|  |  |
| --- | --- |
| **Course code and name:** | B31DG - EMBEDDED SOFTWARE |
| **Type of assessment:** | **Individual** |
| **Coursework Title:** | Assignment 2 |
| **Student Name:** | Mohmadnoor Akhunji |
| **Student ID Number:** | H00429518 |

|  |
| --- |
| **Declaration of authorship. By signing this form:**   * **I declare** that the work I have submitted for individual assessment OR the work I have contributed to a group assessment, is entirely my own. I have NOT taken the ideas, writings or inventions of another person and used these as if they were my own. My submission or my contribution to a group submission is expressed in my own words. Any uses made within this work of the ideas, writings or inventions of others, or of any existing sources of information (books, journals, websites, etc.) are properly acknowledged and listed in the references and/or acknowledgements section. * I confirm that I have read, understood and followed the University’s Regulations on plagiarism as published on the [University’s website](https://www.hw.ac.uk/uk/students/studies/examinations/plagiarism.htm), and that I am aware of the penalties that I will face should I not adhere to the University Regulations. * I confirm that I have read, understood and avoided the different types of plagiarism explained in the University guidance on [Academic Integrity and Plagiarism](https://heriotwatt.sharepoint.com/sites/skillshub/SitePages/Academic-Integrity-and-Plagiarism.aspx)   **Student Signature** *(type your name): Mohmadnoor*  **Date**: *17/03/2023* |

Copy this page and insert it into your coursework file in front of your title page.

For group assessment each group member must sign a separate form and all forms must be included with the group submission.

**Your work will not be marked if a signed copy of this form is not included with your submission.**

**REPORT**

**INTRODUCTION**

**Assignment -2 Question Evolution**

Your cyclic executive will implement a machine monitor system. The system must execute the following 5 periodic tasks, with specified timings.

1. Output a digital signal. This should be HIGH for 200μs, then LOW for 50μs, then HIGH s, then LOW for 50μs, then LOW for 50μs, then HIGH s, then HIGH again for 30μs, then LOW for 50μs, then HIGH s, and repeat the same pattern once every 4ms [Period = 4ms / Rate = 250Hz] Timeline showing when each tasks is active at which time. X axis is time.

2. Measure the frequency of a 3.3v square wave signal, once every 20ms. The frequency will be in the range 333Hz to 1000Hz and the signal will be a standard square wave (50% duty cycle). Accuracy to 2.5% is acceptable. [Period = 20ms / Rate = 50Hz]

3. Measure the frequency of a second 3.3v square wave signal, once every 8ms. The frequency will be in the range 500Hz to 1000Hz and the signal will be a standard square wave (50% duty cycle). Accuracy to 2.5% is acceptable. [Period = 8ms / Rate = 125Hz]

4. Read one analogue input, and compute a filtered analogue value, by averaging the last 4 readings. [Period = 20ms / Rate = 50Hz] The analogue input must be connected to a maximum of 3.3Volts, using a potentiometer. The task should also visualise an error (using a LED) whenever (average\_analogue\_in > half of maximum range).

5. Log the following information once every 100ms [Period = 100ms / Rate = 10Hz] in comma delimited format, i.e. “%d,%d”) to the serial port at a baud rate of 9600 bits per seconds.

a) Frequency value measured by Task 2 (Hz, as integer)

b) Frequency value measured by Task 3 (Hz, as integer)

Important: frequencies should be scaled and bounded between 0 to 99. For example, the output “0,0” will mean Task 2 has measured a frequency of 333Hz, or less, and Task 3 has measured frequency of 500Hz or less. The output “99,99” will mean that both tasks have measured frequencies equal or above 1000H.

**Components That Used**

* ESP32 Wroom Dev Module
* 2 LEDs
* Breadboard
* Jumper Cables
* Oscilloscope
* Potentiometer
* Signal Generator

The ESP32-WROOM-32 is a potent, universal Wi-Fi + Bluetooth + Bluetooth LE MCU module that can handle a wide range of functions, including voice encoding, music streaming, and decoding, as well as low-power sensor networks.

Chart

Description automatically generated with medium confidence

**Figure (1.1) :- ESP32**

**ESP32 Connection Diagram**

ESP32 circuit diagram simulates on WOKWI, Wokwi is an**online simulator for Arduino and Electronics**. It's designed for makers, by makers. You can use Wokwi to learn Arduino programming, prototype your ideas, and share your projects with other makers. Look below figure for the understanding of a circuit diagram.

Diagram, schematic

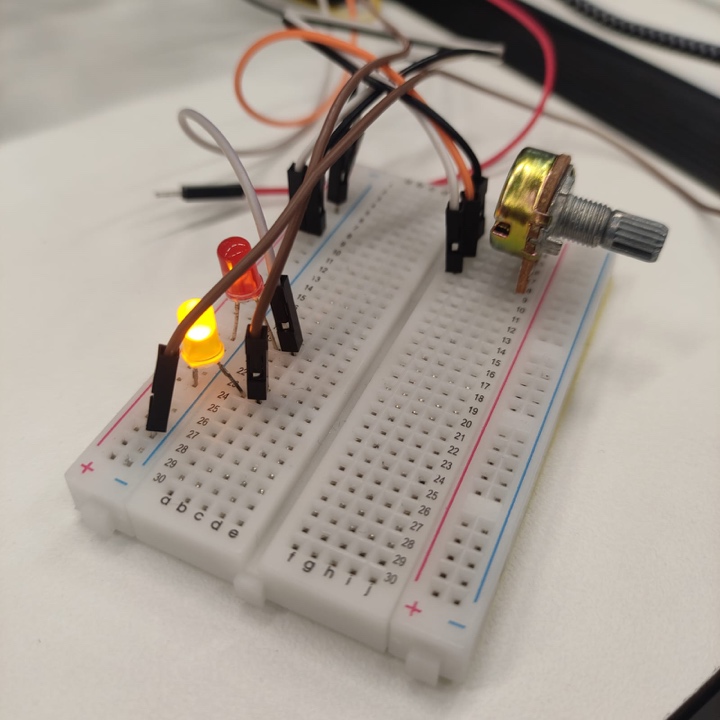
Description automatically generated

**Figure (1.2) :- Circuit Diagram on WOKWI for simulation**

A picture containing indoor

Description automatically generated

**Figure (1.3): Actual Circuit Diagram**



**Figure (1.4): Output (Potentiometer for task 4)**

**Cyclic Executive Scheduling: -**

A cyclic executive is an alternative which works like real-time operating system. It is used for Multitasking execution of tasks in cyclic manner in loop.

In given assignment there is 5 tasks which are needs to be executed using cyclic scheduling.

Frame duration = 4ms

Given periods of each tasks Pi,

1. Task 1 period = 4ms / 250Hz
2. Task2 period = 20ms / 50Hz
3. Task3 period = 8ms / 125Hz
4. Task4 period = 20ms / 50Hz
5. Task5 period = 100ms / 10Hz

After measuring each Ci,

1. Task1 = 1ms
2. Task2 = 3ms
3. Task3 = 3ms
4. Task4 = 1ms
5. Task5 = 1ms

LCM = 200

* In the program of cyclic executive, I have used two libraries one is Provided in the assignment which is “B31DGCyclicExecutiveMonitor” library. Which is used to monitor the tasks and print the violations of the tasks. The second library that I used in a program is “Ticker” library. Which is used to execute tasks one by one by ticking the object of the ticker.

#include <Ticker.h>

#include <B31DGMonitor.h>

Ticker ticker;

B31DGCyclicExecutiveMonitor monitor;

void frame() {

unsigned int slot = frameCounter % 10;

frameCounter++;

switch(slot){ case n: break;

case n+1: }

}

* In a program void loop is empty as we are incrementing frame counter in frame function. In void setup I’m calling a frame function and after that I’m using one ticker object to call frame of 4 milliseconds for the frame function that I created for execute tasks in the switch case statement of slot. And I assigned the pins as accordingly which are input pins and which are output pins.

void setup(void)

{

pinMode(task\_1\_led, OUTPUT); // output task1 led

pinMode(task\_2\_frq, INPUT); // input task2 pin measure frequency

pinMode(task\_3\_frq, INPUT); // input task3 pin measure frequency

pinMode(task4\_ptn, INPUT); // input task4 pin for potentiometer

pinMode(task\_4\_led\_err, OUTPUT); // output task4 led to display error

Serial.begin(9600);

monitor.startMonitoring();

frame();// TO-DO: wait the next frame

ticker.attach\_ms(4,frame);

}

* In Task1, the led is going to high for 200us the low for 50us, then high for 30us and repeat the same every 4ms of time period.

// Task 1, takes 1ms once every 4ms

void JobTask1(void)

{

monitor.jobStarted(1);

digitalWrite(task\_1\_led, HIGH); // set pin 2 high for 200us

delayMicroseconds(200);

digitalWrite(task\_1\_led, LOW); // set pin 2 low for 50us

delayMicroseconds(50);

digitalWrite(task\_1\_led, HIGH); // set pin 2 high for 30us

delayMicroseconds(30);

digitalWrite(task\_1\_led, LOW); // set pin 2 low for remaining period

monitor.jobEnded(1);

}

* Task 2 & Task 3 is used to measure the frequency in which task 2 is in between 333Hz to 1000Hz and in task 3 it’s in between 500Hz to 1000Hz. And both tasks measure 3.3v frequency of square wave signal. Task 2 is running every 20ms and Task 3 is running every 8ms.

// Task 2, takes 3ms once every 20ms

void JobTask2(void)

{

monitor.jobStarted(2);

int count = 0;

count += pulseIn(task\_2\_frq, HIGH); // measure the pulse width of the input signal

count = count\*2;

frequency\_1 = 1000000.0 / (count); // calculate frequency in Hz

frequency\_1 = constrain(frequency\_1, 333, 1000); // frequency between 333 and 1000 Hz

//Serial.println(frequency\_1); // output frequency

monitor.jobEnded(2);

}

// Task 3, takes 3ms once every 8ms

void JobTask3(void)

{

monitor.jobStarted(3);

int count2 = 0;

count2 += pulseIn(task\_3\_frq, HIGH); // measure the pulse width of the input signal

count2 = count2\*2;

frequency\_2 = 1000000.0 / (count2 ); // calculate frequency in Hz

frequency\_2 = constrain(frequency\_2, 500, 1000); // frequency between 500 and 1000 Hz

//Serial.println(frequency\_2); // output frequency

monitor.jobEnded(3);

}

* Task 4 period is also 20ms in cyclic scheduling. But it also has some input data using potentiometer which is used for this. In this task 4 if the analog input of 3.3v using potentiometer’s half of the maximum range is greater than the last 4 readings then turn led on and give error.

// Task 4, takes 1ms once every 20ms

void JobTask4(void)

{

monitor.jobStarted(4);

const int max\_analog\_input = 1023;

const int num\_readings = 4;

int readings[num\_readings];

int indx = 0;

int total = 0;

int fltr\_val = 0;

for (int i = 0; i < num\_readings; i++)

{

readings[i] = 0;

}

int analogValue = analogRead(task4\_ptn); // Read analog input

total = total - readings[indx]; // Subtract the oldest reading from the total

total = total + analogValue; // Add to the total

readings[indx] = analogValue; // Store reading in the readings array

indx++; // Increment the indx

if (indx >= num\_readings)

{

indx = 0;

}

fltr\_val = total / num\_readings; // Compute the filtered value of 4 readings

if (fltr\_val > max\_analog\_input / 2) {

digitalWrite(task\_4\_led\_err, HIGH);

}

else {

digitalWrite(task\_4\_led\_err, LOW);

}

monitor.jobEnded(4);

}

* Task 5 is basically mapping the frequency which is measured in Task 2 and Task 3 into “0,0” and “99,99” and the Task 5 is printing frequency in this format every 100ms.

// Task 5, takes 1ms once every 100ms

void JobTask5(void)

{

monitor.jobStarted(5);

int task\_2\_frq = 0;

int task\_3\_frq = 0;

task\_2\_frq = pulseIn(task\_2\_frq, HIGH, 20000) == 0 ? 0 : 1000000 / pulseIn(task\_2\_frq, HIGH, 20000);

task\_2\_frq = map(task\_2\_frq, 333, 1000, 0, 99); // map frequency between to 0-99

task\_3\_frq = pulseIn(task\_3\_frq, HIGH, 8000) == 0 ? 0 : 1000000 / pulseIn(task\_2\_frq, HIGH, 8000);

task\_3\_frq = map(task\_3\_frq, 500, 1000, 0, 99); // map frequency value between 0-99

// Serial.println(task\_2\_frq); //To print frequency of Task2

// Serial.println(task\_3\_frq); //To print frequency of Task3

monitor.jobStarted(5);

}

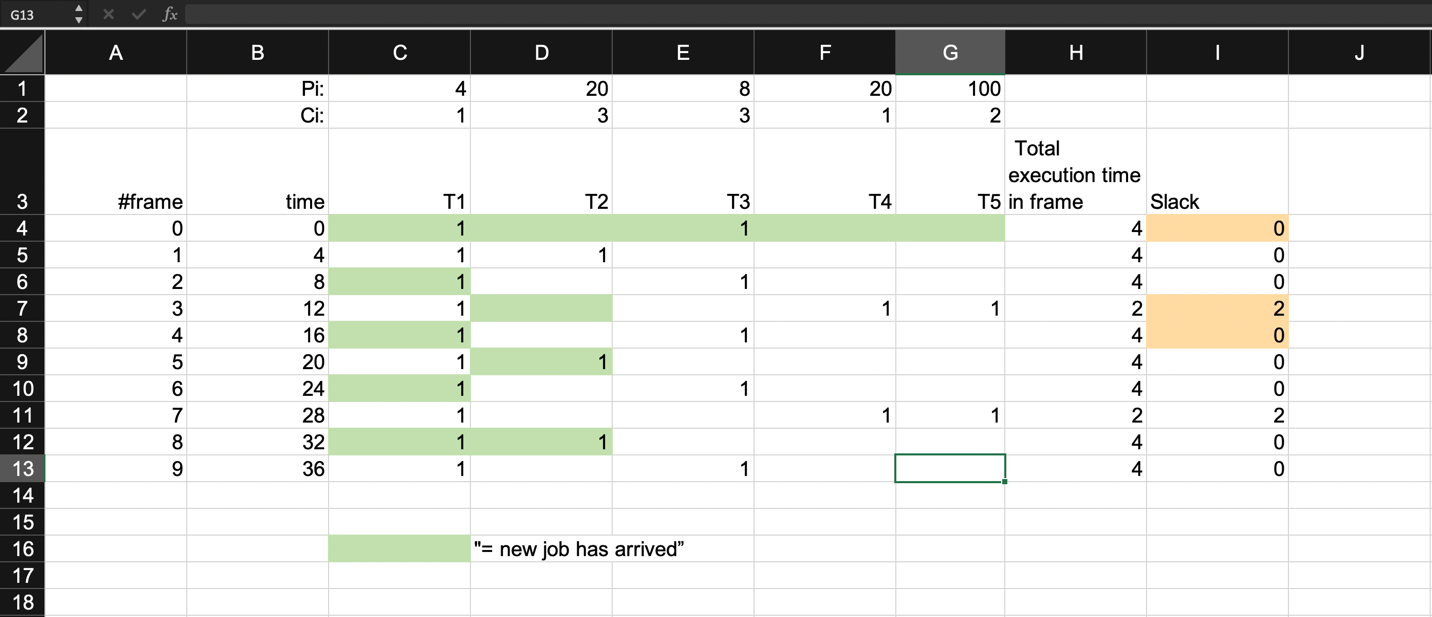
* **Output:-**

By using given monitor library, it prints the violation for monitoring of cyclic scheduling executive tasks missed deadlines every 10 seconds. So, in output program I got so far for 4 Tasks I’m not getting any kind of violations and not missing any deadlines for 4 tasks. I tried for task 5 but not getting the violation measurement for task 5 and one more thing if I’m printing anything like any measured frequency by task2, task3 and task4 then the violation is happening for these tasks because printing statements take more memory.

**Graphical user interface, text, application, Word

Description automatically generated**

**Excel Sheet Cyclic executive scheduling**



**Arduino IDE Code**

#include <Ticker.h>

#include <B31DGMonitor.h>

Ticker ticker;

B31DGCyclicExecutiveMonitor monitor;

#define FRAME\_DURATION\_MS 4 // 4ms

int task\_1\_led = 25; //output port for LED of task 1

int task\_2\_frq = 12;//input port to measure task-2 frequency

int task\_3\_frq = 14;//input port to measure task-3 frequency

int task4\_ptn = 26;//input port to show analog frequency

int task\_4\_led\_err = 27;//output port to blink the led for error using potentiometer

float frequency\_2;

float frequency\_1;

unsigned long frameTime = 0;//Initializing frameTimer

unsigned long frameCounter = 0;//Initializing frameCounter

unsigned int counter = 0;

//----------------------------------------------------------------------------------------------------------------------------------------------

void frame() {

unsigned int slot = frameCounter % 10;

frameCounter++;

// Increase frame counter and reset it after 10 frames

switch (slot) {

case 0: JobTask1(); JobTask3(); break;

case 1: JobTask1();JobTask2(); break;

case 2: JobTask1(); JobTask3(); break;

case 3: JobTask1(); JobTask4();break;

case 4: JobTask1(); JobTask3(); break;

case 5: JobTask1();JobTask2(); break;

case 6: JobTask1(); JobTask3(); break;

case 7: JobTask1(); JobTask4();break;

case 8: JobTask1(); JobTask3(); break;

case 9: JobTask1();

}

}

//----------------------------------------------------------------------------------------------------------------------------------------------

void setup(void)

{

pinMode(task\_1\_led, OUTPUT); // output task1 led

pinMode(task\_2\_frq, INPUT); // input task2 pin measure frequency

pinMode(task\_3\_frq, INPUT); // input task3 pin measure frequency

pinMode(task4\_ptn, INPUT); // input task4 pin for potentiometer

pinMode(task\_4\_led\_err, OUTPUT); // output task4 led to display error

Serial.begin(9600);

while(!Serial);

Serial.println("Ready");

monitor.startMonitoring();

frame();// TO-DO: wait the next frame

ticker.attach\_ms(4,frame);

}

//----------------------------------------------------------------------------------------------------------------------------------------------

void loop(void) // Single time slot function of the Cyclic Executive (repeating)

{

}

//----------------------------------------------------------------------------------------------------------------------------------------------

// Cyclic executive tasks

//----------------------------------------------------------------------------------------------------------------------------------------------

//----------------------------------------------------------------------------------------------------------------------------------------------

// Task 1, takes 1ms once every 4ms

void JobTask1(void)

{

monitor.jobStarted(1);

digitalWrite(task\_1\_led, HIGH); // set pin 2 high for 200us

delayMicroseconds(200);

digitalWrite(task\_1\_led, LOW); // set pin 2 low for 50us

delayMicroseconds(50);

digitalWrite(task\_1\_led, HIGH); // set pin 2 high for 30us

delayMicroseconds(30);

digitalWrite(task\_1\_led, LOW); // set pin 2 low for remaining period

monitor.jobEnded(1);

}

//----------------------------------------------------------------------------------------------------------------------------------------------

// Task 2, takes 3ms once every 20ms

void JobTask2(void)

{

monitor.jobStarted(2);

int count = 0;

count += pulseIn(task\_2\_frq, HIGH); // measure the pulse width of the input signal

count = count\*2;

frequency\_1 = 1000000.0 / (count); // calculate frequency in Hz

frequency\_1 = constrain(frequency\_1, 333, 1000); // frequency between 333 and 1000 Hz

//Serial.println(frequency\_1); // output frequency //uncoment this statement to view the current input voltage

monitor.jobEnded(2);

}

//----------------------------------------------------------------------------------------------------------------------------------------------

// Task 3, takes 3ms once every 8ms

void JobTask3(void)

{

monitor.jobStarted(3);

int count2 = 0;

count2 += pulseIn(task\_3\_frq, HIGH); // measure the pulse width of the input signal

count2 = count2\*2;

frequency\_2 = 1000000.0 / (count2 ); // calculate frequency in Hz

frequency\_2 = constrain(frequency\_2, 500, 1000); // frequency between 500 and 1000 Hz

//Serial.println(frequency\_2); // output frequency//uncoment this statement to view the current input voltage

monitor.jobEnded(3);

}

//----------------------------------------------------------------------------------------------------------------------------------------------

// Task 4, takes 1ms once every 20ms

void JobTask4(void)

{

monitor.jobStarted(4);

const int max\_analog\_input = 1023;

const int num\_readings = 4;

int readings[num\_readings];

int indx = 0;

int total = 0;

int fltr\_val = 0;

for (int i = 0; i < num\_readings; i++)

{

readings[i] = 0;

}

int analogValue = analogRead(task4\_ptn); // Read analog input

total = total - readings[indx]; // Subtract the oldest reading from the total

total = total + analogValue; // Add to the total

readings[indx] = analogValue; // Store reading in the readings array

indx++; // Increment the indx

// Wrap the indx if it exceeds the number of readings

if (indx >= num\_readings)

{

indx = 0;

}

fltr\_val = total / num\_readings; // Compute the filtered value of 4 readings

// If the filtered value is greater than half of the maximum range, turn the led on esle led off

if (fltr\_val > max\_analog\_input / 2) {

digitalWrite(task\_4\_led\_err, HIGH);

}

else {

digitalWrite(task\_4\_led\_err, LOW);

}

// Serial.println(fltr\_val); //uncoment this statement to view the current input voltage

monitor.jobEnded(4);

}

//----------------------------------------------------------------------------------------------------------------------------------------------

// Task 5, takes 1ms once every 100ms

void JobTask5(void)

{

monitor.jobStarted(5);

int task\_2\_frq = 0;

int task\_3\_frq = 0;

// Measure the frequency of Task 2 signal

task\_2\_frq = pulseIn(task\_2\_frq, HIGH, 20000) == 0 ? 0 : 1000000 / pulseIn(task\_2\_frq, HIGH, 20000);

task\_2\_frq = map(task\_2\_frq, 333, 1000, 0, 99); // map frequency between to 0-99

// Measure the frequency of Task 3 signal

task\_3\_frq = pulseIn(task\_3\_frq, HIGH, 8000) == 0 ? 0 : 1000000 / pulseIn(task\_2\_frq, HIGH, 8000);

task\_3\_frq = map(task\_3\_frq, 500, 1000, 0, 99); // map frequency value between 0-99

// Send the frequency values to the serial port

Serial.println(task\_2\_frq); //To print frequency of Task2

Serial.println(task\_3\_frq); //To print frequency of Task3

monitor.jobStarted(5);

}

**Github Repository Screenshot**

**A screenshot of a computer

Description automatically generated with medium confidence**

**Github Link: -**

[**https://github.com/noor-akhunji/Embedded\_Assignment\_2**](https://github.com/noor-akhunji/Embedded_Assignment_2)