# Modul -1: Installasi Library freeRTOS dalam IDE Arduino

## Tujuan:

Aktifitas lab ini bertujuan untuk melakukan proses instalasi source-code atau library freeRTOS dalam IDE (Integrated Development Environment) Arduino.

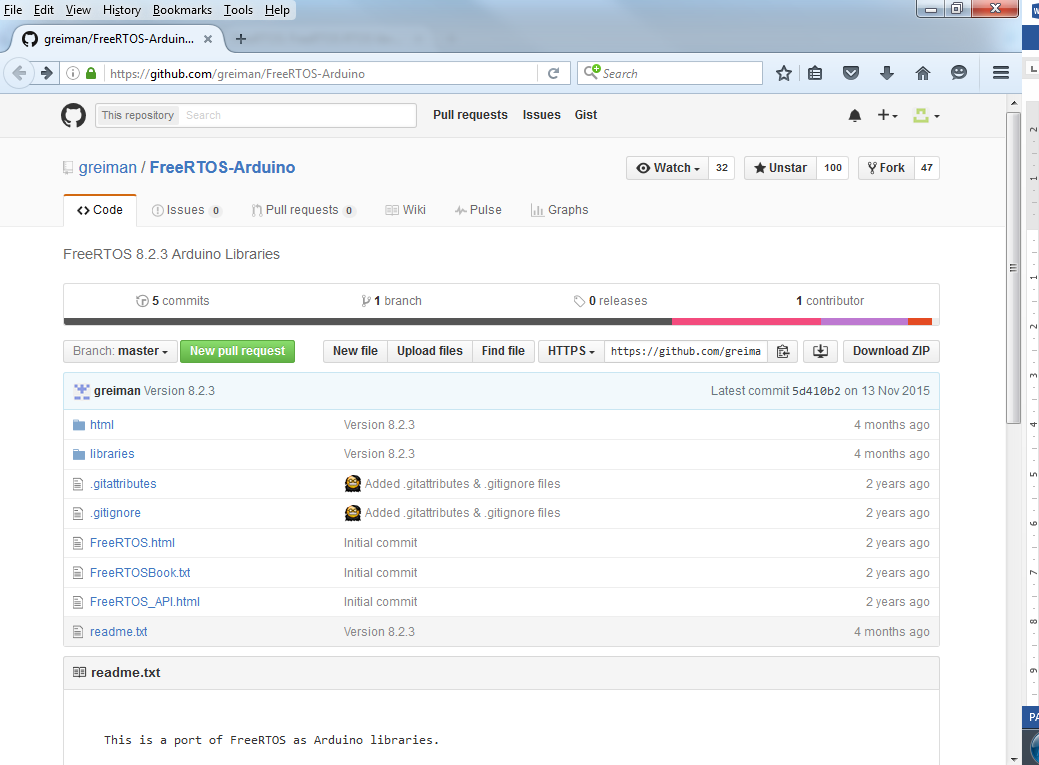
## Peralatan:

Hardware: Arduino UNO Board + kabel USB

Software: (a) Arduino IDE, (b) Library FreeRTOS

## Langkah Kerja Instalasi FreeRTOS:

1. Download source code freeRTOS dari link berikut: <https://github.com/greiman/FreeRTOS-Arduino>. (click pada tombol dengan label ‘Download ZIP’)



Gambar-1. Source code FreeRTOS untuk Arduino

1. Ektrak file zip hasil download pada langkah-1 dalam sebuah folder, dalam file ZIP tersebut dua buah sub-folder (lihat Gambar-2), pastikan salah satu foldernya bernama ‘libraries’.

|  |
| --- |
|  |
| Gambar-2. Struktur folder source-code FreeRTOS |

1. Selanjutnya salin sub-folder dalam folder ‘libraries’ yaitu folder ‘FreeRTOS\_ARM’, ‘FreeRTOS\_AVR’ dan ‘SdFat’ kedalam subfolder ‘libraries’ dalam file instalasi IDE Arduino, biasanya terletak di lokasi folder seperti berikut: ‘C:\Program Files (x86)\Arduino\libraries’, lihat Gambar-3.

|  |
| --- |
|  |
| Gambar-3. Struktur folder Instalasi IDE Arduino |

1. Sampai dengan tahap ini, instalasi FreeRTOS dalam sistem IDE Arduino sudah selesai dan siap digunakan untuk membuat sistem operasi real time dalam Arduino.

## Memeriksa Instalasi:

Untuk memastikan bahwa proses instalasi telah dilakukan dengan benar, maka perlu dicoba satu aplkiasi yang menggunakan libraries ‘FreeRTOS’, hal itu dapat dilakukan dengan cara menjalankan sebuah aplikasi yang disediakan dalam folder ‘Example’ yang disertakan oleh source code FreeRTOS. Jika tidak terdapat kesalahan dalam proses ‘Compilasi’ berarti instalasi

Langkah-langkah untuk menjalankan contoh aplikasi FreeRTOS adalah sebagai berikut:

1. Sambungkan Arduino UNO board dengan komputer menggunakan kabel USB.
2. Jalankan IDE Arduino, sehingga pada layar ditampilkan seperti pada Gambar-4.

|  |
| --- |
|  |
| Gambar-4. Arduino IDE |

1. Buka salah satu contoh aplikasi FreeRTOS, klik menu ‘File’ > ‘Examples’ > ‘FreeRTOS\_AVR’ > ‘FreeRTOSBook’ > ‘Example001’
2. Jalankan menu ‘Verify’, dapat dilkukan dari menu ‘Sketch’ > ‘Verify / Compile’ atau dari tombol paling kiri 
3. Perhatikan informasi mengenai hasil kompilasi, jika terdapat informasi ‘Error’ sehingga proses kompilasi ‘Gagal’ berarti instalasi source code FreeRTOS belum berhasil. Jika terjadi sebaliknya berarti anda telah berhasil menginstalasi FreeRTOS dalam sistem IDE Arduino.
4. Jika proses kompilasi berhasil lanjutkan dengan mengupload aplikasi ke dalam Arduino Board, klik menu ‘File’ > ‘Upload’ atau klik button ‘->’ ( )
5. Output dari program ini dapat dimonitor melalui menu ‘Serial Monitor’, dan akan ditampilkan seperti pada Gambar-5.

|  |
| --- |
|  |
| Gambar-5. Output program ‘Example001.ino’ |

## Kesimpulan:

Pada aktifitas lab ini telah dilakukan instalasi source-code FreeRTOS dalam sistem IDE Arduino. Source code di downlaod dari github.com dan verifikasi intalasi dilakukan dengan cara menjalankan contoh aplikasi FreeRTOS yang disediakan bersama Source-code. Jika menu ‘verify’ berhasil berarti instalasi source code FreeRTOS sudah betul.

# Modul-2: Aplikasi FlipFlop menggunakan FreeRTOS

## Tujuan:

Memahami cara kerja FreeRTOS untuk sebuah kasus sederhana, yaitu mematikan dan menghidupkan lampu LED.

## Peralatan:

Hardware: Arduino UNO Board + kabel USB

Software: (a) Arduino IDE, (b) Library FreeRTOS

## Pendahuluan:

Aplikasi ini merupakan aplikasi yang sangat sederhana, yaitu menghidupkan LED selama waktu tertentu dan kemudian mematikan LED selama waktu tertentu, dan terus menjalankan kedua aplikasi tersebut berulang sampai dengan power off (arduino dimatikan). Jika dilihat dari obyeknya (on/off LED) aplikasi ini memang tidak efisien, tetapi aplikasi ini sangat bagus sebagai sarana untuk memahami cara kerja FreeRTOS dalam board arduino UNO.

Aplikasi ini terdiri dari dua buah TASK, task pertama, diberi nama ‘Thread1’ digunakan untuk mematikan LED (switch OFF), task pertama ini harus menunggu sinyal dari Task ke dua (yang diberi nama ‘Thread2’) untuk mengirimkan data LOW ke pin yang terhubung dengan LED (pin 13).

static void Thread1(void\* arg) {

while (1) {

//menunggu sinyal dari Thread2

xSemaphoreTake(sem, portMAX\_DELAY);

digitalWrite(LED\_PIN, LOW);

}

}

Sedangkan Thread2 berfungsi untuk menghidupkan LED (switch ON), sinyal baru diberikan ke pada ‘Thread1’ ketika LED sudah menyala (ON) selama waktu yang ditentukan oleh fungsi (vTaskDelay). Kode selengkapnya dari Task kedua (Thread2) seperti ditunjukkan di bawah ini:

static void Thread2(void\* arg) {

pinMode(LED\_PIN, OUTPUT);

while (1) {

**// Turn LED on.**

digitalWrite(LED\_PIN, HIGH);

**// Sleep for 200 milliseconds.**

vTaskDelay((200L \* configTICK\_RATE\_HZ) / 1000L);

**// Signal thread 1 to turn LED off.**

xSemaphoreGive(sem);

**// Sleep for 200 milliseconds**.

vTaskDelay((200L \* configTICK\_RATE\_HZ) / 1000L);

}

}

Pada aplikasi ini prioritas task paling tinggi (nomor 2) diberikan pada task ke-1 (‘Thread1’).

## Langkah Kerja:

1. Jalankan program IDE Arduino, dan buat aplikasi baru bernama ‘frFlipFlop.ino’, tuliskan kode pada gambar-1 dalam aplikasi baru anda.

|  |
| --- |
|  |
| Gambar-1 . Code aplikasi flip-flop dengan FreeRTOS |

1. Upload aplikasi di atas ke dalam board Arduino UNO dan amati LED yang terhubung pada pin 13.
2. Jika tidak ada kesalahan maka lampu LED yang terhubung dengan pin 13 akan menyala berkedip (flip-flop).
3. Coba lakukan modifikasi program agar waktu menyala lebih lama dibanding waktu matinya LED.

## Kesimpulan:

Pada aktiftas lab ini telah dibuat sebuah program Flip-Flop berbasis pada FreeRTOS. Terdapat dua buah task (aplikasi) yaitu aplikasi untuk mematikan (switch-off) dan aplikasi untuk menghidupkan (switch-on) LED.

# Modul-3: Multitasking menggunakan FreeRTOS dalam Arduino

## Tujuan:

Memahami cara kerja FreeRTOS untuk menjalankan tiga buah aplikasi secara bersamaan dengan menggunak konsep multitasking.

## Peralatan:

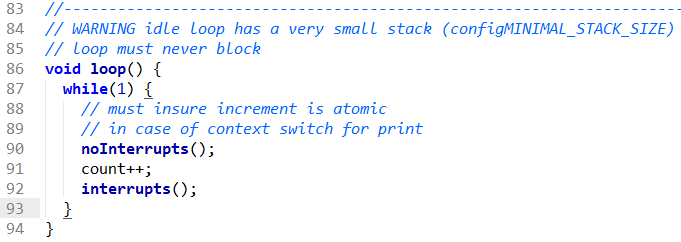
Hardware: Arduino UNO Board + kabel USB

Software: (a) Arduino IDE, (b) Library FreeRTOS

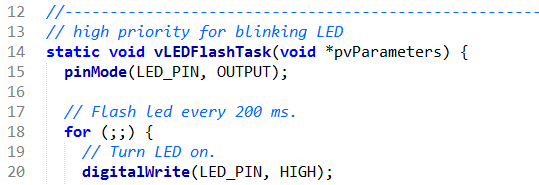
## Pendahuluan:

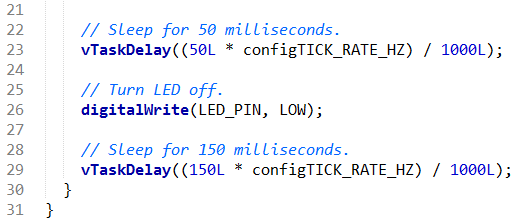
Pada aktifitas lab ini akan dibuat TIGA buah aplikasi (atau disebut dengan TASK dalam freeRTOS) yang di jalankan secara simultan , atau multitasking, oleh sistem operasi freeRTOS dalam Arduino. Aplikasi terdiri atas (1) Menghitung (menambah nilai sebuah variabel dengan angka satu atau increment), (2) Membuat FLIP-FLOP, mengontrol LED pada pin 13 dengan delay waktu tertentu, dan aplikasi (3) adalah menanmpilkan hasil hitungan yang dilakukan oleh aplikasi-1, dan menampilkan informasi mengenai status penggunaan RAM (stack memory) yang digunakan oleh aplikasi.

Source code dari aplikasi adalah sebagai berikut: (a) Aplikasi-1: Menghitung (manambah nilai variabel ‘count’).

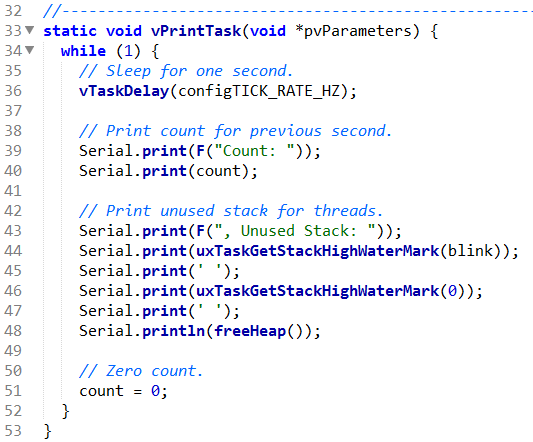


(b) Aplikasi-2: Aplikasi untuk membuat FLIP-FLOP



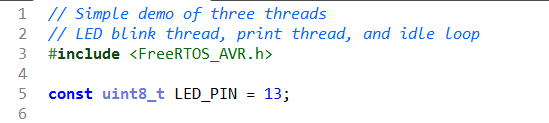


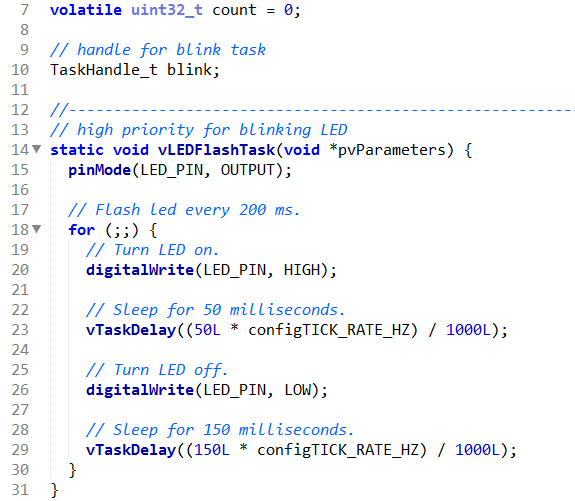
(c ) Aplikasi-3: Mencetak hasil hitungan dari aplikasi-1 melalui serial monitor.

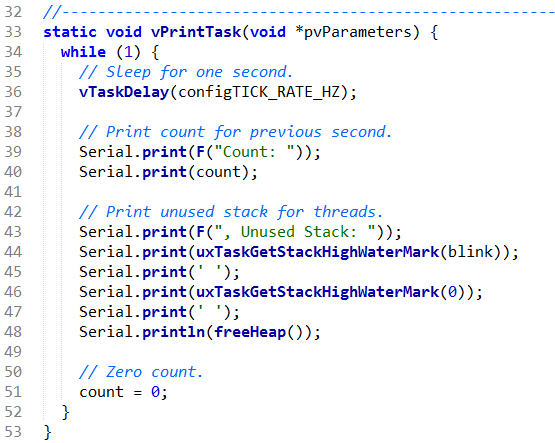


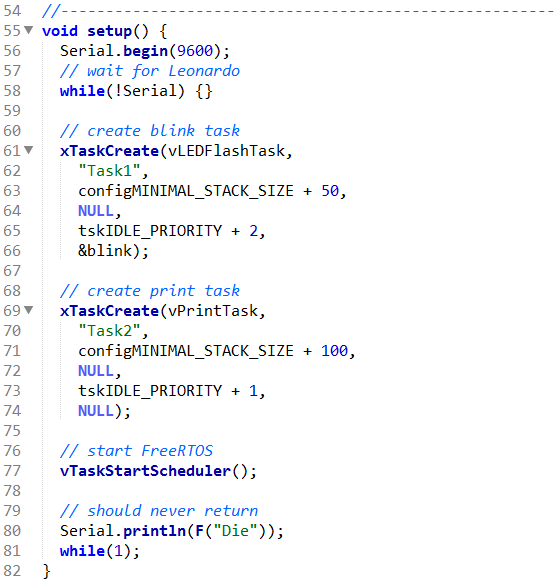
## Langkah Kerja

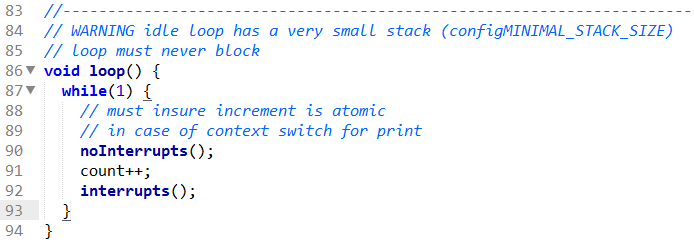
1. Jalankan program arduino IDE, buat file baru bernama ‘multitasking.ino’, kemudian buat kode perogram seperti pada gambar di bawah.



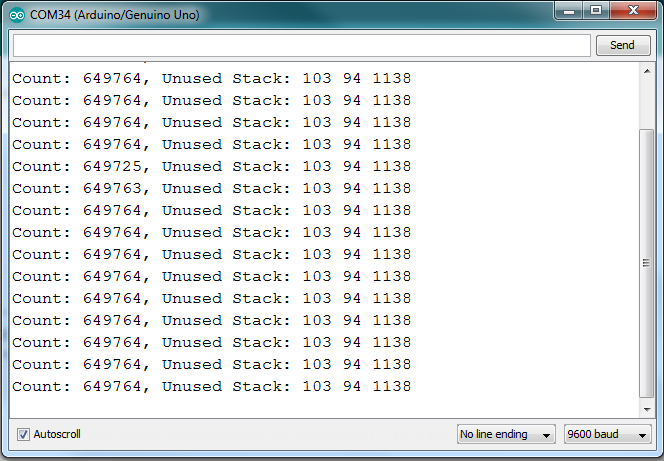








1. Upload ke dalam board arduino, buka menu ‘Serial Monitor’ dan amati LED pada pin 13, serta tampilan pada Serial Monitor.
2. Contoh tampilan pada Serial Monitor adalah seperti di tamapilkan pada gambar di bawah:



Gambar 3-1. Hasil monitoring program multitasking

## Kesimpulan:

Pada aktifitas lab ini, telah di buat program untuk menjalankan tiga buah aplikasi yang dijalankan secara simultan (multi tasking) dengan menggunakan sistem operasi FreeRTOS. Aplikasi terdiri atas aplikasi menghitung angka, menampikan hasil perhitungan, dan pada saat sama juga mengatur LED (flip-flop).

# Modul-4: Manajemen Task

## Tujuan:

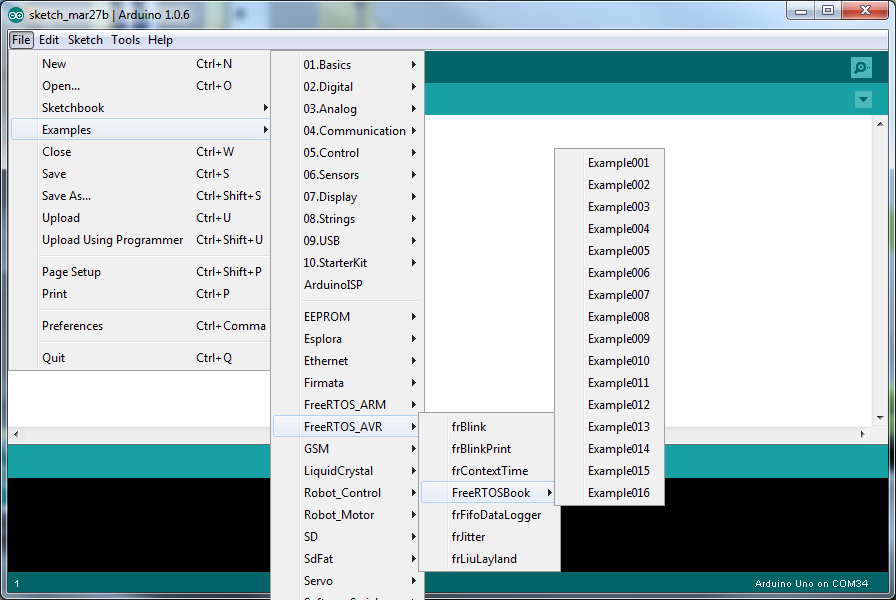
1. Memahami konsep manajemen task (serupa dengan aplikasi dalam sistem operasi window atau linux) dalam sistem operasi real time (freeRTOS)
2. Memahami cara membuat sebuah task
3. Memahami konsep prioritas task
4. Memahmi cara mengontrol task yang memiliki task paling tinggi agar tidak mendominasi proses eksekusi task (cara menghentikan proses prioritas paling tinggi).
5. Memahi konsep ‘idle-task’ (example007)
6. Memahami cara merubah prioritas sebuah task. (Example008)
7. Memahami cara menghapus task. (Example009)

## Peralatan:

Hardware: Arduino UNO Board + kabel USB

Software: (a) Arduino IDE (**pastikan pakai versi 1.0.6**),

(b) Library FreeRTOS



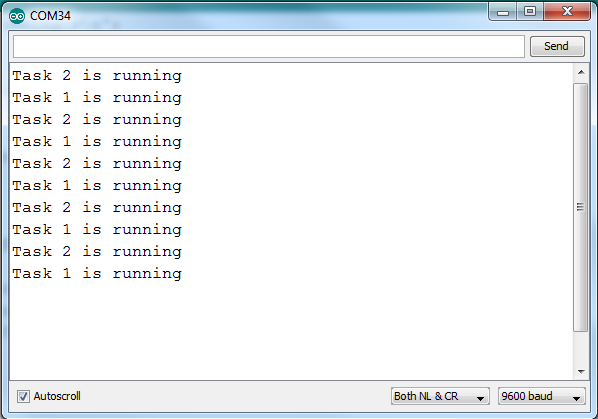
Gambar 4-1. Menu untuk membuka contoh program manajemen task.

## Pendahuluan:

Materi praktikum ini di ambil dari contoh kode program yang diberikan oleh libraries freeRTOS, bagian ‘freeRTOSBook’, yaitu contoh program yang diberikan dalam buku freeRTOS. Bagian Manajemen task terdiri dari contoh program nomor ‘Example001’ sampai dengan ‘Example009’.

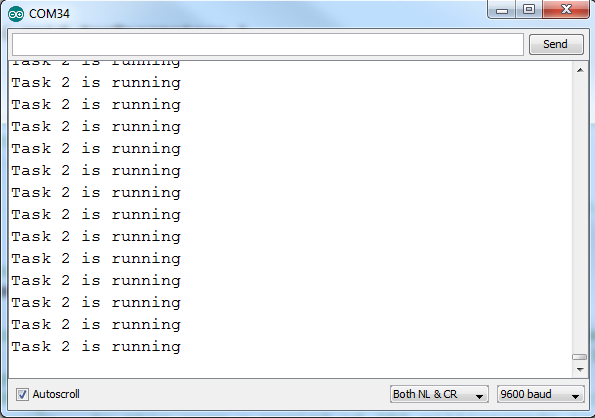
## Cara kerja:

1. Belajar membuat task sederhana: (a) Buka contoh program ‘Example001’, dari menu ‘File|Examples|FreeRTOS\_AVR|FreeRTOSBook|Example001’. Program ini merupakan program aplikasi untuk menjalankan dua buah task dengan prioritas yang sama. Task berupa aplikasi mencetak nama task ke ‘Serial Monitor’ dan menunggu untuk sementara waktu kemudian mencetak lagi dan seterusnya. Contoh hasil keluaran program ini adalah seperti gambar berikut:



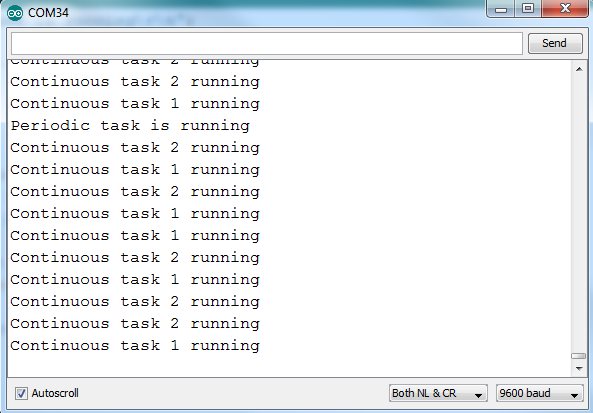
Gambar 4-2. Output dari program ‘Exampl001’

1. Membuat DUA BUAH TASK dari SATU fungsi yang SAMA (‘**vTaskFunction**’) dengan parameter yang berbeda: Buka contoh program ‘Example002’ dengan cara seperti sebelumnya. Hasilnya sama seperti pada program ‘Example001’. Perbedaan terletak pada cara menyusun program aplikasinya. Dalam contoh ini dua buah task tersebut juga memiliki nomor prioritas yang sama.
2. Menjalankan dua buah task dengan nomor prioritas yang berbeda: Jalankan contoh program ‘Example003’. Dalam program tersebut task-1 memiliki prioritas 1 dan task-2 memiliki nomor prioritas-2. Hasilnya task-1 tidak pernah tereksekusi. Penjelasan dapat dilihat di buku ‘Using The FreeRTOS Real Time Kerneel’ halaman-16. Contoh hasil di tampilan Serial Monitor.



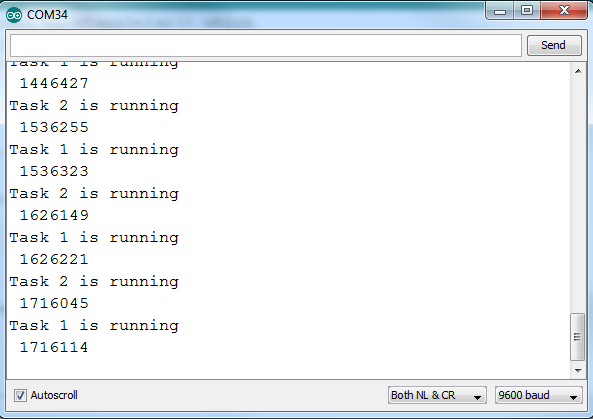
Gambar 4-3. Output program ‘Example003’

1. Mengontrol task dengan menggunakan ‘vTaskDelay’ agar task dengan prioritas paling tinggi tidak mendominasi sistem operasi. Buka contoh aplikasi ‘Example004’. Terdapat perintah ‘vTaskDelay( 250 / portTICK\_PERIOD\_MS );’ sekarang dua buah task dengan prioritas yang berbeda dapat dieksekusi semua. Contoh output seperti pada gambar 4-2.
2. Mengatur sebuah task agar di eksekusi setiap selangg waktu tertentu (sesuai dengan waktu yang ditentukan). Nama API (fungsi) yang digunakan untuk melakukan hal ini adalah ‘vTaskDelayUntil’. Buka contoh program ‘Example005’, ubah waktu yang semula 250 menjadi 3000, amati apa yang terjadi. Contoh output seperti pada gambar 4-2,, dengan waktu kemunculan yang berebeda sesuai dengan waktu yang diberikan
3. Mengatur task agar dieksekusi secara periodik (prioritas tertinggi) bersama dengan dua buah task lain yang dieksekusi secara acak (continues) dengan prioritas lebih rendah. Buka contoh program ‘Example006’, task-1 dan task-2 memiliki prioritas yang sama (yaitu 1) kedua task ini akan dieksekusi secara kontinyu secara bergantian. Sedang Task-3 memiliki prioritas 2 (lebih tinggi) dan akan di eksekusi setiap periode waktu yang ditentukan. Contoh output seperti gambar berikut:



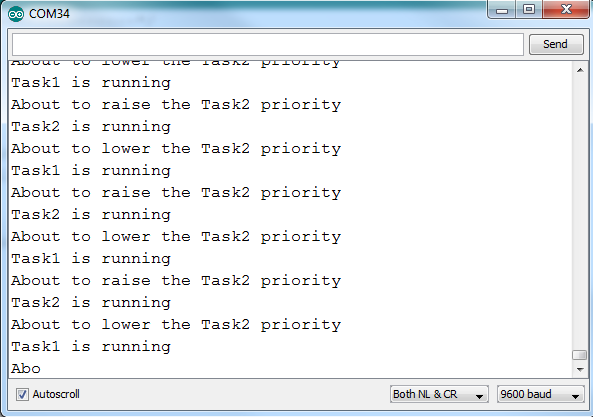
Gambar 4-4. Output contoh program ‘Example006’

1. Membuat fungsi yang akan dieksekusi pada saat sistem operasi dalam keadaan ‘IDLE’ (menganggur atau tidak ada task yang sedang dijalankan). Buka contoh program ‘Example007’. Nama fungsi yang akan dipanggil saat IDLE time adalah ‘vApplicationIdleHook’, biasanya fungsi ini digunakan untuk mengatur sistem dalam mode POWER SAVE. Pada contoh ini, saat idle sistem akan menambah nilai sebuah variabel integer (ulIdleCycleCount++). Contoh output seperti pada gambar berikut:



Gambar 4-5. Output contoh program ‘Example007’

1. Merubah prioritas sebuah task pada saat sistem sudah berjalan. Buka contoh program ‘Example008’. Nama fungsi yang digunakan untuk merubah prioritas adalah ‘vTaskPrioritySet’ dalam contoh ini task-1 (prioritas 2) dan task-2 (prioritas 1) , pada saat task-1 di eksekusi, prioritas task-2 ditambah dengan 1 (dinaikkan) menjadi dua (2).



Gambar 4-6. Output contoh program ‘Example008’

1. Menghapus dan membuat sebuah task pada saat sistem sudah berjalan. Buka contoh program ‘Example009’. Pada contoh ini pertama kali hanya task-1 yang di ‘create’ setelah sistem dijalankan, task-1 akan meng-create task ke dua yaitu ‘task-2’ dengan perintah berikut: ‘xTaskCreate( vTask2, "Task 2", 200, NULL, 2, &xTask2Handle );’ . Pada saat task-2 sudah di jalankan pada akhir baris program menghapus dirinya sendiri dengan perintah ‘vTaskDelete’.

## Kesimpulan:

Pada Aktifitas lab ini telah dilakukan berbagai percobaan yang berkaitan dengan menejemen task. Pengendalian eksekusi task dapat dilakukan dengan cara mengatur nomor prioritas atau mengatur delay, atau dengan logika tertentu untuk meng-create dan menghapus task pada saat sistem sudah berjalan. IDLE TIME dapat dimanfaatkan untuk melakukan sesuatu yang membuat sistem menjadi lebih handal. Misalnya untuk mengatur mode sistem dalam POWER SAVE.

# Modul-5: Manajemen QUEUE dan INTERUPSI

## Tujuan:

1. Memahami konsep manajemen queue (antrian)
2. Memahami cara menggunakan manajemen queue dalam aplikasi freeRTOS.
3. Memahami program denga QUEUE berupa angka integer
4. Memahami program dengan QUEUE tediri dari beberapa tipe data yang berbeda (compound)
5. Memahami cara mengatur interupsi (manajemen interupsi)
6. Memahami porgram dengan menggunakan fasilitas interupsi.

## Peralatan:

Hardware: Arduino UNO Board + kabel USB

Software: (a) Arduino IDE (**pastikan pakai versi 1.0.6**),

(b) Library FreeRTOS

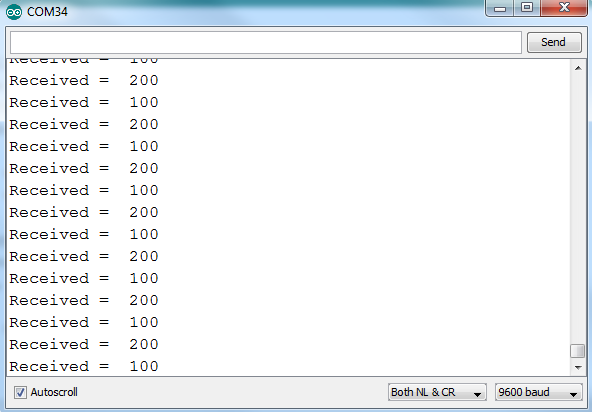
## Pendahulaan:

Setiap aplikasi dalam sistem FreeRTOS disusun dari beberapa task yang TIDAK SALING BERGANTUNG (bebas). Setiap TASK serupa dengan sebuah program yang berjalan sendiri-sendiri di atas sistem operasi FreeRTOS. Agar diantara task dapa saling berkomunikasi, dapat digunakan fasilitas yang disebut QUEUE. Queue dalam freeRTOS berupa sebuah koleksi angka dalam memori yang dapat diakses dengan metode FIFO (First In First Out) oleh semua task yang ada agar task dapat saling bekerja sama.

Biasanya sistem embedded real time harus memberi response tertentu terhadap perubahan yag terjadi disekitarnya, contoh adanya paket data yang datang lewat Serial port, perubahan level tegangan pada pin tertentu dll. Diperlukan manajemen interupsi untuk mengatur situasi tersebut.

## Cara kerja:

1. Membuat aplikasi QUEUE sederhana: Buka contoh program ‘Example010’. Program ini membuat QUEUE terdiri dari variabel ‘long’ sebanyak 5 buah, dengan perintah ‘xQueue = xQueueCreate( 5, sizeof( long ) );’. Teredapat dua task yang akan mengirim (menulis) angka berbeda (100 dan 200) ke dalam QUEUE, kedua task tersebut memiliki prioritas yang sama. (sender-1 dan sender-2). Task ke tiga dengan prioritas tertinggi, yaitu 2, berfungsi sebagai penerima (membaca) data dalam QUEUE.



Gambar 5-1. Output contoh program ‘Example010’

1. Menggunakan QUEUE dengan tipe data yang berbeda (tidak hanya angka integer): Buka contoh program ‘Example011’. Program ini menggunakan QUEQUE dengan data berupa ‘struct’ seperti berikut:

typedef struct

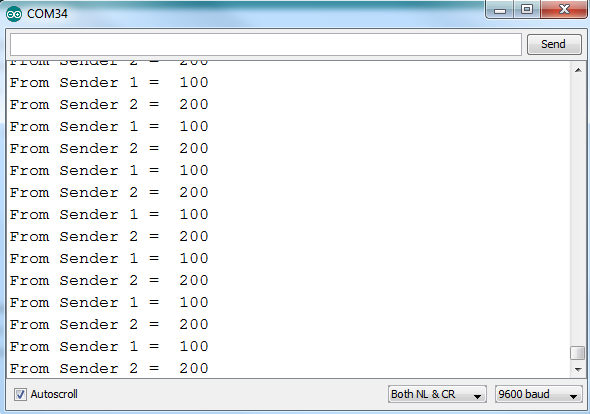
{

unsigned char ucValue;

unsigned char ucSource;

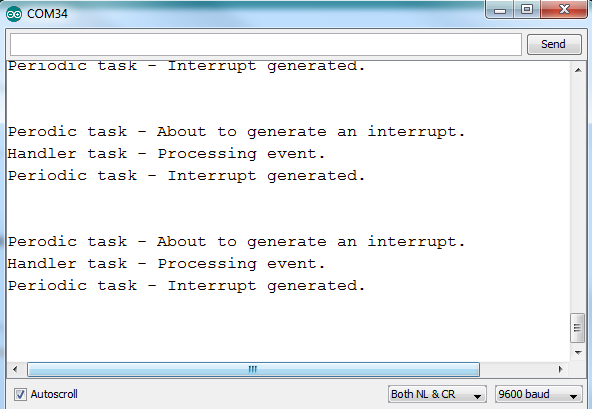
} xData;

Kasusnya seperti pada contoh program ‘Example010’ hanya tipe data yang berbeda. Contoh output seperti pada gambar berikut:



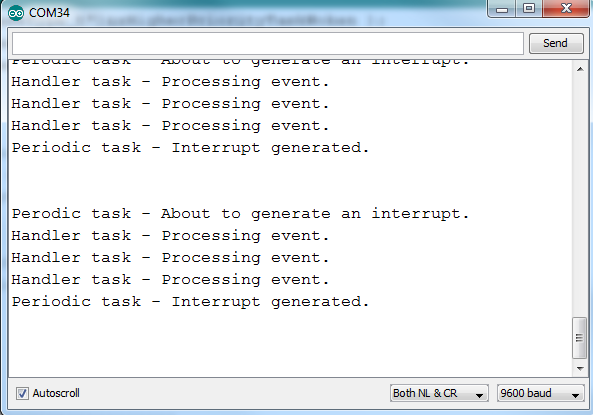
Gambar 5-2. Output contoh program ‘Example011’

1. Manajemen Interupsi: Buka contoh program ‘Example012’. Pada contoh ini dibuat sebuah contoh progam untuk mengontrol interupsi yang datang dari pin nomor 2 (board UNO pin 2 & 3 dapat digunakan sebagi sumber interupsi hardware). Tedapat dua buah task, task-1 (‘Handler’) digunakan untuk meresponse sinyal interupsi yang terjadi pada pin 2. Task-2 generator sinyal interupsi (pembangkit sinyal interupsi). Dalam kasus ini respon diberikan oleh task-1 setelah ada ijin (xSemaphoreTake( xBinarySemaphore, 0); ), ijin akan diberikan oleh fungsi ‘vExampleInterruptHandler’ dengan perintah ‘**xSemaphoreGiveFromISR**’. Contoh output program ini ditampilkan pada gambar berikut:



Gambar 5-3. Output contoh program ‘Example012’

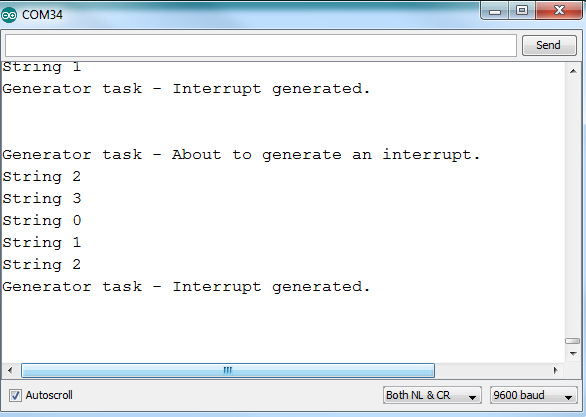
1. Siumlasi beberapa sumber interupsi: Buka contoh program ‘Example013’. Pada contoh ini diberikan contoh cara mengendalikan sinyal interupsi yang datang dari beberapa sumber. Simulasi diberikan dengan cara menjalankan perintah ‘xSemaphoreGiveFromISR’ sebanyak TIGA kali dan akan di response uga sebanyak tiga kali. Contoh output ditunjukkan pada gambar di bawah:



Gambar 5-4. Output contoh program ‘Example013’

1. Menggunakan QUEUE dalam manajemen interupsi: Buka contoh program ‘Example014’. Seperti pada cotoh sebelumnya, pada contoh ini task generator sinyal interupsi juga berfungsi sebagai pengirim (penulis QUEUE) berupa 5 angka integer, yang akan di terima

Oleh task dengan prioritas tertinggi (2) untuk di tampilkan pada Serial monitor. Contoh output seperti pada gambar di bawah:



Gambar 5-5. Output contoh program ‘Example014’

## Kesimpulan:

Pada praktikum ini telah di lakukan percobaan untuk menggunakan fasilits QUEUE dan cara mengendalikan sinyal interupsi dengan menggunakan sistem operasi FreeRTOS.

# Modul-6: Manajemen RESOURCE

## Tujuan:

1. Memahami konsep manajemen resource (sumber daya)
2. Memahami program freeRTOS dengan mutex sebagai alat untuk manajemen resource
3. Memahami program freeRTOS untuk membuat beberap task yang akan menggunakan fasilitas print (Serial print) secara bersamaan

## Peralatan:

Hardware: Arduino UNO Board + kabel USB

Software: (a) Arduino IDE (**pastikan pakai versi 1.0.6**),

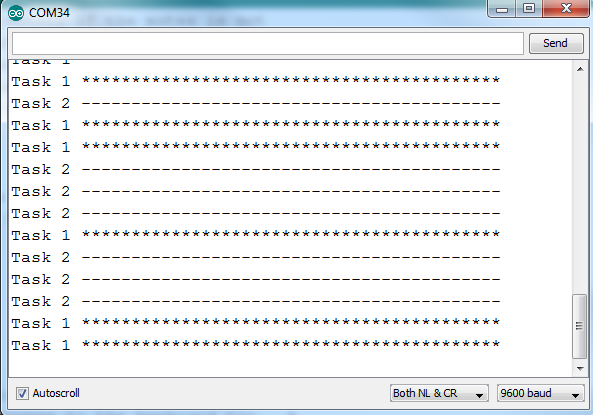
(b) Library FreeRTOS

## Pendahulaan:

Salah satu tema penting yang lain dalam sistem operasi adalah manajemen resource, yaitu pengaturan penggunakan sumber daya yang terdapat dalam sistem, termasuk sumber daya dalam sistem ARDUINO UNO antara lain adalah Serial monitor dan LED pin. Pada praktikum kali ini akan dibuat program untuk mengatur penggunaan fasilitas cetak (Serial Print) oleh dua buah task (aplikasi) dengan menggunakan SEMAP{HORE (mutex).

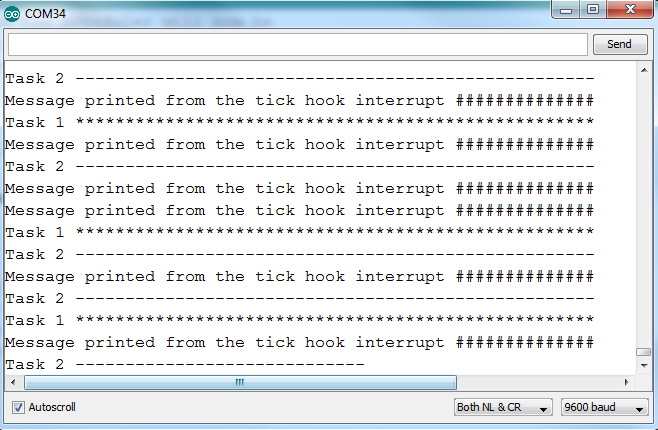
## Cara kerja:

1. Manajemen fasilitas PRINT (Serial print dalam arduino uno). Buka contoh program palikasi ‘Example015’. Pada aplikasi ini dibuat dibuah task dari sebuah fungsi yang sama dengan prioritas berbeda. Keduanya memerlukan fasilitas Serial.Print untuk mencetak data. Sebelum proses mencetak dilakukan tahap pemeriksaan terhadap semaphore (xMutex) untuk mengetahui apakah printer (serial.Print) sedang digunakan oleh aplikasi yang lain. Contoh output ditunjukkan seperti gambar di bawah:



Gambar 6-1. Output contoh program Example015

1. Manejemen RESOURCE dengan menggunakan QUEUE. Buka contoh program ‘Example016’. Logika dalam program ini sama seperti pada contoh program sebelumnya. Perbedaan terletak pada penggunaan QUEUE dan terdapat satu task lain yanhg berfungsi sebagai koordinator, yaitu task dengan nama ‘prvStdioGatekeeperTask’. Contoh output ditunjukkan pada gamabr di bawah:



Gambar 6-2. Ouptut contoh program ‘Example016’

## Kesimpulan :

Pada praktikum ini telah di jalankan program freeRTOS untuk pengaturan resouce (manajemen resource). Dalam contoh kasus ini digunakan Serial.Print sebagai contoh RESOURCE yang akan digunakan oleh beberapa aplikasi pada saat yang sama.