandingan jumlah proses terhadap kinerja sistem dari sistem yang menggunakan RTOS dan tanpa RTOS bisa ditunjukkan pada Gambar 11.



Gambar 11. Perbandingan kinerja sistem dengan dan tanpa RTOS

Dari grafik bisa dijelaskan bahwa pada mikrokontroler 8 bit penggunaan RTOS bisa meningkatkan kinerja sistem lebih dari dua kali lipat. Hal ini ditunjukkan dengan meningkatnya jumlah proses yang bisa dikerjakan dengan menggunakan RTOS dibandingkan dengan tanpa RTOS.

KESIMPULAN DAN SARAN

Dari penelitian yang dilakukan pada peningkatan kinerja perangkat elektronik berbasis mikrokontroler AVR 8 bit dengan menggunakan RTOS maka dapat ditarik kesimpulan bahwa selain bisa meningkatkan kinerja penggunaan RTOS juga memungkinkan untuk mengetahui jumlah maksimun proses atau *job* yang bisa dikerjakan oleh sebuah mikrokontroler.

DAFTAR PUSTAKA

B.C.Goradiya, Dr. H.N.Pandya, V.V. Kamdar. 2011. *Embedding RTOS On ARM 7 Architecture.* National Conference on Recent Trends in Engineering & Technology, 13-14 May 2011.

Ch. S. L. Prasanna, M. Venkateswara Rao. 2011. *Design of uC/Os II RTOS Based Scalable Cost Effective Monitoring System Arm Powered Microcontroller*. International Journal of Scientific and Research Publications, Volume 2, Issue 4, April 2012.

Jake Swart. 2012. *Real Time Operating Systems Implemented in Hardware*. School of Information Techology Univeristy of Ottawa.

Dhananjy V. Gadre. 2001. *Programming And Customizing The AVR Microcontroller*. McGraw-Hill, New York.

Richard Barry. 2009. *Using the FreeRTOS Real Time Kernel*.

DOLINAY J., VAŠEK V., DOSTÁLEK P.. 2011. *Utilization of Simple Real-time Operating system on 8-bit microcontroller. International Journal of Mathematical Models And Method In Applied Sciences*, Issue 4, Volume 5, 2011.