НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ

Лабораторна робота 2

З дисципліни “Інтернет речей та вбудовані системи”  
Варіант 6

Виконав:

Студент 5 курсу

Групи ДА-21мп

Дзиговський В.І.

Київ 2023

Лабораторна робота 2

Завдання 1. Налаштувати незалежний сторожовий таймер на скидання мікроконтролера через 5N мс, де N – номер варіанту.

Код програми:

#include <Arduino.h>

unsigned long interval = 30;

void resetESP32(void\* arg) {

esp\_restart();

}

void setup() {

esp\_timer\_create\_args\_t timerConfig = {

.callback = resetESP32,

.name = "resetESP32"

};

esp\_timer\_handle\_t timer;

esp\_timer\_create(&timerConfig, &timer);

esp\_timer\_start\_once(timer, interval \* 1000);

}

void loop() {

}

resetESP32 – є функцією, яка викликається сторожовим таймером кожні 30 мс та скидує МК.

Результат: мікроконтролер скидується кожні 40мс

Завдання 2. Запустити програму у режимі відлагодження і перевірити точний час скидання МК. Час скидання занести у протокол.

Код програми:

#include <Arduino.h>

unsigned long interval = 30;

void resetESP32(void\* arg) {

Serial.println(String(double(micros())/1000) + "ms");

esp\_restart();

}

void setup() {

Serial.begin(9600);

esp\_timer\_create\_args\_t timerConfig = {

.callback = resetESP32,

.name = "resetESP32"

};

esp\_timer\_handle\_t timer;

esp\_timer\_create(&timerConfig, &timer);

esp\_timer\_start\_once(timer, interval \* 1000);

}

void loop() {

}

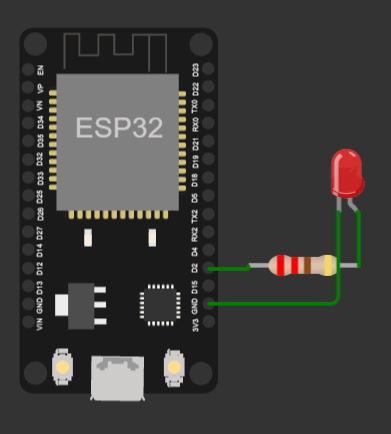
Було додано логування часу скидання МК.

Результат:



Завдання 3. Створити обробник переривання системного таймера. В тілі обробника написати код, який періодично підпалював і гасив світлодіод, під’єднаний до МК. Тривалість ввімкненого і вимкненого стану світлодіода 45мс і 45мс відповідно.

Схема:



Код програми:

#include <esp\_timer.h>

//Если не моргает - увеличь время

#define LED\_PIN 2

#define WORK\_TIME 45000

#define OFF\_TIME 45000

esp\_timer\_handle\_t work\_timer;

esp\_timer\_handle\_t off\_timer;

void IRAM\_ATTR timer\_work(void\* arg) {

digitalWrite(LED\_PIN, LOW);

esp\_timer\_start\_periodic(off\_timer, OFF\_TIME);

}

void IRAM\_ATTR off\_work(void\* arg) {

digitalWrite(LED\_PIN, HIGH);

esp\_timer\_start\_periodic(work\_timer, WORK\_TIME);

}

void setup() {

pinMode(LED\_PIN, OUTPUT);

esp\_timer\_create\_args\_t timer\_cfg = {

.callback = &timer\_work,

.arg = NULL,

.name = "work\_timer"

};

esp\_timer\_create\_args\_t off\_timer\_cfg = {

.callback = &off\_work,

.arg = NULL,

.name = "off\_timer"

};

esp\_timer\_create(&off\_timer\_cfg, &off\_timer);

esp\_timer\_create(&timer\_cfg, &work\_timer);

digitalWrite(LED\_PIN, HIGH);

esp\_timer\_start\_periodic(work\_timer, WORK\_TIME);

}

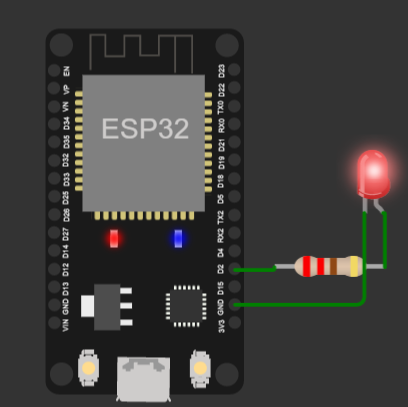
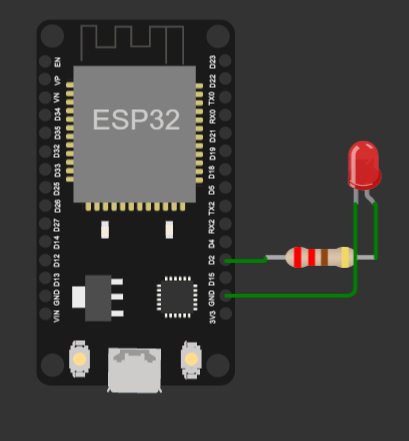
void loop() {

// виконуємо основні дії програми тут, які не повинні блокувати переривання

}

Створюємо два таймера work\_timer та off\_timer, кожен з яких змінює стан світлодіоду та перезапускає один одного.

Результат:



Завдання 4. На основі таймера загального призначення написати програму яка реалізує виклик переривання через певні інтервали часу для виконання деякої задачі. Інтервали часу : Т1 = 1.2мс, Т2 = 65мс, Т3 = 6500мс, Т4 = 150000мс, Т5 = 1.3мс.

Код програми:

#include <Ticker.h>

Ticker task1\_timer;

Ticker task2\_timer;

Ticker task3\_timer;

Ticker task4\_timer;

Ticker task5\_timer;

void setup() {

Serial.begin(9600);

task1\_timer.attach\_ms(1.2, task1);

task2\_timer.attach\_ms(65, task2);

task3\_timer.attach\_ms(6500, task3);

task4\_timer.attach\_ms(150000, task4);

task5\_timer.attach\_ms(1.3, task5);

}

void loop() {

}

void task1() {

Serial.println("Task 1 done");

}

void task2() {

Serial.println("Task 2 done");

}

void task3() {

Serial.println("Task 3 done");

}

void task4() {

Serial.println("Task 4 done");

}

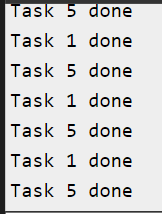
void task5() {

Serial.println("Task 5 done");

}

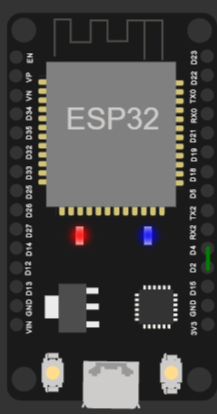
Для спрощення налаштування таймерів використано бібліотеку Ticker.

Результат:



Завдання 5. Написати програму для вимірювання періода ШІМ сигналу. Для цього один з виходів таймера налаштовується на режим ШІМ, інший – на режим захоплення імпульсів ШІМ. Вхід і вихід таймера з’єднуються між собою. Період ШІМ = 60мкс, γ = 40

Схема:



Код програми:

#include <Ticker.h>

#define PWN\_PIN 2

#define IN\_PIN 4

#define PWN\_PERIOD 60

#define GAMMA 40

int startPeriodTime = 0;

int endPeriodTime = 0;

int periodTime = 0;

Ticker setLow\_timer;

Ticker setHigh\_timer;

void setup() {

Serial.begin(9600);

pinMode(PWN\_PIN, OUTPUT);

pinMode(IN\_PIN, INPUT);

startPeriodTime = micros();

digitalWrite(PWN\_PIN, HIGH);

setLow\_timer.once(PWN\_PERIOD, setLow);

}

void loop() {

delay(1000);

Serial.println("Period: " + String(periodTime) + "us");

}

void setLow() {

digitalWrite(PWN\_PIN, LOW);

setHigh\_timer.once(PWN\_PERIOD\*100/GAMMA - PWN\_PERIOD, setHigh);

}

void setHigh() {

endPeriodTime = micros();

periodTime = endPeriodTime - startPeriodTime;

startPeriodTime = micros();

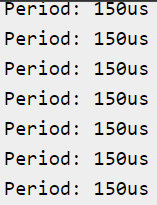
digitalWrite(PWN\_PIN, HIGH);

setLow\_timer.once(PWN\_PERIOD, setHigh);

}

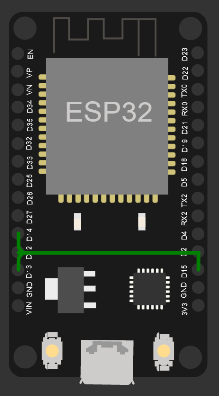
Два таймери визначають час наявності сигналу та його відсутність. Потім вираховують період ШІМ сигналу. Період записується раз в секунду

Результат:



Завдання 6. Написати програму розрахунку середньої напруги каналів АЦП регулярної групи через задані інтервали часу. Часовий інтервал 6мс.

Схема:



Код програми:  
  
#include <Ticker.h>

#define IN\_PIN 2

#define PWN\_PERIOD 6

int const pinsArray[4] = {12, 13, 14, 15};

int voltageArray[4] = {0, 0, 0, 0};

int voltageSum = 0;

int counter = 0;

Ticker timer;

void setup() {

Serial.begin(9600);

pinMode(pinsArray[0], OUTPUT);

pinMode(pinsArray[1], OUTPUT);

pinMode(pinsArray[2], OUTPUT);

pinMode(pinsArray[3], OUTPUT);

pinMode(IN\_PIN, INPUT);

analogWrite(pinsArray[counter], random(0, 1024));

timer.attach\_ms(PWN\_PERIOD, callback);

}

void loop() {

}

void callback() {

voltageArray[counter] = analogRead(IN\_PIN);

analogWrite(pinsArray[counter], 0);

if (counter >= 3) {

voltageSum = voltageArray[0] + voltageArray[1] + voltageArray[2] + voltageArray[3];

Serial.println("V12: " + String(voltageArray[0]) + " V13: " + String(voltageArray[1]) + " V14: " + String(voltageArray[2]) + " V15: " + String(voltageArray[3]) + " Sum: " + String(voltageSum));

counter = 0;

} else {

counter++;

}

analogWrite(pinsArray[counter], random(0, 1024));

}

Налаштовуємо таймер для виклику функції callback кожні 6мс та масиви для збереження каналів та їх значень напруги. У функції callback визначаємо напругу на одному піні та подаємо на інший, змінюючи лічильник.

Результат:

