

REPORT
ASSIGNMENT NO. 2

INTRODUCTION

The code consists of 5 tasks, each performing a specific function:

Task 1: This task blinks an LED at a specific frequency. It uses the digital Write () function to set the LED pin HIGH and LOW, and the delay Microseconds() function to control the timing of the blinking.

Task 2: This task measures the frequency of a signal on an input pin using an interrupt-driven approach. It uses the pulse In () function to measure the pulse width of the signal and calculates the frequency based on the pulse width and the time interval between measurements.

Task 3: This task measures the frequency of a signal on an input pin using a polling approach. It uses the pulse In() function to measure the pulse width of the signal and calculates the frequency based on the pulse width and the time interval between measurements.

Task 4: This task reads an analog voltage on an input pin and turns on an LED if the voltage is above a threshold. It uses the analog Read () function to read the voltage and the digital Write () function to set the LED pin HIGH or LOW.

Task 5: This task measures the frequency of Tasks 2 and 3 and logs the values to the serial monitor. It uses the millis () function to measure the time interval between measurements and calculates the frequency based on the time interval. The code also uses a monitoring library (B31DGMonitor.h) to measure the execution time of each task. The monitor.jobStarted() and monitor.jobEnded() functions are called at the beginning and end of each task, respectively.

1.ESP32

The ESP32 is a single chip that integrates both Wi-Fi and Bluetooth technologies, operating at 2.4 GHz frequency. It has been designed using TSMC's low power 40 nm technology to achieve optimal power and RF performance and can operate reliably and with versatility in a wide range of applications and power scenarios. The ESP32 is equipped with a total of 34 digital pins that are similar to the digital pins found in Arduino. These pins enable the addition of various components such as LEDs, sensors, and more to projects. The ESP32 WROOM module specifically offers 25 GPIO pins, all of which are input pins. Among these input pins, some have input pull up while others do not have internal pull up.



Fig 1.1: ESP32

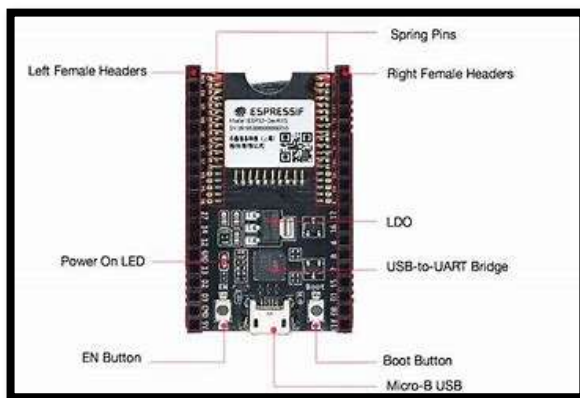


Fig 1.2: ESP32 DIAGRAM

2. Potentiometer



Fig 2.1: Potentiometer

A potentiometer, also referred to as a pot or potmeter, is a type of 3-terminal resistor that allows for the manual adjustment of resistance in order to regulate the flow of electric current. The potentiometer functions as a voltage divider that can be customized to suit specific needs. Potentiometers are passive electronic components that operate by changing the location of a sliding contact along a consistent resistance. The input voltage is applied across the complete length of the resistor in a potentiometer, and the output voltage is measured as the difference in voltage between the stationary and sliding contacts, as illustrated in the diagram.

3.Design of Cyclic Executive

Our cyclic executive is designed to schedule the execution of N periodic tasks in a real-time system, without relying on a real-time operating system. The tasks are executed in a cyclic manner, where each task is executed once during its specified period. The tasks are treated as hard real-time tasks, meaning they must complete before their hard deadline, which is the end of their period.

CIRCUIT DIAGRAM

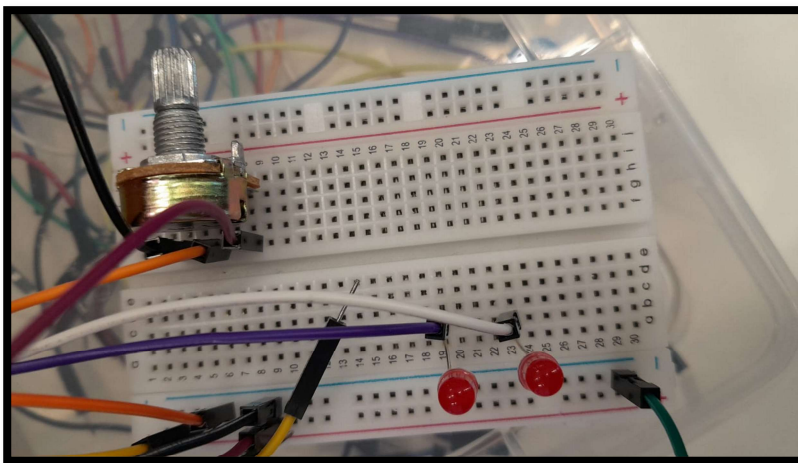


Fig 1.1: Simulation of circuit diagram

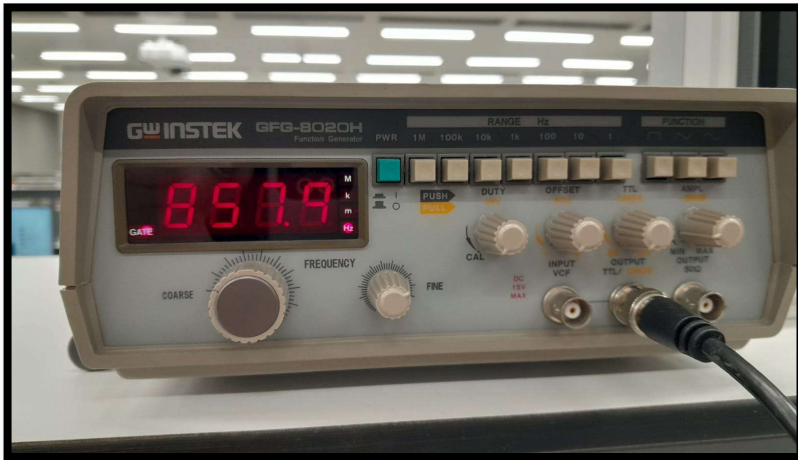


Fig 1.2: Function generator

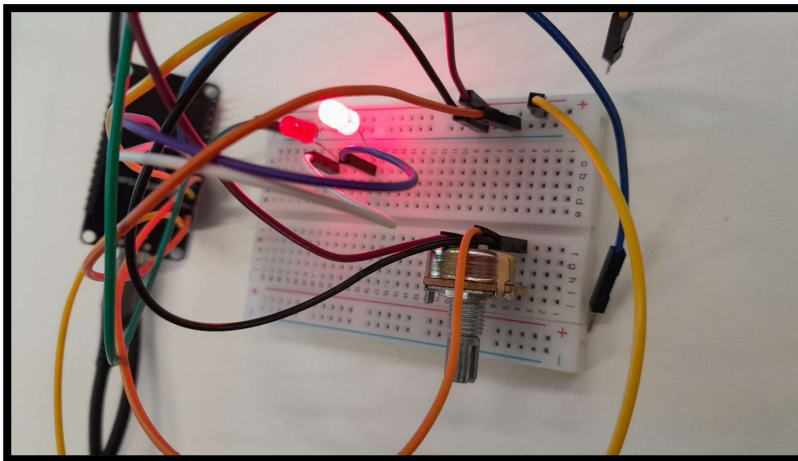


Fig1.2: Simulation of circuit diagram blink light

		Pi:	4	20	8	20			
		Ci:	1	3	3	1			
#frame	time	T1	T2	T3	T4	Total execution time in frame	Slack		
0	0	1		1		4	0		
1	4		1			3	-1		
2	8	1				1	1		
3	12				1	1	1		
4	16		1			3	-1		
5	20	1				1	1		
6	24		1			3	-1		
7	28	1				1	1		
8	32	1				1	1		
9	36		1			3	-1		
10	40	1		1		4	-2		

Fig: Excel sheet

Code :

```
#include <B31DGMonitor.h>

B31DGCyclicExecutiveMonitor monitor;

// Author: GUNJAN DOD

#define Duration of FRAME 4 // 4ms

int L1_LED=19; //output port for LED of task 1

int T2_Freq=12; //input port from signal generator to count task-2Freq_1

int T3_Freq=14; //input port from signal geneerator to count task-3Freq_1

int T4_POT=27; //input port from potentiometer to show analogFreq_1

int LED_error=26; //output port to blink the led for error from potentiometer

unsigned long Timer_FRAME = 0; //Initializing Timer_FRAME

unsigned long Timer_Counter = 0; //Initializing Timer_Counter
```

```
void setup(void)
{
    monitor.startMonitoring();
    Serial.begin(9600);
    while(!Serial);
    Serial.println("Ready");
    pinMode(L1_LED, OUTPUT); // set pin 2 as output for Task_1
    pinMode(T2_Freq, INPUT); // set pin 2 as input for Task_2
    pinMode(T3_Freq, INPUT); // set pin 2 as input for Task_3
    pinMode(T4_POT, INPUT); // set pin 2 as input for Task_4
    pinMode(LED_error, OUTPUT); //Led pin output for Task_4
    // Initialize readings array with 0's
}

// Increase FRAME counter and reset it after 10 FRAMEs

void FRAME() {

    static unsigned int slot = 0; // use static variable to retain value between calls
    slot = (slot + 1) % 10; // increment and wrap slot index

    switch (slot) {
        case 0: JOB_TASK_1();JOB_TASK_3();break;
        case 1: JOB_TASK_1(); JobTask2();break;
        case 2: JOB_TASK_1();JOB_TASK_3();break;
        case 3: JOB_TASK_1();JobTask4();break;
        case 4: JOB_TASK_1();JOB_TASK_3();break;
        case 5: JOB_TASK_1();JobTask2();break;
        case 6: JOB_TASK_1();JOB_TASK_3();break;
```

```
case 7: JOB_TASK_1();JobTask4();break;
case 8: JOB_TASK_1();JOB_TASK_3();break; //JobTask5();break;
case 9: JOB_TASK_1();
}
}

void loop(void) // Single time slot function of the Cyclic Executive (repeating)
{
  /*
  unsigned long bT = micros();
  JobTask4();

  unsigned long timeItTook = micros()-bT;
  Serial.print("Duration SerialOutput Job = ");
  Serial.print(timeItTook);
  exit(0);
  */
  FRAME();// TO-DO: wait the next FRAME

}

// Task 1, takes 0.9ms
void JOB_TASK_1(void)
{
  monitor.jobStarted(1);
  Serial.println("Task 1 Started");
  digitalWrite(L1_LED, HIGH); // set pin 2 high for 200us
  delayMicroseconds(200);
  digitalWrite(L1_LED, LOW); // set pin 2 low for 50us
  delayMicroseconds(50);
```

```
digitalWrite(L1_LED, HIGH); // set pin 2 high for 30us
delayMicroseconds(30);
digitalWrite(L1_LED, LOW); // set pin 2 low for remaining period
//delayMicroseconds(1720); // wait for 4ms minus the time spent in the loop
monitor.jobEnded(1);
Serial.println("Task 1 Done");
}

// Task 2, takes 4ms
void JobTask2(void)
{
    monitor.jobStarted(2);
    Serial.println("Task 2 Started");
    // #define SAMPLES 10 // number of samples to take
    int count = 0;
    // for (int i = 0; i < SAMPLES; i++)
    // {
        count += pulseIn(T2_Freq, HIGH); // count the pulse width of the input signal which is high
    // }

    count = count*2; // Pulse width*2 to calculate positive and negative pulse as whole waveform
    float Freq_1 = 1000000.0 / (count); // SAMPLES); // calculate Freq_1 in Hz
    Freq_1 = constrain(Freq_1, 333, 1000); // bound Freq_1 between 333 and 1000 Hz
    int scaled_Freq_1 = map(Freq_1, 333, 1000, 0, 99); // scale Freq_1 between 0 and 99 for
    Serial.println("Freq_1:"); // output Freq_1 value to serial port
    Serial.println(Freq_1); // output Freq_1 value to serial port
    //delayMicroseconds(800); // wait for 20ms minus the time spent in the loop
    Serial.println("Task 2 Done");
    monitor.jobEnded(2);
}
```



```
// Task 3, takes 4ms
void JOB_TASK_3(void)
{
    monitor.jobStarted(3);
    Serial.println("Task 3 Started");
    //#define SAMPLES 8 // number of samples to take for pulse
    int count2 = 0;
    //for (int i = 0; i < SAMPLES; i++) {
        //count2 += pulseIn(T3_Freq, HIGH); // count the pulse width of the input signal that is high
    //}
    count2 = count2*2; //Pulse width*2 to calculate positive and negative pulse as whole waveform
    float Freq_12 = 1000000.0 / (count2); // SAMPLES); // calculate Freq_1 in Hz
    Freq_12 = constrain(Freq_12, 500, 1000); // bound Freq_1 between 500 and 1000 Hz
    int scaled_Freq_12 = map(Freq_12, 500, 1000, 0, 99); // scale Freq_1 between 0 and 99
    Serial.println("Freq_1_2:"); // output Freq_1 value to serial port
    Serial.println(Freq_12); // output Freq_1 value to serial port
    //delayMicroseconds(920); // wait for 8ms minus the time spent in the loop
    monitor.jobEnded(3);
    Serial.println("Task 3 Done");
}
```

```
// Task 4, takes 4ms
void JobTask4(void)
{
    monitor.jobStarted(4);
    Serial.println("Task 4 Started");
    const int maxAnalogIn = 1023;
    const int numReadings = 4;
    int readings[numReadings];
```

```
int index = 0;
int total = 0;
int filteredValue = 0;
for (int i = 0; i < numReadings; i++)
{
    readings[i] = 0;
}

// Read the analog input value
int analogValue = analogRead(T4_POT);
// Subtract the oldest reading from the total
total -= readings[index];
// Add the new reading to the total
total += analogValue;
// Store the new reading in the readings array
readings[index] = analogValue;
// Increment the index
index++;
// Wrap the index if it exceeds the number of readings
if (index >= numReadings)
{
    index = 0;
}

// Compute the filtered value as the average of the readings
filteredValue = total / numReadings;
// If the filtered value is greater than half of the maximum range, turn on the LED
if (filteredValue > maxAnalogIn / 2) {
    digitalWrite(LED_error, HIGH);
    Serial.println("error led HIGH");
}
```

```
} else {  
    digitalWrite(LED_error, LOW);  
    Serial.println("error led LOW");  
  
}  
  
// Send the filtered value to the serial port  
Serial.println(filteredValue);  
  
// Delay for 20ms  
//delay(20);  
monitor.jobEnded(4);  
Serial.println("Task 4 Done");//End of Task 4  
}  
  
// Task 5, takes 4ms  
void JobTask5(void)  
{  
    //Task 5  
    Serial.println("Task 5 started");//Start of Task 5  
    int task2Freq = 0;  
    int task3Freq = 0;  
    // count theFreq_1 of Task 2 signal  
    task2Freq = pulseIn(T2_Freq, HIGH, 20000) == 0 ? 0 : 1000000 / pulseIn(T2_Freq, HIGH, 20000);  
    // Scale and bound theFreq_1 between to 0-99  
    task2Freq = map(T2_Freq, 333, 1000, 0, 99);  
    // count theFreq_1 of Task 3 signal  
    task3Freq = pulseIn(T3_Freq, HIGH, 8000) == 0 ? 0 : 1000000 / pulseIn(T2_Freq, HIGH, 8000);  
    // Scale and bound theFreq_1 value between 0-99  
    task3Freq = map(T3_Freq, 500, 1000, 0, 99);  
    // Send theFreq_1 values to the serial port  
    Serial.println(task2Freq);//To printFreq_1 of given waveform of Task2
```

```
Serial.println(task3Freq);//To printFreq_1 of given waveform of Task3
```

```
Serial.println("Task 5 Completed");// End of Task 5
```

```
}
```

GitHub Link: