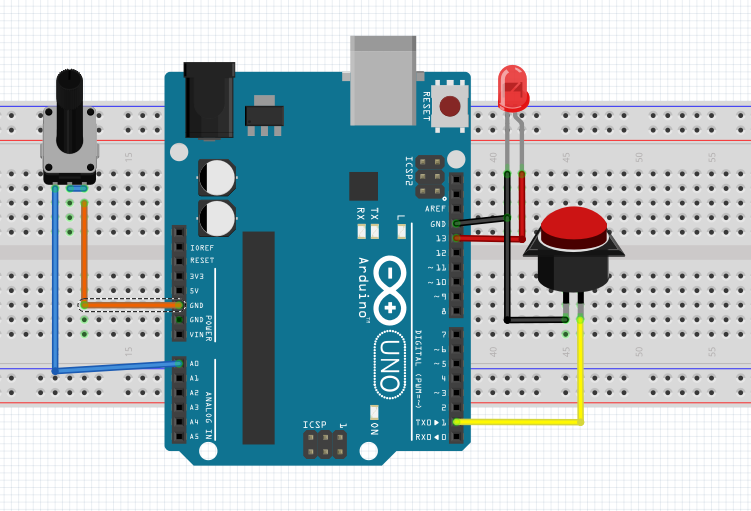
| **KELOMPOK** | C | **ACC** | |
| --- | --- | --- | --- |
| **Tanggal Praktikum** | 27 Maret 2018 |  |  |
|  | Nugroho Nanda Styanto |
|  | Kharisma Aji Satria Tama | **Tanggal ACC :** | |
|  | Puji Hanung Winarko | **Revisi Tanggal :** | |
|  | Ahmad Sujarwo |  | |

**MODUL 1**

3 TASK (ANALOGREAD,BLINK,DIGITALREAD)

1. **TUJUAN**
2. Praktikan dapat memahami salah satu sistem operasi yang ada pada Arduino yaitu FreeRTOS.
3. Praktikan dapat memahami program Blink & Analog Digital read menggunakan sistem operasi FreeRTOS.
4. Praktikan dapat membuat basic program dengan menggunakan sistem oprasi FreeRTOS.
5. **ALAT DAN BAHAN**
6. Laptop / PC yang telah diinstal software Arduino
7. Arduino Uno
8. LED
9. Potensio
10. Pushbutton
11. **HASIL PRAKTIKUM**

C.1 Hasil Percobaan 1

 C.1.1 Skema Rangkaian

C.1.2 Script Program

#include <Arduino\_FreeRTOS.h>

#include <semphr.h>

SemaphoreHandle\_t xSerialSemaphore;

// define tasks for Blink & Analog & DigitalRead

void TaskBlink( void \*pvParameters );

void TaskAnalogRead( void \*pvParameters );

void TaskDigitalRead( void \*pvParameters );

// the setup function runs once when you press reset or power the board

void setup() {

// initialize serial communication at 9600 bits per second:

Serial.begin(9600);

while (!Serial) {

; // wait for serial port to connect. Needed for native USB, on LEONARDO, MICRO, YUN, and other 32u4 based boards.

}

// Semaphores are useful to stop a Task proceeding, where it should be paused to wait,

// because it is sharing a resource, such as the Serial port.

// Semaphores should only be used whilst the scheduler is running, but we can set it up here.

if ( xSerialSemaphore == NULL ) // Check to confirm that the Serial Semaphore has not already been created.

{

xSerialSemaphore = xSemaphoreCreateMutex(); // Create a mutex semaphore we will use to manage the Serial Port

if ( ( xSerialSemaphore ) != NULL )

xSemaphoreGive( ( xSerialSemaphore ) ); // Make the Serial Port available for use, by "Giving" the Semaphore.

}

// Now set up two tasks to run independently.

xTaskCreate(

TaskBlink

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

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

, NULL

, 3 // Priority, with 3 (configMAX\_PRIORITIES - 1) being the highest, and 0 being the lowest.

, NULL );

xTaskCreate(

TaskAnalogRead

, (const portCHAR \*) "AnalogRead"

, 128 // Stack size

, NULL

, 2 // Priority

, NULL );

xTaskCreate(

TaskDigitalRead

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

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

, NULL

, 1 // Priority, with 3 (configMAX\_PRIORITIES - 1) being the highest, and 0 being the lowest.

, NULL );

// Now the task scheduler, which takes over control of scheduling individual tasks, is automatically started.

}

void loop()

{

// Empty. Things are done in Tasks.

}

/\*--------------------------------------------------\*/

/\*---------------------- Tasks ---------------------\*/

/\*--------------------------------------------------\*/

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

{

// digital pin 2 has a pushbutton attached to it. Give it a name:

uint8\_t pushButton = 2;

// make the pushbutton's pin an input:

pinMode(pushButton, INPUT);

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

{

// read the input pin:

int buttonState = digitalRead(pushButton);

// See if we can obtain or "Take" the Serial Semaphore.

// If the semaphore is not available, wait 5 ticks of the Scheduler to see if it becomes free.

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

{

// We were able to obtain or "Take" the semaphore and can now access the shared resource.

// We want to have the Serial Port for us alone, as it takes some time to print,

// so we don't want it getting stolen during the middle of a conversion.

// print out the state of the button:

Serial.println(buttonState);

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.

{

(void) pvParameters;

// initialize digital LED\_BUILTIN on pin 13 as an output.

pinMode(LED\_BUILTIN, OUTPUT);

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

{

digitalWrite(LED\_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)

vTaskDelay( 1000 / portTICK\_PERIOD\_MS ); // wait for one second, 1000/10ms per ticknya

digitalWrite(LED\_BUILTIN, LOW); // turn the LED off by making the voltage LOW

vTaskDelay( 1000 / portTICK\_PERIOD\_MS ); // wait for one second

}

}

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

{

(void) pvParameters;

for (;;)

{

// read the input on analog pin 0:

int sensorValue = analogRead(A0);

// print out the value you read:

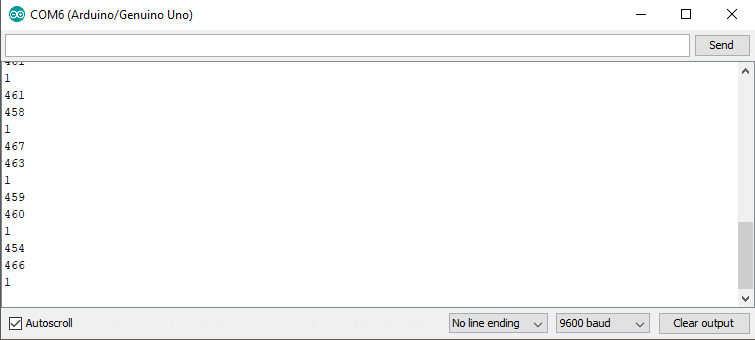
Serial.println(sensorValue);

vTaskDelay(50); // one tick delay (15ms) in between reads for stability

}

}

C 1.3 Serial Monitor



1. **ANALISA**

Analisa Projek

Pada percobaan projek ini meggunakan 3 task Blink Analog Digital read dimana pada projek tersebut menggunakan komponen Led,Pushbutton dan Potensio. Data yang masuk dari potensio iniakan menggerakanaktuator motor servo,Led sebagai indicator

1. **KESIMPULAN**
2. FreeRTOS (Free Real-time operating systems) adalah sebuah *real time operating system* pada Arduino yang banyak digunakan oleh mikrokontroler untuk kebutuhan sistem operasinya.
3. Pada projek kali ini terkadang terdapat masalah saat mengguunakan 2 aktuator motor