EXPERIMENT NO:

ROLL No:

NAME:

TITLE: Porting of FreeRTOS to Arduino/STM32.

Theory:

Real time operating system popularly known as RTOS provides controller with the ability to respond to input and complete tasks within a specific period of time based on priority. On the first look, an RTOS might sound like just any other embedded program or firmware, but it is built on the architecture of an Operating system. Hence, like any operating system RTOS can allow multiple programs to execute at the same time supporting **multiplexing**. As we know the core of a processor or controller can only execute a single instruction at a time, but the RTOS has something called the **scheduler** which decides which instruction to execute first and thus executes the instructions of multiple programs one after the other. Technically an RTOS only creates an illusion of multi-taking by executing paralleled instructions one at a time.

Code:

Arduino code of two task analogeread and blink task:

```
#include <Arduino_FreeRTOS.h>

// define two tasks for Blink & AnalogRead
void TaskBlink( void *pvParameters );
void TaskAnalogRead( 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.
  }

  // Now set up two tasks to run independently.
  xTaskCreate(
    TaskBlink
    ,  "Blink"   // A name just for humans
```

```
    ,  128  // This stack size can be checked & adjusted by reading the
Stack Highwater
    ,  NULL
    ,  2  // Priority, with 3 (configMAX_PRIORITIES - 1) being the
highest, and 0 being the lowest.
    ,  NULL );

  xTaskCreate(
    TaskAnalogRead
    ,  "AnalogRead"
    ,  128  // Stack size
    ,  NULL
    ,  1  // Priority
    ,  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 TaskBlink(void *pvParameters)  // This is a task.
{
  (void) pvParameters;

/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  Most Arduinos have an on-board LED you can control. On the UNO,
LEONARDO, MEGA, and ZERO
  it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN takes
care
  of use the correct LED pin whatever is the board used.

  The MICRO does not have a LED_BUILTIN available. For the MICRO board
please substitute
  the LED_BUILTIN definition with either LED_BUILTIN_RX or LED_BUILTIN_TX.
```

```
  e.g. pinMode(LED_BUILTIN_RX, OUTPUT); etc.

  If you want to know what pin the on-board LED is connected to on your
Arduino model, check
  the Technical Specs of your board  at
https://www.arduino.cc/en/Main/Products

  This example code is in the public domain.

  modified 8 May 2014
  by Scott Fitzgerald

  modified 2 Sep 2016
  by Arturo Guadalupi
*/

  // 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
    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;

/*
  AnalogReadSerial
  Reads an analog input on pin 0, prints the result to the serial monitor.
  Graphical representation is available using serial plotter (Tools >
Serial Plotter menu)
  Attach the center pin of a potentiometer to pin A0, and the outside pins
to +5V and ground.

  This example code is in the public domain.
*/
```

```
  for (;;)
  {
    // read the input on analog pin 0:
    int sensorValue = analogRead(A0);
    // print out the value you read:
    Serial.println(sensorValue);
    vTaskDelay(1);  // one tick delay (15ms) in between reads for
stability
  }
}
```

Result: