EMBEDDED SYSTEMS ASSIGNMENT 2

#include <Arduino.h>

// Task 1

const unsigned long task1\_period = 4000;

const unsigned long task1\_on\_duration[7] = {200, 50, 200, 50, 50, 200, 30};

const unsigned long task1\_off\_duration[7] = {50, 50, 50, 50, 50, 50, 50};

unsigned long task1\_last\_run = 0;

unsigned long task1\_phase = 0;

// Task 2

const unsigned long task2\_period = 20000;

const unsigned long task2\_min\_freq = 333;

const unsigned long task2\_max\_freq = 1000;

unsigned long task2\_last\_run = 0;

unsigned long task2\_freq = 0;

// Task 3

const unsigned long task3\_period = 8000;

const unsigned long task3\_min\_freq = 500;

const unsigned long task3\_max\_freq = 1000;

unsigned long task3\_last\_run = 0;

unsigned long task3\_freq = 0;

// Task 4

const unsigned long task4\_period = 20000;

const unsigned long task4\_read\_period = 5;

const int task4\_analog\_pin = A0;

const float task4\_max\_value = 3.3;

const int task4\_num\_readings = 4;

unsigned long task4\_last\_run = 0;

float task4\_readings[task4\_num\_readings];

int task4\_reading\_index = 0;

float task4\_filtered\_value = 0;

bool task4\_error = false;

// Task 5

const unsigned long task5\_period = 100000;

const int task5\_baud\_rate = 9600;

const int task5\_min\_freq = 0;

const int task5\_max\_freq = 99;

unsigned long task5\_last\_run = 0;

void task1() {

unsigned long current\_time = millis();

if (current\_time - task1\_last\_run >= task1\_period) {

task1\_last\_run = current\_time;

digitalWrite(13, HIGH); // Output HIGH for 200us, 400us, and 600us

delayMicroseconds(task1\_on\_duration[task1\_phase]);

digitalWrite(13, LOW); // Output LOW for 50us

delayMicroseconds(task1\_off\_duration[task1\_phase]);

task1\_phase = (task1\_phase + 1) % 7;

}

}

void task2() {

unsigned long current\_time = millis();

if (current\_time - task2\_last\_run >= task2\_period) {

task2\_last\_run = current\_time;

unsigned long count = 0;

unsigned long start\_time = micros();

while (micros() - start\_time < 1000000) {

if (digitalRead(2) == HIGH) {

count++;

}

}

task2\_freq = count / (task2\_period / 1000.0);

}

}

void task3() {

unsigned long current\_time = millis();

if (current\_time - task3\_last\_run >= task3\_period) {

task3\_last\_run = current\_time;

unsigned long count = 0;

unsigned long start\_time = micros();

while (micros() - start\_time < 400000) {

if (digitalRead(3) == HIGH) {

count++;

}

}

task3\_freq = count / (task3\_period / 1000.0);

}

}

Task 1: The first task in the cyclic executive is to output a digital signal. The signal should be HIGH for 200μs, then LOW for 50μs, then HIGH for 150μs, then LOW for 50μs, then LOW for 50μs, then HIGH for 150μs, then HIGH again for 30μs, then LOW for 50μs, then HIGH for 150μs, and then repeat the same pattern once every 4ms [Period = 4ms / Rate = 250Hz]. This task will require precise timing to ensure that the signal is generated correctly.

Task 2: The second task is to measure the frequency of a 3.3v square wave signal, once every 20ms. The signal’s frequency will be 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]. This task will involve counting the number of rising edges of the square wave within a fixed time and then using that count to calculate the frequency.

Task 3: The third task is to measure the frequency of a second 3.3v square wave signal, once every 8ms. The signal’s frequency will be 500Hz to 1000Hz, and the signal will be a standard square wave (50% duty cycle). An accuracy of 2.5% is acceptable. [Period = 8ms / Rate = 125Hz]. Similar to Task 2, this task will involve counting the number of rising edges of the square wave within a fixed time period and then using that count to calculate the frequency.

#include <Arduino.h>

#define ANALOG\_IN A0

#define LED\_PIN 13

#define ANALOG\_MAX 1023 // maximum analog input value

float filtered\_analog\_value = 0; // filtered value

int analog\_sum = 0; // sum of the last 4 analog readings

int analog\_readings[4]; // array to hold the last 4 analog readings

int analog\_index = 0; // index for analog\_readings array

void task4() {

// Read the analog input

analog\_sum -= analog\_readings[analog\_index];

analog\_readings[analog\_index] = analogRead(ANALOG\_IN);

analog\_sum += analog\_readings[analog\_index];

analog\_index = (analog\_index + 1) % 4;

// Calculate the filtered value

filtered\_analog\_value = analog\_sum / 4.0;

// Visualize an error if the filtered value is greater than half of maximum range

if (filtered\_analog\_value > ANALOG\_MAX / 2) {

digitalWrite(LED\_PIN, HIGH);

} else {

digitalWrite(LED\_PIN, LOW);

}

}

Read one analogue input, and compute a filtered analogue value, by averaging the last 4 readings. [Period = 20ms / Rate = 50Hz]

**EXPLANATION**

We can use a moving average filter to smooth out the values from the analog input to accomplish this task. We will read the analog input four times in each period and take the average of the four readings.

we are using the built-in **analogRead()** function in the Arduino code, and the **analogio.AnalogIn().value** method in the CircuitPython code, to read the analog input. We are also using the built-in **digitalWrite()** function in the Arduino code, and the **digitalio.DigitalInOut().value** method in the CircuitPython code, to control the LED.