

Центральноукраїнський національний технічний університет

Факультет будівництва, транспорту та енергетики

Кафедра "Автоматизації виробничих процесів"

"Допущено до захисту"

Зав.кафедрою АВП

к.т.н., професор

_____ Дідик О.К.

" _____ " _____ 2023р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої
освіти

на тему

" Система автоматичного контролю безпеки підприємства за
допомогою ідентифікатора особи "

Виконав: здобувач вищої освіти

III курсу, групи АК-20-3ск

ОПП «Автоматизація та комп'ютерно-
інтегровані технології»

спеціальність 151 «Автоматизація та
комп'ютерно-інтегровані
технології»

_____ Братушко Д. А.

Керівник роботи

доцент, канд.техн.наук

_____ Мірошніченко М. С.

" _____ " _____ 2023р.

Рецензент _____

м. Кропивницький

Центральноукраїнський національний технічний університет
Факультет Будівництва, транспорту та енергетики
Кафедра Автоматизації виробничих процесів
Рівень вищої освіти бакалавр
Галузь знань 15 Автоматизація та приладобудування
Спеціальність 151 «Автоматизація та комп'ютерно-інтегровані технології»
Освітньо-професійна програма Автоматизація та комп'ютерно-інтегровані технології

ЗАТВЕРДЖУЮ
Завідувач кафедри Автоматизації
виробничих процесів
Дідик О. К.
“ ” 2023 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ
за першим (бакалаврським) рівнем вищої освіти
здобувача вищої освіти

Братушко Дмитра Анатолійовича
(прізвище, ім'я, по батькові)

1.Тема роботи Система автоматичного контролю безпеки підприємства за допомогою ідентифікатора особи

2.Керівник роботи Мірошніченко М.С., к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

3.Строк подання роботи до захисту 7 червня 2023р

4. Мета та завдання випускової кваліфікаційної роботи
Метою кваліфікаційної роботи є: вдосконалення системи контролю безпеки великого підприємства . Розробка програмного коду для сучасного мікроконтролеру

5. Завдання кваліфікаційної роботи: Провести огляд відомих рішень та обґрунтувати вибір прийнятого принципу побудови пристрою. Опис та обґрунтування обраних проектних рішень та конструкцій. Зробити розрахунки, що підтверджують працездатність та надійність пристрою, надійність спроектованої системи

5. Консультанти по роботі, із зазначенням розділів роботи

Розділ	Консультант	Підпис, дата	
		завдання видав	завдання прийняв

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Призначення та область застосування пристрою	5.04.2023	
2	Огляд відомих рішень та обґрунтування вибору пристрою	15.04.2023	
3	Опис та обґрунтування обраних проектних рішень та конструкцій	29.04.2023	
4	Розрахунки, що підтверджують працездатність та надійність пристрою	5.05.2023	
5	Оформлення бакалаврської роботи	12.05.2023	

6. Дата видачі завдання

“ ____ ” _____ 20__ р.

Підпис керівника

_____ Мірошніченко М.С.

Завдання прийнято до виконання

“ ____ ” _____ 20__ р.

Підпис здобувача

_____ Братушко Д.А.

АНОТАЦІЯ

Кваліфікаційна бакалаврська робота присвячена розробці автоматизованої системи контролю доступу до приміщення та вивченню нових технологій в автоматизації на основі мікропроцесорних систем.

В роботі розглянутий теоретико-методологічний розділ, який містить коротку характеристику сфери призначення системи, а також обґрунтування її актуальності; розглянуті існуючі рішення реалізації проекту, наведена порівняльна характеристика комплектуючих системи та обґрунтовано прийняте рішення побудови.

Бакалаврська робота містить схеми та опис розробленої системи, а також описуються використані компоненти та їх детальна характеристика.

Приведені в роботі матеріали пояснювальної записки будуть корисними для вивчення сфери автоматизації і для вивчення побудови мікропроцесорних систем.

Програмне забезпечення розроблено за допомогою мови програмування C++ з використанням спеціальних бібліотек.

ANNOTATION

The qualifying bachelor's work is devoted to the development of an automated system for controlling access to the premises and the study of new technologies in automation based on microprocessor systems.

The theoretical and methodological section is considered in the work, which contains a brief description of the scope of the system's purpose, as well as justification of its relevance; the existing solutions for the implementation of the project are considered, the comparative characteristics of the components of the system are given, and the construction decision is justified.

The bachelor's thesis contains diagrams and a description of the developed system, as well as describes the used components and their detailed characteristics.

The materials of the explanatory note presented in the work will be useful for studying the field of automation and for studying the construction of microprocessor systems.

The software is developed using the C++ programming language using special libraries.

Зміст

Вступ	2
1 Призначення та область застосування пристрою	4
2 Огляд відомих рішень та обґрунтування вибору прийнятого принципу побудови пристрою	15
3 Опис та обґрунтування обраних проектних рішень та конструкцій	25
3.1 Опис структурної схеми роботи пристрою	25
3.2 Опис електричної принципової схеми пристрою	32
3.3 Arduino UNO	35
3.4 Опис датчика AT77C101B	39
3.5 Опис мікроконтролера Atmega 328	41
3.6 Блок-схема алгоритму роботи програми	43
3.7 Опис дисплея 1602	45
4. Розрахунки та експериментальні матеріали, що підтверджують працездатність та надійність пристрою	47
4.1 Розрахунок надійності спроектованої системи	47
4.2 Розрахунок однофазного випрямляча малої потужності	54
5. Опис організації робіт, пов'язаних із застосуванням виробу	57
Висновки	62
Список літератури	63
Додатки	

Вступ

Система контролю управління доступом (СКУД) дозволить візуально контролювати вхід/вихід, і навіть спробу входу в недозволене доступом приміщення співробітника або відвідувача об'єкта, що охороняється. Система контролю управління доступом дозволить, крім обмеження доступу, призначити кожному співробітнику індивідуальний часовий графік роботи, зберегти та потім переглянути інформацію про події за день. Системи можуть працювати в автономному режимі та під управлінням комп'ютера. Залежно від розмірів об'єкта, кількості приміщень і чисельності персоналу може вибиратися той чи інший вид устаткування та програмного забезпечення.

Система контролю доступу в якості компонента загальної системи безпеки дозволяє:

- виключити потрапляння на територію підприємства сторонніх осіб. Система виводить на монітор охорони фотографію людини, проходить через прохідну, здійснює пропуск співробітників строго відповідно до графіка роботи даної категорії персоналу, а також виконує облік робочого часу;
- здійснити автоматичний доступ в різні приміщення тільки тих співробітників підприємства, які внесені в список допущених для проходу в кожне з контрольованих приміщень;
- формувати протокол подій.

Система контролю управління доступом складається з:

- основних елементів (комп'ютер, контролери, програматор, зчитувачі);
- виконавчі елементи (електромеханічні замки, електрозащіпки, електромагнітні замки, турнікети);
- якщо підбір основного обладнання залежить від складності об'єкта, що захищається, та кількості користувачів в системі, то виконавчі елементи залежать від матеріалу (поверхні), на яку вони будуть змонтовані, при цьому необхідно враховувати також естетичні нюанси.

Сучасні СКУД легко інтегруються з 1С-бухгалтерією (для формування табеля робочого часу) та іншими системами безпеки. СКУД також використовуються в сфері розважального бізнесу - це забезпечення платного проходу та оплата послуг на об'єктах де є зацікавленість використання єдиної системи оплати (єдина карта з якимось грошовим еквівалентом) - це катки, гірськолижні курорти та ін.

При виборі систем безпеки необхідно враховувати, що з часом системи безпеки мають властивість морально застарівати та стають більш уразливими для зловмисників. З кожним роком з'являються нові й більш доступні засоби та методи, за допомогою яких можна обдурити систему безпеки та завдати шкоди у вигляді викрадення матеріальних цінностей або інформації. В середньому розрахований термін служби системи безпеки 5-10 років, по закінченню цього терміну систему доводиться модернізувати або міняти повністю (в залежності від того, яке обладнання було встановлено спочатку).

Ефективність системи контролю доступу помітно зростає при інтеграції з системами відеоспостереження та сигналізації [1].

Для обслуговування великого об'єкта бажано встановлювати комплекс автоматизованої системи безпеки, який складається з:

- системи пожежної сигналізації;
- системи оповіщення про пожежу;
- системи охоронної сигналізації;
- системи охоронного відеоспостереження;
- системи контролю управління доступом.

1 Призначення та область застосування виробу

Основне призначення систем контролю доступу – організувати контрольно-пропускний режим на підприємстві в автоматичному режимі, щоб особи з правом доступу до конкретних приміщень та територій могли безперешкодно туди потрапити, одночасно при цьому обмеживши доступ сторонніх осіб.

Установка СКУД дозволить забезпечити більш високий рівень безпеки на підприємстві.

Система контролю і управління доступом призначена для автоматичного управління входом і виходом людей з будівлі і приміщення організації, в'їздом і виїздом транспорту на територію.

Монтаж СКУД дозволить забезпечити більш високий рівень безпеки на підприємстві.

Крім цього з'явиться можливість отримання точної інформації про дотримання робочого графіка співробітниками, гнучкого завдання режимів доступу, обліку і контролю робочого часу з урахуванням індивідуальних і змішаних графіків роботи, обмеження доступу по прохідним, за часом доби, по днях тижня з урахуванням святкових днів.

З урахуванням даних СКУД можливо нарахування заробітної плати.

Розглянемо принцип роботи системи контролю і управління доступом.

Кожен співробітник або відвідувач фірми одержує ідентифікатор - пластикову картку або брелок з вмісту в ній індивідуальним кодом.

«Електронні ключі» видаються в результаті реєстрації перерахованих осіб за допомогою засобів системи.

Фото та відомості про власника «електронного ключа» заносяться в персональну «електронну картку».

Персональна «електронна картка» власника і код його «електронного ключа» зв'язуються один з одним і заносяться в спеціально організовані комп'ютерні бази даних.

В приміщення, які підлягають контролю, встановлюються зчитувачі, що зчитують з карток їх код та інформацію про права доступу власника карти і передають цю інформацію в контролер системи.

У системі кожному коду поставлена у відповідність інформація про права власника картки.

На основі зіставлення цієї інформації і ситуації, при якій була пред'явлена картка, система приймає рішення: контролер відкриває або блокує двері (замки, турнікети), переводить приміщення в режим охорони, включає сигнал тривоги і т.д.

Усі факти пред'явлення карток і пов'язані з ними дії (проходи, тривоги і т.д.) фіксуються в контролері і зберігаються в комп'ютері.

Інформація про події, викликаним пред'явленням карток, може бути використана в подальшому для отримання звітів по обліку робочого часу, порушень трудової дисципліни та ін.

Системи контролю й керування доступом можуть мати різні конфігурації: від простих, розрахованих на одні двері до дуже складних, розрахованих на забезпечення контролю та управління доступом заводів, підприємств, банків і т.д.

Кожна система доступу, якою б складною вона не була, обов'язково включає в себе наступне: контролери, зчитувачі (пристрої ідентифікації) та обладнання обмеження доступу-виконавчі пристрої (електромагнітні замки, засувки, турнікети і т.д.).

Контролери - це основна частина системи контролю доступом. Саме контролери приймають рішення на дозвіл або заборону доступу в приміщення.

Коли співробітник або відвідувач пред'являє ідентифікатор (наприклад, proximity карту), з нього код порівнюється з кодом, що зберігається в пам'яті контролера.

На підставі порівняння охоронна система дозволяє або, забороняє прохід на територію охороняемого об'єкта.

За способом управління пристроями (дверима, турнікетами, шлюзами і т.д.) усі системи можна класифікувати наступним чином:

- автономні (локальні) – для управління одним або декількома пристроями, без передачі інформації на центральний пульт і без контролю з боку оператора;
- централізовані (мережеві) – для управління пристроями з обміном інформацією з центральним пультом, контролем і управлінням системою з боку оператора;
- універсальні, що включають функції як автономних, так і мережевих систем, що працюють в мережевому режимі під керівництвом центрального пристрою управління і перехідні в автономний режим при виникненні відмов у мережевому обладнанні або центральному пристрої.

За кількістю точок доступу і користувачів СКУД можна підрозділяти на:

- малі – одиниця точок доступу (офіси);
- середні – десятки точок доступу і тисячі користувачів (банки, підприємства, установи, готелі);
- великі – сотні точок доступу і десятки тисяч користувачів (великі промислові підприємства, аеропорти).

Автономні системи призначені для забезпечення контролю і управління доступом в окреме приміщення.

Така система складається з автономного контролера, що зберігає в собі базу даних ідентифікаторів і керує роботою інших елементів системи. Як виконавчий пристрій використовується електромагнітний замок.

Для ідентифікації користувача використовуються різні типи карт з відповідними зчитувачами (магнітні, проксіміті, штрихові).

Контролер працює, як правило, з одним зчитувачем на вхід (зчитувач на вихід використовується лише в особливо важливих приміщеннях); для

відкриття замку зсередини приміщення зазвичай використовується або кнопка, або датчик-комутатор руху.

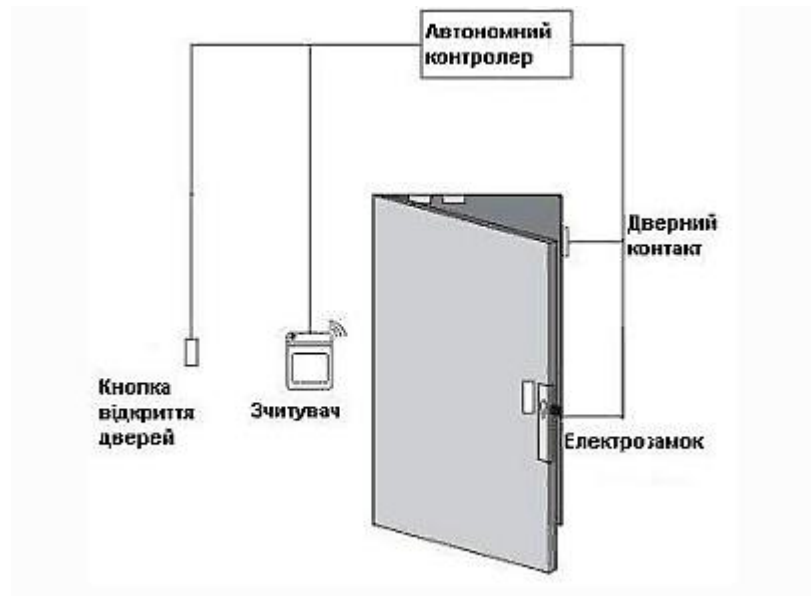


Рисунок 1.1 – Автономні системи контролю доступу

Для забезпечення правильності роботи всієї системи використовується датчик положення дверей [2].

Єдиною відмінністю є можливість системи накопичувати інформацію про всі проходи через точку проходу (двері, турнікет) – час, дата, ідентифікаційний номер проксиміті карти або брелка Touch Memory.



Рисунок 1.2 – Считувач Proximity ZKTeco KR600E

Для забезпечення нормального режиму роботи системи контролю доступу, двері оснащуються доводчиком, а контролер – резервним джерелом живлення [2].



Рисунок 1.3 – Брелок Touch memory

Автономні системи з накопиченням інформації, виконують ті ж завдання, що і системи описані вище [2].

Дана інформація зберігається в пам'яті контролера і може бути за потребою переглянута. Для отримання інформації і представлення її в наочному вигляді використовується спеціальне програмне забезпечення (ПЗ), яке дозволяє оператору системи вирішувати такі завдання: відстеження переміщення співробітників по території, обліку робочого часу співробітників, візуально контролювати особу власника проксиміті картки або брелка Touch Memory.



Рисунок 1.4 – Автономні системи доступу зі збором даних

Виділяють чотири характерних типи точок доступу, де може бути застосований контроль:

- прохідні;
- офісні приміщення;

- приміщення з підвищеним рівнем безпеки;
- об'єкти на вулиці (ворота, шлагбауми для автостоянок і т.д.).

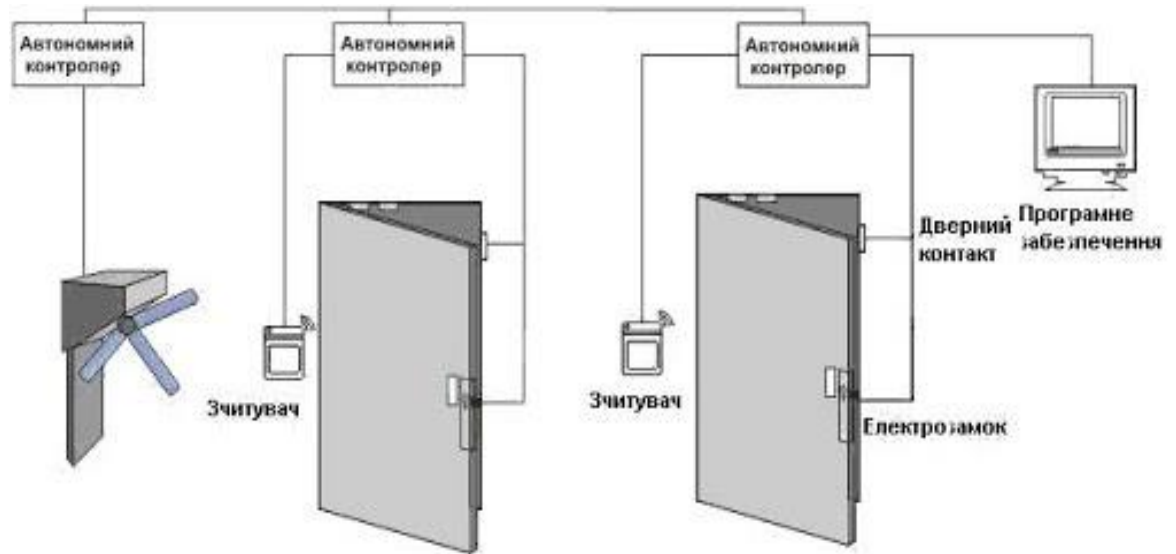


Рисунок 1.5 – Мережеві системи контролю

Мережеві системи призначені для забезпечення контролю і управління доступом на великих об'єктах (банки, установи, підприємства і т.п.).

Прохідні призначені для контролю за доступом при вході на об'єкт. У більшості випадків тут використовуються турнікети або шлюзи і, як правило, присутня охорона, яка проводить додаткову візуальну перевірку і стежить за виникненням критичних ситуацій [2].



Рисунок 1.6 – Різновид турнікетів

При великій кількості службовців виникають проблеми з пропускною спроможністю, тому додатковий контроль проводиться тільки для відвідувачів.

Турнікети можуть бути напівростового типу, рідше використовуються турнікети з перекриттям отвору в повний зріст. Турнікети в більшості своїй не забезпечують фізичної перешкоди для проникнення, а служать тільки для впорядковування потоку людей і допомагають службі охорони контролювати вхід на об'єкт.

Таким чином, основне їх завдання – обмежити можливість проходу декількох чоловік по одному допуску (ідентифікатору). Турнікети можуть бути обладнані засобами сигналізації, що спрацьовують при спробах обходу, перестрибуванні.

Для цього використовуються ІЧ-бар'єри і т.п. Шлюзи або кабінки відносяться до пристроїв блокуючого типу. Вони можуть при необхідності тримати людину в точці доступу.

Шлюзи застосовуються на об'єктах з підвищеним рівнем безпеки. При цьому вони часто обладнуються додатковими засобами контролю (металодетекторами, детекторами вибухових і радіоактивних речовин і т.п.).

На прохідних може виникнути потреба в телевізійному контролі. При цьому введення коду, або зчитування ідентифікатора користувача, супроводжується введенням зображень з бази даних і з телекамери на монітор охоронця, (режим відеоідентифікації), який проводить їх порівняння.

Для підвищення рівня безпеки застосовується дворівнева ідентифікація наприклад, карта і код, карта і вага.

Як правило, в самих офісних приміщеннях потрібний невисокий рівень захисту. Тут можна застосувати електромагнітні замки та зчитувачі дистанційного типу з великою відстанню зчитування (для того, щоб службовці не виймали картки з кишені – принцип вільних рук).

Як виконавчі пристрої можуть застосовуватися електромагнітні або електромеханічні замки. Слід врахувати, що для їх нормального функціонування необхідні дверні доводчики.



Рисунок 1.7 – Електромагнітний замок



Рисунок 1.8 – Доводчик дверей

У приміщеннях такого типу повинна застосовуватися багаторівнева ідентифікація особи (не менше двох рівнів), при необхідності – з біометричною ідентифікацією і клавіатурними зчитувачами. Тут також може використовуватися принцип «доступу двох або більше осіб» [2].

Клавіатурні зчитувачі, хоча і вважаються недостатньо захищеними від різних маніпуляцій (підбір коду, спостереження), мають певні переваги, наприклад, розрядність коду може бути вибрана довільно, причому код встановлюється користувачем і невідомий оператору системи.

Особливість даних об'єктів обумовлена застосуванням спеціальних загороджуючих пристроїв (автоматичні ворота, шлагбауми), а також

спеціальним кліматичним виконанням зчитувачів і іншої апаратури для вуличної роботи.



Рисунок 1.9 – Автоматичні ворота

Як об'єкти контролю на вулиці може служити автотранспорт. Для цього СКУД мають у своєму складі спеціальні автомобільні ідентифікатори, зчитувачі для установки під полотном дороги, дистанційні зчитувачі.

Мережеві системи контролю та управління доступом можуть забезпечувати ряд додаткових можливостей:

- збір і обробку інформації про переміщення осіб по об'єкту;
- організацію і облік робочого часу;
- управління режимами роботи і автоматикою автостоянок.

Контролери, що забезпечують контроль і керування доступом в одне або декілька приміщень, об'єднуються в мережу (як правило на основі інтерфейсу RS-485) і підключаються до керуючого комп'ютера.

Така побудова СКУД, дозволяє оператору централізовано програмувати контролери, отримувати повну інформацію про факти і спроби проходів на всьому контрольованому об'єкті в реальному масштабі часу, оперативно блокувати/деблокувати проходи в екстрених ситуаціях. На малих і середніх об'єктах, як правило, організовують одне робоче місце оператора СКУД, на яке збирається інформація з усієї системи.

При обриві комунікацій або виході з ладу керуючого комп'ютера, контролери переходять в автономний режим роботи, накопичуючи події проходів у власній пам'яті.

При відновленні комунікацій контролери пересилають все, що накопичилося на керуючий комп'ютер, забезпечуючи інформаційну цілісність роботи системи.

Основною особливістю побудови СКД на великих об'єктах, є організація декількох автоматизованих робочих місць (АРМ) на базі персональних комп'ютерів, об'єднаних в локальну комп'ютерну мережу.

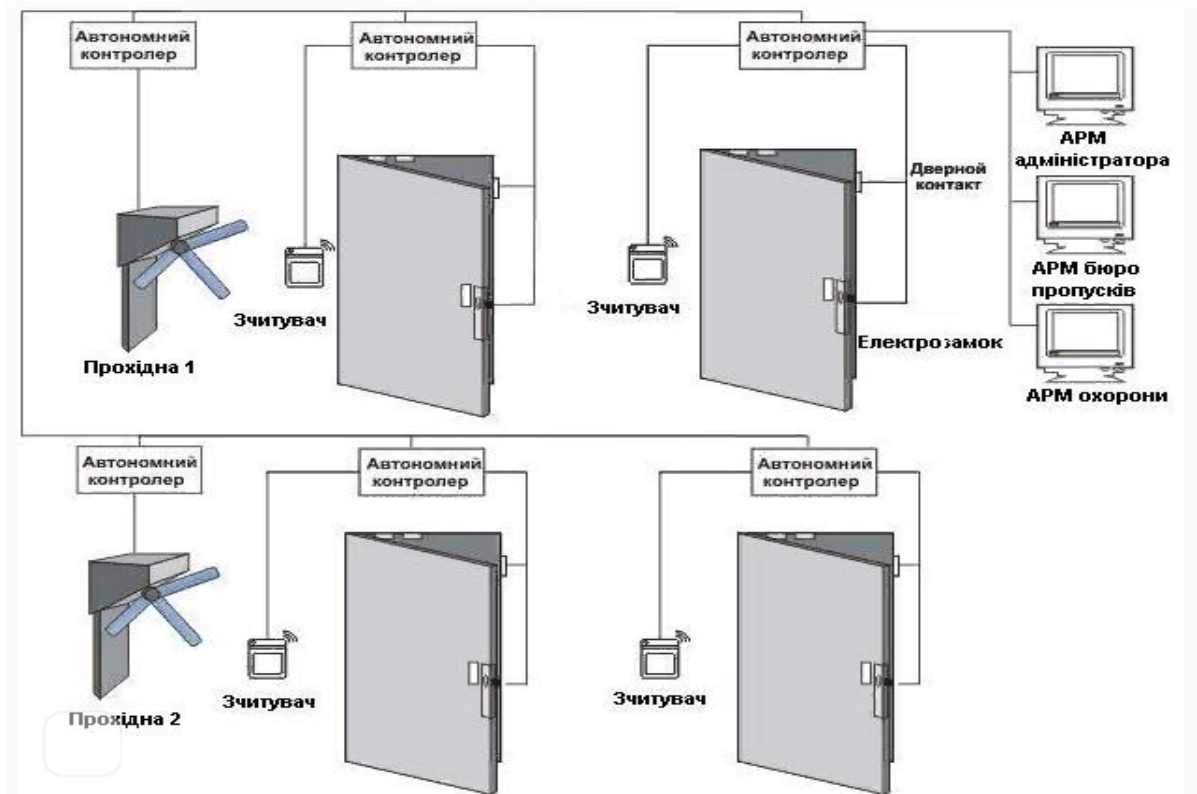


Рисунок 1.10 – Особливості мережевих СКУД для великих об'єктів

Весь персонал, що обслуговує СКУД, має доступ до єдиної бази даних, що знаходиться на сервері системи. Такі системи зазвичай містять АРМ: «прохідна» (з можливістю відеоідентифікації), бюро перепусток, адміністратор, черговий служби безпеки [2].

У системі використовується кілька мереж контролерів, кожна з яких підключається до територіально найближчого комп'ютера, але управління всією системою може здійснюватися з кожного робочого місця при наявності відповідних прав у оператора.

Побудована за такою схемою СКУД може забезпечити централізоване управління на об'єкті, що містить декілька будівель, значно віддалених один від одного.

Система контролю доступу може являти собою одну з частин інтегрованої системи безпеки, встановленої на підприємстві. Інтегрована система безпеки поєднує в собі комплекс технічних і програмних засобів щодо забезпечення безпеки підприємства і його життєдіяльності.

Інтегрована система дозволяє вирішувати такі основні завдання - управління доступом; охоронні функції; управління системами життєзабезпечення [2].

Управління доступом:

- захист від проникнення сторонніх осіб;
- збір та обробку інформації про переміщення осіб по об'єкту;
- організацію та облік робочого часу;
- управління режимами роботи і автоматикою автостоянок.

Охоронні функції:

- забезпечення охоронної сигналізації;
- забезпечення пожежною сигналізацією;
- управління системою телевізійного спостереження (СТН).

Управління системами життєзабезпечення:

- управління електропостачанням;
- управління освітленням;
- управління ліфтами і ескалаторами;
- управління системою вентиляції і кондиціонування.

2 Огляд відомих рішень та обґрунтування вибору прийнятого принципу побудови виробу.

Ідентифікація за відбитками пальців (дактилоскопія) - найбільш поширена на сьогоднішній день технологія, що застосовується в біометричних системах контролю доступу. В основі технології - унікальність малюнка папілярних візерунків на пальцях людей.

Відбиток, отриманий за допомогою сканера, перетворюється в цифровий код, який і зберігається в базі даних, а потім порівнюється з раніше введеними і перетвореними «кодами відбитків пальців». Приклад сканера відбитків пальця наведено на рисунку 1.



Рисунок 2.1 – Сканер відбитків пальця

Переваги біометричного контролю доступу з використанням відбитка пальця - легкість у використанні, зручність і надійність, висока достовірність і низька вартість пристроїв, скануючих зображення відбитка пальця.

Серед недоліків: порушення папілярного візерунка дрібними подряпинами, порізами, хімічними реактивами; неможливість зчитування відбитку деякими сканерами при надмірно сухій шкірі.

Середнє значення показника FAR контролю доступу з використанням відбитка пальця - 0.001%. Стабільна робота системи ідентифікації при FAR = 0.001% можлива при чисельності персоналу $N \approx 300$.

На даний момент системи ідентифікації за відбитками пальців складають більше половини ринку біометрії.

Системи контролю доступу по відбитку пальця названі більш ефективними з економічної точки зору, не дивлячись на більш низьку вартість традиційних зчитувачів.

В першу чергу, мова йде про потенційні недоліки захисту від несанкціонованого доступу традиційних СКУД, які біометрія може зменшити.

Оскільки біометричний ідентифікатор не можна втратити або забути де або - зростає надійність системи, особливо для великих об'єктів.

Знижується додана вартість системи для об'єктів з великою плинністю постійних користувачів, таких як студентські гуртожитки, де заміна ідентифікаторів може значно збільшити бюджет проекту

Неможливість передати біометричний ідентифікатор третій особі забезпечує точність в роботі систем обліку робочого часу, а, отже, і розрахунку зарплати співробітників.

Крім того, системи ідентифікації за відбитками пальців дозволяють з мінімальними витратами реалізувати рішення, використовувані для різних додаткових цілей, таких як управління системою харчування, бібліотеками, архівами тощо.

Технологія Match-on-Host є галузевим стандартом. Система складається з датчика відбитків пальців, що зчитує біометричні дані з відправкою на центральний зовнішній процесор. Вся обробка та пошук співпадаючих відбитків здійснюється на зовнішньому сервері.

Така система аунтифікації за відбитками пальців доступна, економічна, може бути досить просто і швидко інтегрована в будь-яку вже існуючу СКУД.

Популярність Match-on-Host технології привела до інновацій в суміжних областях, таких як Fast Identity Online (FIDO) сформованої Universal Authentication Framework (UAF).

Технологія Match-in-Sensor має архітектуру, замкнуту на самому чіпі (system-on-a-chip або SoC). Зчитування відбитка пальця і вся подальша обробка біометричних даних здійснюється безпосередньо в ІС-датчику. Така архітектура є більш безпечною, адже шаблони відвідуваності зашифровані і підписані за допомогою датчика, а вся інформація зберігається в приватній флеш-пам'яті.

У доповіді Electronic Design, Synaptics 'VP of marketing Anthony Gioeli робиться однозначний вибір на користь останнього методу аунтифікації за відбитками пальців, оскільки така замкнута система стає набагато менш доступною для злому і крадіжки хакерами унікальної інформації про біометричні дані користувача.

Існування стандартного шаблону відбитка пальця INCITS 378 дає можливість розробки повністю сумісних програм, що дозволяють здійснювати швидкий і точний обмін компактними біометричними шаблонами і подальшу обробку даних. На практиці, біометричні дані, згенеровані різними виробниками можуть, оброблятися з меншою швидкістю і точністю, збільшуючи кількість відмов системи.

Програма тестування шаблонів відбитка пальця і алгоритмів їх обробки MINEX III (Minutiae Interoperability Exchange Test) дозволяє здійснити вимірювання продуктивності і сумісності кодування шаблонів ядра (включаючи обмеження на швидкість роботи алгоритму, споживання пам'яті, час виконання і т.д.), що допомагає оцінити відповідні можливості для користувачів, постачальників і зацікавлених сторін.

Крім того, тест використовується для встановлення відповідності генераторів шаблонів і обчислювачів шаблонів для урядової програми США - Personal Identity Verification (PIV).

Сьогодні сервіс по тестуванню шаблонів для з'ясування рівня їх стандартизації MINEX III вводить більш високі вимоги. Новий рівень відповідності PIV Level 2 сходиться до більшої точності обробки шаблонів одного відбитка пальця: тепер з FNMR меншим або рівним 0,02 і FMR меншим або рівним 0,0001. При цьому, розробники MINEX III обіцяють суттєво збільшити набір генеруються даних (до 2 мільйонів одиночних шаблонів) і масштаб тестування (до 3 мільйонів операції за тестовою ідентифікації відбитка пальця). А результати тестування планується надавати з докладними статистичними даними та їх графічної візуалізацією.

На ринку представлені різні запатентовані технології виявлення підроблених відбитків пальців.

Одна з них під найменуванням fake-finger-detection (FFD), відома також як Liveness Detection SDK, є розробкою NexID Biometrics LLC. Компанія має свою власну лабораторію, а також співпрацює з біометричними лабораторіями в Університеті Кларксона і в Університеті Західної Вірджинії.

"Дослідження дали можливість визначити додаткові особливості зображення знятого відбитка пальця, які дуже ефективні в диференціації справжнього (живого) відбитка від підроблених. По суті, ми просто розширили наше портфоліо особливостей помилка пальця, що призвело до високої точності роботи ПО," - розповідає про технології операційний директор NexID, Марк Корнетт.

Версія 2.0 SDK NextID демонструє діапазон точності виявлення шахрайських спроб ідентифікації від 96,5% до 99,5%, в залежності від механізмів створення фальшивого відбитка.

"Продовжуємо досліджувати способи звести кількість помилок до нуля, щоб клієнти, що розгортають біометричну СКУД на основі розпізнавання відбитків пальців, були впевнені в точності роботи датчиків," - говорить Марк.

Крім роботи зі збільшення точності роботи систем і вивчення постійно змінюються методів спуфинга, творці fake-finger-detection приділяють велику увагу вдосконаленню технічних характеристик програмного забезпечення. Намагаючись знизити вимоги до обчислювальної

продуктивності і вільним ресурсам, зберігши при цьому повний набір функцій аналізу і захисту від фальсифікації відбитків, NexID домоглася успішної інтеграції ПО з SoC-системами ("match-on-chip").

Для досягнення цього, було створено нові та інноваційні методи підвищення якості зображення, виділення особливостей і статистичного аналізу, щоб виконувати алгоритми ідентифікації підроблених відбитків в умовах обмежених ресурсів," - стверджує Корнетт.

Завдяки мінімальним системним вимогам, технологія розпізнавання живих відбитків пальців SDK може бути успішно інтегрована в будь-які системи контролю обліку доступу, в тому числі в банкомати, касові термінали і т.п. Fake-finger-detection може застосовуватися як самостійно, так і в якості складової частини багатofакторної системи аутентифікації (наприклад, на додаток до пароллю або розпізнаванню особи). Крім того, існує рішення, спеціально створене NexID для мобільних пристроїв з датчиками відбитків пальців, зокрема смартфонів від Apple і Samsung.

Незалежно від того, чи вважати технології створення фальшивих відбитків пальця для верифікації в біометричних системах простими або складними, доцільними чи ні - не можна не визнавати, що така загроза існує. Ще в 2014 році Chaos Computer Club опублікував інформацію про можливості отримання вихідних даних для створення підробки не тільки контактним способом (якщо людина доторкнувся до будь-якого об'єкта з полірованою поверхнею, будь то склянку або екран смартфона), а й за допомогою звичайного фотоапарата.

За для прикладу ось декілька варіантів замків зі сканером відбитку пальця.

Контроль доступом ZKTeco F18



Рисунок 2.2 – Пристрій доступу по відбитку пальця ZKTeco F18

Пристрій доступу по відбитку пальця ZKTeco F18 - це обладнання нового покоління розроблений з урахуванням потреб приватних користувачів. Через меню дисплея, користувачеві, з правами адміністратора, легко вносити налаштування, додавати або видаляти користувачів з системи. Ним комфортно користуватися не лише в офісі, але і вдома.

Завдяки наявності внутрішньої пам'яті і виходами підключення до ПК або мережі, можливий збір статистики відвідувань користувачами і формування звітів по робочому часу співробітників. Для цих цілей можливе використання програмного забезпечення обліку часу і контролю доступу.

Технічні характеристики терміналу контролю доступу по відбитку пальця F18:

- ємність пам'яті шаблонів відбитка пальця, шт 1500;
- ємність пам'яті карт, шт: 5000 (опція EM);
- ємність журналу відвідуваності, шт: 30000;
- тип біометричного датчика: оптичний, від ZKTeco;
- алгоритм розпізнавання папілярного малюнка: ZK-Finger 10.0;
- способи з'єднання: RS485, TCP / IP, USB-хост;

- функції управління доступом: управління замком, контроль положення дверей, вбудований дзвінок, подача сигналу тривоги, роз'єм для кнопки «ВИХІД»;
- інтерфейси підключення зовнішніх пристроїв: Wiegand-вхід і Wiegand-ВИСНОВОК;
- опції: вбудований зчитувач карт EM-Marine або Mifare;
- джерело живлення: 12 Вольт постійного струму;
- температура експлуатації: 0 - 45 ° C;
- допустима вологість: 20 - 80%.

Собівартість виробу складає 9864 грн.

Пристрій обліку робочого часу по обличчю і відбитку ZKTeco MB10



Рисунок 2.3 – Пристрій обліку робочого часу ZKTeco MB10

Система обліку часу MB10 - це пристрій бюджетної серії з непоганим рівнем функціональності. Пристрій добре підійде для малих і середніх офісів. На відміну від більш дорогих аналогів на Linux, тут використовується спрощена неоновлювана прошивка. Пристрій розпізнає користувача і робить записи в пам'яті про прихід або відхід з роботи.

Завдяки використанню двох варіантів верифікації, вирішується проблема деяких користувачів зі стертим папілярних малюнком на пальцях - вони можуть відзначатися по розпізнаванню особи. Ми рекомендуємо

проводити монтаж даного пристрою на стіну, в прохідному місці таким чином, щоб на термінал не падали прямі сонячні промені або світло яскравої лампи. Це позбавить пристрій від засвічень як камери, так і дактилоскопічного сенсора.

Характеристики бюджетного лічильника відпрацьованого часу ZKTeco MB10:

- пам'ять шаблонів відбитків: 500 шт;
- пам'ять шаблонів осіб: 200 шт;
- пам'ять безконтактних карт: 1000 шт;
- журнал подій відвідуваності: 50000 шт;
- сенсор пальців: ZK-Optical з алгоритмом ZK VX10;
- передача даних по: Ethernet-кручений парі і USB-роз'єму для флешок;
- дисплей: кольоровий, 2.8 дюйма;
- функції робочого часу:., перехід на зимовий і літній час, оповіщення; на екрані, перемикання статусів Прихід / Догляд за розкладом, дев'ятизначний ID користувача, T9;
- функції реле: роз'єм для підключення 12В замку (вимагає окремого живлення), кнопки виходу, геркона (необов'язковий);
- адаптер живлення: 5В 0.8А;
- робоча температура навколишнього середовища: 0-45 С;
- відносна вологість 20-80%;
- розміри терміналу: 167,9 x 151,5 x 30 мм.

Собівартість приладу складає 6791 грн.

Вибір користувача впав на сканер відбитку пальця на Arduino. Його переваги у тому, що його легко зібрати власноруч і у нього мала собівартість.

Arduino і біометричний сканер відбитків пальців



Рисунок 2.4 – Сканер відбитку на Arduino

Оптичні датчики відбитків пальців зазвичай використовуються в системах безпеки. Ці сенсори включають в себе DSP чіп, який обробляє зображення, робить необхідні розрахунки для виявлення відповідності між записаними і поточними даними. Недорогі датчики відбитку пальців дозволяє вмістити до 162 різних відбитків пальців.

Сенсор поставляється з софтом для Windows, що значно полегшує його тестування. З використанням рідної софта можна навіть відобразити фотографію вашого відбитка на моніторі! Ну і саме чудове - є окрема бібліотека для Arduino, з використанням якої можна налаштувати датчик менше ніж за 10 хвилин.

Технічні характеристики датчика відбитків пальців:

- напруга живлення: 3.6 - 6.0 В (постійний струм);
- робоча сила струму: 120 мА;
- максимальна сила струму: 150 мА;
- час обробки зображення відбитка: <1.0 секунди;
- розмір вікна: 14 мм x 18 мм;
- кількість одночасно записуються файлів: 162 файлів;
- рівень безпеки (від 1 до 5);
- інтерфейс (підключення): TTL послідовний;

- швидкість передачі даних (Baud rate): 9600, 19200, 28800, 38400, 57600 (за замовчуванням 57600);
- робочий діапазон температур: від -20 C до +50 C;
- допустимий рівень вологості: 40% - 85% RH;
- габаритні розміри: 56 x 20 x 21.5 мм;
- Вага: 20 грам.

Запис в пам'ять і пошук даних

При використанні датчика відбитку пальців є два основних етапи. Спочатку вам треба записати дані в пам'ять сенсора, тобто привласнити свій унікальний ID кожного відбитка, який ви будете використовувати для порівняння надалі. Після запису даних, ви можете переходити до «пошуку», порівнюючи поточне зображення відбитка з тими, які записані в пам'яті датчика.

Для запису відбитків пальців можна використовувати софт (найлегший і зручний варіант, так як ви бачите знімок, який зроблений) або скетч для Arduino (актуально для тих, у кого не варто Windows).

Запис нових відбитків через програму для Windows

Як вже говорилося вище, найпростіший шлях записи нових даних в пам'ять оптичного датчика відбитків пальців - програма для Windows [3].

3 Опис та обґрунтування обраних проектних рішень та конструкцій

3.1 Опис структурної схеми роботи пристрою

На рисунку 3.1.1 зображено структурно принципову схему роботи пристрою автоматизованої системи контролю доступом до приміщення.

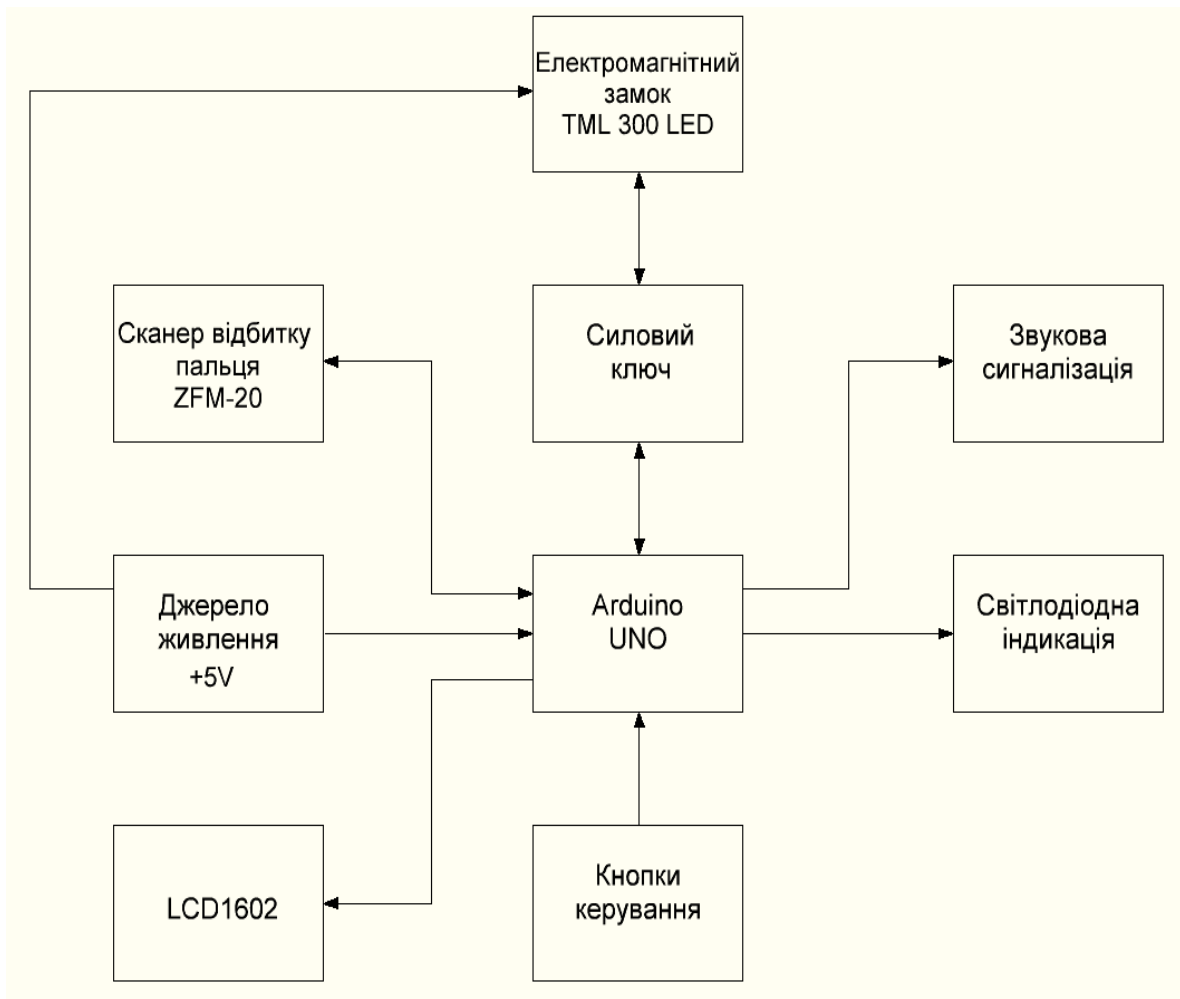


Рисунок 3.1 - Принцип роботи пристрою автоматизованої системи контролю доступом до приміщення. Схема принципова структурна.

Структурна схема має в своєму складі такі елементи як: Arduino UNO, сканер відбитку пальця, електромагнітний замок, силовий ключ, звукова сигналізація, світлодіодна індикація, кнопки керування, LCD дисплей, та джерело живлення.

Живиться система від джерела живлення +5V.

На вхід подається джерело живлення у 5 Вольт. Живлення йде на Arduino UNO, а з нього на сканер відбитку пальців, звукову сигналізацію, світлодіодну індикацію та LCD дисплей.

Сканер відбитку пальців отримує живлення і в свою чергу зчитує інформацію про відбиток людини та передає дані про нього до Arduino UNO.

Кнопки керування слугують для керування та налаштування приладу. За допомогою кнопок можна в різні розділи меню. Додавати та видаляти відбитки пальців.

LCD дисплей виконує важливу роль. Він показує про статус електромагнітного замка відчинено/зачинено. Також він відображає id кожного відбитку пальця користувача.

Світлодіодна індикація слугує для відображення статусу сканера відбитку пальця. Якщо користувач прикладе палець якого немає у пам'яті – то світлодіодна індикація загориться червоним кольором. Це свідчить про те, що електромагнітний замок не спрацює, і не пустить людину. Якщо прикласти палець який є у пам'яті – то світлодіодна індикація загориться зеленим кольором і електромагнітний замок відчиниться.

Звукова сигналізація відображає статус відкриття/закриття електромагнітного замку. Коли користувач прикладає палець и замок відчиняється або зачиняється, то спрацює звукова сигналізація.

Силовий ключ виконаний на польовому транзисторі IRF520 і призначений для вмикання/вимикання потужного навантаження, яке живиться напругою постійного струму. Управляти силовим ключем можна за допомогою Arduino UNO або іншого мікроконтролера, при подачі на вхід ключа високого рівня від 5 В, він відкриється і вмикає навантаження.

Електромагнітний замок отримує сигнал від силового ключа і відчиняється/зачиняється у відповідності сигналу.

Транзистор VT1 керує зумером BZ1. Якщо відбиток співпадає з базою даних, то електромагнітний замок відкривається і зумер сповіщує про це звуковим сигналом тривалістю 100 мс та з частотою 2кГц.

Резистори R3, R4, R5, R9, R11 обмежують струм бази ключів транзисторів.

Катушка L1 є катушкою електромагнітного замка. Arduino UNO призначений для сканування, зберігання, видалення, порівняння відбитків, а також для видачі команд на комутуючий транзистор та LCD індикації, і прийому команд від кнопок керування.

LCD дисплей призначений для відображення станів електромагнітного замка та меню.

Кнопки SB1 та SB2 призначені для входу в меню, зберігання та видалення відбитків.

Робота розробленого пристрою для аналізу автоматизованого контролю доступом до приміщення за допомогою сканера відбитку пальця базується на мікроконтролері ATmega328 який реалізує взаємодію системи з іншими елементами, живиться від +5В.

ATmega328 - це однокристальний мікроконтролер, створений Atmel в сімействі megaAVR. Він позначений на схемі як DD1.

Дисплей LCD1602 на схемі позначений як HL3. Слугує для відображення, видалення та налаштування пристрою.

Сканер відбитків пальців ZFM-20 - призначений для сканування, збереження відбитків пальців. На схемі він позначений як U1.

Характеристики:

- напруга живлення: 3,6 ... 6 В постійного струму;
- струм: 90 ... 110 мА, піковий до 150 мА;
- інтерфейс: UART, використовуються рівні TTL;
- швидкість передачі даних: $9600 * N$ біт / с, де $N = 1 \dots 12$ (за замовчуванням $N = 1$ (9600N біт / с) або 6 (57600N біт / с)) встановлюється користувачем;
- час обробки зображення: <1 сек;
- середній час пошуку: <1 сек (1: 880);
- ємність накопичувача: 120/375/880;
- розмір файлу 256 байт;
- розмір шаблону: 512 байт;

Сканер відбитку пальця зображено на рисунку 3.3.



Рисунок 3.3 – Сканер відбитку пальця ZFM-20

Модуль, для передачі даних, використовує шину UART. Для підключення модуля до arduino можна використовувати як апаратну шину UART (висновки 0 і 1), так і програмно-реалізовану (за допомогою бібліотеки SoftwareSerial, яка входить в стандартний набір IDE Arduino [8].

Електромагнітний замок для Arduino – використовується для створення проектів систем доступу. На схемі він позначений як L1.

При подачі напруги на замок язичок переміщається всередину корпусу, при знятті напруги язичок повертається у вихідне положення. Електромагнітний замок для Arduino зображен на рисунку 3.4.

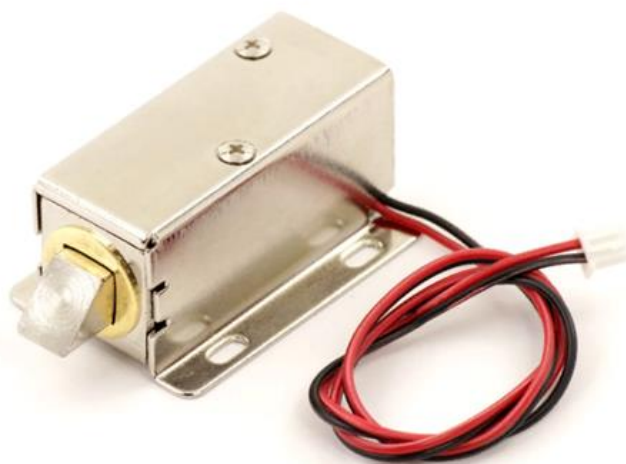


Рисунок 3.4 – Електромагнітний замок для Arduino UNO

Характеристики:

- робоча напруга 12 В;
- максимальний струм до 0.8А;
- максимальний час роботи не повинна перевищувати 10 секунд;
- можливість змінювати напрямки язичка в залежності від напрямку відчинення дверей.

Кнопки позначені на схемі як SB1 та SB2. Трема-модуль кнопка - це тактова кнопка, яка може служити джерелом сигналів (команд) для Ваших проектів. Кнопки використовуються для управління пристроями, подачі команд, здійснення налаштувань, введення даних і т.д [8].

Характеристики:

- робоча напруга: до 5,5 В;
- комутований струм: до 50 мА;
- час брязкоту при натисканні: <3 мкс;
- час брязкоту при відпуску: <5 мс;
- опір притискає резистора: 10 кОм;
- опір S - V: > 100 МОм (в розімкнутому стані);
- опір S - V: <100 мОм (в замкнутому стані);
- робоча температура: -20 ... 70 ° С;
- габарити: 30x30x15.

На рисунку 3.5. зображена кнопка.

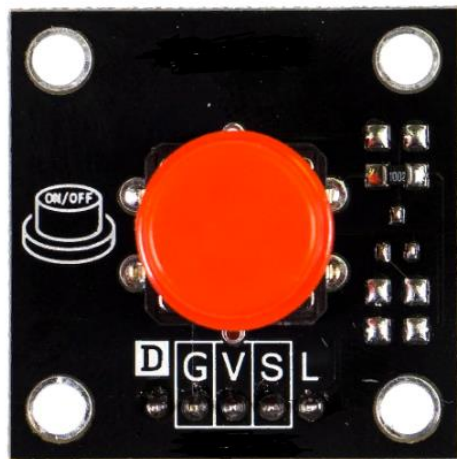


Рисунок 3.5 – Трема-модуль кнопка

Trema-модуль світлодіод - буде корисний для індикації і підсвічування в різних проектах. Виконано на базі яскравого світло діода [8].

Виконаний в лінійці Trema-модулів, завдяки цьому вам не доведеться нічого паяти і збирати на макетної платі. Позначений як HL1 та HL2. В схемі використано два світлодіода. Червоний та зелений. Червоний спрацьовує при відмові відбитка пальця. Зелений спрацьовує при правильному відбитку пальця який є у пам'яті. Світлодіод зображений на рисунку 3.6.

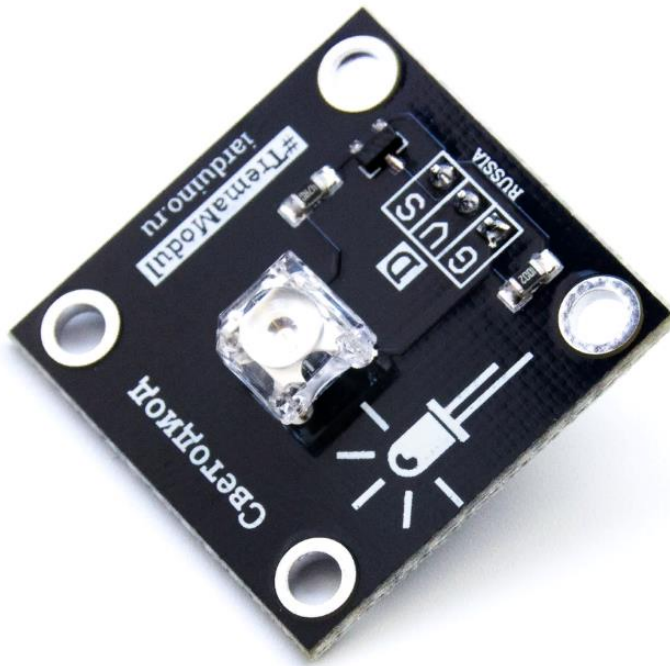


Рисунок 3.6 – Trema-модуль світлодіод для Arduino

Trema-модуль силовий ключ - допоможе керувати навантаженням, наприклад - яскравістю світлодіодним стрічки або відмиканням електронного замка і т.д [8].

Перевага силового ключа перед звичайним електромеханічним у відсутності звукового клацання і можливість використовувати ШІМ.

Характеристика:

- комутує тільки постійна напруга;
- максимальний комутований струм 10А;
- максимальна напруга, що комутується 30V;
- напруга живлення: від 3 В до 5,5 В;
- на базі польового транзистора Ir8113;
- габарити: 31 × 31 мм.

Силовий ключ зображений на рисунку 3.7.



Рисунок 3.7 – Трета-модуль силовий ключ

Трета-модуль Зуммер - дозволяє випромінювати звук певної тональності (частоти). На схемі позначений як BZ1. Виконує роль звукової індикації роботи приладу [8].

Характеристики:

- напруга живлення: 5 В;
- струм: до 30 мА;
- інтенсивність звуку: ≥ 85 дБ;
- опір обмотки: 40 Ом;
- робоча температура: $-20 \dots 70^\circ \text{C}$;
- габарити: 30x30x9.

Приклад звукової індикації наведено на рисунку 3.8.



Рисунок 3.8 – Трета-модуль зуммер

3.2 Опис електричної принципової схеми пристрою

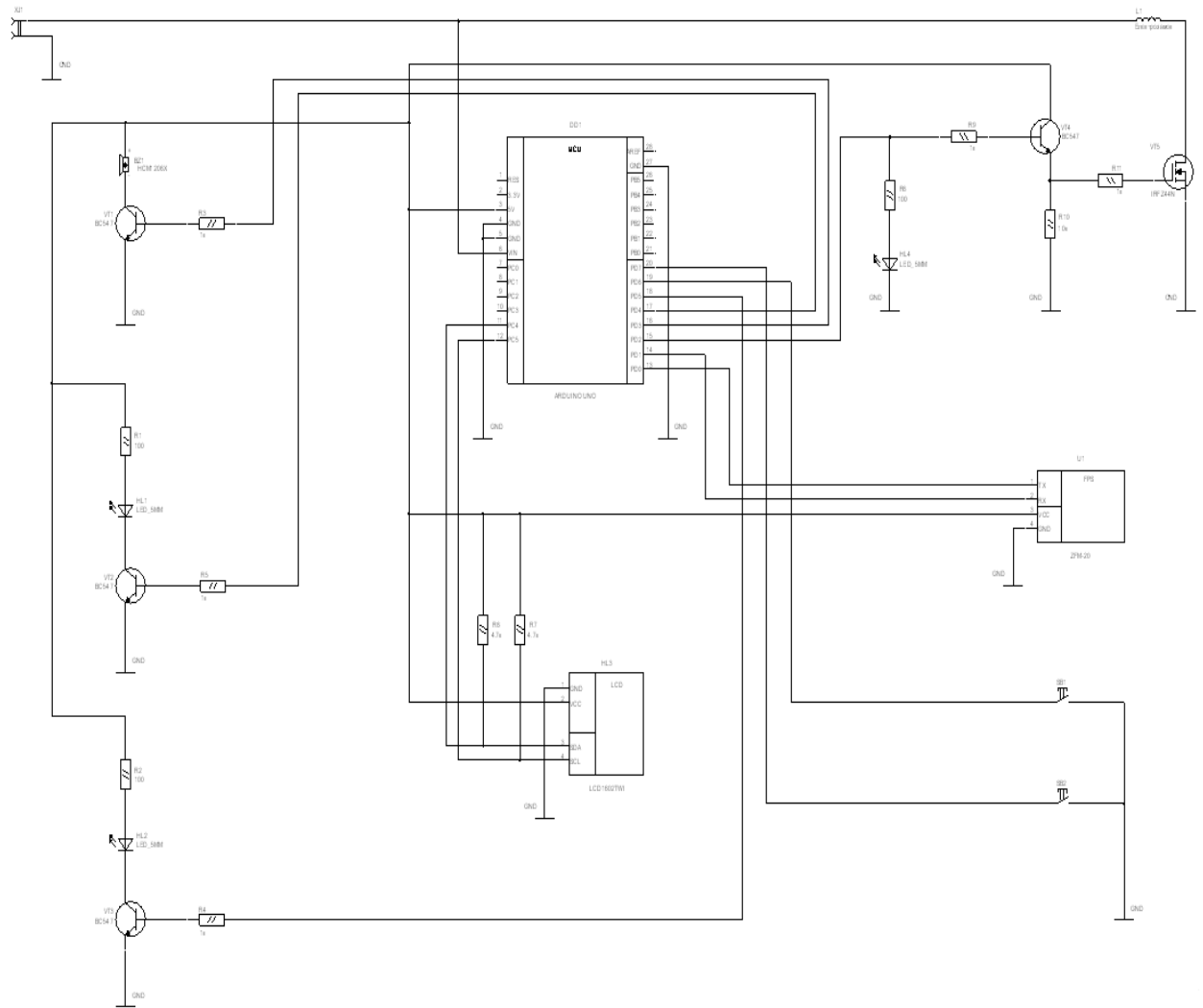


Рисунок 3.2 – Схема електрична принципова

На рисунку 3.2 зображена електрична принципова схема пристрою.

Принцип її роботи полягає у наступному. Сканер відбитку пальця ZFM-20 який позначений на схемі U1 призначений для сканування, збереження відбитків пальців.

Транзистори VT4 та VT5 відкривають електромагнітний замок L1. Світлодіоди HL1 та HL2 показують стан електромагнітного замка (відкритий/закритий). Транзистор VT2 комутує світлодіод HL1 зеленого кольору, який показує, що відбиток є в базі даних. Транзистор VT3 комутує світлодіод HL2 червоного кольору, що означає, що відбиток не співпадає з тим що є у базі даних.

Джерело живлення (12В / 2А) - призначений для живлення устаткування постійним стабілізованою напругою 5В [8].

Характеристики:

- виробник: ROBITON;
- тип: імпульсний;
- напруга живлення джерела: 220В;
- напруга на виході: 12В;
- максимальний струм споживання: 2А;
- роз'єм DC jack 5,5 * 2,5 мм;

Приклад джерела живлення зображено на рисунку 3.9.



Рисунок 3.9 – Джерело живлення 12В

Програма P-CAD 2006 є дуже популярною, що обумовлено, з одного боку, гарною функціональністю програми та, з іншого боку, по асоціації з поширеними тут старими версіями P-CAD 4.5 - 8.7. У 1996 р фірма ACCEL Technologies вперше представила версію широко відомої системи розробки друкованих плат P-CAD на платформі Windows

Оновлений продукт отримав нову назву ACCEL EDA. З цього моменту продукт ACCEL EDA набув широкої популярності серед розробників електронних пристроїв. У вересні 1999 р вийшла остання 15 версія продукту. 17 січня 2000 р відбулося злиття двох провідних розробників систем САПР друкованих плат - фірм Protel International і ACCEL Technologies, які об'єднали свої спільні зусилля під торговою маркою Protel (нині Altium). З березня 2000 р продукт ACCEL EDA змінив свою назву на P-CAD. Продаваний в даний час продукт носить назву P-CAD 2006.

Система P-CAD 2006 виконує повний цикл проектування друкованих плат, а саме:

- графічний введення електричних схем;
- змішане аналого-цифрове моделювання на основі ядра SPICE3;
- упаковку схеми на друковану плату;
- інтерактивне розміщення компонентів;
- інтерактивну і автоматичне трасування провідників;
- контроль помилок в схемі і друкованій платі;
- випуск документації;
- аналіз цілісності сигналів і перехресних спотворень;
- підготовку файлів Gerber і NC Drill для виробництва друкованих плат;
- підготовку бібліотек символів, топологічних посадкових місць і моделей компонентів.

Основні можливості P-CAD 2006:

- зручний інтерфейс, схожий на більшість популярних програм для Windows;
- зберігання проектної інформації в бінарних і текстових файлах;
- зручна довідкова система;
- проект схеми може містити 999 аркушів, проект плати - до 999 шарів (11 з них стандартних);
- число ланцюгів в проекті - до 64000;
- число вентилів в компоненті - до 5000;
- максимальне число висновків у компонента – 10000;
- максимальні розміри листа схеми або креслення друкованої плати 60х60 дюймів;
- підтримка дюймової і метричною систем заходів;
- граничне дозвіл 0.0001 дюйма (0.1 мила) або 0.01 мм (10 мікрон);
- мінімальний кут повороту компонентів на платі - 0.1 град;
- довжина імен компонентів - до 30 символів, максимальний обсяг текстових написів і атрибутів - до 20000 символів;

- механізм перенесення змін друкованої плати на схему і навпаки (Engineering Change Order, ECO);
- бібліотеки компонентів, що містять більше 27000 елементів і сертифіковані за стандартом ISO 9001.

3.3 Вибір Arduino UNO

Для реалізації роботи системи використовується Arduino UNO, структурна схема якого зображена на рисунку 3.10.

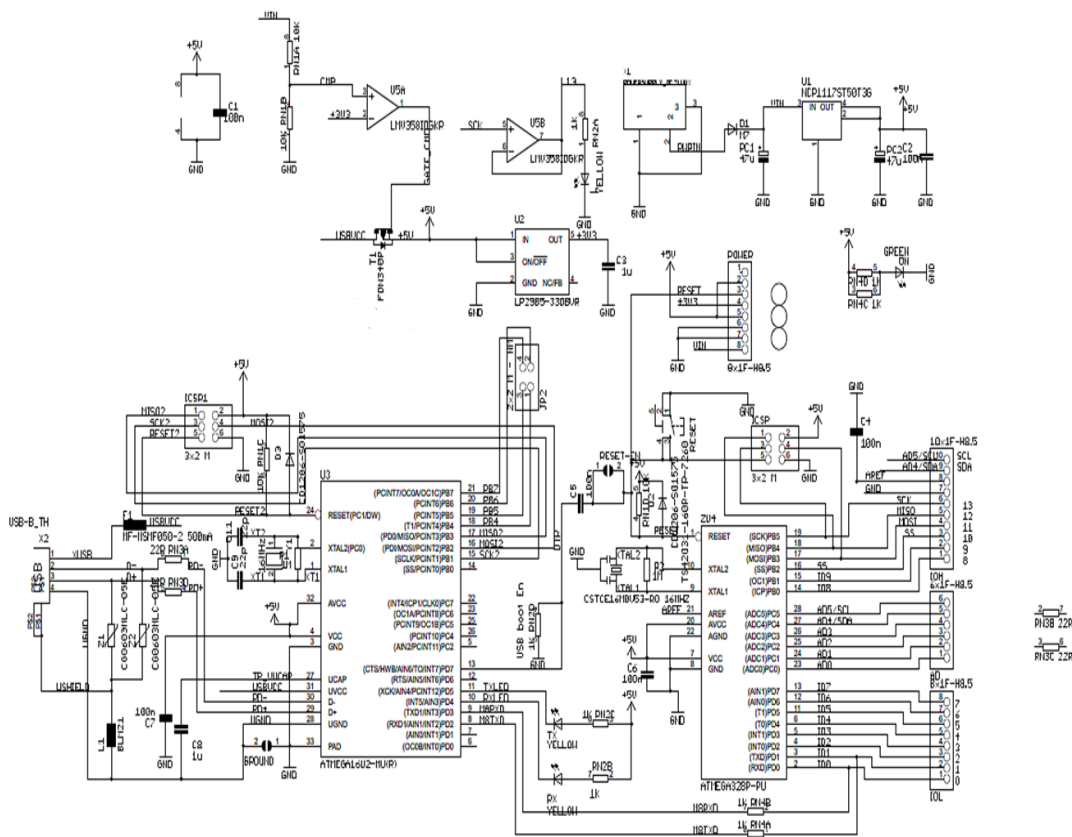


Рисунок 3.10– Структурна схема Arduino UNO

Arduino Uno контролер побудований на ATmega328. Платформа має 14 цифрових входів / виходів (6 з яких можуть використовуватися як виходи ШІМ), 6 аналогових входів, кварцовий генератор 16 МГц, роз'єм USB, силовий роз'єм, роз'єм ICSP і кнопку перезавантаження. Для роботи необхідно підключити платформу до комп'ютера за допомогою кабелю USB, або подати живлення за допомогою адаптера AC / DC або батареї. Приклад зовнішнього вигляду плати Arduino UNO зображено на рисунку 3.11 [5].



Рисунок 3.11 – Зовнішній вигляд Arduino UNO

Arduino UNO має наступні характеристики:

- мікроконтролер ATmega328;
- робоча напруга 5В;
- напруга живлення (рекомендований) 7-12В;
- напруга живлення (граничне) 6-20В;
- цифрові входи / виходи 14 (з них 6 можуть використовуватися в якості ШІМ-виходів);
- аналогові входи 6;
- максимальний струм одного виведення 40 мА;
- максимальний вихідний струм виводу 3.3V 50 мА;
- flash-пам'ять 32 КБ (ATmega328) з яких 0.5 КБ використовуються загрузчиком;
- SRAM 2 КБ (ATmega328);
- EPROM 1 КБ (ATmega328);
- тактова частота 16 МГц.

Arduino Uno може бути живиться від USB або від зовнішнього джерела живлення - тип джерела вибирається автоматично.

В якості зовнішнього джерела живлення (НЕ USB) може використовуватися мережевий AC / DC-адаптер або акумулятор / батарея. Штекер адаптера (діаметр - 2.1мм, центральний контакт - позитивний) необхідно вставити у відповідний роз'єм живлення на платі. У разі живлення

від акумулятора / батареї, її проводу необхідно під'єднати до висновків Gnd і Vin роз'єму POWER [6].

Напруга зовнішнього джерела живлення може бути в межах від 6 до 20 В. Однак, зменшення напруги живлення нижче 7В призводить до зменшення напруги на виводі 5V, що може стати причиною нестабільної роботи пристрою. Використання напруги більше 12В може призводити до перегріву стабілізатора напруги і виходу плати з ладу. З огляду на це, рекомендується використовувати джерело живлення з напругою в діапазоні від 7 до 12В.

З використанням функцій `pinMode ()`, `digitalWrite ()` і `digitalRead ()` кожен з 14 цифрових висновків може працювати в якості входу або виходу. Рівень напруги на висновках обмежений 5В. Максимальний струм, який може віддавати або споживати один висновок, становить 40 мА. Всі висновки пов'язані з внутрішніми підтягуються резисторами (за замовчуванням відключеними) номіналом 20-50 кОм. Крім цього, деякі висновки Arduino можуть виконувати додаткові функції:

- послідовний інтерфейс: висновки 0 (RX) і 1 (TX). Використовуються для отримання (RX) і передачі (TX) даних по послідовному інтерфейсу. Ці висновки з'єднані з відповідними висновками мікросхеми ATmega8U2, яка виконує роль перетворювача USB-UART;
- зовнішні переривання: висновки 2 і 3. Чи можуть служити джерелами переривань, що виникають при фронті, спаді або при низькому рівні сигналу на цих висновках. Для отримання додаткової інформації див. Функцію `attachInterrupt ()`;
- ШІМ: виводи 3, 5, 6, 9, 10 і 11. За допомогою функції `analogWrite ()` можуть виводити 8-бітові аналогові значення в вигляді ШІМ-сигналу;
- інтерфейс SPI: виводи 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). Із застосуванням бібліотеки SPI дані висновки можуть здійснювати зв'язок по інтерфейсу SPI;
- світлодіод: вивод 13. Вбудований світлодіод, приєднаний до виводу 13. При відправці значення HIGH світлодіод включається, при відправці LOW – вимикається;

– в Arduino Uno є 6 аналогових входів (A0 - A5), кожен з яких може уявити аналогову напругу у вигляді 10-бітного числа (1024 різних значення). За замовчуванням, вимір напруги здійснюється щодо діапазону від 0 до 5 В. Проте, верхню межу цього діапазону можна змінити, використовуючи висновок AREF і функцію `analogReference()`.

Крім цього, деякі з аналогових входів мають додаткові функції:

– TWI: вивод A4 або SDA і висновок A5 або SCL. З використанням бібліотеки `Wire` дані висновки можуть здійснювати зв'язок по інтерфейсу TWI.

Крім перерахованих на платі існує ще кілька висновків:

– AREF. Опорна напруга для аналогових входів. Може бути задіяний функцією `analogReference()`;

– Reset. Формування низького рівня (LOW) на цьому висновку призведе до перезавантаження мікроконтролера. Зазвичай цей висновок служить для функціонування кнопки скидання на платах розширення.

В Arduino Uno є відновлювані запобіжники, що захищають USB-порт комп'ютера від коротких замикань і перевантажень. Незважаючи на те, що більшість комп'ютерів мають власний захист, такі запобіжники забезпечують додатковий рівень захисту. Якщо від USB-порту споживається струм більше 500 мА, запобіжник автоматично розірве з'єднання до усунення причин короткого замикання або перевантаження [6].

3.4 Опис датчика AT77C101B

Системи ідентифікації особистості повинні працювати швидко, надійно і мати низьку ціну. Звичайні методи ідентифікації засновані на використанні документів, підписів та інших подібних способів. Ці традиційні підходи не задовольняють сучасним вимогам забезпечення безпеки. Перспективні напрямки майбутнього - біометрія.

Відбиток пальця являє собою конфігурацію виступів, які містять індивідуальні пори, розділені западинами. Це означає, що для отримання

зображення відбитка пальця можуть бути використані такі параметри як світло, тепло чи електрична ємність [4].

У пристрої використано тепловий датчик AT77C101B від Atmel який зображено на рисунку 3.12

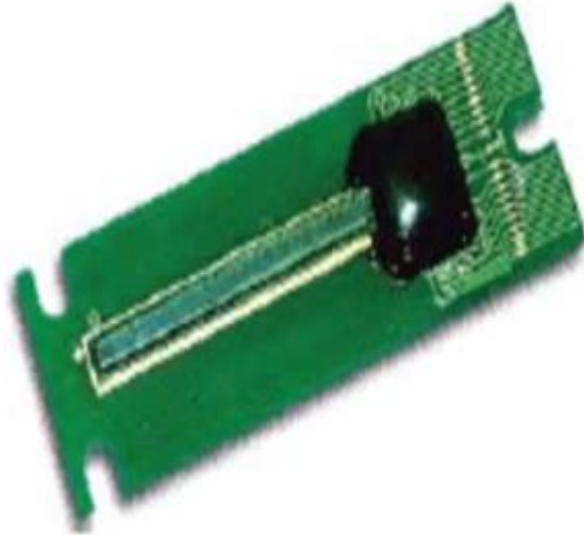


Рисунок 3.12 – Тепловий датчик AT77C101B

Сенсор містить масив з 8 рядків і 280 стовпців, що нараховує в сукупності 2240 термочутливих пікселів.

Кожен піксель має розмір 50x50 мкм, забезпечуючи дозвіл 500 при розмірах чутливої області 0,4x14мм.

Величина цього дозволу відповідає специфікації, яка визначає якість зображення. Частота тактування пікселів програмується і може досягати 2 МГц, забезпечуючи 1780 кадрів в секунду на виході пристрою.

Зображення повноцінного відбитка пальця реконструюється з успішно одержаних кадрів за допомогою програмного забезпечення фірми Atmel.

Функціональна схема датчика AT77C101B зображена на рисунку 3.13

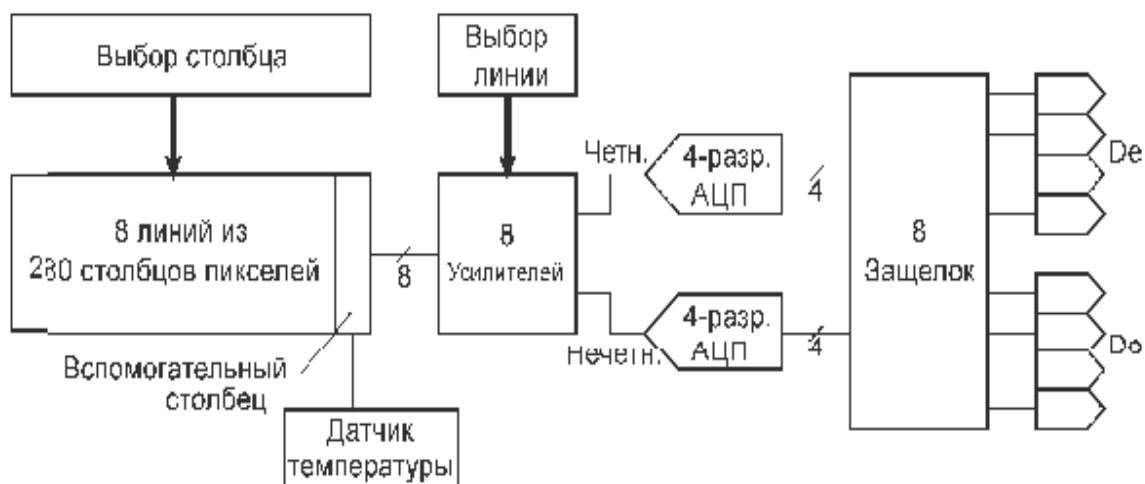


Рисунок 3.13 - Функціональна схема датчика AT77C101B

AT77C101B - представник сімейства інтегральних схем для зчитування відбитку пальців (FingerChip) фірми Atmel, для яких не потрібна додаткова оптика, призма і джерело світла.

Датчик дозволяє програмувати частоту знімання зображення (число зображень в секунду), при цьому вбудований аналогово-цифровий перетворювач формує цифровий сигнал, який може бути сприйнятий інтерфейсом паралельного порту EPP, USB мікроконтролером або безпосередньо мікропроцесором. Таким чином, не потрібні додаткові схеми захоплення зображень або погоджують пристрою для передачі зображення [4].

3.5 Опис мікроконтролера Atmega 328

Мікроконтролер ATmega328 є 8-ми розрядним CMOS мікроконтролером з низьким енергоспоживанням, заснованим на вдосконаленій AVR RISC архітектурі.

На рисунку 3.14 зображено зовнішній вигляд мікроконтролера Atmega 328.



Рисунок 3.14 – Мікроконтролер Atmega 328

Пам'ять:

- 32 kB Flash (пам'ять програм, що має можливість самопрограмування);
- 2 kB ОЗУ;
- 1 kB EEPROM (постійна пам'ять даних).

Характеристика:

- два 8-бітних таймера / лічильника з модулів порівняння;
- 16-бітний таймер / лічильник з модулем порівняння і дільником частоти, а також з режимом запису;
- лічильник реального часу з окремим генератором;
- шість каналів PWM (аналог ЦАП);
- 6-канальний ЦАП з вбудованим датчиком температури;
- програмований послідовний порт USART;
- послідовний інтерфейс SPI;
- інтерфейс I2C;
- програмований сторожовий таймер з окремим внутрішнім генератором;
- внутрішня схема порівняння напруг;

– блок обробки переривань і пробудження при зміні напруги на висновках мікроконтролера [5].

На рисунку 3.15 зображена схема електрична структурна мікроконтролера Atmega 328.

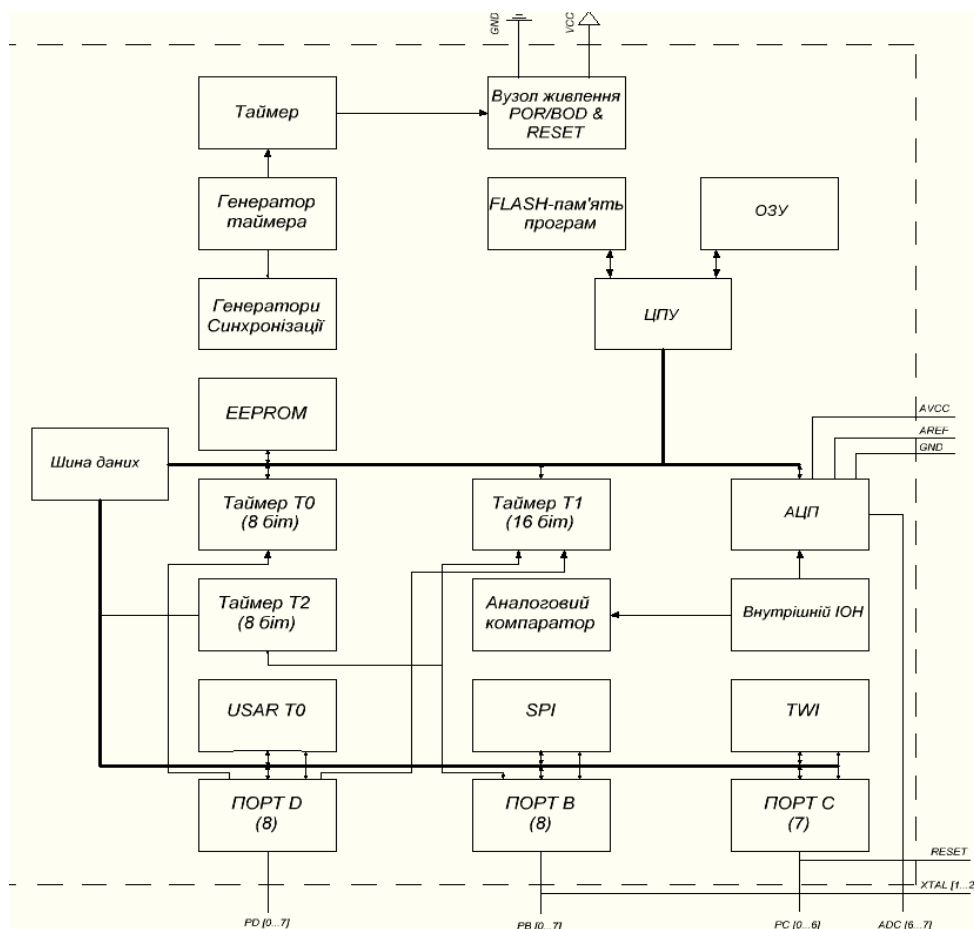


Рисунок 3.15 – Мікроконтролер Atmega328. Схема електрична структурна

Основною функцією ядра CPU є забезпечення правильного виконання програми. Таким чином, центральний процесор повинен мати можливість доступу до пам'яті, виконувати обчислення, керувати периферійними пристроями та обробляти переривання.

Отримана архітектура є більш ефективною, має можливість досягнення пропускнуої здатності в десять разів швидше, ніж звичайні мікроконтролери CISC.

Щоб максимально збільшити продуктивність і паралельність, AVR використовує новий тип архітектури Harvard - з окремими пам'яттю та шинами для програми та даних.

У типовій операції ALU з файлу реєстрації виводяться два операнда, операція виконується, і результат зберігається назад в реєстровому файлі - в одному тактовому циклі.

Шість з 32 реєстрів можуть використовуватися як три 16-бітові непрямі адреси реєстр показчиків для адресації даних про простір - забезпечення ефективних обчислень адрес.

Один з цих адресних показчиків також може використовуватися як адресний показчик для пошуку таблиць у програмі Flash.

Ці додані функціональні реєстри являють собою 16-бітний X-, Y- і Z-реєстр, описаний далі в цьому розділі. ALU підтримує арифметичні та логічні операції між реєстрами або між константами і реєстром.

Операції з одним реєстром також можуть виконуватися в ALU.

Після виконання арифметичної операції реєстр статусу оновлюється для відображення інформації про результат операції.

3.6 Блок-схема алгоритму роботи програми

Алгоритм роботи програми побудований наступним чином. Спочатку йде підключення бібліотеки для роботи з шиною I2C, з програмним UART, зі сканером відбитку пальців та LCD дисплеєм.

Після цього йде оголошення змінних та констант. Тут вказуємо до якого виводу будуть підключені червоний та зелений світлодіоди, зумер, ключ та кнопки.

Далі керування передається на кнопки. За допомогою кнопок переходимо у меню та заносимо відбиток у базу даних. Після цієї процедури відбиток потрібно підтвердити для його повного збереження в базі даних. В базі даних відбитку буде присвоєне власне id.

Якщо прикласти палець відбиток якого немає у базі даних - то спрацює червоний світлодіод і доступ буде заборонений. Якщо ж прикласти палець відбиток якого є у базі даних - то результат буде виведено на дисплей і спрацює зелений світлодіод та звукова індикація, і відповідно відчиниться електромагнітний замок.



Рисунок 3.16 – Блок-схема алгоритму роботи програми

3.7 Опис дисплея LCD 1602

Символьний дисплей LCD1602 I2C з синім підсвічуванням - рідкокристалічний дисплей (Liquid Crystal Display) екран якого здатний відобразити одночасно до 32 символів (16 стовпців, 02 рядки).

На схемі позначений як HL3 та оснащений платою конвертером для перетворення паралельного 8-бітного інтерфейсу дисплея в шину I2C по якій він і підключається до Arduino за адресою 0x3F або 0x27.

Характеристики:

- тип виведеної інформації: символний;
- мова в ПЗУ дисплея: латиниця, японський;
- можливість завантаження власних символів є;
- формат виведеної інформації: 16 × 02 символів;
- тип дисплея: LCD;
- технологія дисплея: STN;
- кут огляду: 180 °;
- тип підсвічування: LED;
- колір підсвічування: синій;
- колір символів: білий;
- контролер: HD44780;
- інтерфейс: I2C;
- адреса на шині I2C: 0x3F або 0x27 (залежить від конвертера);
- напруга живлення 5 В;
- робоча температура: -20 ... +70 ° С;
- температура зберігання -30 ... +80 ° С;
- габарити: 80x36 мм.

Дисплей підключається до апаратної шині I2C Arduino UNO. Найпростіший варіант підключення дисплея до Arduino UNO - по 4-провідного шлейфу через Trema Shield, так як на ньому є підписана колодка шини I2C (рисунок ліворуч). Якщо підключати дисплей безпосередньо, то виводи GND і VCC дисплея підключаються до напруги 5 В, а виводи SDA і SCL до апаратної шині I2C (у Arduino UNO вивод SDA = A4, а вивод SCL = A5).

Зображення електричної функціональної схеми LCD 1602 зображено на рисунку 3.17.

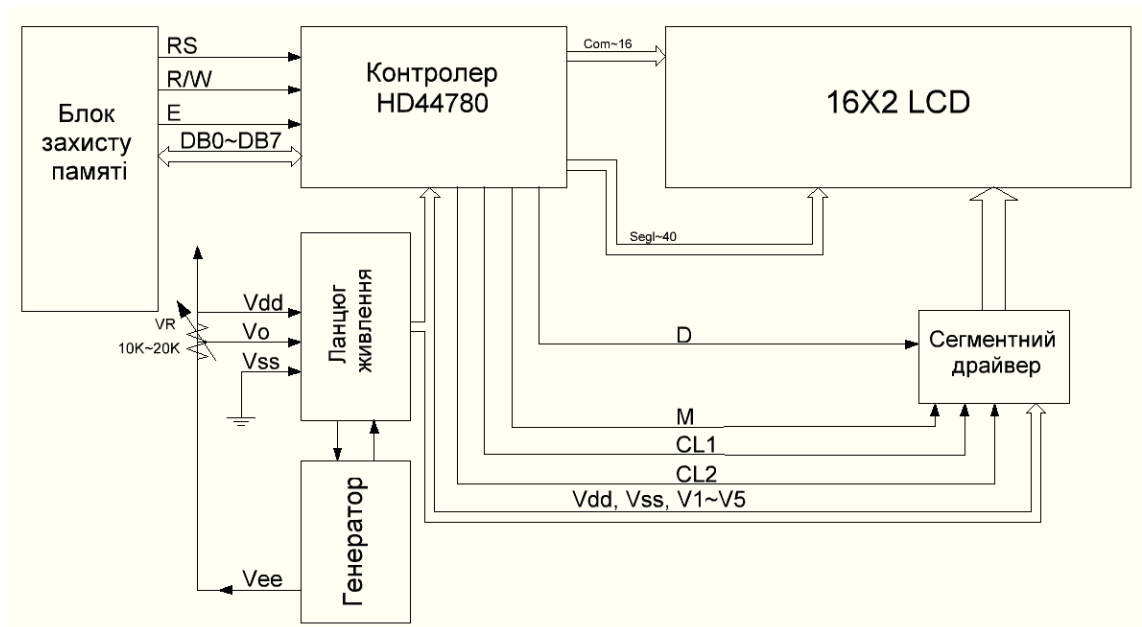


Рисунок 3.17 – LCD 1602. Схема електрична функціональна

Символьний дисплей побудований на базі ЖК дисплея типу STN (Super Twisted Nematic) під керуванням контролера HD44780 і має синхронний паралельний 8-бітний інтерфейс підключений до конвертера для перетворення паралельного інтерфейсу дисплея в шину I2C по якій він і підключається до Arduino UNO. Наявність конвертера полегшує підключення дисплея до Arduino, тому що шина I2C використовує всього 2 виведення для передачі даних і 2 виведення харчування. Дисплей обладнений світлодіодним підсвічуванням синього кольору. Дисплей здатний одночасно відображати до 32 символів від чого і пішла назва дисплея: LCD1602.

Контролер HD44780 має ПЗУ в якій зберігаються цифри, символи латиниці і деякі ієрогліфи японської мови, для їх відображення на дисплеї. Відсутні символи, в т.ч. і символи кирилиці, можна завантажувати в пам'ять ОЗУ контролера, для виведення на дисплей написів на Російському мовою або нестандартних символів наприклад «смайликів» [8].

4 Розрахунки та матеріали, що підтверджують працездатність пристрою

4.1 Розрахунок однофазного випрямляча малої потужності

Таблиця 4.1 – Вихідні дані для розрахунку випрямляча

Показник	Значення
U_d , В	12
I_d , А	0,2
K_n , %	5
U_m , В	220
f_m , Гц	50

Вихідними даними для розрахунку є:

- 1) U_d , В – середнє значення випрямленої напруги за номінального навантаження;
- 2) I_d , А – середнє значення випрямленого струму (струму навантаження);
- 3) K_n , % - коефіцієнт пульсації випрямленої напруги;
- 4) U_m , В – напруга мережі живлення;
- 5) f_m , Гц – частота мережі живлення.

Визначимо орієнтовні значення параметрів вентилів та габаритну потужність трансформатора [14].

Для цього необхідно задати значення коефіцієнтів В, D, F. Для мостової схеми їх вибирають в інтервалах $B=0,95-1,01$; $D=2,1-2,2$; $F=6,8-7,2$.

Нехай $B=1,0$; $D=2,1$; $F=7,1$. (4.1)

Тоді амплітуда зворотної напруги на вентилі становитиме:

$$U_{вм} = 1,5 \cdot U_d = 1,5 \cdot 12 = 18 \text{ (В)} \quad (4.2)$$

Середнє та амплітудне значення струму через вентиль відповідно:

$$I_a = \frac{I_d}{2}; \quad (4.3)$$

$$I_{am} = I_d \cdot 0,5F. \quad (4.4)$$

$$\text{Отже,} \quad I_a = \frac{0,2}{2} = 0,1 \text{ (A)}$$

$$I_{am} = 0,2 \cdot 0,5 \cdot 7,1 = 0,71 \text{ (A)}$$

Габаритну потужність трансформатора визначимо:

$$S_T = U_d I_d \cdot 0,707BD; \quad (4.5)$$

$$S_T = 12 \cdot 0,2 \cdot 0,707 \cdot 1 \cdot 2,1 = 3,56 \text{ (ВА)}.$$

За визначеним значенням габаритної потужності знаходимо максимальне значення індукції B_m для сталі Э360, забезпечуючи виконання умови

$$S_T > 3,56 \text{ ВА:}$$

$$B_m = 1,2 \text{ Тл для } S_T = 10 \text{ ВА.}$$

Вибираємо тип вентилів за табл. 11.7.3 [1]. При цьому необхідно забезпечити виконання умов:

$$U_{\text{вм.мах}} > U_{\text{вм}}; \quad (4.6)$$

$$I_{amax} > I_a; \quad (4.7)$$

$$I_{am \text{ max}} = \pi \cdot I_{amax} > I_{am}. \quad (4.8)$$

В якості вентилів вибираємо кремнієві діоди типу КД 202Б, що мають наступні параметри:

$$U_{\text{вм.мах}} = 50 \text{ В} > 18 \text{ В,}$$

$$I_{amax} = 1 \text{ А} > 0,1 \text{ А;}$$

$$I_{am \text{ max}} = \pi \times 1 = 3,14 \text{ А} > 0,71 \text{ А;}$$

$$U_{\text{пр}} = 1 \text{ (В)}.$$

Знаходимо опір діода у провідному стані:

$$r_{\text{пр}} = \frac{U_{\text{пр}}}{I_{a \text{ max}}}; \quad (4.9)$$

$$r_{\text{пр}} = \frac{1}{1} = 1 \text{ (Ом)}.$$

Знайдемо величину активного опору обмоток трансформатора:

$$r_T = \frac{k_r U_d}{I_d f_m B_m} \sqrt{\frac{S f_m B_m}{U_d I_d}}, \quad (4.10)$$

де k_r – коефіцієнт, що залежить від схеми випрямлення: для мостової схеми $k_r = 3,5$;

B_m – амплітуда магнітної індукції в магнітопроводі трансформатора, Тл;

S – число стержнів трансформатора, що на них розміщено обмотки:

Для броньового трансформатора з Ш-подібними пластинами магнітопроводу $S=1$.

$$r_T = \frac{3,5 \cdot 12}{0,2 \cdot 50 \cdot 1,2} \sqrt[4]{\frac{1 \cdot 50 \cdot 1,2}{12 \cdot 0,2}} = 7,83 \text{ (Ом)}.$$

Знаходимо індуктивність розсіювання обмоток трансформатора:

$$L_S = k_L S \frac{U_d}{I_d \cdot f_M \cdot B_m} \frac{1}{\sqrt[4]{\frac{S f_M \cdot B_m}{U_d \cdot I_d}}}, \quad (4.11)$$

де k_L – коефіцієнт, що залежить від схеми випрямлення: для мостової схеми $k_L = 5 \cdot 10^{-3}$.

$$L_S = 5 \cdot 10^{-3} \cdot 1 \cdot \frac{12}{0,2 \cdot 50 \cdot 1,2} \frac{1}{\sqrt[4]{\frac{1 \cdot 50 \cdot 1,2}{12 \cdot 0,2}}} = 2,23 \cdot 10^{-3} \text{ (Гн)}.$$

Визначаємо кут φ , що характеризує співвідношення між індуктивним і активним опором випрямляча:

$$\varphi = \arctg \frac{2\pi f_M L_S}{r}, \quad (4.12)$$

де r – активний опір випрямляча.

У загальному випадку

$$r = r_T + n_q r_{пр}, \quad (4.13)$$

де n_q – кількість послідовно увімкнених і одночасно працюючих вентилів, для мостової схеми $n_q = 2$.

$$r = 7,83 + 2 \cdot 1 = 9,83 \text{ (Ом)};$$

$$\varphi = \arctg \frac{2 \cdot 3,14 \cdot 50 \cdot 2,23 \cdot 10^{-3}}{9,83} = 4^\circ.$$

Знаходимо величину основного розрахункового коефіцієнта:

$$A_0 = \frac{I_d r}{m U_d}, \quad (4.14)$$

де m – число фаз випрямляча (для мостової схеми $m=2$).

$$A_0 = \frac{0,2 \cdot 9,83}{2 \cdot 12} = 0,082.$$

За знайденими значеннями A_0 і кута φ за графіками, наведеними на рис. 11.7.2-11.7.5. знаходимо величини допоміжних коефіцієнтів

$$B=1,04; D=2,13; F=5,65; H=19,90 \cdot 10^3.$$

Знаючи величини коефіцієнтів B , D , F і H , можна знайти уточнені параметри трансформатора і вентиля, за якими перевіримо правильність їх вибору.

Діюче значення напруги вторинної обмотки трансформатора становить:

$$U_2 = B U_d; \quad (4.15)$$

$$U_2 = 1,04 \cdot 12 = 12,48 \text{ (В)}.$$

Діюче значення струму вторинної обмотки трансформатора:

$$I_2 = 0,707 D I_d; \quad (4.16)$$

$$I_2 = 0,707 \cdot 2,13 \cdot 0,2 = 0,3 \text{ (А)}.$$

Повна потужність вторинної обмотки трансформатора:

$$S_2 = 0,707 B D I_d U_d; \quad (4.17)$$

$$S_2 = 0,707 \cdot 1,04 \cdot 2,13 \cdot 0,2 \cdot 12 = 3,76 \text{ (ВА)}.$$

Діюче значення струму первинної обмотки трансформатора:

$$I_1 = I_2 n, \quad (4.18)$$

де $n = \frac{U_2}{U_1}$ – коефіцієнт трансформації трансформатора ($U_1 = U_m$).

$$n = \frac{U_2}{U_1} = \frac{12,48}{220} = 0,056;$$

$$I_1 = 0,3 \cdot 0,056 = 0,0168 \text{ (А)}.$$

Повна потужність первинної обмотки трансформатора:

$$S_1 = 0,707 \cdot B D I_d U_d = U_1 I_1, \quad (4.20)$$

$$S_1 = 220 \cdot 0,0168 = 3,7 \text{ (ВА)}.$$

Уточнимо повну (габаритну) потужність трансформатора:

$$S_T = \frac{S_1 + S_2}{2}; \quad (4.21)$$

$$S_T = \frac{3,7 + 3,76}{2} = 3,73 < 10(\text{ВА}).$$

Уточнимо значення параметрів діода:

$$U_{\text{BM}} = 1,41BU_d; \quad (4.22)$$

$$U_{\text{BM}} = 1,41 \cdot 1,04 \cdot 12 = 17,59 < 50 (\text{В});$$

$$I_a = \frac{I_d}{2}; \quad (4.23)$$

$$I_a = \frac{0,2}{2} = 0,1 (\text{А}) < 1 (\text{А});$$

$$I_{\text{ам}} = 0,5FI_d; \quad (4.24)$$

$$I_{\text{ам}} = 0,5 \cdot 5,65 \cdot 0,2 = 0,56 \text{ А} < 3,14(\text{А}).$$

Отже тип діода вибрано правильно.

Знаходимо величину ємності конденсатора фільтра:

$$C > \frac{100H}{rk_n f_M}; \quad (4.25)$$

$$C = \frac{100 \cdot 19,90 \cdot 10^3}{9,83 \cdot 5 \cdot 50} = 809,7 (\text{мкФ}).$$

З табл. 11.3.5 [14] вибираємо конденсатор типу К50-35 з параметрами ємністю 1000 мкФ на напругу $25 \text{ В} > \sqrt{2}U_2 = 1,41 \cdot 12,4 = 17,59 \text{ В}$.

Будуємо зовнішню (навантажувальну) характеристику випрямляча $U_d = f(I_d)$. За допомогою цієї характеристики можна визначити відхилення випрямленої напруги U_d від заданого значення при різних величинах струму навантаження I_d , утому числі напругу холостого ходу $U_{d \text{ х.х.}}$, струм короткого замикання $I_{d \text{ к.з.}}$ та величину внутрішнього опору випрямляча r_0 .

Для розрахунку зовнішньої характеристики будемо задавати значення та знаходити відповідні їм значення допоміжного коефіцієнта

$$\gamma_0 = \frac{I_d r}{mU_2}; \quad (4.26)$$

$$\gamma_0 = \frac{I_d \cdot 9,83}{2 \cdot 12,48} = I_d \cdot 0,39$$

За графіком (рис. 11.7.6 [14]) знаходимо відповідні значення величини $\sqrt{2}\cos\theta$ (4-5значень) залежно від γ_0 та φ , де θ – кут відтинання. Тоді відповідні їм значення вихідної напруги випрямляча можна розрахувати за формулою:

$$U_d = U_2 \sqrt{2} \cos\theta; \quad (4.27)$$

$$U_d = 12,48 \cdot \sqrt{2} \cos\theta$$

Таблиця 4.2 – Результати розрахунку навантажувальної характеристики випрямляча за $U_d = 12 \text{ В}$, а $I_d = 0,2 \text{ А}$

$I_d, \text{А}$	γ_0 для $\varphi = 10^\circ$	$\sqrt{2}\cos\theta$	$U_d, \text{В}$
0	0	1,39	17,34
0,05	0,026	1,18	14,72
0,1	0,052	1,06	13,22
0,2	0,078	0,96	11,98

З цим можна зробити висновок, що параметри розрахованого випрямляча відповідають завданню, бо при $I_d = 0,2 \text{ А}$ маємо $U_d = 11,98 \text{ В}$, що відрізняється від заданого значення $U_d = 12 \text{ В}$ на 0,2 %. Це відповідає допустимій точності при інженерних розрахунках ($\pm 5 \%$)[14].

Знаходимо значення напруги холостого ходу випрямляча:

$$U_{d \text{ х.х.}} = U_{2m} = U_2 \sqrt{2}; \quad (4.28)$$

$$U_{d \text{ х.х.}} = 12,48 \cdot 1,41 = 17,59 \text{ (В)}.$$

Величина струму короткого замикання становить:

$$I_{d \text{ к.з.}} = \frac{\sqrt{2}U_2}{r}; \quad (4.29)$$

$$I_{d \text{ к.з.}} = \frac{17,59}{9,83} = 1,79 \text{ (А)}.$$

Величина внутрішнього опору випрямляча складає:

$$r_0 = \frac{U_{d \text{ х.х.}} - U_d}{I_d}; \quad (4.30)$$

$$r_0 = \frac{17,59 - 12}{0,2} = 27,9 \text{ (Ом)}.$$

Знайдемо величину к.к.д. випрямляча:

$$\eta = \frac{U_d I_d}{U_d I_d + P_T + P_B}, \quad (4.31)$$

де P_T – втрати потужності в трансформаторі з к.к.д. $\eta_T = 0,86$;

P_B – втрати потужності у одночасно працюючих діодах: $n_q = 2$.

Втрати потужності у трансформаторі

$$P_T = S_T(1 - \eta_T); \quad (4.32)$$

$$P_T = 3,73 \cdot (1 - 0,86) = 0,52 \text{ (ВА)}.$$

$$P_B = I_a U_{np} n_q; \quad (4.33)$$

$$P_B = 0,1 \cdot 1 \cdot 2 = 0,2 \text{ (ВА)}.$$

Тоді

$$\eta = \frac{12 \cdot 0,2}{12 \cdot 0,2 + 0,52 + 0,2} = 0,72.$$

Електричну принципову схему розрахованого випрямляча наведено на рис. 4.1

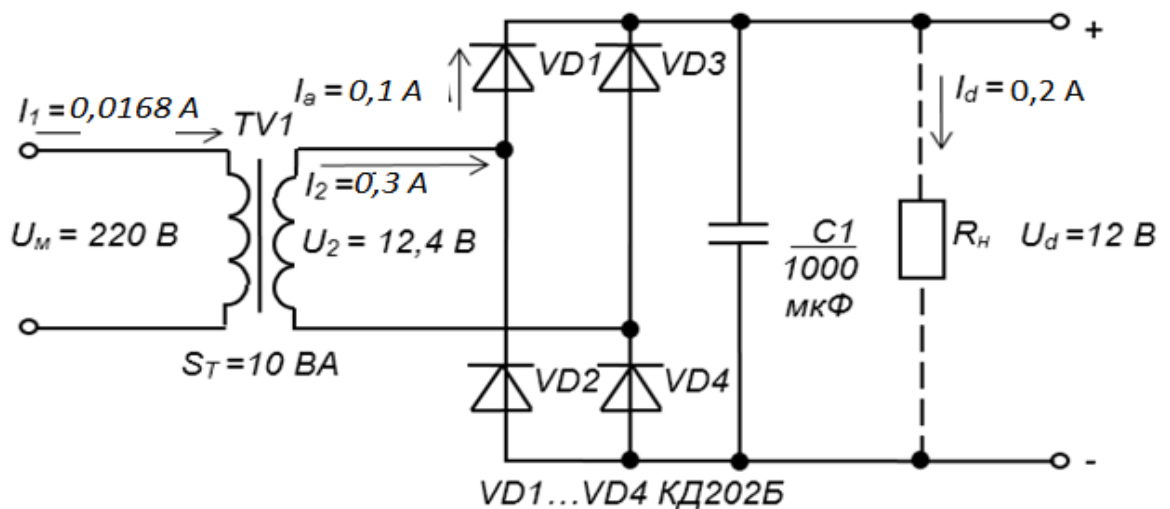


Рисунок 4.1 – Однофазний мостовий випрямляч з ємнісним фільтром.
Схема електрична принципова

4.2 Розрахунок надійності спроектованої системи

Однією з головних властивостей пристрою є надійність. Надійність задовольняє пристрій виконувати свої функції в певних умовах експлуатації зі збереженням безвідмовності в роботі та ремонтпридатності.

Кожен з використаних елементів має свою характеристику і властивості, наприклад крихкість матеріалів, ймовірність перегріву,

швидкість зношення та термін придатності. Принцип визначення показників надійності пристрою за показниками надійності комплектуючих елементів лежить в основі розрахунку надійності, для цього потрібно знати інтенсивність відмов електричних елементів пристрою.

Таблиця 4.2 – Інтенсивність відмов елементів спроектованої системи

Найменування різного типу елементів	Число однотипних елементів, шт.	Інтенсивність відмов елементів, $\lambda_i \cdot 10^{-6}$ /год	Результуюча інтенсивність, $\lambda_i \cdot 10^{-6}$ /год
Світлодіоди	4	0,45	1,8
Кварцовий резонатор	1	0,01	0,01
Перемикач	1	0,25	0,25
Мікроконтролер	1	0,8	0,8
Мікросхеми	2	0,8	1,6
Резистори	7	0,03	0,21
Кнопки	3	0,06	0,18
Конденсатори	6	0,1	0,6
Батарея	1	0,1	0,1
Стабілізатор напруги	1	0,15	0,15

Отже,

$$\lambda_{pi} = \lambda_{p1} + \lambda_{p2} + \lambda_{p3} + \lambda_{p4} + \lambda_{p5} + \lambda_{p6} + \lambda_{p7} + \lambda_{p8} + \lambda_{p9} + \lambda_{p10} =$$

$$= 1,8 + 0,01 + 0,25 + 0,8 + 1,6 + 0,21 + 0,18 + 0,6 + 0,1 + 0,15 = 5,7 \cdot 10^{-6}$$

/год

Вихід з ладу хоча б одного елемента призводить до відмови всієї системи. Застосовуючи відому з теорії ймовірностей теорему про множення ймовірностей можна стверджувати що:

$$P_{\text{сист}}(t) = P_1(t) * P_2(t) * \dots * P_n(t) \quad (4.1)$$

де $P_{\text{сист}}(t)$ – ймовірність безвідмовної роботи системи, а $P_1(t), \dots, P_n(t)$ ймовірності безвідмовної роботи окремих елементів.

Формула для визначення нормального періоду роботи системи:

$$P_1(t)=e^{-\lambda_1*t}, P_2(t)=e^{-\lambda_2*t}, P_n(t)=e^{-\lambda_n*t} \quad (4.2)$$

Таким чином,

$$P_{\text{сист}}(t)=e^{-(\lambda_1+\lambda_2+\dots+\lambda_n)*t}=e^{-\lambda_{\text{сист}}*t} \quad (4.3)$$

$$\text{де} \quad \lambda_{\text{сист}} = \lambda_1 + \lambda_2 + \dots + \lambda_n$$

Візьмемо безвідмовний строк роботи системи: $t=1000$ год.

Отже, відповідно до формули, отримаємо:

$$t_1=1000 \quad P_{\text{сист}}(t) = e^{-5,7*0,001} = 0,99$$

$$t_2=2000 \quad P_{\text{сист}}(t) = e^{-5,7*0,002} = 0,98$$

$$t_3=3000 \quad P_{\text{сист}}(t) = e^{-5,7*0,003} = 0,98$$

$$t_4=4000 \quad P_{\text{сист}}(t) = e^{-5,7*0,004} = 0,97$$

Чим ближче значення до 1 тим більша ймовірність безвідмовної роботи пристрою.

В розрахунках до розробленого пристрою це значення наближене до 1, а саме в мажах від 0,97 до 0,99, що є показником надійної безвідмовної роботи системи.

Середній час безвідмовної роботи системи:

$$T_{\text{сист}}=1/\lambda_{\text{сист}}, \quad (4.4)$$

$$\text{тоді} \quad T_{\text{сист}} = 1/5,7*10^{-6} = 175438,59 \text{ год}$$

Будуємо графік залежності безвідмовної роботи системи від часу:

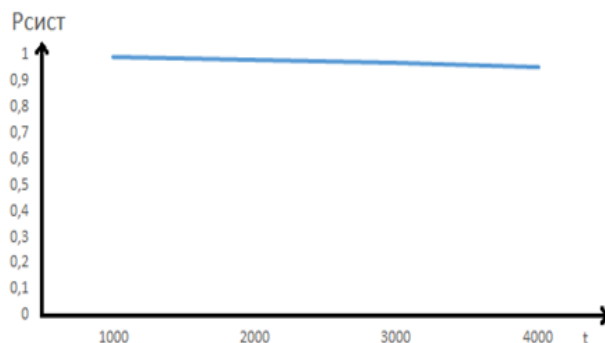


Рисунок 4.2 – Графік залежності безвідмовної роботи пристрою

Проаналізувавши отримані результати можемо зробити висновки, що система працездатна і безвідмовна, робота спроектованого пристрою гарантована на протязі досить тривалого часу.

В розробленій системі використовуються лише надійні та якісні елементи та модулі.

5 Опис організації робіт, пов'язаних із застосуванням виробу

При експлуатації даного пристрою шанс на помилку, збій чи поломку дуже малий. Пристрій є простим аналогом більш складних систем які потребують більших навичок роботи з ними, але при роботі з даним пристроєм також потрібні певні знання.

При роботі з системою контролю доступом зі сканером відбитку пальця потрібно: розуміти як працює пристрій, щоб в малоімовірній поломці чи збої в роботі, можна було усунути проблему, розуміти принцип роботи мікроконтролера, знати його плюси та мінуси, та розуміти програмний код який вшитий в пристрій.

Проблеми з якими може зіткнутися пристрій не так і багато, це може бути збої в безпосередньому у сканері відбитку пальця або його вихід з ладу.

Для реалізації проекту необхідно встановити бібліотеки:

- бібліотека `LiquidCrystal_I2C_V112` (для роботи з дисплеями LCD1602 по шині I2C);
- бібліотека `Adafruit_Fingerprint` (для роботи з модулем відбитків пальців);
- бібліотеки `Wire` і `SoftwareSerial` входять в стандартний набір IDE Arduino.

Сканер відбитків пальців, в цьому пристрої, виконує три дії:

- реєстрація відбитків пальців. При реєстрації відбитків пальців, користувач повинен покласти палець до оптичного датчика двічі. Модуль обробить обидва зображення і на їх основі створить шаблон, який буде збережений в бібліотеці відбитків пальців.;
- порівняння відбитків пальців. При порівнянні відбитків пальців, користувач прикладає палець до оптичного датчика, модуль згенерує шаблон пальця і порівняє його зі збереженими шаблонами в бібліотеці відбитків пальців;
- видалення відбитків пальців. При видаленні відбитка пальця, модуль видаляє шаблон відбитка з бібліотеки.

При старті (в коді `setup`), скетч виконує 3 завдання:

- установка режимів роботи висновків;
- ініціалізація і висновок інформації на дисплей;
- ініціалізація (і перевірка) сенсора відбитків пальців.

Циклічно (в коді loop), скетч виконує 4 завдання:

- опитування стану кнопок - виконується функцією `Func_buttons_control`;
- спілкування з сенсором відбитків пальців - виконується функцією `Func_sensor_communication ()`;
- висновок даних на LCD дисплей - виконується функцією `Func_display_show ()`;
- управління модулями: зумером, світлодіодами, силовим ключем - виконується в коді loop.

Тип задач виконуваних функціями `Func_buttons_control ()`, `Func_sensor_communication ()` і `Func_display_show ()`, визначається значенням змінної `VAR_mode_MENU`, яка зберігає номер режиму меню:

- 0-9 поза меню;
- 10-19 Menu> Set state;
- 20-29 Menu> Show ID;
- 30-39 Menu> New ID;
- 40-49 Menu> Del ID;
- 99 Menu> Exit.

Управління модулями здійснюється при встановленому прапорі `FLG_state_WORK` - вказує, що пристрій знаходиться в стані ENABLE.

Відкриття замку (ввімкнення зумера, перемикання світлодіодів) здійснюється за встановленим прапору `FLG_mode_ACCESS`, який, в решті-коду loop, скидається через 5 секунд після установки.

Дисплей підключається до шини I2C, всі інші модулі підключаються до цифрового виводу Arduino.

Після повної сборки підключаємо живлення. При старті Arduino попросить сканер відбитку пальця. Система дає можливість записувати, видаляти і дивитися відбитки пальців. Для кожної дії існує свій розділ меню.

Щоб потрапити у меню потрібно одночасно натиснути кнопки А і В. Для переходу по розділах меню потрібно натиснути кнопки А або В. Для виходу з меню необхідно вибрати розділ «Exit» або тримати натиснутими обидві кнопки А і В більше двох секунд. При одночасному натисканні кнопок А і В ми потрапимо в меню.

В меню можна побачити такі розділи як: “New ID”, “Show ID”, “Del ID” та “Exit”.

Для того, щоб додати новий відбиток пальця необхідно перейти в меню, потім в розділ “New ID”. Скетч пропонує вибрати номер ID майбутнього відбитку пальця. Скетч пропонує прикласти палець до сканеру відбитку пальця.

Управління:

- вхід в меню - короткочасне натискання на обидві кнопки;
- перехід до наступного пункту меню - натискання на кнопку «А»;
- перехід до попереднього пункту меню - натискання на кнопку «В»;
- вибір пункту меню - короткочасне натискання на обидві кнопки;
- вихід з меню - утримання двох кнопок «А» і «В» довше 2 с, або вибір пункту меню «Exit»;
- відкриття замку зсередини - натискання на кнопку «А» або «В» без входу в меню;
- блокування замка - вибір параметра «DISABLE» в пункті меню «State»;
- розблокування замка - вибір параметра «ENABLE» в пункті меню «State»;

перегляд зайнятих ID шаблонами відбитків пальців - вибір пункту меню «Show ID»;

створення шаблону відбитка пальця - вибір пункту меню «New ID»;

видалення шаблону відбитка пальця - вибір пункту меню «New ID».

Для запису потрібно прикласти палець два рази. Відбиток пальця збережено. Це можна перевірити у розділі “Show ID”. Якщо ви побачите палець, який записаний у пам'яті, то замок відчиниться на кілька секунд. Час відкриття замку можна регулювати в програмі коду.

Якщо відбудеться сбій і знадобиться перепрошити Arduino UNO, то це можна зробити різними способами. Наприклад через: через інше Arduino або за допомогою Arduino IDE.

Для того щоб прошити одну плату за допомогою іншої, потрібно взяти 2 Arduino, дроти і USB. В першу чергу потрібно налаштувати плату, яка буде виступати в якості програматора. Її потрібно підключити до комп'ютера, відкрити середовище розробки Arduino IDE і знайти в прикладах спеціальний скетч ArduinoISP. Потрібно вибрати цей приклад і прошити плату.

Тепер можна підключати другу плату, яку потрібно прошити, до першої. Після цього потрібно зайти в меню Інструменти і виставити там тип плати і тип програматора.

Можна почати прошивати пристрій. Як тільки прошивка буде відкрита або написана, потрібно перейти в меню Скетч >> завантажити через програматор. Для заливання прошивки не підходить стандартна кнопка завантаження, так як в цьому випадку прошивка буде завантажена на першу плату, на якій вже є прошивка.

Для того щоб прошити плату за допомогою середовища розробки Arduino IDE потрібно завантажити і встановити саму програму Arduino IDE. Також додатково потрібно завантажити і встановити драйвер CH341. Плату Arduino потрібно підключити до комп'ютера і почекати кілька хвилин, поки Windows її впізнає і запам'ятає.

Після цього потрібно завантажити програму Arduino IDE і вибрати потрібну плату: Інструменти - Плата. Також потрібно вибрати порт, до якого вона підключена: Інструменти - Порт. Готова прошивка відкривається подвійним кліком, щоб її завантажити на плату, потрібно натиснути кнопку «Завантажити» вгорі панелі інструментів.

У деяких ситуаціях може виникнути помилка через наявність кирилиці (російських букв) в шляху до папки з кодами. Для цього файл зі скетчами краще створити і зберегти в корені диска з англійським найменуванням.

Ще одним варіантом перепрошивки контролера є використання іншої плати Arduino або спеціальних програматорів, які використовують мікросхеми CP2102 CH340, FTDI та інші. Цей метод вимагає додаткових зусиль і витрат, але дозволяє гнучко змінювати параметри прошивки. Для новачків, безумовно, першим кроком стане використання Arduino IDE, її творці зробили все, щоб спростити цей процес.

У підсумку до даного розділу можна зробити такі висновки, що персонал який займається обслуговуванням даної системи повинен мати знання та уміння як із технічного налагодження системи так із програмного.

Висновки

В сучасне виробництво все більше впроваджується різні сучасні системи контролю доступом і один з таких систем є автоматична система контролю доступом зі сканером відбитку пальця.

Розроблений пристрій призначений для контролю доступом до приміщення, та уникання потрапляння сторонніх людей на територію.

Так як і в сучасних автоматизованих системах в даному пристрої використовується сучасний мікроконтролер який працює заздалегідь складеним алгоритмом, що записаний у вигляді програмного коду на мові програмування Сі в пам'ять мікроконтролера.

Проведено відповідні розрахунки надійності системи, наведено правила експлуатації.

Під час проектування була виконана реальна частина – виготовлення універсального мультитестера, який працює з наступними типами радіодеталей:

- біполярні PNP і NPN транзистори з $H_{fe} < 1000$;
- польові N- і P-канальні MOSFET транзистори (із збагаченим і збідненим каналом);
- N- і P-канальні JFET транзистори;
- резистори 1Ом ... 10МОм постійні і змінні (прилад розпізнає їх як два резистора з одним загальним виводом);
- конденсатори 30пФ ... 5000мкФ (в тому числі електролітичні);
- діоди і світлодіоди;
- тиристори й сімістори.

В результаті можна зробити висновок, що система є працездатною, надійною та актуальною.

Список літератури

1. Система контролю доступу [Електронний ресурс] – Режим доступу до ресурсу: <https://leater.com/ua/services/sistema-kontrolyu-dostupu.html>
2. Системи контролю доступу – що це і як працює [Електронний ресурс] – Режим доступу до ресурсу: <https://ssbb.com.ua/uk/sistemy-kontrolya-dostupa/sistema-kontrolyu-dostupu/sistemy-kontrolya-dostupa-chto-eto-i-kak-eto-rabotaet/>
3. Мащенко Т.Г. Елементи і пристрої автоматики і систем керування. Учебний посібник.- Х.: НТУ ХПІ, 2006.-192 с.
4. Аппаратная платформа Arduino UNO [Електронний ресурс] – Режим доступу до ресурсу: <http://arduino.ru/Hardware/ArduinoBoardUno>
5. Невлюдов І.Ш. Технічні засоби автоматизації: Підручник / І.Ш. Невлюдов, А.О. Андрусевич, О.І. Филипенко, Н.П. Демська, С.П. Новоселов. – Кривий Ріг Криворізький коледж НАУ, 2019 р. – 366 с
6. Кондратець В.О., Віхрова Л.Г. та ін. Дискретні пристрої автоматики. Навчальний посібник- Кіровоград: КНТУ, 2004.-216с.
7. Віхрова Л.Г. Виконавчі пристрої систем автоматики. Навчальний посібник.-Кіровоград: КНТУ, 2004.-145с.
8. Колонтаєвський Ю.П., Сосков А.Г. Промислова електроніка та мікросхемотехніка: теорія і практикум: Навч. Посіб [Електронний ресурс] – Режим доступу до ресурсу: <http://www.twirpx.com/file/128372/>
9. Техніка безпеки при роботі з ПК. [Електронний ресурс] – Режим доступу до ресурсу: <http://moodle.nati.org.ua/mod/book/view.php?id=10031&chapterid=3329>

Додаток А
(обов'язковий)
Лістинг програми

```
// Підключаємо бібліотеки:
#include <Wire.h> // підключаємо бібліотеку для роботи з шиною
I2C
#include <SoftwareSerial.h> // підключаємо бібліотеку для роботи
з програмним UART
#include <LiquidCrystal _ I2C.h> // підключаємо бібліотеку для
роботи з LCD дисплеєм
#include <Adafruit _ Fingerprint.h> // підключаємо бібліотеку
для роботи з модулем відбитків пальців
LiquidCrystal _ I2C lcd (0x27,16,2); // оголошуємо об'єкт lcd
для роботи з LCD дисплеєм, вказуючи параметри дисплея (адреса
I2C = 0x27, кількість стовпців = 16, кількість рядків = 2)
SoftwareSerial mySerial (2,
3); // оголошуємо об'єкт mySerial для роботи з бібліотекою
SoftwareSerial ИМЯ_ОБЪЕКТА ( RX, TX ); // Можна вказувати
будь-які виведення, що підтримують переривання PCINTx
Adafruit _ Fingerprint finger = Adafruit _ Fingerprint (&mySerial);
// оголошуємо об'єкт finger для роботи з бібліотекою Adafruit
Adafruit _ Fingerprint ИМЯ_ОБЪЕКТА = Adafruit _ Fingerprint (ПАРАМЕТР);
// ПАРАМЕТР посилання на об'єкт для роботи з UART до якого
підключений модуль, наприклад: &Serial1
// Оголошуємо змінні і константи :
const uint8 _ t PIN _ led _ OPEN =
4; // вказуємо номер виведення arduino, до якого підключений
зелений світлодіод
const uint8 _ t PIN _ led _ CLOSED =
5; // вказуємо номер виведення arduino, до якого підключений
червоний світлодіод
const uint8 _ t PIN _ beep _ OPEN =
6; // вказуємо номер виведення arduino, до якого підключений
Trema зумер
const uint8 _ t PIN _ key _ OPEN =
7; // вказуємо номер виведення arduino, до якого підключений
Trema ключ
const uint8 _ t PIN _ button _ A =
8; // вказуємо номер виведення arduino, до якого підключена
кнопка А
const uint8 _ t PIN _ button _ B =
9; // вказуємо номер виведення arduino, до якого підключена
кнопка В
uint16 _ t TIM _ Button _ A =
0; // час утримання кнопки А (у сотих частках секунди)
uint16 _ t TIM _ Button _ B =
0; // час утримання кнопки В (у сотих частках секунди)
uint32 _ t TIM _ mode _ ACCESS =
0; // час установки прапора FLG _ mode _ ACCESS замку, що вказує
про необхідність відкриття
int ACK _ finger _ FUN =
0; // результат виконання функції бібліотеки
Adafruit _ Fingerprint
uint8 _ t VAR _ mode _ MENU =
0; // поточний режим виведення меню
```

```

uint8 _ t VAR _ count _ ID = 0; // кількість знайдених ID у базі
uint8 _ t VAR _ first _ ID =
0; // номер першого виведеного ID з усіх знайдених
uint8 _ t VAR _ this _ ID =
0; // номер ID, шаблон якого вимагається зберегти/видалити
uint8 _ t VAR _ array _ ID[162]; // масив знайдених ID у базі
bool FLG _ result _ FUN =
0; // прапор вказує на результат збереження шаблону
bool FLG _ mode _ ACCESS =
0; // прапор вказує про необхідність відкриття замку
bool FLG _ state _ WORK =
1; // прапор вказує про стан роботи замку
bool FLG _ display _ UPD =
1; // прапор вказує про необхідність оновлення інформації на
дисплеї
void setup () {
pinMode (PIN _ button _ A, INPUT); // встановлюємо режим роботи
виведення PIN _ button _ A, як "вхід"
pinMode (PIN _ button _ B, INPUT); // встановлюємо режим роботи
виведення PIN _ button _ B, як "вхід"
pinMode (PIN _ led _ OPEN, OUTPUT); // встановлюємо режим роботи
виведення PIN _ led _ OPEN, як "вихід"
pinMode (PIN _ led _ CLOSED, OUTPUT); // встановлюємо режим
роботи виведення PIN _ led _ CLOSED, як "вихід"
pinMode (PIN _ key _ OPEN, OUTPUT); // встановлюємо режим роботи
виведення PIN _ key _ OPEN, як "вихід"
pinMode (PIN _ beep _ OPEN, OUTPUT); // встановлюємо режим
роботи виведення PIN _ beep _ OPEN, як "вихід"
lcd.init (); // ініціюємо LCD дисплей
lcd.backlight (); // включаємо підсвічування LCD дисплея
lcd.clear (); // стираємо інформацію з дисплея
lcd.setCursor (0,
0); lcd.print (F ("iArduino.ru")); // виводимо текст
"iArduino.ru"
delay (500); // обов'язкова затримка перед ініціалізацією модуля
відбитків пальців
finger.begin (9600);
// finger.begin (57600); // ініціюємо модуль відбитків пальців,
з підключенням через програмний UART на швидкості 57600
(швидкість модуля за умовчанням)
lcd.clear (); // стираємо інформацію з дисплея
lcd.setCursor (0,
0); lcd.print (F ("Scan sensor...")); // виводимо текст
"Scan sensor.".
lcd.setCursor (0, 1); // встановлюємо курсор в позицію:
0 стовпець, 1 рядок
if (finger.verifyPassword()){lcd.print (F ("Found sensor"));}
// якщо модуль відбитків виявлений, виводимо повідомлення
"сенсор виявлений"
else {lcd.print (F ("Sensor not found")); while (1);} // якщо
модуль відбитків не виявлений, виводимо повідомлення "сенсор
не виявлений" і входимо в нескінченний цикл: while (1);
delay (1000); // необов'язкова затримка, щоб можна було
прочитати повідомлення про виявлення модуля
}
void loop () {

```

```

// Передаємо управління кнопкам
Func _ buttons _ control (); // викликаємо функцію
Func _ buttons _ control ();
// Оновлюємо інформацію на дисплеї
if (FLG _ display _ UPD){Func _ display _ show ();} // якщо
встановлений прапор FLG _ display _ UPD (треба оновити
інформацію на дисплеї), то викликаємо функцію
Func _ display _ show ();
// Спілкуємося з модулем відбитків пальців
Func _ sensor _ communication (); // викликаємо функцію Func _ s
ensor _ communication ();
// Управляємо замком, світлодіодами і зумером
if (FLG _ state _ WORK){ // якщо встановлений прапор
FLG _ state _ WORK (замок працює, "State: ENABLE"), то ...
digitalWrite (PIN _ key _ OPEN, FLG _ mode _ ACCESS); // якщо ви
користується електромагнітний замок, де 0-открыто, а 1-
закрито, то другий параметр вказується зі знаком оклику :
digitalWrite (PIN _ key _ OPEN !FLG _ mode _ ACCESS);
digitalWrite (PIN _ led _ OPEN, FLG _ mode _ ACCESS); // включає
мо або вимикаємо світлодіод підключений до виведення
PIN _ led _ OPEN
digitalWrite (PIN _ led _ CLOSED !FLG _ mode _ ACCESS); //
включаємо або вимикаємо світлодіод підключений до виведення
PIN _ led _ CLOSED
if (FLG _ mode _ ACCESS){tone (PIN _ beep _ OPEN, 2000,
100);} // якщо встановлений прапор FLG _ mode _ ACCESS (замок
відкритий), то відправляємо меандр на виведення
PIN _ beep _ OPEN (до якого підключений Трема зумер) тривалістю
100 мс з частотою 2000 Гц
}
// Скидаємо прапор FLG _ mode _ ACCESS, що вказує про
необхідність відкриття замку, через 5 секунд після його установк
и
if (FLG _ mode _ ACCESS){
if (TIM _ mode _ ACCESS >millis()){FLG _ mode _ ACCESS=0;
FLG _ display _ UPD =
1; if (VAR _ mode _ MENU==1){VAR _ mode _ MENU=0;}}
if ((TIM _ mode _ ACCESS 5000)<millis()){FLG _ mode _ ACCESS=0;
FLG _ display _ UPD =
1; if (VAR _ mode _ MENU==1){VAR _ mode _ MENU=0;}}
}
}
// Функція управління кнопками:
void Func _ buttons _ control (){
TIM _ Button _ A=0; // час утримання кнопки А (у сотих частках с
екунди)
TIM _ Button _ B=0; // час утримання кнопки В (у сотих частках с
екунди)
uint8 _ t f=0; // 1-зафіксовано натиснення кнопки, 2-
зафіксовано натиснення кнопки і очищений дисплей, 3-
обе кнопки утримувалися довше 2 сек
if (digitalRead (PIN _ button _ A)){noTone (PIN _ beep _ OPEN);}
// якщо натиснута кнопка А те відключаємо меандр на виведенні
PIN _ beep _ OPEN
if (digitalRead (PIN _ button _ B)){noTone (PIN _ beep _ OPEN);}
// якщо натиснута кнопка В те відключаємо меандр на виведенні

```

```

PIN _ beep _ OPEN
while (digitalRead (PIN _ button _ A)
||digitalRead (PIN _ button _ B)){ // якщо натиснута кнопка А і/
або кнопка В, то створюємо цикл, поки кнопка (и) не буде (ут)
відпущена (ы)
if (digitalRead (PIN _ button _ A)
&&TIM _ Button _ A<65535){TIM _ Button _ A ; if (f==0){f=1;}} //
якщо утримується кнопка А, то збільшуємо час її утримання
if (digitalRead (PIN _ button _ B)
&&TIM _ Button _ B<65535){TIM _ Button _ B ; if (f==0){f=1;}} //
якщо утримується кнопка В, то збільшуємо час її утримання
if (f==1){lcd.clear (); f=2;} // якщо зафіксовано натиснення
кнопки, то стираємо інформацію з дисплея
if (f<3 && TIM _ Button _ A>200
&& TIM _ Button _ B>200){VAR _ mode _ MENU=0; Func _ display _ s
how (); f=3;} // якщо обидві кнопки утримуються довше 2 сік, то
виходимо з меню (VAR _ mode _ MENU=0) і оновлюємо інформацію на
дисплеї (Func _ display _ show ());
delay (10); // пропускаємо 0,01с (пригнічуємо брязкіт кнопок)
}
if (f==2){ FLG _ display _ UPD =
1; // якщо зафіксовано натиснення на кнопку, то ...
switch (VAR _ mode _ MENU){
case 0 : /* поза меню */ if (TIM _ Button _ A >0
&& TIM _ Button _ B >0){VAR _ mode _ MENU = 10;
} // увійти до меню
if (TIM _ Button _ A >0
&& TIM _ Button _ B==0){FLG _ mode _ ACCESS =
1; TIM _ mode _ ACCESS=millis (); } // відкрити замок
if (TIM _ Button _ A==0
&& TIM _ Button _ B >0){FLG _ mode _ ACCESS =
1; TIM _ mode _ ACCESS=millis (); } // відкрити замок
break;
case 10 : /* 1 рівень меню -
вкл/выкл замок */ if (TIM _ Button _ A >0
&& TIM _ Button _ B==0){VAR _ mode _ MENU = 99;
} // перейти до попереднього пункту меню
if (TIM _ Button _ A==0
&& TIM _ Button _ B >0){VAR _ mode _ MENU = 20;
} // перейти до наступного пункту меню
if (TIM _ Button _ A >0
&& TIM _ Button _ B >0){VAR _ mode _ MENU = 11;
} // вибрати поточний пункт меню
break;
case 11 : /* 2 рівень меню -
вкл/выкл замок */ if (TIM _ Button _ A >0
&& TIM _ Button _ B==0){FLG _ state _ WORK = FLG _ state _ WORK?
0: 1; } // вкл/выкл
if (TIM _ Button _ A==0
&& TIM _ Button _ B >0){FLG _ state _ WORK = FLG _ state _ WORK?
0: 1; } // вкл/выкл
if (TIM _ Button _ A >0
&& TIM _ Button _ B >0){VAR _ mode _ MENU = 10;
} // вийти з поточного пункту меню
break;

```

```

case 20 : /* 1 рівень меню -
показати ID */ if (TIM _ Button _ A >0
&& TIM _ Button _ B==0){VAR _ mode _ MENU = 10;
} // перейти до попереднього пункту меню
if (TIM _ Button _ A==0
&& TIM _ Button _ B >0){VAR _ mode _ MENU = 30;
} // перейти до наступного пункту меню
if (TIM _ Button _ A >0
&& TIM _ Button _ B >0){VAR _ mode _ MENU = 21;
} // вибрати поточний пункт меню
break;
// case 21 : /* 2 рівень меню -
показати ID */ це пошук ID з написом Please wait ...
// без реакції на натиснення кнопок
//
case 22 : /* 3 рівень меню -
показати ID */ if (TIM _ Button _ A >0
&& TIM _ Button _ B==0){VAR _ first _ ID -
= VAR _ first _ ID>0?3: 0;
} // зменшити перший ID, що виводиться на дисплей
if (TIM _ Button _ A==0
&& TIM _ Button _ B >0){VAR _ first _ ID =
((VAR _ first _ ID 3)<VAR _ count _ ID)?3:
0;} // збільшити перший ID, що виводиться на дисплей
if (TIM _ Button _ A >0 &&
case 30 : /* 1 рівень меню -
зберегти ID */ if (TIM _ Button _ A >0
&& TIM _ Button _ B==0){VAR _ mode _ MENU = 20;
} // перейти до попереднього пункту меню
if (TIM _ Button _ A==0
&& TIM _ Button _ B >0){VAR _ mode _ MENU = 40;
} // перейти до наступного пункту меню
if (TIM _ Button _ A >0
&& TIM _ Button _ B >0){VAR _ mode _ MENU = 31;
} // вибрати поточний пункт меню
break;
// case 31 : /* 2 рівень меню -
зберегти ID */ це пошук ID з написом Please wait ...
// без реакції на натиснення кнопок
//
case 32 : /* 3 рівень меню -
зберегти ID */ if (TIM _ Button _ A >0
&& TIM _ Button _ B==0){VAR _ this _ ID -= VAR _ this _ ID>0?1:
0; } // зменшити ID по якому буде збережений відбиток
if (TIM _ Button _ A==0
&& TIM _ Button _ B >0){VAR _ this _ ID = VAR _ this _ ID<162?1:
0; } // збільшити ID по якому буде збережений відбиток
if (TIM _ Button _ A >0
&& TIM _ Button _ B >0){VAR _ mode _ MENU = 33;
} // вибрати поточний пункт меню
break;
// case 33 : /* 4 рівень меню -
зберегти ID */ це очікування пальця (вперше), що прикладається,
з написом Put finger ...
// без реакції на натиснення кнопок
//

```

```

// case 34 : /* 5 рівень меню -
зберегти ID */ це очікування конвертації зображення в шаблон (в
перше) з написом converting ...
// без реакції на натиснення кнопок
//
// case 35 : /* 6 рівень меню -
зберегти ID */ це очікування відсутності пальця перед сканером
з написом remove finger ...
// без реакції на натиснення кнопок
//
// case 36 : /* 7 рівень меню -
зберегти ID */ це очікування пальця (в другий раз), що
прикладається, з написом Put finger ...
// без реакції на натиснення кнопок
//
// case 37 : /* 8 рівень меню -
зберегти ID */ це очікування конвертації зображення в шаблон (в
другий раз) з написом converting ...
// без реакції на натиснення кнопок
//
// case 38 : /* 9 рівень меню -
зберегти ID */ це очікування створення і збереження шаблону з
написом Saved ...
// без реакції на натиснення кнопок
//
case 39 : /*10 рівень меню -
зберегти ID */ if (TIM _ Button _ A >0
&& TIM _ Button _ B==0){VAR _ mode _ MENU = 30;
} // у початок поточного пункту меню
if (TIM _ Button _ A==0
&& TIM _ Button _ B >0){VAR _ mode _ MENU = 30;
} // у початок поточного пункту меню
if (TIM _ Button _ A >0
&& TIM _ Button _ B >0){VAR _ mode _ MENU = 30;
} // у початок поточного пункту меню
break;
case 40 : /* 1 рівень меню -
видалити ID */ if (TIM _ Button _ A >0
&& TIM _ Button _ B==0){VAR _ mode _ MENU = 30;
} // перейти до попереднього пункту меню
if (TIM _ Button _ A==0
&& TIM _ Button _ B >0){VAR _ mode _ MENU = 99;
} // перейти до наступного пункту меню
if (TIM _ Button _ A >0
&& TIM _ Button _ B >0){VAR _ mode _ MENU =
41; VAR _ this _ ID=0; } // вибрати поточний пункт меню
break;
case 41 : /* 2 рівень меню -
видалити ID */ if (TIM _ Button _ A >0
&& TIM _ Button _ B==0){VAR _ this _ ID = VAR _ this _ ID==255?0
: 255; } // 255 - all / 0 - one
if (TIM _ Button _ A==0
&& TIM _ Button _ B >0){VAR _ this _ ID = VAR _ this _ ID==255?0
: 255; } // 255 - all / 0 - one

```

```

if (TIM _ Button _ A >0
&& TIM _ Button _ B >0){VAR _ mode _ MENU = VAR _ this _ ID==255
?42: 43; } // вибрати поточний пункт меню
break;
// case 42 : /* 3 рівень меню -
видалити ID */ це видалення усіх ID з написом Please wait ...
// без реакції на натиснення кнопок
//
// case 43 : /* 4 рівень меню -
видалити ID */ це пошук ID з написом Please wait ...
// без реакції на натиснення кнопок
case 44 : /* 5 рівень меню -
видалити ID */ if (TIM _ Button _ A >0
&& TIM _ Button _ B==0){VAR _ this _ ID -= VAR _ this _ ID>0?1:
0; } // зменшити ID по якому буде видалений відбиток
if (TIM _ Button _ A==0
&& TIM _ Button _ B >0){VAR _ this _ ID = VAR _ this _ ID<162?1:
0; } // збільшити ID по якому буде видалений відбиток
if (TIM _ Button _ A >0
&& TIM _ Button _ B >0){VAR _ mode _ MENU = 45;
} // вибрати поточний пункт меню
break;
// case 45 : /* 6 рівень меню -
видалити ID */ це видалення вибраного ID з написом Please wait
...
// без реакції на натиснення кнопок
//
case 99 : /* 1 рівень меню -
вийти з меню */ if (TIM _ Button _ A >0
&& TIM _ Button _ B==0){VAR _ mode _ MENU = 40;
} // перейти до попереднього пункту меню
if (TIM _ Button _ A==0
&& TIM _ Button _ B >0){VAR _ mode _ MENU = 10;
} // перейти до наступного пункту меню
if (TIM _ Button _ A >0
&& TIM _ Button _ B >0){VAR _ mode _ MENU = 0;
} // вибрати поточний пункт меню
break;
}
}
}
// функція виведення інформації на дисплей
void Func _ display _ show (){ FLG _ display _ UPD=0; lcd.clear
());
switch (VAR _ mode _ MENU){
case 0 : /* поза меню */ lcd.setCursor (0,
0); lcd.print (F ("State: ")); lcd.print (FLG _ state _ WORK ?
F ("ENABLE") : F ("DISABLE") );
lcd.setCursor (0, 1); lcd.print (F ("Access: ")); lcd.print
(FLG _ mode _ ACCESS ? F ("OPENED") : F ("CLOSED" ) );
break;
case 1 : /* замок відкритий відбитком */ lcd.setCursor (0, 0);
lcd.print (F ("Found ID : ")); if (VAR _ this _ ID<100)
{lcd.print (F ("0"));} if (VAR _ this _ ID<10){lcd.print (F ("0"
));} lcd.print (VAR _ this _ ID);
lcd.setCursor (0, 1); lcd.print (F ("Access: OPENED") );

```

```

break;
case 10 : /* 1 рівень меню - вкл/выкл замок */ lcd.setCursor (0,
0); lcd.print (F ("Menu > Set state"));
break;
case 11 : /* 2 рівень меню - вкл/выкл замок */ lcd.setCursor (0,
0); lcd.print (F ("State:"));
lcd.setCursor (9, 0); lcd.print (F ("ENABLE"));
lcd.setCursor (9, 1); lcd.print (F ("DISABLE"));
lcd.setCursor (7, FLG _ state _ WORK?0: 1); lcd.print (F (">"));
break;
case 20 : /* 1 рівень меню - показати ID */ lcd.setCursor (0,
0); lcd.print (F ("Menu > Show ID"));
break;
case 21 : /* 2 рівень меню - показати ID */ lcd.setCursor (0,
0); lcd.print (F ("Scan sensor "...));
lcd.setCursor (0, 1); lcd.print (F ("Please wait "...));
break;
case 22 : /* 3 рівень меню - показати ID */ lcd.setCursor (0,
0); lcd.print (F ("Found ")); lcd.print (VAR _ count _ ID); lcd.
print (F (" ID:"));
lcd.setCursor (0, 1); lcd.print (VAR _ first _ ID==0?F (" ")
: F("<")); for (int i=0; i<3; i ){if (VAR _ count _ ID>VAR _
first _ ID i){lcd.print (i==0?F (" ")
: F(", ")); if (VAR _ array _ ID[(VAR _ first _ ID i)]<100){lcd.
print (F ("0"));} if (VAR _ array _ ID[(VAR _ first _ ID i)]<10)
{lcd.print (F ("0"));} lcd.print (VAR _ array _ ID[(VAR _ first
_ ID i)]);}} if (VAR _ count _ ID>(VAR _ first _ ID 3)){lcd.setC
ursor (15,1); lcd.print (F (">"));}
break;
case 30 : /* 1 рівень меню - зберегти ID */ lcd.setCursor (0,
0); lcd.print (F ("Menu > New ID"));
break;
case 31 : /* 2 рівень меню - зберегти ID */ lcd.setCursor (0,
0); lcd.print (F ("Scan sensor "...));
lcd.setCursor (0, 1); lcd.print (F ("Please wait "...));
break;
case 32 : /* 3 рівень меню - зберегти ID */ lcd.setCursor (0,
0); lcd.print (F ("Menu>NewID>SetID"));
lcd.setCursor (6,
1); if (VAR _ this _ ID<100){lcd.print (F ("0"));}
} if (VAR _ this _ ID<10){lcd.print (F ("0"));} lcd.print
(VAR _ this _ ID);
lcd.setCursor (4, 1); if (VAR _ this _ ID>0
){lcd.print (F ("<"));}
lcd.setCursor (10,1); if (VAR _ this _ ID<162){lcd.print
(F (">"));}
break;
case 33 : /* 4 рівень меню - зберегти ID */ lcd.setCursor (0,
0); lcd.print (F ("Menu > NewID ")); if (VAR _ this _ ID<100)
{lcd.print (F ("0"));} if (VAR _ this _ ID<10){lcd.print
(F ("0"));} lcd.print (VAR _ this _ ID);
lcd.setCursor (0, 1); lcd.print (F ("Put finger "...));
break;
case 34 : /* 5 рівень меню - зберегти ID */ lcd.setCursor (0,
0); lcd.print (F ("Menu > NewID ")); if (VAR _ this _ ID<100)
{lcd.print (F ("0"));} if (VAR _ this _ ID<10){lcd.print

```

```

(F ("0"));} lcd.print (VAR _ this _ ID);
lcd.setCursor (0, 1); lcd.print (F ("Converting "...));
break;
case 35 : /* 6 рівень меню - зберегти ID */ lcd.setCursor (0,
0); lcd.print (F ("Menu > NewID ")); if (VAR _ this _ ID<100)
{lcd.print (F ("0"));} if (VAR _ this _ ID<10){lcd.print
(F ("0"));} lcd.print (VAR _ this _ ID);
lcd.setCursor (0, 1); lcd.print (F ("Remove finger "...));
break;
case 36 : /* 7 рівень меню - зберегти ID */ lcd.setCursor (0,
0); lcd.print (F ("Menu > NewID ")); if (VAR _ this _ ID<100)
{lcd.print (F ("0"));} if (VAR _ this _ ID<10){lcd.print
(F ("0"));} lcd.print (VAR _ this _ ID);
lcd.setCursor (0, 1); lcd.print (F ("Put finger "...));
break;
case 37 : /* 8 рівень меню - зберегти ID */ lcd.setCursor (0,
0); lcd.print (F ("Menu > NewID ")); if (VAR _ this _ ID<100)
{lcd.print (F ("0"));} if (VAR _ this _ ID<10){lcd.print
(F ("0"));} lcd.print (VAR _ this _ ID);
lcd.setCursor (0, 1); lcd.print (F ("Converting "...));
break;
case 38 : /* 9 рівень меню - зберегти ID */ lcd.setCursor (0,
0); lcd.print (F ("Menu > NewID ")); if (VAR _ this _ ID<100)
{lcd.print (F ("0"));} if (VAR _ this _ ID<10){lcd.print
(F ("0"));} lcd.print (VAR _ this _ ID);
lcd.setCursor (0, 1); lcd.print (F ("Saved "...));
break;
case 39 : /*10 рівень меню - зберегти ID */ lcd.setCursor (0,
0); lcd.print (F ("Menu > NewID ")); if (VAR _ this _ ID<100)
{lcd.print (F ("0"));} if (VAR _ this _ ID<10){lcd.print
(F ("0"));} lcd.print (VAR _ this _ ID);
lcd.setCursor (0,
1); lcd.print (FLG _ result _ FUN?F ("Finger saving!"):F
("ERROR :("));
break;
case 40 : /* 1 рівень меню - видалити ID */ lcd.setCursor
(0, 0); lcd.print (F ("Menu > Del ID"));
break;
case 41 : /* 2 рівень меню - видалити ID */ lcd.setCursor
(0, 0); lcd.print (F ("Menu>DelID"));
lcd.setCursor (13,0); lcd.print (F ("All"));
lcd.setCursor (13,1); lcd.print (F ("One"));
lcd.setCursor (11, VAR _ this _ ID==255?0:
1); lcd.print (F (">"));
break;
case 42 : /* 3 рівень меню - видалити ID */ lcd.setCursor
(0, 0); lcd.print (F ("Deleted ID "...));
lcd.setCursor (0, 1); lcd.print (F ("Please wait "...));
break;
case 43 : /* 4 рівень меню - видалити ID */ lcd.setCursor
(0, 0); lcd.print (F ("Scan sensor "...));
lcd.setCursor (0, 1); lcd.print (F ("Please wait "...));
break;
case 44 : /* 5 рівень меню - видалити ID */ lcd.setCursor
(0, 0); lcd.print (F ("Menu>Del One ID"));

```

```

lcd.setCursor (6,
1); if (VAR _ this _ ID<100){lcd.print (F ("0"));
} if (VAR _ this _ ID<10){lcd.print (F ("0"));}
lcd.print (VAR _ this _ ID);
lcd.setCursor (4, 1); if (VAR _ this _ ID>0
){lcd.print (F ("<"));}
lcd.setCursor (10,1); if (VAR _ this _ ID<162){lcd.print
(F (">"));}
break;
case 45 : /* 6 рівень меню - видалити ID */ lcd.setCursor
(0, 0); lcd.print (F ("Menu>Del ID ")); if (VAR _ this _ ID<100)
{lcd.print (F ("0"));} if (VAR _ this _ ID<10)
{lcd.print (F ("0"));} lcd.print (VAR _ this _ ID);
lcd.setCursor (0, 1); lcd.print (F ("Please wait "...));
break;
case 49 : /*10 рівень меню - видалити ID */ lcd.setCursor
(0, 0); lcd.print (F ("Menu> Del ")); if (VAR _ this _ ID==255)
{lcd.print (F ("All ID"));}else{lcd.print (F ("ID "));
} if (VAR _ this _ ID<100){lcd.print (F ("0"));} if
(VAR _ this _ ID<10){lcd.print (F ("0"));} lcd.print (VAR _ this
_ ID);
lcd.setCursor (0,
1); lcd.print (FLG _ result _ FUN?F ("Finger deleted!"):F
("ERROR :("));
break;
case 99 : /* 1 рівень меню - вийти з меню */ lcd.setCursor (0,
0); lcd.print (F ("Menu > Exit"));
break;
}
}
// функція спілкування з модулем відбитків пальців
void Func _ sensor _ communication (){
switch (VAR _ mode _ MENU){
// Якщо вимагається звіряти відбитки пальців
case 0 : if (!FLG _ mode _ ACCESS){
if (millis () P0 == 0){ // перевіряємо кожні 500 мс
if (finger.getImage ()
== FINGERPRINT _ OK){ // Захоплюємо зображення, якщо результат в
иконання дорівнює константі FINGERPRINT _ OK (коректне
завантаження зображення), то проходимо далі
if (finger.image2Tz ()
== FINGERPRINT _ OK){ // Конвертуємо отримане зображення, якщо
результат виконання дорівнює константі FINGERPRINT _ OK
(зображення сконвертировано), то проходимо далі
if (finger.fingerFastSearch ()
== FINGERPRINT _ OK){ // Знаходимо відповідність у базі цих
відбитків пальців, якщо результат виконання дорівнює константі
FINGERPRINT _ OK (знайдена відповідність), то проходимо далі
VAR _ this _ ID = finger.fingerID;
VAR _ mode _ MENU = 1;
FLG _ display _ UPD = 1;
FLG _ mode _ ACCESS = 1;
TIM _ mode _ ACCESS = millis ();
}else{finger.fingerFastSearch ();}}}}}} // Цей рядок не являється
обробником відсутності відповідей у базі цих відбитків паль
ців! Просто без неї в деяких версіях Arduino IDE компілятор не

```

```

зможе коректно підставити результат функції fingerFastSearch ()
для порівняння з константою FINGERPRINT _ OK в умові оператора
if ().
break;
// Якщо вимагається знайти усе ID збережені у базі відбитків
case 21
: VAR _ first _ ID=0; VAR _ count _ ID=0; // скидаємо кількість
знайдених Id і номер першого серед відображених
for (uint8 _ t id =
0; id<162; id ){ // прохід по 162 ідентифікаторам
if (finger.loadModel (id)
== FINGERPRINT _ OK){ // якщо по вказаному ідентифікатору
знайдений шаблон відбитку, то ...
VAR _ array _ ID[VAR _ count _ ID]=id; VAR _ count _ ID ; //
зберігаємо номер ідентифікатора і збільшуємо кількість знайдених
ID
}
} VAR _ mode _ MENU=22; FLG _ display _ UPD=1; // переходимо в 3
рівень меню ( показати ID ) і встановлюємо прапор що вказує п
ро необхідність оновити інформацію на дисплеї
break;
// Якщо вимагається знайти перший вільний ID у базі відбитків
для його зміни на потрібний для збереження
case 31
: VAR _ this _ ID=161; // встановлюємо номер 161 для першого
вільного ID у базі відбитків
for (uint8 _ t id =
0; id<162; id ){ // прохід по 162 ідентифікаторам
if (VAR _ this _ ID==161){if (finger.loadModel (id)!=FINGERPRINT
_ OK){VAR _ this _ ID=id;}} // якщо по вказаному ідентифікатору
не знайдений шаблон відбитку, то зберігаємо його
} VAR _ mode _ MENU=32; FLG _ display _ UPD=1; // переходимо в 3
рівень меню ( зберегти ID ) і встановлюємо прапор що вказує про
необхідність оновити інформацію на дисплеї
break;
// Якщо вимагається визначити наявність пальця на сканері і
зісканувати його
case 33 : ACK _ finger _ FUN=finger.getImage (); // захоплюємо
зображення, зі збереженням результату в змінну ACK _ finger _ FU
N
if (ACK _ finger _ FUN==FINGERPRINT _ OK){ // зображення
відбитку пальця коректно завантажилось
VAR _ mode _ MENU=34; FLG _ display _ UPD=1; // переходимо в 5 р
івень меню ( зберегти ID ) і встановлюємо прапор що вказує про
необхідність оновити інформацію на дисплеї
}else if (ACK _ finger _ FUN==FINGERPRINT _ NOFINGER){ // сканер
не виявив відбиток пальця, залишаємося в поточному рівні
}else{FLG _ result _ FUN=0; VAR _ mode _ MENU=39; FLG _ display
_ UPD=1;} // переходимо в 10 рівень меню ( результат збереження
ID ) і встановлюємо прапор що вказує про необхідність оновити
інформацію на дисплеї
break;
// Якщо потрібно сконвертировать перше отримане відскановане
зображення
case 34
: FLG _ result _ FUN=0; FLG _ display _ UPD=1; // встановлюємо

```

```

прапор що вказує про необхідність оновити інформацію на дисплеї
VAR _ mode _ MENU=finger.image2Tz (1) ==FINGERPRINT _ OK?35:
39; // конвертуємо перше зображення, якщо результат виконання
цієї операції == FINGERPRINT _ OK, означає зображення
сконвертированно
break;
// Якщо вимагається зафіксувати відсутність пальця на сканері
case 35
: delay (500); VAR _ mode _ MENU=36; FLG _ display _ UPD=1; //
чекаємо 0.5 сік і намагаємося перейти в 7 рівень меню ( зберегти
ID )
while (finger.getImage ()!= FINGERPRINT _ NOFINGER){;} // не
виходимо з циклу, поки палець не перестане скануватися
break;
// Якщо вимагається визначити наявність пальця на сканері і
зісканувати його
case 36 : ACK _ finger _ FUN=finger.getImage (); // захоплюємо
зображення, зі збереженням результату в змінну
ACK _ finger _ FUN
if (ACK _ finger _ FUN==FINGERPRINT _ OK){ // зображення
відбитку пальця коректно завантажилось
VAR _ mode _ MENU=37; FLG _ display _ UPD=1; // переходимо в 8
рівень меню ( зберегти ID ) і встановлюємо прапор що вказує про
необхідність оновити інформацію на дисплеї
}else if (ACK _ finger _ FUN==FINGERPRINT _ NOFINGER){ // сканер
не виявив відбиток пальця, залишаємося в поточному рівні
}else{FLG _ result _ FUN=0; VAR _ mode _ MENU=39; FLG _ display
_ UPD=1;} // переходимо в 10 рівень меню ( результат збереження
ID ) і встановлюємо прапор що вказує про необхідність оновити
інформацію на дисплеї
break;
// Якщо потрібно сконвертировать друге отримане відскановане
зображення
case 37
: FLG _ result _ FUN=0; FLG _ display _ UPD=1; // встановлюємо
прапор що вказує про необхідність оновити інформацію на дисплеї
VAR _ mode _ MENU=finger.image2Tz (2) ==FINGERPRINT _ OK?38:
39; // Конвертуємо друге зображення, якщо результат виконання
цієї операції == FINGERPRINT _ OK, означає зображення
сконвертированно
break;
// Якщо вимагається створити і зберегти шаблон з перших двох
відсканованих зображень
case 38
: FLG _ result _ FUN=0; VAR _ mode _ MENU=39; FLG _ display _ UP
D=1; // намагаємося перейти в 10 рівень меню ( результат
збереження ID ) і встановлюємо прапор що вказує про необхідність
оновити інформацію на дисплеї
if (finger.createModel ()==FINGERPRINT _ OK){ // створюємо
модель (шаблон) відбитку пальця, по двох зображеннях
if (finger.storeModel (VAR _ this _ ID)
==FINGERPRINT _ OK){ // зберігаємо модель (шаблон) відбитку
пальця, по двох зображеннях
FLG _ result _ FUN=1; // якщо створення і збереження модель
(шаблону) виконано успішно, то встановлюємо прапор що вказує про
успішне збереження

```

```

}else{finger.storeModel (VAR _ this _ ID);} // Цей рядок не є
обробкою помилки збереження моделі (шаблону) відбитку пальця!
Просто без неї в деяких версіях Arduino IDE компілятор не
зможе коректно підставити результат функції storeModel () для
порівняння з константою FINGERPRINT _ OK в умові оператора if().
}else{finger.createModel ();} // Цей рядок не є обробкою помилки
створення моделі (шаблону) відбитку пальця! Просто без неї в
деяких версіях Arduino IDE компілятор не зможе коректно
підставити результат функції createModel () для порівняння з
константою FINGERPRINT _ OK в умові оператора if ().
break;
case 39
: delay (2000); FLG _ result _ FUN=0; VAR _ mode _ MENU=30;
FLG _ display _ UPD=1; // виходимо з меню через 2 сік
break;
// Якщо вимагається видалити усе ID з бази відбитків
case 42
: FLG _ result _ FUN=1; VAR _ mode _ MENU=49; FLG _ display _ UP
D=1; // встановлюємо результат і намагаємося перейти в 10 рівень
меню ( результат видалення ID ) встановивши прапор що вказує
про необхідність оновити інформацію на дисплеї
for (uint8 _ t id =
0; id<162; id ){ // прохід по 162 ідентифікаторам
if (finger.loadModel (id)
== FINGERPRINT _ OK){ // якщо по вказаному ідентифікатору
знайдений шаблон відбитку, то ...
if (finger.deleteModel (id)!=FINGERPRINT _ OK)
{FLG _ result _ FUN=1;} // якщо шаблон не віддаляється, скидаємо
результат
}
}
break;
// Якщо вимагається знайти останній зайнятий ID у базі відбитків
для його зміни на потрібний для збереження
case 43
: VAR _ this _ ID=0; // скидаємо номер останнього зайнятого ID у
базі відбитків
for (uint8 _ t id =
0; id<162; id ){ // прохід по 162 ідентифікаторам
if (finger.loadModel (id)
==FINGERPRINT _ OK){VAR _ this _ ID=id;} // якщо по вказаному
ідентифікатору знайдений шаблон відбитку, то зберігаємо його
} VAR _ mode _ MENU=44; FLG _ display _ UPD=1; // переходимо в 5
рівень меню ( видалити ID ) і встановлюємо прапор що вказує про
необхідність оновити інформацію на дисплеї
break;
// Якщо вимагається видалити вибраний ID з бази відбитків
case 45
: FLG _ result _ FUN=0; VAR _ mode _ MENU=49; FLG _ display _ UP
D=1; // намагаємося перейти в 10 рівень меню ( результат
видалення ID ) і встановлюємо прапор що вказує про необхідність
оновити інформацію на дисплеї
if (finger.deleteModel (VAR _ this _ ID)
==FINGERPRINT _ OK){FLG _ result _ FUN=1;} // Видаляємо шаблон
відбитку пальця VAR _ this _ ID з бази даних, при успішному
виконанні, встановлюємо прапор FLG _ result _ FUN

```

```
break;
case 49
: delay (2000); FLG _ result _ FUN=0; VAR _ mode _ MENU=40; FLG
 _ display _ UPD=1; // виходимо з меню через 2 сік
break;
default: break;
}
}
```