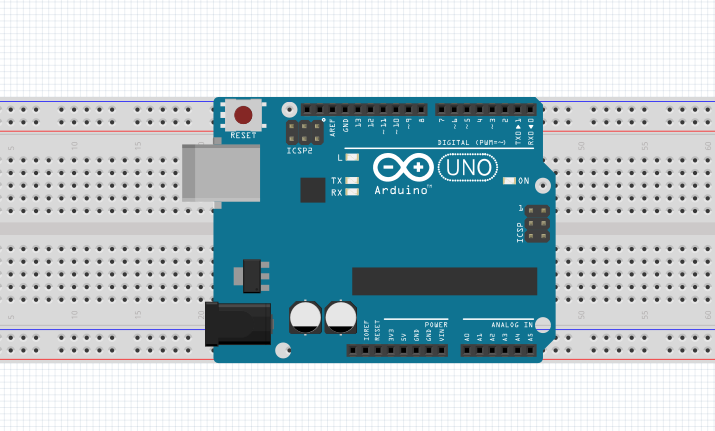
|  |  |  |  |
| --- | --- | --- | --- |
| **KELOMPOK** | C | **ACC** | |
| **Tanggal Praktikum** | 4 Mei 2018 |  |  |
|  | Nugroho Nanda S (D400150082) |
|  | Kharisma Aji S (D400150095) |
|  | Ahmad Sujarwo (D400150142) | **Tanggal ACC :** | |
|  | Puji Hanung W (D400150115) | **Revisi Tanggal :** | |

**MODUL 4**

Management Resource dalam FreeRTOS (Project)

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

**C 1.1Skema Ragkaian**

D.1 Hasil Percobaan Script

#include "FreeRTOS\_AVR.h"

static void prvPrintTask (void \*pvParameters);

static void prvNewPrintString(const char \*pcString);

SemaphoreHandle\_t xMutex;

void setup() {

Serial.begin(9600);

xMutex = xSemaphoreCreateMutex();

srand(100);

if(xMutex !=NULL);

{

xTaskCreate(prvPrintTask, "Print 1", 200, (void\*)"Task 1 Running \r\n",1,NULL);

xTaskCreate(prvPrintTask, "Print 1", 200, (void\*)"Task 2 Running \r\n",2,NULL);

xTaskCreate(prvPrintTask, "Print 1", 200, (void\*)"Task 3 Running \r\n",3,NULL);

vTaskStartScheduler();

}

for(;;);

}

static void prvNewPrintString(const char \*pcString)

{

xSemaphoreTake(xMutex,portMAX\_DELAY);

{

Serial.print(pcString);

Serial.flush();

}

xSemaphoreGive(xMutex);

if(Serial.available())

{

vTaskEndScheduler();

}

}

static void prvPrintTask(void \*pvParameters)

{

char \*pcStringToPrint;

pcStringToPrint = (char \*) pvParameters;

{

prvNewPrintString (pcStringToPrint);

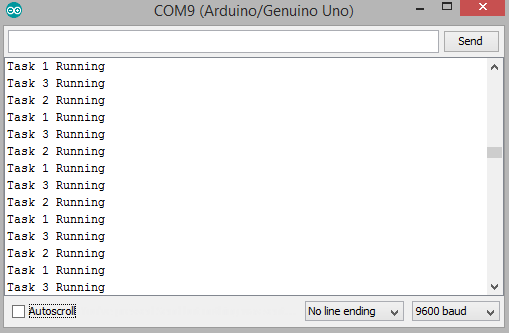
vTaskDelay((rand()& 0x1FF));

}}

void loop (){

}

D.2 Serial Monitor



1. **ANALISA**

Analisa Percobaan

Pada praktikum ini menggunakan fungsi management resource task pada program. 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. Sehingga program berjalan secara terus menerus (forever loop) pada serial monitor.

vTaskDelay digunakan untuk menampilkan data 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 sesuai urutan yang ditentukan sesuai prioritas. Sedangkan untuk fungsi dari xTaskCreate(prvPrintTask, "Print 1", 200, (void\*)"Task 1 Running \r\n",1,NULL); digunakan untuk meng*create* Task dan menentukan nomor prioritas untuk pemunculan pada serial monitor.

SemaphoreHandle\_t xMutex digunakan untuk mengendalikan management resource yang kita buat. Sehingga program berjalan secara terus menerus (forever loop) pada serial monitor secara bergantian tanpa ada yang keluara yang double.

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 teratur karena adanda pengaturan management resource pada SempahoreHandle\_t xMutex.