IJEIS, Vol.2, No.2, October 2012, pp. 175~186
ISSN: 2088-3714

Implementasi DuinOS pada Purwarupa Sistem Penyortiran Barang Berbasis Arduino Uno

Ferry Agusta Putra*[1], Andi Dharmawan[2], Triyogatama Wahyu Widodo[3]
[1]Program Studi Elektronika dan Instrumentasi, JIKE, FMIPA, UGM, Yogyakarta
[2,3]Jurusan Ilmu Komputer dan Elektronika, FMIPA, UGM, Yogyakarta
e-mail: [*1]**ferryagusta@gmail.com**, [2]dharmawan.andi@gmail.com, [3]yogatama@ugm.ac.id

Abstrak

Telah dibuat sebuah purwarupa sistem penyortiran barang berdasarkan ketinggian menggunakan RTOS. RTOS sendiri merupakan hasil pengembangan pada bidang IT yang kemudian bisa diadaptasikan untuk bidang otomatisasi, salah satunya digunakan untuk merancang sistem kontrol secara real time dan multitasking.

Sistem ini menggunakan devolepment board Arduino Uno sebagai pusat pengendali. Sedangkan RTOS yang digunakan adalah DuinOS. Untuk mendeteksi ketinggian benda digunakan sensor SRF06 sedangkan untuk mengetahui keadaan suhu mesin konveyor digunakan sensor LM35. Antarmuka pada Personal Computer menggunakan Borland Delphi 7. Pada antar muka ini untuk mengendalikan motor dan pengaturan setpoint batas ketinggian benda yang akan disortir kemudian data disimpan pada sebuah file.

Sistem yang telah dibuat mampu melakukan proses otomasi pendeteksian ketinggian barang dan dilanjutkan dengan proses penyortiran barang. Model barang terdapat 5 buah dengan variasi ketinggian yang berbeda-beda mulai dari 3 cm hingga 7 cm dengan panjang 5 cm dan lebar 3 cm. Antarmuka yang telah dibuat mampu melakukan pengaturan dan pemantauan sistem. Memori yang digunakan sistem sebesar 11.332 byte pada memori flash dan 1680 byte digunakan pada memori RAM. Tick time default sebesar 1 milidetik telah digunakan pada sistem dan sistem sudah bisa bekerja dengan baik.

Kata Kunci*— DuinOS, Arduino UNO, RTOS, task, multitasking.*

Abstract

Have made a prototype system based on the height of sorting items using RTOS. RTOS itself is the result of the development in the field of IT which can then be adapted to the field of automation, one of which is used to design the control system in real time and multitasking.

The system uses an Arduino Uno board devolepment as the central controller. While the RTOS used is DuinOS. To detect the height of the goods, is used SRF06 sensor. while to see how the engine temperature conveyors, is used sensor LM35. Interface on Personal Computer using Borland Delphi 7. At this interface is able to control the motor and height limit setpoint adjustment items to be sorted then the data is stored in a file.

The system has been made capable of performing detection process automation the height of goods and continued with the process of sorting goods. Models goods are 5 pieces with a variety of different heights ranging from 3 cm to 7 cm by 5 cm long and 3 cm wide. The interface has been made capable of setting and monitoring the system. The memory used by the system to 11 332 bytes and 1680 bytes of flash memory used in RAM memory. The default tick time of 1 ms was is used in the system and the system is able to work properly.Keywords: DuinOS, Arduino UNO, RTOS, task, multitasking.

Keywords*— DuinOS, Arduino UNO, RTOS, task, multitasking.*

Received October 5th,2012; Revised October 22th, 2012; Accepted May 2nd, 2013

1. PENDAHULUAN

Dunia industri merupakan salah satu aspek penting dan strategis dalam menopang perekonomian suatu negara. Negara maju selalu identik dengan dunia *manufacturing* yang maju pula. Oleh karena itu diperlukan suatu teknologi yang handal agar bisa memaksimalkan baik kuantitas maupun kualitas hasil produksi.

Namun tidak semua proses produksi dalam dunia industri berjalan lancar. Sering ditemukan kesalahan-kesalahan produksi yang disebabkan oleh *human error*. Selain itu pada proses produksi konvensional hasil produksi dari segi kuantitas dan kualistas masih rendah. Dan tidak ada nya data base membuat *monitoring* produksi tidak berjalan dengan baik.

Dari beberapa permasalahan tersebut diperlukan suatu sistem yang dapat menanggulangi permasalahan tersebut. Sehingga banyak industri yang berlomba-lomba menghasilkan proses produksi yang efektif serta efisien. *Automation* atau otomasi sistem kontrol kemudian menjadi jawaban untuk menghasilkan tujuan tersebut.

Untuk mempermudah perancangan sistem kontrol tersebut maka pada proses perancangan ikut diimplementasikan *real time operating systems* (RTOS). RTOS sendiri merupakan hasil pengembangan pada bidang IT yang kemudian bisa diadaptasikan untuk bidang otomatisasi, Salah satunya digunakan untuk merancang sistem kontrol secara *real time* dan multitasking. Diharapkan dengan penggunaan RTOS ini respon sistem mampu berjalan lebih baik karena sistem produksi dalam dunia industri *modern* membutuhkan keakuratan waktu yang tinggi.

Didasarkan atas uraian-uraian teknologi tersebut, maka diangkatlah tema tentang pengimplementasian RTOS pada purwarupa sistem penyortiran barang dengan mikrokontroler AT mega 328P yang terdapat pada *board* Arduino Uno sebagai pusat pengendali.

2. METODE PENELITIAN

2.1 Tinjauan Pustaka

Penerapan *Real Time Operating System* pada purwarupa sistem penyortiran barang ini diharapkan dapat memberikan gambaran tentang kelebihan penerapan RTOS dibandingkan dengan motode sekeuensial. Sedangkan pada *plant* yang dikontrol diharapkan juga akan memberikan gambaran tentang teknik otomasi yang sering diterapkan dalam dunia industri guna tercapainya produktvitas dan evektivitas kerja.

Berbagai penelitian mengenai otomatisasi kontrol dengan pada konveyor telah banyak dilakukan, Syamsyuhadi [1], telah membuat konveyor berbasis PLC Omron CPM2A yang digunakan untuk sistem pemilahan barang. Sistem konveyor tersebut mampu melakukan proses pemilahan dengan parameter ketinggian barang. Dalam perancangan sistem tersebut sensor yang digunakan menghitung ketinggian menggunakan dioda laser merah dan LDR.

Selain itu penelitian pada otomasi kontrol juga tidak hanya dilakukan menggunakan PLC saja. Ahmad [2], melakukan penelitian otomasi berbasis mikrokontroler dengan *plant* proses otomatisasi pengisi gula pada kantong plastik. Pada penelitian ini konveyor menjadi media transportasi kantong plastik yang nantinya diisi dengan gula. Pada sistem ini selain menggunakan Arduino Mega 1280. Ahmad juga menggunakan sensor Load Cell, relay, motor DC, valve, dan *limit switch* sebagai komponen pendukung sistem.

Sedangkan Wahyudi [3], pernah melakukan penelitian tentang penerapan *Real Time Operating System* pada sistem pengatur suhu ruangan otomatis. Pada penelitian ini purwarupa yang dibuat adalah dua ruangan yang terdapat sensor suhu LM35 dan sensor cahaya LDR. Sedangkan untuk aktuator terdapat kipas dan lampu led *super bright* pada masing-masing ruangan. Pada penelitian ini bisa dilihat pembagian tugas *(task)* pada sistem dan kita bisa leluasa untuk mengaturnya.

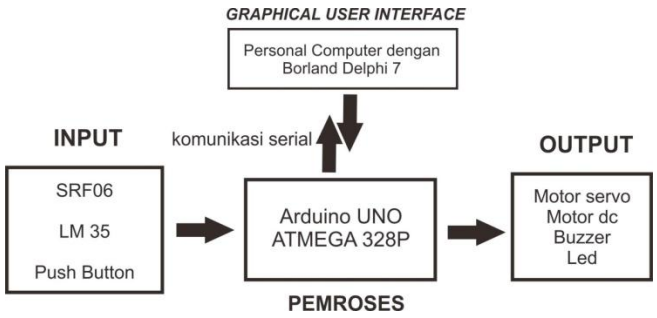
Pada penelitian ini penulis akan membuat purwarupa sistem otomasi dengan mengimplementasikan RTOS pada unit pemroses berbasis *baord* Arduino Uno dengan

mikrokontroler AT Mega 328 dan sebuah *plant* purwarupa konveyor yang berjalan untuk melakukan peyortiran barang berdasarkan parameter ketinggian. Diharapkan pada hasil akhir penelitian ini bisa dianalisis tentang penggunaan RTOS pada sistem otomasi itu sendiri guna mendapatkan kinerja yang optimal beserta teknologi *plant* yang ada pada sistem tersebut.

2.2 Analisis dan Perancangan Sistem

Perancangan *hardware* terdiri dari perancangan mekanik miniatur konveyor dan rangkaian elektronik sedanganken pada perancangan *software* dibahas mengenai perancangan program RTOS yang akan ditanamkan pada sistem pemroses dan perancangan antar muka sistem. Selain itu pada bab ini juga akan disinggung tentang metode pengujian yang akan diterapkan.

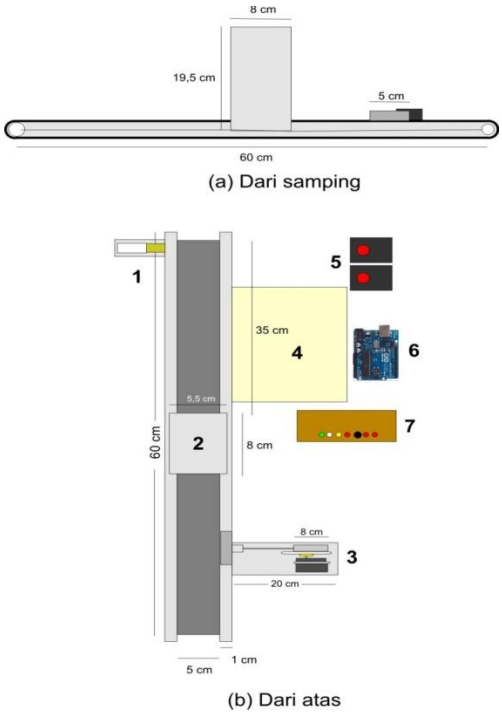
Pada perancangan program menggunakan RTOS oleh karena itu perancangan program tidak merupakan program sekuensial melainkan program akan dibagi ke beberapa *task* yang seolah-olah berjalan bersamaan. Gambar 1 merupakan diagram blok dari sistem secara keseluruhan.



