

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
РАДІОТЕХНІЧНИЙ ФАКУЛЬТЕТ
КАФЕДРА ПРИКЛАДНОЇ РАДІОЕЛЕКТРОНІКИ**

До захисту допущено:

В.о.зав. кафедри

_____ Михайло СТЕПАНОВ

«___» _____ 2022 р.

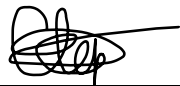
**Дипломний проєкт
на здобуття ступеня бакалавра
за освітньою-професійною програмою «Інтелектуальні технології мікро-
системної радіoeлектронної техніки»
спеціальності 172 Телекомунікації та радіотехніка
на тему: «Будильник з функцією імітації сходу сонця»**

Виконав:

студент IV курсу, групи РІ-81

Свереда Микола Вікторович

Прізвище, ім'я та по батькові



підпис

Керівник:

к.т.н., ст. викладач Зінгер Яна Леонідівна

Посада, науковий ступінь, вчене звання, Прізвище, ім'я та по батькові



підпис

Рецензент:

доцент, к.т.н. каф. РТС Піддубний Володимир

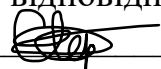
Олексійович

Посада, науковий ступінь, вчене звання, Прізвище, ім'я та по батькові



підпис

Засвідчую, що у цьому дипломному
проєкті немає запозичень з праць інших
авторів без відповідних посилань.

Студент _____


Київ – 2022 року

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЄКТУ

№ з/п	Формат	Позначення	Найменування	Кількість	При-мітка
1	A2	RI81.676831.002 ЕЗ	Схема електрична принципова ЕЗ	1	
2	A2	PI81.758746.001 СК	Складальний кресленик	1	
3	A2	PI81.758746.002 СК	Складальний кресленик	1	
4	A2	RI81.676831.001 ЕЗ	Схема електрична принципова ЕЗ	1	
5	A4	PI81.676831.001 ПЕ	Перелік елементів	2	
6	A4	PI81.676831.002 ПЕ	Перелік елементів	1	
7	A4	RI81.676831.002	Специфікація на електронний модуль індикації	1	
8	A4	RI81.676831.001	Специфікація на електронний модуль управління	1	
9	A3	PI81.758746.001	Креслення плати управління	1	
10	A3	PI81.758746.002	Креслення плати індикації	1	

				PI81.464419.001		
	ПІБ	Підп.	Дата	Відомість дипломного проєкту		
Розробн.						
Керівн.						
Консульт.						
Н/контр.						
Зав.каф.						
					Лист	Листів
					1	1
				КПІ ім. Ігоря Сікорського Каф.ПРЕ, Гр. PI-81		

**Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»**

Радіотехнічний факультет

Кафедра прикладної радіоелектроніки

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 172 Телекомунікації та радіотехніка

Освітньо-професійна програма «Інтелектуальні технології мікросистемної радіоелектронної техніки»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Михайло СТЕПАНОВ

«__» _____ 20__ р.

ЗАВДАННЯ

на дипломний проєкт студенту

Свереди Миколи Вікторовича

1. Тема проєкту «Будильник з функцією імітації сходу сонця», керівник проєкту Зінгер Яна Леонідівна, к.т.н., старший викладач, затверджені наказом по університету від «01»червня 2022 р. № 822-с

2. Термін подання студентом проєкту 09 червня 2022 року

3. Вихідні дані до проєкту Діапазон напруги живлення 4,5 — 5 В, бездротове з'єднання по мережі WI-FI, робочий діапазон відносної вологості 0 — 95 %, діапазон робочих температур 0 — 40 °С.

4. Зміст пояснювальної записки Вступ, Розгляд аналогів, Створення структурної схеми приладу, Створення схеми електричної принципової, Створення корпусу, Створення макету, Визначення габаритів друкованої плати, Трасування провідників, Виготовлення друкованого вузла, Тестування.

5. Перелік графічного матеріалу. Схеми електричні принципові, креслення друкованих плат, друкованого вузла, плакат.

6. Дата видачі завдання 01 травня 2022 року

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Ознайомлення з темою, пошук попередньої інформації	02.05 – 03.05	
2	Пошук аналогів, їхній аналіз	03.05 – 04.05	
3	Створення структурної схеми приладу	04.05 – 05.05	
4	Створення схеми принципової	05.05 – 06.05	
5	Аналіз та підбір елементної бази	06.05 – 09.05	
6	Назначення корпусів елементів	09.05 – 10.05	
7	Трасування плати	10.05 – 11.05	
8	Замовлення плати на виробництві	11.05 – 12.05	
9	Пошук і покупка необхідних компонентів	12.05 – 13.05	
10	Отримання плати і монтаж компонентів на плату	13.05 – 16.05	
11	Повна збірка прототипу і перевірка якості	16.05 – 17.05	
12	Написання програми і прошивка прототипу	17.05 – 18.05	
13	Тестування прототипу	18.05 – 19.05	
14	Оформлення текстової та графічної документації	19.05 – 12.06	

Студент

Микола СВЕРЕДА

Керівник

Яна ЗІНГЕР

АНОТАЦІЯ

Дипломний проект складається з пояснювальної записки обсягом 36 сторінок, що містить 37 ілюстрацій, 2 таблиць, 1 креслень, 6 додатків та 16 посилань.

Метою даного дипломного проекту є розробка будильника світлового типу. Основним призначенням даного приладу є допомога в правильному пробудженні людини. У проекті був проведений огляд аналогів, враховуючі їх переваги та недоліки було спроектовано наш прилад. Було синтезовано схему електричну принципову, розроблено друковану плату та конструкцію. Був виготовлений макет приладу. А також спроектований корпус для макету. Після написання програмного забезпечення були проведені тестування які підтвердили його ефективність.

Ключові слова: будильник, світло, сигнал, пробудження, біохакинг.

ANNOTATION

The diploma project consists of an explanatory note of 36 pages, containing 37 illustrations, 2 tables, 1 drawings, 6 appendices and 16 references.

The purpose of this diploma project is to develop a light-type alarm clock. The main purpose of this device is to help in the proper awakening of man. The project reviewed analogues, taking into account their advantages and disadvantages, our device was designed. The electrical circuit diagram was synthesized, the printed circuit board and the design were developed. A model of the device was made. And also the designed case for the model. After writing the software, tests were conducted to confirm its effectiveness.

Key words: alarm clock, light, signal, awakening, biohacking.

ПОЯСНЮВАЛЬНА ЗАПИСКА
до дипломного проекту

на тему: Будильник з функцією імітації сходу сонця.

Київ — 2022 року

ЗМІСТ

Перелік скорочень.....	2
Вступ.....	3
1 Огляд існуючих рішень.	4
2 Огляд та аналіз схемотехнічних рішень	13
3 Синтез схеми.....	15
3.1 Розробка структурної схеми	15
3.2 Вибір та обґрунтування елементної бази	16
3.3 Створення схеми електричної принципової	20
4 Проектування приладу	22
4.1 Проектування електронного модуля	22
4.1.1 Обґрунтування методу виготовлення друкованої плати	22
4.1.2 Обґрунтування вибору матеріалу плати.....	22
4.1.3 Обґрунтування вибору класу точності плати	22
4.2 Робота в редакторі Altium PCB	23
4.2.1 Визначення габаритів друкованої плати	23
4.2.2 Трасування провідників	25
4.3 Проектування корпусу	26
5 Макетування та тестування.....	29
Висновки	32
Перелік джерел посилань	34
Додаток А Перелік елементів до схеми модуля індикації.....	36
Додаток Б Перелік елементів до схеми модуля управління.....	37

					PI81.464419.001 ПЗ						
ЗМ.	Лист	№ докум.	Підпис	Дата	Будильник світлового типу			Лім.	Лист	Листів	
Розробив	Свереда М.В.									1	
Перевірів	Зінгер Я.Л.										
Н. Контр.											
Затвердив	Степанов М.				PI-81 РТФ						

Додаток В Лістинг розрахунку ширини провідника	39
Додаток Г Специфікація на електронний модуль індикації.....	41
Додаток Д Специфікація на електронний модуль управління.....	42
Додаток Е Лістинг програми	43

					PI81.464419.001 ПЗ				
ЗМ.	Лист	№ докум.	Підпис	Дата	Будильник світлового типу	Лім.	Лист	Листів	
Розробив		Свереда М.В.					1		
Перевірів		Зінгер Я.Л.							
						PI-81 РТФ			
Н. Контр.									
Затвердив		Степанов М.							

ПЕРЕЛІК СКОРОЧЕНЬ

ДП — Друкована плата

ДФ — Додаткові функції

ШІМ — Широтно-імпульсна модуляція

SMD — surface mounted device

WIFI — Wireless Fidelity

					<i>PI81.464419.001 ПЗ</i>	Лист
						2
Зм.	Лис	№ докум.	Підпис	Дата		

ВСТУП

Під час сну зорові рецептори реагують на рівень освітленості та відсилають сигнали в мозок, який керує виробленням гормонів, необхідних для сну та пробудження. Під ранок, разом із світанком, організм перестає виробляти гормон сну, а рівень гормону бадьорості підвищується.

В результаті такого налаштування організму ви поступово прокидаєтеся і починаєте ранок без сонливості та поганого настрою, і ранок задає темп як мінімум першій половині дня. Взимку та восени кількість світла вранці значно знижується і біоритми організму не відповідають нашому робочому графіку.

Виходить, щоб забезпечити собі бадьорість вранці, потрібно більше світла. Але просто увімкнути яскраве світло після пробудження - не найкращий вихід. Ваш організм ще не підготувався до пробудження, ви піддали його стресу від звуку будильника, а потім ще й різко увімкнули світло після повної темряви.

Вихід один – світло потрібно включати плавно ще до того, як почати прокидатися.

У даній роботі розроблено конструкторську документацію на виготовлення будильника з функцією імітації сходу сонця. Даний прилад призначений для поліпшення пробудження людини в умовах поганого (відсутнього) освітлення в кімнаті або внаслідок поганих погодних умов.

					<i>PI81.464419.001 ПЗ</i>	Лист
						3
Зм.	Лис	№ докум.	Підпис	Дата		

1 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ.

На сьогодні існує не багато моделей будильників світлового типу, більшість з них мало відомих брендів. Далі розглянуто їх будову і можливості а також технічні характеристики.

Будильник «Hatch Restore», який зображено на рис. 1.1 [1].



Рисунок 1.1 — Будильник «Hatch Restore»

Цей прилад має такі властивості:

- Максимальна яскравість: —
- Додаткові функції (ДФ): —
- Управління: додаток, кнопки
- Живлення: 220 В зовнішній блок живлення.
- Габарити: 89x180x220 мм.
- Маса: 950 г.
- Ціна: 5600 гривень.

Зі всіх розглянутих варіантів ця модель виділяється тим що її корпус покритий тканиною а також для її використання необхідно поновлювати платну підписку в програмі. В переваги хочу додати унікальний дизайн і матовий пластик розсіювача який не буде збирати відбитки пальців. Головний недолік це доволі висока ціна і платна підписка.

					PI81.464419.001 ПЗ	Лист
						4
Зм.	Лис	№ докум.	Підпис	Дата		

Будильник «NOKU SUNSHINE», який зображено на рис. 1.2 [2].



Рисунок 1.2 — Будильник «NOKU SUNSHINE»

Цей прилад має такі властивості:

- Максимальна яскравість: 150 люкс.
- ДФ: радіо.
- Управління: додаток, кнопки, голосове.
- Живлення: зовнішній блок живлення, вбудований акумулятор.
- Габарити: 110x186x186 мм.
- Маса: 320 г.
- Ціна: 1600 гривень.

Одна з найпростіших моделей з найбільш стандартним для цього типу приладів дизайном. При низькій ціні має майже такі ж можливості щой дорожчі моделі а місцями даже кращі. Із мінусів хочу відмітити глянцекий пластик який в поєднанні з сенсорним управлінням почне з часом збирати бруд.

					PI81.464419.001 ПЗ	Лист
						5
Зм.	Лис	№ докум.	Підпис	Дата		

Будильник «Sunrise», який зображено на рис. 1.3 [3].



Рисунок 1.3 — Будильник «Sunrise»

Цей прилад має такі властивості:

- Максимальна яскравість: 100 люкс.
- ДФ: радіо.
- Управління: кнопки
- Живлення: 220 В зовнішній блок живлення, батарейкі.
- Габарити: 130x185x185 мм.
- Маса: 290 г.
- Ціна: 1500 гривень.

Дана модель майже нічим не виділяється окрім можливості житись від батарейок що дуже непрактично і неекологічно.

					PI81.464419.001 ПЗ	Лист
						6
Зм.	Лис	№ докум.	Підпис	Дата		

Будильник «TITIROBA MY-10», який зображено на рис. 1.4 [4].



Рисунок 1.4 — Будильник «TITIROBA MY-10»

Цей прилад має такі властивості:

- Максимальна яскравість: 50 люкс.
- ДФ: —
- Управління: кнопки
- Живлення: 220 В зовнішній блок живлення.
- Габарити: 80x80x190 мм.
- Маса: 380 г.
- Ціна: 1300 гривень.

Чудо китайської інженерії. Будильник зроблений на основі лампи нічника. Має доволі примітивні характеристики і є самою дешевою моделлю на ринку.

					<i>PI81.464419.001 ПЗ</i>	Лист
						7
Зм.	Лист	№ докум.	Підпис	Дата		

Будильник «Beurer WL 50», який зображено на рис. 1.5 [5].

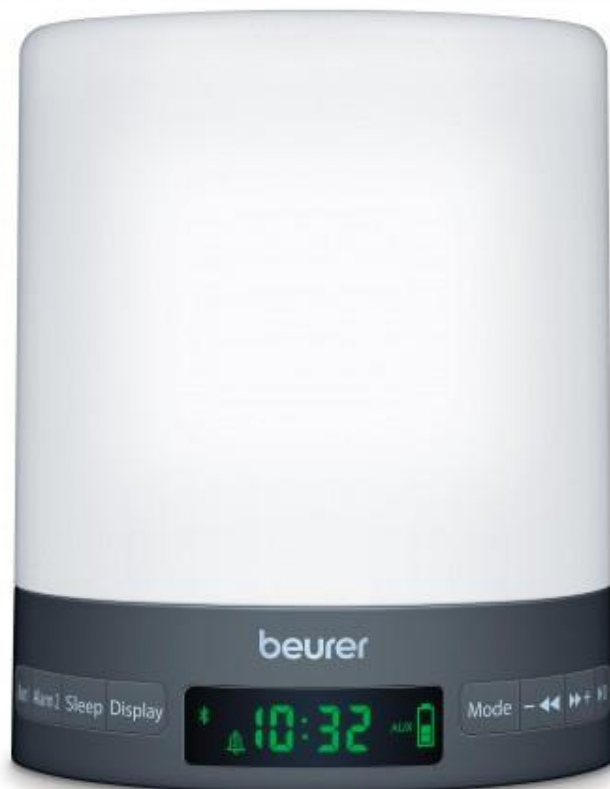


Рисунок 1.5 — Будильник «Beurer WL 50»

Цей прилад має такі властивості:

- Максимальна яскравість: 150 люкс.
- ДФ: радіо.
- Управління: кнопки
- Живлення: 220 В зовнішній блок живлення.
- Габарити: 130x195x195 мм.
- Маса: 820 г.
- Ціна: 2900 гривень.

Ще одна модель зроблена на подібні нічної лампи. Має всенаправлений розсіювач і можливість використання як безпроводний гучномовець.

					PI81.464419.001 ПЗ	Лист
						8
Зм.	Лис	№ докум.	Підпис	Дата		

Будильник «Beurer WL 75», який зображено на рис. 1.6 [6].



Рисунок 1.6 — Будильник «Beurer WL 75»

Цей прилад має такі властивості:

- Максимальна яскравість: 2000 люкс.
- ДФ: —
- Управління: додаток, кнопки
- Живлення: 220 В зовнішній блок живлення.
- Габарити: 225x185x95 мм.
- Маса: 1150 г.
- Ціна: 4400 гривень.

Сама потужна по яскравості модель на ринку. Її яскравості хватить не тільки для того щоб вас розбудити і освітити всю кімнату а ще й для того щоб отримувати засмагу не вилазячі з ліжка)

					PI81.464419.001 ПЗ	Лист
						9
Зм.	Лис	№ докум.	Підпис	Дата		

Будильник «Philips HF3651», який зображено на рис. 1.7 [7].



Рисунок 1.7 — Будильник «Philips Wake-up Light HF3651»

Цей прилад має такі властивості:

- Максимальна яскравість: 150 люкс.
- ДФ: радіо.
- Управління: додаток, кнопки
- Живлення: 220 В зовнішній блок живлення.
- Габарити: 120x190x190 мм.
- Маса: 960 г.
- Ціна: 9400 гривень.

Найдорожчий будильник зі всіх мою найдених. Впевнений всі кошти пішли на дизайн бо нічого унікального в характеристиках не було помічено.

					PI81.464419.001 ПЗ	Лист
						10
Зм.	Лис	№ докум.	Підпис	Дата		

Будильник «Philips HF3505», який зображено на рис. 1.8 [8].



Рисунок 1.8 — Будильник «Philips Wake-up HF3505»

Цей прилад має такі властивості:

- Максимальна яскравість: 200 люкс.
- ДФ: радіо.
- Управління: додаток, кнопки
- Живлення: 220 В зовнішній блок живлення.
- Габарити: 120x180x180 мм.
- Маса: 760 г.
- Ціна: 7200 гривень.

Доволі непогана модель виготовлена з якісних матеріалів і з нормальними характеристиками. З мінусів можна відмітити тільки високу ціну.

					PI81.464419.001 ПЗ	Лист
						11
Зм.	Лис	№ докум.	Підпис	Дата		

Таблиця 1.1 – Порівняння загальних властивостей

	Люкси	ДФ.	Управління	Ціна(грн)
Hatch Restore	—	—	додаток, кнопки	5600
NOKU SUNSHINE	150	радіо	додаток, кнопки, голос	1600
Sunrise	100	радіо	кнопки	1500
TITIROBA MY-10	50	—	кнопки	1300
Beurer WL 50	150	радіо	кнопки	2900
Beurer WL 75	2000	—	додаток, кнопки	4400
Philips Light HF3651	150	радіо	додаток, кнопки	9400
Philips HF3505	200	радіо	додаток, кнопки	7200

З переваг та недоліків вище зазначених приладів можна зробити висновки:

Середнє значення світлової потужності дорівнює 150 люксам цього більшим достатньо щоб розбудити людину. Для зручнішого налаштування прилад має мати можливість налаштовуватись з програми на смартфоні. Також для практичності і зручності розсіювач має бути матовим і односторонньо напруженим. І для конкурентоспроможності на ринку має вкладуватись в ціновий діапазон від 1500 до 2500 гривень.

2 ОГЛЯД ТА АНАЛІЗ СХЕМОТЕХНІЧНИХ РІШЕНЬ

Оскільки на ринку не дуже багато приладів такого типу їхні схеми доволі подібні і відрізняються незначно. Розглянемо декілька для прикладу.

На рис. 2.1 зображена схема будильника на основі мікросхеми ATmega168 налаштування такого будильника здійснюється за допомогою блютуз модуля HC-05 а елементом освітлення виступає світлодіодна стрічка яка керується через мосфет транзистори за допомогою ШІМ сигналу.

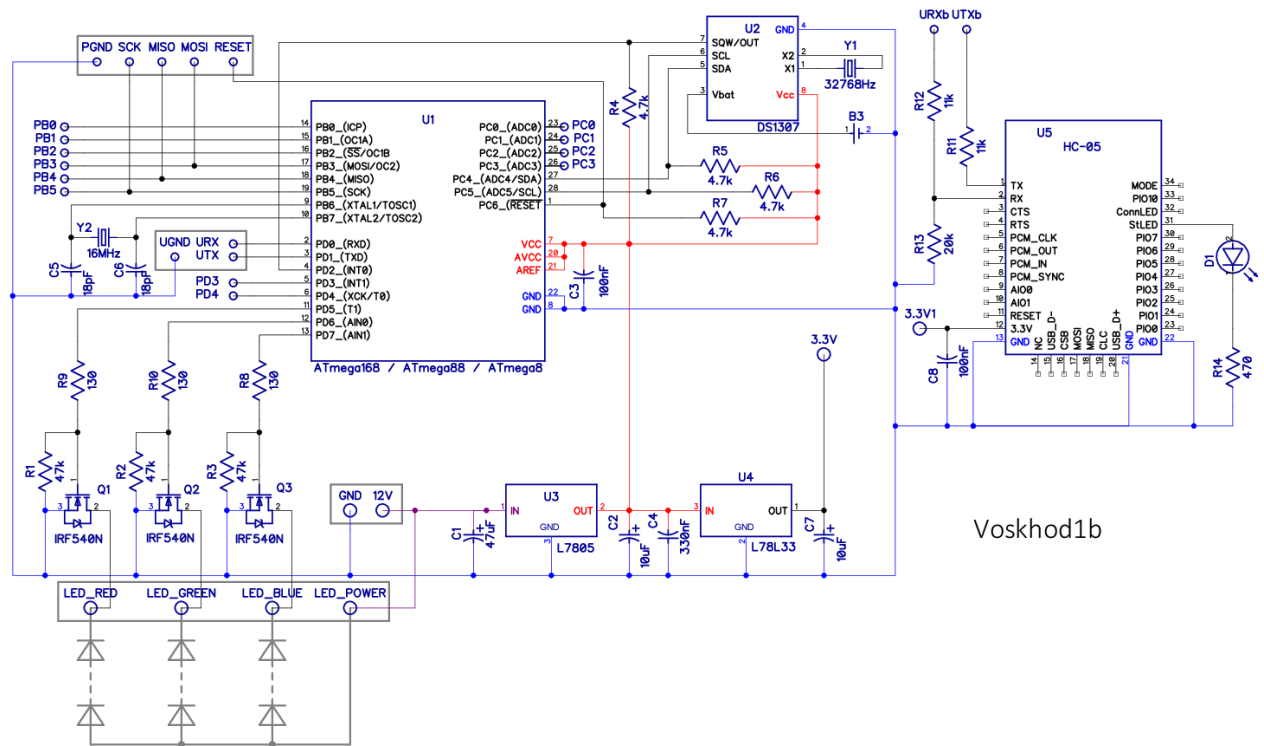


Рисунок 2.1 — Принципова схема будильника [9].

Основні недоліки даної схеми це сильно ускладнена компонентна база. Використання додаткового модулю Bluetooth, для світлової індикації використовується трьохкольорова LED стрічка яка керується трьома мосфет транзисторами, за рахунок чого збільшується габаритність схеми та вартість самої компонентної бази необхідної для виготовлення пристрою.

Розглянемо другий варіант схеми, наведений на рис. 2.2.

Зм.	Лис	№ докум.	Підпис	Дата

PI81.464419.001 ПЗ

Листм

13

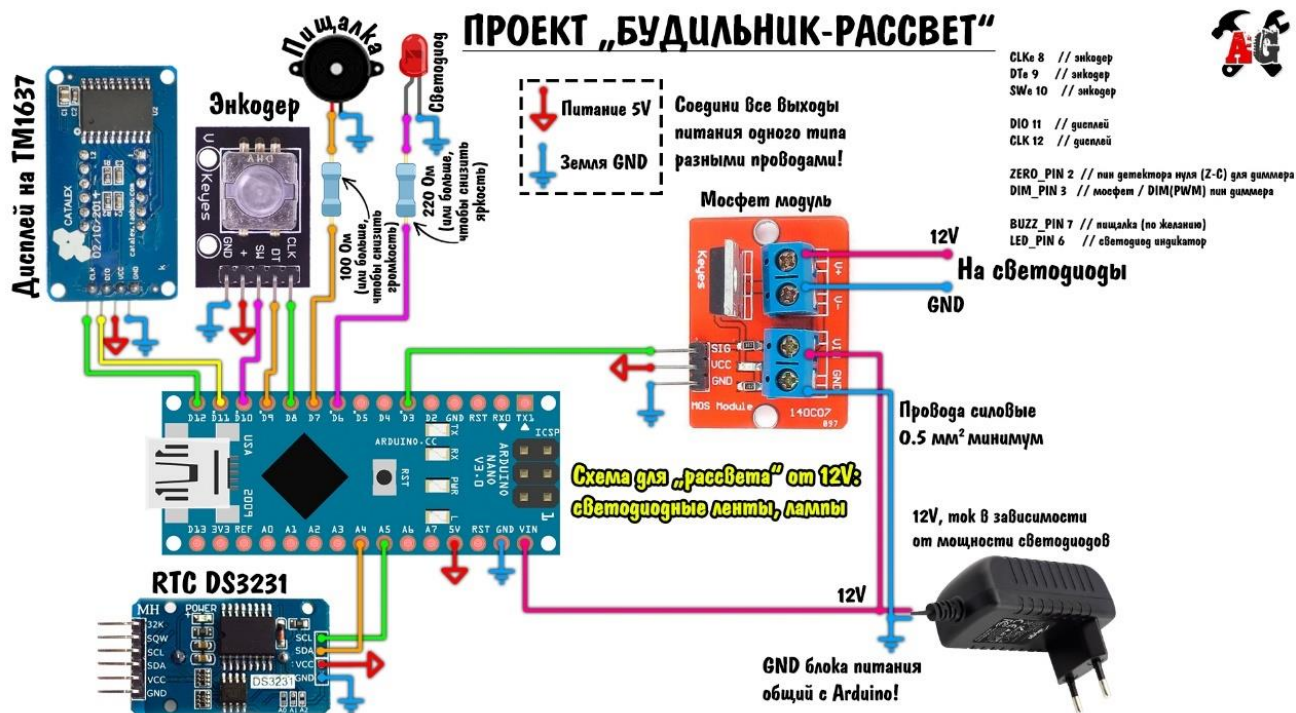


Рисунок 2.2 — Принципова схема будильника [10].

Схема на рис. 2.2 схожа на попередню (рис. 2.1), але тут використовуються готові модулі що збільшує її габарити і собі вартість. Також ця схема має восьми сегментний дисплей для відтворення реального часу і немає блютуз модуля тому всі налаштування здійснюються через енкодер.

Проаналізувавши ці дві схеми було вирішино сконструювати свій прилад на основі мікроконтролера ESP8266 який має встроєний WIFI модуль і дозволяє написання більш складних програм. Керування будильником буде здійснюватись через WEB інтерфейс і вам не потрібен буде інтернет або додаткові програми для його налаштування. Також на корпусі будуть передбачені кнопки для "класичного" налаштування. Щоб зменшити габарити пристрою вирішено використовувати одну матрицю і для індикації часу і для освітлення під час пробудження.

Зм.	Лис	№ докум.	Підпис	Дата

PI81.464419.001 ПЗ

Лист

14

3 СИНТЕЗ СХЕМИ

3.1 Розробка структурної схеми

Проаналізувавши готові схемотехнічні рішення, зваживши всі переваги та недоліки була запропонована структурна схема яка зображена на рис. 3.1

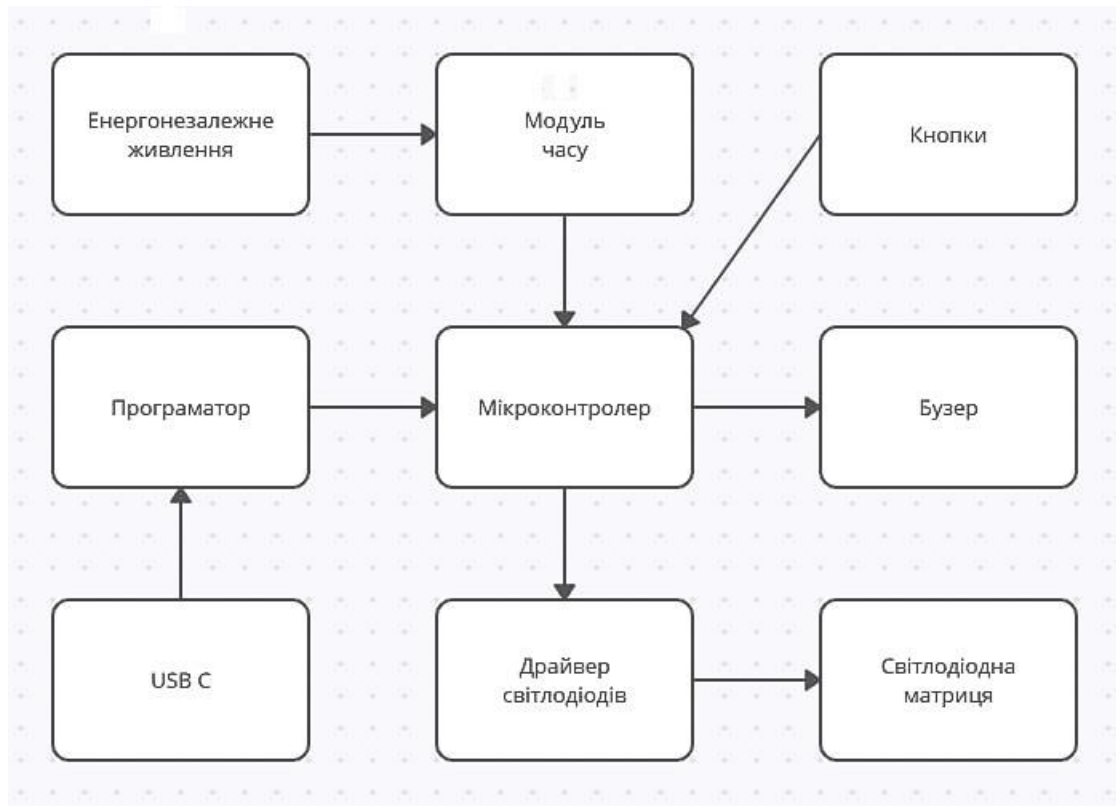


Рисунок 3.1 — Структурна схема приладу.

Для управління всім приладом потрібен мікроконтролер який за допомогою вмонтованого програматора можна бистро перепрошивати через USB C роз'єм. За допомогою кнопок користувач зможе налаштовувати будильник. Також для більш точного визначення часу буде встановлений модуль часу який буде мати резервне живлення, це зроблено для того щоб непотрібно було налаштовувати час заново після кожного відключення приладу від мережі живлення. Мікроконтролер буде виводити візуальну і звукову інформацію за допомогою бужера і світлодіодної матриці, але оскільки в контролера замало виводів для управління матрицею буде використаний світлодіодний драйвер який вирішить цю проблему.

Зм.	Лис	№ докум.	Підпис	Дата

PI81.464419.001 ПЗ

Лист
15

3.2 Вибір та обґрунтування елементної бази

Вибір елементної бази ґрунтувався на двох критеріях: сучасність та доступність, всі резистори та неполярні конденсатори взяті розміром 1206. Даний розмір був обраний через те що це мінімальний можливий розмір неполярних конденсаторів що необхідні в приладі.

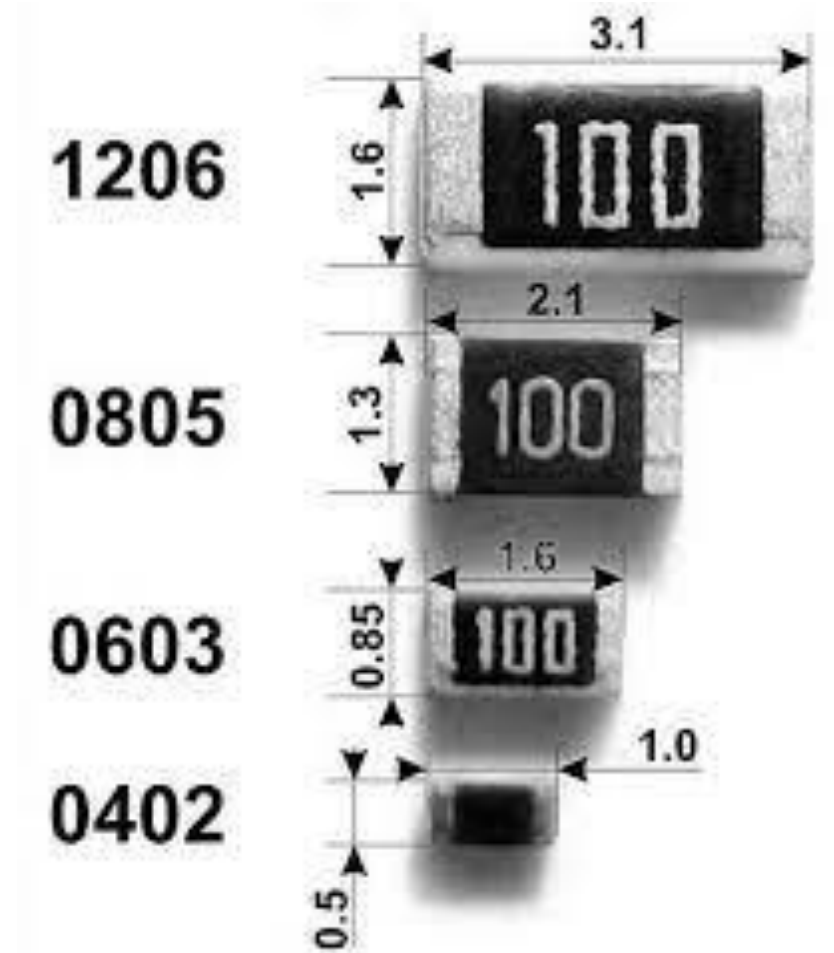


Рисунок 3.2 — Існуючі форм-фактори SMD компонентів.

Всі можливі компоненти взяті в SMD корпусі для спрощення і здешевлення виробництва. В наступній версії приладу планується використовувати всі без винятку компоненти типу SMD і зменшити розмір матриці на $\frac{1}{5}$ і постараюсь розмістити всі компоненти на одній платі зробивши її чотирьохслоїною.

Основа блока управління — мікроконтролер ESP8266.

Зм.	Лис	№ докум.	Підпис	Дата

PI81.464419.001 ПЗ

Лист
16

ESP-12E PINOUT

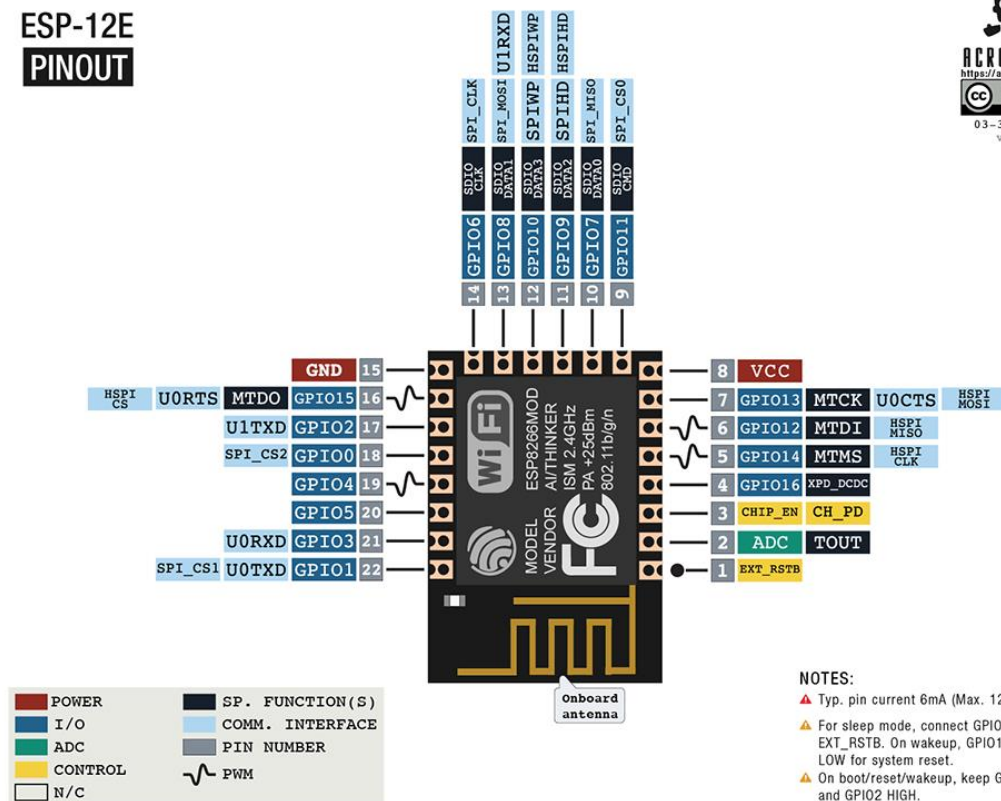


Рисунок 3.3 — Мікроконтролер ESP8266(12E).

Він був обраний через його розмір, доступність і можливості. Оскільки він має вбудований WIFI не потрібно буде доставляти додатковий модуль для підтримання зв'язку між будильником і інтернетом. Також він має достатню кількість контактів щоб можна було підключити всі інші модулі. Він використовується для передачі інформації на вивідну матрицю через мікросхеми MAX7219.



Рисунок 3.4 — Led матриця 8 на 8 модель 1088AS.

Зм.	Лис	№ докум.	Підпис	Дата

PI81.464419.001 ПЗ

Лист
17

Було вирішино встановити готові матриці оскільки робити їх самому більш затратно і ускладнюється спосіб виготовлення.



Рисунок 3.5 — Мікросхема MAX7219 в SMD корпусі.

Ця мікросхема була спеціально розроблена для керування невеликими Led дисплеями і вона ідеально нам підходить .

В якості мікросхеми реального часу була обрана DS3231M яка є енерго-незалежною і збереже час навіть якщо прилад буде обесточеним.

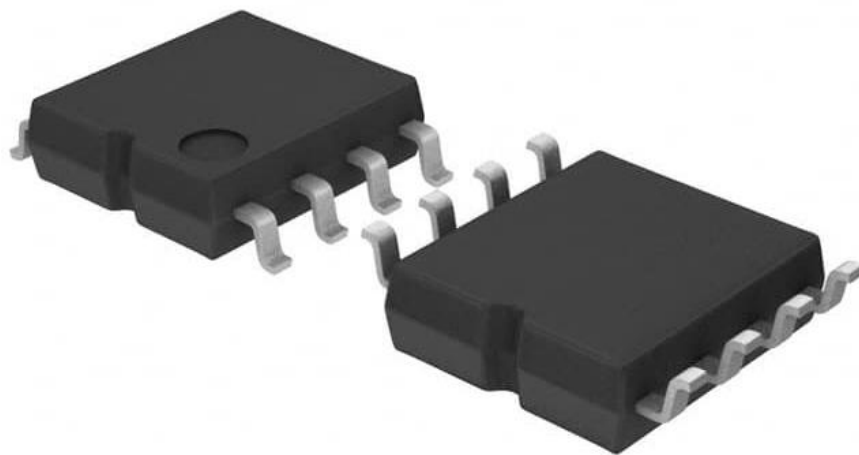


Рисунок 3.6 — Мікросхема DS3231M в SMD корпусі.

Для програмування мікроконтролера буде використовуватись програма-тор СН340.

					<i>PI81.464419.001 ПЗ</i>	Лист
						18
Зм.	Лис	№ докум.	Підпис	Дата		

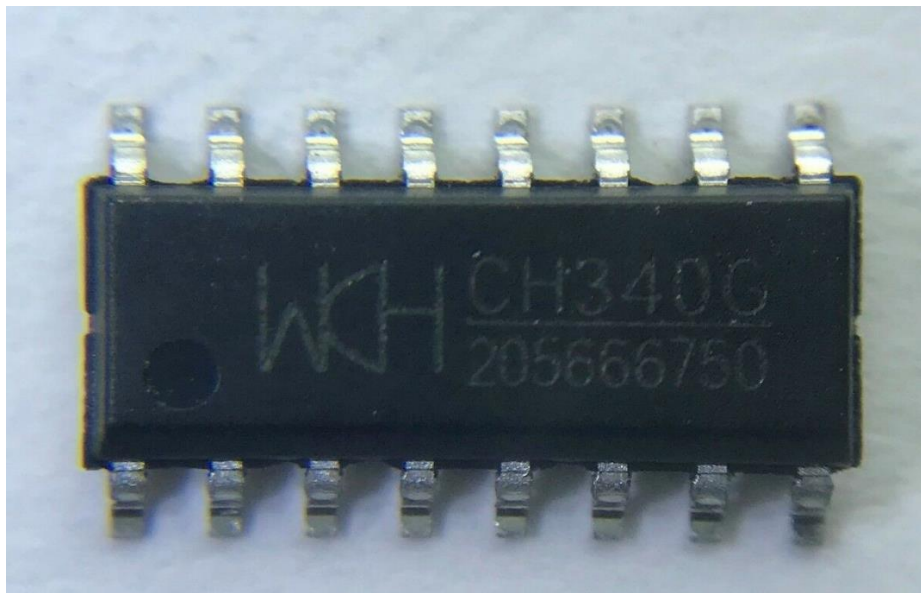


Рисунок 3.7 — Мікросхема CH340 в SMD корпусі.

Даний програматор був обраний через його малі габарити, невимогливість до живлення і доступність.

Живиться блок управління від джерела з напругою 5 В. Необхідна для живлення контролера напруга 3,3 В формує лінійний стабілізатор *ams-1117*.



Рисунок 3.8 — Стабілізатор *ams-1117* в SMD корпусі.

Оскільки живлення 3.3 В використовує тільки логічна частина нам непотрібний високий струм і такого маленького і недорогого стабілізатора буде вистачати.

3.3 Створення схеми електричної принципової

На основі обраної елементної бази та структурної схеми синтезовані наступні схеми електричні принципові.

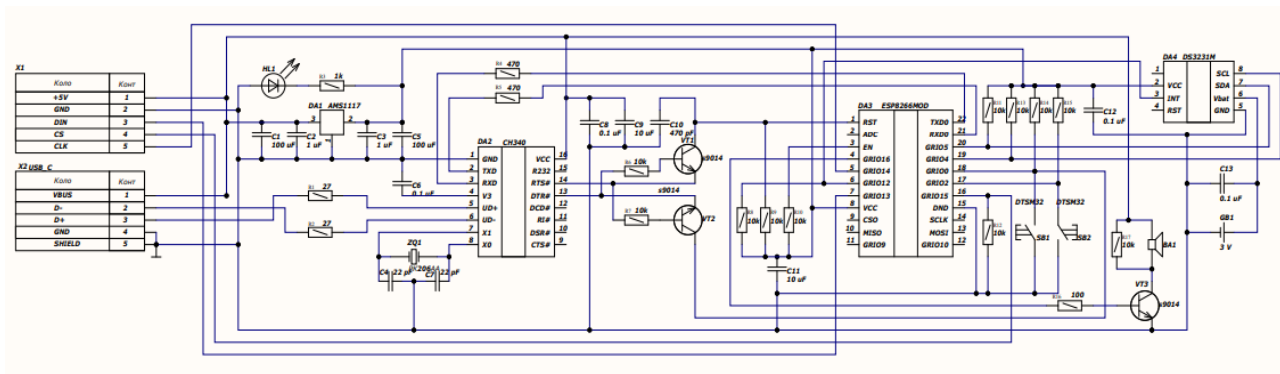


Рисунок 3.9 — Електрично принципова схема першої плати.

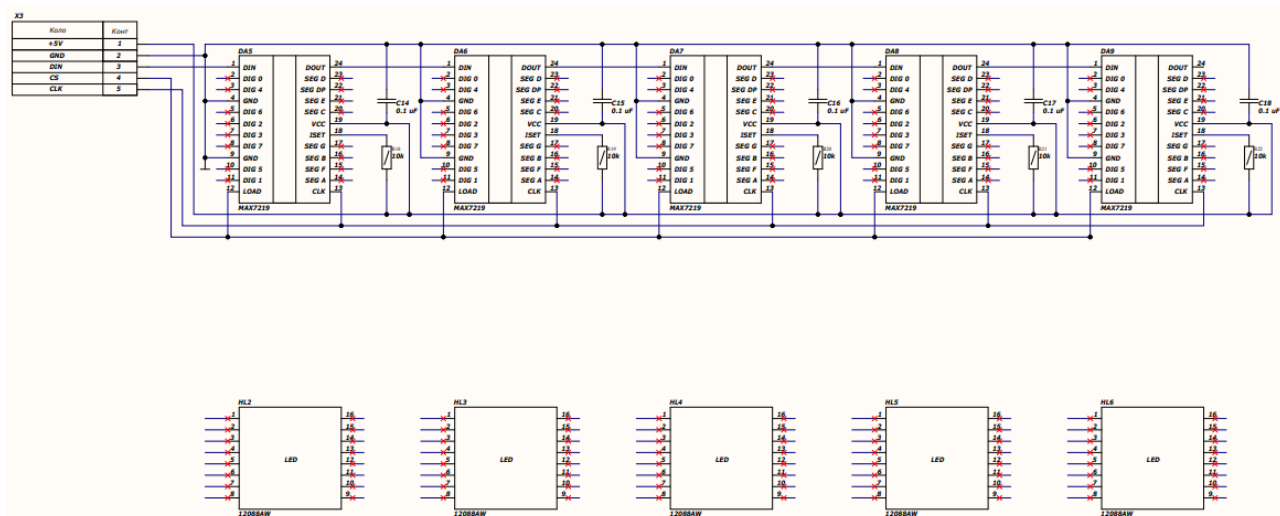


Рисунок 3.10 — Електрично принципова схема другої плати.

Для зручності тестування прототипу було прийнято рішення розділити прилад на дві плати. В наступній версії приладу планується зробити його одноплатним меншим і без встроєного програматора.

Основа блока управління — мікроконтролер ESP8266. Його порти GRIO13-GRIO15 використовуються для передачі інформації на вивідну матрицю HL2-HL6 через мікросхеми MAX7219(DA5-DA9). Порти GRIO12, GRIO4, GRIO5 підключений до мікросхеми реального часу DS3231M(DA4) яка є енергонезалежною і збереже час навіть якщо прилад буде безтоочним. А порти GRIO0, GRIO2 підключені до кнопок SB1, SB2 якими здійснюється користувацьке налаштування приладу. До порта GRIO16 підключений транзистор VT3, який керує роботою бузера BA1. Для програмування

Зм.	Лис	№ докум.	Підпис	Дата

PI81.464419.001 ПЗ

Листм

20

мікроконтролера використовується комутатор CH340(DA2). Живиться блок управління від джерела з напругою 5 В. Необхідна для живлення контролера напруга 3,3 В формує лінійний стабілізатор DA1.

					<i>PI81.464419.001 ПЗ</i>	Лист
						21
Зм.	Лис	№ докум.	Підпис	Дата		

4 ПРОЕКТУВАННЯ ПРИЛАДУ

4.1 Проектування електронного модуля

4.1.1 Обґрунтування методу виготовлення друкованої плати

Так як схема, має велику кількість елементів поверхневого монтажу і відні елементи, то дана плата матиме двосторонній монтаж та два шари металізації, тож її доцільно виготовляти комбінованим позитивним методом, тобто спочатку металізуються отвори, а потім витравляються провідники. Цей метод має такі переваги: можливість відтворення всіх типів друкованих елементів з високою роздільною здатністю; хороша надійність ізоляції; хороша міцність зчеплення металевих елементів плати з діелектричною основою .

4.1.2 Обґрунтування вибору матеріалу плати

Матеріалом плати обрано фольгований склотекстоліт двосторонній FR4. Склотекстоліт має високу механічну міцність, термостійкість, низькі втрати, високий поверхневий опір. Склотекстоліт FR-4 1oz/1oz 1.5 повністю відповідає ГОСТ 26246.5-89 [11].

Відмінною характеристикою цього матеріалу є: високе значення адгезії фольги до підкладки діелектрика під впливом високої температури, високий об'ємний та поверхневий електричний опір, висока температура склування та стабільність геометричних розмірів.

4.1.3 Обґрунтування вибору класу точності плати

Був обраний 5 клас точності плати оскільки вона містить дрібні елементи такі як USB type C (рис. 4.1).

Розрахунок ширини провідника виконувався в MathCad, лістинг розрахунку наведено в Додатку В. Для розрахунку ширини друкованих провідників необхідно знати який максимальний струм та напруга проходять через силові та сигнальні ланцюги. Аналізуючи отримане завдання отримуємо, що для сигнальних провідників $I_{\max} = 0,0025\text{A}$, а для силових $I_{\max} = 1.2\text{A}$. Результати розрахунку наведені в табл. 4.1.

					PI81.464419.001 ПЗ	Лист
						22
Зм.	Лис	№ докум.	Підпис	Дата		

$$S_{\text{дп}} = \sum kSn \quad (4.1)$$

де: n — кількість елементів, k — коефіцієнт форми, S — загальна площа елементів певного корпусу, Ss — мінімальна площа плати.

Корпус	S	n	Загальна площа	k	SS	1 Сторона	2 Сторона
1 Плата			1986,7477		2832,0832	2079,7142	1288,2565
HYG8505	94,15	1,00	94,152	1	94,152		
C1206	8,75	13,00	113,711	1	113,711		
AMS1117	61,12	1,00	61,12	1,5	91,68		
CH340	75,53	1,00	75,527	2	151,054		
ESP8266	454,40	1,00	454,402	2	908,804		
DS3231M	43,32	1,00	43,319	2	86,638		
ZQ3225	15,87	1,00	15,866	1	15,866		
CR2032	658,22	1,00	658,217	1	658,217		
LED1206	8,74	1,00	8,739	1	8,739		
USB C	78,83	1,00	78,827	2	157,654		
R1206	8,03	17,00	136,4437	1	136,4437		
ТАСТ 6x6	89,07	2,00	178,132	1	178,132		
S9014	10,30	3,00	30,891	1	30,891		
TSW 5 PIN	37,40	1,00	37,401	1,5	56,1015		
Стойкі	36	4	144	1	144		
2 Плата			6279,6265		12600,687	2279,187	10521,60
MAX7219	199,52	5,00	997,61	2	1995,22		
12088AW	1032,15	5,00	5160,75	2	10321,5		
TSW 5 PIN	37,40	1,00	37,401	1,5	56,1015		
R1206	8,03	5,00	40,1305	1	40,1305		
C1206	8,75	5,00	43,735	1	43,735		
Стойкі	36	4	144	1	144		

Рисунок 4.2 — Розрахунок габаритів друкованої плати.

Отримаємо, що мінімальна площа плати — 12600 мм², площа, яку займають елементи поверхневого монтажу — 2300 мм², площа, яку займають вивідні елементи — 10500 мм². Оскільки площа, яку займають вивідні елементи, більша, то будемо використовувати її в якості мінімальної площі для друкованої плати. Співвідношення сторін буде майже 1 до 5 бо основні вивідні елементи це квадратні матриці які мають встановлюватись в ряд в кількості 5 штук. Маючи площу і співвідношення сторін отримуємо габарити 180мм на 40мм. Контур плати зображено на рис.4.3.

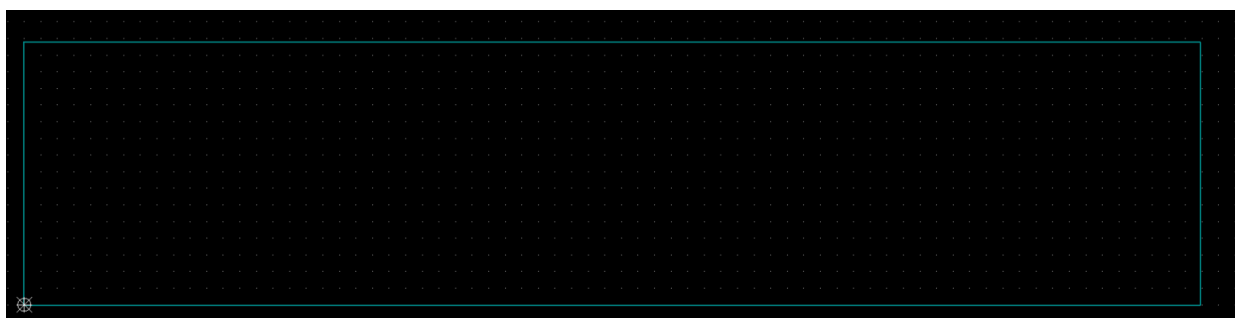


Рисунок 4.3 — Контур плати.

4.2.2 Трасування провідників

В редакторі РСВ проведемо трасування, тобто створимо доріжки та полігони. Були застосовані правила трасування рекомендовані виробником в якого в майбутньому буде здійснене замовлення нашої плати. Компоновка елементів зроблена так щоб на масовому виробництві можна було її автоматизувати. Аналізуючи розрахунки для силових провідників бачимо, що ширина друкованого провідника перевищує 7 мм. На платі дані провідники провести неможливо, тому для з'єднання елементів даного ланцюга застосуємо полігон. Заливка землі (ланцюг GND) також виконаний полігом з обох сторін плати. За вимогою виробника були нанесені реперні мітки. Також було вирішено зробити отвір під WIFI антену щоб збільшити зону покриття приладу. Результати трасування для першої плати наведено на рис 4.4-4.5, а для другої — рис.4.6-4.7.

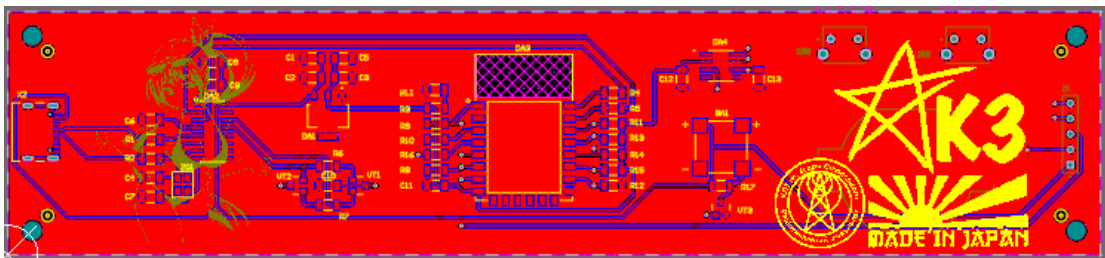


Рисунок 4.4 — Трасування першої плати у верхньому шарі

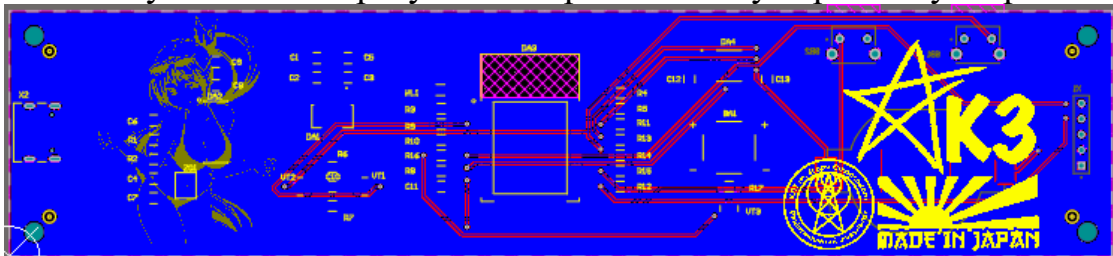


Рисунок 4.5 — Трасування першої плати у нижньому шарі

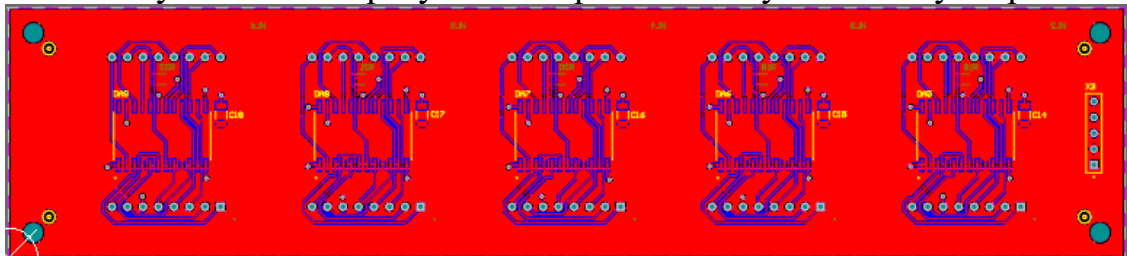


Рисунок 4.6 — Трасування другої плати у верхньому шарі

Зм.	Лис	№ докум.	Підпис	Дата

PI81.464419.001 ПЗ

Лист

25

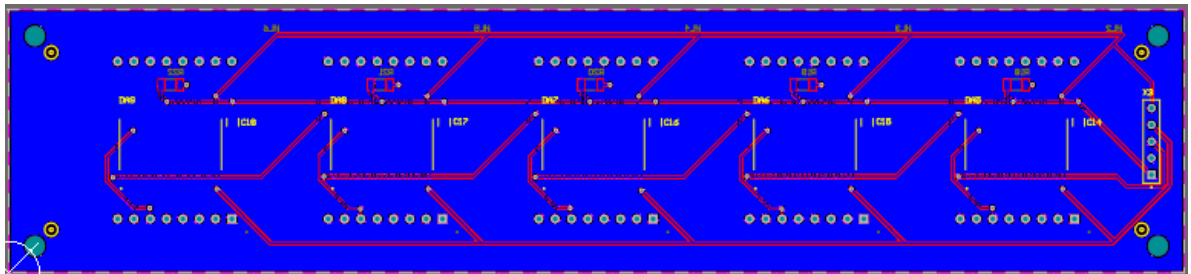


Рисунок 4.7 — Трасування другої плати у нижньому шарі

Проаналізовано завдання з точки зору технолога та визначено: функціонал приладу, метод виготовлення друкованої плати, матеріал та клас точності. Проведено розрахунки розмірів контактних майданчиків, виводів та створено бібліотеку елементів. Розроблено схему електричну принципову. Визначено площу плати, розміри друкованих провідників та проведено їх трасування. Виконано перевірку на помилки правил трасування. Всі помилки виправлені, тому можна переходити до проектування корпусу приладу.

4.3 Проектування корпусу

Вигляд 3Д моделей плат в програмі Altium [13] зображений на рис.4.8-4.11.

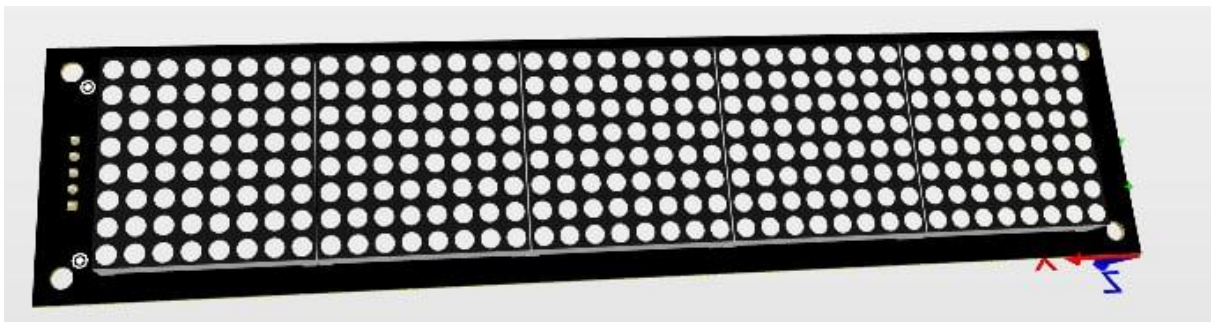


Рисунок 4.8 — Верхня сторона першої плати.



Рисунок 4.9 — Нижня сторона другої плати.

Зм.	Лис	№ докум.	Підпис	Дата

PI81.464419.001 ПЗ

Лист

26

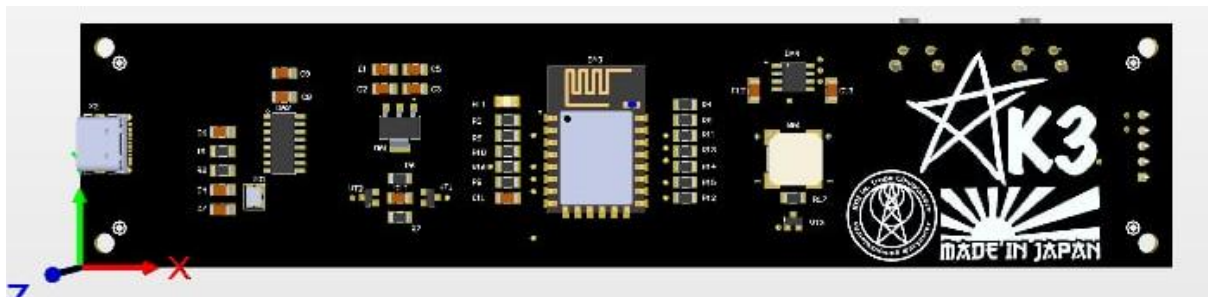


Рисунок 4.10 — Верхня сторона другої плати.



Рисунок 4.11 — Нижня сторона другої плати.

Плати були експортовані в програму SolidWorks [12]. Корпус був спроектований максимально спрощено щоб спростити виробництво і здешевити кінцевий продукт.

Перша версія корпусу:

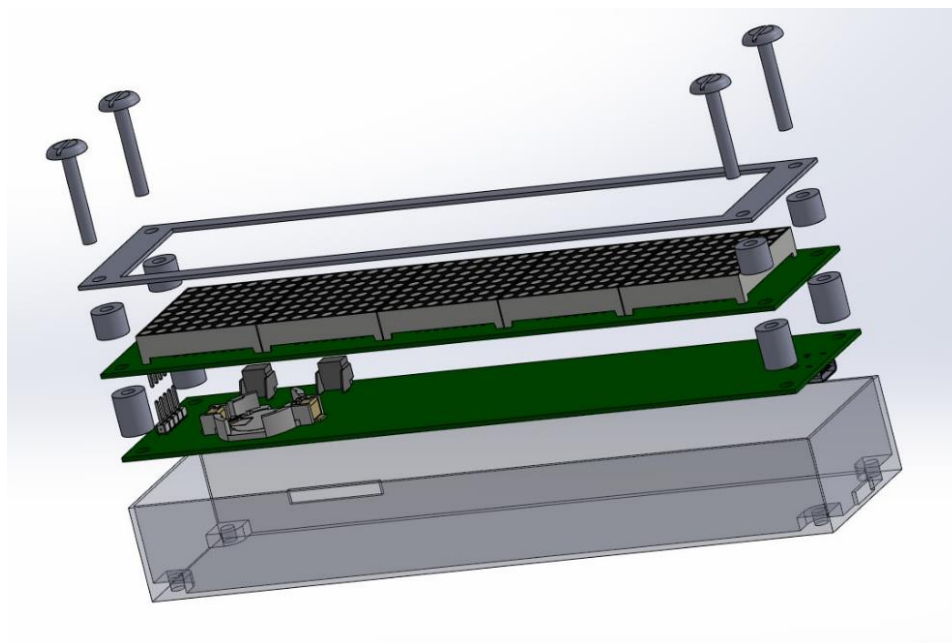


Рисунок 4.12 — Перша версія корпусу

Встановлення плати виконується за допомогою пластикових стійок і стержнів. Таке рішення дозволяє спростити виробництво корпусу, особливо на 3D принтер.

					PI81.464419.001 ПЗ	Лист
Зм.	Лис	№ докум.	Підпис	Дата		27

Даний корпус був дуже простим, непрактичним і негарним томо було прийняте рішення розробити другу версію корпусу яка буде мати гарний вигляд і буде зручною в використанні.

Друга версія корпусу:

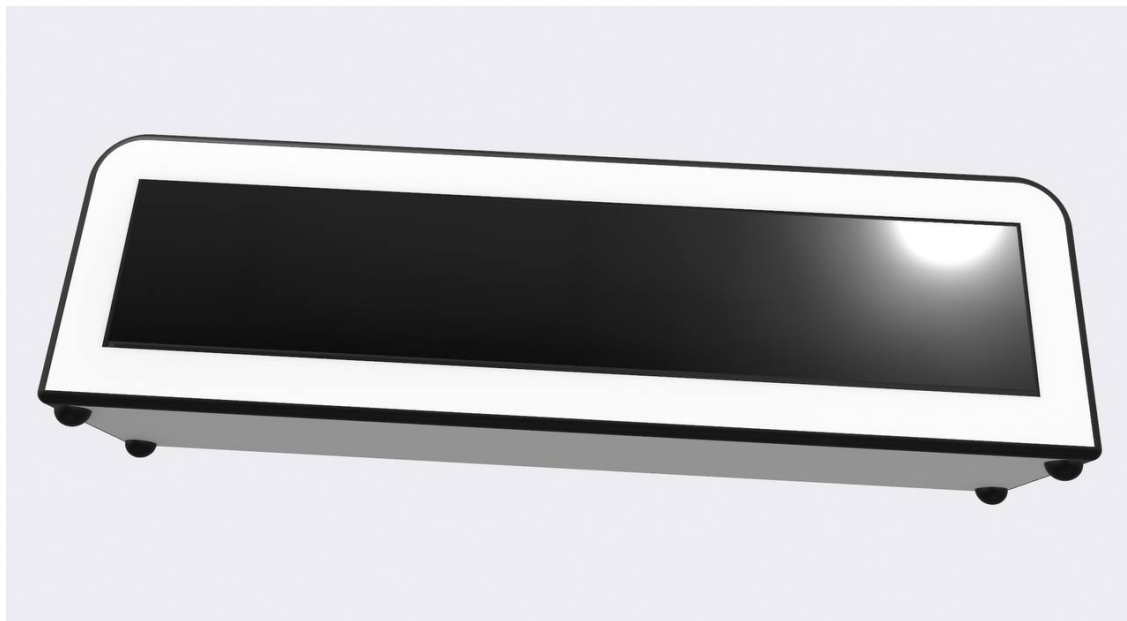


Рисунок 4.13 — Друга версія корпусу

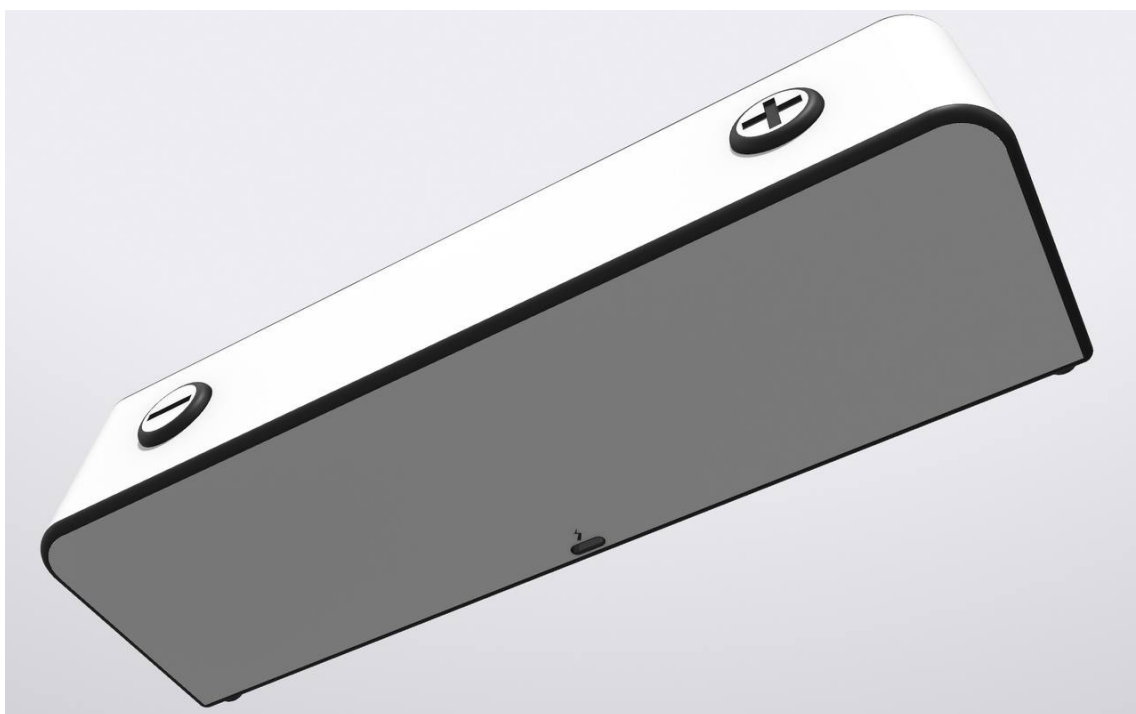


Рисунок 4.14 — Друга версія корпусу вид заду.

В новій версії корпусу був сутєво перероблений дизайн. Кнопки налаштування були зроблені більш зручними і розєм живлення виведений на задню панель замість бокової також були добавлені цупкі ніжки.

Зм.	Лис	№ докум.	Підпис	Дата

PI81.464419.001 ПЗ

Лист

28

5 МАКЕТУВАННЯ ТА ТЕСТУВАННЯ

По завершенню трасування було здійснено замовлення плат у виробника JLSPCB [14], готові плати зображені на рис. 5.1.

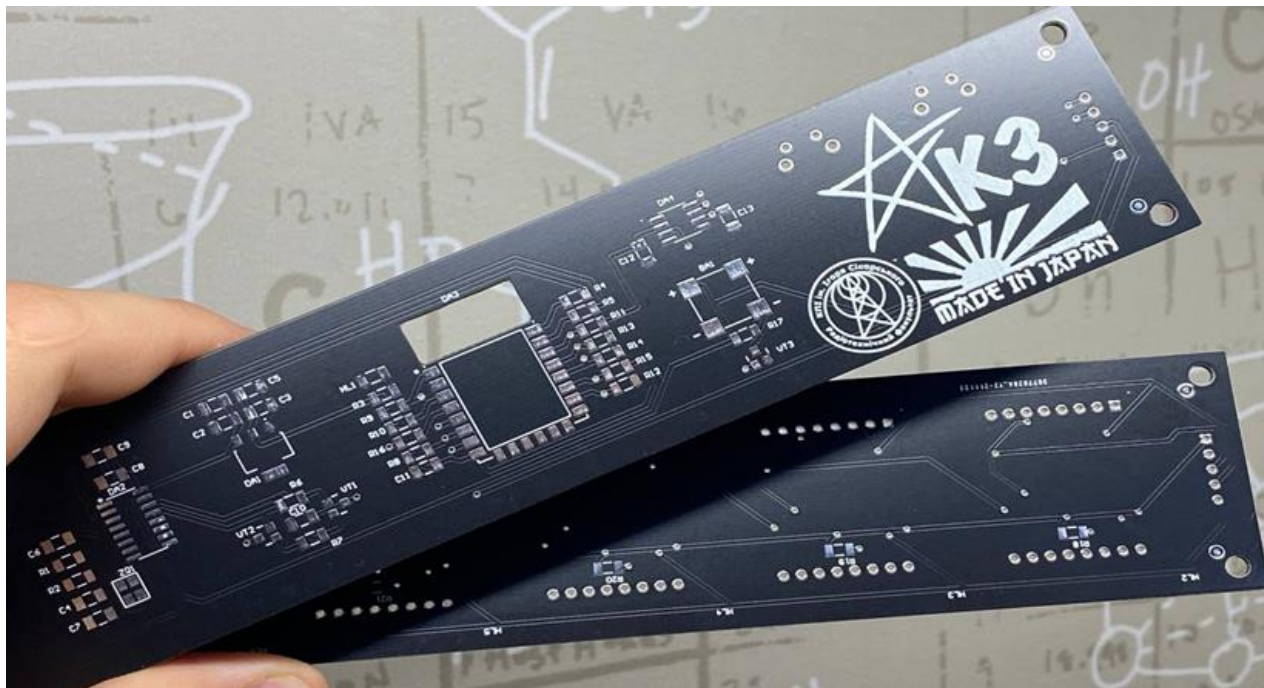


Рисунок 5.1 — Друковані плати.

Після отримання замовлення було здійснено монтаж радіо компонентів. Плати з усіма компонентами зображенні на рис. 5.2. Монтаж елементів здійснювався самостіно (в ручну).

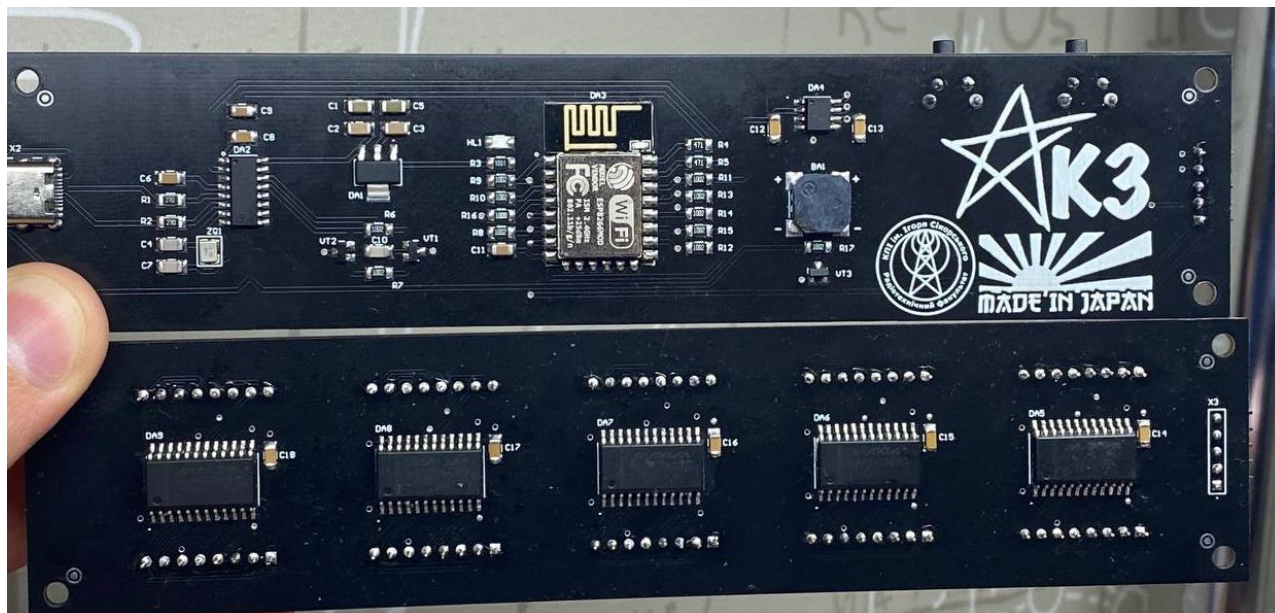


Рисунок 5.2 — Готові плати.

Зм.	Лис	№ докум.	Підпис	Дата

PI81.464419.001 ПЗ

Лист
29

Після успішного монтажу всіх компонентів була здійснена перевірка на працездатність. При першому прошиванні за допомогою тестово коду [15] для мікроконтролера все вдало запустилося. Було підтверджено працездатність таких вузлів як живлення, програматор, бузер, модуль часу і робота Wi-Fi модуля вмонтованого в мікроконтролер.

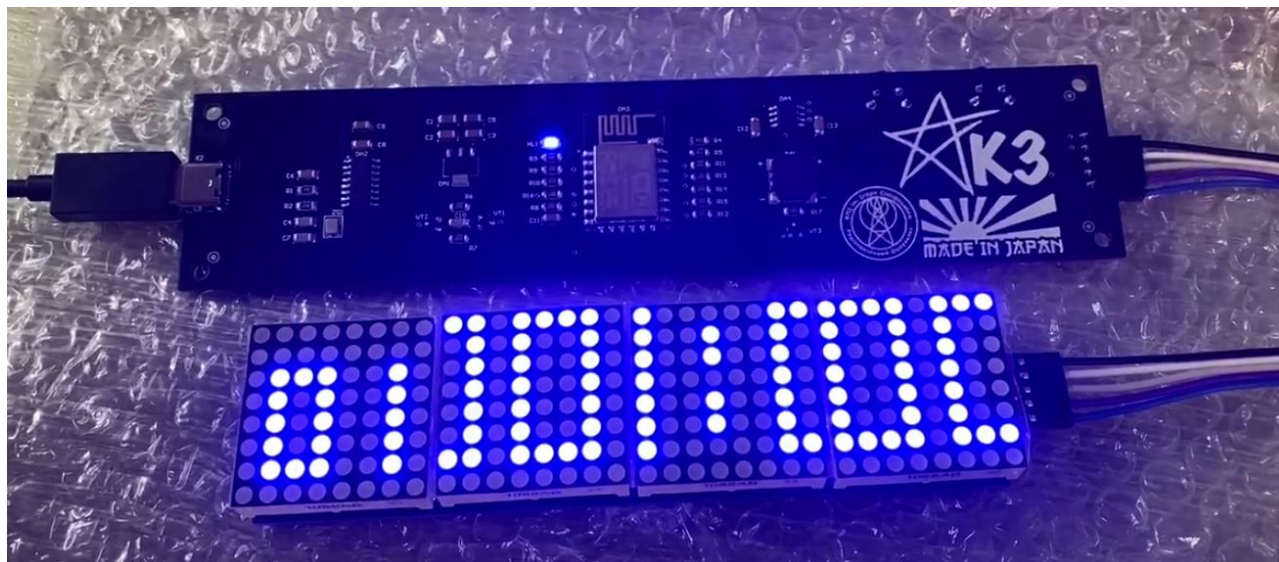


Рисунок 5.3 — Перша перевірка.

Після підтвердження працездатності плат було здійснено прошивання на повноцінний програмний код (Додаток Ж) з подальшим налаштуванням для проведення тестувань.

Алгоритм роботи програмного забезпечення будильника зображений на рис. 5.4. Після першого запуску будильник запропонує ввести новий системний час або залишити уже наявний. Далі буде запропоновано вказати час сигналу будильника після чого пристрій перейде в графічний режим де він буде відображати час. Далі в залежності від того чи був наведений сигнал будильник запустить функцію пробудження. Після вимкнення сигналу будильник перейде знов в графічний режим до нового сигналу.



Рисунок 5.4 — Алгоритм роботи.

Після повної збірки та прошивки будильнику, він був переданий на тестування користувачу. Через три дні користування даним приладом була підтверджена його ефективність і зручність.

Зм.	Лис	№ докум.	Підпис	Дата

PI81.464419.001 ПЗ

Лист
31

ВИСНОВКИ

1. Проаналізуємо ринок та розглянемо найпопулярніші моделі будильників світлового типу. Було визначено їх переваги та недоліки, що було враховано при подальшому проектуванні приладу. За результатами огляду існуючих рішень визначено їх переваги та недоліки, що було враховано при подальшому проектуванні приладу. Зроблена порівняльна таблиця з якої були вибрані оптимальні характеристики для нашого приладу

2. Розглянемо два схемотехнічних рішення. З аналізу їх переваг та недоліків були прийняті певні рішення які були враховані при подальшому проектуванні. А, саме було прийнято рішення про необхідність використання однієї матриці для відображення інформації і для здійснення світлового сигналу щоб зменшити загальні габарити пристрою.

3. Була синтезована структурна схема, обрана та обґрунтована елементна база а також створена схема електрична-принципова з детальним поясненням роботи всіх компонентів.

4. Проаналізувавши завдання з точки зору конструктора та точки зору технолога, визначено: функціонал приладу, метод виготовлення друкованої плати, матеріал та клас точності. Оскільки було обрано виробником плат компанію JLCPCB більшість параметрів для проектування плат були задані у відповідності із можливостями виробництва компанією. За допомогою програмного забезпечення Altium Designer спроектовано друковані плати індикації та управління. В програмному забезпеченні SolidWorks створено першу версію корпусу пристрою. Проаналізувавши всі недоліки першої версії корпусу, а саме не ергономічний та примітивний дизайн, була створена друга версія корпусу яка стала кінцевою.

5. Було здійснено замовлення друкованих плат на промисловому виробництві JLCPCB. Після отримання плат, був зібраний макет який пройшов перевірку на працездатність. Написано код і прошиито наш прилад, було проведено тестування які допомогли виявити деякі нюанси які допоможуть під час розробки нової версії приладу в майбутньому. Нюанси такі як складність

					<i>PI81.464419.001 ПЗ</i>	Лист
						32
Зм.	Лист	№ докум.	Підпис	Дата		

програмного забезпечення для звичайних людей буде складно налаштувати такий будильник тому було прийнято рішення в наступній версії спростити функціонал будильник з можливістю його модифікацій за рахунок відкритого коду.

6. Головна відмінність даного будильки відносно промислових аналогів це використання одного джерела світла і для індикації часу і для пробудження людини. Таке рішення допомогло суттєво зменшити габарити приладу а також зробити більш мінімалістичний дизайн. Оскільки розроблений прилад має вбудований програматор користувач зможе в будьякий момент перепрошити його під свої потреби.

					<i>PI81.464419.001 ПЗ</i>	Лист
						33
Зм.	Лис	№ докум.	Підпис	Дата		

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Будильник «Hatch Restore» [Електронний ресурс] – Режим доступу до ресурсу: <https://smart-gadget.club/smart-fitness/alarm-clock/hatch-restore>
2. Будильник «NOKU SUNSHINE» [Електронний ресурс] – Режим доступу до ресурсу: https://profmedmarket.com.ua/p1365258959-svetovoj-budilnik-beuber.html?source=merchant_center
3. Будильник «Sunrise» [Електронний ресурс] – Режим доступу до ресурсу:
https://www.wish.com/product/617d9521f6ce6bacb2f704f9?from_ad=goog_shopping_organic&_display_country_code=UA&_force_currency_code=UAH&pid=googleadwords_int&c=%7BcampaignId%7D&ad_cid=617d9521f6ce6bacb2f704f9&ad_cc=UA&ad_lang=UK&ad_curr=UAH&ad_price=9672.13&hide_login_modal=true&share=web
4. Будильник «TITIROBA MY-10» [Електронний ресурс] – Режим доступу до ресурсу:
https://www.wish.com/product/606097f3c748f6ad8fc1c90f?from_ad=goog_shopping_organic&_display_country_code=UA&_force_currency_code=UAH&pid=googleadwords_int&c=%7BcampaignId%7D&ad_cid=606097f3c748f6ad8fc1c90f&ad_cc=UA&ad_lang=UK&ad_curr=UAH&ad_price=7432.22&hide_login_modal=true&share=web
5. Будильник «Beurer WL 50» [Електронний ресурс] – Режим доступу до ресурсу: <https://makeup.com.ua/ua/product/969738/>
6. Будильник «Beurer WL 75» [Електронний ресурс] – Режим доступу до ресурсу: https://tekhnoservice.com/ua/p1298564130-svetovoj-budilnik-nochnik.html?source=merchant_center&utm_source=google&utm_medium=cpc&utm_campaign=13381929753&utm_term=&utm_content=525775108063&utm_position=&utm_matchtype=&utm_placement=&utm_network=u&gclid=Cj0KCQjwspKUBhCvARIsAB2IYuvrE3hWMiNrZHyvUr_wlctFcZ-PGz4D18iNPqwgYjw8J-KV-12AGnEaAiyAEALw_wcB

					<i>PI81.464419.001 ПЗ</i>	Лист
						34
Зм.	Лис	№ докум.	Підпис	Дата		

7. Будильник «Philips HF3651» [Електронний ресурс] – Режим доступу до ресурсу:
https://rozetka.com.ua/268670761/p268670761/?gclid=Cj0KCQjwspKUBhCvARIsAB2IYuvTq2R_pGySrelc3MosOQ1rp3r5RZshCyouqVITfHvNQgiuFKwFsHAaAhPsEALw_wcB
8. Будильник «Philips HF3505» [Електронний ресурс] – Режим доступу до ресурсу: https://prom.ua/ua/p1498497300-budilnik-titiroba-imitatsiej.html?utm_source=google_pmax&utm_medium=cpc&utm_content=pla&utm_campaign=Pmax_cpa_war&gclid=Cj0KCQjwspKUBhCvARIsAB2IYuvAnMhSq99-8iGY1bSDUBByOTIUREvKu5uhi9uWNYrRzXY7OeTs690YaAh3LEALw_wcB
9. Перше схемотехнічне рішення [Електронний ресурс] – Режим доступу до ресурсу: <https://habr.com/ru/post/249535/>
10. Друге схемотехнічне рішення [Електронний ресурс] – Режим доступу до ресурсу: <https://github.com/AlexGyver/Dawn-Clock>
11. ГОСТ 26246.5-89 [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.cntd.ru/document/1200011703>
12. Програмне забезпечення для створення 3D моделей [Електронний ресурс] – Режим доступу до ресурсу: <https://www.solidworks.com>
13. Програмне забезпечення для створення плат [Електронний ресурс] – Режим доступу до ресурсу: <https://www.altium.com>
14. Виробництво друкованих плат [Електронний ресурс] – Режим доступу до ресурсу: <https://jlcpcb.com/>
15. Arduino. Blink [Електронний ресурс] – Режим доступу до ресурсу: <https://www.arduino.cc/en/Tutorial/BuiltInExamples/Blink>

ДОДАТОК А ПЕРЕЛІК ЕЛЕМЕНТІВ ДО СХЕМИ МОДУЛЯ ІНДИКАЦІЇ

Позн.	Найменування	Кіл.	Примітки
	<u>Конденсатори</u>		
C14-C18	1206 100nF 50V X5R (CL31A104KBCNNNC)	5	
	<u>Мікросхеми</u>		
DA5-DA9	MAX7219	5	
	<u>Світлодіоди</u>		
HL2-HL6	12088AW	5	
	<u>Резистори</u>		
R18-R22	SMD 10K 1206 1%	5	
	<u>З'єднання</u>		
X3	TSW 5 PIN		

ДОДАТОК Б ПЕРЕЛІК ЕЛЕМЕНТІВ ДО СХЕМИ МОДУЛЯ УПРАВЛІННЯ

Позн.	Найменування	Кіл.	Примітки
	<u>Гучномовці</u>		
BA1	HYG8505A	1	
	<u>Конденсатори</u>		
C1	1206 100uF 6.3V X5R (CL31A107MQHNNWE)	1	
C2-C3	1206 1uF 50V X7R (CL31B105KBHNNNE)	2	
C4	1206 22pF 50V NP0 (CL31C220JBCNNNC)	1	
C5	1206 100uF 6.3V X5R (CL31A107MQHNNWE)	1	
C6	1206 100nF 50V X5R (CL31A104KBCNNNC)	1	
C7	1206 22pF 50V NP0 (CL31C220JBCNNNC)	1	
C8	1206 100nF 50V X5R (CL31A104KBCNNNC)	1	
C9	1206 10uF 25V X5R (CL31A106KAHNNNE)	1	
C10	1206 470pF 50V NP0 (CL31C471JBCNNNC)	1	
C11	1206 10uF 25V X5R (CL31A106KAHNNNE)	1	
	<u>Мікросхеми</u>		
DA1	AMS1117	1	
DA2	CH340	1	
DA3	ESP8266MOD	1	
DA4	DS3231M		
	<u>Відсіки для акумуляторів</u>		
GB1	CR2032-05 SMD-type	1	
RI81.676831.001 ПЕ			
Зм.	Арк.	№ докум.	Підпис
Розробив	Сверєда		
Перевір.	Микола		
Реценз.			
Н. Контр	Полсуй		
Затверд.			
		Лім.	Арк.
			1
		КПІ ім. Ігоря Сікорського, РТФ	
		Аркушів	2

[illegible]

ДОДАТОК В ЛІСТИНГ РОЗРАХУНКУ ШИРИНИ ПРОВІДНИКА

Розрахунок ширини друкованих провідників для всіх присутніх у схемі рівнів сигналу

Для сигнальних:

Мінімальне значення ширини друкованого провідника $t1$ у вузькому місці, мм:

Вибираємо за класом точності - 4 $t_{minD_S} := 0.15 \text{ mm}$

Допуск на ширину провідника (нижнє відхилення) $\Delta t_{noS} := -0.05 \text{ mm}$

$$t1V_S := t_{minD_S} + |\Delta t_{noS}| = 0.2 \text{ mm}$$

ПРИМІТКА: Для завдань підвищеної складності - клас точності 3, для завдань звичайної складності - клас точності 2 а для ваще крутих 4 клас точності)

Мінімальне значення ширини друкованого провідника $t1$ у широкому місці, мм:

Вибираємо за класом точності (на один менше) $t_{minD_S} := 0.15 \text{ mm}$

Допуск на ширину провідника (нижнє відхилення) $\Delta t_{noS} := -0.1 \text{ mm}$

$$t1III_S := t_{minD_S} + |\Delta t_{noS}| = 0.25 \text{ mm}$$

ПРИМІТКА: Для завдань підвищеної складності - клас точності 2, для завдань звичайної складності - клас точності 1.

Мінімально допустиму ширину провідника $t2$ з урахуванням допустимого падіння напруги на ньому (3%):

Питомий опір провідників ($\text{Om}^* \text{mm}^2/\text{m}$)- $\rho_S := 0.0175 \frac{\Omega \cdot \text{mm}^2}{\text{m}}$

Довжина провідника (м)- $l_S := 0.050 \text{ m}$

Товщина фольги (мм)- $h_S := 35 \cdot 10^{-3} \text{ mm}$

Прикладена напруга (В)- $U_{живS} := 5 \text{ V}$

Максимальний струм (А)- $I_{maxS} := 0.0025 \text{ A}$

$$t2_S := \frac{l_S \cdot I_{maxS} \cdot \rho}{h_S \cdot U_{живS} \cdot 0.03} = 4.167 \times 10^{-4} \cdot \text{mm}$$

ПРИМІТКА: Прикладена напруга і максимальний струм визначаються за схемою електричною принциповою та DataSheet на корпус. Довжина провідника обирається згідно довжини з'єднання в файлі pcb3.pcb.

Мінімально допустиму ширину провідника $t3$ з урахуванням допустимого рівня струму на ньому:

Максимальний струм (А)- $I_{maxS} := 0.0025 \text{ A}$

Товщина фольги (мм)- $h_S := 35 \cdot 10^{-3} \text{ mm}$

Допустима щільність струму в провіднику (A/mm^2)- $j_S := 20 \frac{\text{A}}{\text{m}^2}$

$$t3_S := \frac{I_{maxS}}{h_S \cdot j_S} = 3.571 \times 10^{-3} \cdot \text{mm}$$

Для силових:

Мінімальне значення ширини друкованого провідника t1 у вузькому місці, мм:

Вибираємо за класом точності - 4

$$t_{minDp} := 0.15 \text{ mm}$$

Допускна ширину провідника (нижнє відхилення)

$$\Delta t_{nop} := -0.05 \text{ mm}$$

$$t1Vp := t_{minDp} + |\Delta t_{nop}| = 0.2 \cdot \text{mm}$$

ПРИМІТКА: Для завдань підвищеної складності - клас точності 3, для завдань звичайної складності - клас точності 2.

Мінімальне значення ширини друкованого провідника t1 у широкому місці, мм

Вибираємо за класом точності-

$$t_{minDp} := 0.15 \text{ mm}$$

Допуск на ширину провідника (нижнє відхилення)

$$\Delta t_{nop} := -0.1 \text{ mm}$$

$$t1IIIp := t_{minDp} + |\Delta t_{nop}| = 0.25 \cdot \text{mm}$$

ПРИМІТКА: Для завдань підвищеної складності - клас точності 2, для завдань звичайної складності - клас точності 1.

Мінімально допустиму ширину провідника t2 з урахуванням допустимого падіння напруги на ньому (3%):

$$\text{Питомий опір провідників (Ом*мм}^2\text{/м)}- \quad \rho := 0.0175 \frac{\Omega \cdot \text{mm}^2}{\text{m}}$$

Довжина провідника (м)-

$$l_p := 0.05 \text{ m}$$

Товщина фольги (мм)-

$$h_p := 35 \cdot 10^{-3} \text{ mm}$$

Прикладена напруга (В)-

$$U_{живp} := 5 \text{ V}$$

Максимальний струм (А)-

$$I_{maxp} := 1.2 \text{ A}$$

$$t2_p := \frac{l_p \cdot I_{maxp} \cdot \rho}{h_p \cdot U_{живp} \cdot 0.03} = 0.2 \cdot \text{mm}$$

ПРИМІТКА: Прикладена напруга і максимальний струм визначаються за схемою електричною принциповою та DataSheet на корпус. Довжина провідника обирається згідно довжини з'єднання в файлі pcb3.pcb.

Мінімально допустиму ширину провідника t3 з урахуванням допустимого рівня струму на ньому:

Максимальний струм (мА)-

$$I_{maxp} := 1.2 \text{ A}$$

Товщина фольги (мм)-

$$h_p := 35 \cdot 10^{-3} \text{ mm}$$

Допустима щільність струму в провіднику (А/мм²)-

$$j_p := 20 \frac{\text{A}}{\text{mm}^2}$$

$$t3_p := \frac{I_{maxp}}{h_p \cdot j_p} = 1.714 \cdot \text{mm}$$

ДОДАТОК Г СПЕЦИФІКАЦІЯ НА ЕЛЕКТРОННИЙ МОДУЛЬ ІНДИКАЦІЇ

Форм.	Зона	Поз.	Позначення	Назва	Кільк.	Прим.		
				<u>Документація</u>				
A2			RI81.676831.002 E3	Схема електрична принципова E3				
A4			PI81.676831.002 ПЕ	Перелік елементів				
A2			PI81.758746.002 СК	Складальний кресленик				
			PCB2.GTL	Топологія верхнього шару плати				
			PCB2.GBL	Топологія нижнього шару плати				
			PCB2-SlotHoles-Plated.txt	Координати отворів				
			PCB2.GBS	Захисна маска нижнього шару				
			PCB2.GTS	Захисна маска верхнього шару				
			PCB2.GTO	Верхній шар шовкографії				
			PCB2.GBO	Нижній шар шовкографії				
				<u>Деталі</u>				
A3			PI81.758746.002	Друкована плата				
				RI81.676831.002				
З мі н.	Ар- куш	№ докум.	Підп	Дата				
Розроб.	Свереда				Друкований вузол	Літ.	Ар- куш	Аркушів
Перев.	Микола						1	1
Т.конт р.	Попсуй					НТУУ «КПІ», РТФ, гр. PI-81		
Н.конт р.	Попсуй							
Затв.	Зінгер							

ДОДАТОК Д СПЕЦИФІКАЦІЯ НА ЕЛЕКТРОННИЙ МОДУЛЬ УПРАВЛІННЯ

Форм.	Зона	Поз.	Позначення	Назва	Кільк.	Прим.		
				<u>Документація</u>				
A2			RI81.676831.001 E3	Схема електрична принципова E3				
A4			PI81.676831.001 ПЕ	Перелік елементів				
A2			PI81.758746.001 СК	Складальний кресленик				
			PCB1.GTL	Топологія верхнього шару плати				
			PCB1.GBL	Топологія нижнього шару плати				
			PCB1-SlotHoles-Plated.txt	Координати отворів				
			PCB1.GBS	Захисна маска нижнього шару				
			PCB1.GTS	Захисна маска верхнього шару				
			PCB1.GTO	Верхній шар шовкографії				
			PCB1.GBO	Нижній шар шовкографії				
				<u>Деталі</u>				
A3			PI81.758746.001	Друкована плата				
				RI81.676831.001				
3 мі н.	Ар- куш	№ докум.	Підп	Дата				
Розроб.	Свереда				Друкований вузол	Літ.	Ар- куш	Аркушів
Перев.	Микола						1	1
Т.конт р.	Попсуй					НТУУ «КПІ», РТФ, гр. PI-81		
Н.конт р.	Попсуй							
Затв.	Зінгер							

ДОДАТОК Е ЛІСТИНГ ПРОГРАМИ

```
#include <SPI.h>
#include <Ticker.h>
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
// #include <WiFiUdp.h>
#include <PubSubClient.h>

#include <WiFiManager.h>
#include <ArduinoJson.h>
#include <Wire.h>
#include <time.h>
#include "FS.h"
#include "character.h"

// #include <ESP8266WebServer.h>
// #include <ESP8266HTTPUpdateServer.h>
// #include <ESP8266mDNS.h>

// *****
*****

#define SDA    5    // Pin sda (I2C)
#define SCL    4    // Pin scl (I2C)
#define CS     15   // Pin cs  (SPI)
#define anzMAX  5    // max number of 8x8 led matrix led module
// #define REVERSE_HORIZONTAL 1
// #define REVERSE_VERTICAL 1

//////////DS3231          ALARM          MODE          DEFINI-
TIONS//////////
```

```
#define ALARM1_EVERY_SECOND      0x0F
#define ALARM1_SEC_MATCH        0x0E
#define ALARM1_MIN_SEC_MATCH     0x0C
#define ALARM1_HOUR_MIN_SEC_MATCH 0x08

#define ALARM1_DATE_HOUR_MIN_SEC_MATCH 0x00
#define ALARM1_DAY_HOUR_MIN_SEC_MATCH  0x10
```

```
#define ALARM2_EVERY_MINUTE      0x8E
#define ALARM2_MIN_MATCH        0x8C
#define ALARM2_HOUR_MIN_MATCH   0x88
```

```
#define ALARM2_DATE_HOUR_MIN_MATCH 0x80
#define ALARM2_DAY_HOUR_MIN_MATCH  0x90
```

```
#define NTP_PACKET_SIZE        48
#define MSGMAXLEN 20
```

```
/////////////////////////////////CONSTANTS/////////////////////////////////
```

```
const unsigned char DS3231_ADDRESS = 0x68;
const unsigned char secondREG = 0x00;
const unsigned char minuteREG = 0x01;
const unsigned char hourREG = 0x02;
const unsigned char weekREG = 0x03;           //weekday
const unsigned char dateREG = 0x04;
const unsigned char monthREG = 0x05;
const unsigned char yearREG = 0x06;
const unsigned char alarm1M1SecREG = 0x07;
const unsigned char alarm1M2MinREG = 0x08;
const unsigned char alarm1M3HrREG = 0x09;
```

```
const unsigned char alarm1M4DateREG = 0x0A;
```

```
const unsigned char alarm2M2MinREG = 0x0B;
```

```
const unsigned char alarm2M3HrREG = 0x0C;
```

```
const unsigned char alarm2M4DateREG = 0x0D;
```

```
const unsigned char controlREG = 0x0E;
```

```
const unsigned char statusREG = 0x0F;
```

```
const unsigned char ageoffsetREG = 0x10;
```

```
const unsigned char tempMSBREG = 0x11;
```

```
const unsigned char tempLSBREG = 0x12;
```

```
const unsigned char btnA = 2;
```

```
const unsigned char btnB = 0;
```

```
const unsigned char buz = 16;
```

```
const unsigned int intPin = 12;
```

```
//uint8_t MsgBuffer[MSGMAXLEN];
```

```
unsigned short const wordSpace = 6;
```

```
unsigned short maxPosX = anzMAX * 8 - 1;
```

```
unsigned short LEDarr[anzMAX][8];
```

```
unsigned short helpArrMAX[anzMAX * 8];
```

```
unsigned short helpArrPos[anzMAX * 8];
```

```
unsigned int zPosX = 0;
```

```
unsigned int dPosX = 0;
```

```
bool fTckr1s = false;
```

```
bool fTckr50ms = false;
```

```
bool fTckr500ms = false;
```

```
unsigned long epoch = 0;
```

```
unsigned int localPort = 2391;           // local port to listen for UDP packets
```

```

const char* ntpServerName = "time1.aliyun.com";
byte packetBuffer[NTP_PACKET_SIZE];
tm *tt, ttm;

unsigned long lastReconnectAttempt = 0; // for mqtt
bool mqttReceived = false;
char mqttMsg0[200] = {};
char mqttMsg1[200] = {};
bool mqttOn = false;
uint8_t mqttMsgType = 0;
bool mqttButton = false;

struct DateTime
{
    //sec = sec hour = hour year = year date = date weekday = weekday
    unsigned short sec1, sec2, sec12, min1, min2, min12, hour1, hour2, hour12;
    unsigned short date1, date2, date12, mon1, mon2, mon12, year1, year2, year12,
weekday;
    } MEZ;

struct configs
{
    char callsign[7] = {};
    uint8_t showCall = 1;
    uint16_t brightness = 0;
    int8_t timeZone = 8;
    uint8_t h24_12 = 1; //0 for 24-hour format
    uint8_t alarmOn = 0;
    uint8_t amPm = 0; //0 for am and 1 for pm

```

```

char mqttServer[40];
char mqttPort[6] = "1800";
char mqttUser[16] = " ";
char mqttPass[16] = " ";

} config1;

struct alarm_time
{
    bool alarmNow;
    uint8_t alarmSec;
    uint8_t alarmMin;
    uint8_t alarmHour;
    uint8_t alarmDate;
    uint8_t alarmDay;
    bool  alarmAmPm;
} alarmTime;

struct utc_time
{
    int8_t utcHour;
    uint8_t utcAmPm;
    int8_t utcBackOrPlusOneDay = 0;
    int8_t utcBackOrPlusOneMon = 0;
    int8_t utcBackOrPlusOneYear = 0;
    bool  utcAct = false;
} utc;

const unsigned short initArr[7][2] =
{
    { 0x0C, 0x00 }, // display off

```

```

        { 0x00, 0xFF }, // no LEDtest
        { 0x09, 0x00 }, // BCD off
        { 0x0F, 0x00 }, // normal operation
        { 0x0B, 0x07 }, // start display
        { 0x0A, 0x04 }, // brightness
        { 0x0C, 0x01 } // display on
};

//months
char monthArray[12][5] = { { ' ', 'J', 'A', 'N', ' ' }, { ' ', 'F', 'E', 'B', ' ' },
    { ' ', 'M', 'A', 'R', ' ' }, { ' ', 'A', 'P', 'R', ' ' }, { ' ', 'M', 'A',
        'Y', ' ' }, { ' ', 'J', 'U', 'N', ' ' }, { ' ', 'J', 'U', 'L', ' ' }, {
        ' ', 'A', 'U', 'G', ' ' }, { ' ', 'S', 'E', 'P', ' ' }, { ' ', 'O', 'C',
        'T', ' ' }, { ' ', 'N', 'O', 'V', ' ' }, { ' ', 'D', 'E', 'C', ' ' } };

//days
char weekdayArray[7][4] = { { 'S', 'U', 'N', ' ' }, { 'M', 'O', 'N', ' ' }, { 'T', 'U', 'E', ' ' }, {
}, {
        'W', 'E', 'D', ' ' }, { 'T', 'H', 'U', ' ' }, { 'F', 'R', 'I', ' ' }, { 'S', 'A', 'T', ' ' } };

void ICACHE_RAM_ATTR handleInterrupt();

//////////SUB          ROUTINES          DECLARA-
TIONS//////////

//////////SUB          ROUTINES          BEGIN
HERE//////////

//*****NTP          SYNCHRONIZA-
TION*****//
tm* connect_ntp()

```

```

{
  WiFi.hostByName(ntpServerName, timeServerIP);
  Serial.println(timeServerIP);
  Serial.println("sending NTP packet...");
  memset(packetBuffer, 0, NTP_PACKET_SIZE);
  // Initialize values needed to form NTP request
  // (see URL above for details on the packets)
  packetBuffer[0] = 0b11100011; // LI, Version, Mode
  packetBuffer[1] = 0; // Stratum, or type of clock
  packetBuffer[2] = 6; // Polling Interval
  packetBuffer[3] = 0xEC
  packetBuffer[12] = 49;
  packetBuffer[13] = 0x4E;
  packetBuffer[14] = 49;
  packetBuffer[15] = 52;
  udp.beginPacket(timeServerIP, 123); //NTP requests are to port 123
  udp.write(packetBuffer, NTP_PACKET_SIZE);
  udp.endPacket();
  delay(1000); // wait to see if a reply is available
  int cb = udp.parsePacket();
  udp.read(packetBuffer, NTP_PACKET_SIZE); // read the packet into the
buffer

  epoch = secsSince1900 - seventyYears; //+2000ms processing time
  time_t t;

  t = epoch + 3600 * config1.timeZone + 2;
  tm* tt;
  tt = localtime(&t);
  Serial.println(epoch);
  Serial.println(asctime(tt));
  if (cb == 48)

```

```

        return (tt);
    else
        return (NULL);
}

```

```

//*****50MS
TIMER*****//
void timer_50ms()
{
    static unsigned int cnt50ms1 = 0, cnt50ms2 = 0, cnt50ms3 = 0;
    static uint8_t btnCntA, btnCntB;
    fTckr50ms = true;
    cnt50ms1++;
    cnt50ms2++;

    if (alarmTime.alarmNow)
        cnt50ms3++;

    if(!digitalRead(btnA))
    {
        btnCntA++;
    }
    else
        btnCntA = 0;

    if(btnCntA > 3)
    {
        btnCntA = 0;
        Serial.println("btnA triggered...");
        utc.utcAct = !utc.utcAct;
        Serial.printf("utc.utcAct = %d\n", utc.utcAct);
    }
}

```

```
}
```

```
if(!digitalRead(btnB))
```

```
{
```

```
    btnCntB++;
```

```
}
```

```
else
```

```
    btnCntB = 0;
```

```
if(btnCntB > 3)
```

```
{
```

```
    btnCntB = 0;
```

```
    mqttButton = true;
```

```
    mqttMsgType = mqttMsgType + 1;
```

```
    if(mqttMsgType == 2)
```

```
        mqttMsgType = 0;
```

```
    Serial.println("mqttButton ON...");
```

```
    Serial.printf("mqttMsgType = %d\n", mqttMsgType);
```

```
}
```

```
if (cnt50ms1 == 20)
```

```
{
```

```
    fTckr1s = true; // 1 sec
```

```
    cnt50ms1 = 0;
```

```
}
```

```
if (cnt50ms2 == 10)
```

```
{
```

```
    fTckr500ms = true; // 0.5 sec
```

```
    cnt50ms2 = 0;
```

```
}
```

```

    if (cnt50ms3 == 1000)
    {
        //fTckr100ms = true; // 50 sec
        cnt50ms3 = 0;
        alarmTime.alarmNow = false;
        noTone(buz);
    }
    if(mqttOn)
        mqtt_keep_connect();
}

```

```

//*****SPIFFS TO LOAD CONFIGURA-
TIONS*****//

```

```

void fs_start()
{
    //clean FS, for testing
    // SPIFFS.format();

    int iStart = 0, iEnd = 0;
    //read configuration from FS json
    Serial.println("mounting FS...");

    if (SPIFFS.begin())
    {
        Serial.println("mounted file system");
        if (SPIFFS.exists("/config.json"))
        {
            //file exists, reading and loading
            Serial.println("reading config file");
            File configFile = SPIFFS.open("/config.json", "r");

```

```

if (configFile)
{
    Serial.println("opened config file");
    size_t size = configFile.size();
    // Allocate a buffer to store contents of the file.
    std::unique_ptr<char[]> buf(new char[size]);

    configFile.readBytes(buf.get(), size);
    DynamicJsonBuffer jsonBuffer;
    JsonObject& json = jsonBuffer.parseObject(buf.get());
    json.printTo(Serial);
    if (json.success())
    {
        Serial.println("\nparsed json");

        strcpy(config1.callsign, json["call"]);
        Serial.println(config1.callsign);

        String showCall = json["showCall"];
        if(showCall == "yes")
            config1.showCall = 1;
        else
            config1.showCall = 0;
        Serial.println(showCall);

        config1.brightness = atoi(json["brightness"]);
        Serial.printf("brightness = %d", config1.brightness);
        Serial.println();

        config1.timeZone = atoi(json["timeZone"]);
        Serial.printf("timeZone = %d", config1.timeZone);
    }
}

```

```
Serial.println();
```

```
String hourFormat = json["hourFormat"];
```

```
if(hourFormat == "24")
```

```
    config1.h24_12 = 0;
```

```
else
```

```
    config1.h24_12 = 1;
```

```
Serial.println(hourFormat);
```

```
String alarmOn = json["alarmOn"];
```

```
if (alarmOn == "yes")
```

```
{
```

```
    config1.alarmOn = 1;
```

```
}
```

```
else
```

```
{
```

```
    config1.alarmOn = 0;
```

```
}
```

```
Serial.printf("alarmOn = %d", config1.alarmOn);
```

```
Serial.println();
```

```
if (config1.alarmOn == 1)
```

```
{
```

```
    String alarm = json["alarm"];
```

```
    iEnd = alarm.indexOf(":", iStart);
```

```
    String alarm_hour = alarm.substring(iStart, iEnd);
```

```
    alarmTime.alarmHour = alarm_hour.toInt();
```

```
    Serial.printf("alarm_hour = %d", alarmTime.alarmHour);
```

```
    Serial.println();
```

```
iStart = iEnd + 1;
```

```
iEnd = alarm.indexOf(":", iStart);  
String alarm_min = alarm.substring(iStart, iEnd);  
alarmTime.alarmMin = alarm_min.toInt();  
Serial.printf("alarm_min = %d", alarmTime.alarmMin);  
Serial.println();
```

```
iStart = iEnd + 1;  
iEnd = alarm.indexOf(":", iStart);  
String alarm_sec = alarm.substring(iStart, iEnd);  
alarmTime.alarmSec = alarm_sec.toInt();  
Serial.printf("alarm_sec = %d", alarmTime.alarmSec);  
Serial.println();
```

```
iStart = iEnd + 1;  
iEnd = alarm.indexOf(",", iStart);  
String alarm_ampm = alarm.substring(iStart, iEnd);  
if(alarm_ampm == "PM")  
    alarmTime.alarmAmPm = true;  
else  
    alarmTime.alarmAmPm = false;
```

```
Serial.printf("alarm_ampm = %d", alarmTime.alarmAmPm);  
Serial.println();
```

```
}
```

```
strcpy(config1.mqttServer, json["mqttServer"]);  
strcpy(config1.mqttPort, json["mqttPort"]);  
strcpy(config1.mqttUser, json["mqttUser"]);  
strcpy(config1.mqttPass, json["mqttPass"]);
```

```
Serial.printf("mqtt_server, port, api = %s, %s, %s, %s", config1.mqttServer, config1.mqttPort, config1.mqttUser, config1.mqttPass);
```

```
    // if(json["ip"]) {
    //   Serial.println("setting custom ip from config");
    //   strcpy(static_ip, json["ip"]);
    //   strcpy(static_gw, json["gateway"]);
    //   strcpy(static_sn, json["subnet"]);
    //   Serial.println(static_ip);
    // } else {
    //   Serial.println("no custom ip in config");
    // }

  }
else
{
  Serial.println("failed to load json config");
}
}
}
}
else
{
  Serial.println("failed to mount FS");
}
//end read
}
```

```
//*****
*****//
```

```

void setup()
{
    pinMode(btnA, INPUT);
    pinMode(btnB, INPUT);
    pinMode(buz, OUTPUT);
    pinMode(CS, OUTPUT);
    digitalWrite(CS, HIGH);
    Serial.begin(115200);

    SPI.begin();
    pinMode(intPin, INPUT);
    fs_start();

    max7219_help_arr_init();
    max7219_init();
    rtc_init(SDA, SCL);
    max7219_clear_display();
    max7219_refresh_display(); //take info into LEDarr
    tckr.attach(0.05, timer_50ms); // every 50 msec
    delay(2000);
    tone(buz, 500, 300);
    Serial.println("press to enter config mode");
    delay(2000);
    tone(buz, 1000, 300);

    wifi_config();
    Serial.print("chip ID:");
    Serial.println(ESP.getChipId());
    if(wifi_auto_config())
    {
        tm* tt;

```

```

        tt = connect_ntp();
        if (tt != NULL)
            rtc_set_param(tt);
        else
            Serial.println("no timepacket received");
        mqtt_connect();
    }
    else
    {
        mqttOn = false;
        //    Serial.printf("controlREG = 0x%d\n", rtc_read(controlREG));
        Serial.println("no WiFi in RTC mode.");
    }

    attachInterrupt(digitalPinToInterrupt(intPin), handleInterrupt, FALLING);

}

//*****EXTERNAL
HANDLER*****//
void ICACHE_RAM_ATTR handleInterrupt()
{
    alarmTime.alarmNow = true;
    Serial.println("Interrupt Detected");
    tone(buz, 1000);
    printf("intPin = %d\n",digitalRead(intPin));
}

//*****
*****//

```

```
void loop()
```

```
{
```

```
    max7219_clock_disp();
```

```
}
```

```
unsigned short const font1[99][9] = { { 0x07, 0x00, 0x00, 0x00, 0x00, 0x00,
```

```
    0x00, 0x00, 0x00 }, // 0x20, Space
```

```
    { 0x07, 0x04, 0x04, 0x04, 0x04, 0x04, 0x00, 0x04, 0x00 }, // 0x21, !
```

```
    { 0x07, 0x09, 0x09, 0x12, 0x00, 0x00, 0x00, 0x00, 0x00 }, // 0x22, "
```

```
    { 0x07, 0x0a, 0x0a, 0x1f, 0x0a, 0x1f, 0x0a, 0x0a, 0x00 }, // 0x23, #
```

```
    { 0x07, 0x04, 0x0f, 0x14, 0x0e, 0x05, 0x1e, 0x04, 0x00 }, // 0x24, $
```

```
    { 0x07, 0x19, 0x19, 0x02, 0x04, 0x08, 0x13, 0x13, 0x00 }, // 0x25, %
```

```
    { 0x07, 0x04, 0x0a, 0x0a, 0x0a, 0x15, 0x12, 0x0d, 0x00 }, // 0x26, &
```

```
    { 0x07, 0x04, 0x04, 0x08, 0x00, 0x00, 0x00, 0x00, 0x00 }, // 0x27, '
```

```
    { 0x07, 0x02, 0x04, 0x08, 0x08, 0x08, 0x04, 0x02, 0x00 }, // 0x28, (
```

```
    { 0x07, 0x08, 0x04, 0x02, 0x02, 0x02, 0x04, 0x08, 0x00 }, // 0x29, )
```

```
    { 0x07, 0x04, 0x15, 0x0e, 0x1f, 0x0e, 0x15, 0x04, 0x00 }, // 0x2a, *
```

```
    { 0x07, 0x00, 0x04, 0x04, 0x1f, 0x04, 0x04, 0x00, 0x00 }, // 0x2b, +
```

```
    { 0x04, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x01, 0x02 }, // 0x2c, ,
```

```
    { 0x07, 0x00, 0x00, 0x00, 0x1f, 0x00, 0x00, 0x00, 0x00 }, // 0x2d, -
```

```
    { 0x04, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0x03, 0x00 }, // 0x2e, .
```

```
    { 0x07, 0x01, 0x01, 0x02, 0x04, 0x08, 0x10, 0x10, 0x00 }, // 0x2f, /
```

```
    { 0x07, 0x0F, 0x09, 0x09, 0x09, 0x09, 0x09, 0x0F, 0x00 }, // 0x30, 0
```

```
    { 0x07, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x00 }, // 0x31, 1
```

```
    { 0x07, 0x0F, 0x01, 0x01, 0x0F, 0x08, 0x08, 0x0F, 0x00 }, // 0x32, 2
```

```
    { 0x07, 0x0F, 0x01, 0x01, 0x0F, 0x01, 0x01, 0x0F, 0x00 }, // 0x33, 3
```

```
    { 0x07, 0x09, 0x09, 0x09, 0x0F, 0x01, 0x01, 0x01, 0x00 }, // 0x34, 4
```

```
    { 0x07, 0x0F, 0x08, 0x08, 0x0F, 0x01, 0x01, 0x0F, 0x00 }, // 0x35, 5
```

```
    { 0x07, 0x0F, 0x08, 0x08, 0x0F, 0x09, 0x09, 0x0F, 0x00 }, // 0x36, 6
```

{ 0x07, 0x0F, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x00 }, // 0x37, 7
{ 0x07, 0x0F, 0x09, 0x09, 0x0F, 0x09, 0x09, 0x0F, 0x00 }, // 0x38, 8
{ 0x07, 0x0F, 0x09, 0x09, 0x0F, 0x01, 0x01, 0x0F, 0x00 }, // 0x39, 9
{ 0x04, 0x00, 0x01, 0x01, 0x00, 0x01, 0x01, 0x00, 0x00 }, // 0x3a, :
{ 0x07, 0x00, 0x0c, 0x0c, 0x00, 0x0c, 0x04, 0x08, 0x00 }, // 0x3b, ;
{ 0x07, 0x02, 0x04, 0x08, 0x10, 0x08, 0x04, 0x02, 0x00 }, // 0x3c, <
{ 0x04, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 }, // 0x3d, =
{ 0x07, 0x08, 0x04, 0x02, 0x01, 0x02, 0x04, 0x08, 0x00 }, // 0x3e, >
{ 0x07, 0x0e, 0x11, 0x01, 0x02, 0x04, 0x00, 0x04, 0x00 }, // 0x3f, ?
{ 0x07, 0x0e, 0x11, 0x17, 0x15, 0x17, 0x10, 0x0f, 0x00 }, // 0x40, @
{ 0x07, 0x04, 0x0a, 0x11, 0x11, 0x1f, 0x11, 0x11, 0x00 }, // 0x41, A
{ 0x07, 0x1e, 0x11, 0x11, 0x1e, 0x11, 0x11, 0x1e, 0x00 }, // 0x42, B
{ 0x07, 0x0e, 0x11, 0x10, 0x10, 0x10, 0x11, 0x0e, 0x00 }, // 0x43, C
{ 0x07, 0x1E, 0x11, 0x11, 0x11, 0x11, 0x11, 0x1E, 0x00 }, // 0x44, D
{ 0x07, 0x1f, 0x10, 0x10, 0x1c, 0x10, 0x10, 0x1f, 0x00 }, // 0x45, E
{ 0x07, 0x1f, 0x10, 0x10, 0x1f, 0x10, 0x10, 0x10, 0x00 }, // 0x46, F
{ 0x07, 0x0e, 0x11, 0x10, 0x10, 0x13, 0x11, 0x0f, 0x00 }, // 0x37, G
{ 0x07, 0x11, 0x11, 0x11, 0x1f, 0x11, 0x11, 0x11, 0x00 }, // 0x48, H
{ 0x07, 0x0e, 0x04, 0x04, 0x04, 0x04, 0x04, 0x0e, 0x00 }, // 0x49, I
{ 0x07, 0x1f, 0x02, 0x02, 0x02, 0x02, 0x12, 0x0c, 0x00 }, // 0x4a, J
{ 0x07, 0x11, 0x12, 0x14, 0x18, 0x14, 0x12, 0x11, 0x00 }, // 0x4b, K
{ 0x07, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x1f, 0x00 }, // 0x4c, L
{ 0x07, 0x11, 0x1b, 0x15, 0x11, 0x11, 0x11, 0x11, 0x00 }, // 0x4d, M
{ 0x07, 0x11, 0x11, 0x19, 0x15, 0x13, 0x11, 0x11, 0x00 }, // 0x4e, N
{ 0x07, 0x0e, 0x11, 0x11, 0x11, 0x11, 0x11, 0x0e, 0x00 }, // 0x4f, O
{ 0x07, 0x1e, 0x11, 0x11, 0x1e, 0x10, 0x10, 0x10, 0x00 }, // 0x50, P
{ 0x07, 0x0e, 0x11, 0x11, 0x11, 0x15, 0x12, 0x0d, 0x00 }, // 0x51, Q
{ 0x07, 0x1e, 0x11, 0x11, 0x1e, 0x14, 0x12, 0x11, 0x00 }, // 0x52, R
{ 0x07, 0x0e, 0x11, 0x10, 0x0e, 0x01, 0x11, 0x0e, 0x00 }, // 0x53, S
{ 0x07, 0x1f, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x00 }, // 0x54, T
{ 0x07, 0x11, 0x11, 0x11, 0x11, 0x11, 0x11, 0x0e, 0x00 }, // 0x55, U

{ 0x07, 0x11, 0x11, 0x11, 0x11, 0x11, 0x0a, 0x04, 0x00 }, // 0x56, V
{ 0x07, 0x11, 0x11, 0x11, 0x15, 0x15, 0x1b, 0x11, 0x00 }, // 0x57, W
{ 0x07, 0x11, 0x11, 0x0a, 0x04, 0x0a, 0x11, 0x11, 0x00 }, // 0x58, X
{ 0x07, 0x11, 0x11, 0x0a, 0x04, 0x04, 0x04, 0x04, 0x00 }, // 0x59, Y
{ 0x07, 0x1f, 0x01, 0x02, 0x04, 0x08, 0x10, 0x1f, 0x00 }, // 0x5a, Z
{ 0x07, 0x0e, 0x08, 0x08, 0x08, 0x08, 0x08, 0x0e, 0x00 }, // 0x5b, [
{ 0x07, 0x10, 0x10, 0x08, 0x04, 0x02, 0x01, 0x01, 0x00 }, // 0x5c, \
{ 0x07, 0x0e, 0x02, 0x02, 0x02, 0x02, 0x02, 0x0e, 0x00 }, // 0x5d,]
{ 0x07, 0x04, 0x0a, 0x11, 0x00, 0x00, 0x00, 0x00, 0x00 }, // 0x5e, ^
{ 0x07, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1f, 0x00 }, // 0x5f, _
{ 0x07, 0x04, 0x04, 0x02, 0x00, 0x00, 0x00, 0x00, 0x00 }, // 0x60, `
{ 0x07, 0x00, 0x0e, 0x01, 0x0d, 0x13, 0x13, 0x0d, 0x00 }, // 0x61, a
{ 0x07, 0x10, 0x10, 0x10, 0x1c, 0x12, 0x12, 0x1c, 0x00 }, // 0x62, b
{ 0x07, 0x00, 0x00, 0x0E, 0x10, 0x10, 0x10, 0x0E, 0x00 }, // 0x63, c
{ 0x07, 0x01, 0x01, 0x01, 0x07, 0x09, 0x09, 0x07, 0x00 }, // 0x64, d
{ 0x07, 0x00, 0x00, 0x0e, 0x11, 0x1f, 0x10, 0x0f, 0x00 }, // 0x65, e
{ 0x07, 0x06, 0x09, 0x08, 0x1c, 0x08, 0x08, 0x08, 0x00 }, // 0x66, f
{ 0x07, 0x00, 0x0e, 0x11, 0x13, 0x0d, 0x01, 0x01, 0x0e }, // 0x67, g
{ 0x07, 0x10, 0x10, 0x10, 0x16, 0x19, 0x11, 0x11, 0x00 }, // 0x68, h
{ 0x05, 0x00, 0x02, 0x00, 0x06, 0x02, 0x02, 0x07, 0x00 }, // 0x69, i
{ 0x07, 0x00, 0x02, 0x00, 0x06, 0x02, 0x02, 0x12, 0x0c }, // 0x6a, j
{ 0x07, 0x10, 0x10, 0x12, 0x14, 0x18, 0x14, 0x12, 0x00 }, // 0x6b, k
{ 0x05, 0x06, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x00 }, // 0x6c, l
{ 0x07, 0x00, 0x00, 0x0a, 0x15, 0x15, 0x11, 0x11, 0x00 }, // 0x6d, m
{ 0x07, 0x00, 0x00, 0x16, 0x19, 0x11, 0x11, 0x11, 0x00 }, // 0x6e, n
{ 0x07, 0x00, 0x00, 0x0e, 0x11, 0x11, 0x11, 0x0e, 0x00 }, // 0x6f, o
{ 0x07, 0x00, 0x00, 0x1c, 0x12, 0x12, 0x1c, 0x10, 0x10 }, // 0x70, p
{ 0x07, 0x00, 0x00, 0x07, 0x09, 0x09, 0x07, 0x01, 0x01 }, // 0x71, q
{ 0x07, 0x00, 0x00, 0x16, 0x19, 0x10, 0x10, 0x10, 0x00 }, // 0x72, r
{ 0x07, 0x00, 0x00, 0x0f, 0x10, 0x0e, 0x01, 0x1e, 0x00 }, // 0x73, s
{ 0x07, 0x08, 0x08, 0x1c, 0x08, 0x08, 0x09, 0x06, 0x00 }, // 0x74, t

```

        { 0x07, 0x00, 0x00, 0x11, 0x11, 0x11, 0x13, 0x0d, 0x00 }, // 0x75, u
        { 0x07, 0x00, 0x00, 0x11, 0x11, 0x11, 0x0a, 0x04, 0x00 }, // 0x76, v
        { 0x07, 0x00, 0x00, 0x11, 0x11, 0x15, 0x15, 0x0a, 0x00 }, // 0x77, w
        { 0x07, 0x00, 0x00, 0x11, 0x0a, 0x04, 0x0a, 0x11, 0x00 }, // 0x78, x
        { 0x07, 0x00, 0x00, 0x11, 0x11, 0x0f, 0x01, 0x11, 0x0e }, // 0x79, y
        { 0x07, 0x00, 0x00, 0x1f, 0x02, 0x04, 0x08, 0x1f, 0x00 }, // 0x7a, z
        { 0x07, 0x06, 0x08, 0x08, 0x10, 0x08, 0x08, 0x06, 0x00 }, // 0x7b, {
        { 0x07, 0x04, 0x04, 0x04, 0x00, 0x04, 0x04, 0x04, 0x00 }, // 0x7c, |
        { 0x07, 0x0c, 0x02, 0x02, 0x01, 0x02, 0x02, 0x0c, 0x00 }, // 0x7d, }
        { 0x07, 0x08, 0x15, 0x02, 0x00, 0x00, 0x00, 0x00, 0x00 }, // 0x7e, ~
        { 0x07, 0x1f, 0x1f, 0x1f, 0x1f, 0x1f, 0x1f, 0x1f, 0x00 }, // 0x7f, DEL
        { 0x08, 0x00, 0x01, 0x02, 0x44, 0x28, 0x10, 0x00, 0x00 }, //0x80 custom-
        ized patterns
        { 0x08, 0x2a, 0x36, 0x3e, 0x36, 0x22, 0x22, 0x1c, 0x00 }, //0x81 custom-
        ized patterns
        { 0x08, 0x2a, 0x36, 0x3a, 0x32, 0x32, 0x22, 0x1c, 0x00 } //0x82 custom-
        ized patterns

};

```

```

unsigned short const font2[96][9] = { { 0x07, 0x00, 0x00, 0x00, 0x00, 0x00,
        0x00, 0x00, 0x00 }, // 0x20, Space
        { 0x07, 0x04, 0x04, 0x04, 0x04, 0x04, 0x00, 0x04, 0x00 }, // 0x21, !
        { 0x07, 0x09, 0x09, 0x12, 0x00, 0x00, 0x00, 0x00, 0x00 }, // 0x22, "
        { 0x07, 0x0a, 0x0a, 0x1f, 0x0a, 0x1f, 0x0a, 0x0a, 0x00 }, // 0x23, #
        { 0x07, 0x04, 0x0f, 0x14, 0x0e, 0x05, 0x1e, 0x04, 0x00 }, // 0x24, $
        { 0x07, 0x19, 0x19, 0x02, 0x04, 0x08, 0x13, 0x13, 0x00 }, // 0x25, %
        { 0x07, 0x04, 0x0a, 0x0a, 0x0a, 0x15, 0x12, 0x0d, 0x00 }, // 0x26, &
        { 0x07, 0x04, 0x04, 0x08, 0x00, 0x00, 0x00, 0x00, 0x00 }, // 0x27, '
        { 0x07, 0x02, 0x04, 0x08, 0x08, 0x08, 0x04, 0x02, 0x00 }, // 0x28, (
        { 0x07, 0x08, 0x04, 0x02, 0x02, 0x02, 0x04, 0x08, 0x00 }, // 0x29, )

```

```

{ 0x07, 0x04, 0x15, 0x0e, 0x1f, 0x0e, 0x15, 0x04, 0x00 }, // 0x2a, *
{ 0x07, 0x00, 0x04, 0x04, 0x1f, 0x04, 0x04, 0x00, 0x00 }, // 0x2b, +
{ 0x04, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x01, 0x02 }, // 0x2c, ,
{ 0x07, 0x00, 0x00, 0x00, 0x1f, 0x00, 0x00, 0x00, 0x00 }, // 0x2d, -
{ 0x04, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0x03, 0x00 }, // 0x2e, .
{ 0x07, 0x01, 0x01, 0x02, 0x04, 0x08, 0x10, 0x10, 0x00 }, // 0x2f, /
{ 0x07, 0x00, 0x00, 0x07, 0x05, 0x05, 0x05, 0x07, 0x00 }, // 0x30, 0
{ 0x07, 0x00, 0x00, 0x02, 0x02, 0x02, 0x02, 0x02, 0x00 }, // 0x31, 1
{ 0x07, 0x00, 0x00, 0x07, 0x01, 0x07, 0x04, 0x07, 0x00 }, // 0x32, 2
{ 0x07, 0x00, 0x00, 0x07, 0x01, 0x07, 0x01, 0x07, 0x00 }, // 0x33, 3
{ 0x07, 0x00, 0x00, 0x05, 0x05, 0x07, 0x01, 0x01, 0x00 }, // 0x34, 4
{ 0x07, 0x00, 0x00, 0x07, 0x04, 0x07, 0x01, 0x07, 0x00 }, // 0x35, 5
{ 0x07, 0x00, 0x00, 0x07, 0x04, 0x07, 0x05, 0x07, 0x00 }, // 0x36, 6
{ 0x07, 0x00, 0x00, 0x07, 0x01, 0x01, 0x01, 0x01, 0x00 }, // 0x37, 7
{ 0x07, 0x00, 0x00, 0x07, 0x05, 0x07, 0x05, 0x07, 0x00 }, // 0x38, 8
{ 0x07, 0x00, 0x00, 0x07, 0x05, 0x07, 0x01, 0x07, 0x00 }, // 0x39, 9
{ 0x04, 0x00, 0x03, 0x03, 0x00, 0x03, 0x03, 0x00, 0x00 }, // 0x3a, :
{ 0x07, 0x00, 0x0c, 0x0c, 0x00, 0x0c, 0x04, 0x08, 0x00 }, // 0x3b, ;
{ 0x07, 0x02, 0x04, 0x08, 0x10, 0x08, 0x04, 0x02, 0x00 }, // 0x3c, <
{ 0x04, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 }, // 0x3d, =
{ 0x07, 0x08, 0x04, 0x02, 0x01, 0x02, 0x04, 0x08, 0x00 }, // 0x3e, >
{ 0x07, 0x0e, 0x11, 0x01, 0x02, 0x04, 0x00, 0x04, 0x00 }, // 0x3f, ?
{ 0x07, 0x0e, 0x11, 0x17, 0x15, 0x17, 0x10, 0x0f, 0x00 }, // 0x40, @
{ 0x07, 0x04, 0x0a, 0x11, 0x11, 0x1f, 0x11, 0x11, 0x00 }, // 0x41, A
{ 0x07, 0x1e, 0x11, 0x11, 0x1e, 0x11, 0x11, 0x1e, 0x00 }, // 0x42, B
{ 0x07, 0x0e, 0x11, 0x10, 0x10, 0x10, 0x11, 0x0e, 0x00 }, // 0x43, C
{ 0x07, 0x1E, 0x11, 0x11, 0x11, 0x11, 0x11, 0x1E, 0x00 }, // 0x44, D
{ 0x07, 0x1f, 0x10, 0x10, 0x1c, 0x10, 0x10, 0x1f, 0x00 }, // 0x45, E
{ 0x07, 0x1f, 0x10, 0x10, 0x1f, 0x10, 0x10, 0x10, 0x00 }, // 0x46, F
{ 0x07, 0x0e, 0x11, 0x10, 0x10, 0x13, 0x11, 0x0f, 0x00 }, // 0x37, G
{ 0x07, 0x11, 0x11, 0x11, 0x1f, 0x11, 0x11, 0x11, 0x00 }, // 0x48, H

```

```

{ 0x07, 0x0e, 0x04, 0x04, 0x04, 0x04, 0x04, 0x0e, 0x00 }, // 0x49, I
{ 0x07, 0x1f, 0x02, 0x02, 0x02, 0x02, 0x12, 0x0c, 0x00 }, // 0x4a, J
{ 0x07, 0x11, 0x12, 0x14, 0x18, 0x14, 0x12, 0x11, 0x00 }, // 0x4b, K
{ 0x07, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x1f, 0x00 }, // 0x4c, L
{ 0x07, 0x11, 0x1b, 0x15, 0x11, 0x11, 0x11, 0x11, 0x00 }, // 0x4d, M
{ 0x07, 0x11, 0x11, 0x19, 0x15, 0x13, 0x11, 0x11, 0x00 }, // 0x4e, N
{ 0x07, 0x0e, 0x11, 0x11, 0x11, 0x11, 0x11, 0x0e, 0x00 }, // 0x4f, O
{ 0x07, 0x1e, 0x11, 0x11, 0x1e, 0x10, 0x10, 0x10, 0x00 }, // 0x50, P
{ 0x07, 0x0e, 0x11, 0x11, 0x11, 0x15, 0x12, 0x0d, 0x00 }, // 0x51, Q
{ 0x07, 0x1e, 0x11, 0x11, 0x1e, 0x14, 0x12, 0x11, 0x00 }, // 0x52, R
{ 0x07, 0x0e, 0x11, 0x10, 0x0e, 0x01, 0x11, 0x0e, 0x00 }, // 0x53, S
{ 0x07, 0x1f, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x00 }, // 0x54, T
{ 0x07, 0x11, 0x11, 0x11, 0x11, 0x11, 0x11, 0x0e, 0x00 }, // 0x55, U
{ 0x07, 0x11, 0x11, 0x11, 0x11, 0x11, 0x0a, 0x04, 0x00 }, // 0x56, V
{ 0x07, 0x11, 0x11, 0x11, 0x15, 0x15, 0x1b, 0x11, 0x00 }, // 0x57, W
{ 0x07, 0x11, 0x11, 0x0a, 0x04, 0x0a, 0x11, 0x11, 0x00 }, // 0x58, X
{ 0x07, 0x11, 0x11, 0x0a, 0x04, 0x04, 0x04, 0x04, 0x00 }, // 0x59, Y
{ 0x07, 0x1f, 0x01, 0x02, 0x04, 0x08, 0x10, 0x1f, 0x00 }, // 0x5a, Z
{ 0x07, 0x0e, 0x08, 0x08, 0x08, 0x08, 0x08, 0x0e, 0x00 }, // 0x5b, [
{ 0x07, 0x10, 0x10, 0x08, 0x04, 0x02, 0x01, 0x01, 0x00 }, // 0x5c, \
{ 0x07, 0x0e, 0x02, 0x02, 0x02, 0x02, 0x02, 0x0e, 0x00 }, // 0x5d, ]
{ 0x07, 0x04, 0x0a, 0x11, 0x00, 0x00, 0x00, 0x00, 0x00 }, // 0x5e, ^
{ 0x07, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1f, 0x00 }, // 0x5f, _
{ 0x07, 0x04, 0x04, 0x02, 0x00, 0x00, 0x00, 0x00, 0x00 }, // 0x60, `
{ 0x07, 0x00, 0x0e, 0x01, 0x0d, 0x13, 0x13, 0x0d, 0x00 }, // 0x61, a
{ 0x07, 0x10, 0x10, 0x10, 0x1c, 0x12, 0x12, 0x1c, 0x00 }, // 0x62, b
{ 0x07, 0x00, 0x00, 0x0E, 0x10, 0x10, 0x10, 0x0E, 0x00 }, // 0x63, c
{ 0x07, 0x01, 0x01, 0x01, 0x07, 0x09, 0x09, 0x07, 0x00 }, // 0x64, d
{ 0x07, 0x00, 0x00, 0x0e, 0x11, 0x1f, 0x10, 0x0f, 0x00 }, // 0x65, e
{ 0x07, 0x06, 0x09, 0x08, 0x1c, 0x08, 0x08, 0x08, 0x00 }, // 0x66, f
{ 0x07, 0x00, 0x0e, 0x11, 0x13, 0x0d, 0x01, 0x01, 0x0e }, // 0x67, g

```

```

    { 0x07, 0x10, 0x10, 0x10, 0x16, 0x19, 0x11, 0x11, 0x00 }, // 0x68, h
    { 0x05, 0x00, 0x02, 0x00, 0x06, 0x02, 0x02, 0x07, 0x00 }, // 0x69, i
    { 0x07, 0x00, 0x02, 0x00, 0x06, 0x02, 0x02, 0x12, 0x0c }, // 0x6a, j
    { 0x07, 0x10, 0x10, 0x12, 0x14, 0x18, 0x14, 0x12, 0x00 }, // 0x6b, k
    { 0x05, 0x06, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x00 }, // 0x6c, l
    { 0x07, 0x00, 0x00, 0x0a, 0x15, 0x15, 0x11, 0x11, 0x00 }, // 0x6d, m
    { 0x07, 0x00, 0x00, 0x16, 0x19, 0x11, 0x11, 0x11, 0x00 }, // 0x6e, n
    { 0x07, 0x00, 0x00, 0x0e, 0x11, 0x11, 0x11, 0x0e, 0x00 }, // 0x6f, o
    { 0x07, 0x00, 0x00, 0x1c, 0x12, 0x12, 0x1c, 0x10, 0x10 }, // 0x70, p
    { 0x07, 0x00, 0x00, 0x07, 0x09, 0x09, 0x07, 0x01, 0x01 }, // 0x71, q
    { 0x07, 0x00, 0x00, 0x16, 0x19, 0x10, 0x10, 0x10, 0x00 }, // 0x72, r
    { 0x07, 0x00, 0x00, 0x0f, 0x10, 0x0e, 0x01, 0x1e, 0x00 }, // 0x73, s
    { 0x07, 0x08, 0x08, 0x1c, 0x08, 0x08, 0x09, 0x06, 0x00 }, // 0x74, t
    { 0x07, 0x00, 0x00, 0x11, 0x11, 0x11, 0x13, 0x0d, 0x00 }, // 0x75, u
    { 0x07, 0x00, 0x00, 0x11, 0x11, 0x11, 0x0a, 0x04, 0x00 }, // 0x76, v
    { 0x07, 0x00, 0x00, 0x11, 0x11, 0x15, 0x15, 0x0a, 0x00 }, // 0x77, w
    { 0x07, 0x00, 0x00, 0x11, 0x0a, 0x04, 0x0a, 0x11, 0x00 }, // 0x78, x
    { 0x07, 0x00, 0x00, 0x11, 0x11, 0x0f, 0x01, 0x11, 0x0e }, // 0x79, y
    { 0x07, 0x00, 0x00, 0x1f, 0x02, 0x04, 0x08, 0x1f, 0x00 }, // 0x7a, z
    { 0x07, 0x06, 0x08, 0x08, 0x10, 0x08, 0x08, 0x06, 0x00 }, // 0x7b, {
    { 0x07, 0x04, 0x04, 0x04, 0x00, 0x04, 0x04, 0x04, 0x00 }, // 0x7c, |
    { 0x07, 0x0c, 0x02, 0x02, 0x01, 0x02, 0x02, 0x0c, 0x00 }, // 0x7d, }
    { 0x07, 0x08, 0x15, 0x02, 0x00, 0x00, 0x00, 0x00, 0x00 }, // 0x7e, ~
    { 0x07, 0x1f, 0x1f, 0x1f, 0x1f, 0x1f, 0x1f, 0x1f, 0x00 } // 0x7f, DEL
};

//void max7219_init();

//void max7219_set_brightness(unsigned short br);

//*****MAX7219
INITIALIATION*****//

void max7219_init() //all MAX7219 init

```

```

{
    unsigned short i, j;
    for (i = 0; i < 7; i++)
    {
        digitalWrite(CS, LOW);
        delayMicroseconds(1);
        for (j = 0; j < anzMAX; j++)
        {
            SPI.write(initArr[i][0]); //register
            SPI.write(initArr[i][1]); //value
        }
        digitalWrite(CS, HIGH);
        max7219_set_brightness(config1.brightness);
    }
}

```

```

//*****MAX7219
BRIGHTNESS*****//
void max7219_set_brightness(unsigned short br) //brightness MAX7219
{
    unsigned short j;
    if (br < 16)
    {
        digitalWrite(CS, LOW);
        delayMicroseconds(1);
        for (j = 0; j < anzMAX; j++)
        {
            SPI.write(0x0A); //register
            SPI.write(br); //value
        }
        digitalWrite(CS, HIGH);
    }
}

```

```

    }

}

//*****

*****

void max7219_help_arr_init(void) //helperarray init
{
    unsigned short i, j, k;
    j = 0;
    k = 0;
    for (i = 0; i < anzMAX * 8; i++) //k = 0 to 3, j = 0 to 7
    {
        helpArrPos[i] = (1 << j); //bitmask
        helpArrMAX[i] = k;
        j++;
        if (j > 7)
        {
            j = 0;
            k++;
        }
    }
}

//*****

*****

void max7219_clear_display() //clear all
{
    unsigned short i, j;
    for (i = 0; i < 8; i++) //8 rows
    {
        digitalWrite(CS, LOW);
        delayMicroseconds(1);
        for (j = anzMAX; j > 0; j--) {

```

```

        LEDarr[j - 1][i] = 0;    //LEDarr clear
        SPI.write(i + 1);        //current row
        SPI.write(LEDarr[j - 1][i]);
    }
    digitalWrite(CS, HIGH);
}

}

//*****
*****

void max7219_rotate_90() // for Generic displays
{
    for (uint8_t k = anzMAX; k > 0; k--)
    {

        uint8_t i, j, m, imask, jmask;
        uint8_t tmp[8]={0,0,0,0,0,0,0,0};
        for ( i = 0, imask = 0x01; i < 8; i++, imask <=< 1)
        {
            for (j = 0, jmask = 0x01; j < 8; j++, jmask <=< 1)
            {
                if (LEDarr[k-1][i] & jmask)
                {
                    tmp[j] |= imask;
                }
            }
        }
        for(m=0; m<8; m++)
        {
            LEDarr[k-1][m]=tmp[m];
        }
    }
}

```

```

    }

    /*******
*****

void max7219_refresh_display() //take info into LEDarr
{
    unsigned short i, j;

#ifdef ROTATE_90
    max7219_rotate_90();
#endif

    for (i = 0; i < 8; i++)    //8 rows
    {
        digitalWrite(CS, LOW);
        delayMicroseconds(1);
        for (j = 1; j <= anzMAX; j++)
        {
            SPI.write(i + 1); //current row

#ifdef REVERSE_HORIZONTAL
            SPI.setBitOrder(LSBFIRST);    // bitorder for reverse columns
#endif

#ifdef REVERSE_VERTICAL
            SPI.write(LEDarr[j - 1][7-i]);
#else
            SPI.write(LEDarr[j - 1][i]);
#endif

#ifdef REVERSE_HORIZONTAL
            SPI.setBitOrder(MSBFIRST);    // reset bitorder

```

```

#endif
    }
    digitalWrite(CS, HIGH);
}
}

//*****
*****

void max7219_char_to_arr(unsigned short ch, int PosX, short PosY)
{ //characters into arr
    int i, j, k, l, m, o1, o2, o3, o4; //in LEDarr
    PosX++;
    k = ch - 32;          //ASCII position in font
    if ((k >= 0) && (k < 99))    //character found in font?
    {
        o4 = font1[k][0];      //character width
        o3 = 1 << (o4 - 2);
        for (i = 0; i < o4; i++)
        {
            if (((PosX - i <= maxPosX) && (PosX - i >= 0))
                && ((PosY > -8) && (PosY < 8))) //within matrix?
            {
                o1 = helpArrPos[PosX - i];
                o2 = helpArrMAX[PosX - i];
                for (j = 0; j < 8; j++)
                {
                    if (((PosY >= 0) && (PosY <= j)) || ((PosY < 0) && (j < PosY + 8)))
//scroll vertical
                    {
                        l = font1[k][j + 1];
                        m = (l & (o3 >> i)); //e.g. o4=7 0zzzzz0, o4=4 0zz0

```

```

        if (m > 0)
            LEDarr[o2][j - PosY] = LEDarr[o2][j - PosY] | (o1); //set point
        else
            LEDarr[o2][j - PosY] = LEDarr[o2][j - PosY] & (~o1); //clear
point
    }
}
}
}
}
}

//*****
*****

void max7219_char_to_to_arr(unsigned short ch, int PosX, short PosY)
{ //characters into arr
    int i, j, k, l, m, o1, o2, o3, o4; //in LEDarr
    PosX++;
    k = ch - 32;                //ASCII position in font
    if ((k >= 0) && (k < 96))    //character found in font?
    {
        o4 = font2[k][0];        //character width
        o3 = 1 << (o4 - 2);
        for (i = 0; i < o4; i++)
        {
            if (((PosX - i <= maxPosX) && (PosX - i >= 0))
                && ((PosY > -8) && (PosY < 8))) //within matrix?
            {
                o1 = helpArrPos[PosX - i];
                o2 = helpArrMAX[PosX - i];
                for (j = 0; j < 8; j++) {

```

```

        if (((PosY >= 0) && (PosY <= j)) || ((PosY < 0) && (j < PosY + 8)))
//scroll vertical
    {
        l = font2[k][j + 1];
        m = (l & (o3 >> i)); //e.g. o4=7 0zzzzz0, o4=4 0zz0
        if (m > 0)
            LEDarr[o2][j - PosY] = LEDarr[o2][j - PosY] | (o1); //set point
        else
            LEDarr[o2][j - PosY] = LEDarr[o2][j - PosY] & (~o1); //clear
point
    }
}
}
}
}
}
}
}

```

```

//*****
*****

```

```

void max7219_clock_disp()
{
    unsigned int sec1 = 0, sec2 = 0, min1 = 0, min2 = 0, hour1 = 0, hour2 = 0;
    unsigned int sec11 = 0, sec12 = 0, sec21 = 0, sec22 = 0;
    unsigned int min11 = 0, min12 = 0, min21 = 0, min22 = 0;
    unsigned int hour11 = 0, hour12 = 0, hour21 = 0, hour22 = 0;
    signed int x = 0; //x1,x2;
    signed int y = 0, y1 = 0, y2 = 0, y3=0;
    bool updown = false;
    unsigned int sc1 = 0, sc2 = 0, sc3 = 0, sc4 = 0, sc5 = 0, sc6 = 0;
    bool f_scrollend_y = false;

```

```

unsigned int f_scroll_x = false;
bool callsignOn = false;
zPosX = maxPosX;
dPosX = -8;
// x=0; x1=0; x2=0;

max7219_refresh_display();
updown = true;
if (updown == false)
{
    y2 = -9;
    y1 = 8;
}
if (updown == true)
{ //scroll up to down
    y2 = 8;
    y1 = -8;
}
while (true)
{
    yield();

    if ( MEZ.hour12==0 && MEZ.min12==20 && MEZ.sec12==0 ) //syn-
cronisize RTC every day 00:20:00
    {
        max7219_clear_display();
        delay(500);
        ESP.restart();
    }
    if (fTckr1s == true)    // flag 1sec
    {

```

```

rtc_read_param();
sec1 = MEZ.sec1;
sec2 = MEZ.sec2;
min1 = MEZ.min1;
min2 = MEZ.min2;
hour1 = MEZ.hour1;
hour2 = MEZ.hour2;
//      if(utc)
//      ;

y = y2;
sc1 = 1;
sec1++;
if (sec1 == 10)
{
    sc2 = 1;
    sec2++;
    sec1 = 0;
}
if (sec2 == 6)
{
    min1++;
    sec2 = 0;
    sc3 = 1;
}
if (min1 == 10)
{
    min2++;
    min1 = 0;
    sc4 = 1;
}

```

```
if (min2 == 6)
{
    hour1++;
    min2 = 0;
    sc5 = 1;
}
if (hour1 == 10)
{
    hour2++;
    hour1 = 0;
    sc6 = 1;
}
if ((hour2 == 2) && (hour1 == 4))
{
    hour1 = 0;
    hour2 = 0;
    sc6 = 1;
}
```

```
sec11 = sec12;
sec12 = sec1;
sec21 = sec22;
sec22 = sec2;
min11 = min12;
min12 = min1;
min21 = min22;
min22 = min2;
hour11 = hour12;
hour12 = hour1;
hour21 = hour22;
hour22 = hour2;
```

```

fTckr1s = false;

if (MEZ.sec12 == 10)
    f_scroll_x = true;

if (MEZ.sec12 == 30)
    if (config1.showCall)
        callsignOn = true;
} // end 1s

if (fTckr50ms == true)
{
    fTckr50ms = false;
    if (f_scroll_x == true)
    {
        zPosX++;
        dPosX++;
        if (dPosX == 125) //101
            zPosX = 0;
        if (zPosX == maxPosX)
        {
            f_scroll_x = false;
            dPosX = -8;
        }
    }
}

if(utc.utcAct)
{
    if(config1.h24_12 == 1)
    {
        if(utc.utcAmPm) //pm

```

```

        max7219_char_to_arr(0x82, zPosX - 34, 0); //utc pm icon
    else //am
        max7219_char_to_arr(0x81, zPosX - 34, 0); //utc am icon
    }
else
    max7219_char_to_arr('U', zPosX - 34, 0);
}
else
{
    if(config1.h24_12 == 1)
    {
        if(config1.amPm == 0)
            max7219_char_to_arr('A', zPosX - 34, 0);
        else
            max7219_char_to_arr('P', zPosX - 34, 0);
    }
    else
        max7219_char_to_arr(0x80, zPosX - 34, 0);
}

```

```

if (sc1 == 1)
{
    if(updown == 1)
        y--;
    else
        y++;
    y3 = y;

    if (y3 > 0)
        y3 = 0;
}

```

```

max7219_char_to_to_arr(48 + sec12, zPosX - 27, y3);
max7219_char_to_to_arr(48 + sec11, zPosX - 27, y + y1);

if (y == 0)
{
    sc1 = 0;
    f_scrollend_y = true;
}
}
else
    max7219_char_to_to_arr(48 + sec1, zPosX - 27, 0);

if (sc2 == 1)
{
    max7219_char_to_to_arr(48 + sec22, zPosX - 23, y3);
    max7219_char_to_to_arr(48 + sec21, zPosX - 23, y + y1);
    if (y == 0)
        sc2 = 0;
}
else
    max7219_char_to_to_arr(48 + sec2, zPosX - 23, 0);

if (sc3 == 1)
{
    max7219_char_to_arr(48 + min12, zPosX - 18, y);
    max7219_char_to_arr(48 + min11, zPosX - 18, y + y1);
    if (y == 0)
        sc3 = 0;
}
else

```

```
max7219_char_to_arr(48 + min1, zPosX - 18, 0);
```

```
if (sc4 == 1)
```

```
{
```

```
    max7219_char_to_arr(48 + min22, zPosX - 13, y);
```

```
    max7219_char_to_arr(48 + min21, zPosX - 13, y + y1);
```

```
    if (y == 0)
```

```
        sc4 = 0;
```

```
}
```

```
else
```

```
    max7219_char_to_arr(48 + min2, zPosX - 13, 0);
```

```
max7219_char_to_arr(':', zPosX - 10 + x, 0);
```

```
if (sc5 == 1)
```

```
{
```

```
    max7219_char_to_arr(48 + hour12, zPosX - 4, y);
```

```
    max7219_char_to_arr(48 + hour11, zPosX - 4, y + y1);
```

```
    if (y == 0)
```

```
        sc5 = 0;
```

```
}
```

```
else
```

```
    max7219_char_to_arr(48 + hour1, zPosX - 4, 0);
```

```
if (sc6 == 1)
```

```
{
```

```
    max7219_char_to_arr(48 + hour22, zPosX + 1, y);
```

```
    max7219_char_to_arr(48 + hour21, zPosX + 1, y + y1);
```

```
    if (y == 0)
```

```
        sc6 = 0;
```

```
}
```

else

max7219_char_to_arr(48 + hour2, zPosX + 1, 0);

//day of the week

max7219_char_to_arr(' ', dPosX+5, 0);

max7219_char_to_arr(weekdayArray[MEZ.weekday][0], dPosX - 1, 0);

max7219_char_to_arr(weekdayArray[MEZ.weekday][1], dPosX - 7, 0);

max7219_char_to_arr(weekdayArray[MEZ.weekday][2], dPosX - 13, 0);

max7219_char_to_arr(weekdayArray[MEZ.weekday][3], dPosX - 19, 0);

//month

max7219_char_to_arr(monthArray[MEZ.mon12 - 1][0], dPosX - 24, 0);

//39

max7219_char_to_arr(monthArray[MEZ.mon12 - 1][1], dPosX - 28, 0);

//43

max7219_char_to_arr(monthArray[MEZ.mon12 - 1][2], dPosX - 34, 0);

//49

max7219_char_to_arr(monthArray[MEZ.mon12 - 1][3], dPosX - 40, 0);

//55

max7219_char_to_arr(monthArray[MEZ.mon12 - 1][4], dPosX - 46, 0);

//61

//date

max7219_char_to_arr(48 + MEZ.date2, dPosX - 55, 0); //24

max7219_char_to_arr(48 + MEZ.date1, dPosX - 61, 0); //30

//year

max7219_char_to_arr('2', dPosX - 72, 0); //68

max7219_char_to_arr('0', dPosX - 78, 0); //74

```
max7219_char_to_arr(48 + MEZ.year2, dPosX - 84, 0); //80  
max7219_char_to_arr(48 + MEZ.year1, dPosX - 90, 0); //86
```

```
max7219_refresh_display(); //alle 50ms
```

```
if (f_scrollend_y == true)  
{  
    f_scrollend_y = false;  
}  
if (callsignOn)  
{  
    callsignOn = false;  
    //max7219_scroll_down("BD5XM", 31, 6, 10);  
    max7219_callsign_amination_1();  
    max7219_callsign_amination_2();  
}
```

```
} //end 50ms
```

```
else if (mqttReceived)  
{  
    if(mqttMsgType == 0)  
    {  
        max7219_scroll_left(mqttMsg0, 0, 0, 3);  
    }  
    else  
        max7219_scroll_left(mqttMsg1, 0, 0, 3);
```

```
    mqttReceived = false;
```

```
}  
else if (mqttButton && mqttOn)  
{
```

```

    Serial.printf("wifi and mqtt is ON");
    if(mqttMsgType == 0)
    {
        max7219_scroll_left(mqttMsg0, 0, 0, 3);
    }
    else
        max7219_scroll_left(mqttMsg1, 0, 0, 3);
    mqttButton = false;

}
if (y == 0)
{
    // do something else

}
} //end while(true)
//this section can not be reached
}

void max7219_callsign_amination_1()
{
    max7219_display_static(config1.callsign, maxPosX - 3, -1);
    max7219_scroll_up(config1.callsign, maxPosX - 3, -1, 5);
    max7219_scroll_down(config1.callsign, maxPosX - 3, 7, 5);
    max7219_display_static(config1.callsign, maxPosX - 3, -1);
    delay(1000);
    // max7219_display_static("BD5XM", 31, -1);
}

void max7219_callsign_amination_2()
{

```

```

yield();
max7219_display_one_by_one(config1.callsign, maxPosX - 3, 0, 8);
max7219_display_static(config1.callsign, maxPosX - 3, -0);
delay(1000);
max7219_clear_display();
delay(500);
max7219_display_static(config1.callsign, maxPosX - 3, 0);
delay(1000);
max7219_scroll_down_one_by_one(config1.callsign, maxPosX - 3, 7, 2);
}

```

```

void max7219_display_static(char *p, int8_t x, int8_t y)
{
    max7219_clear_display();

    for(int index = 0; index < strlen(p); index++)
    {
        max7219_char_to_arr(p[index], x - wordSpace * index, y);
    }
    max7219_refresh_display(); //alle 50ms
}

```

```

void max7219_display_one_by_one(char *p, int8_t x, int8_t y, uint8_t sp)
{
    bool completed = false;
    uint16_t counter = 0;
    uint8_t index = 0;
    while(!completed)
    {
        yield();

```

```

    if (fTckr50ms == true)
    {
        counter++;
        fTckr50ms = false;
    }
    if (counter >= sp)
    {
        max7219_clear_display();
        max7219_char_to_arr(p[index], x - wordSpace * index, y);
        max7219_refresh_display();
        counter = 0;
        index++;
    }
    if(index > strlen(p))
        completed = true;
}

void max7219_scroll_down_one_by_one(char *p, int8_t x, int8_t y, uint8_t sp)
{
    bool completed = false;
    uint16_t counter = 0;
    int8_t index = 0, y1 = y;
    max7219_clear_display();
    while(!completed)
    {
        yield();
        if (fTckr50ms == true)
        {
            counter++;
            fTckr50ms = false;

```

```
}
```

```
if (counter >= sp)
```

```
{
```

```
    //max7219_clear_display();
```

```
    max7219_char_to_arr(p[index], x - wordSpace * index, y1);
```

```
    max7219_refresh_display();
```

```
    //Serial.printf("conter = %d\n", counter);
```

```
    counter = 0;
```

```
    y1--;
```

```
    //Serial.printf("y1 = %d\n", y1);
```

```
    if(y1 < 0)
```

```
    {
```

```
        //Serial.println("y1 < -8");
```

```
        index++;
```

```
        if(index > strlen(p))
```

```
            completed = true;
```

```
        y1 = y;
```

```
    }
```

```
}
```

```
}
```

```
}
```

```
void max7219_scroll_down(char *p, int8_t x, int8_t y, uint8_t sp)
```

```
{
```

```
    static int8_t y1 = y;
```

```
    bool completed = false;
```

```
    uint16_t counter = 0;
```

```

while(!completed)
{
    yield();
    if (fTckr50ms == true)
    {
        counter++;
        fTckr50ms = false;

    }
    if (counter >= sp)
    {
        max7219_clear_display();

        for(int index = 0; index < strlen(p); index++)
        {
            max7219_char_to_arr(p[index], x - wordSpace * index, y1);
        }
        max7219_refresh_display(); //alle 50ms
        counter = 0;
        y1--;
        if (y1 < -8)
        {
            y1 = y;
            completed = true;
        }
    }
}

```

```

void max7219_scroll_up(char *p, int8_t x, int8_t y, uint8_t sp)
{

```

```

static int8_t y1 = y;
bool completed = false;
uint16_t counter = 0;
while(!completed)
{
    yield();
    if (fTckr50ms == true)
    {
        counter++;
        fTckr50ms = false;

    }
    if (counter >= sp)
    {
        max7219_clear_display();
//      max7219_char_to_arr(p[0], x, y1);
//      max7219_char_to_arr(p[1], x - wordSpace, y1);
//      max7219_char_to_arr(p[2], x - wordSpace * 2, y1);
//      max7219_char_to_arr(p[3], x - wordSpace * 3, y1);
//      max7219_char_to_arr(p[4], x - wordSpace * 4, y1);
        for(int index = 0; index < strlen(p); index++)
        {
            max7219_char_to_arr(p[index], x - wordSpace * index, y1);
        }
        max7219_refresh_display(); //alle 50ms
        counter = 0;
        y1++;
        if (y1 > 8)
        {
            y1 = y;

```

```

        completed = true;
    }
}
}
}

```

```

void max7219_scroll_left(char *p, int8_t x, int8_t y, uint8_t sp)
{

```

```

    static int16_t x1 = x;
    bool completed = false;
    uint16_t counter = 0;
    Serial.printf("strlen = %d\n", strlen(p));
    while(!completed)
    {
        yield();
        if (fTckr50ms == true)
        {
            counter++;
            fTckr50ms = false;

        }

        if (counter >= sp)
        {
            max7219_clear_display();
            //    max7219_char_to_arr(p[0], x1, y);
            //    max7219_char_to_arr(p[1], x1 - wordSpace, y);
            //    max7219_char_to_arr(p[2], x1 - wordSpace * 2, y);
            //    max7219_char_to_arr(p[3], x1 - wordSpace * 3, y);
            //    max7219_char_to_arr(p[4], x1 - wordSpace * 4, y);

```

```

    for(int index = 0; index < strlen(p) + 1; index++)
    {
        max7219_char_to_arr(p[index], x1 - wordSpace * index, y);
    }
    max7219_refresh_display(); //alle 50ms
    counter = 0;
    x1++;
    if (x1 > ((strlen(p) * 6) + 40))
    {
        x1 = x;
        completed = true;
    }
}
}
}

```

```

void max7219_scroll_right(char *p, int8_t x, int8_t y, uint8_t sp)
{

    static int8_t x1 = x;
    bool completed = false;
    uint16_t counter = 0;
    while(!completed)
    {
        yield();
        if (fTckr50ms == true)
        {
            counter++;
            fTckr50ms = false;
        }
    }
}

```

```

    if (counter >= sp)
    {
        max7219_clear_display();
//      max7219_char_to_arr(p[0], x1, y);
//      max7219_char_to_arr(p[1], x1 - wordSpace, y);
//      max7219_char_to_arr(p[2], x1 - wordSpace * 2, y);
//      max7219_char_to_arr(p[3], x1 - wordSpace * 3, y);
//      max7219_char_to_arr(p[4], x1 - wordSpace * 4, y);
        for(int index = 0; index < strlen(p); index++)
        {
            max7219_char_to_arr(p[index], x1 - wordSpace * index, y);
        }
        max7219_refresh_display(); //alle 50ms
        counter = 0;
        x1--;
        if (x1 < -1)
        {
            x1 = x;
            completed = true;
        }
    }
}

String topic0 = "/MatrixClock/msg" + String(ESP.getChipId());
String topic1 = "/MatrixClock/other" + String(ESP.getChipId());

void mqtt_connect()
{
    //Serial.printf("\nmqtt connecting to %s \n", config1.mqttServer);
    //client.setServer(config1.mqttServer, 1800);
    client.setKeepAlive (120);

```

```

client.setTimeout(20);
client.setServer(config1.mqttServer, atoi(config1.mqttPort));
client.setCallback(mqtt_callback);
while (!client.connected())
{
    Serial.println("Connecting to MQTT...");

    // if (client.connect("MatrixClock")) //if (client.connect("ESP8266Client", mqtt-
tUser, mqttPassword ))
        if (client.connect("MatrixClock", config1.mqttUser, config1.mqttPass ))
        {

            Serial.println("connected");

        } else {

            Serial.print("failed with state ");
            Serial.print(client.state());

        }
}
Serial.printf("topic0 = %s\n", topic0.c_str());
Serial.printf("topic1 = %s\n", topic1.c_str());
client.subscribe(topic0.c_str());
client.subscribe(topic1.c_str());
//if its chtos nayde z mene sotka(066577784)
mqttOn = true;

}

bool mqtt_reconnect()

```

```

{
  // Loop until we're reconnected
  while (!client.connected())
  {
    yield();
    Serial.print("Attempting MQTT connection...");
    // Attempt to connect
    if (client.connect("MatrixClock"))
    {
      Serial.println("connected");
      // Once connected, publish an announcement...
      client.publish(topic0.c_str(),"reconnected");
      // ... and resubscribe
      client.subscribe(topic0.c_str());
      client.subscribe(topic1.c_str());
      return client.connected();
    }
    else
    {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      return !client.connected();
      // Wait 5 seconds before retrying
      //delay(5000);
    }
  }
}

```

```

void mqtt_callback(char* topic, byte* payload, unsigned int length)

```

```

{
  mqttReceived = true;

  String myString = String(topic);
  Serial.print("Message arrived in topic: ");
  Serial.println(topic);
  Serial.print("Message:");
  if(myString == topic0.c_str())
  {
    tone(buz, 800, 600);
    int len = strlen(mqttMsg0);
    for (int i = 0; i < len; i++)
      mqttMsg0[i] = '\0';

    for (int i = 0; i < length; i++)
    {
      //Serial.print((char)payload[i]);
      mqttMsg0[i] = (char)payload[i];
      mqttMsgType = 0;
      Serial.print(mqttMsg0[i]);
    }
  }

  else
  {
    tone(buz, 600, 100);
    int len = strlen(mqttMsg1);
    for (int i = 0; i < len; i++)
      mqttMsg1[i] = '\0';

    for (int i = 0; i < length; i++)

```

```

    {
        //Serial.print((char)payload[i]);
        mqttMsg1[i] = (char)payload[i];
        mqttMsgType = 1;
        Serial.print(mqttMsg1[i]);
    }
}

Serial.println();
Serial.println("-----");
}

void mqtt_keep_connect()
{
    if (!client.connected())
    {
        {
            long now = millis();
            if (now - lastReconnectAttempt > 5000)
            {
                //lastReconnectAttempt = now;
                // Attempt to reconnect
                if (!mqtt_reconnect())
                {
                    //lastReconnectAttempt = 0;
                    lastReconnectAttempt = now;
                }
            }
        }
    }
    else
    {
        // Client connected
    }
}

```

```

        client.loop();
    }

}

void rtc_init(unsigned char sda, unsigned char scl)
{
    Wire.begin(sda, scl);
    //  rtc_write(controlREG, 0x00);
}

//*****
*****

// BCD Code

//*****
*****

unsigned char rtc_dec2bcd(unsigned char x)
{ //value 0...99
    unsigned char z, e, r;
    e = x % 10;
    z = x / 10;
    z = z << 4;
    r = e | z;
    return (r);
}

unsigned char rtc_bcd2dec(unsigned char x) { //value 0...99
    int z, e;
    e = x & 0x0F;
    z = x & 0xF0;
    z = z >> 4;
    z = z * 10;
    return (z + e);
}

```

```

//*****

*****

// RTC I2C Code

//*****

*****

unsigned char rtc_read(unsigned char regaddress)
{
    Wire.beginTransmission(DS3231_ADDRESS);
    Wire.write(regaddress);
    Wire.endTransmission();
    Wire.requestFrom((unsigned char) DS3231_ADDRESS, (unsigned char) 1);
    return (Wire.read());
}

void rtc_write(unsigned char regaddress, unsigned char value)
{
    Wire.beginTransmission(DS3231_ADDRESS);
    Wire.write(regaddress);
    Wire.write(value);
    Wire.endTransmission();
}

//*****

*****

//unsigned char rtc_get_second()
unsigned char rtc_get_second()
{
    return (rtc_bcd2dec(rtc_read(secondREG)));
}

unsigned char rtc_get_minute()
{
    return (rtc_bcd2dec(rtc_read(minuteREG)));
}

```

```
}
```

```
//unsigned char rtc_hour ()
```

```
unsigned char rtc_get_hour()
```

```
{
```

```
    uint8_t temp1 = rtc_read(hourREG);
```

```
    uint8_t temp2 = temp1 & 0x40;
```

```
    uint8_t temp3 = temp1 & 0x20;
```

```
    if(temp2)
```

```
    {
```

```
        config1.h24_12 = 1; //12h
```

```
        if(temp3)
```

```
            config1.amPm = 1; //pm
```

```
        else
```

```
            config1.amPm = 0; //am
```

```
        return (rtc_bcd2dec(temp1 & 0x1F));
```

```
    }
```

```
    else
```

```
    {
```

```
        config1.h24_12 = 0; //24h
```

```
        return (rtc_bcd2dec(temp1 & 0x3F));
```

```
    }
```

```
}
```

```
//unsigned char rtc_get_weekday()
```

```
unsigned char rtc_get_weekday()
```

```
{
```

```
    return (rtc_bcd2dec(rtc_read(weekREG)));
```

```
}
```

```
//unsigned char rtc_get_date()
unsigned char rtc_get_date()
{
    return (rtc_bcd2dec(rtc_read(dateREG)));
}

//unsigned char rtc_get_month()
unsigned char rtc_get_month()
{
    return (rtc_bcd2dec(rtc_read(monthREG)));
}

//unsigned char rtc_get_year()
unsigned char rtc_get_year()
{
    return (rtc_bcd2dec(rtc_read(yearREG)));
}

//void rtc_set_second(unsigned char sec)
void rtc_set_second(unsigned char sec)
{
    rtc_write(secondREG, (rtc_dec2bcd(sec)));
}

//void rtc_set_minute(unsigned char min)
void rtc_set_minute(unsigned char min)
{
    rtc_write(minuteREG, (rtc_dec2bcd(min)));
}
```

```

//void rtc_hour(unsigned char hour)
void rtc_set_hour(unsigned char hour) //bit 6 of register 0x02 sets the 12/24 for-
mat 1<<6 | rtc_dec2bcd(hour)
{
    if(config1.h24_12) //12h format
    {
        uint8_t temp1 = 1 << 6;
        uint8_t temp2;
        if(config1.amPm) //pm
        {
            temp2 |= 1 << 5;
            rtc_write(hourREG, (rtc_dec2bcd(hour)) | temp1 | temp2); //must set bit5 of
register 0x02 to 1 with temp2
        }
        else //am
        {
            rtc_write(hourREG, ((rtc_dec2bcd(hour)) | temp1) & 0xDF); //0xDF =
11011111, must set bit5 of register 0x02 to 0
        }
    }
    else //24h format
    {
        rtc_write(hourREG, (rtc_dec2bcd(hour)) & 0xBF); //0xBF = 10111111, must
set bit6 of register 0x02 to 0
    }
}

//void rtc_weekday(unsigned char wt)
void rtc_set_weekday(unsigned char wt)
{

```

```
    rtc_write(weekREG, (rtc_dec2bcd(wt)));
}
```

```
//void rtc_set_date(unsigned char date)
void rtc_set_date(unsigned char date)
{
    rtc_write(dateREG, (rtc_dec2bcd(date)));
}
```

```
//void rtc_month(unsigned char mon)
void rtc_set_month(unsigned char mon)
{
    rtc_write(monthREG, (rtc_dec2bcd(mon)));
}
```

```
//void rtc_set_year(unsigned char year)
void rtc_set_year(unsigned char year)
{
    rtc_write(yearREG, (rtc_dec2bcd(year)));
}
```

```
/*
 * Set the alarm selected via the 'type' value (see defined values in DS3231.h)
 * See example program to see how this is used.
 *
 * Important: the time format (24 hr, AM/PM) used to define the Alarm must
match
 * the time format of the DS3231's clock or the Alarm will not trip.
 * const unsigned char alarm1M1SecREG = 0x07;
```

```

* const unsigned char alarm1M2MinREG = 0x08;
* const unsigned char alarm1M3HrREG = 0x09;
* const unsigned char alarm1M4DateREG = 0x0A;
*
* const unsigned char alarm2M2MinREG = 0x0B;
* const unsigned char alarm2M3HrREG = 0x0C;
* const unsigned char alarm2M4DateREG = 0x0D;
*/

```

```

void rtc_set_alarm1 (uint8_t type)
{
    Wire.beginTransaction(DS3231_ADDRESS);
    boolean alarm1 = (type & 0x80) == 0;
    Wire.write(alarm1 ? 0x07 : 0x0B);
    if (alarm1)
    {
        Wire.write(rtc_dec2bcd(alarmTime.alarmSec) | ((type & 1) << 7));    //
secondss
    }
    Wire.write(rtc_dec2bcd(alarmTime.alarmMin) | ((type & 2) << 6));    //
minutes

```

```

if(config1.h24_12) //12h format
{
    uint8_t temp1 = 1 << 6;
    uint8_t temp2;
    if(alarmTime.alarmAmPm) //pm
    {
        temp2 = 1 << 5;

```

```

        Wire.write((rtc_dec2bcd(alarmTime.alarmHour) | temp1 | temp2) | ((type &
4) << 5)); //set bit5 to 1 with temp2, bit6 to 1 with temp1, 12h and pm
        //    Serial.printf("0x09H = %x\n", (rtc_dec2bcd(alarmTime.alarmHour) | temp1
| temp2) | ((type & 4) << 5));

    }
    else //am
    {
        Wire.write(((rtc_dec2bcd(alarmTime.alarmHour) | temp1) & 0xDF) | ((type
& 4) << 5)); //0xDF = 11011111, set bit6 to 1 with temp1, bit5 to 0 with 0xDF, 12h
and am
        //rtc_write(alarm1M3HrREG, ((rtc_dec2bcd(alarmTime.alarmHour) | temp1)
& 0xDF) | ((type & 4) << 5));
    }
}
else //24h
{
    Wire.write((rtc_dec2bcd(alarmTime.alarmHour) | ((type & 4) << 5)) & 0xBF);
//set bit6 to 0 with 0xBF, 24h
    //rtc_write(alarm1M3HrREG, (rtc_dec2bcd(alarmTime.alarmHour) | ((type &
4) << 5)) & 0xBF); //set bit6 to 0 with 0xBF, 24h
}

if (type & 0x10)
{
    Wire.write(rtc_dec2bcd(alarmTime.alarmDay) | (((type & 8) << 4) | 0x40)); //
day of week
}
else
{

```

```

        Wire.write(rtc_dec2bcd(alarmTime.alarmDate) | ((type & 8) << 4));    //
date
    }
    Wire.endTransmission();

    Serial.printf("alarm1M1SecREG = %x\n", rtc_read(alarm1M1SecREG));
    Serial.printf("alarm1M2MinREG = %x\n", rtc_read(alarm1M2MinREG));
    Serial.printf("alarm1M3HrREG = %x\n", rtc_read(alarm1M3HrREG));
    Serial.printf("alarm1M3DateREG = %x\n", rtc_read(alarm1M4DateREG));
    Serial.printf("hourREG = %x\n", rtc_read(hourREG));
    Serial.printf("statusREG = %x\n", rtc_read(statusREG));
    Serial.println();

}

void rtc_enable_alarm(uint8_t alarm, boolean enable)
{
    uint8_t mask = alarm & 0x03;
    uint8_t ctrlRegVal = rtc_read(controlREG);
    rtc_write(controlREG, ctrlRegVal = (enable ? (ctrlRegVal | mask) :
(ctrlRegVal & ~mask)) | 0x04);
    Serial.printf("controlReg = %x", rtc_read(controlREG));
}

void rtc_clear_alarm()
{
    rtc_write(statusREG, rtc_read(statusREG) & 0xFC);
}

```

```

//*****
*****

void rtc_set_param(tm* tt)
{
    rtc_clear_alarm();

    if (config1.alarmOn)
    {
        rtc_set_alarm1(ALARM1_HOUR_MIN_SEC_MATCH);
        rtc_enable_alarm(1, (bool)config1.alarmOn);
    }

    rtc_set_second((unsigned char) tt->tm_sec);
    rtc_set_minute((unsigned char) tt->tm_min);
    if(config1.h24_12 == 0) //24h format
    {

        rtc_set_hour((unsigned char) tt->tm_hour);

    }
    else //12h format
    {
        if((unsigned char) tt->tm_hour < 12 && (unsigned char) tt->tm_hour != 0)
//1-11 (24h)
        {
            config1.amPm = 0;
            rtc_set_hour((unsigned char) tt->tm_hour);

```

```

    }
    else if((unsigned char) tt->tm_hour < 12 && (unsigned char) tt->tm_hour ==
0) //0 (24h)
    {
        config1.amPm = 0;
        rtc_set_hour((unsigned char) tt->tm_hour + 12);

    }
    else if((unsigned char) tt->tm_hour == 12) //12 (24h)
    {
        config1.amPm = 1;
        rtc_set_hour((unsigned char) tt->tm_hour);

    }
    else if((unsigned char) tt->tm_hour > 12 && (unsigned char) tt->tm_hour <=
23) //13-23 (24h)
    {
        config1.amPm = 1;
        rtc_set_hour((unsigned char) tt->tm_hour - 12);

    }

}

rtc_set_date((unsigned char) tt->tm_mday);
rtc_set_month((unsigned char) tt->tm_mon + 1);
rtc_set_year((unsigned char) tt->tm_year - 100);
if (tt->tm_wday == 0)
{
    rtc_set_weekday(7);
}

```

```

else
    rtc_set_weekday((unsigned char) tt->tm_wday);

}

//*****

*****

float rtc_get_temp()
{
    float t = 0.0;
    unsigned char lowByte = 0;
    signed char highByte = 0;
    lowByte = rtc_read(tempLSBREG);
    highByte = rtc_read(tempMSBREG);
    lowByte >>= 6;
    lowByte &= 0x03;
    t = ((float) lowByte);
    t *= 0.25;
    t += highByte;
    return (t); // return temp value
}

//*****

*****

uint8_t numberOfDaysInMonth( uint16_t year, uint8_t month )
{
    if ( month == 2 )
        return ( (year % 400 == 0) || (year % 4 == 0 && year % 100 != 0) ) ? 29 : 28;

    else if ( month == 4 || month == 6 || month == 9 || month == 11 )

```

```
return 30;
```

```
return 31;
```

```
}
```

```
/**  
*****
```

```
*****
```

```
void rtc_read_param()
```

```
{
```

```
    unsigned short Hour, Date, Month, Year, Minute, Second;
```

```
    signed short WeekDay;
```

```
    Year = rtc_get_year();//年
```

```
    if (Year > 99)
```

```
        Year = 0;
```

```
    Month = rtc_get_month();//月
```

```
    if (Month > 12)
```

```
        Month = 0;
```

```
    Date = rtc_get_date();//天
```

```
    if (Date > 31)
```

```
        Date = 0;
```

```
    WeekDay = rtc_get_weekday(); //weekday
```

```
    if (WeekDay == 7)
```

```
        WeekDay = 0;
```

```
    Hour = rtc_get_hour();//小时
```

```
    if (Hour > 23)
```

```
        Hour = 0;
```

```

Minute = rtc_get_minute();//分钟

if (Minute > 59)
    Minute = 0;

Second = rtc_get_second();//秒

if (Second > 59)
    Second = 0;


if(utc.utcAct)
{
    utc.utcHour = (int8_t)Hour - config1.timeZone;
    //Serial.printf("utc.utcHour = %d\n", utc.utcHour);
    if(utc.utcHour < 0)
    {
        if(config1.h24_12) //12h
        {
            utc.utcHour = utc.utcHour + 12;
            if(config1.amPm == 0) //am
            {
                utc.utcBackOrPlusOneDay = -1;
            }
            utc.utcAmPm = !config1.amPm;
        }
        else //24
        {
            utc.utcHour = utc.utcHour + 24;
            utc.utcBackOrPlusOneDay = -1;
        }
    }
    else if(utc.utcHour == 0)
    {

```

```

        if(config1.h24_12) //12h
        {
            utc.utcHour = 12;
//            utc.utcHour = utc.utcHour + 12;
            utc.utcAmPm = 0;
//            utc.utcAmPm = config1.amPm;
            //Serial.println("you should be here");
        }
    }
    else if((utc.utcHour >= 12) && (utc.utcHour <= 24) && (config1.h24_12))
//12h
    {
        utc.utcHour = utc.utcHour -12;
        if(config1.amPm) //pm
        {
            utc.utcBackOrPlusOneDay = 1;
            utc.utcAmPm = 0;
        }
        else //am
        {
            utc.utcBackOrPlusOneDay = 0;
            utc.utcAmPm = config1.amPm;
        }
    }
    else if(utc.utcHour >= 24 && config1.h24_12 == 0) //24h
    {
        utc.utcBackOrPlusOneDay = 1;
        utc.utcHour = utc.utcHour - 24;
    }
    else if((config1.h24_12)&&(Hour == 12))
    {

```

```

if(config1.amPm == 0) //am
{
    utc.utcBackOrPlusOneDay = -1;
}
utc.utcAmPm = !config1.amPm;
}
else
{
    utc.utcBackOrPlusOneDay = 0;
    utc.utcAmPm = config1.amPm;
}
}

```

```

//So=0, Mo=1, Di=2 ...

```

```

MEZ.sec1 = Second % 10;
MEZ.sec2 = Second / 10;
MEZ.sec12 = Second;

```

```

MEZ.min1 = Minute % 10;
MEZ.min2 = Minute / 10;
MEZ.min12 = Minute;

```

```

if(utc.utcAct)
{
    MEZ.hour1 = utc.utcHour % 10;
    MEZ.hour2 = utc.utcHour / 10;
    MEZ.hour12 = utc.utcHour;

```

```

uint8_t totalDays;

```

```
totalDays = numberOfDaysInMonth(Year, Month);
```

```
Date = utc.utcBackOrPlusOneDay + Date;
```

```
if(Date > totalDays)
```

```
{
```

```
    Date = 1;
```

```
    utc.utcBackOrPlusOneMon = 1;
```

```
    Month = utc.utcBackOrPlusOneMon + Month;
```

```
    if(Month > 12)
```

```
    {
```

```
        Month = 1;
```

```
        utc.utcBackOrPlusOneYear = 1;
```

```
        Year = utc.utcBackOrPlusOneYear + Year;
```

```
    }
```

```
}
```

```
else if(Date <= 0)
```

```
{
```

```
    utc.utcBackOrPlusOneMon = -1;
```

```
    Month = utc.utcBackOrPlusOneMon + Month;
```

```
//    Month = Month - 1;
```

```
    if (Month <= 0)
```

```
    {
```

```
        Month = 12;
```

```
        utc.utcBackOrPlusOneYear = -1;
```

```
        Year = utc.utcBackOrPlusOneYear + Year;
```

```
//    Year = Year - 1;
```

```
}
```

```
totalDays = numberOfDaysInMonth(Year, Month);
```

```
Date = totalDays;
```

```
}
```

```
WeekDay = utc.utcBackOrPlusOneDay + WeekDay;
```

```
if (WeekDay == 7)
```

```
    WeekDay = 0;
```

```
else if(WeekDay < 0)
```

```
    WeekDay = 6;
```

```
}
```

```
else
```

```
{
```

```
    MEZ.hour1 = Hour % 10;
```

```
    MEZ.hour2 = Hour / 10;
```

```
    MEZ.hour12 = Hour;
```

```
}
```

```
MEZ.weekday = WeekDay;
```

```
MEZ.date12 = Date;
```

```
MEZ.date1 = Date % 10;
```

```
MEZ.date2 = Date / 10;
```

```
MEZ.mon12 = Month;
```

```
MEZ.mon1 = Month % 10;
```

```
MEZ.mon2 = Month / 10;
```

```
MEZ.year12 = Year;
```

```
MEZ.year1 = Year % 10;
```

```
MEZ.year2 = Year / 10;
```

```
}
```

```
bool shouldSaveConfig = false;
```

```
void saveConfigCallback ()
{
  Serial.println("Should save config");
  shouldSaveConfig = true;
}
```

```
void wifi_config()
{
  char callsign[10] = "BD5XM";
  char showcall[10] = "yes or no";
  char brightness[10] = "1-16";
  char timezone[10] = "-12 to 12";
  char hour_format[10] = "12 or 24";
  char alarm_on[10] = "yes or no";
  char alarm[20] = "06:30:10:AM";

  char mqtt_server[40] = "test.mosquitto.org";
  char mqtt_port[6] = "1883";
  char mqtt_user[16] = "user name";
  char mqtt_pass[16] = "user password";
```

```
if ( digitalRead(btnA) == LOW )
{
  WiFiManager wm;
  delay(50);
  if( digitalRead(btnA) == LOW )
  {
    Serial.println("Button Pressed");
    tone(buz, 500);
    delay(300);
```

```
tone(buz, 800);  
delay(300);  
tone(buz, 500);  
delay(300);  
noTone(buz);
```

```
wm.setSaveConfigCallback(saveConfigCallback);
```

```
WiFiManagerParameter custom_callsign("0", "CALL&NAME", callsign,  
10);
```

```
WiFiManagerParameter custom_showcall("1", "SHOWCALL", showcall,  
10);
```

```
WiFiManagerParameter custom_brightness("2", "BRIGHTNESS", bright-  
ness, 10);
```

```
WiFiManagerParameter custom_timezone("3", "TIMEZONE", timezone,  
10);
```

```
WiFiManagerParameter custom_hour_format("4", "H24_12", hour_format,  
10);
```

```
WiFiManagerParameter custom_alarm_on("5", "ALARM_ON", alarm_on,  
10);
```

```
WiFiManagerParameter custom_alarm("6", "ALARM", alarm, 20);
```

```
WiFiManagerParameter custom_mqtt_server("7", "MQTT_SERVER",  
mqtt_server, 40);
```

```
WiFiManagerParameter custom_mqtt_port("8", "MQTT_PORT", mqtt_port,  
6);
```

```
WiFiManagerParameter custom_mqtt_user("9", "USER_MAME",  
mqtt_user, 16);
```

```
WiFiManagerParameter custom_mqtt_pass("10", "USER_PASSWORD",  
mqtt_pass, 16);
```

```
WiFiManagerParameter custom_chip_id("11", "CHIP_ID",  
String(ESP.getChipId()).c_str(), 16);
```

```
wm.addParameter(&custom_callsign);  
wm.addParameter(&custom_showcall);  
wm.addParameter(&custom_brightness);  
wm.addParameter(&custom_timezone);  
wm.addParameter(&custom_hour_format);  
wm.addParameter(&custom_alarm_on);  
wm.addParameter(&custom_alarm);  
wm.addParameter(&custom_mqtt_server);  
wm.addParameter(&custom_mqtt_port);  
wm.addParameter(&custom_mqtt_user);  
wm.addParameter(&custom_mqtt_pass);  
wm.addParameter(&custom_chip_id);
```

```
Serial.println("Starting config portal");  
wm.setConfigPortalTimeout(300);
```

```
if (!wm.startConfigPortal("CLOCK"))  
{  
  Serial.println("failed to connect or hit timeout");  
  delay(3000);  
  ESP.restart();  
}  
else  
{  
  Serial.println("connected...yeey :)");  
  strcpy(callsign, custom_callsign.getValue());  
  strcpy(showcall, custom_showcall.getValue());
```

```
strcpy(brightness, custom_brightness.getValue());
strcpy(timezone, custom_timezone.getValue());
strcpy(hour_format, custom_hour_format.getValue());
strcpy(alarm_on, custom_alarm_on.getValue());
strcpy(alarm, custom_alarm.getValue());
strcpy(mqtt_server, custom_mqtt_server.getValue());
strcpy(mqtt_port, custom_mqtt_port.getValue());
strcpy(mqtt_user, custom_mqtt_user.getValue());
strcpy(mqtt_pass, custom_mqtt_pass.getValue());
```

```
//save the custom parameters to FS
```

```
if (shouldSaveConfig)
```

```
{
```

```
    Serial.println("saving config");
```

```
    DynamicJsonBuffer jsonBuffer;
```

```
    JsonObject& json = jsonBuffer.createObject();
```

```
    json["call"] = callsign;
```

```
    json["showCall"] = showcall;
```

```
    json["brightness"] = brightness;
```

```
    json["timeZone"] = timezone;
```

```
    json["hourFormat"] = hour_format;
```

```
    json["alarmOn"] = alarm_on;
```

```
    json["alarm"] = alarm;
```

```
    json["mqttServer"] = mqtt_server;
```

```
    json["mqttPort"] = mqtt_port;
```

```
    json["mqttUser"] = mqtt_user;
```

```
    json["mqttPass"] = mqtt_pass;
```

```
File configFile = SPIFFS.open("/config.json", "w");
```

```
if (!configFile) {
```

```

        Serial.println("failed to open config file for writing");
    }

    json.prettyPrintTo(Serial);
    json.printTo(configFile);
    configFile.close();
    shouldSaveConfig = false;
}
}
}
}
}

bool wifi_auto_config()
{
    WiFi.mode(WIFI_STA);
    Serial.print("Connecting to ");
    Serial.println(WiFi.SSID());

    WiFi.begin();           // connect to the stored wifi credentials

    for (int i = 0; i < 10; i++)
    {
        max7219_char_to_arr('W', maxPosX - 3, 0);
        max7219_char_to_arr('I', maxPosX - 9, 0);
        max7219_char_to_arr('F', maxPosX - 15, 0);
        max7219_char_to_arr('I', maxPosX - 21, 0);
        max7219_refresh_display();
        max7219_char_to_arr('>', maxPosX - 27, 0);
        max7219_refresh_display();
        delay(1000);
    }
}

```

```
max7219_char_to_arr('>', maxPosX - 33, 0);  
max7219_refresh_display();  
delay(1000);
```

```
if (WiFi.status() == WL_CONNECTED)  
{  
    Serial.println("AutoConfig Success");  
    Serial.printf("SSID:%s\r\n", WiFi.SSID().c_str());  
    Serial.printf("PSW:%s\r\n", WiFi.psk().c_str());  
    max7219_clear_display();  
    max7219_char_to_arr('W', maxPosX - 3, 0);  
    max7219_char_to_arr('I', maxPosX - 9, 0);  
    max7219_char_to_arr('F', maxPosX - 15, 0);  
    max7219_char_to_arr('I', maxPosX - 21, 0);  
    max7219_char_to_arr(0x80, maxPosX - 27, 0);  
    max7219_refresh_display();
```

```
    Serial.println("WiFi connected");  
    Serial.println(WiFi.localIP());  
    Serial.println("Starting UDP");  
    udp.begin(localPort);  
    Serial.print("Local port: ");  
    Serial.println(udp.localPort());  
    delay(1000);  
    return true;  
}
```

```
else
```

```
{  
    Serial.print("WiFi Connectiong Failed.....");  
    Serial.println(WiFi.status());  
    delay(1000);
```

```

    }
}

max7219_clear_display();
max7219_char_to_arr('R', 25, 0);
max7219_char_to_arr('T', 19, 0);
max7219_char_to_arr('C', 12, 0);
max7219_char_to_arr('!', 6, 0);
max7219_refresh_display();
delay(1000);
Serial.println("Ops! TimeOut!" );
Serial.println("Clock use RTC!" );
return false;
}

//ss

void wifi_show_IP()
{

    String myIP = WiFi.softAPIP().toString();
    Serial.print("AP IP address: ");
    Serial.println(myIP);
    char buf[myIP.length() + 1];
    myIP.toCharArray(buf, myIP.length());
    max7219_scroll_left(buf, 31, -1, 10);

}

```