|  |  |  |
| --- | --- | --- |
| **KELOMPOK** | 4 | **ACC** |
| **Tanggal Praktikum** | 27 Maret 2018 | Ast1 |
|  | Ridho Surya Kusuma (D400150096) | Ast2 |
|  | Raka Siwi Pamungkas (D400150107) | **Tanggal ACC :** |
|  | Rahmad Bahrudin Afrianto (D400150156) | **Revisi Tanggal :** |

**MODUL 1**

**ANALOG DIGITAL READ DENGAN FREERTOS**

1. **TUJUAN**
2. Praktikan dapat memahami fungsi Operating System yang ada pada arduino, salah satunya FreeRTOS.
3. Praktikan dapat mensimulasikan contoh program yang ada pada FreeRTOS.
4. Praktikan dapat menganalisa program yang sudah dibuat dan disimulasikan dengan OS FreeRTOS.
5. **ALAT DAN BAHAN**
6. Laptop / PC yang telah diinstal software Arduino
7. Arduino Uno
8. LED merah
9. LED RGB
10. Kabel jumper
11. Project Board
12. **DASAR TEORI**

**FreeRTOS**

FreeRTOS (Free Real-time operating systems) adalah sebuah rela time operating system untuk embeded devices. freeRTOS ini banyak digunakan oleh microcontrollers untuk kebutuhan sistem operasinya. FreeRTOS didistribusikan secara gratis lewat GPL dengan beberapa pengecualian untuk kalangan industri tertentu. Beberapa varian microcontrollers yang menggunakan FreeRTOS antara lain: ARM architechture, Atmel AVR, AVR32, PIC Microcontrollers, MicroBlaze dan masih banyak lagi.

Yang dimaksud dengan RTOS (Real Time Operating System) adalah sistem operasi yang digunakan untuk memenuhi kebutuhan aplikasi yang bersifat real time. Real time disini berarti ia membutuhkan waktu kinerja setiap saat dimana ia dibutuhkan saat itu juga. Salah satu kunci dari keberhasilan RTOS adalah kemampuannya untuk melakukan kerja secara konsisten baik secara waktu yang ia butuhkan maupun secara task aplikasi yang mampu ia kerjakan.

RTOS memiliki kemampuan algoritma untuk proses penjadwalan. Penjadwalan memungkinkan fleksibilitas dalam melakukan orchestration dari proses dan melakukan proses prioritas dari sistem komputer.

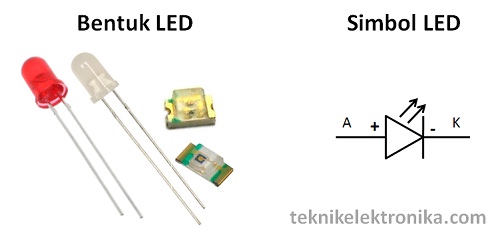
Desain yang sering digunakan untuk RTOS biasanya menggandung:

Event driven, pemrograman di dalam FreeRTOS dan RTOS pada umumnya berlandaskan pada event-driven, yaitu program berjalan berdasarkan event-event yang ada dan telah didefinisikan sebelumnya. Event-driven erat kaitannya dngan sensor dan interaksi user dalam pengerjaannya.

Time-sharing, yang bertugas melakukan kontrol terhadap clock interrupt dan on event atau yang lebih sering disebut dengan round robbin (sebuah algoritma penjadwalan yang sering digunakan pada FreeRTOS adalah sebuah sistem operasi yang dibuat dalam skala kecil dan sederhana. Kernel FreeRTOS hanya terdiri dari tiga atau empat file C (biasanya kernel FreeRTOS dibangun dari bahasa C). Penggunaan bahasa C digunakan agar memudahkan untuk code readable, memudahkan koneksi ke port, dan mudah dalam melakukan perawatan

**LED**

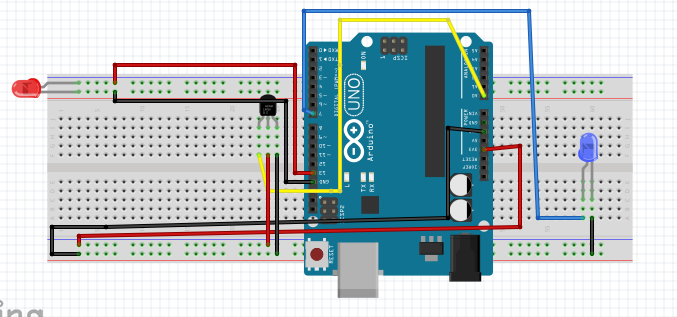
Light Emitting Diode atau sering disingkat dengan LED adalah komponen elektronika yang dapat memancarkan  cahaya monokromatik ketika diberikan tegangan maju. LED merupakan keluarga Dioda yang terbuat dari bahan semikonduktor. Warna-warna Cahaya yang dipancarkan oleh LED tergantung pada jenis bahan semikonduktor yang dipergunakannya. LED juga dapat memancarkan sinar inframerah yang tidak tampak oleh mata seperti yang sering kita jumpai pada Remote Control TV ataupun Remote Control perangkat elektronik lainnya.



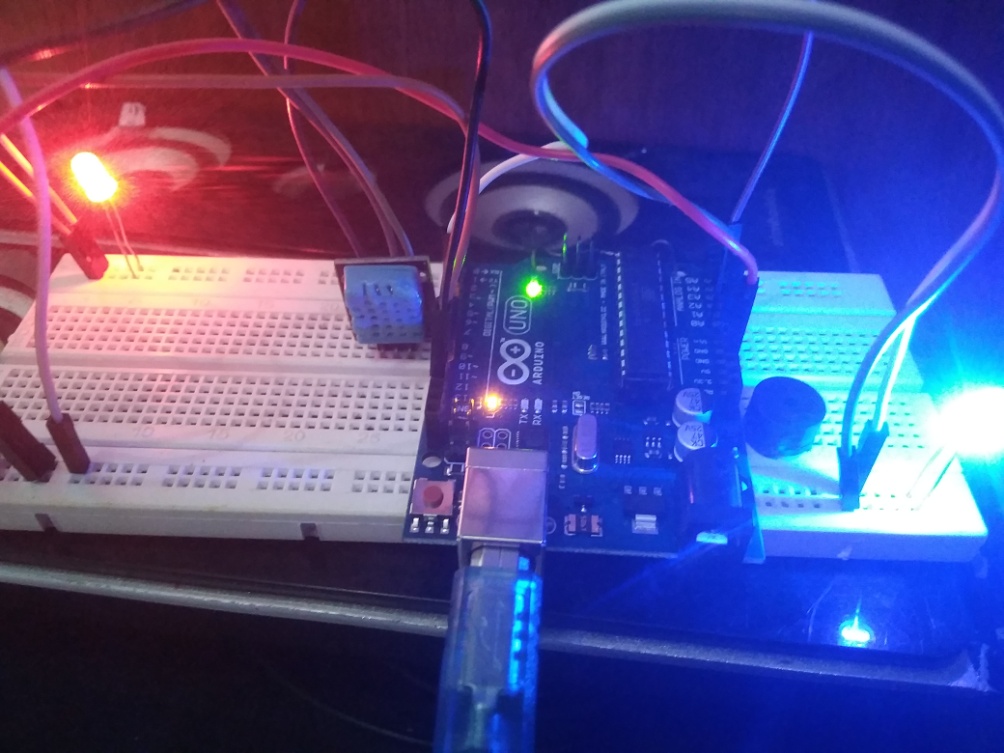
1. **HASIL PRAKTIKUM**

D.1 Hasil Percobaan 1

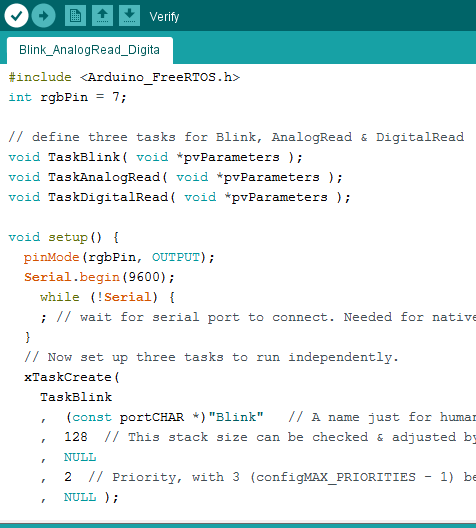
D.1.1 Skema Rangkaian



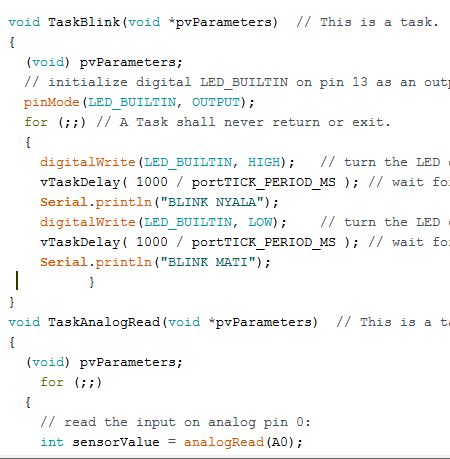
D.1.2 Foto Rangkaian

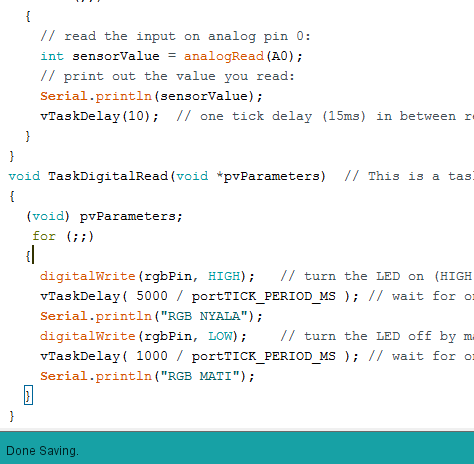


D.1.3 Script Program









1. **ANALISA**

E.1. Analisa Percobaan 1

Pada percobaan ini menggunakan 1 buah Arduino Uno, 1 buah LED merah, 1 LED RGB 2 kaki, dan sensor sebagai pendukung pembacaan analog dari arduino. Sedangkan percobaan ini memiliki 3 Task yaitu task LedBlink, task AnalogRead dan task DigitalRead, yang secara berurutan memiliki nilai prioritas 2, 1, 3. Cara kerja pada percobaan ini yaitu task AnalogRead membaca setiap 10ms, sedangkan task LedBlink hidup setiap 1000ms dan mati selama 1000ms. Lalu untuk task DigitalRead yaitu menyalakan RGB Led selama 2000ms dan mati 1000ms.

1. **KESIMPULAN**
2. FreeRTOS (Free Real-time operating systems) adalah sebuah rela time operating system untuk embeded devices.
3. Untuk dapat menggunakan operating system FreeRTOS, terlebih dahulu menginstal library FreeRTOS.
4. Dalam FreeRTOS “void loop()” pada arduino ide tidak digunakan karena fungsi looping ditaruh secara tersendiri pada setiap task yang sudah dibuat.
5. Prioritas sebuah task tergantung nilai yang dibuat, semakin besar nilai maka semakin tinggi prioritas task tersebut.

|  |  |  |
| --- | --- | --- |
| **KELOMPOK** | D | **ACC** |
| **Tanggal Praktikum** |  | Ast1 |
|  | Ridho Surya Kusuma (D400150096) | Ast2 |
|  | Raka Siwi P (D400150107) | **Tanggal ACC :** |
|  | Rahmad Bahrudin A (D400150156) | **Revisi Tanggal :** |

**PROJECT MODUL 1**

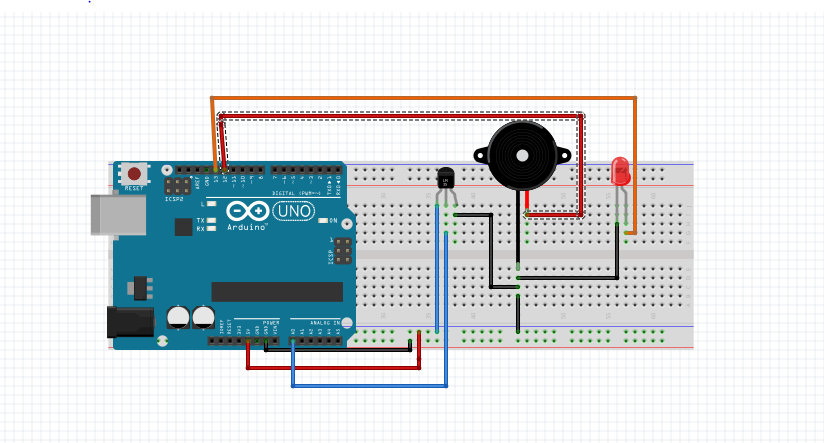
**ANALOG DAN DIGITAL WRITE DENGAN FREERTOS**

1. **TUJUAN**
2. Praktikan dapat memahami salah satu sistem operasi yang ada pada Arduino yaitu FreeRTOS.
3. Praktikan dapat memahami program Analog Digital Read menggunakan sistem operasi FreeRTOS.
4. Praktikan dapat mengembangkan program dengan menggunakan sistem oprasi FreeRTOSdan mengaplikasikan dalam bentuk hardware.
5. **ALAT DAN BAHAN**
6. Laptop / PC yang telah diinstal software Arduino
7. Arduino Uno .
8. Buzzer
9. Sensor Suhu LM35
10. Kabel jumper
11. LED

**C. HASIL PROJEK**

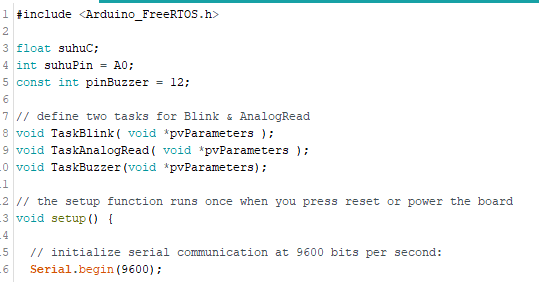
**C.1. PERCOBAAN 1**

**C.1.1. SKEMA RANGKAIAN**

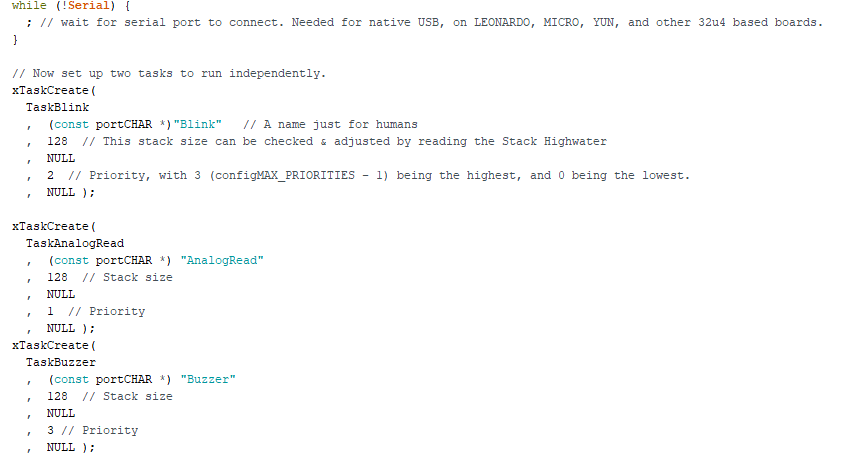
****

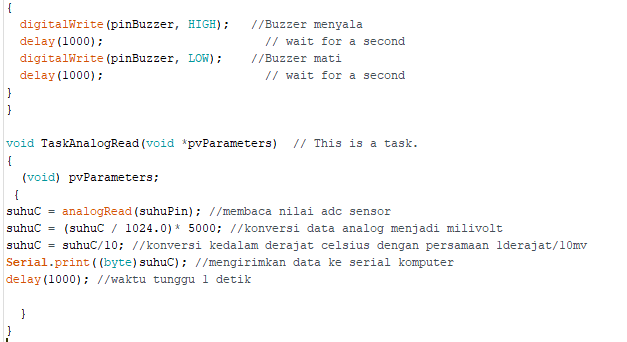
**C.1.2. FOTO RANGKAIAN**

****

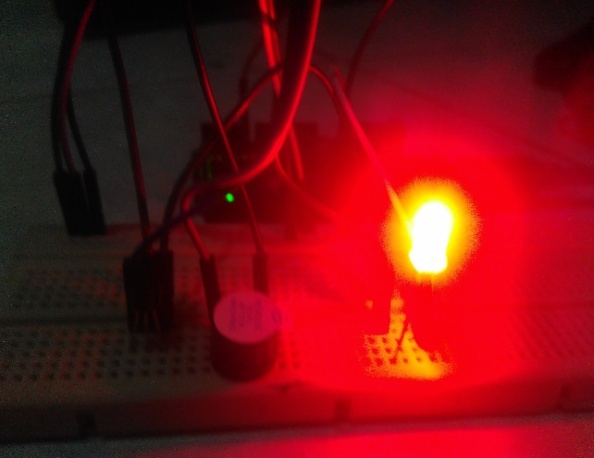
 **C.1.3 SCRIPT PROGRAM**



****

****

**C.1.4 HASIL**

****

**D .ANALISA**

Pada projek ini terdapat 3 task, yaitu task membaca LED, task membaca sensor suhu dan task yang ketiga adalah membaca indikator buzzer,.Program dimulai dari mendefinisikan task, pembuatan skala prioritas dan membuat program pada tiap tasknya. Program dijalnkan secara multitasking sistem berjalan sesuai dengan scheduling dan melaksanakan perintah di dalamnya.

Konsep pada program kali ini yaitu untuk pendeteksian kebakaran dengan menggunakan sensor LM35 yang di setpoint nilainya, sehingga apabila melebihi nilainya akan membuat LED dan Buzzer beroperasi sebagai pertanda terjadinya kebakaran.

1. **KESIMPULAN**
2. Prioritas berfungsi agar task berjalan sesuai dengan tingkat kepentingan dengan mendahulukan task sesuai dengan angkanya.
3. Penentuan tingkat prioritas tertinggi bisa dilakukan dengan melihat periode dari seberapa seringnya muncul task tersebut. Semakin sering task tersebut muncul maka semakin tinggi prioritas dai task tersebut
4. Program dapat berjalan namun belum sempurna sehingga perlu pengembangan dari algoritma pemrograman maupun penambahan komponen..
5. Sistem operasi RTOS dalam program dibagi dalam beberapa task yang dapat diatur urutannya sesuai dengan kebutuhan.

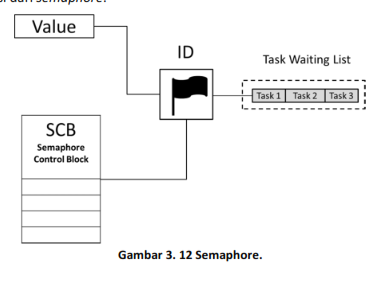
|  |  |  |
| --- | --- | --- |
| **KELOMPOK** | D | **ACC** |
| **Tanggal Praktikum** | 3 April 2018 | Ast1 |
|  | Ridho Surya Kusuma (D400150096) | Ast2 |
|  | Raka Siwi P (D400150107) | **Tanggal ACC :** |
|  | Rahmad Bahrudin A (D400150156) | **Revisi Tanggal :** |

**MODUL 2**

**SEMAPHORE DALAM FREERTOS**

1. **TUJUAN**
2. Praktikan dapat memahami dan mengetahui fungsi Semaphore dalam FreeRTOS.
3. Praktikan dapat membuat program dengan fungsi Semaphore dalam FreeRTOS.
4. Praktikan dapat membuat program dengan 3 task dengan menggunakan fungsi Semaphore dalam FreeRTOS.
5. **ALAT DAN BAHAN**
6. Laptop / PC yang telah diinstal software Arduino
7. Arduino Uno
8. **DASAR TEORI**

Semaphore ialah objek kernel dimana satu atau lebih task menggunakannya untuk sinkronisasi antar task. Setiap semaphore memiliki nomer ID unik, sebuah value binary atau count, serta tersambung pada waiting list dan Semaphore Control Block (SCB). Gambar 3. 12 menunjukkan ilustrasi dari semaphore.



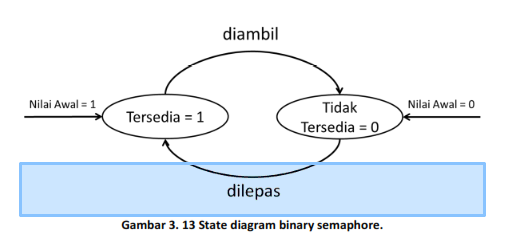
Semaphore digunakan sebagai tanda bahwa task yang mengakuisisinya memiliki hak untuk melaksanakan operasi tertentu juga mengakses resource yang ada. Akuisisi semaphore dibatasi oleh waktu dalam periode tertentu. Semaphore dapat diibaratkan sebagai kunci duplikat rumah yang dipegang oleh kepala keluarga. Anggota keluarga lain dapat meminjam kunci duplikat tersebut. Kunci duplikat tersedia dalam jumlah terbatas sehingga ketika sudah habis maka kepala keluarga tidak dapat memberikan lagi kuncinya. Pemakaian kunci juga terbatas, ketika sudah habis masa pakainya, maka anggota keluarga tidak bisa meminjam kuncinya sampai ada anggota lain mengembalikan kunci.

A. Jenis Semaphore

Terdapat beberapa jenis semaphore yang didukung oleh kernel RTOS seperti binary semaphore, counting semaphore, dan mutex semaphore.

1. Binary Semaphore

Digunakan pada mutual exclusion dan sinkronisasi. Sinkronisasi antar task atau antara task dan interrupt. Binary semaphore hanya memiliki dua nilai integer yaitu 1 atau 0. Binary semaphore bernilai 0 menunjukkan semaphore tidak tersedia/kosong. Sedangkan ketika semaphore bernilai 1 menunjukkan semaphore tersedia/penuh.



Gambar 3. 13 menunjukkan state diagram dari semaphore. Nilai awal ketika semaphore dibuat bisa berisi 1 atau 0. Pada state semaphore tersedia (nilai 1) lalu diambil oleh task maka state akan berubah menjadi tidak tersedia (nilai 0) begitupun sebaliknya. Semaphore pada state tidak tersedia lalu dilepas oleh suatu task maka statenya akan berubah menjadi tersedia.

2. Counting Semaphore

Counting semaphore merupakan semaphore yang menggunakan perhitungan suatu variablecounter untuk menentukan ketersediaan semaphore tersebut. Counting semaphore dapat digunakan oleh setiap taskyang membutuhkannya. Nilai maskimal dari counting dapat didefinisikan di saat semaphore dibuat atau dikenal dengan istilah bounded count. Nilaimaksimal counting juga dapat tidak didefinisikan atau disebut unboundedcount. Pada unboundedcount nilai maksimal tergantung pada nilai maksimalyang dapat ditampung sesuai tipe data yang digunakan misalnya unsigne dinteger atau unsigned long, dan lain sebagainya. Nilai nol (counter = 0) menunjukkan semaphore tidak tersedia. Sedangkannilai counter lebih dari nol (counter > 0) menunjukkan semaphore tersedia.Counting semaphore dapat diakuisisi/diambil ataupun dilepas beberapa kalitergantung nilai maksimal counter-nya.

3. Mutual Exclusion (Mutex) Semaphore

Mutex semaphore merupakan semaphore khusus yang memilikibeberapa fitur seperti penghapusan task, akses recursive, kepemilikan, sertaprotocol untuk menghindari masalah inheritance (pewarisan) pada mutualexclusion. Pada mutex terdapat dua state yaitu locked dan unlocked. State locked bernilai 1 dan unlocked bernilai 0. Saat diinisialisasi, mutex beradapada state unlocked. Pada kondisi ini mutex dapat diakuisisi oleh suatu task.Akuisisi ini mengakibatkan mutex berada pada locked state. Sebaliknyaketika task melepas mutex maka state mutex berubah menjadi unlocked.

B. Perintah pada Semaphore

Terdapat beberapa perintah yang dapat digunakan untuk melakukanbeberapa operasi terhadap semaphore. Perintah tersebut diantaranya:

B.1 Membuat dan menghapus semaphore

Proses pembuatan semaphore perlu memperhatikan support kernel yang digunakan terhadap jenis-jenis semaphore yang ada. Terdapathal-hal yang perlu dideklarasikan untuk menginisialisasi semaphoresesuai dengan masing-masing tipe semaphore yang ada seperti:

1. Binary: deklarasikan state awal semaphore dan urutan pada taks - waiting order

2. Counting: deklarasikan initial semaphore count dan task waiting order

3. Mutex: deklarasikan task-waiting order dan set aktif task deletion safety, recursion, serta priority inversion avoidance.

B2. Mengakusisi dan melepas semaphore

Operasi untuk mengakuisisi dan melepas semaphore dapat berbed-abedasesuai dengan kernel yang digunakan seperti, take dan give,sm\_p dan sm\_v, send dan post, ataupun lock dan unlock. Sebelumproses akusisi semaphore dilakukan, suatu task dapat memintanya

dengan salah satu cara di bawah ini:

1. Wait forever: task akan di blok hingga dapat mengakuisisi semaphore

2. Wait with a timeout: task akan di blok dalam jangka waktu tertentu. Ketika task belum mendapat giliran mengakuisisisemaphore, sedangkan waktu tunggunya sudah habis, maka taskakan dihapus dari daftar task waiting list kemudian diletakkanpada ready state atau running state. not wait: task tidak di blok ketika permintaannya terhadapsemaphore tidak bisa dipenuhi pada suatu waktu.

B.3 Membersihkan semaphore task-waiting list

Operasi ini dapat dilakukan dengan melakukan perintah flush.Perintah ini akan mengkosongkan semua task yang sedang beradatask waiting list.

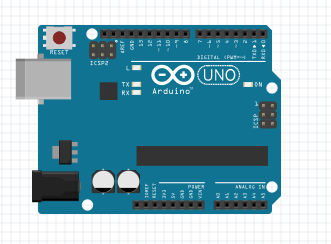
B.4 Mengambil informasi semaphore

Semaphore memiliki sejumlah informasi yang dapat digunakan untukdebugging. Terdapat dua perintah yang mungkin dilakukandiantaranya: show info untuk menampilkan informasi umummengenai semaphore dan show blocked list untuk menampilkandaftar ID dari task yang di blok karena menunggu akusisi semaphore.

**D. HASIL PRAKTIKUM**

**D.1. PERCOBAAN 1**

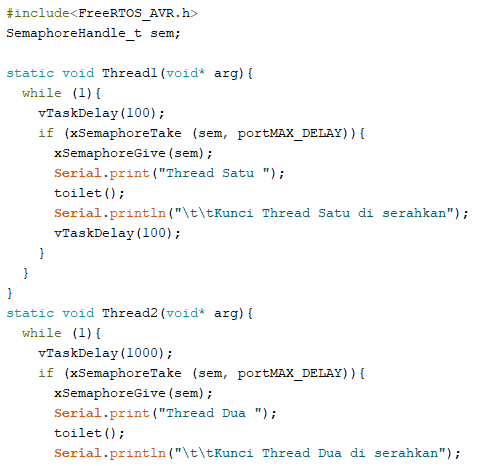
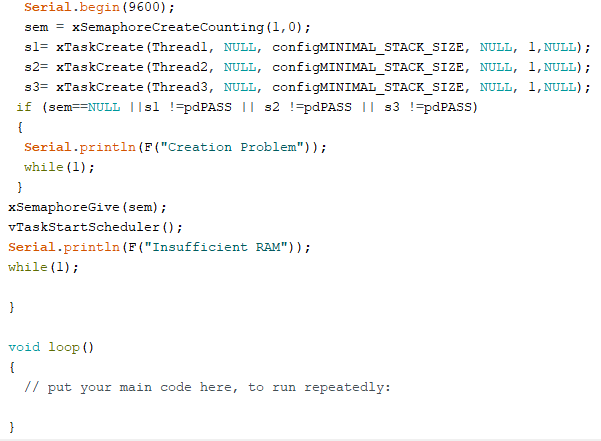
**D.1.1. SKEMA RANGKAIAN**



**D.1.2. FOTO RANGKAIAN**

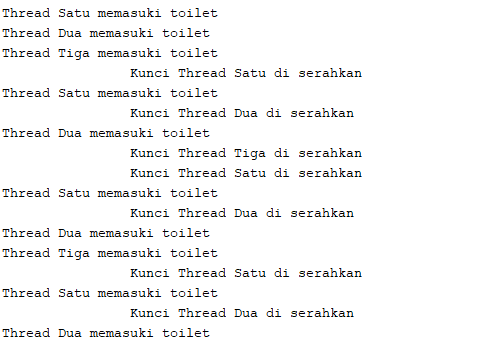


**D.1.3 SCRIPT.**

****

****

**D.1.4. FOTO HASIL**

****

**E. ANALISA**

E.1. Analisa Percobaan

Pada praktikum kali ini menggunakan fungsi perintah semaphore dalam mengakses sumber daya dan terdapat 3 perintah .SemaphoreHandle\_t xSerialSemaphore digunakan untuk mendeklarasikan semaphore handle mutex yang akan digunakan untuk mengatur port serial dan untuk memastikan hanya satu perintah saja yang akan diakses pada waktu tertentu. void TaskDigitalRead( void \*pvParameters ) adalah definisi perintah yang ingin dieksekusi dalam program ini. Serial.begin(9600) Menginisialisasi komunikasi serial berjalan 9600 bits per second:

Kinerja dari prinsip sebuah program diatas yaitu ketiga toilet dapat di masuk I secara bersamaan atau ketiga program dapat berjalan bersamaan . program tersebut akan masuk atau on apabila sudah diberi ijin setelah kunci diserahkan , di dalam program tersebut mempunya delay masing-masing dimana hanya satu toilet yang digunakan dan terkadang 2 toilet dan juga bias ketiga toilet dapat digunakan bersamaan dengan syarat kunci tadi sudah di serahkan.

if ( xSerialSemaphore == NULL ) pemeriksaan untuk konfirmasi bahwa semaphore serial tidak digunakan atau dalam keadaan bebas. xSerialSemaphore = xSemaphoreCreateMutex() membuat semaphore mutex yang akan digunakan untuk mengatur port serial if ( ( xSerialSemaphore ) != NULL ) xSemaphoreGive( ( xSerialSemaphore ) )membuat semaphore tersedia untuk diakses . xTaskCreate membuat set up perintah berjalan dengan prioritas yang diinginkan,semakin besar nilainya maka akan dieksekusi yang pertama.void TaskDigitalRead( void \*pvParameters ) merupakan fungsi untuk mengatur bagaimana Perintah tersebut akan berjalan . Program dibuat dapat berjalan secara terus menerus secara gantian pada masing- masing task nya. xSemaphoreGive digunakan untuk memasukkan data kemudian akan diambil oleh xSemaphoreTake dan kemudian ditampilkan pada Serial Monitor. Program berjalan secara terus menerus (forever loop) pada serial monitordengan penundaan pemunculan data di serial monitor. Pada setiap task program tidak menggunakan delay pada xSemaphoreGive dan XsemaphorTake karena agar dapat berjalan saling bergantian. Pada Serial.println digunakan untuk menapilkan data keserial monitor.

1. **KESIMPULAN**
2. Semaphore adalah suatu isyarat yang digunakan untuk menentukan akses untuk berbagi sumber daya sistem.
3. xSemaphoreGive dan xSemaphoreTake dalam praktikum ini saling berhubungan dalam jalannya program
4. Dari Serial Monitor terlihat hasil dari xSemaphoreGive dan xSemaphoreTake secara acak karena pengaruh vTaskDelay
5. Dari segi programming, penanganan masalah sinkronisasi dengan semaphore umumnya rapi dan teratur, sehingga mudah untuk dibuktikan kebenarannya

|  |  |  |
| --- | --- | --- |
| **KELOMPOK** | D | **ACC** |
| **Tanggal Praktikum** | 4 Mei 2018 |  |
|  | Ridho Surya Kusuma (D400150096) |  |
|  | Raka Siwi P (D400150107) | **Tanggal ACC :** |
|  | Rahmad Bahrudin A (D400150156) | **Revisi Tanggal :** |

**PROJECT MODUL 2**

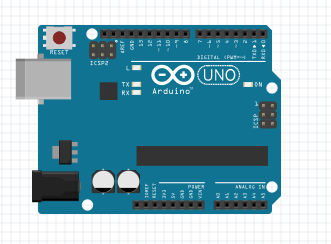
**SEMAPHORE DALAM FREERTOS**

1. **TUJUAN**
2. Praktikan dapat memahami dan mengetahui fungsi Semaphore dalam FreeRTOS.
3. Praktikan dapat membuat program dengan fungsi Semaphore dalam FreeRTOS.
4. Praktikan dapat membuat program dengan 3 task dengan menggunakan fungsi Semaphore dalam FreeRTOS.
5. **ALAT DAN BAHAN**
6. Laptop / PC yang telah diinstal software Arduino
7. Arduino Uno .

**C. HASIL PROJEK**

**C.1. PERCOBAAN 1**

**C.1.1. SKEMA RANGKAIAN**



**C.1.2. FOTO RANGKAIAN**



**C.1.3 SCRIPT PROGRAM**

#include <Arduino\_FreeRTOS.h>

#include <semphr.h> // add the FreeRTOS functions for Semaphores (or Flags).

//mendeklarasikan semahore handle mutex yang akan digunakan untuk mengatur port serial .

//untuk memastikan hanya satu perintah saja yang akan diakses pada waktu tertentu

SemaphoreHandle\_t xSerialSemaphore;

// define three Tasks for DigitalRead & AnalogRead/Mendefinisikan perintah

void TaskDigitalRead( void \*pvParameters );

void TaskAnalogRead( void \*pvParameters );

void TaskBlink(void \*pvParameters);

// Fungsi setup akan berjalan sekali ketika menekan reset atau power pada board

void setup() {

// Menginisialisasi komunikasi serial berjalan 9600 bits per second:

Serial.begin(9600);

if ( xSerialSemaphore == NULL ) // check untuk konfirmasi bahwa semaphore serial tidak digunakan

{

xSerialSemaphore = xSemaphoreCreateMutex(); //membuat semaphore mutex yang akan digunakan untuk mengatur port serial

if ( ( xSerialSemaphore ) != NULL )

xSemaphoreGive( ( xSerialSemaphore ) ); //membuat semaphore tersedia untuk digunakan

}

// membuat set up perintah berjalan

xTaskCreate(

TaskDigitalRead

, (const portCHAR \*)"DigitalRead" // A name just for humans

, 128 // This stack size can be checked & adjusted by reading the Stack Highwater

, NULL

, 2 // Priority, with 1 being the highest, and 4 being the lowest.

, NULL );

xTaskCreate(

TaskAnalogRead

, (const portCHAR \*) "AnalogRead"

, 128 // Stack size

, NULL

, 1 // Priority

, NULL );

xTaskCreate(

TaskBlink

, (const portCHAR \*) "BlinkLED"

, 128 // Stack size

, NULL

, 3 // Priority

, NULL );

}

void loop()

{

// Empty. Things are done in Tasks.

}

void TaskDigitalRead( void \*pvParameters ) // This is a Task.

{

uint8\_t pushButton = 2;

// make the pushbutton's pin an input:

pinMode(pushButton, INPUT);

for (;;)

{

int buttonState = digitalRead(pushButton);

if ( xSemaphoreTake( xSerialSemaphore, ( TickType\_t ) 5 ) == pdTRUE )

{

Serial.print("Tombol Emergency \n");

Serial.println(buttonState);

xSemaphoreGive( xSerialSemaphore ); // Now free or "Give" the Serial Port for others.

}

vTaskDelay(400); // one tick delay (15ms) in between reads for stability

}

}

void TaskAnalogRead( void \*pvParameters ) // This is a Task.

{

for (;;)

{

// read the input on analog pin 0:

int sensorValue = analogRead(A0);

if ( xSemaphoreTake( xSerialSemaphore, ( TickType\_t ) 5 ) == pdTRUE )

{

Serial.print("Sensor telah terbaca dengan baik \n");

Serial.println(sensorValue);

xSemaphoreGive( xSerialSemaphore ); // Now free or "Give" the Serial Port for others.

}

vTaskDelay(100); // one tick delay (15ms) in between reads for stability

}

}

void TaskBlink( void \*pvParameters ) // This is a Task.

{

uint8\_t LEDku = 4;

// make the pushbutton's pin an input:

pinMode(LEDku, INPUT);

for (;;) // A Task shall never return or exit.

{

// read the input pin:

int LEDState = digitalRead(LEDku);

if ( xSemaphoreTake( xSerialSemaphore, ( TickType\_t ) 5 ) == pdTRUE )

{

Serial.print("Lampu Nyala\n");

Serial.println(LEDku);

xSemaphoreGive( xSerialSemaphore ); // Now free or "Give" the Serial Port for others.

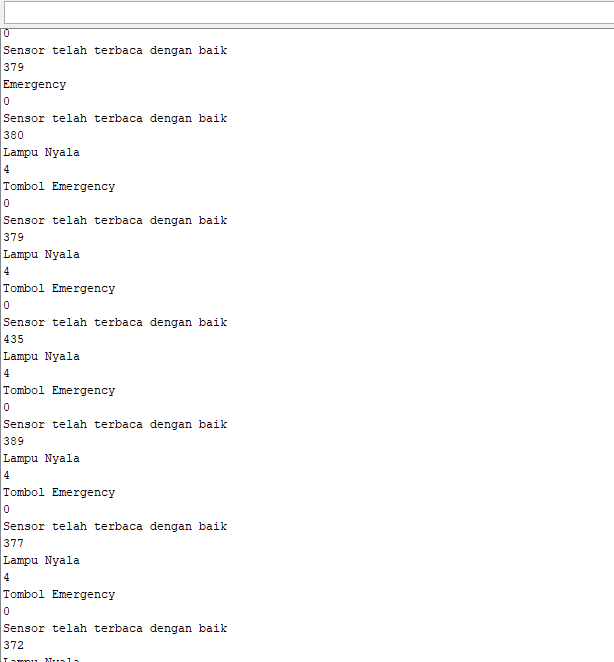
}

vTaskDelay(200);

}

}

**C.1.4 HASIL PADA SERIAL MONITOR**



**D.ANALISA**

Pada projek kali ini menggunakan beberapa fungsi perintah semaphore dalam mengakses sumber daya dan terdapat perintah. .Projek diatas mempunyai konsep sistem keamanan menggunakan kombinasi berupa pembacaan data dari sensor, indikator lampu dari LED dan bunyi dari Buzzer. Untuk yang dijadikan sebagai prioritas yakni pembacaaan sensor suhu terlebih dahulu setelah itu eksekusi perintah blink LED dan Buzzer .

SemaphoreHandle\_t xSerialSemaphore digunakan untuk mendeklarasikan semaphore handle mutex yang akan digunakan untuk mengatur port serial dan untuk memastikan hanya satu perintah saja yang akan diakses pada waktu tertentu. void TaskDigitalRead( void \*pvParameters ) adalah definisi perintah yang ingin dieksekusi dalam program ini. Serial.begin(9600) Menginisialisasi komunikasi serial berjalan 9600 bits per second:

if ( xSerialSemaphore == NULL ) pemeriksaan untuk konfirmasi bahwa semaphore serial tidak digunakan atau dalam keadaan bebas. xSerialSemaphore = xSemaphoreCreateMutex() membuat semaphore mutex yang akan digunakan untuk mengatur port serial if ( ( xSerialSemaphore ) != NULL ) xSemaphoreGive( ( xSerialSemaphore ) )membuat semaphore tersedia untuk diakses . xTaskCreate membuat set up perintah berjalan dengan prioritas yang diinginkan,semakin besar nilainya maka akan dieksekusi yang pertama.void TaskDigitalRead( void \*pvParameters ) merupakan fungsi untuk mengatur bagaimana Perintah tersebut akan berjalan . Program dibuat dapat berjalan secara terus menerus secara gantian pada masing- masing task nya. xSemaphoreGive digunakan untuk memasukkan data kemudian akan diambil oleh xSemaphoreTake dan kemudian ditampilkan pada Serial Monitor. Program berjalan secara terus menerus (forever loop) pada serial monitor dengan penundaan pemunculan data di serial monitor. Pada setiap task program tidak menggunakan delay pada xSemaphoreGive dan XsemaphorTake karena agar dapat berjalan saling bergantian. Pada Serial.println digunakan untuk menapilkan data keserial monitor.

**E.KESIMPULAN**

1. Semaphore adalah suatu karnal yang digunakan untuk sinkronisasi perintah sehingga perintah dapat berjalan secara bergantian dan dapat mengakses sumber daya secara optimal
2. Dari Serial Monitor terlihat hasil dari xSemaphoreGive dan xSemaphoreTake sesuai delay dan XTaskCreate yang berkaitan dengan prioritas
3. Projek dapat berjalan sesuai dengan rancangan namun perlu diaplikasikan dalam bentuk hardware

|  |  |  |
| --- | --- | --- |
| **KELOMPOK** | 4 | **ACC** |
| **Tanggal Praktikum** | 2 Juni 2018 | Ast1 |
|  | Ridho Surya Kusuma (D400150096) | Ast2 |
|  | Raka Siwi Pamungkas (D400150107) | **Tanggal ACC :** |
|  | Rahmad Bahrudin Afrianto (D400150156) | **Revisi Tanggal :** |

**MODUL 3**

**Flexible and Absolute Timing On**

1. **Tujuan**
2. Praktikan dapat memahami konsep flexible and absolute timing on pada FreeRTOS
3. Praktikan dapat membuat program sederhana yang menyertakan konsep atau fungsi flexible and absolute timing on
4. Praktikan dapat mengembangkan program yang sudah dipahami.
5. **Alat dan Bahan**
6. Laptop/PC yang telah diinstal arduino IDE
7. Kit Arduino UNO
8. **Dasar Teori**

Dalam FreeRTOS fungsi pewaktuan setiap task (timing / delay) terdapat sedikitnya ada 2 jenis, yaitu absolute timing dan flexible timing.

1. **Absolut timing ( vTaskDelayUntil() )**

Tunda task hingga waktu yang ditentukan. Fungsi ini dapat digunakan oleh tugas-tugas periodic untuk memastikan frekuensi eksekusi konstan. Fungsi ini berbeda dari vTaskDelay() dalam satu aspek penting: vTaskDelay() menentukan waktu di mana tugas ingin membuka blokir relative terhadap waktu di mana vTaskDelay() dipanggil, sedangkan vTaskDelayUntil() menentukan waktu absolut di mana tugas ingin untuk membuka blokir atau mengaktifkan task selanjutnya. vTaskDelay() akan menyebabkan delay aktif dari task selama sejumlah ticks yang ditentukan dari waktu vTaskDelay() dipanggil. Oleh karena itu sulit untuk menggunakan vTaskDelay() dengan sendirinya untuk menghasilkan frekuensi eksekusi tetap sebagai waktu antara task membuka blokir setelah panggilan kevTaskDelay() dan task berikutnya.

Sedangkan vTaskDelay() menentukan waktu bangun relative terhadap waktu saat fungsi dipanggil, vTaskDelayUntil() menentukan waktu absolut (tepat) di mana ia ingin membuka blokir atau mengkatifkan task selanjutnya.

Perlu dicatat bahwa vTaskDelayUntil() akan segera kembali (tanpa pemblokiran) jikadigunakan untuk menentukan waktu bangun yang sudah di masa lalu. Oleh karena itu, task menggunakan vTaskDelayUntil() untuk melakukan secara berkala harus menghitung ulang waktu bangun yang diperlukan jika eksekusi periodic dihentikan karena alasan apa pun (misalnya, tugas untuk sementara ditempatkan ke status ditangguhkan) yang menyebabkan task untuk melewatkan satu atau eksekusi lebih berkala. Ini dapat dideteksi dengan memeriksa variabel yang dilewatkan oleh referensi sebagai parameter pxPreviousWakeTime terhadap hitungan tick saat ini. Namun ini tidak diperlukan di sebagian besar scenario penggunaan.

PortTICK\_PERIOD\_MS konstan dapat digunakan untuk menghitung waktu actual dari tingkat tick – dengan resolusi satu periode tick. Fungsi ini tidak boleh dipanggil ketika scheduler RTOS telah ditangguhkan oleh panggilan kevTask Suspend All ().

Parameters:

pxPreviusWakeTime: Penunjuk ke variabel yang menyimpan waktu saat task terakhir diblokir. Variabel harus diinisialisasi dengan waktu saat ini sebelum penggunaan pertama (lihatcontoh di bawah). Setelah ini, variable secara otomatis diperbarui dalam vTaskDelayUntil ().

xTimeIncrement : Periode waktu siklus. Task akan dibebaskan pada waktunya (\* pxPreviousWakeTime+xTimeIncrement). Memanggil vTask Delay Until dengan nilai parameter xTime Increment yang sama akan menyebabkan task dijalankan dengan periode interval tetap.

1. **Flexible timing ( vTaskDelay() )**

vtaskDelay() adalah delay yang berdasarkan jumlah ticks, sehingga lama waktu actual suatu task di blok atau ditahan bergantung pada tick rate. Constant port TICK\_PERIOD\_MS dapat digunakan untuk mengkalkulasi waktu actual dari tingkat tick – dengan resolusi satu periode tick.

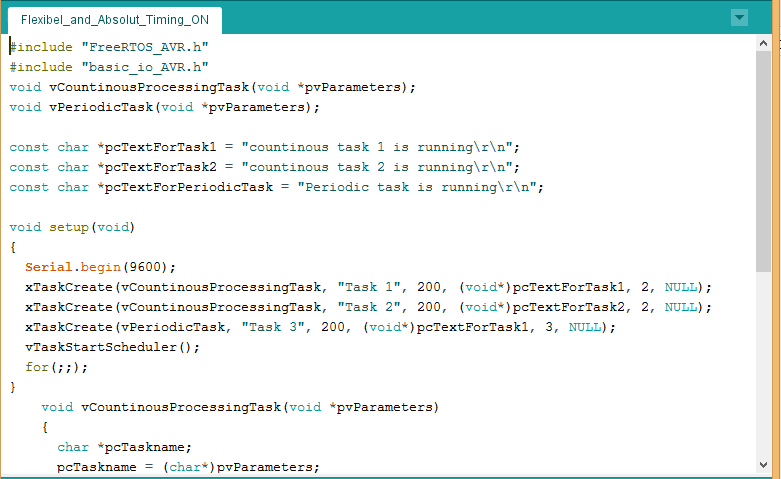
vTaskDelay() menentukan waktu dimana task ingin mulai dijalankan relative terhadap waktu dimana vTaskDelay() dipanggil. Misalnya, menentukan jangka waktu delay 100 tick akan menyebabkan task mulai aktif dijalankan 100 tick setelah vTaskDelay() dipanggil. Oleh karena itu, vTaskDelay() tidak menyediakan metode yang baik untuk mengendalikan frekuensi tugas periodic karena jalur yang diambil melalui kode, serta task dan interupsi lainnya akan mempengaruhi frekuensi dimana vTaskDelay() dipanggil dan waktu dimana task selanjutnya aktif. vTaskDelay() sesuai untuk digunakan pada task yang flexible.

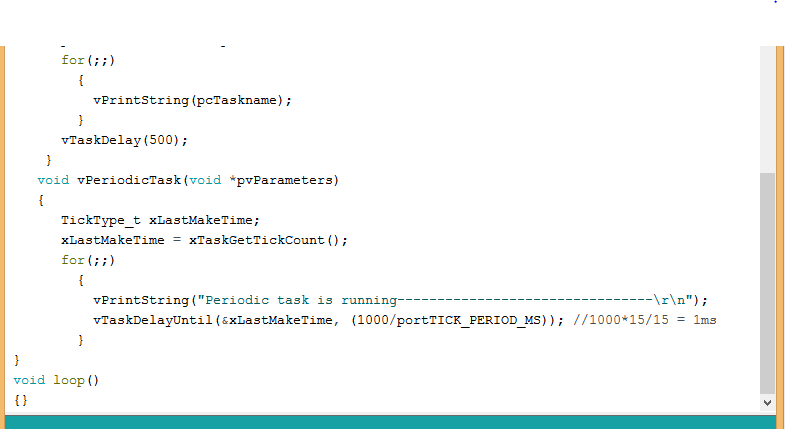
Parameters:

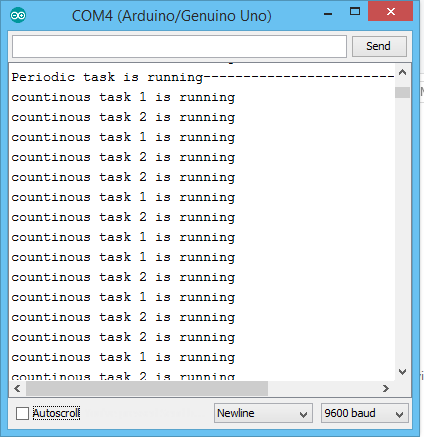
xTicksToDelay : Jumlah waktu dalam tick periode

1. **Hasil Praktikum**

D.1 Script program





 D.2 Serial Monitor

1. **Analisa**

Pada percobaan ini membuat program dengan 3 task, dimana task 1-2 aktif secara bergantian dengan waktu yang hampir sama dan berjalan secara terus menerus karena menggunakan vCountinousProcessingTask. Pada setiap task 1-2 memiiki fungsi tersendiri. Sedangkan untuk task 3 dengan prioritas tertinggi dan berfungsi menampilkan Periodic task is running dan akan aktif setiap 1 detik dan memblok task 1-2 serta bisa diubah periode waktunya tergantung kebutuhan, hal ini dapat terjadi karena task 3 menggunakan fungsi vPeriodicTask.