Gambar 1 Diagram blok sistem secara keseluruhan

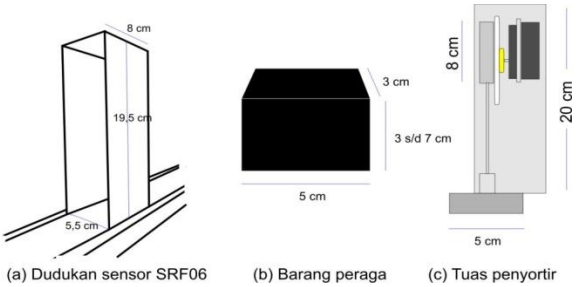
Arduino Uno yang tertanam ATmega 328 menjadi otak sistem berhubungan dengan GUI *(Graphical User Interface)* melalui komunikasi serial. GUI akan memantau pembacaan tinggi barang dan suhu mesin beserta grafiknya melalui PC serta pembacaan *counter* atau penghitung barang yang telah tersortir. Selain itu pada GUI juga bisa melakukan kontrol *on-off* motor dan pengaturan *setpoint* ketinggian minimal barang yang akan disortir. Arduino Uno mendapat masukan dari sensor yang terdiri dari sensor jarak ultrasonik SRF06, sensor suhu LM35, dan *push button*. Serta memiliki aktuator berupa motor servo sebagai tuas penyortir, motor dc sebagai mesin konveyor, *buzzer*, dan led sebagai indikator sistem.

Konveyor ini dirancang sebagai *plant* media penyortiran barang. Penyortiran barang dilakukan berdasarkan ketinggian barang dan konveyor ini akan digunakan sebagai media transportasi barang. Alat peraga barang berupa balok kotak dengan ketinggian yang berbeda-beda diletakkan di atas konveyor. Konveyor diletakkan pada sebuah plat aluminium yang dipasang motor servo sebagai tuas penyortir utama, sistem sensor yang meliputi sensor jarak menggunakan sensor ultrasonik SRF06, dan sensor suhu dengan LM35. Selain itu juga akan dipasang sistem *driver* motor, panel indikator beserta *buzzer* sebagai alarm, dan led sebagai indikator.



Gambar 2 Perancangan *hardware* konveyor

Nomor 1 adalah peletakan dari motor DC yang menjadi sumbu penggerak alas konveyor. Pada nomor 1 ini juga akan ditempatkan sensor LM35 sebagai sensor suhu. Peletakan sensor ultrasonik SRF06 seperti pada nomor 2 akan ditempatkan pada media gawang alumunium didesain sedemikian rupa sehingga sistem sensor tersebut mampu mendeteksi ketinggian barang yang melewati dibawahnya. Ketinggian barang akan didapat dengan perhitungan dari tinggi sensor dari alas dikurangi pembacaan jarak dari sensor jarak tersebut. Selanjutnya untuk perancangan tuas penyortir akan digunakan motor servo yang akan diletakkan seperti pada nomor 3. Nomor 4 adalah kotak yang berisi rangkaian-rangkaian elektronik pada nantinya. Perancangan mekanik tuas penyortir dibuat sedemikian rupa sehingga jika motor servo berputar akan membuat tuas pendorong maju kedepan sehingga mampu untuk melakukan proses penyortiran barang. Pada sistem ini juga terdapat pengontrol hidup-mati mesin konveyor menggunakan *pushbutton* seperti pada nomor 5. Sedangkan untuk perancangan papan indikator yang akan digunakan untuk memberikan tanda motor, task, dan alarm aktif atau tidak seperti pada nomor 7. Sedangkan untuk detil tentang komponen tambahan konveyor tersebut terdapat pada Gambar 3.

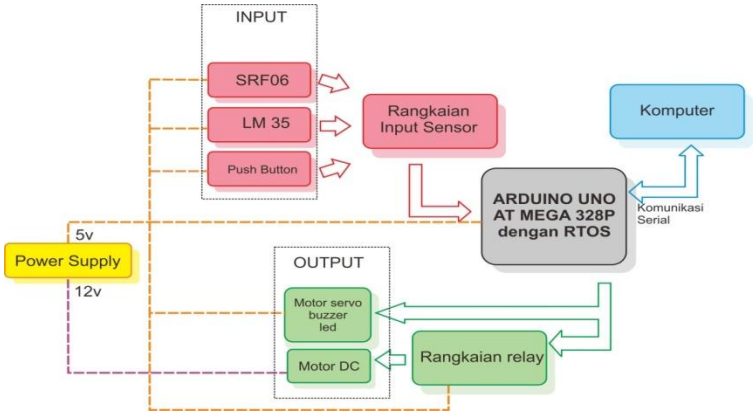


Gambar 3 Rancangan komponen pendukung konveyor

Gambar 3 tersebut merupakan komponen pendukung konveyor yang meliputi dudukan sensor SRF06 pada gambar (a), dudukan tersebut mempunyai dimensi 5,5 cm x 8 cm x 19,5 cm. Gambar (b) adalah gambar perancangan barang pada purwarupa ini dengan dimensi 5 cm x 3

cm x 3 s/d 7 cm. Sedangkan gambar (c) adalah perancangan tuas penyortir menggunakan motor servo.

Pada rancangan *hardware* selanjutnya adalah sistem rangkaian elektronik. Rancangan rangkaian elektronik pada sistem ini ada 2, yakni: rangkaian sensor dan rangkaian pengendali motor. Rangkaian sensor merupakan rangkaian input sensor sebelum diteruskan pada Arduino sedangkan rangkaian pengendali *(driver)* motor merupakan rangkaian relay yang mendapat masukan dari output Arduino. Sistem *hardware* purwarupa ini memiliki blok diagram yang ditunjukkan pada Gambar 4.



Gambar 4 Diagram Blok Sistem *Hardware*

Ada tiga task yang akan dirancang oleh mikrokontroler pada sistem ini. Task pertama yakni taskPING, yang mempunyai tugas utama kontrol penyortiran. *Device* yang bekerja adalah sensor jarak ultrasonik SRF06 dan tuas penyortir motor servo. Bila sensor mendeteksi barang dengan ketinggian lebih besar atau sama dengan dari *setpoint* maka barang tersebut akan disortir oleh tuas pendorong dan disaat bersamaan melakukan *counting* atau perhitungan jumlah barang yang tersortir.

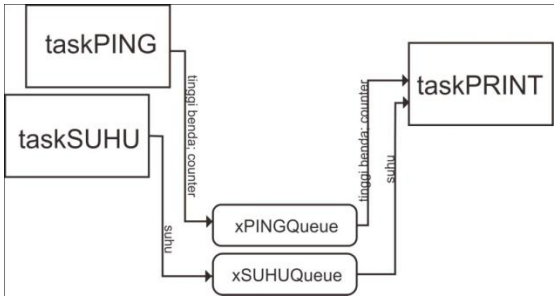
Tabel 1 Pembagian task dan prioritasnya

Prioritas	**Tugas (task)**
HIGH	taskPING
NORMAL	taskSUHU
LOW	taskPRINT

Task PING diberi prioritas tertinggi *(high_priority)* karena task ini menjadi fungsi utama yang berfungsi melakukan penyortiran barang. Task ini tidak boleh tertunda oleh task lain karena bila tertunda akan menganggu proses penyortiran yang menjadi inti dari sistem ini.

Kemudian taskSUHU yang berfungsi sebagai *monitoring* suhu mesin. Selain itu task ini juga bertugas mematikan mesin dan menyalakan alarm (led dan buzzer) jika terdeteksi suhu *overheat*. Karena task ini cukup penting tetapi tidak sefital taskPING maka diberikan prioritas nomer dua atau *normal_priority,* sehingga jika terjadi keterlambatan sepersekian detik masih bisa diberikan toleransi.

Selanjutnya taskPRINT bertugas menerima data dari taskPING dan taskSUHU dan selanjutnya mengirimkan data tersebut ke port serial. Karena kedua task sebelumnya bisa dikatakan lebih penting dan vital dari task ini maka pada taskPRINT ini diberikan prioritas ke-tiga atau *low_priority*.



Gambar 5 Blok diagram keterkaitan antar task

Diagram blok pada Gambar 5 menjelaskan keterkaitan antar task dalam sistem ini. Semua task tersebut diatur sedemikian rupa oleh DuinOS. Sebelumnya pada program awal dilakukan pembuatan tiga buah task dan 2 buah *queue*. Selanjutnya dilakukan *setup* dan pengaturan *setpoint*, setelah itu jika proses *setup* sudah selesai maka untuk pemicu atau *trigger* agar mulai masuk ke task loop ditulislah huruf 'x' ke serial. Suatu task sudah masuk dalam keadaan ready maka task tersebut akan segera dieksekusi. Disini *queue* berfungsi sebagai jembatan antar task. TaskPING dan taskSUHU akan memberikan informasi yang akan diberikan kepada taskPRINT melalui *queue* ini. Untuk tugas awal sebelum masuk ke task loop hanya dijalankan 1 kali saja, sedangkan setelah masuk ke dalam task loop instruksi akan terus berputar (*looping*). Untuk lebih jelasnya tentang perancangan tiap-tiap task akan dipaparkan sebagai berikut.

TaskPING menjadi task utama dan mempunyai peran penting pada sistem ini bertugas untuk menyortir barang yang melewati konveyor. Task ini mendapat masukan dari sensor ultrasonik SRF06, masukan dari sensor SRF06 tersebut kemudian diolah menggunakan *library* NewPing.h dan dihasilkan data jarak dalam satuan cm. Data keluaran tersebut diolah lagi agar bisa mengetahui tinggi barang yang melewatinya dengan perhitungan jarak sensor dengan alas (18 cm) dikurangi pembacaan data jarak sensor SRF06. Selanjutnya untuk penyortiran dibandingkan apakah pembacaan data sensor jarak ≤ 18 – *setpoint*, Jika demikian maka barang akan disortir oleh tuas pendorong, sebaliknya jika tidak barang akan diteruskan oleh konveyor. Waktu antara pendeteksi barang dan penggerak tuas dengan waktu tunda 2,7 detik. 1,9 detik tersebut merupakan waktu yang dibutuhkan untuk perjalanan barang dari daerah sensor menuju daerah pendorong dan 800 milidetik untuk menjalankan servo. Selain itu pada task ini juga terdapat tugas penghitung atau *counter* barang yang telah tersortir. Data tinggi barang tadi dan penghitung kemudian dikirimkan ke taskPRINT melalui xPINGqueue.

Task kedua adalah taskSUHU mempunyai tugas utama *monitoring* kondisi suhu motor konveyor. Task ini mempunyai waktu tunda 500 milidetik. Masukan suhu didapat dari sensor suhu LM35 selanjutnya data dioleh melalui perhitungan (aref_voltage * analogRead(LM1) * 100.0) / 1024.0) dengan nilai aref_voltage adalah 5.0 sehingga didapat data suhu dengan satuan celcius. Tugas kedua dari task ini adalah sebagai pengaman sistem jika terjadi *overheat*. Batas atas suhu normal pada sistem ini ditentukan pada suhu 50 ^{0}C. Kemudian jika suhu terdeteksi dalam keadaan *overheat* maka secara otomastis sistem akan mematikan mesin dan menyalakan alarm pada waktu yang bersamaan. TaskSUHU ini juga akan mengirimkan data suhu ke taskPRINT melalui xSUHUqueue. Selain kedua tugas tersebut task ini juga bertugas sebagai kontrol *on-off* mesin konveyor.

TaskPRINT bertugas untuk mengirimkan data tinggi barang, counter, dan suhu. Kedua data tersebut diambil dari xPINGqueue dan xSUHUqueue dan selanjutnya mengirimkan data tersebut ke serial. Selain itu pada task ini juga bertugas untuk mengirimkan variabel 'setjarak' untuk pengaturan *setpoint* ketinggian barang. Kesemuanya nantinya untuk digunakan oleh GUI pada *personal computer*. Task ini mempunyai waktu tunda 1 detik.

Perangkat lunak antarmuka atau GUI dibuat dengan menggunakan Borland Delphi 7. Fungsi utama antarmuka ini adalah untuk menampilkan data dari tinggi barang dan suhu yang

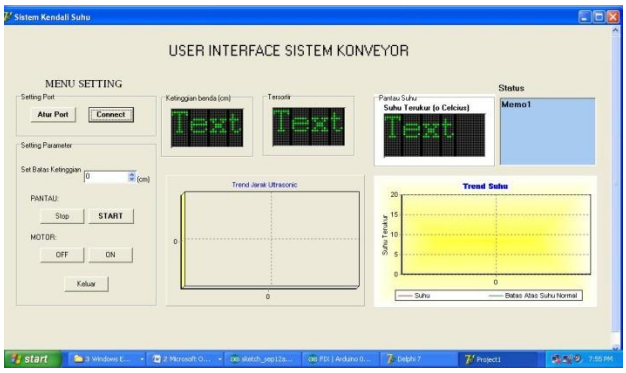
dikirim oleh xPINGqueue dan xSUHUqueue. Pada antar muka ini juga terdapat pengaturan *setpoint* ketinggian minimal barang yang akan disortir. Selain itu, antarmuka tersebut juga berfungsi untuk menampilkan sinyal *emergency* apabila terjadi suhu *overheat*.

2.3 Implementasi Sistem

Agar dapat menjelaskan cara kerja sistem pemilahan barang ini secara lebih detail maka dibuatlah peraga konveyor beserta dengan tuas pendorong yang berguna menyortir barang tersebut. Sistem konveyor tersebut diletakkan pada sebuah papan kayu dengan ukuran 50 cm x 70 cm. Pada papan kayu tersebut diletakkan konveyor dan komponen pendukungnya. Untuk konveyor sendiri berukuran panjang x lebar x tinggi adalah 60 x 7 x 8 cm. Konveyor dibuat dengan bahan dasar aluminium sebagai badan konveyor. Sedangkan bahan yang dipaka untuk balutan alas konveyor yang berputar digunakan karet. Pada bagian motor penggerak konveyor ini, dibuat *gear* untuk menghubungkan motor dengan konveyor.

Gambar 6 Peraga konveyor dan komponen pendukung

Untuk perangkat lunak yang akan ditanamkan pada komputer menggunakan Borland Delphi 7. Bahasa pemrograman yang digunakan pada Borland Delphi ini menggunakan struktur bahasa pemrograman Pascal. Seperti yang sudah dijelaskan pada bab sebelumnya pada panampil ini akan ditampilkan data tinggi benda dari xPINGQueue beserta grafiknya, data suhu dari xSUHUQueue beserta grafiknya, pengaturan setpoint batas bawah benda yang ingin disortir dan kendali hidup mati motor konveyor, serta akan ditampilkan *counter* untuk benda yang sudah disortir.



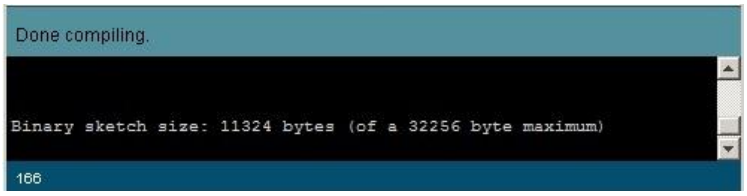
Gambar 7 Tampilan implementasi *graphical user interface*

3. HASIL DAN PEMBAHASAN

3.1 Pengujian Memory

Pada penggunaan *real time operating system* salah satu aspek penting yang perlu diperhatikan adalah aspek penggunaan memori. Memori digunakan untuk menyimpan data dan kode. Biasanya akan memakan banyak memori yang tersedia pada *operating system*. Penggunaan memori juga berbeda-beda bergantung pada jenis RTOS yang dipilih dan memiliki besar memori yang berbeda-beda pula. Pada penelitian ini digunakan DuinOS yang OS ini mempunyai cara pengalokasian memori sama seperti pada FreeRTOS sehingga mudah dalam implementasi dan konfigurasi pada sistem.

Tinjauan pertama dilakukan untuk mengetahui jumlah memori flash yang digunakan saat semua task dijalankan. Pengujian dilakukan dengan cara melakukan *compiling* langsung pada IDE. Jika proses *compiling* berhasil maka akan muncul angka jumlah memori flash yang digunakan seperti pada Gambar 8.

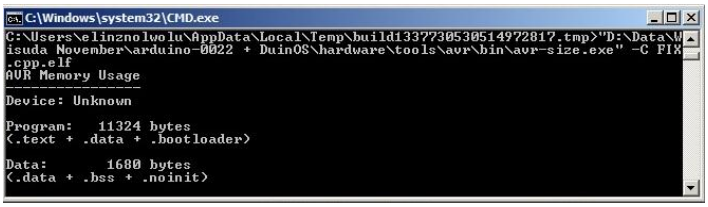


Gambar 8 Hasil pengujian memori flash

Dari cuplikan Gambar 7 terlihat jumlah memori yang digunakan pada memori flash sebesar 11324 byte. Bagian tersebut sudah termasuk untuk bagian *text* untuk *code* program, data, dan *bootloader*. Sehingga memori sisa yang terdapat pada sistem tinggal sebesar 32256 byte dikurangi 11324 byte sama dengan 20932 byte. Dari pengujian tersebut dapat disimpulkan bahwa memori yang tersedia dari segi memori flash masih cukup *space* yang tersedia.

Walaupupun dari segi memori flash masih terdapat cukup banyak *space* yang tersisa, itu belum menjamin bahwa manajemen memori sudah baik untuk sistem dan sistem terbebas dari memori RAM *overflow*. Dari berbagai referensi juga mengatakan bahwa sisi negatif dari penggunaan RTOS adalah boros dari segi penggunaan memori. Oleh karena itu perlu dicek apakah sistem yang sudah dibuat ini mengalami memori RAM *overflow* atau tidak sehingga sistem mampu bekerja dengan baik. Untuk mengetahui jumlah RAM yang digunakan dilakukan motode pengujian menggunakan bantuan program *avr-size*. Avr-size ini sudah terdapat pada folder Arduino IDE tepatnya ada pada folder: hardware\tools\avr\bin. Namun sebelumnya kita harus menemukan folder *compile* yang berisi berbagai file komponen proses *compile*. Pada prinsipnya setelah proses *compile* dilakukan akan ada file yang disimpan pada komputer. Salah satu informasi yang bisa dilihat adalah tentang perkiraan penggunaan memori RAM. Pada Arduino IDE versi 0020 diatas jika setelah proses *compile* selesai folder compile tersebut tersimpan pada folder build***.tmp.

Setelah ditemukan folder compile selanjutnya kembali kepada proses untuk mencari nilai memori RAM yang digunakan. Kita perlu menggunakan bantuan *command prompt* untuk proses eksekusi. Dilanjutkan dengan menulis “lokasi file avr-size\avr-size” – C ***.cpp.elf, dimana *** adalah nama program yang telah dibuat tadi. Jika proses eksekusi berhasil dijalankan akan muncul jumlah memori yang digunakan, baik memori RAM maupun memori flash. Gambar 9 merupakan cuplikan dari hasil pengujian penggunaan memori ini.



Gambar 9 Hasil pengujian penggunaan memori

Terlihat bahwa penggunaan memori flash sebesar 11324 sama seperti hasil pengujian dari yang terpampang pada Arduino IDE setelah proses compile. Kemudian untuk memori RAM yang digunakan sebesar 1680 byte. Sehingga jumlah memori RAM yang masih tersedia sebesar 2048 byte dikurangi 1680 byte sama dengan 368 byte. Dari hasil pengujian tersebut dapat disumplkan bahwa sistem tidak terjadi *overflow RAM memory* karena jumlah memori yang digunakan masih tertampung pada jumlah memori yang tersedia pada sistem.

3.2 Pengujian Pewaktuan

Salah satu faktor penting dalam implementasi RTOS selain penggunaan memori adalah pewaktuan. Pengaturan pewaktuan diperlukan dalam penggunaan RTOS agar koordinasi antar task bisa berjalan dengan baik. Kernel RTOS akan melakukan *context switching* bersasarkan waktu yang ditentukan, dan dalam hal ini digunakan *Clock tick* atau *tick time. Clock tick* merupakan interupsi spesial yang muncul secara periodik. *Clock tick* bisa dianalogikan sebagai detak jantung dari sistem yang berfungsi sebagai dasar untuk menentukan *timer* pada sistem *real time* dengan RTOS. Waktu untuk tiap munculnya *clock tick* dapat ditentukan pada saat perancangan sistem RTOS. Semakin cepat *clock tick*, semakin tinggi frekuensi eksekusi yang bisa dikerjakan namun besar beban yang ditanggung oleh CPU juga semakin tinggi.

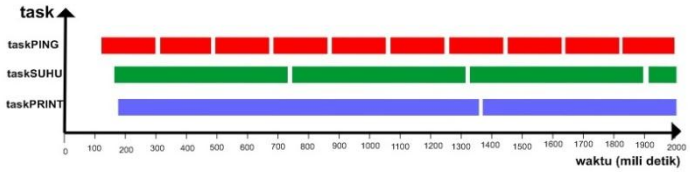
Pengujian tersebut digunakan bantuan fungsi *micros( )*. Fungsi *micros( )* tersebut akan mencatat waktu eksekusi pada program yang diujikan. Untuk pengujian fungsi *micros( )* tadi disisipkan pada masing-masing task dengan 2 lokasi yang berbeda yakni pada awal task dan akhir task kemudian didapat data pengujian pewaktuan seperti pada Tabel 3.

Tabel 2 Hasil pengujian pewaktuan dalam rentang 2 detik

Eksekusi	**taskPING (us)**		**taskSUHU (us)**		**taskPRINT (us)**	
ke-	**Mulai**	**Selesai**	**Mulai**	**Selesai**	**Mulai**	**Selesai**
1	113.988	289.016	164088	738032	183.172	13560000
2	305.676	480.016	754708	1316096	1376432	
3	496676	671016	1333792	1900096		
4	687672	862016	1917800			
5	878676	1053020				
6	1070724	1247016				
7	1264724	1441020				
8	1460672	1637024				
9	1654724	1831016				
10	1848724					

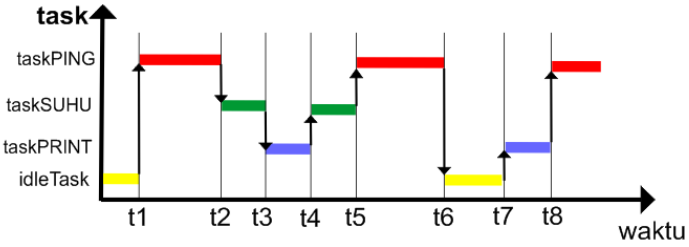
Dari cuplikan pengujian pewaktuan dan rangkuman data pewaktuan seperti pada Gambar 10 dan Tabel 2 awal eksekusi taskPING pada waktu 113.988 mikro detik dan diakhiri pada 289.016 mikrodetik. Akan tetapi ditengah-tengah proses eksekusi tersebut terpotong oleh eksekusi taskSUHU yang dimulai pada 164.088 mikrodetik, task ini juga tidak langsung diselesaikan dalam waktu itu juga namun juga terpotong oleh proses penyelesaian eksekusi taskPING dan taskSUHU ini yang sempat terpotong oleh penyelesaian taskPING hingga diselesaikan pada waktu 738.032 mikrodetik. Disaat yang hampir sama itu taskPRINT juga

memulai proses eksekusi pada waktu 183.172 mikrodetik dan diakhiri pada 13.560.000 mikrodetik. Gambar 10 merupakan diagram pewaktuan dari proses eksekusi masing-masing task.



Gambar 10 Diagram pewaktuan task

Pada prinsipnya setiap satu waktu CPU hanya bisa menjalankan satu task, task akan dimanajemen oleh Kernel sedemikian sehingga bisa dieksekusi bergantian dalam tiap waktunya. Gambar 11 adalah gambar timing diagram yang lebih detil yang menggambarkan koordinasi antar task oleh Kernel dengan *preemptive*.



Gambar 11 Diagram pewaktuan *preemptive* multitasking

Dari Gambar 11 bisa dilihat tentang sistem *preemptive* bekerja, saat Kernel tidak ada task yang diuji maka taskIdle akan dijalankan. Saat taskPING yang mempunyai prioritas tertinggi sudah dalam keadaan *ready* maka task tersebut langsung dijalankan. Saat taskPING dijalankan task ini bisa mendapatkan interupsi dari task yang mempunyai prioritas lebih rendah. Bisa dilihat taskPING mendapatkan interupsi dari taskSUHU dan taskPRINT yang mempunyai prioritas lebih rendah. Pengujian selanjutnya dilakukan untuk mencari lama waktu task untuk sekali dieksekusi.

Tick time default sebesar 1 mili detik dirasa cukup memadahi untuk menjalankan sistem karena waktu eksekusi task paling cepat sudah dalam hitungan ratusan mili detik. Selain itu setelah dijalankan, sistem juga telah bekerja dengan baik. Oleh karena itu pada sistem ini digunakan *tick time* 1 mili detik. Akan tetapi jika pada aplikasi kita ingin mengubah nilai *tick time*, digunakan #define configTICK_RATE_HZ yang terdapat pada FreeRTOS.h.

3.3. Pengujian Pendeteksian Ketinggian Barang

Pada pengujian ini diharapkan sistem pendeteksian ketinggian sudah dapat berjalan dengan baik sehingga proses penyortiran juga akan baik pula. Tinjauan pertama yakni mengetahui kecepatan konveyor yang ada pada sistem. Untuk mengetahui kecepatan konveyor, tinjauan pertama dilakukan dengan mencari lama waktu perjalanan barang dari pangkal konveyor menuju ujung konveyor.

Setelah dilakukan 10 kali pengambilan data kemudian dilakukan perhitungan nilai rata-rata waktu yang diperlukan barang untuk transportiasi dari pangkal konveyor menuju ujung konveyor sebesar 6,28 sekon.

Setelah didapatkan nilai waktu rata-rata, kemudian nilai tersebut dimasukkan kedalam perhitungan untuk mengetahui kecepatan konveyor seperti pada persamaan (1).

$$v\ konveyor = \frac{\text{s jarak tempuh}}{\text{t waktu yang dibutuhkan}} \quad (1)$$

$$v\ konveyor = \frac{0{,}6\ \text{m}}{6{,}28\ s}$$
$$= 9{,}55\ \text{cm/s}$$
$$= 0{,}09\ \text{m/s}$$

Dari perhitungan diatas didapat kecepatan konveyor sebesar 9,55 cm/s atau 0,09 m/s. Dengan kecepatan tersebut tinjauan selanjutnya mengetahui waktu lama barang saat kondisi tegak lurus dengan sensor SRF06. Selanjutnya perhitungan lama waktu tersebut mulai dihitung ketika ujung benda mulai tegak lurus posisinya dengan sensor SRF06 dan diakhiri saat ekor barang mulai meninggalkan area tegak lurus dengan sensor.

