

Міністерство освіти і науки України
Національний технічний університет України
«Київський Політехнічний Інститут імені Ігоря Сікорського»
 Кафедра конструювання електронно-обчислювальної апаратури

ЗВІТ

З виконання розрахунково-графічної роботи
з дисципліни “Обчислювальні та мікропроцесорні засоби в РЕА - 2”
на тему
«Розумний годинник»

Виконав:
студент групи ДК-72
Волинко Н.А.
Перевірив:
ст. вик. Бондаренко Н.О.

Київ – 2020

Технічне завдання:

Написати програму, в якій буде реалізовано розумний годинник. Даний годинник повинен мати можливість виведення всієї необхідної інформації для користувача, а саме повинен бути точний час, дата та температура навколошнього середовища. Користувач, визначившись в важливості даних параметрів, повинен змогти налаштувати виведення інформації найбільш зручним стилем для себе. Даний годинник повинен містити зручну можливість налаштування параметрів дати та часу, а також важливу на сьогодняшній день функцію будильника, з можливістю його точного налаштування та з достатньо помітним сигналом спрацювання.

Алгоритм роботи:

Враховуючи перераховані вимоги в технічному завданні, було прийнято використовувати для виведення інформації дисплей LCD 1602, адже враховуючи кількість інформації, яку можна вивести за допомогою цього дисплею, можна сказати, що інформативність годинника з таким дисплеєм буде достатньою. Для вимірювання температури було прийнято взяти датчик LM-385, адже він є прецизійний інтегральний датчик температури з широким діапазоном температур з високою точністю, завдяки чому є досить популярним. Для управління годинником взято 5 тактових кнопок, які розміщені у вигляді джойстика і таким чином надають змогу зручно їм користуватись, а також дозволяють забезпечити весь необхідний функціонал. Для індикації режиму налаштувань та сповіщення будильника взято світлодіоди, адже вони досить яскраві і добре привертають увагу, а враховуючи, що даний годинник є настільним, цього достатньо, для того щоб сповістити користувача, який працює за столом. Для роботи з цими всіма блоками було прийнято використовувати мікроконтролер STM32F407VG на базі плати Discovery, так як це багатофункціональний сучасний мікроконтролер, який може забезпечити надійну та безвідмовну роботу. Також було прийнято, що для кнопок не варто застосовувати переривання, адже воно перериватиме і збиватиме час затримки,

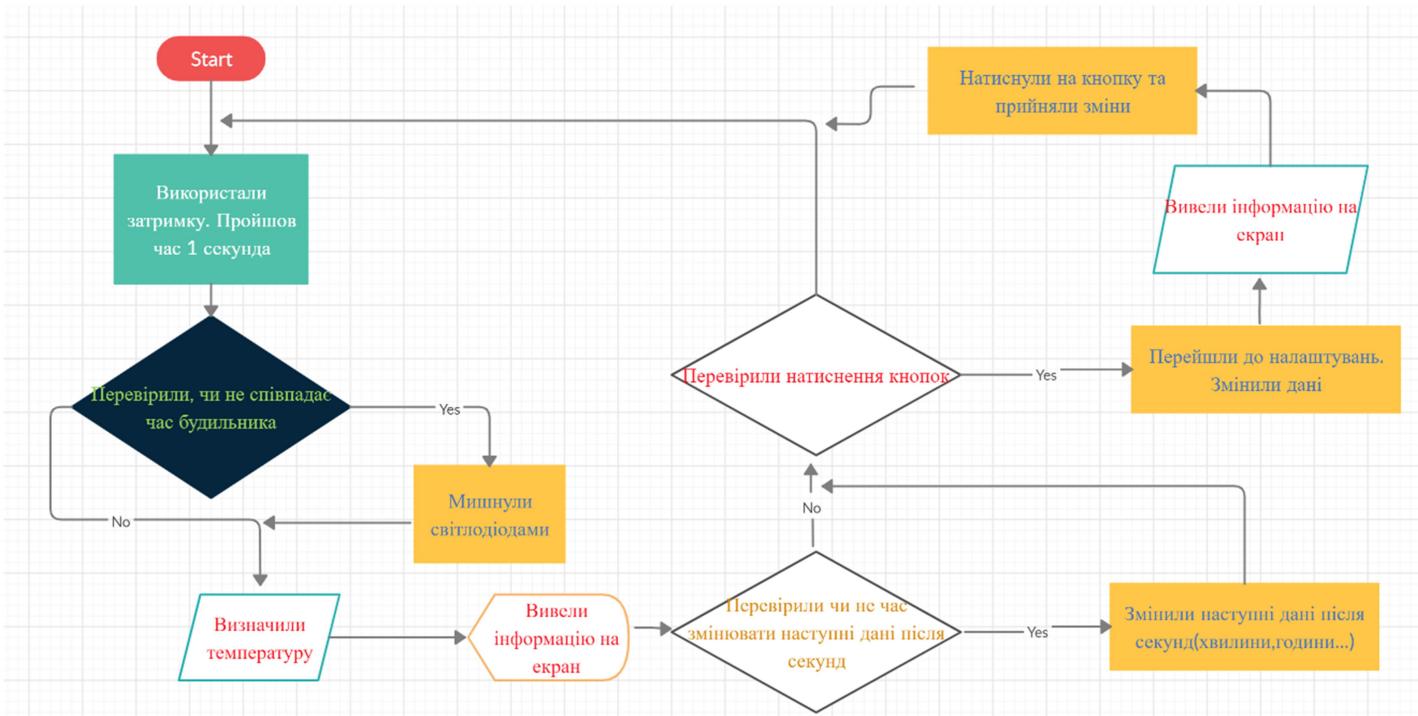
і тому годинник не буде працювати під час переривання. Тому кнопки застосовували як звичайні вхідні порти.

Спираючись на всі перераховані моменти створення годинника та спираючись на технічне завдання, було поставлено наступний алгоритм роботи:

1. Підключити LCD дисплей, бібліотеки для нього та налаштувати виведення інформації;
2. Налаштувати роботу системного таймера SysTick та написати функцію затримки, яка відповідатиму часу однієї секунди;
3. Налаштувати виведення дати, часу та зміну часу;
4. Додати можливість використання тактових кнопок та світлодіодів;
5. Підключити датчик температури та визначити принцип його роботи;
6. Активувати АЦП мікроконтролера та налаштувати виведення температури;
7. Додати різні режими виведення інформації та забезпечити можливість їх зміни;
8. Додати функцію будильника і додати можливість встановлення часу спрацювання будильника ;
9. Реалізувати можливість налаштувань часу та дати;
10. Перевірити роботоздатність розумного годинника та всіх його функцій.

Блок схема програми

На основі складеного алгоритму роботи можна скласти приблизну блок-схему роботи розумного годинника:

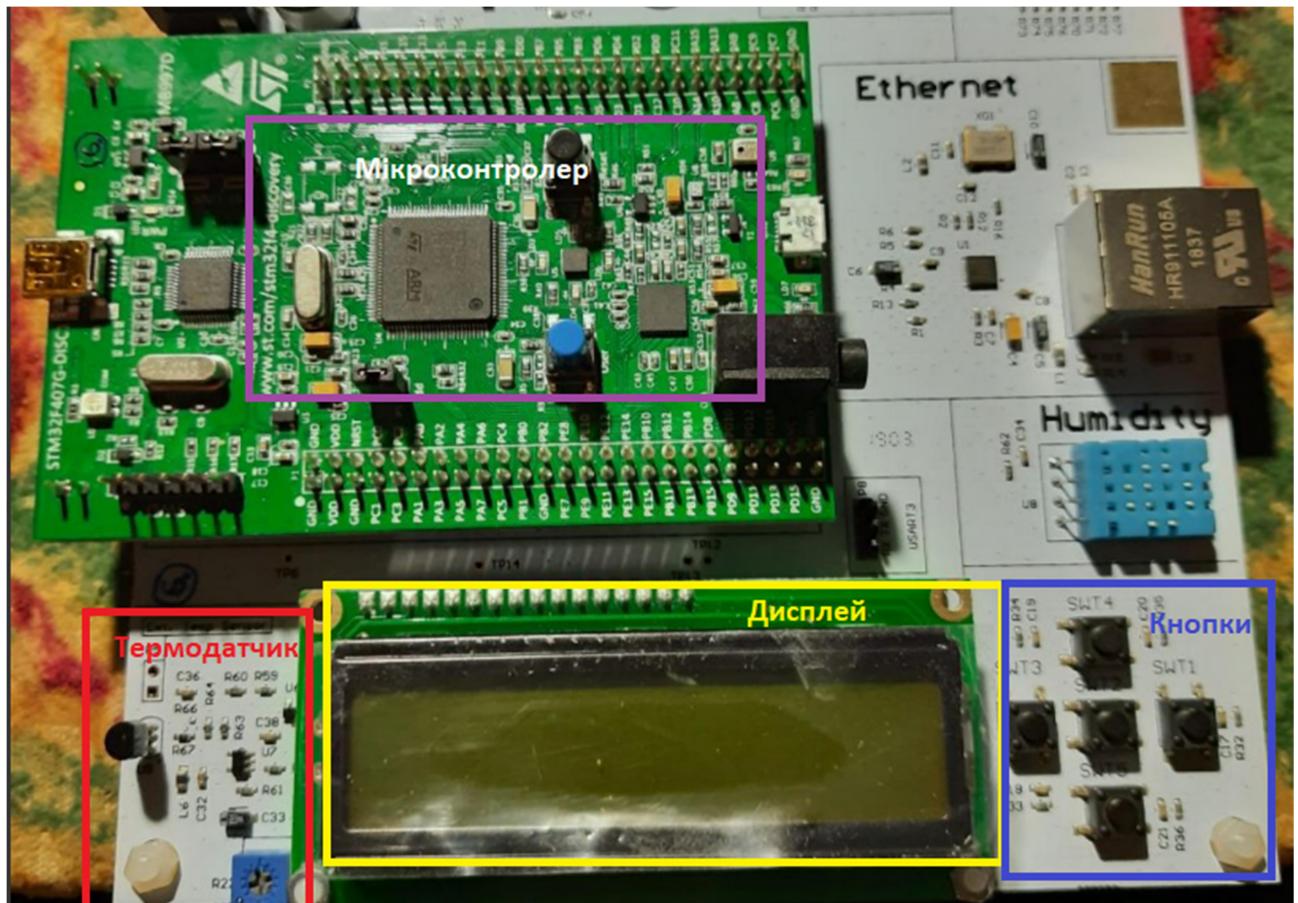


Хід роботи:

Спираючись на вище приведений алгоритм та технічне завдання, першим чином було взято плату STM32F407VG Discovery та плату Global Logic, з уже підключеними усіма необхідними блоками. Ми отримали наступний вигляд цих плат в зборі:



Можемо бачити, що всі необхідні блоки вже підключено:



```
    volatile uint32_t one_sec = 950;
    volatile uint32_t delay_cnt = 0;
```

One_sec – змінна яка відповідає одній секунді, і вміщує 950 мс. Було використано не 1000 мс, тому що перевірка if() та інші функції теж займають час, тому було компенсовано 50 мс. Таким чином приблизно відповідає реальній 1 секунді.

Потім було оголошено змінні, які відповідатимуть параметрам виведення на екран(деякі значення було прийнято за змовчуванням, спираючись на день розробки пристрою):

```
    volatile uint32_t second = 00;
    volatile uint32_t minute = 30;
    volatile uint32_t hour = 12;
    volatile uint32_t day = 05;
    volatile uint32_t month = 06;
    volatile uint32_t check_month = 00;
    volatile uint32_t year = 2020;
    double temperature = 00;
    volatile uint32_t alarm_h = 12;
    volatile uint32_t alarm_m = 20;
    volatile uint32_t style = 1;
    int adc_result;
```

Наступним кроком було ініціалізовано аналогово-цифровий перетворювач. Так як було відомо, що термодатчик приєднано до порту PB-1, то відповідно було активовано АЦП-1 мікроконтроллера і інформацію брали з дев'ятого каналу АЦП. Враховуючи ці дані було виконано ініціалізацію:

```
39 void adc_init(void) //ініціалізуємо АЦП 1 для зчитування напруги з термодатчика
40 {
41     RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1, ENABLE);
42     ADC_DeInit();
43
44     RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOB, ENABLE);
45     GPIO_InitTypeDef portb_setup;
46     GPIO_StructInit(&portb_setup);
47     portb_setup.GPIO_Mode = GPIO_Mode_AN;
48     portb_setup.GPIO_Pin = GPIO_Pin_1;
49     GPIO_Init(GPIOB, &portb_setup);
50
51     ADC_InitTypeDef adc_setup;
52     adc_setup.ADC_ContinuousConvMode = DISABLE;
53     adc_setup.ADC_Resolution = ADC_Resolution_12b;
54     adc_setup.ADC_ScanConvMode = DISABLE;
55     adc_setup.ADC_ExternalTrigConv = ADC_ExternalTrigConvEdge_None;
56     adc_setup.ADC_DataAlign = ADC_DataAlign_Right;
57
58     ADC_Init(ADC1, &adc_setup);
59     ADC_Cmd(ADC1, ENABLE);
60
61     GPIO_InitTypeDef port_setup;
62     RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOD, ENABLE);
63     port_setup.GPIO_Mode = GPIO_Mode_OUT;
64     port_setup.GPIO_OType = GPIO_OType_PP;
65     port_setup.GPIO_PuPd = GPIO_PuPd_DOWN;
66     port_setup.GPIO_Pin = GPIO_Pin_All;
67     port_setup.GPIO_Speed = GPIO_Speed_2MHz;
68     GPIO_Init(GPIOD, &port_setup);
69 }
70 }
```

Написали функцію для отримання даних з АЦП-1:

```
71 ul6 readADC1(u8 channel) // функція для зчитування даних з дев'ятого каналу АЦП
72 {
73     ADC_RegularChannelConfig(ADC1, channel, 1, ADC_SampleTime_3Cycles);
74     ADC_SoftwareStartConv(ADC1);
75     while(ADC_GetFlagStatus(ADC1, ADC_FLAG_EOC) == RESET);
76     return ADC_GetConversionValue(ADC1);
77 }
78 }
```

Наступним кроком звернулись до даташиту плати Global Logic, і дізнались наступну інформацію, а саме залежність вихідної напруги від температури навколошнього середовища:

Термосенсор

adc12_in9 - PB1

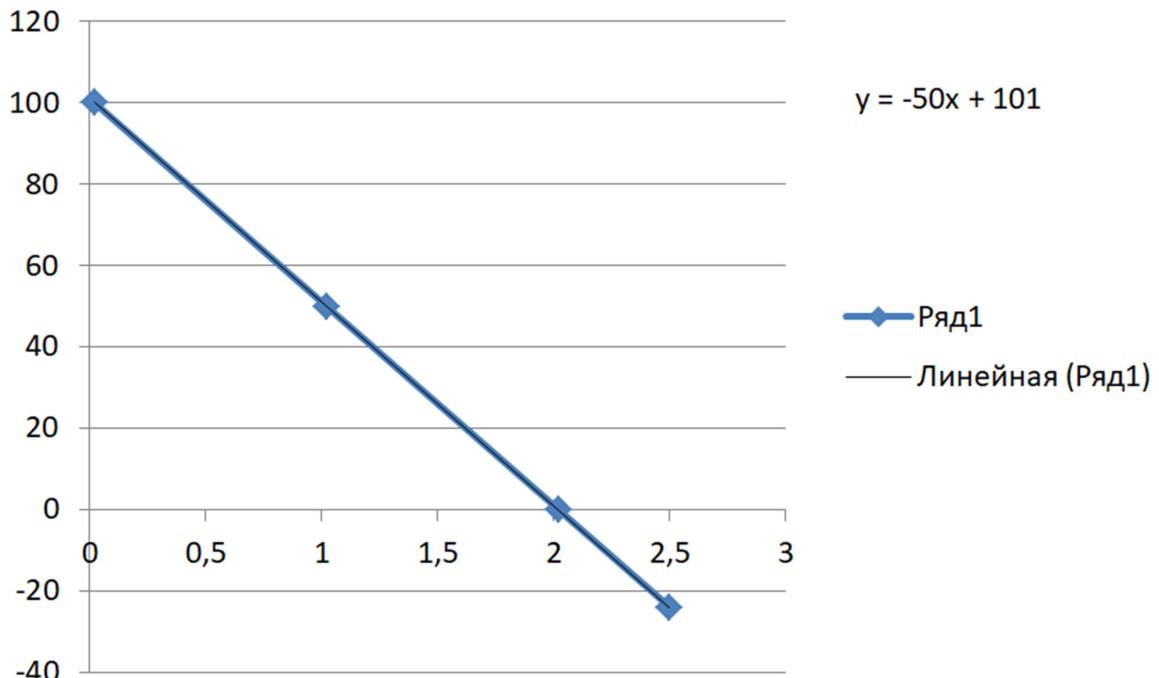
-24C = 2.5v

0C = 2.02v

50C = 1.02v

100C = 0.02v

На основі цього було побудовано залежність температури від напруги:



Потім було додано лінію тренда і виведено з неї формулу залежності температури від напруги на виході термодатчика. Використали цю інформацію при написанні коду і отримали наступне:

```
//визначаємо температуру
adc_result = readADC1(ADC_Channel_9);
temperature = -55*adc_result/2*0.0001+101;
```

Наступним кроком ініциалізували кнопки та світлодіоди(приведено частину коду):

```
void port_ini(void)
{
    GPIO_InitTypeDef InitC; //для кнопки
    GPIO_InitTypeDef InitD; //для світлодіодів

    RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOB, ENABLE); //включим такттування порта D

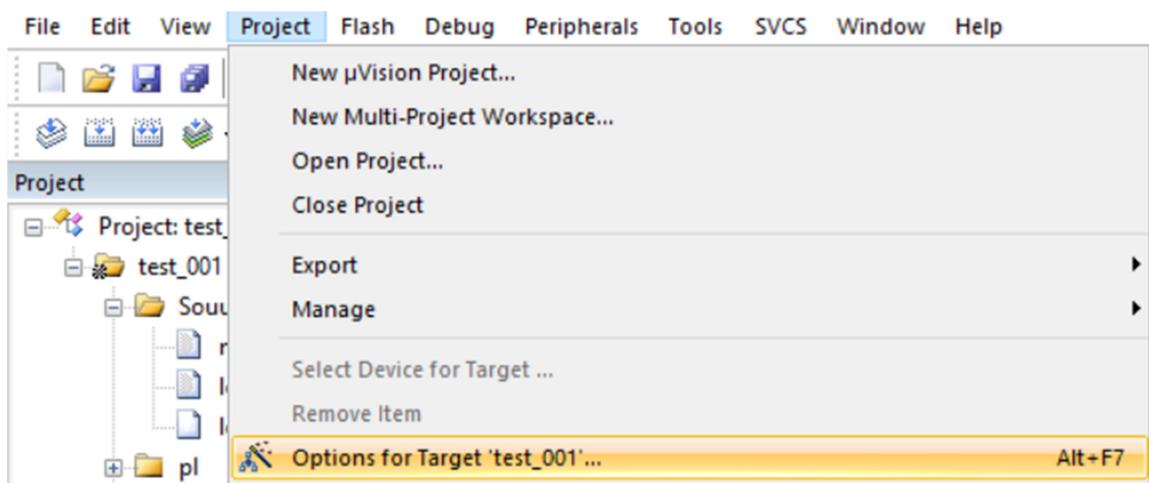
    //світлодіоди
    InitD.GPIO_Pin = GPIO_Pin_12|GPIO_Pin_13|GPIO_Pin_14|GPIO_Pin_15;
    InitD.GPIO_Mode = GPIO_Mode_OUT;
    InitD.GPIO_OType = GPIO_OType_PP;
    InitD.GPIO_Speed = GPIO_Speed_2MHz;
    InitD.GPIO_PuPd = GPIO_PuPd_NOPULL;
    GPIO_Init(GPIOB,&InitD); //ініціалізуємо ножки порта

    RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA, ENABLE); //включим тактирование порта C

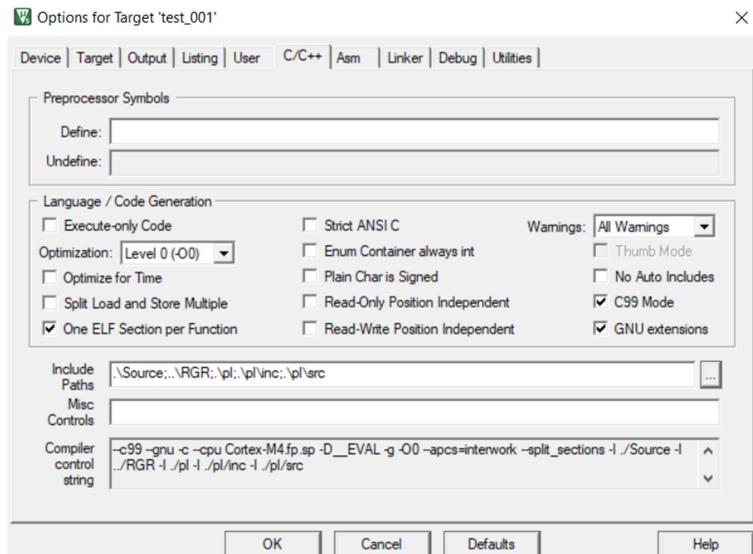
    //кнопка
    InitC.GPIO_Pin = GPIO_Pin_6;
    InitC.GPIO_Mode = GPIO_Mode_IN;
    InitC.GPIO_OType = GPIO_OType_OD;
    InitC.GPIO_Speed = GPIO_Speed_2MHz;
    InitC.GPIO_PuPd = GPIO_PuPd_NOPULL;
    GPIO_Init(GPIOA,&InitC); //ініціалізуємо ножки порта
```

Тепер у нас є все необхідне, щоб написати програму з необхідним функціоналом згідно технічного завдання, код програми та архів з проектом розташований за посиланням: [посилання](#).

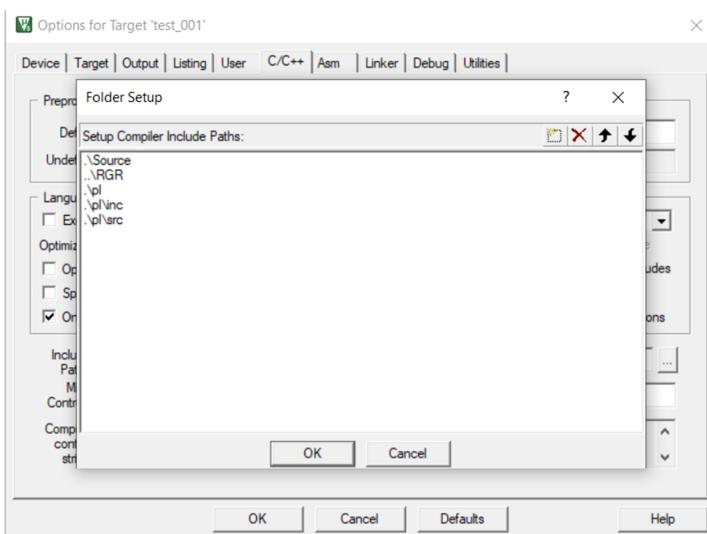
Хотілось би відмітити, що при завантаженні проекту, після його розархівування, для того щоб моливо було його скомпілювати, необхідно змінити шляхи до бібліотек проекту наступним чином:



Перейти до налаштувань проекту та відкрити пункт C/C++:



Відкрити пункт Include Path:



Додати шляхи до папок, назви яких приведені на скриншоті. Дані папки розташовані безпосередньо в самому проекті.

Після цього було скомпільовано проект та зашито в плату. Отримали наступні результати. Перший режим роботи:



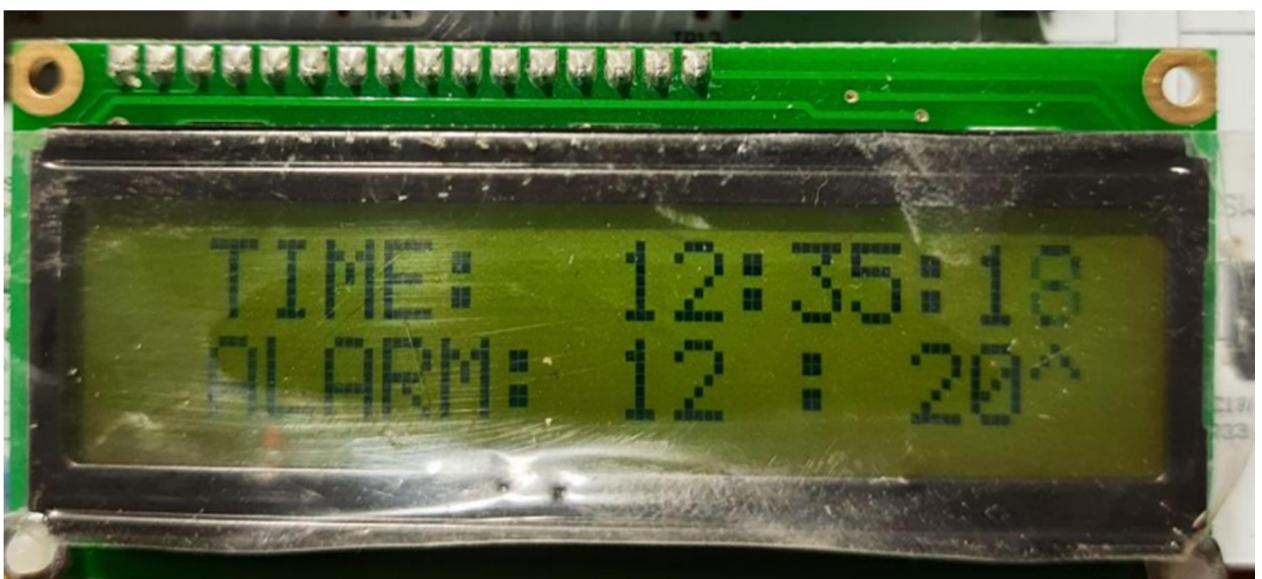
Другий режим роботи:



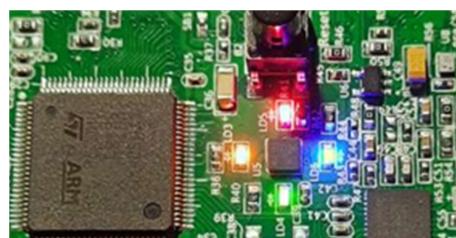
Третій режим роботи:



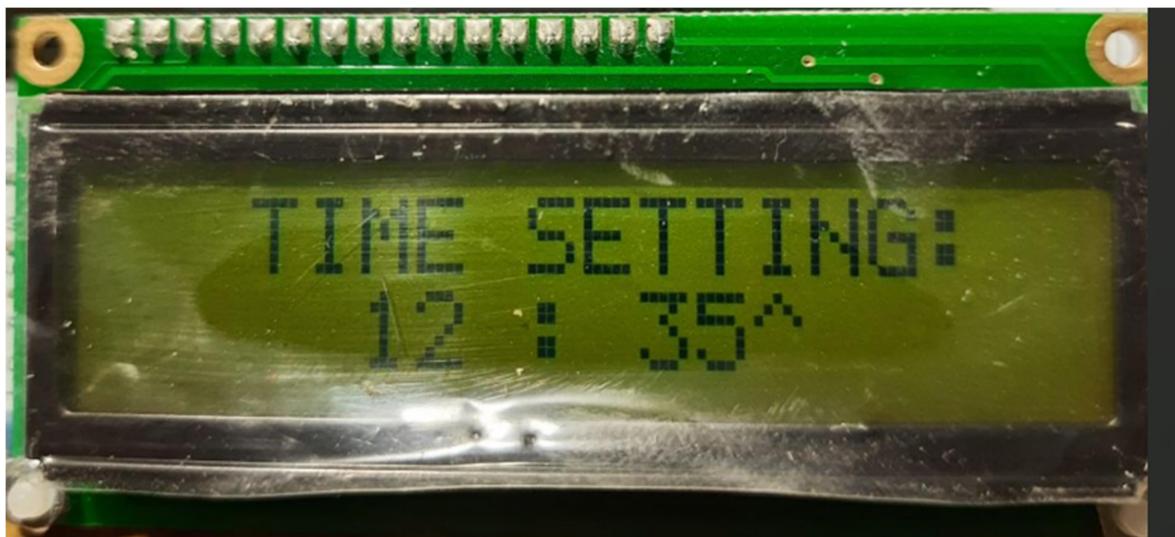
Налаштування будильника:



Спрацював будильник(почали мигати 4 світлодіоди):



Налаштування часу:



Налаштування дати:



Особливості програми при користуванні:

1. Будильник буде працювати протягом однієї хвилини;
2. Сигнал будильника – мигання чотирьох світлодіодів;
3. При включенні будь-якого режиму налаштувань, посекундно буде мигати два світлодіоди;
4. Курсор при налаштуванні показаний у вигляді знаку степеня «^»;
5. При натисканні будь якої кнопки, щоб вона спрацювала, необхідно її утримувати близько однієї секунди;
6. Для зміни стилю відображення застосовувати кнопки SWT5 та SWT4.
7. Для переходу до налаштувань будильника натиснути SWT2;

8. Для переходу до налаштувань часу натиснути SWT3;
9. Для переходу до налаштувань дати натиснути SWT1;
10. В будь-якому пункті налаштувань для переміщення курсора застосовувати SWT3 та SWT1;
11. В будь-якому пункті налаштувань для збільшення виділеного значення курсором застосовувати SWT4;
12. В будь-якому пункті налаштувань для зменшення виділеного значення курсором застосовувати SWT5;
12. В будь-якому пункті налаштувань для підтвердження введених даних та виходу з даного пункту налаштувань натиснути SWT2;
12. Термодатчик реагує на зміну температури не швидко, адже для нього необхідно, щоб температура його корпуса зрівнялась з температурою навколишнього середовища, а на це потрібен час;
13. При виведенні даних температури в якості значка градуса використовується значок степеню « \wedge »;
14. Програма не враховує високосний рік;

Перевірка коректності роботи:

Відео з тестом всіх модулів пристрою та перевіркою всіх функцій розташоване за [посилання](#).

Висновок:

На основі виконаної роботи, ми отримали програму, яка може відображати час, дату, температуру навколошнього середовища, може працювати як будильник, сповіщаючи користувача в заданий час, має зручний функціонал для налаштування. За рахунок використання сучасного мікроконтроллеру є доволі швидкою в роботі. За рахунок вбудованого системного таймеру, точно може змінювати час з плином часу. Завдяки вбудованому АЦП в мікроконтроллері програма зчитує з аналогового входу значення постійної напруги і, за допомогою математичних операцій і отриманої залежності з даташиту, переводить дане значення в температуру. Варто відмітити, що програма є частково розділена на модулі (бібліотека LCD) та функції (ініціалізація АЦП, світлодіодів, кнопок, функція затримки), в деяких моментах має корисні коментарі, а це в свою чергу значно спрощує розуміння та редактування, для подальших модернізацій, отриману програму. Тому в результаті, можна сказати, що було зроблено немалий крок в напрямку розвитку своїх умінь, у написанні програми на мові С, а також освоєно та закрілено роботу з аналогово-цифровим перетворювачем.