

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування інституту, факультету)

Кафедра комп'ютерних систем та мереж

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавра

(ступінь вищої освіти)

на тему Розробка системи охоронно-пожежної сигналізації

Виконав: студент(ка) 4 курсу, групи КНТ-517

Спеціальності 123 «Комп'ютерна інженерія»
(код і найменування спеціальності)

Освітня програма (спеціалізація)
«Комп'ютерна інженерія»

Оверченко Кирило Андрійович
(прізвище та ініціали)

Керівник Грушко С.С.
(прізвище та ініціали)

Рецензент Фурманова Н.І.
(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування вищого навчального закладу)

Інститут, факультет інформатики та радіоелектроніки, комп'ютерних наук і технологій
 Кафедра «Комп'ютерні системи та мережі»
 Ступінь вищої освіти (освітній ступінь) бакалаврський
 Спеціальність 123 Комп'ютерна інженерія
 (код і найменування)

Освітня програма (спеціалізація) Комп'ютерна інженерія
 (назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри Кудерметов Р.К.

“ ” 2021 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТА

Оверченко Кирилу Андрійовичу

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Розробка системи охоронно-пожежної сигналізації

керівник проекту (роботи) Грушко Світлана Сергіївна, к. т. н., доцент

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від “ 17 березня 2021 року № 81

2. Строк подання студентом проекту (роботи) 11 травня 2021 року

3. Вихідні дані до проекту (роботи) об'єкт моніторингу з контрольними точками, необхідність передавати сигнал на зазначений номер телефону за запитом і при перевищенні граничного значення температури

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1) Аналіз предметної області

2) Вибір теплових датчиків для цифрової системи моніторингу температури

3) Розробка програмної складової системи пожежної сигналізації

4) Розробка апаратної частини

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

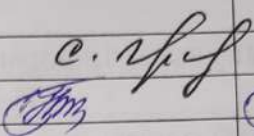
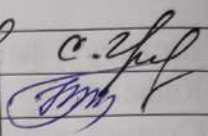
Характеристика плати Raspberry Pi;

Архітектура системи;

Підключення тензодатчиків;

Робота з ioBroker

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-3	Грушко С.С., к. т. н., доцент		
Нормоконтроль	Щербак Н.В., ст.викл. каф. КСМ		

7. Дата видачі завдання 01.03.2021 р.

КАЛЕНДАРНИЙ ПЛАН

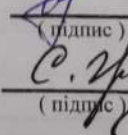
№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Аналіз технічного завдання. Огляд сучасних рішень	01.03.2021 р.	
2	Розробка структури системи	15.03.2021 р.	
3	Розробка алгоритму роботи системи	20.03.2021 р.	
4	Реалізація проекту	01.04.2021 р.	
5	Дослідження роботи системи	10.04.2021 р.	
6	Оформлення отриманих результатів у ПЗ	15.04.2021 р.	
7	Оформлення графічного матеріалу	20.04.2021 р.	
8	Оформлення допоміжного матеріалу	01.05.2021 р.	

Студент


(підпис)

К.А. Оверченко
(ініціали та прізвище)

Керівник проекту (роботи)


(підпис)

С.С. Грушко
(ініціали та прізвище)

РЕФЕРАТ

ПЗ: 65 с., 13 рис., 8 табл., 1 додаток, 12 джерел.

GSM, PDU, USART, ДАТЧИК ТЕМПЕРАТУРИ, МІКРОКОНТРОЛЕР

Об'єкт розробки – система охоронно-пожежної сигналізації.

Мета роботи – розробка системи, яка буде здійснювати контроль теплових датчиків в приміщеннях об'єкту, що охороняється і забезпечувати передачу інформації по провідному каналу на ПК, з подальшою передачею до спеціалізованих служб із застосуванням радіочастотного каналу GSM чи інших систем зв'язку.

В дипломній роботі розглянуто принципи побудови систем охоронно-пожежних сигналізацій. Проаналізовано апаратне і програмне забезпечення проектування і розробки таких систем. Проведено аналіз сучасних датчиків температури, мікроконтролерів і SMS-повідомлень в форматі PDU. В роботі обрано і детально описано принципи роботи мікроконтролера AtMega88 і температурного датчика DS18B20. Також описано їх функціональні можливості і конструкції, програмування мікроконтролера і робота з вищезгаданими датчиками в Bascom.

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

GSM	Global System for Mobile Communications - глобальний стандарт цифрового мобільного стільникового зв'язку
GND	Ground - «земля» VDD - плюс живлення
ROM	Read-Only Memory - постійний запам'ятовуючий пристрій
TH	Trigger High - старший регістр тригера
TL	Trigger Low - молодший регістр тригера
EEPROM	Electrically Erasable Programmable Read-Only Memory - перепрограмований постійний запам'ятовуючий пристрій, що електрично стирається
MOSFET	The Metal-Oxide-Semiconductor Field-Effect Transistor - польовий транзистор з ізолюваним затвором
CRC	Cyclic Redundancy Check - циклічний надлишковий код
SRAM	Static Random Access Memory - статична оперативна пам'ять з довільним доступом
ОЗП	Оперативний запам'ятовуючий пристрій
КМОН	Комплементарна структура метал-оксид-напівпровідник
RISC	Restricted (Reduced) Instruction Set Computer - обчислювач з RISC
MIPS	Million Instructions Per Second - мільйон інструкцій в секунду АЛП - арифметико-логічний пристрій
CISC	Complex Instruction Set Computing - обчислювач з повним набором команд
ШИМ	Широтно-імпульсна модуляція

АЦП	Аналого-цифровий перетворювач
USART	Universal Asynchronous Receiver-Transmitter -універсальний асинхронний прийомопередавач
SPI	Serial Peripheral Interface - послідовний периферійний інтерфейс МК - мікроконтролер
ДП	Друкована плата
УГП	Умовно-графічне позначення
САПР	Система автоматизованого проектування
PDU	Protocol DataUnit - протокол блоку даних
SMS	Short Messaging Service - служба коротких повідомлень
LCD	Liquid Crystal Display - рідкокристалічний дисплей
TDMA	Time Division Multiple Access - множинний доступ з поділом за часом
ПК	Персональний комп'ютер
ПЕОМ	Персональна електронно-обчислювальна машина

ЗМІСТ

Вступ.....	9
1 Аналіз технічного завдання	10
2 Вибір апаратного та програмного забезпечення.....	14
2.1 Вибір теплових датчиків для цифрової системи моніторингу температури	14
2.2.1 Огляд датчиків.....	14
2.2.2 Режим конвертації температури.....	17
2.2.3 Живлення DS18B20	19
2.2.4 64-бітовий код датчика.....	19
2.2.5 Пам'ять	20
2.2.6 Регістр конфігурації.....	21
2.2.7 Конфігурація апаратних засобів.....	22
2.2.8 Послідовність операцій для звернення до DS18B20	23
2.3 Вибір керуючого мікроконтролера	25
2.4 Система проектування.....	26
2.4.1 Огляд систем проектувань	26
2.4.2 Proteus ISIS Professional.....	27
2.4.3 Збірка схеми.....	29
2.4.4 Налаштування схеми.....	30
2.4.5 Формування програми мікроконтролерного керування	33
2.4.6 Змінні в BASCOM.....	34
2.4.7 Рядки в BASCOM.....	35
2.4.8 Масиви в BASCOM.....	37
2.4.9 Функції та підпрограми	37
2.4.10 Локальні змінні в підпрограмах і функціях	39
2.4.11 Переривання	39

2.4.12 Робота з EEPROM	40
2.5 Редактор коду і симулятор	42
3 Розробка програмної складової системи пожежної сигналізації.....	43
3.1 Формування SMS-повідомлення про вимірювану температуру.....	43
3.1.1 Подання повідомлень в форматі PDU.....	43
3.1.2 Отримання повідомлення в режимі PDU	46
3.1.3 Відправлення повідомлень в режимі PDU	48
3.2 Програмна реалізація опитування теплових датчиків	49
4 Розробка апаратної частини.....	52
4.1 Загальний опис	52
4.2 Загальні характеристики стандарту GSM.....	54
Висновки	56
Перелік джерел посилання.....	57
Додаток А.....	58

ВСТУП

На будь-якому об'єкті існує загроза нанесення шкоди майну і здоров'ю людей при виникненні неконтрольованого спалаху або пожежі. Єдиний спосіб звести в цьому випадку можливі втрати до мінімуму - це побудувати ефективну систему виявлення і ліквідації загоряння. Основним способом вирішення цієї проблеми є установка системи пожежної сигналізації, яка призначається для виявлення вогнищ загоряння і управління системами оповіщення людей про пожежу, установками автоматичного пожежогасіння, а також технологічним обладнанням.

Завданням дипломної роботи є розробка системи, яка буде здійснювати контроль теплових датчиків в приміщеннях об'єкту, що охороняється і забезпечувати передачу інформації по провідному каналу на ПК, з подальшою передачею до спеціалізованих служб із застосуванням радіочастотного каналу GSM чи інших систем зв'язку. Причому, як у відповідності із заданими часовими інтервалами, так і по сигналу запиту; окремо формувати і передавати сигнали в випадках, коли реальні значення температури виходять за встановлені допустимі межі.

Передбачається, що на початковому етапі розробки буде змодельована система адресної пожежної сигналізації за допомогою прикладного програмного комплексу ProteusVSM. Даний прикладний програмний комплекс ще на стадії розробки дозволить нам з мінімальними витратами аналізувати працездатність спроектованих пристроїв і сформованих програм, моделюючи роботу готового пристрою.

1 АНАЛІЗ ТЕХНІЧНОГО ЗАВДАННЯ

В даний час можна виділити кілька основних типів пожежної сигналізації:

- традиційна неадресна пожежна сигналізація;
- адресна пожежна сигналізація;
- адресно-аналогова пожежна сигналізація.

Традиційна неадресна пожежна сигналізація на рис. 1.1 представляє собою систему, в якій приймально-контрольний прилад визначає зону виникнення тривожного сповіщення в межах шлейфу. В шлейф пожежної сигналізації такого типу включаються звичайні порогові (неадресні, активні або пасивні) датчики. При спрацьовуванні датчика його номер і місце розташування не вказуються, ініціюється тільки номер шлейфу. Конкретне місце виникнення загоряння може визначити лише черговий персонал шляхом огляду всіх приміщень зони.

Недоліки даної системи - низька інформативність (відсутність інформації про несправності сповіщувача), висока ймовірність помилкових спрацьовувань, дорогий монтаж.

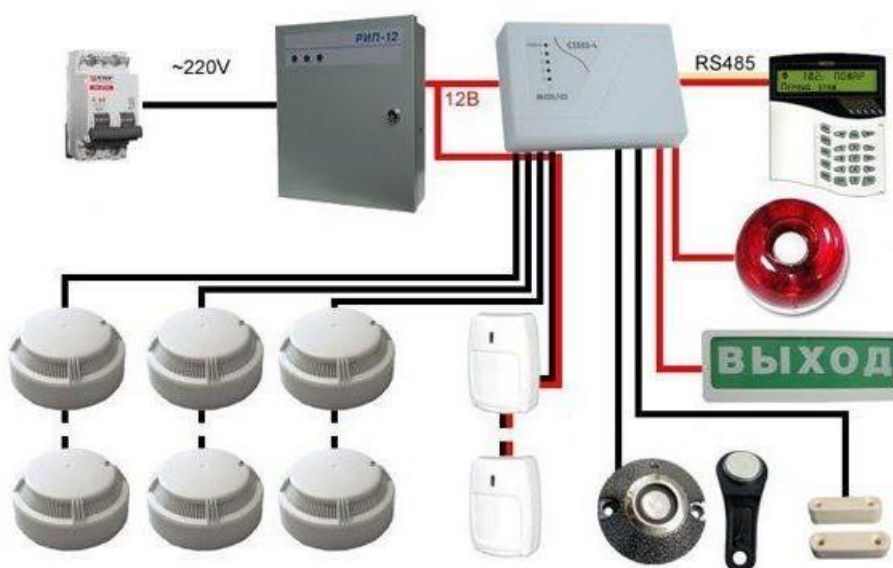


Рисунок 1.1 - Традиційна неадресна пожежна сигналізація

Адресні системи пожежної сигналізації на рис. 1.2 дозволяють визначити не тільки зону, але і точне місце розташування спрацювання датчика. При спрацюванні датчик передає по шлейфу адресу у вигляді послідовного коду, який відображається на дисплеї приймально-контрольного приладу. У кожному датчику розташована схема установки адреси. Таким чином, система визначає конкретне місце формування сигналу про загоряння, що підвищує оперативність реагування спеціальних служб і виключення помилкових спрацювань.



Рисунок 1.2 - Адресна пожежна сигналізація

Адресно-аналогові системи пожежної сигналізації на рис. 1.3, мають найбільш розвинені функціональні можливості, надійність і гнучкість, є центром збору інформації, що надходить від датчиків.

Важливою відмінністю адресно-аналогових систем пожежної сигналізації є те, що в них сповіщувач є лише вимірником параметра і транслює на приймально-контрольний прилад його значення і свою адресу, а приймально-контрольний прилад оцінює величину і швидкість зміни цього параметра, а також управляє індикацією, включаючи відповідний режим. Тобто всі рішення по контролю і

управлінню пожежною ситуацією на об'єкті дозволені приймально-контрольним приладом.

Адресно-аналогова система також дозволяє гнучко організувати роботу і взаємодію всіх інженерних систем життєзабезпечення будівлі.



Рисунок 1.3 - Адресно-аналогова пожежна сигналізація

В ході проведеного аналізу було визначено, що найбільш ефективною є адресна пожежна сигналізація із застосуванням модулів GSM для бездротової передачі в пункт збору інформації.

Для розробки даної системи будуть використовуватися цифрові термодатчики.

Для управління опросом датчиків і виведення інформації в модуль GSM передбачається використовувати мікроконтролер фірми AtmelATMega88.

Як правило, об'єкти, що охороняються, знаходяться на видаленні від центральних пультів спостереження. У зв'язку з цим постає питання передачі інформації від систем моніторингу до керуючих систем. Було вирішено використовувати стільниковий зв'язок GSM. Як передавач буде використовуватися модуль GSMSIM900. Він підтримує стандарт GSM 900/1800 МГц, підтримує

передачу SMS-повідомлень способом блоку протокольних даних - PDU (Protocol Data Unit), а так само можливо його підключення до ПК за допомогою інтерфейсу RS232, що дозволить змодельовати і налагодити роботу в програмному комплексі Proteus.

Проектування і налагодження обраної структури системи передбачається реалізовувати в програмному середовищі Proteus розроблений компанією Labcenter Electronics. Пакет базується на основі моделей електронних компонентів прийнятих в PSpice.

PROTEUS VSM дозволяє моделювати роботу програмованих пристроїв: мікроконтролерів, мікропроцесорів, цифрових сигнальних процесорів і інших цифрових пристроїв.

Розробка програми для обраного мікроконтролера буде здійснюється в Bascom-AVR. Це середовище розробки програм для мікроконтролерів сімейства AVR на мові високого рівня близького до стандартного Бейсік. До складу Bascom входить компілятор програм, текстовий редактор, відладчик-симулятор, спрощений емулятор терміналу, контекстна інформаційно-довідкова система і утилення завантаження апаратних емуляторів і програматорів.

Все сказане вище дозволяє зробити висновок, що загальна структура проекрованої системи представляється у форматі схеми взаємодії температурних датчиків, керуючого мікроконтролера і послідовного інтерфейсу модуля GSM (рис. 1.4).

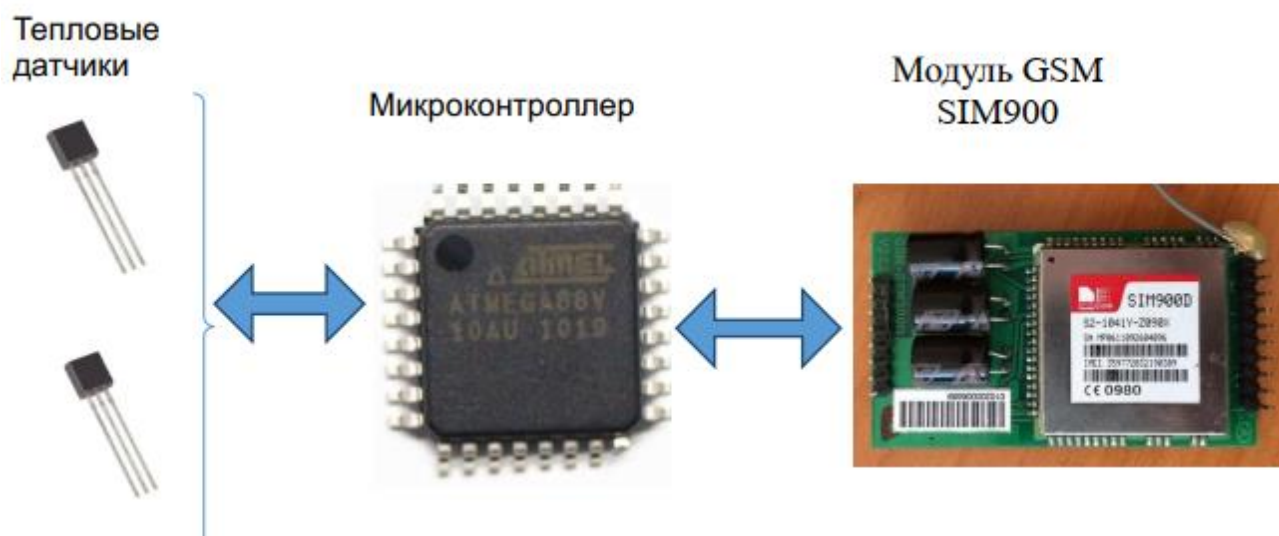


Рисунок 1.4 - Загальна структура проектованої системи

2 ВИБІР АПАРАТНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Вибір теплових датчиків для цифрової системи моніторингу температури

2.2.1 Огляд датчиків

Сучасний ринок засобів реалізації пожежних сигналізацій пропонує широку номенклатуру теплових датчиків вітчизняного та імпортного виробництва з різними фізичними принципами перетворення температури в електричні сигнали (як аналогові, так і цифрові). Причому для тих, що розробляються на перспективу систем очевидна доцільність застосування саме цифрових, оскільки в цьому випадку можна обійтися без аналого-цифрових перетворювачів необхідних для подальшої обробки цифрових еквівалентів значень температури. Крім того при цьому можна обійтися і без необхідності формування підсистеми контролю правильності передачі CRC. І найпопулярнішим і недорогими на сьогоднішній день

є універсальний теплової датчик DS18B20 з відповідним діапазоном вимірювань і показниками точності для реалізації пропонованої системи.

Він є цифровим термометром з програмованим дозволом від 9 до 12 біт і функцією тривожного сигналу контролю за температурою. Обмін інформацією відбувається за однопровідною шиною. При цьому на лінії може бути як один пристрій, так і кілька. Управління процесами на шині проводиться центральним мікроконтролером.

Діапазон вимірювань від -55°C до $+125^{\circ}\text{C}$ з точністю 0.5°C в діапазоні від -10°C до $+85^{\circ}\text{C}$.

Кожен датчик має унікальний 64-бітний послідовний код, який дозволяє, спілкуватися з безліччю таких же датчиків, встановлених на одній шині, що дозволяє використовувати один мікроконтролер для контролю великої кількості датчиків, розподілених по великій ділянці.

На рис. 2.1 зображена блок-схема DS18B20, а в таблиці 2.1 перераховані номери контактів і їх опис для різних типів корпусів, в яких випускаються ці датчики.

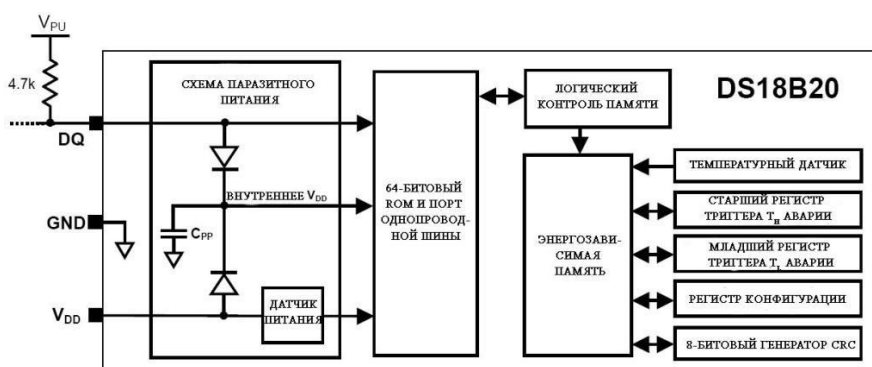


Рисунок 2.1 - Блок-схема DS18B20

Таблиця 2.1 - Таблиця висновків

SO *	SOP *	TO-92	СИМВОЛ	ОПИС
5	4	1	GND	Загальний
4	1	2	DQ	Контакт даних введення / виведення (Input / Output pin). Використовується для підключення до однопровідної шини. З цієї лінії підключений до джерела живлення в режимі роботи з паразитним живленням
3	8	3	VDD	VDD Контакт живлення. Для режиму роботи з паразитним живленням VDD необхідно з'єднати з загальним проводом

* Всі інші контакти повинні залишатися не підключеними.

64-бітовий ROM запам'ятовує унікальний послідовний код приладу. Оперативна пам'ять містить 2-байтовий температурний регістр, в якому зберігається значення температури по закінченню температурного перетворення, два однобайтових регістра температури контролю температури (триггерной схеми ТН і ТЛ), і регістр конфігурації. Регістр конфігурації дозволяє користувачеві встановлювати роздільну здатність цифрового перетворювача температури рівну 9, 10, 11, або 12 бітам, що впливає на час конвертації температури. Регістри ТН, ТЛ і регістри конфігурації енергонезалежні (EEPROM), таким чином, вони збережуть дані, коли прилад вимкнений.

DS18B20 використовує виключно протокол зв'язку з однопровідною шиною, при цьому формується з'єднання, яке здійснює комунікацію на шині, використовуючи всього один керуючий сигнал. Шина повинна бути підключена до джерела живлення через підтягуючий резистор, так як всі пристрої, пов'язані з шиною, використовують з'єднання через Z-стан або вхід відкритого стоку. Використовуючи цю шину мікропроцесор (пристрій управління) ідентифікує і звертається до датчиків температури, використовуючи 64-бітовий код приладу.

Оскільки кожен прилад має унікальний код, число приладів, до яких можна звернутися на одній шині, теоретично необмежено.

Інша особливість DS18B20 це здатність працювати без зовнішнього живлення. Ця можливість надається через підтягуючий резистор. Високий сигнал шини заряджає внутрішній конденсатор (C_{PP}), який живить прилад, коли на шині низький рівень. Цей метод носить назву «Паразитне живлення». При цьому максимальна вимірювана температура становить + 100 ° C.

Для розширення діапазону температур до +125 ° C необхідно використовувати зовнішнє живлення. [10]

Система з однопровідною шиною використовує один головний пристрій управління, щоб керувати одним або більше підлеглими пристроями. DS18B20 завжди використовується як підлеглий пристрій.

Всі дані і команди передаються, починаючи з молодшого біта.

2.2.2 Режим конвертації температури

Основні функціональні можливості DS18B20 - його температурний перетворювач. Роздільна здатність температурного перетворювача може бути змінена користувачем, і становить 9, 10, 11, або 12 бітів, відповідаючи приросту (дискретності вимірювання температури) 0.5 ° C, 0.25 ° C, 0.125 ° C, і 0.0625 ° C відповідно. Роздільна здатність за замовчуванням встановлена 12-біт. У початковому стані DS18B20 знаходиться в стані спокою (в неактивному стані). Щоб почати температурне вимірювання і перетворення, ведучий повинен подати команду початку конвертації температури [0x44]. Після конвертації, отримані дані запам'ятовуються в 2-байтовому регістрі температури в оперативній пам'яті, і DS18B20 повертається до неактивного стану. Якщо DS18B20 включений із зовнішнім живленням, ведучий може контролювати конвертування температури (після команди [0x44]) станом шини. DS18B20 формуватиме логічний «0» коли відбувається температурне перетворення, і логічну «1», коли конвертування виконано. Вихідні температурні дані DS18B20 калібровані в градусах Цельсія. Температурні дані запам'ятовуються як 16-бітове число зі знаком (рис. 2.2). Біти

ознаки (S) вказують, чи є значення температури позитивним або негативним: для позитивних чисел $S = 0$, а для негативних $S = 1$. Якщо DS18B20 буде налаштований для конвертації 12-бітного дозволу, то все біти в температурному регістрі будуть містити дійсні дані. Для 11-бітної роздільної здатності, біт 0 невизначений. Для 10-бітної роздільної здатності, біти 1 і 0 не визначені, і для 9 бітної роздільної здатності біти 2, 1 і 0 не визначені.

	7	6	5	4	3	2	1	0
LS БАЙТ	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
	15	14	13	12	11	10	9	8
MS БАЙТ	S	S	S	S	S	2^6	2^5	2^4

Рисунок 2.2 - Формат регістра температури

У таблиці 2.2 наведені приклади співвідношення даних з цифрового виходу і значення температури для 12-бітної роздільної здатності.

Таблиця 2.2 - Співвідношення температури і даних

Температура	Цифровий вихід (двійковий)	Цифровий вихід (Hex)
+125 ° C	0000 0111 1101 0000	07D0h
+ 85 ° C *	0000 0101 0101 0000	0550h
+ 25.0625 ° C	0000 0001 1001 0001	0191h
+ 10.125 ° C	0000 0000 1010 0010	00A2h
+ 0.5 ° C	0000 0000 0000 1000	0008h
0 ° C	0000 0000 0000 0000	0000h
-0.5 ° C	1111 1111 1111 1000	FFF8h
-10.125 ° C	1111 1111 0101 1110	FF5Eh
-25.0625 ° C	1111 1110 0110 1111	FE6Fh
-55 ° C	1111 1100 1001 0000	FC90h

* При подачі живлення в температурний регістр записано число + 85 ° C

2.2.3 Живлення DS18B20

DS18B20 може також бути підключений до живлення звичайним методом, приєднанням зовнішнього електроживлення до контакту VDD, як показано на рис. 2.3. Перевага цього методу полягає в тому, що немає необхідності у використанні MOSFET-транзистора. А на шині можуть передаватися дані протягом часу температурного перетворення.

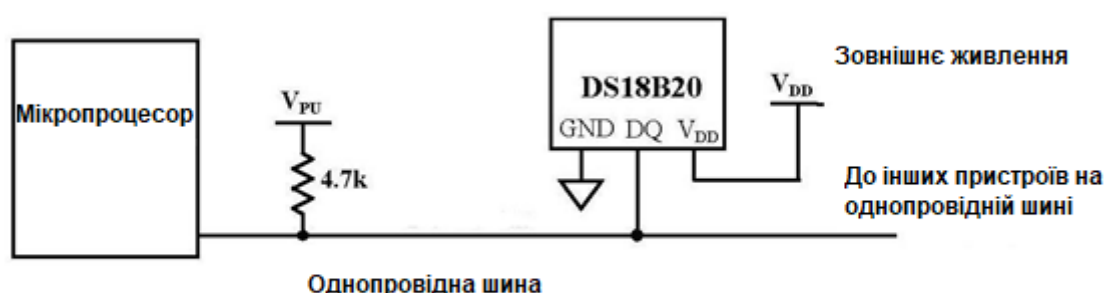


Рисунок 2.3 - Підключення DS18B20 в режимі зовнішнього живлення

2.2.4 64-бітовий код датчика

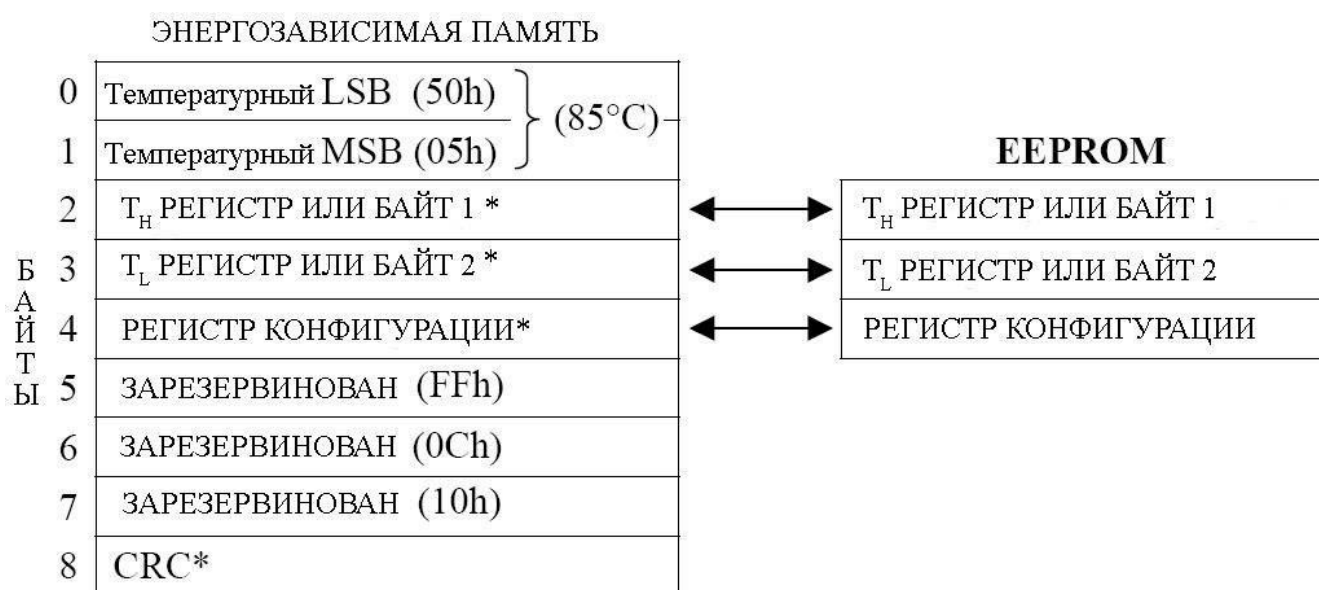
Кожен DS18B20 містить унікальний 64-бітовий код (рис. 6.5), збережений в ROM. Молодші 8 бітів коду ROM містять код приналежності до однопровідного сімейства (28h). Наступні 48 бітів містять унікальний серійний номер. Старші 8 бітів містять байт циклічного контролю надмірності (CRC), який обчислюється з перших 56 бітів коду ROM. 64-бітовий код ROM і пов'язана функція ROM управляють логікою і дозволяють DS18B20 працювати як однопровідний пристрій.

8-БИТОВЫЙ CRC		48-БИТОВЫЙ СЕРИЙНЫЙ НОМЕР		8-БИТОВЫЙ КОД СЕМЕЙСТВА(28h)	
MSB	LSB	MSB	LSB	MSB	LSB

Рисунок 2.4 - 64-бітовий код ROM

2.2.5 Пам'ять

На рис. 2.4 зображена схема організації пам'яті DS18B20. Пам'ять складається з оперативної SRAM пам'яті і незалежній пам'яті EEPROM. Перші два регістри - це регістри конвертора температури, далі йдуть регістри порушення верхньої і нижньої межі (THiTL) і регістр конфігурації. Зверніть увагу, що, якщо функція виходу за межі не використовується, то регістри THiTL можуть служити осередками універсальної пам'яті.



* Стан після включення живлення залежить від значень, збереженого в EEPROM

Рисунок 2.4 -DS18B20 карта пам'яті

Байт 0 і байт 1 з ОЗУ містить молодший і старший біт температурного регістра відповідно. Ці байти тільки для читання. Байти 2 і 3 забезпечують доступ до регістрів THiTL. Байт 4 містить дані регістра конфігурації. Байти 5, 6, і 7 зарезервовані для внутрішнього використання пристроєм і запис в ці регістри неможливий; при читанні ці байти повернуть «1» у всіх розрядах.

8 байт ОЗУ тільки для читання і містять код циклічного контролю надмірності, обчислений для байтів 0 - 7 ОЗУ.

Для запису даних в байти 2, 3, і 4 ОЗУ використовуємо команду запису [4Eh]; дані повинні бути передані до DS18B20, починаючи з байта 2. Щоб перевірити коректність запису даних, необхідно виконати читання (використовуючи команду читання [BEh]) після того, як дані будуть записані. Зверніть увагу, що при читанні дані DS18B20 починає передавати з 0 байта. Щоб зберегти TH, TL і дані регістра конфігурації в EEPROM, пристрій управління має послати команду копіювання даних з ОЗУ [48h].

Дані, збережені в регістрах EEPROM, при включенні живлення завантажуються в ОЗУ.

Дані можуть бути перезавантажувались з EEPROM в ОЗУ в будь-який час командою [B8h]. Пристрій управління може контролювати цей процес виклику даних з EEPROM в ОЗУ шляхом видачі синхроімпульсу після команди і контролю стану шини - якщо 0 операція перезавантаження триває, якщо 1, то процес вважається виконаним.

2.2.6 Регістр конфігурації

Байт 4 пам'яті містить регістр конфігурації, організація якого проілюстрована на рис. 2.5.

7	6	5	4	3	2	1	0
0	R1	R0	1	1	1	1	1

Рисунок 2.5 - Регістри конфігурації

Користувач може налаштувати конверсійну роздільну здатність DS18B20, використовуючи біти R0 і R1 в цьому регістрі відповідно до таблиці 3. Значення цих бітів за замовчуванням при включенні живлення R0 = 1 і R1 = 1 (включена 12-бітова роздільна здатність). Зверніть увагу, що є пряма залежність між роздільною здатністю і конверсійним часом. Біт 7 і біти від 0 до 4 в регістрі конфігурації зарезервовані для внутрішнього використання пристроєм і не можуть бути змінені або використані користувачем, при читанні ці біти повертають «1».

Таблиця 2.3 - Налаштування конфігурації роздільної здатності

R1	R0	Розширення	Максимальний час конвертації	
0	0	9-bit	93.75 ms	(TCONV / 8)
0	1	10-bit	187.5 ms	(TCONV / 4)
1	0	11-bit	375 ms	(TCONV / 2)
1	1	12-bit	750 ms	(TCONV)

2.2.7 Конфігурація апаратних засобів

Однопровідна шина за визначенням має тільки єдину лінію даних. Кожен пристрій (управляє або підлеглий пристрій), використовує доступ до шини даних через пристрої з відкритим колектором або порт з трьома станами. Це дозволяє кожному пристрою "випускати" лінію даних, коли пристрій не передає дані, таким чином шина доступна для використання іншими пристроями. Порт однопровідної шини у DS18B20 еквівалентний схемою на рис. 2.6.

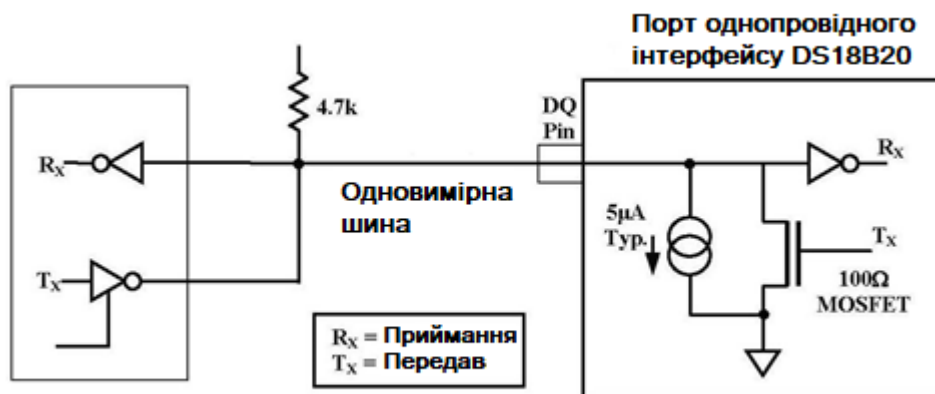


Рисунок 2.6- Конфігурація шини

Однопровідна шина вимагає зовнішнього підтягуючого резистора приблизно 5 кОм; таким чином, неактивний стан для неї - логічна одиниця. Якщо з якої-небудь причини обмін даними повинен бути припинений, шину потрібно залишити в неактивному стані. Відновлення живлення відбувається між передачею бітів, коли шина знаходиться в високому стані. Якщо шина буде переведена в

низький стан на час більш ніж 480 мкс, то всі компоненти на шині будуть скинуті (ініціалізовані).

2.2.8 Послідовність операцій для звернення до DS18B20

Послідовність кроків така:

- ініціалізація;
- команда ROM (супроводжувана будь-яким потрібним обміном даних);
- функціональна команда DS18B20 (супроводжувана будь-яким потрібним обміном даних).

Дуже важливо дотримуватися цієї послідовності кожен раз, коли звертаються до DS18B20, оскільки датчик не буде відповідати в зворотному випадку.

Винятки з цього правила становлять команди пошуку ROM [F0h] і пошуку аварії [ECh].

Після подачі будь-якої з цих команд, контролер управління (керована програма) повинен повернутися до кроку 1 в послідовності операцій звернення.

Всі операції на шині починаються з послідовності ініціалізації. Послідовність ініціалізації складається з імпульсу скидання, переданого пристроєм управління шиною і імпульсом або імпульсами присутності, переданими підлеглими пристроями. Імпульс присутності дозволяє пристрою управління шиною дізнатися, що підлеглі пристрої (типу DS18B20) присутні на шині і готові до роботи.

Після того, як пристрій керування шиною виявив імпульс присутності, він може формувати команди ROM. Ці команди оперують унікальними 64-бітовими кодами ROM для кожного підлеглого пристрою, і дозволяють пристрою управління вибрати певний пристрій, з безлічі пристроїв, присутніх на шині.

Ці команди також дозволяють пристрою управління визначити, як багато пристроїв яких типів присутні на шині, а також визначити пристрій, що знаходиться в стані «аварія».

Є п'ять команд ROM, і кожна команда 8 бітів довжиною. Головний пристрій повинен передати відповідну команду ROM перед передачею команди функції DS18B20.

Пошук ROM [F0h]. Коли система спочатку включена, головний пристрій повинен ідентифікувати коди ROM всіх підлеглих пристроїв на шині. Ця команда дозволяє пристрою управління визначати номери і типи підлеглих пристроїв. Пристрій управління вивчає коди ROM через процес усунення, що вимагає, щоб головний пристрій виконало цикл пошуку ROM. Цю процедуру необхідно виконати стільки раз, скільки необхідно, щоб ідентифікувати всі підлеглі пристрої. Якщо на шині є тільки один підлеглий пристрій, то замість пошуку ROM може використовуватися простіша команда читання ROM. Після кожного циклу пошуку, пристрій керування шиною має повернутися до ініціалізації.

Читання ROM [33h]. Ця команда може використовуватися, тільки коли на шині є одне підлеглий пристрій. Ця команда дозволяє пристрою управління шиною читати ROM підлеглого пристрою, не використовуючи процедуру пошуку ROM. При використанні цієї команди там, де на шині більш ніж один підлеглий пристрій, то станеться конфлікт на рівні даних, так як всі підлеглі пристрої зроблять спробу відповісти в один і той же час.

Відповідність ROM [55h]. Команда відповідності ROM, супроводжувана послідовністю 64-бітового коду ROM дозволяє пристрою управління шиною звертатися до певного підлеглого пристрою на шині. Тільки підлеглий пристрій, який точно відповідає 64 бітам послідовності коду ROM, відповість на функціональну команду, випущену головним пристроєм. Всі інші підлеглі пристрої на шині чекатиме імпульсу скидання.

Пропуск ROM [CCh]. Головне пристрій може використовувати цю команду, щоб звернутися до всіх пристроїв на шині одночасно. Наприклад, головний пристрій може змусити, щоб все датчики на шині, почали одночасно температурні перетворення. Для цього необхідно видати на шину команду пропуску ROM, супроводжувану командою температурного перетворення [44h]. Зверніть увагу, що

команда читання пам'яті [BEh] може слідувати за командою пропуску ROM, тільки якщо на шині присутній одне підлеглий пристрій. Команда пропуску ROM, супроводжувана командою читання пам'яті викличе конфлікт на рівні даних на шині, якщо на шині буде більш одного підлеглого пристрою, так як всі пристрої будуть намагатися одночасно передавати дані.

Пошук стану аварії [ECh]. Операція цієї команди ідентична операції команди пошуку ROM за винятком того, що тільки датчики DS18B20 з встановленим прапором аварії дадуть. Ця команда дозволяє головному пристрою визначати, які датчики випробували сигнальний стан протягом недавнього температурного перетворення. Після кожного циклу пошуку аварії, пристрій керування шиною має повернутися до ініціалізації

2.3 Вибір керуючого мікроконтролера

Мікроконтролер - мікросхема, призначена для управління електронними пристроями.

ATMega88 - низькоспоживаючий 8 бітний КМОП мікроконтролер з AVR RISC архітектурою. Виконуючи команди за один цикл, ATMega88 досягає продуктивності 1 MIPS (Million Instructions Per Second) при частоті генератора, що задає 1 МГц, що дозволяє розробнику оптимізувати ставлення споживання до продуктивності.

AVR ядро об'єднує багату систему команд і 32 робочих регістра загального призначення. Всі 32 регістра безпосередньо пов'язані з арифметико логічним пристроєм (АЛП), що дозволяє отримати доступ до двох незалежних регістрів при виконанні однієї команди. В результаті ця архітектура дозволяє забезпечити в десятки разів більшу продуктивність, ніж стандартна CISC архітектура.

ATMega88 підтримується різними програмними засобами та інтегрованими засобами розробки, такими як компілятори C, макроасемблера, програмні відладники / симулятори, внутрисхемні емулятори та ознайомчі набори.

Технічні дані ATMega88:

- RISC архітектура;
- 130 команд, більшість може виконуватися за один такт;
- 32 8-бітних робочих регістра;
- статична архітектура;
- продуктивність до 16 MIPS при частоті 16 МГц;
- інтегрований, двох циклової множник;
- незалежна пам'ять програм і даних.
- 8 КБ внутрісистемна програмована Flash пам'ять, розрахована на 10 000 циклів запису і стирання.

2.4 Система проектування

2.4.1 Огляд систем проектувань

Для розробки керуючої програми системи температурного моніторингу можна використовувати ряд пакетів програм проектування електронних схем (Electronic WorkBentch, MicroSIM, Protius) але для даної розробки доцільніше застосування останнього програмного комплексу, так як в його бібліотеку входить програмна модель мікроконтролера AtMega88 прийнята для реалізації проектованої системи.

Пакет являє собою систему схемотехнічного моделювання, що базується на основі моделей електронних компонентів. Відмінною рисою пакета є можливість моделювання роботи програмованих пристроїв: мікроконтролерів,

мікропроцесорів і інших. Бібліотека компонентів містить довідкові дані. Додатково в пакет Proteus VSM входить система проектування друкованих плат. Пакет Proteus складається з двох підпрограм: ISIS - програма синтезу та моделювання безпосередньо електронних схем і ARES - програма розробки друкованих плат. Разом з програмою встановлюється набір демонстраційних проектів для ознайомлення.

Також до складу восьмої версії входить середовище розробки VSM Studio, що дозволяє досить швидко написати програму для використовуваного в проекті мікроконтролера і скомпілювати її.

До програмного пакету додається велика кількість прикладів розташованих в папці «Приклади» (Samples).

Папка «Схеми і розкладка шарів» (Schematic & PCBLayout) - містить приклади виконання схем і друкованих плат, тобто майже кожен приклад проекту в ISIS має однойменний приклад проекту в ARES.

Найбільш вражає - 3D візуалізація в ARES. Для цього, відкривши будь-який проект з цієї папки в ARES, досить вибрати пункт 3D Visualization в закладці «Вихід» (Output) верхнього меню. Далі, утримуючи натиснуту ліву кнопку мишки, ви можна обертати плату, переміщати, видаляти або наближати за допомогою прокрутки колеса миші.

Папка «Інтерактивна симуляція» (InteractiveSimulation) містить приклади, підтримують симуляцію в реальному часі. «Графічна симуляція» (GraphBasedSimulation) містить приклади дослідження схем за допомогою графіків або графів.

«Керівництво» (Tutorials) - приклади створення власних модулів і компонентів. Ряд папок починаються з VSM і далі - характерний тип мікроконтролерів, приклади для яких поміщені в однойменній теці.

2.4.2 Proteus ISIS Professional

Використовується для збірки і налагодження схеми. Головне вікно програми складається з 4-х частин:

- основне робоче простір;
- вікно навігації;
- список використаних пристроїв (або властивостей інструменту);
- панелі інструментів і функцій.

Найбільша область екрану - основний робочий простір (рис. 2.7). У ній можна розміщувати компоненти і зв'язки між ними. Менша область справа вгорі екрану - вікно навігації. Зазвичай вікно навігації використовують для огляду повного малюнка - синій блок показує край поточної карти, а зелений блок - область карти, відображену в даний момент у вікні редагування.

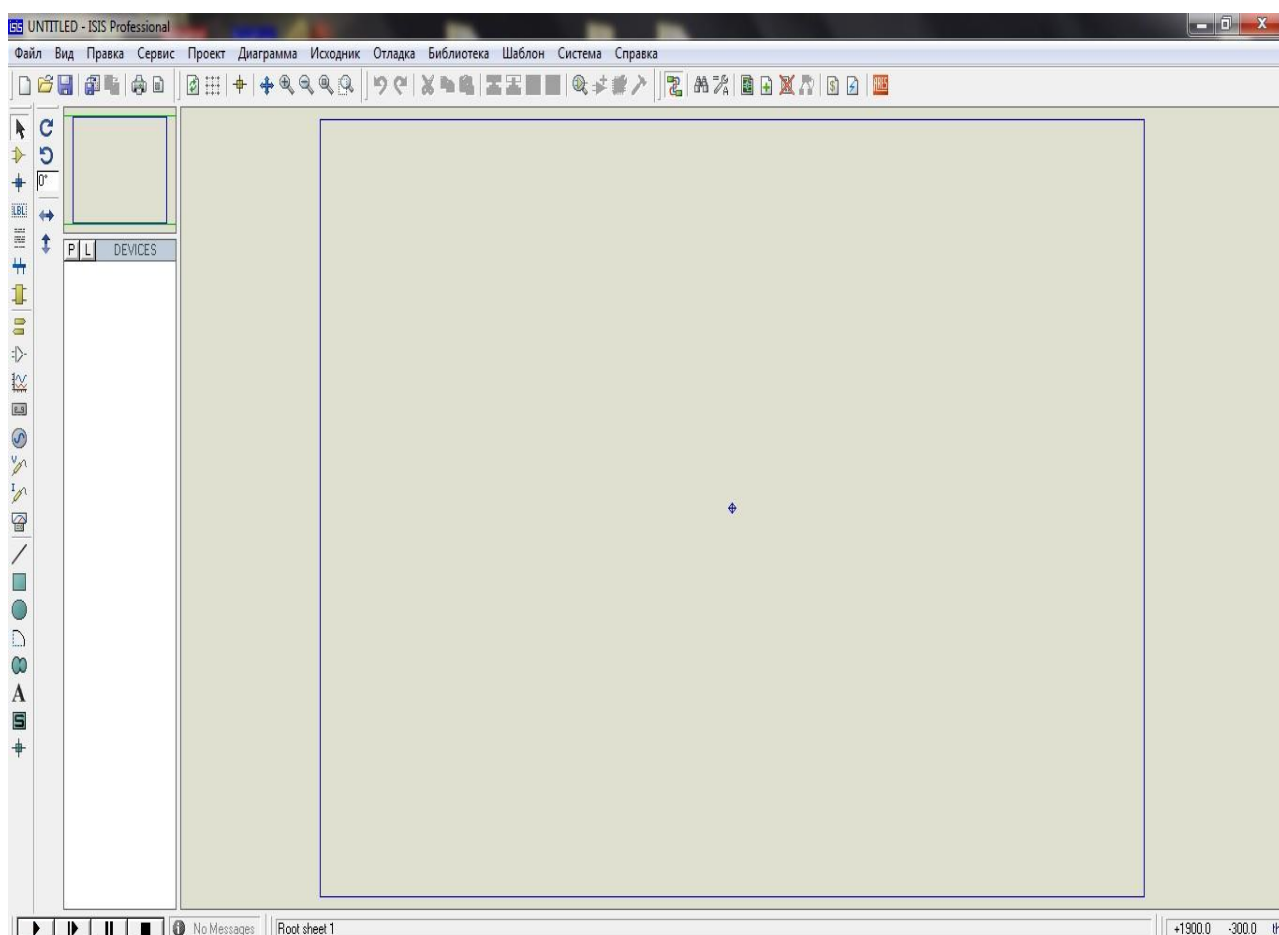


Рисунок 2.7 - Стартове вікно Proteus ISIS Professional

Ви можете коригувати область рисунка, відображеного в основному робочому просторі наступними способами:

- за допомогою прокрутки, встановивши покажчик миші в центр необхідної області і затиснувши клавішу F5;
- зрушення здійснюється утримуванням клавіші SHIFT і клацанням миші на межі робочого простору;
- якщо Ви хочете перемістити робочий простір в абсолютно іншу області малюнка, то найшвидшим способом є вказівка центру нової області у вікні навігації клацанням в потрібній точці лівою кнопкою миші.

2.4.3 Збірка схеми

Натиснувши кнопку Рв верхньому лівому кутку списку використовуваних пристроїв, виклинемо вікно бібліотеки пристроїв. Далі знайдемо необхідні компоненти в бібліотеці.

Це можна зробити декількома способами:

- якщо ви знаєте назву необхідного пристрою, ви можете почати вводити його в рядку пошуку. Під час введення назви програма автоматично буде підбирати підходящі пристрої з наявних бібліотек, і виводити їх список на екран;
- також можна здійснювати пошук за ключовими словами в описі пристрою;
- інший спосіб пошуку компонентів - якщо ви не знаєте точної назви шуканого пристрою, можна самостійно переглядати розділи бібліотеки і вибирати відповідний компонент, виходячи з наведених описах компонентів;
- подвійним клацанням по вибраного компоненту ми можемо додати його в список компонентів проекту.

Щоб додати компонент в робочий простір, досить вибрати його в списку компонентів проекту, а після клацнути лівою кнопкою миші в потрібній області робочого простору.

Після розміщення в робочому просторі всіх необхідних компонентів, потрібно з'єднати їх між собою, розмістивши між ними шини. ISIS має вбудовану інтелектуальну систему для розміщення зв'язків, яка називається «Провідний автоматичний маршрутизатор». Все, що необхідно робити для його роботи - це вказувати початкову і кінцеву точку з'єднань покажчиком миші.

Якщо запропонований автоматичним маршрутизатором варіант розміщення зв'язків вам не подобається, ви можете відредагувати їх вручну. Для цього одиночним клацанням правої кнопки миші виділіть його, а потім покажчиком зсувайте різні відрізки редагованої зв'язку.

Так само розміщення зв'язку можна задавати вручну, відзначаючи лівою кнопкою миші точки, через які повинен пройти провід на схемою.

2.4.4 Налагодження схеми

Після складання електричної схеми процес проектування передбачає тестування зібраного модуля на працездатність, відповідність вхідних і вихідних параметрів технічним завданням, і налагодження в цілому схеми, або окремих її модулів.

В процесі симуляції додаток може видавати різні повідомлення, покликані полегшити працю проектувальника в процесі налагодження схеми.

Обов'язково звертайте увагу на повідомлення в логах симуляції (Simulation Log) поряд з кнопкою «СТОП» симуляції. Вікно логів симуляції є і після зупинки при натисканні на ньому лівою кнопкою миші. Тут діє принцип світлофора:

- зелені - повідомлення про успішну компіляції;
- жовті – попередження;
- червоні - помилка симуляції.

Розглянемо повідомлення, що найбільш часто вилітають в процесі симуляції в додатку з описом можливих дій по їх усуненню.

Помилка симуляції: «No model specified for (позначення елемента) Simulation FAILED due partition analysis error (s)». При виборі моделі елемента з бібліотеки потрібно звернути увагу на верхнє праве вікно «Огляд схеми»

(Schematic Preview), де над схемним зображенням елемента з'являється його характеристика. Якщо там є повідомлення «Немає моделі для симуляції» (No Simulator Model), не слід намагатися додавати його в схему. Ці елементи (до речі, до них відносяться майже всі роз'єми) призначені для створення принципової схеми з передачею в іншу програму, що входить в пакет «Proteus» - «ARES», для подальшої розробки друкованої плати пристрою. Всі інші моделі призначені для симуляції.

Помилка симуляції: «(External) ModelDLL (позначення елемента) not found Simulation FAILED due partition a nalysiserror (s)». Найбільш часто зустрічається при спробі запустити симуляцію проекту створеного в пізнішій версії програми «Proteus». Це можливо з двох причин; або у вашій версії відсутня бібліотека для цього елемента, або автор застосував свій власний елемент і не доклав до нього бібліотеку.

Помилка симуляції: «PROGRAM property not specified Real Time Simulation failed to start». При симуляції схеми, що містить мікроконтролер, в його властивості не вказано (або неправильно вказано) файл прошивки. В якості такого файлу можуть використовуватися бінарні з розширенням .hex, а також файли, створювані деякими компіляторами з мов високого рівня з розширенням .cof, .elf. Останній необхідно набрати безпосередньо з клавіатури, тому що у вікні вибору у властивостях мікроконтролера він явно пропонуватися не буде. Попередження симуляції: «Simulation is not running in real time due to excessive CPU load». При цьому зазвичай внизу фігурує «CPU load 100%». Симуляція не може виконуватися в режимі реального часу через стовідсоткове завантаження процесора комп'ютера. Зазвичай проявляється на схемах перевантажених аналоговими компонентами.

Попередження симуляції: «[SPICE] TRAN: Timestep to small; timestep = (значення): trouble with node # значення # branch ». Ключовим у цьому повідомленні є фраза: «Time step to small». Зазвичай цього попередження передують ще кілька попереджень «Spice про DELMIN» і «GMIN». Причина виникнення помилки в тому, що крок за часом досяг мінімальної величини. Якщо програма не може знайти

рішення, то відбувається пошук заново все більш дрібними кроками по часу, поки набудуть досягнута мінімальна дозволена користувачем величина «TMIN». Подібні проблеми збіжності вирішуються наступним чином. Зайдя в меню «System-> Setanimationoption -> Spiceoption -> Transient - TMIN = 1E-25, NUMSTEPS = 500», на вкладці кількості ітерацій для пошуку рішення «Iteration» починаючи з третього параметра збільшувати допустиму кількість ітерацій в 10 разів, наприклад: «SRCSTEPS = 1200, GMINSTEPS = 1200» і так далі. Якщо рішення не буде сходитися, то можна дозволити програмі шукати його ще більшу кількість кроків. Далі на вкладці «Tolerance» можна послаблювати вимоги до точності обчислення, наприклад: «ABSTOL = 1e10, VNTOL = 1e -5 CHGTOL = 1E-10 GMIN = 1E-10 RSHUNT = 1e10 TRANSGMIN = 1E-8 »і так далі.

Попередження симуляції: «Logic contention (s) detected on net # (номер ланцюга)». У цифровому ланцюзі з номером # (номер ланцюга) виявлений конфлікт сигналів. Якщо клацнути мишкою по номеру відкриється список ланцюга, в якій виявлений конфлікт. Зазвичай, коли два або більше виходів з різними станами 0 і 1 об'єднані в один ланцюг в поточний момент часу. Часто з'являється, коли використовується інтерактивна кнопка, яка замикає контакт на землю або шину живлення без додаткового резистора, тобто при симуляції її замкнули, а в цей момент на контакті елемента з'явився протилежний сигнал.

Іноді однотипні жовті попередження симулятора йдуть суцільним потоком і забивають вікно «Simulation Log», заважаючи процесу налагодження, хоча симуляція при цьому йде успішно.

Якщо проектувальник впевнений у своїй правоті, то можна відключити це повідомлення, так само, як і інші повідомлення діагностики, увійшовши у вкладку: Debug => ConfigureDiagnostics ... для обраного елемента, встановивши прапор Disabled для даного типу повідомлень.

2.4.5 Формування програми мікроконтролерного керування

Для формування програми мікроконтролерного керування процесом вимірювання температури доцільно використовувати програмний комплекс BASCOM.

BASCOM розроблений компанією MSC Electronics і поставляється в трьох варіантах:

- BASCOM-LT для МК серії AT89Cх051 фірми Atmel;
- BASCOM-8051 для мікроконтролерів серії 8051;
- BASCOM-AVR для мікроконтролерів серії AVR фірми Atmel.

BASCOM - це додаток для ПК, що дозволяє:

- написання програм на мові BASIC;
- компіляцію їх;
- симуляцію роботи скомпільованої коду;
- програмування мікроконтролерів з допомогою зовнішніх утиліт або безпосередньо з середовища розробки.

BASCOM - засіб для швидкої розробки, в ньому вбудована підтримка багатьох вбудованих функцій AVR, таких як: лічильники, таймери, універсальний асинхронний приймач, аналого-цифровий перетворювач, широтно-імпульсну модуляцію, послідовну шину даних, яка використовує дві двонаправлені лінії зв'язку. І безліч зовнішніх пристроїв: кнопки, буквено-цифрові і графічні індикатори, 3х4 або 4х4 клавіатури, PS / 2 клавіатура, інфрачервоне дистанційне керування.

Саме це і робить BASCOM привабливим засобом для розробки нових пристроїв, так як він дуже економить час вбудованими функціями.

Розробники BASCOM стверджують, що він не є "діалектом" BASIC. Однак його код неймовірно схожий на Microsoft QBasic. Основна відмінність BASCOM від нього в тому, що в BASCOM допустима лише одна дія на рядку. BASCOM не дозволяє виробляти більше однієї математичної операції на одному рядку. Ця особливість пояснюється тим, що мікроконтролер в будь-якому випадку не виконає

за один такт дві дії. А автоматично складну рядок на складові BASCOM не розбивати [11].

2.4.6 Змінні в BASCOM

Ім'я змінної може бути до 32 символів довжиною. У назві можна використовувати цифри і латинські символи. Переконайтеся, що обраний ім'я не збігається зі службовими словами. Наприклад, не можна створювати змінну з назвою "Config". Імена змінних не чутливі до регістру.

У BASCOM є такі типи змінних:

Bit. Зберігає значення 0 або 1;

Byte. Зберігає в пам'яті один байт (8 біт). Використовуйте цей тип для зберігання числа від 0 до 255 або для символу;

Integer. Ця змінна займає в пам'яті два байта і зберігає числа від -32768 до +32767. Так як один біт зарезервований під знак числа, що залишилися 15 діляться навпіл на позитивні і негативні значення. Зберігайте в цьому типі всі числа, яким потрібен знак, а також всі числа, які вписуються в зазначений діапазон. При переповненні змінної даного типу, її значення зміниться з +32767 на -32768;

Word. Це двухбайтний тип, який не має знака. Для зберігання числа доступні всі 16 біт, тому діапазон допустимих чисел від 0 до 65535. Цю зміну можна використовувати для порівняно великих величин, які не мають знака. При переповненні значення 65535 змінюється до 0;

Long. Змінна такого типу займає в пам'яті 4 байта. З 32 біт один відводиться під знак, а всі інші зберігають модуль числа. Таким чином, зберігається діапазон становить від -2147483648 до +2147483647. В змінної типу Long можна зберігати всі значення, які не входять до Integer. При переповненні, як і всі змінні, скинеться до мінімального значення діапазону;

Single. Чотирьохбайтний тип для зберігання чисел з рухомою комою. Запам'ятовує числа від 1.5×10^{45} до 3.4×10^{38} зі знаком. Для зберігання дрібних чисел використовуйте тільки його. Змінна типу Single може зберігати великий

діапазон дрібних позитивних і негативних чисел. Але ви можете втратити деяку точність при перетвореннях;

String. Змінна цього типу пам'ятає рядок з декількох символів. Кожен символ займає в пам'яті один байт. Послідовність символів закінчується так званим "нульовим байтом", тому він також займає місце в пам'яті. Головне у виборі типу змінної - визначитися, дані якого типу будуть в ній зберігатися. Вибирайте тип найменшого відповідного розміру, щоб економити пам'ять. Також пам'ятайте, що при використанні типу **Integer** програма буде набагато компактніше і швидше, ніж при використанні типу **Single**. Операції з плаваючими точками займають багато часу. В ліст. 2.1 можна подивитися приклад оголошення числа типу **Single** в **BASCOM AVR** і результат спроби присвоєння йому значення, що перевищує допустимі межі для типу.

Лістинг 2.1 - Приклад оголошення числа типу **Single** в **BASCOM AVR**

```
DimAsSingle 'Об'являємо змінну A типу Single
A = 0.987654321 'Покладемо в неї деяке число
'Після цих операцій в змінної A виявиться значення
0.9876543.
```

2.4.7 Рядки в **BASCOM**

Рядок - це змінна, що зберігає в собі послідовність символів. Вона використовується для введення / виведення тексту або довгих чисел. У ліст. 2.2 приведений приклад оголошення рядка в **BASCOM**.

Лістинг 2.2 - Приклад оголошення рядка в **BASCOM**

```
DimMessageAsString * 10
```

Змінна **Message** може зберігати до 10 символів. Кожен символ займає один байт. **BASCOM** додає додатковий байт, що позначає кінець рядка. Максимальна довжина рядка таким чином становить 254 символу.

У BASCOM є велике число функцій для роботи з рядками:

- Bin - переводить число в його двійкове подання;
- Format - форматує вхідні рядок згідно деякого формату.
- Fusing - форматує змінну типу Single згідно формату рядка, при необхідності виконує округлення.
- Hex - переводить число в його шістнадцяткове подання.
- Hexval – переводить рядок з шістнадцятковим числом в десяткове число.
- Instr - повертає положення підрядка в рядку.
- Lcase - звертає все символи рядка в нижній регістр.
- Left - повертає кілька символів з початку рядка.
- Len - повертає довжину рядка.
- Ltrim - видаляє пробіли на початку рядка.
- Mid - повертає частину з середини рядка. Mid також можна використовувати для заміщення частини рядка іншим рядком.
- Right - повертає кілька символів з кінця рядка.
- Rtrim - видаляє пробіли в кінці рядка.
- Space - створює рядок, що складається з прогалін.
- Str - перетворює число в строкове значення.
- String - повертає рядок з декількох символів зазначеного ASCII коду.
- Trim - видаляє пробіли на початку і в кінці рядка.
- Ucase - звертає все символи рядка в верхній регістр.
- Val- конвертує рядок в число.

У ліст. 2.3 приведений приклад реалізації функції Format і результат її роботи.

Лістинг 2.3 - Приклад реалізації функції Format в BASCOM

```

Dim      VxodAs
String   *   6   Dim
VyxodAs String *   8
Vxod = "38.869"

      Vyxod = Format (Vxod,
+000.00") Vyxod = "+038.86"
```

2.4.8 Масиви в BASCOM

Масив - це група змінних одного типу, об'єднаних загальною назвою і розрізняються індексом - порядковим номером елемента. В ліст. 2.4 приведений приклад оголошення масиву в BASCOM і доступу до всіх його елементів по черзі.

Лістинг 2.4 - Приклад оголошення масиву в BASCOM

```

Dim Mas (10) as Integer Dim Index as Integer

For Index = 1 to 10
Print "Mas:"; Mas (Index) NextIndex
```

де Mas – назва масиву, число в дужках - кількість елементів в ньому.

Тип змінної Index може бути будь-яким, крім Bit, Single і Double.

Варто так само відзначити, що в BASCOM нумерація елементів масиву завжди починається з 1, а не з 0, як це робиться в багатьох інших компіляторах.

2.4.9 Функції та підпрограми

Підпрограми і функції використовуються, коли є кілька однакових фрагментів коду в різних частинах програми.

Підпрограма має назву і може мати додаткові параметри, які можуть повертатися підпрограмою після досягнення її кінця - EndSub. У ліст. 2.5 приведений приклад оголошення і опису підпрограми.

Лістинг 2.5 - Приклад оголошення і опису підпрограми в BASCOM

```

Dimaasbyte Dimbasbyte

Declare Sub First (Byval a as byte, b as byte)
'об'явлення підпрограми

Sub First (Byval a as byte, b as byte) 'тут
знаходиться тіло підпрограми EndSub

```

Можна вказати підпрограмі, що дані їй параметри потрібно залишити незміненими. Для цього при оголошенні використовується команда `Byval`, тоді BASCOM створить в пам'яті копію цієї змінної і буде працювати з нею. Вихідна змінна залишиться недоторканою, що б не відбувалося з нею в підпрограмі.

Зворотний параметр - `Byref` - встановлюється за умовчанням. Його можна не вказувати. У тому випадку всі зміни, які підпрограма зробить зі змінною, збережуться.

Функції подібні підпрограм, але вони повертають хоча б одне значення програмі, що їх визвала. У ліст. 2.6 приведений приклад оголошення і опису функції.

Лістинг 2.6 - Приклад оголошення і опису функції в BASCOM

```

Dim a as byte

Declare Function Second (byref a as long) as long 'об'явлення
підпрограми

Function Second (byref a as long) as long 'тут знаходиться тіло
підпрограми EndFunction

```

2.4.10 Локальні змінні в підпрограмах і функціях

У підпрограмі або функції можна оголосити змінні, що існують тільки всередині неї. Вони недоступні з основної програми. Для оголошення такої змінної використовуйте оператор `Local`, в ліст. 2.7 приведений приклад.

Лістинг 2.7 - Приклад оголошення змінної типу `Local` в BASCOM

```
Localtemp1 asInteger Localtemp2asLong
```

Всі типи змінних, за винятком `Bit`, можуть бути оголошені як локальні. Змінні типу `Bit` завжди глобальні.

2.4.11 Переривання

Переривання змінюють нормальний хід виконання програми і змушують МК реагувати на зовнішні події так само, як і на внутрішні.

Всі переривання в AVR мають однаковий пріоритет. Це відрізняє їх від інших мікроконтролерів, де ви можете задати пріоритет для кожного виду переривання по відношенню до іншого. Подробиці про переривання можна дізнатися з технічної документації для відповідної моделі контролера.

За замовчуванням переривання відключені. Їх потрібно включити за допомогою команди, розміщеної в ліст. 2.8.

Лістинг 2.8 - Команда включення переривань в BASCOM

```
EnableInterrupts
```

Переривання включаються всією групою. Аналогічно, вони можуть бути і відключені. Приклад приведений в лістингу 2.9.

Лістинг 2.9 - Команда відключення переривань в BASCOM

```
DisableInterrupts
```

Кожне переривання має мати свою підпрограму обробки подій. Підпрограма - це фрагмент коду з міткою, набором команд і функцією повернення Return. Для кожного включеного переривання ви повинні вказати підпрограму, на яку потрібно перейти при виникненні події. Пам'ятайте, що всі підпрограми починаються з позначки-назви, що закінчується ":" (двокрапкою).

У той момент, коли програма переривається і переходить до підпрограми обробки переривання, всі включені переривання відключаються на час роботи цієї підпрограми. Коли її виконання досягає оператора Return і переходить до того місця в головному коді, на якому його виконання було перервано, відключені переривання знову включаються. Це робиться для того, щоб підпрограми обробки не переривали самі себе. Така поведінка, що відрізняється від мікроконтролерів більшості інших фірм, закладена в самій AVR архітектурі, не в BASCOM.

Прагніть зробити підпрограми переривань якомога коротше. Намагайтеся не робити в переривання нічого складнішого, ніж зміна лічильників або прапорів. Фрагменти обробки даних найкраще розмістити в головному коді.

2.4.12 Робота з EEPROM

Читати і записувати дані в EEPROM можна за допомогою відповідних команд Readeeprom і Writeeprom. Але перед їх використанням потрібно особливим чином розмістити дані в незалежній пам'яті. Це можна зробити двома способами. Перший - використовувати \$ eeprom, приклад приведений в ліст. 2.10.

Лістинг 2.10 - Приклад використання \$ eeprom в BASCOM

```

$ eeprom в BASCOM Dim ADCcal as Byte
...
$

Eeprom
ADCcal
1:

Data & H34
$ data
...
```

```
Readeeprom, ADCcal, ADCcall
```

Рядок \$ eeprom необхідно розмістити в програмі перед тим місцем, в якому відбувається звернення до констант в пам'яті. Ця команда змушує перемикатися BASCOM з роботи з блоками даних Data на блоки Eeprom. Команда \$ data виконує зворотну дію. Найкраще розмістити рядок \$ Eeprom перед рядком Data.

Другий метод роботи з EEPROM - через оператор Dim. Приклад приведений в ліст. 2.11

Лістинг 2.11 - Приклад використання оператора Dim в BASCOM

```
...
Dim ADCcal as Byte
Dim ADCcall AsEram Byte
...
ADCcal = ADCcall 'привласнюємо значення ADCcalli
з EEPROM змінній ADCcal ADCcall = ADCcal' поміщаємо
значення змінної ADCcal в область EEPROM, що
зарезервована під зберігання значення ADCcall
```

Тут змінна ADCcal оголошена як змінна типу byte в EEPROM.

Якщо дані в EEPROM розміщувалися за допомогою команди \$ eeprom, то BASCOM створить окремий .eep файл. Тому компіляція поточного проекту дасть два вихідних файлу. Програмування мікроконтролера цьому випадку складатиметься з двох етапів: спочатку програмування його логічної частини за допомогою бінарного файлу, після - програмування його енергозалежної пам'яті за допомогою файлу, створеного командою.

Для роботи з EEPROM годяться обидва методи, але перевагу можна віддати першому, тому що при використанні значень в EEPROM за замовчуванням набагато зручніше окремо запрограмувати зону цієї пам'яті.

2.5 Редактор коду і симулятор

Вікно редактора BASCOM використовується для введення і редагування тексту програми. Він має зручну підсвітку синтаксису:

- зеленим позначаються коментарі: це може бути будь-який текст після апострофа (');
- червоним позначаються спецсимволи: * + -, . / \ ; < > = & ^ % # @ ~ () [] | ? ! @ { };
- напівжирним синім виділяються ключові слова BASCOM. До речі, за цими словами доступна довідка - потрібно встановити курсор на потрібне слово і натиснути F1;
- чорним позначаються змінні.

Ці кольори встановлені за замовчуванням, але їх можна змінити в налаштуваннях.

BASCOM здійснює форматування коду програми у міру його введення, роблячи його більш охайним і легким для читання. Наприклад, він робить кожне слово, яке Ви з великої літери. Ще він прибирає зайві прогалини і додає їх там, де треба, роблячи текст більш розрідженим. Все це можна так само змінити в тих же налаштуваннях.

BASCOM зберігає зміни, внесені в код, перед кожним компіляція.

Симулятор дозволяє вам покроково перевірити програму без програмування контролера і взагалі без використання реальних пристроїв. Можна дивитися стан портів введення / виводу, значення змінних, апаратних регістрів і програмної пам'яті. Також можна подивитися, як контролер взаємодіє з іншими пристроями. Можна симулювати підключений до мікроконтролеру індикатор, змінювати стан портів контролера віртуальними кнопками, підводити потрібне значення напруги

до АЦП і т.п. Запускати симуляцію можна як в невинному режимі, так і в режимі покроковому.

З РОЗРОБКА ПРОГРАМНОЇ СКЛАДОВОЇ СИСТЕМИ ПОЖЕЖНОЇ СИГНАЛІЗАЦІЇ

3.1 Формування SMS-повідомлення про вимірювану температуру

3.1.1 Подання повідомлень в форматі PDU

У разі надсилання текстового SMS-повідомлення, згідно із затвердженим організацією ETSI документу (документи GSM 03.40 і GSM 03.38), може застосовуватися як 7-бітна кодова схема, що включає в себе латинський алфавіт і обмежений набір спецсимволів, так і 16-бітна, що включає в себе більшість відомих алфавітів, в тому числі і кирилицю. У разі надсилання текстового повідомлення в 7-бітному кодуванні загальне число символів не може перевищувати 160, якщо ж повідомлення передається в 16-бітному кодуванні (кирилиця), максимально допустиму кількість використовуваних символів значно менше - 69. Сучасні SMS-С підтримують передачу повідомлень як в 7 -, так і в 16- бітної кодуванні UCS-2, що гарантує прийом, зберігання і маршрутизацію повідомлень російською мовою.

Є два способи посилки і отримання SMS повідомлення: текстовий спосіб і спосіб блоку протокольних даних (PDU - ProtocolDataUnit). Текстовий спосіб недоступний деяким телефонами.

Розглянемо структуру SMS-повідомлення та дії для його відправки. У режимі відправки текстових повідомлень можна передавати тільки букви англійського алфавіту. Для передачі російських символів необхідно використовувати формат PDU, тому нас буде цікавити саме він. Розрізняють вхідні (дивись таблицю 3.1) і вихідні повідомлення (дивись таблицю 3.2). Призначення полів описано в таблиці 3.3. Всім невживаних бітам присвоюється значення 0.

Таблиця 3.1 - Формат вхідного PDU-повідомлення

байти							
1-12	1	2-12	1	1	7	1	0-140
SCA	PDU-type	OA	PID	DCS	SCTS	UDL	UD
Біти поля PDU-type							
7	6	5	4	3	2	1	0
RP	UDHI	SRI	не використовуються		MMS	MTI	

Таблиця 3.2 - Формат вихідного PDU-повідомлення

байти								
1-12	1	1	1-12	1	1	0, 1 або 7	1	0 - 140
SCA	PDUtype	MR	DA	PID	DCS	VP	UDL	UD
Біти поля PDU-type								
7	6	5	4	3	2	1	0	
RP	UDH	SRR		VPF		RD	MTI	

Таблиця 3.3 - Призначення полів PDU-повідомлення

Поле	Призначення
SCA	номер SMSC (SMS-центру)
PDU-Type	поле даних протоколу
MR	кількість успішно переданих з телефону повідомлень (0..255)
OA	телефон-джерело
DA	телефон-приймач
PID	ідентифікатор протоколу - вказує SMSC, як обробляти повідомлення
DCS	схема кодування даних в поле даних
SCTS	час отримання повідомлення SMS-сервером
VP	час дії повідомлення (якщо повідомлення не буде отримано абонентом в Протягом цього часу, SMSC НЕ буде його передавати)
UDL	довжина поля даних
UD	поле даних
RP	вказує на наявність поля відповіді

Продовження таблиці 3.3

Поле	Призначення
UDHI	вказує на наявність заголовка в поле UD
SRI	вимога приймача отримати статус повідомлення
SRR	вимога джерела отримати статус повідомлення
VPF	прапор наявності поля VP
MMS	кількість не відправлених повідомлень в SMSC
RD	видалити дублікати
MTI	тип повідомлення: 00 - входить, 01 - виходить

Всі ці поля, за винятком бітових, - шістнадцяткові, але в телефон вони потрапляють у вигляді символів. Наприклад, якщо поле має значення 4Eh, то передаються два символу: 34h ("4") і 45h ("E"). Розглянемо призначення деяких полів більш докладно.

SCA (ServiceCenterAddress). Телефонний номер SMSC (SMS-центру). Це поле складається з двох частин: власне телефонного номера (від 0 до 6 байт) і типу цього номера (1 байт): 81h (національний) або 91h (міжнародний). Поле номера кодується так: кожна пара цифр міняється місцями. Якщо номер складається з непарної кількості цифр, то в кінці додається 0Fh. Наприклад, для SMSC-оператора UMC 38050000501 поле SCA буде виглядати так: 07918350000005F1. Якщо довжина поля дорівнює 0, то телефон візьме номер SMSC зі своїх налаштувань. Але щоб уникнути помилок краще вказувати номер SMSC явно.

PDU Type (ProtocolDataUnitType). Поля даних протоколу. Всі вони встановлюються в 0, за винятком SRI і MMS, яким присвоюється номер SMSC, і MTI, яке може приймати два значення: 00 - вхідне повідомлення, 01 - виходить.

OAI DA (OriginatorAddress, DestinationAddress). Джерело і приймач повідомлення. Формуються аналогічно полю SCA.

PID (ProtocolIdentifier). Ідентифікатор протоколу: вказує SMSC, як обробляти повідомлення. Повідомляє транспортному рівню, який протокол вищого рівня повинен обробляти це повідомлення. Наприклад, 00h - обробляти

повідомлення як звичайне, 41h - заміщати повідомлення типу 1, і т. Д. До 47h - заміщати повідомлення типу 7.

DCS (DataCodingScheme) Схеми кодування даних в поле даних. Ми будемо використовувати тільки два варіанти значень поля: 80h - режим кодування UCS2 (70 знаків) і 00h - режим 7-розрядної кодування (160 знаків, тільки англійські символи).

UDL (UserDataLength). Довжина поля даних в байтах. Якщо повідомлення складається з одного символу UCS2, то його довжина становить 2 байта, а якщо з одного символу в 7-розрядної кодуванні, то 1 байт.

UD (UserData). Поле даних користувача. У цьому полі, власне, і міститься текст переданого повідомлення в заданій кодуванні. Кодування UCS2 повторює Unicode. Для символів латиниці просто додається 00h. Для символів кирилиці (від C0h до FFh - в кодуванні Windows) можна використовувати таке правило: з байта віднімається C0h і додається 410h (крім букв "е" і "Е", коди яких - 0451h і 0401h).

У 7-розрядної кодуванні коди букв латиниці не відрізняються від 8-розрядної. Упаковка 160 символів в 140 байт проводиться таким чином:

перший байт: старший біт береться з молодшого біта 2-го байта, 7 інших біт - біти першого символу;

другий байт: два старших біти беруться з молодших розрядів 3-го байта, а шість молодших - з решти бітів другого символу і т. д.

3.1.2 Отримання повідомлення в режимі PDU

Пакет PDU містить не тільки повідомлення, але також і службову інформацію (номер SMSC, часова мітка і т.д.). Все це представлено в формі шістнадцятирічних октетів або десяткових октетів. Розглянемо конкретний приклад отриманого SMS-повідомлення і розберемо його структуру.

Код повідомлення: 07917283010010F5040BC87238880900F10000993092516195800AE8329B.

Повідомлення "hellohello" представлено 8 - розрядними байтами перетворені з 7 - розрядних.

Опис структури повідомлення наведено нижче в таблиці 3.4.

Таблиця 3.4 - Опис структури вхідного SMS- повідомлення

Октети	Опис
07	Індикатор показує довжину номера SMSC (в цьому випадку 7 байт).
91	Код інтернаціональності SMSC.
7283010010F5	Номер центру обробки повідомлень - SMSC (представлений десятковими октетами). Довжина номера непарна (11), тому в кінець додано 0F, щоб сформувати належні октети. Номер телефону SMSC - "+27381000015"
04	Перші октети вихідного SMS - повідомлення.
0B	Довжина номера відправника (0BHEX = 11DEC)
C8	Тип адреси номера відправника
7238880900F1	Телефонний номер відправника (десяткові октети). Номер закодований аналогічно номеру SMSC.
00	TP - PID. ідентифікатор протоколу
00	TP - DCS. Схема кодування даних (00 - режим 7-розрядної кодування).
99309251619580	Часова мітка (октети).
0A	TP - UDL. Довжина поля даних користувача (довжина повідомлення). Поле TP - DCS вказало на режим 7-розрядної кодування, тобто повідомлення представлено 10 7-розрядними байтами. Якщо TP - DCS було б встановлено в режим 8 або 16-розрядної кодування, то число байт в цьому випадку дорівнювало б 9
E8329BFD4697D9EC37	TP - UD. Поле даних користувача. Повідомлення "hellohello" представлено 8-розрядними байтами, перетвореними з 7-розрядних.

Всі октети, описані вище в таблиці – шістнадцяткові 8-розрядні октети, крім номера SMSC, номера відправника та часової мітки; вони - десяткові октети. Частина повідомлення в кінці пакета PDU складається з шістнадцятирічних 8-розрядних октетів, але ці октети представляють 7-розрядні дані.

Десяткові полуоктети, наприклад номер відправника, отримані в результаті перестановки двох сусідніх символів всередині полуоктета від "72 38 88 09 00 F1" до "27 83 88 90 00 1F". Зміст часової мітки "99 03 29 15 16 59 08" може бути інтерпретовано наступним чином: 6 перших знаків представляють дату, наступні 6 представляють час, і останні два представляють часовий пояс, пов'язаний за Гринвічем.

3.1.3 Відправлення повідомлень в режимі PDU

Наступний приклад показує, як послати повідомлення "hellohello" в PDU - режимі. Для того щоб відправити повідомлення використовуємо команду AT + CMGS.

Робимо запит пересилання повідомлення з 23 октет за винятком двох перших нулів: AT + CMGS = 23. Повідомлення має наступний вигляд: 0011000B916407281553F80000AA0AE8329BFD4697D9EC37.

Повідомлення складається з 23 октет (46 символів). Перший октет "00" в даному випадку не береться до уваги. Цей октет - індикатор довжини номера SMSC, що не має інформації. Опис октетів наведено нижче в таблиці 3.5.

Таблиця 3.5 - Опис SMS-повідомлення, що відправляється

Октети	Опис
00	Індикатор, що показує довжину номера SMSC. Якщо він дорівнює "00", то використовується номер SMSC закладений в настройках телефону. Цей октет необов'язковий і в деяких телефонах може бути опущений.
11	Перший октет прийнятого SMS - повідомлення

Продовження таблиці 3.5

Октети	Опис
00	TP - Message - Reference. Параметр, який використовується для ідентифікації повідомлення
0B	Довжина телефонного номера одержувача (11)
91	Тип адреси. "91" вказує на міжнародний формат телефонного номера.
6407281553F8	Телефонний номер одержувача в десяткових октетах (46708251358).
00	TP - PID. Ідентифікатор протоколу.
00	TP - DCS. Схема кодування даних (00 - режим 7-розрядної кодування). Наявність "02" замість "00" вказувало б на режим 8-розрядної кодування, що використовується для передачі даних.
AA	TP- Validity - Period. Час дії повідомлення.
0A	TP - User - Data - Length. Довжина поля даних користувача (довжина повідомлення). Поле TP - DCS вказало на режим 7-розрядної кодування, тобто повідомлення представлено 10ю 7-ми розрядними байтами. Якщо TP - DCS було б встановлено в режим 8 або 16-розрядної кодування, то число байт в цьому випадку дорівнювало б 9.
E8329BFD4697D9 EC37	Повідомлення "hellohello" представлене октетами. Перехід від 7-розрядних байтів до октетам було показано вище.

3.2 Програмна реалізація опитування теплових датчиків

Система однопровідної шини використовує один головний пристрій управління, щоб керувати одним або більше підлеглими пристроями. Датчик DS18B20 завжди використовується як підлеглий пристрій.

У даній роботі було вирішено створити систему з 4 датчиками, які попарно підключені до двох різних однопровідних шин, щоб продемонструвати основні можливості цієї технології.

Логіка роботи системи така:

- керуючий пристрій отримує 64-бітовий код кожного датчика;
- ПО черзі проводиться обмін інформацією між керуючим пристроєм і кожним з датчиків на першій шині. Послідовність обмінів залежить від того, який датчик надіслав свій код раніше. Якщо якийсь датчик не відповідає, то для нього виставляється ситуація;
- «Аварії»;
- у міру отримання інформації, контролер обробляє її на допустимість значень.

Алгоритм повторюється для наступного шини.

На рис. 3.1 представлений код програми для опитування температурних датчиків

```

'//подпрограмма измерения температуры и обработки на допустимые значения//
Sub Temperature

    Tem_err = 0

    'Узнаём CRC каждого датчика
    Dsid1(1) = 1wsearchfirst(portd , 5)
    Dsid2(1) = 1wsearchnext(portd , 5)
    Dsid3(1) = 1wsearchfirst(portd , 4)
    Dsid4(1) = 1wsearchnext(portd , 4)

    1wreset                                     ' сброс шины
    1wwrite &HCC : 1wwrite &H44                 ' начать измерения
    waitms 800                                  ' ожидаем окончания преобразования

    'считываем 9 байт из датчика 1 и вычисляем Т с точностью 0.1 С

    1wreset Portd , 5                           ' сброс 1ware шины
    1wwrite &HCC , 1 , Portd , 5                 ' читаем internal RAM
    1wverify Dsid1(1)                            ' адрес датчика
    1wwrite &HBE , 1 , Portd , 5                 ' читаем 9 байтов данных
    Bd1(1) = 1wread(9 , Portd , 5 )              ' читаем байты в массиве
    1wreset Portd , 5                           ' сброс шины

    1wreset Portd , 5                           ' сброс шины
    1wwrite &HCC , 1 , Portd , 5                 ' читаем internal RAM
    1wverify Dsid2(1)                            ' адрес датчика
    1wwrite &HBE , 1 , Portd , 5                 ' читаем 9 байтов данных
    Bd2(1) = 1wread(9 , Portd , 5 )              ' читаем байты в массиве
    1wreset Portd , 5                           ' сброс шины

    If Bd1(9) = Crc8(bd1(1) , 8) Then             'сравнение полученных CRC
        T1 = Makeint(bd1(1) , Bd1(2))
        T1 = T1 * 10 : T1 = T1 / 16
    Else
        Tem_err = 1
        Message = Message + "CRC 1 error"
    End If

    If Err = 1 Then                               ' если сенсор не обнаружен
        Tem_err = 2
        Message = Message + "DS18B20 1 not found"
    Else
        T_str = Str(t1)
        Tmp = Len(t_str) - 1
        Message = Message + "T1 = " + Left(t_str , Tmp) + "," + Right(t_str , 1) + "C. ERR=" + Str(tem_err)
        Locate 2 , 1 : Lcd Message
        If T1 < 20 Then
            Goto Snd_sms
        End If
    End If

    If Bd2(9) = Crc8(bd2(1) , 8) Then
        T2 = Makeint(bd2(1) , Bd2(2))
        T2 = T2 * 10 : T2 = T2 / 16
    Else
        Tem_err = 1
        Message = Message + "CRC 2 error"
    End If

    If Err = 1 Then                               ' если сенсор не обнаружен
        Tem_err = 2
        Message = Message + "DS18B20 2 not found"
    Else
        T_str = Str(t2)
        Tmp = Len(t_str) - 1
        Message = Message + ""
        Message = Message + " T2 = " + Left(t_str , Tmp) + "," + Right(t_str , 1) + "C. ERR=" + Str(tem_err)
        Locate 2 , 1 : Lcd Message
        If T2 > 20 Then
            Goto Snd_sms
        End If
    End If
End If

```

Рисунок 3.1 - Частина підпрограми опитування датчиків

4 РОЗРОБКА АПАРАТНОЇ ЧАСТИНИ

4.1 Загальний опис

Розроблений апаратно-програмний комплекс контролю температурних режимів різних процесів, реалізований на широко поширеному мікроконтролері AtMega88 фірми ATMEL.

Сформоване і налагоджене програмне забезпечення дозволяє реалізувати наступні функції спроектованої системи:

- вимірювання температури в контрольних точках об'єкта моніторингу;
- передачу інформації за заданою адресою із застосуванням радіочастотного каналу GSM на заздалегідь заданий номер, причому, як відповідно до заданих часових інтервалів, так і за сигналом запиту;
- виявлення і передачу сигналів в тих випадках, коли реальні значення температури виходять за встановлені допустимі межі.

Так само за допомогою програмного комплексу Proteus була створена модель пристрою, що підтримує обробку інформації з чотирьох датчиків.

У даній дипломній роботі крім мікроконтролера сімейства AVR, температурного датчика DS18B20 і мобільного телефону Siemens також використовувався LCD-дисплей. Даний дисплей використовується для виведення інформації, такої, як час і поточна температура.

На рис. 4.1 представлена схема підключення апаратних компонентів

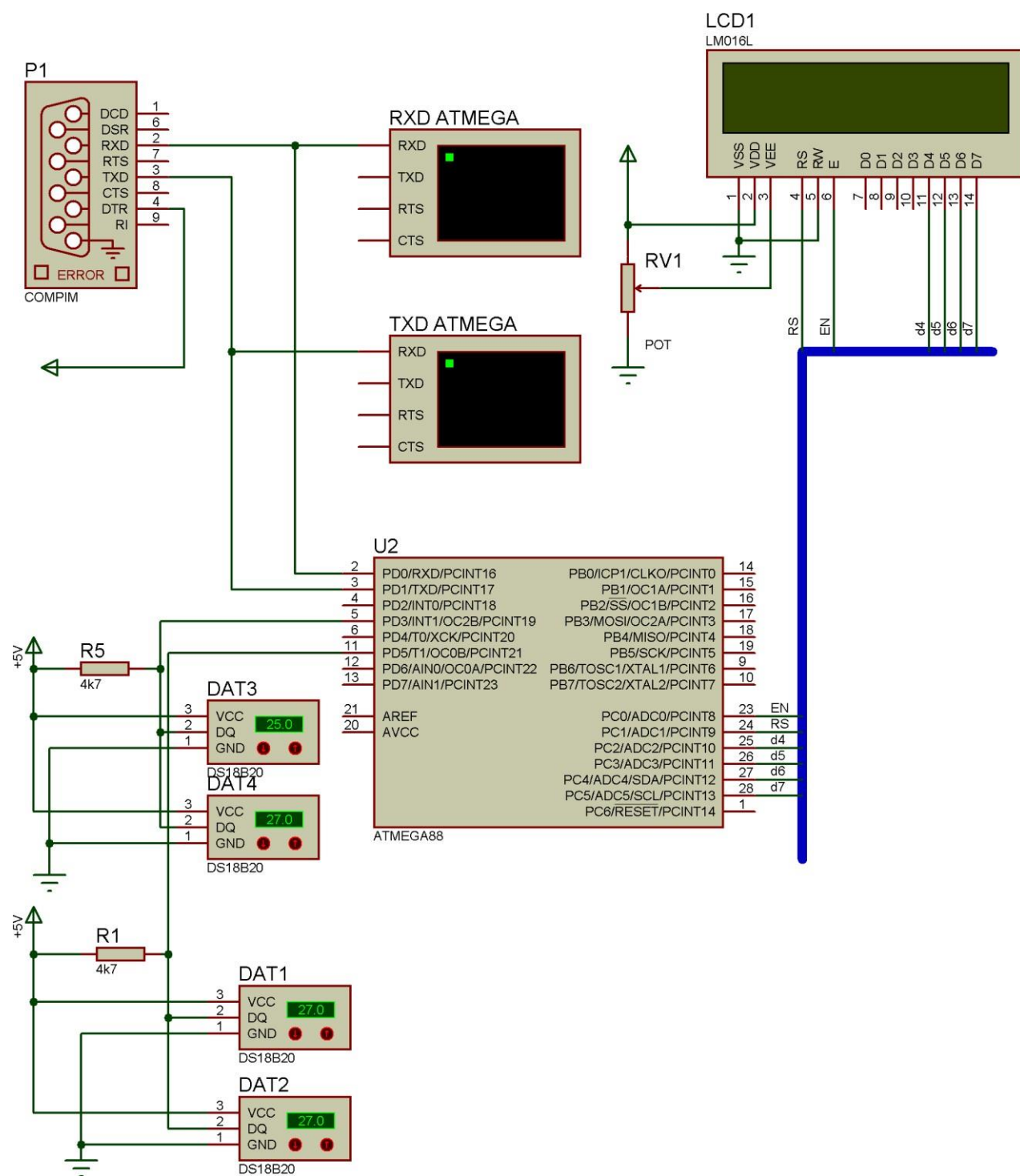


Рисунок 4.1 - Схема підключення датчиків DS18B20 до AtMega88

4.2 Загальні характеристики стандарту GSM

Відповідно до рекомендації СЕРТ 1980 р стосується використання спектра частот рухомого зв'язку в діапазоні частот 862-960 МГц, стандарт GSM на цифрову загальноєвропейську (глобальну) стільниковий систему наземної рухомого зв'язку передбачає роботу передавачів у двох діапазонах частот: 890-915 МГц (для передавачів рухомих станцій), 935-960 МГц (для передавачів базових станцій).

У стандарті GSM використовується вузькосмуговий багатостанційний доступ з часовим поділом каналів (TDMA).

Для захисту від помилок в радіоканалах при передачі інформаційних повідомлень застосовується блокове і сверточне кодування з перемеженням. Підвищення ефективності кодування і перемеження при малій швидкості переміщення рухливих станцій досягається повільним переключенням робочих частот в процесі сеансу зв'язку зі швидкістю 217 стрибків в секунду.

Для боротьби з інтерференційними завмираннями прийнятих сигналів, викликаними багатопроменим поширенням радіохвиль в умовах міста, в апаратурі зв'язку використовуються еквалайзери, що забезпечують вирівнювання імпульсних сигналів зі середньоквадратичним відхиленням часу затримки до 16 мкс.

Система синхронізації розрахована на компенсацію абсолютного часу затримки сигналів до 233 мкс, що відповідає максимальній дальності зв'язку або максимальному радіусу осередку (стільники) 35 км.

У стандарті GSM обрана гаусовська частотна маніпуляція з мінімальним частотним зсувом. Обробка мови здійснюється в рамках прийнятої системи переривчастої передачі мови, яка забезпечує включення передавача тільки при наявності мовного сигналу і відключення передавача в паузах і наприкінці розмови. Як мовоперетворюючий пристрій обраний мовний кодек з регулярним імпульсним збудженням і лінійним предикативним кодуванням з передбачуванням. Загальна швидкість перетворення мовного сигналу - 13 кбіт /с.

У стандарті GSM досягається високий ступінь безпеки передачі повідомлень; здійснюється шифрування повідомлень за алгоритмом шифрування з відкритим ключем.

В цілому система зв'язку, діюча в стандарті GSM, розрахована на її використання в різних сферах. Вона надає користувачам широкий діапазон послуг і можливість застосовувати різноманітне обладнання для передачі мовних повідомлень і даних, викличних та аварійних сигналів; підключатися до телефонних мереж загального користування, мереж передачі даних і цифрових мережах з інтеграцією служб.

Основні характеристики стандарту GSM:

- частоти передачі мобільної станції і прийому базової станції - 890-915 МГц;
- частоти прийому рухомий станції і передачі базової станції - 935-960 МГц;
- двобічний рознос частот прийому і передачі - 45 МГц;
- швидкість передачі повідомлень в радіоканалі - 270, 833 кбіт/с;
- швидкість перетворення мовного кодека – 13 кбіт/с;
- ширина смуги каналу зв'язку - 200 кГц;
- максимальна кількість каналів зв'язку – 124;
- максимальна кількість каналів, організованих в базовій станції 16-20;
- вид модуляції – гаусовська частотна маніпуляція з мінімальним частотним зсувом;
- індекс модуляції - BT 0,3;
- ширина смуги предмодуляційного гаусовського фільтра - 81,2 кГц;
- кількість стрибків по частоті в секунду – 217;
- часове рознесення в інтервалах TDMA кадру (передача / прийом) для рухомої станції – 2;
- вид мовного кодека - з регулярним імпульсним збудженням і лінійним предикативним кодуванням з передбаченням.

ВИСНОВКИ

У ході дипломної роботи було розглянуто принципи побудови систем охоронно-пожежних сигналізацій. Проаналізовано апаратне і програмне забезпечення проектування і розробки таких систем. Проведено аналіз сучасних датчиків температури, мікроконтролерів і SMS-повідомлень в форматі PDU. В роботі обрано і детально описано принципи роботи мікроконтролера AtMega88 і температурного датчика DS18B20. Також описано їх функціональні можливості і конструкції, програмування мікроконтролера і робота з вищезгаданими датчиками в Bascom. На базі мікроконтролера в програмному середовищі Proteus розроблено віртуальний пристрій, який вимірює температуру в контрольних точках об'єкта моніторингу та забезпечує її передачу за заданою адресою із застосуванням радіочастотного каналу GSM. Причому, як у відповідності із заданими часовими інтервалами, так і за сигналом запиту. Система окремо формує і передає сигнали в випадках, коли реальні значення температури виходять за встановлені допустимі межі.

В ході проведення натуральних випробувань в середовищі Proteus була продемонстрована повна працездатність всього комплексу в цілому. А це показує принципову можливість його застосування для налагодження програм і попередньої перевірки їх працездатності з наступним програмуванням реальних мікроконтрольних пристроїв управління.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Белов А.В. Конструирование устройств на микроконтроллерах.- СПб.: Наука и Техника, 2005. 256 с.
2. Бородин В.Б., Калинин А.В. Системы на микроконтроллерах и БИС программируемой логики. М.: Издательство ЭКОМ, 2002. 400 с.
3. Гребнев В.В. Микроконтроллеры семейства AVR фирмы ATMEL. М.: ИП РадиоСофт, 2002. 174 с.
4. Блох А.И., Гопкало В.Н. Проектирование радиоэлектронных устройств в программной среде «Proteus». Хабаровск:Издательство ДВГУПС, 2010. 20 с.
5. Карташевский В.Н., Фирстова Т.В. Сети подвижной связи. М.: Эко-трендз, 2001. 304 с.
6. Библиотека Гумер. Частная электронная библиотека URL: <http://www.gumer.info/> (дата звернення 06.05.2021).
7. Гигапром. Офіційний сайт фірми Гигапром URL: <http://www.gigaprom.ru/> (дата звернення 06.05.2021).
8. iButton. Офіційний сайт фірми iButton URL: <http://www.ibutton.ru/>(дата звернення 06.05.2021).
9. Maxim integrated. Сайт компанії Maxim integrated URL: <http://www.maximintegrated.com/en.html>(дата звернення 06.05.2021).
10. Decada. Інформаційний ресурс за тематикою мікроелектроніки URL: <http://decada.org.ru> (дата звернення 06.05.2021).
11. LabcenterElectronics. Сайт компанії LabcenterElectronics URL: <http://www.labcenter.com/index.cfm> (дата звернення 06.05.2021).
12. MCSelectronics. Сайт компанії MC Selectronics URL: <http://www.mcselec.com/> (дата звернення 06.05.2021).

ДОДАТОК А

Лістинги програм

Лістинг А1 - Config.h

```
#ifndef CONFIG_H_
#define CONFIG_H_

#define F_CPU          8000000UL
#define ONE_WIRE_PORT  PORTB
#define ONE_WIRE_DDR   DDRB
#define ONE_WIRE_PIN    PINB

#endif /* CONFIG_H_ */
```

Лістинг А2 - OneWire.h

```
#ifndef ONEWIRE_H_
#define ONEWIRE_H_

#define CMD_CONVERTTEMP    0x44
#define CMD_RSCRATCHPAD    0xbe
#define CMD_WSCRATCHPAD    0x4e
#define CMD_CPYSCRATCHPAD  0x48
#define CMD_RECEEPROM      0xb8
#define CMD_RPWRSUPPLY      0xb4
#define CMD_SEARCHROM      0xf0
#define CMD_READROM        0x33
#define CMD_MATCHROM       0x55
#define CMD_SKIPROM        0xcc
#define CMD_ALARMSEARCH    0xec

void oneWireInit(uint8_t);
void writeBit(uint8_t);
void writeByte(uint8_t);
void setDevice(uint64_t);
void searchRom(uint64_t*, uint8_t&);
void skipRom(void);
uint8_t readByte(void);
uint8_t readBit(void);
uint8_t reset(void);
uint8_t crcCheck(uint64_t, uint8_t);
uint64_t readRoom(void);
uint64_t searchNextAddress(uint64_t, uint8_t&);
```

```
extern uint8_t ONE_WIRE_DQ;

#endif /* ONEWIRE_H_ */
```

Лістинг A3 - OneWire.cpp

```
#define DEVICES_ERROR 1
#include "config.h"
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include "OneWire.h"

uint8_t ONE_WIRE_DQ = PINB0;

void oneWireInit(uint8_t pin) {
    ONE_WIRE_DQ = pin;
    ONE_WIRE_PORT |= (1 << ONE_WIRE_DQ);
    ONE_WIRE_DDR |= (1 << ONE_WIRE_DQ); // выход
}

/*
 * сброс
 */
uint8_t reset() {
    uint8_t response;

    // импульс сброса, минимум 480us
    ONE_WIRE_PORT &= ~(1 << ONE_WIRE_DQ);
    ONE_WIRE_DDR |= (1 << ONE_WIRE_DQ); // выход
    _delay_us(480);

    // Когда ONE WIRE устройство обнаруживает положительный перепад,
    он ждет от 15us до 60us
    ONE_WIRE_DDR &= ~(1 << ONE_WIRE_DQ); // вход
    _delay_us(60);

    // и затем передает импульс присутствия, перемещая шину в
    логический «0» на длительность от 60us до 240us.
    response = (ONE_WIRE_PIN & (1 << ONE_WIRE_DQ));
    _delay_us(410);

    // если 0, значит есть ответ от датчика, если 1 - нет
    return response;
}

/*
 * отправить один бит
```

```

*/
void writeBit(uint8_t bit) {
    if (bit & 1) {
        cli();
        // логический «0» на 1us
        ONE_WIRE_PORT &= ~(1 << ONE_WIRE_DQ);
        ONE_WIRE_DDR |= (1 << ONE_WIRE_DQ); // выход
        _delay_us(10);
        sei();
        ONE_WIRE_DDR &= ~(1 << ONE_WIRE_DQ); // вход
        _delay_us(55);
    } else {
        cli();
        // логический «0» на 1us
        ONE_WIRE_PORT &= ~(1 << ONE_WIRE_DQ);
        ONE_WIRE_DDR |= (1 << ONE_WIRE_DQ); // выход
        _delay_us(65);
        ONE_WIRE_DDR &= ~(1 << ONE_WIRE_DQ); // вход
        sei();
        _delay_us(5);
    }
}

/*
 * отправить один байт
 */
void writeByte(uint8_t byte) {
    uint8_t i = 8;
    while (i--) {
        writeBit(byte & 1);
        byte >>= 1;
    }
}

/*
 * получить один байт
 */
uint8_t readByte() {
    uint8_t i = 8, byte = 0;
    while (i--) {
        byte >>= 1;
        byte |= (readBit() << 7);
    }
    return byte;
}

/*
 * получить один бит
 */
uint8_t readBit(void) {
    uint8_t bit = 0;

```

```

cli();
// логический «0» на 1us
ONE_WIRE_PORT &= ~(1 << ONE_WIRE_DQ);
ONE_WIRE_DDR |= (1 << ONE_WIRE_DQ); // вход
_delay_us(3);

// освободить линию и ждать 14us
ONE_WIRE_DDR &= ~(1 << ONE_WIRE_DQ); // вход
_delay_us(10);

// прочитать значение
if (ONE_WIRE_PIN & (1 << ONE_WIRE_DQ)) {
    bit = 1;
}

// ждать 45us и вернуть значение
sei();
_delay_us(45);
return bit;
}

/*
 * читать ROM подчиненного устройства (код 64 бита)
 */
uint64_t readRoom(void) {
    uint64_t oneWireDevice;
    if(reset() == 0) {
        writeByte(CMD_READROM);
        // код семейства
        oneWireDevice = readByte();
        // серийный номер
        oneWireDevice |= (uint16_t)readByte()<<8 |
(uint32_t)readByte()<<16 | (uint32_t)readByte()<<24 |
(uint64_t)readByte()<<32 | (uint64_t)readByte()<<40 |
(uint64_t)readByte()<<48;
        // CRC
        oneWireDevice |= (uint64_t)readByte()<<56;
    } else {
        return 1;
    }
    return oneWireDevice;
}

/*
 * Команда соответствия ROM, сопровождаемая последовательностью
 * кода ROM на 64 бита позволяет устройству управления шиной
 * обращаться к определенному подчиненному устройству на шине.
 */
void setDevice(uint64_t rom) {
    uint8_t i = 64;
    reset();

```

```

writeByte(CMD_MATCHROM);
while (i--) {
    writeBit(rom & 1);
    rom >>= 1;
}
}

/*
 * проверка CRC, возвращает "0", если нет ошибок
 * и не "0", если есть ошибки
 */
uint8_t crcCheck(uint64_t data8x8bit, uint8_t len) {
    uint8_t dat, crc = 0, fb, stByte = 0;
    do {
        dat = (uint8_t) (data8x8bit >> (stByte * 8));
        for (int i = 0; i < 8; i++) { // счетчик битов в байте
            fb = crc ^ dat;
            fb &= 1;
            crc >>= 1;
            dat >>= 1;
            if (fb == 1) {
                crc ^= 0x8c; // полином
            }
        }
        stByte++;
    } while (stByte < len); // счетчик байтов в массиве
    return crc;
}

/*
 * поиск устройств
 */
void searchRom(uint64_t * roms, uint8_t & n) {
    uint64_t lastAddress = 0;
    uint8_t lastDiscrepancy = 0;
    uint8_t err = 0;
    uint8_t i = 0;
    do {
        do {
            lastAddress = searchNextAddress(lastAddress, lastDiscrepancy);
            if (lastAddress != DEVICES_ERROR) {
                uint8_t crc = crcCheck(lastAddress, 8);
                if (crc == 0) {
                    roms[i++] = lastAddress;
                    err = 0;
                } else {
                    err++;
                }
            } else {
                err++;
            }
        }
    } while (lastAddress != DEVICES_ERROR);
}

```

```

        if (err > 3) {
            return;
        }
        } while (err != 0);
    } while (lastDiscrepancy != 0 && i < n);
    n = i;
}

/*
 * поиск следующего подключенного устройства
 */
uint64_t searchNextAddress(uint64_t lastAddress, uint8_t &
lastDiscrepancy) {
    uint8_t searchDirection = 0;
    uint64_t newAddress = 0;
    uint8_t idBitNumber = 1;
    uint8_t lastZero = 0;
    reset();
    writeByte(CMD_SEARCHROM);

    while (idBitNumber < 65) {
        uint8_t idBit = readBit();
        uint8_t cmpIdBit = readBit();

        // id_bit = cmp_id_bit = 1
        if (idBit == 1 && cmpIdBit == 1) {
            return DEVICES_ERROR;
        } else if (idBit == 0 && cmpIdBit == 0) {
            // id_bit = cmp_id_bit = 0
            if (idBitNumber == lastDiscrepancy) {
                searchDirection = 1;
            } else if (idBitNumber > lastDiscrepancy) {
                searchDirection = 0;
            } else {
                if ((uint8_t) (lastAddress >> (idBitNumber - 1)) & 1) {
                    searchDirection = 1;
                } else {
                    searchDirection = 0;
                }
            }
        }
        if (searchDirection == 0) {
            lastZero = idBitNumber;
        }
        } else {
            // id_bit != cmp_id_bit
            searchDirection = idBit;
        }
        newAddress |= ((uint64_t) searchDirection) << (idBitNumber - 1);
        writeBit(searchDirection);
        idBitNumber++;
    }
}

```

```

    lastDiscrepancy = lastZero;
    return newAddress;
}

/*
 * пропуститъ ROM
 */
void skipRom() {
    reset();
    writeByte(CMD_SKIPROM);
}

```

Лістинг A4 - main.cpp

```

#include "config.h"

#include <avr/io.h>
#include <util/delay.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include "OneWire.h"
#include "LCD.h"

// 123.4
// numbers[0] = 123
// numbers[1] = 4
inline void explodeDoubleNumber(int* numbers, double flt) {
    numbers[0] = abs((int) flt);
    numbers[1] = abs((int) ((flt - ((int) flt)) * 10));
}

inline void printTemp(double d, uint8_t i) {
    char text[17] = "T[";
    int fs[2];
    char num[5];

    itoa(i, num, 10);
    strcat(text, num);
    strcat(text, "]=");

    explodeDoubleNumber(fs, d);
    if (d < 0) {
        strcat(text, "-");
    }
    itoa(fs[0], num, 10);
    strcat(text, num);
    strcat(text, ".");
    itoa(fs[1], num, 10);
    strcat(text, num);
}

```

```

    strcat(text, "'C");
    lcdClear();
    lcdGotoXY(0, 0);
    lcdPuts(text);
}

double getTemp(uint8_t pin) {
    uint8_t temperatureL;
    uint8_t temperatureH;
    double retD = 0;

    oneWireInit(pin);

    skipRom();
    writeByte(CMD_CONVERTTEMP);

    _delay_ms(750);

    skipRom();
    writeByte(CMD_RSCRATCHPAD);

    temperatureL = readByte();
    temperatureH = readByte();

    retD = ((temperatureH << 8) + temperatureL) * 0.0625;

    return retD;
}

int main(void) {
    _delay_ms(100);
    lcdInit();
    lcdClear();
    lcdSetDisplay(LCD_DISPLAY_ON);
    lcdSetCursor(LCD_CURSOR_OFF);

    double temperature;

    uint8_t pin = 0;

    while (1) {
        temperature = getTemp(pin);
        printTemp(temperature, pin);
        if (pin == 4) {
            pin = 0;
        } else {
            pin++;
        }
        _delay_ms(500);
    }
}

```

Загальна структура системи охоронно-пожежної сигналізації

Теплові датчики



Мікроконтролер



Модуль GSM SIM 900



					13.02070849.51714 ПЛ1				
					Розробка системи охоронно-пожежної сигналізації Загальна структура системи охоронно-пожежної сигналізації	Лист	Маса	Масштаб	
Зам.	Лист	№ докум.	Поп.	Дати					
Розроб.		Сверченко К.А.		2018/05/05					
Перев.		Грушко С.С.							
Т.контр.						Лист 1		Листів 1	
Н.контр.		Щербак Н.В.				НУ «Запорізька політехніка» КНТ-517			
Затв.		Кудерметов Р.К.							

Опис SMS-повідомлення, що відправляється

Окети	Опис
00	Індикатор, що показує довжину номера SMSC. Якщо він дорівнює "00", то використовується номер SMSC закладений в настройках телефону. Цей октет необов'язковий і в деяких телефонах може бути опущений.
11	Перший октет прийнятого SMS - повідомлення
00	TP - Message - Reference. Параметр, який використовується для ідентифікації повідомлення
0B	Довжина телефонного номера одержувача (11)
91	Тип адреси. "91" вказує на міжнародний формат телефонного номера.
6407281553F8	Телефонний номер одержувача в десяткових октетах (46708251358).
00	TP - PID. Ідентифікатор протоколу.
00	TP - DCS. Схема кодування даних (00 - режим 7-розрядної кодування). Наявність "02" замість "00" вказувало б на режим 8-розрядної кодування, що використовується для передачі даних.
AA	TP- Validity - Period. Час дії повідомлення.
0A	TP - User - Data - Length. Довжина поля даних користувача (довжина повідомлення). Поле TP - DCS вказало на режим 7-розрядної кодування, тобто повідомлення представлено 10ю 7-ми розрядними байтами. Якщо TP - DCS було б встановлено в режим 8 або 16-розрядної кодування, то число байт в цьому випадку дорівнювало б 9.
E8329BFD4697D9 EC37	Повідомлення "hellohello" представлене октетами. Перехід від 7-розрядних байтів до октетам було показано вище.

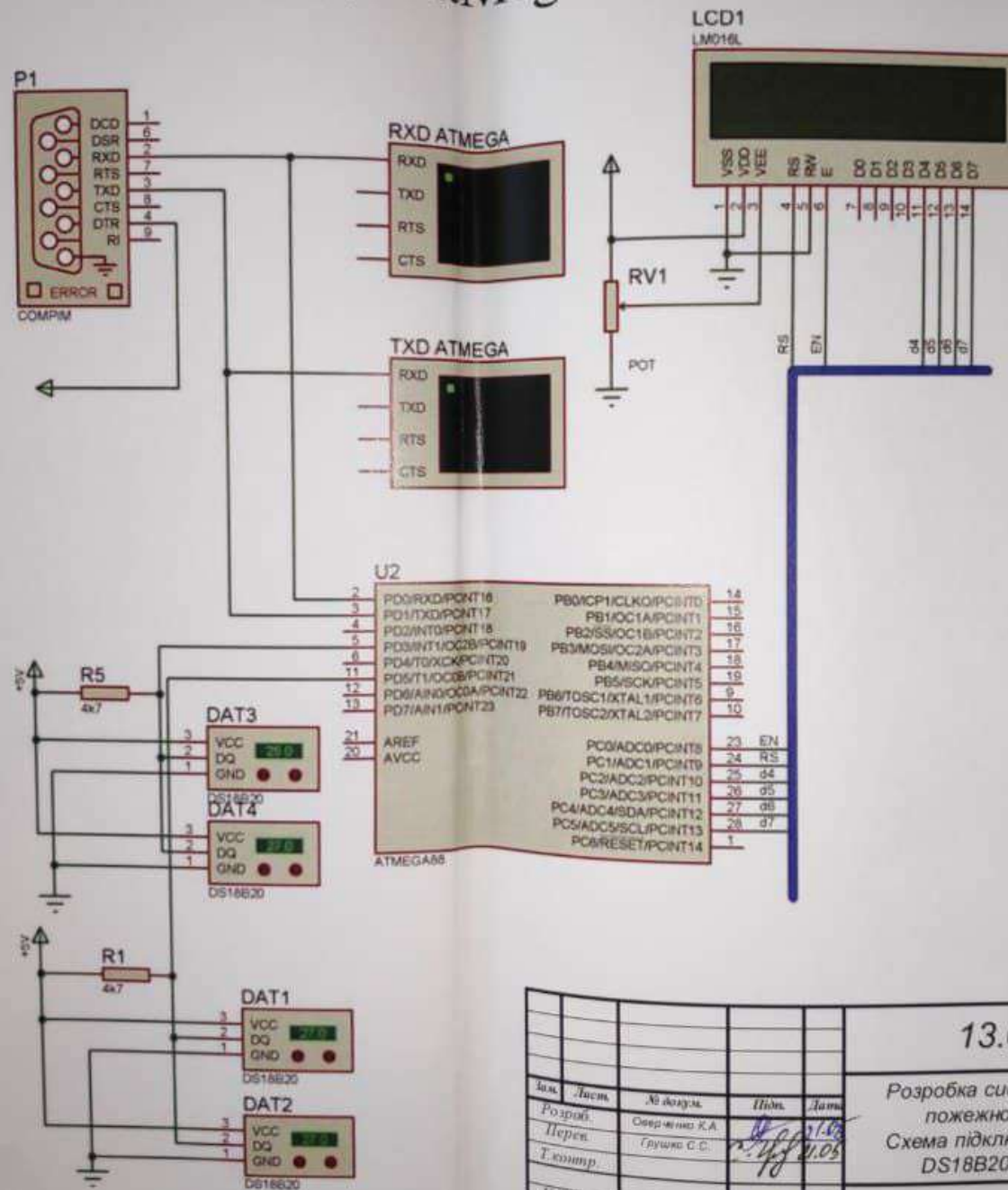
					13.02070849.51714 ПЛ2		
					Розробка системи охоронно-пожежної сигналізації		
					Опис SMS-повідомлення, що відправляється		
Вам.	Лист.	№ докум.	Підп.	Дат.	Лист.	Маса	Масам.
Розроб.	Овчаренко К.А.						
Перев.	Грушко С.С.				Лист 1		Листів 1
Т.контр.					НУ «Запорізька політехніка»		
Н.контр.	Щербак Н.В.				КНТ-517		
Зам.	Кудерметов Р.К.						

Опис структури вхідного SMS- повідомлення

Октети	Опис
07	Індикатор показує довжину номера SMSC (в цьому випадку 7 байт).
91	Код інтернаціональності SMSC.
7283010010F5	Номер центру обробки повідомлень - SMSC (представлений десятковими октетами). Довжина номера непарна (11), тому в кінець додано 0F, щоб сформувати належні октети. Номер телефону SMSC - "+ 27381000015 "
04	Перші октети вихідного SMS - повідомлення.
0B	Довжина номера відправника (0BHEX = 11DEC)
C8	Тип адреси номера відправника
7238880900F1	Телефонний номер відправника (десяткові октети). Номер закодований аналогічно номеру SMSC.
00	TP - PID. ідентифікатор протоколу
00	TP - DCS. Схема кодування даних (00 - режим 7 розрядної кодування).
99309251619580	Часова мітка (октети).
0A	TP - UDL. Довжина поля даних користувача (довжина повідомлення). Поле TP - DCS вказало на режим 7-розрядної кодування, тобто повідомлення представлено 10 7-розрядними байтами. Якщо TP - DCS було б встановлено в режим 8 або 16-розрядної кодування, то число байт в цьому випадку дорівнювало б 9
E8329BFD4697D9EC37	TP - UD. Поле даних користувача. Повідомлення "hellohello" представлено 8-розрядними байтами, перетвореними з 7-розрядних.

13.02070849.51714 ПЛЗ				
Розробка системи охоронно-пожежної сигналізації				
Опис структури вхідного SMS-повідомлення				
Зам.	Лист	№ докум.	Шук.	Дата
Розроб.		Сверченко К.А.		21.02.2013
Перев.		Гришко С.С.		21.02.2013
Т. контр.				
Н. контр.		Щербак Н.В.		
Затв.		Кудряшов Р.К.		
Лист 1				
Листів 1				
НУ «Запорізька політехніка» КНТ-517				

Схема підключення датчиків DS18B20 до AtMega88



13.02070849.51714 ПЛ4				
Вид. Лист	№ докум.	Підп.	Дата	Розробка системи охоронно-пожежної сигналізації
Розроб.	Оверченко К.А.	В.В.В.	21.05	Схема підключення датчиків DS18B20 до AtMega88
Перев.	Грушка С.С.	В.В.В.	21.05	
Т. контр.				
Н. контр.	Щербак Н.В.			
Зам.	Куберикова Р.Х.			
Лист 1				
Листів 1				
НУ «Запорізька політехніка» КНТ-517				