1. **Kesimpulan**
2. vTaskDelay() menentukan waktu di mana tugas ingin membuka blokir relative terhadap waktu di manav TaskDelay() dipanggil.
3. vTaskDelayUntil() menentukan waktu absolut di mana tugas ingin untuk membuka blokir atau mengaktifkan task selanjutnya.

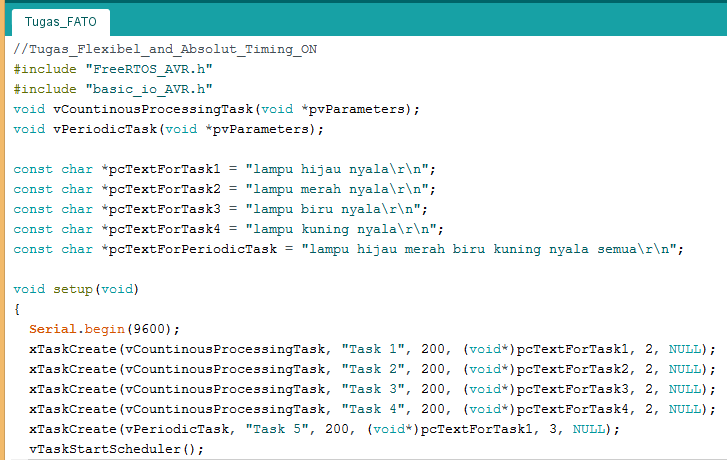
|  |  |  |
| --- | --- | --- |
| **KELOMPOK** | 4 | **ACC** |
| **Tanggal Praktikum** | 2 Juni 2018 | Ast1 |
|  | Ridho Surya Kusuma (D400150096) | Ast2 |
|  | Raka Siwi Pamungkas (D400150107) | **Tanggal ACC :** |
|  | Rahmad Bahrudin Afrianto (D400150156) | **Revisi Tanggal :** |

**PROJECT MODUL 3**

**Flexible and Absolute Timing On Nyala Lampu**

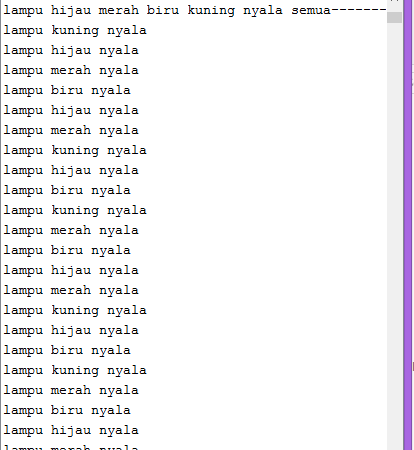
1. **Tujuan**
2. Praktikan dapat memahami konsep flexible and absolute timing on pada FreeRTOS
3. Praktikan dapat membuat program sederhana yang menyertakan konsep atau fungsi flexible and absolute timing on
4. Praktikan dapat mengembangkan program yang sudah dipahami.
5. **Alat dan Bahan**
6. Laptop/PC yang telah diinstal arduino IDE
7. Kit Arduino UNO
8. **Hasil Praktikum**

C.1 Script program





C.2 Serial Monitor



1. **Analisa**

Pada project ini kami membuat program dengan 5 task, dimana task 1-4 aktif secara bergantian dengan waktu yang hampir sama dan berjalan secara terus menerus karena menggunakan vCountinousProcessingTask. Pada setiap task 1-4 memiiki fungsi tersendiri, yakni menghidupkan lampu berwarna hijau merah biru kuning secara acak dan bergantian. Sedangkan untuk task 5 dengan prioritas tertinggi dan berfungsi menghidupkan semua lampu akan aktif setiap 1 detik dan memblok task 1-4 serta bisa diubah periode waktunya tergantung kebutuhan, hal ini dapat terjadi karena task 5 menggunakan fungsi vPeriodicTask.

1. **Kesimpulan**
2. vTaskDelay() menentukan waktu di mana tugas ingin membuka blokir relative terhadap waktu di manav TaskDelay() dipanggil.
3. vTaskDelayUntil() menentukan waktu absolut di mana tugas ingin untuk membuka blokir atau mengaktifkan task selanjutnya.

|  |  |  |
| --- | --- | --- |
| **KELOMPOK** | 4 | **ACC** |
| **Tanggal Praktikum** | 4 Mei 2018 | Ast1 |
|  | Ridho Surya Kusuma (D400150096) | Ast2 |
|  | Raka Siwi Pamungkas (D400150107) | **Tanggal ACC :** |
|  | Rahmad Bahrudin A. (D400150156) | **Revisi Tanggal :** |

**MODUL 4**

**MANAGEMENT RESOURCE DALAM FREERTOS**

1. **Tujuan**
2. Praktikan dapat mengetahui tentang Management Resource pada FreeRTOS.
3. Praktikan dapat mengetahui fungsi Management Resource.
4. Praktikan dapat membuat dan mensimulasikan program menggunakan fungsi Management Resource pada FreeRTOS.
5. Praktikan dapat mengembangkan program yang sudah dipahami.
6. **Alat Dan Bahan**
7. Laptop / PC yang telah diinstal software Arduino
8. Kit Arduino Uno
9. **Dasar Teori**

Dalam penggunaan ini, nilai hitungan menunjukkan jumlah sumber daya yang tersedia. Untuk mendapatkan kontrol sumber daya, tugas harus terlebih dahulu mendapatkan semaphore - mengurangi nilai hitungan semaphore. Ketika nilai hitungan mencapai nol tidak ada sumber daya gratis. Ketika tugas selesai dengan sumber daya itu 'memberi' semaphore kembali - incrementing nilai hitungan semaphore. Dalam hal ini diharapkan agar nilai hitung awal sama dengan nilai hitung maksimum, yang menunjukkan bahwa semua sumber daya gratis.

**Perencanaan Sumber Daya**

Sumber daya terbatas, jadi harus merencanakan penggunaannya. Hanya dapat menempatkan setiap orang di satu tempat, pada satu waktu, mengerjakan satu tugas, sesuai dengan keahliannya yang unik. Perencanaan sumber daya yang efektif membantu memberikan proyek tepat waktu tanpa mengubah tenggat waktu, mengubah prioritas, dan evolusi proyek umum. Perencanaan sumber daya menjaga tingkat pemanfaatan pada tingkat yang menguntungkan, mengurangi waktu yang tidak dapat ditagih yang dihabiskan oleh sumber daya untuk proyek.

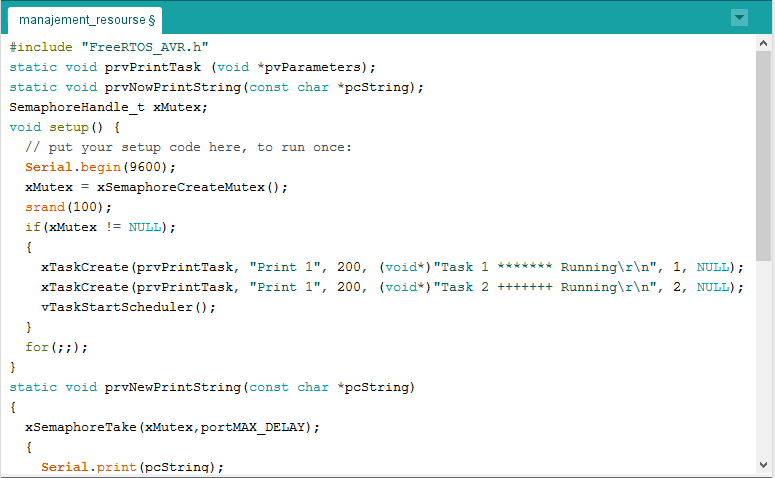
**Keuntungan Menggunakan Mnaagement Resource**

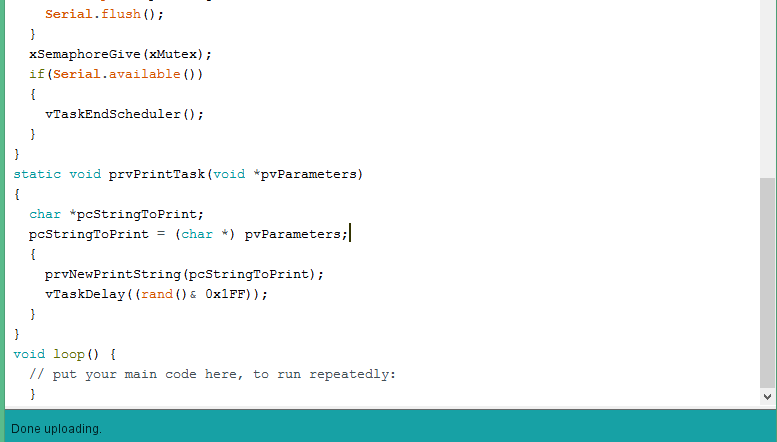
Perangkat lunak perencanaan sumber daya memecahkan tantangan khusus untuk mengetahui sesuatu ditempatkan di tempat yang tepat untuk memenuhi kebutuhan klien dan mencapai tujuan bisnis. Ini membantu mengukur pemanfaatan, sehingga apakah cukup, terlalu banyak, atau terlalu sedikit. Dapat memastikan bahwa mereka menyelesaikan pekerjaan itu tepat waktu, dan dapat mengalokasikannya kembali ketika jadwal proyek bergeser. Selain itu, perangkat lunak perencanaan sumber daya membantu memperkirakan skala. Ketika sebuah proyek berlipat ganda dalam ukuran atau memenangkan klien baru, perangkat lunak perencanaan sumber daya menunjukkan di mana dapat menarik sumber daya untuk mencapai tugas-tugas yang diperlukan. Beberapa aplikasi perencanaan sumber daya menggabungkan informasi keuangan atau berintegrasi dengan solusi CRM dan akuntansi. Ini memungkinka melihat manfaat-biaya dari pengalokasian ulang sumber daya untuk tugas-tugas lain, serta kesehatan keuangan keseluruhan. Akhirnya, perangkat lunak perencanaan sumber daya menawarkan ruang kerja pusat berkolaborasi.

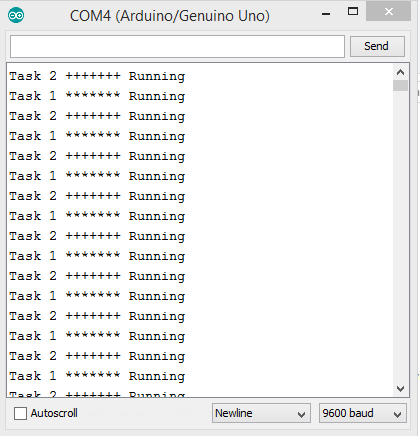
**Kerugian Menggunakan Management Resource**

Salah satu hambatan terbesar untuk mengadopsi perangkat lunak perencanaan sumber daya khusus adalah integrasi mahal yang diperlukan untuk mendapatkan manfaat yang diantisipasi. Dapat mengurangi resiko ini dengan memilih perangkat lunak perencanaan sumber daya yang merupakan bagian dari rangkaian aplikasi, dan / atau yang terintegrasi dengan perangkat lunak manajemen akuntansi dan proyek pilihan di luar rak. Menggunakan perangkat lunak perencanaan sumber daya daripada spreadsheet juga akan membutuhkan pelatihan (dan kemungkinan rekayasa ulang proses bisnis) untuk berhasil. Pengalaman pengguna bervariasi secara drastis di antara solusi yang berbeda, dan sangat penting untuk memilih satu dengan antarmuka pengguna yang akan digunakan. Karena alasan ini, penting untuk memilih penyedia perangkat lunak yang menunjukkan antarmuka pengguna dan penawaran layanan pelanggan yang memadai yang memenuhi standar.

1. **HASIL PRAKTIKUM**

D.1 Script



D.2 Serial Monitor

1. **ANALISA**

Pada percobaan ini membuat program dengan fungsi management resource yang diterapkan pada 2 task. Program dapat berjalan secara terus menerus secara bergantian dan urut sesuai prioritas. Pada program ini menggunakan xSemaphoreGive untuk memasukkan data kemudian akan diambil oleh xSemahporeTake kemudian baru bisa di tampilkan pada Serial Monitor. Serta untuk mengendalikan management resource kami menggunakan semaphoreHandle\_t xMutex agar program berjalan secara continue.

1. **KESIMPULAN**
2. xSemaphoreGive digunakan untuk memasukkan data dan xSemaphoreTake digunakan untuk mengambil data setelah itu baru bisa ditampilkan.
3. Hasil dari serial monitor terlihat secara teratur karena menggunkan management resource pada SempahoreHandle\_t xMutex.

|  |  |  |
| --- | --- | --- |
| **KELOMPOK** | 4 | **ACC** |
| **Tanggal Praktikum** | 4 Mei 2018 | Ast1 |
|  | Ridho Surya Kusuma (D400150096) | Ast2 |
|  | Raka Siwi Pamungkas (D400150107) | **Tanggal ACC :** |
|  | Rahmad Bahrudin A. (D400150156) | **Revisi Tanggal :** |

**PROJEC MODUL 4**

**MANAGEMENT RESOURCE DALAM FREERTOS 3 TASK**

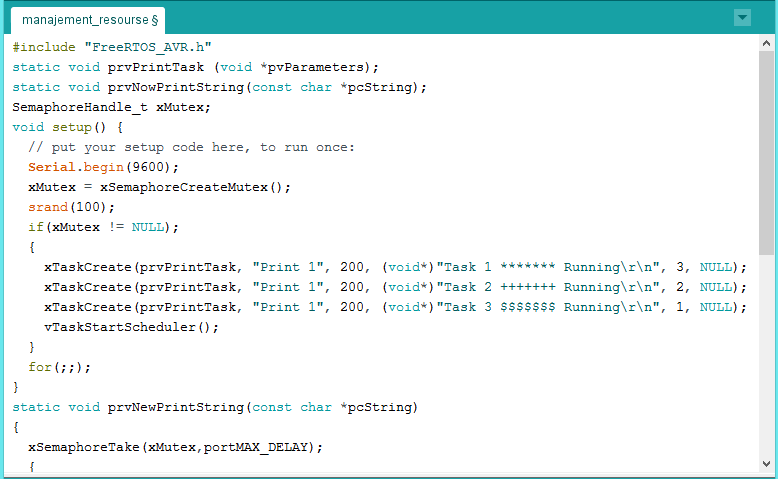
1. **Tujuan**
2. Praktikan dapat mengetahui tentang Management Resource pada FreeRTOS..
3. Praktikan dapat mengetahui fungsi Management Resource.
4. Praktikan dapat membuat dan mensimulasikan program menggunakan fungsi Management Resource pada FreeRTOS.
5. Praktikan dapat mengembangkan program yang sudah dipahami.
6. **Alat Dan Bahan**

1. Laptop / PC yang telah diinstal software Arduino

2. Kit Arduino Uno

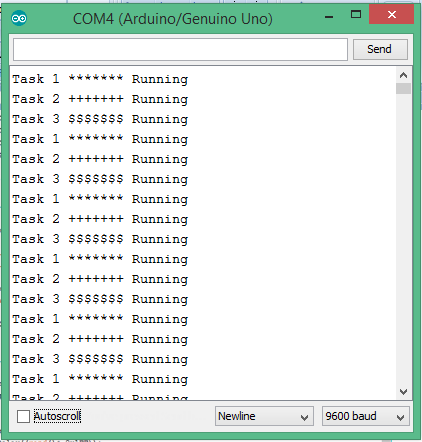
1. **Hasil Praktikum**

C.1 Script





C.2 Serial Monitor



1. **ANALISA**

Pada projek ini kami membuat program dengan fungsi management resource yang diterapkan pada 3 task. Program dapat berjalan secara terus menerus secara bergantian dan urut sesuai prioritas. Pada program ini menggunakan xSemaphoreGive untuk memasukkan data kemudian akan diambil oleh xSemahporeTake kemudian baru bisa di tampilkan pada Serial Monitor. Serta untuk mengendalikan management resource kami menggunakan semaphoreHandle\_t xMutex agar program berjalan secara continue.

1. **KESIMPULAN**
2. xSemaphoreGive digunakan untuk memasukkan data dan xSemaphoreTake digunakan untuk mengambil data setelah itu baru bisa ditampilkan.
3. Hasil dari serial monitor terlihat secara teratur karena menggunkan management resource pada SempahoreHandle\_t xMutex.