Selanjutnya untuk mengetahui lawa waktu barang pada kondisi tersebut bisa diperkirakan dengan perhitungan dari kecepatan konveyor dengan panjang barang peraga yang telah diketahui sebelumnya. Perhitungan waktu tersebut seperti pada persamaan (2).

$$v\ konveyor = \frac{\text{s}}{t} \tag{2}$$
$$\text{t} = \frac{\text{s}}{v\ konveyor}$$
$$= \frac{0{,}05\ \text{m}}{0{,}09\ m/s}$$
$$= 0{,}55\ \text{sekon}$$

Dengan panjang barang 5 cm dan kondisi sumber tegangan 12 volt sehingga menghasilkan kecepatan konveyor sebesar 0,09 m/s. Dengan kecepatan konveyor 0,09 m/s maka barang dengan panjang 5cm akan melewati area sensor selama 0,55 detik. Sehingga didapatkan nilai banyaknya ping yang akan diterima barang saat melewati area sensor seperti pada persamaan (3).

$$\text{Jumlah ping pada barang} = \frac{\text{waktu barang tegak lurus dengan sensor}}{\text{waktu yang dibutuhkan tiap sekali ping}} \tag{3}$$
$$= \frac{0{,}55\ \text{detik}}{0{,}17\ \text{detik}}$$
$$= 3$$

Kemudian pengujian selanjutnya dilakukan untuk mengetahui apakah dengan spesifikasi pendeteksian yang telah diuraikan tersebut sanggup memenuhi tugas dari pendeteksian ketinggian barang yang berjalan pada media konveyor. Pengujian dilakukan dengan menaruh barang dengan ketinggian tertentu kemudian diamati apakah nilai yang terbaca oleh sistem sama dengan nilai yang sesungguhnya.

Tabel 3 Hasil pengujian pendeteksian ketinggian barang

Ketinggian barang (cm)	**Output/ ukur (cm)**
3	3
4	4
5	5
6	6
7	7

Dari data pada Tabel 3 bisa dilihat bahwa nilai tinggi benda sesuai dengan pembacaan pada sistem. Dari percobaan tersebut dapat disimpulkan bahwa motode pendeteksian ketinggian barang sudah mampu memenuhi tugas pendeteksian ketinggian barang dengan baik.

4. KESIMPULAN

Kesimpulan yang dapat diambil dari penelitian ini adalah sebagai berikut:

1. Implementasi RTOS untuk purwarupa sistem penyortiran barang dengan *board* Arduino UNO telah berhasil dilakukan dan sistem dapat bekerja dengan baik.
2. Penggunaan memori flash digunakan oleh sistem sebesar 11.332 byte dan penggunaan memori RAM sebesar 1680 byte sehingga sistem terhindar dari *memory overflow* karena kebutuhan memori masih tertampung oleh kapasitas memori.
3. Diketahui perkiraan penggunaan memori masing-masing task, dimana taskPING menggunakan memori flash sebesar 1.148 byte dan RAM 20 byte, taskSUHU menggunakan memori flash sebesar 1.220 byte dan RAM sebesar 14 byte, dan taskPRINT menggunakan memori flash sebesar 1006 byte dan RAM 14 byte.
4. Diketahui perkiraan waktu eksekusi masing-masing task, dimana taskPING membutuhkan waktu sekitar 176 milidetik, taskSUHU membutuhkan waktu 521 milidetik, dan taskPRINT membutuhkan waktu 1.076 milidetik.
5. *Tick time* default duinOS sebesar 1 mili detik telah dipakai untuk sistem dan sistem dapat bekerja dengan baik.

5. SARAN

Pada penelitian ini masih terdapat banyak hal yang harus disempurnakan. Berikut ini disampaikan saran - saran untuk menyempurnakan penelitian dan sistem yang dibuat.

1. Pembagian *task* beserta prioritasnya bisa lebih diatur lebih baik lagi sehingga *delay* bisa diperkecil.
2. Metode pendeteksian ketinggian benda perlu disempurnakan agar pengukuran dapat lebih akurat dan stabil dengan memberikan sensor tambahan seperti laser untuk *trigger* kerja sensor SRF06.

DAFTAR PUSTAKA

[1] Syamsyuhadi, N., 2011. *Rancang Bangun Sistem Kontrol dan Monitoring Sistem Penyortiran Barang Berdasarkan Ketinggian Berbasis PLC Omron CPM2A*. Elektronika dan Instrumentasi, FMIPA, Universitas Gadjah Mada: Yogyakarta.

[2] Ahmad, A., 2011. *Otomatisasi Pengisi Gula pada Kantong Plastik Berbasis Mikrokontroler*. Surabaya: Jurusan Teknik Elektronika, Politeknik Elektronika Negeri Surabaya.

[3] Wahyudi, R., 2012. *Implementasi RTOS untuk purwarupa sistem pengendalian suhu otomatis dengan pengendali mikrokontroler*. Elektronika dan Instrumentasi, FMIPA, Universitas Gadjah Mada: Yogyakarta.