МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ І.І.МЕЧНИКОВА

Факультет математики, фізики та інформаційних технологій
Кафедра математичного забезпечення комп'ютерних систем

Курсовий проект

з курсу «Комп'ютерна схемотехніка та програмування мікроконтролерів»

на тему: «Система аутентифікації на основі RFID-RC522»

Виконав студент 3 курсу
спеціальність 123 «Комп'ютерна інженерія»
Панов Володимир Михайлович
Керівник: Гунченко Юрій Олександрович

Одеса 2023

Анотація

У курсовому проекті розроблено систему аутентифікації на основі мікроконтролеру Arduino UNO R3 та модуля зчитування ID карток — RFID-RC522.

Метою курсового проекту є проектування та розробка системи, що буде здатна виконувати допуск користовачів по ID ключам, також система має можливість додавати та видаляти ключі з пам’яті мікроконтролеру. Для захисту від отримання доступу дітям система має перед тим як надати доступ запитати у користовача простий математичний приклад, у випадку правильної відповіді на який доступ буде надано.

Цей звіт влючає у себе функціональний прототип такої системи, яка буде мати окрім базового функціоналу надання доступу за допомогою ID ключів, функціонал картки адміністратора з можливістю додавання та видалення ключів, а також з захистом від надання доступу дітям.

Даний проект може бути використаний у безлічі побутових та промислових областях, починаючи від використання у Інтернеті речей для доступу до розумних домів та закінчуючи системою авторизації на промислових об’єктах.

Зміст

Перелік умовних скорочень

- Arduino IDE (англ. Integrated Development Environment) – інтегроване середовище розробки Arduino;
- ID ключ (англ. Identity Document) — унікальний ключ;
- LCD (англ. Liquid Crystal Display) — рідинно-кристалічний дисплей;
- DC (англ. Direct Current) — постійний струм;
- USBasp — програматор для мікроконтролеру Atmel AVR;
- I/O device (англ. Input/Output) — пристрій з можливістю вводу/виводу.

Вступ

Активне впровадження у життя технологій доступу призвело до збільшення використання систем з технологією ID ключів. Великий попит на такі технології викликаний багатьма причинами, серед них такі як: дешевизна кожного ключа, технічна можливість створення мильйонів безпечних ключів, зручність користування та ін.

Даний курсовий проект має на меті створити прототип такої системи. Основні задачі прототипу становлять забезпечення базового функціоналу та створення макетного стенду для тестування розробленого приладу.

Окрім плати Arduino UNO модуля RFID-RC522 для зчитування інформації ключів та LCD екрану для контакту з користовачем, у проекті використовуються багато інших електронних елементів, роботу та цілі впровадження яких буде пояснено у теоретичній частині звіту.

Курсовий проект має на меті вдосконалити та поглибити знання у області комп’ютерної схемотехніки та програмування мікроконтролерів. Практична складова роботи допоможе скористатися на практиці знаннями, отриманними під час проектування лабораторних робіт. Досвіт роботи з проектом насиченим периферійними пристроями являє собою важливу складову вивчаємої дисципліни.

1 Теоретична частина

1.1 Платформа Arduino

Рис. 1: Вигляд плати Arduino Uno R3

Arduino (Ардуіно) — апаратна обчислювальна платформа для аматорського конструювання, основними компонентами якої є плата мікроконтролера з елементами вводу/виводу та середовище розробки на мові програмування, що є спрощеною підмножиною C/C++. Arduino може використовуватися як для створення автономних інтерактивних об'єктів, так і підключатися до програмного забезпечення. Інформація про плату (малюнок друкованої плати, специфікації елементів, програмне забезпечення) знаходяться у відкритому доступі і можуть бути використані у будь-якому аматорському проекті.

Arduino UNO — плата мікроконтролера на основі ATmega328P. Вона має 14 цифрових входів/виходів (з яких 6 можна використовувати як ШІМ-виходи), 6 аналогових входів, керамічний резонатор 16 МГц, USB-з’єднання, роз’єм живлення, роз’єм ICSP і кнопку скидання. Зовнішній вигляд плати на рис.1.

Мікроконтролер містить все необхідне для підтримки початкового використання. Для перевірки його роботи достатньо просто підключити його до комп’ютера за допомогою кабелю USB або заживити мікроконтролер від акумулятора. Технічні характеристики мікроконтролеру наведені у табл.1.

Мікроконтролер	ATmega328P	
USB-з'єднання	USB-B	
Контакти підключення	Вбудований LED пін	13
	Цифровий I/O пін	
	Аналогові I/O пінів	6
	ШІМ пінів	6
Зв’язок з периферією	UART	
	I2C	
	SPI	
Живлення	I/O Напруга	5V
	Вхідна напруга	7-12V
	Сила струму на пін	20mA
Тактова частота	Освновний процесор	ATmega328P 16MHz
	Послідовний USB процесор	ATmega16U2 16 MHz
Пам’ять	ATmega328P	2KB SRAM, 32KB FLASH, 1KB EEPROM

Табл. 1: Технічні характеристики Arduino Uno R3

1.2 Модуль RFID-RC522

Рис. 2: Вигляд модуля RFID-RC522

У цьому проекті використовується найпоширеніший модуль безконтактної ідентифікації RFID RC522. Девайс служить для безконтактного читання RFID-тегів, працюючи на частоті 13.56 МГц. Був вибраний саме SPI інтерфейс для підключення до Arduino, воліючи його за його надійність

порівняно з UART та I2C. Модуль підтримує різні типи карток, включаючи MIFARE S50, MIFARE S70, MIFARE UltraLight, MIFARE Pro, MIFARE DESfire. Основні характеристики:

- •Заснований на мікросхемі MFRC522
- •Напруга живлення: 3.3V
- •Струм: 13-26mA
- •У режимі очікування: 10-13mA
- •У сплячому режимі: менше 80mkA
- •Робоча частота: 13.56MHz
- •Дальність зчитування: 0 ~ 60mm
- •Інтерфейс: SPI, максимальна швидкість передачі 10Мбіт / с

1.3 Дисплей LCD 1602A

Рис. 3: Вигляд дисплею LCD 1602A

Экран 1602 LCD може видавати дві строки по 16 символів інформації одночастно, приведемо основні параметри 1602LCD:

- Ємність дисплея: 16×2 символи
- Робоча напруга мікросхеми: 4,5 ~ 5,5V

- Робочий струм: 2,0mA (5,0 V)
- Оптимальна робоча напруга модуля 5,0V
- Розмір символу: 2,95 × 4,35 (Ш×В) мм.

Опис інтерфейсу:

1. Два роз'єма для живлення, один для живлення модуля, інший для підсвічування, зазвичай використовують 5В. У цьому проекті ми використовуємо 5В для підсвічування.

2. VL – штифт для налаштування коефіцієнта контрастності; він зазвичай підключає потенціометр (не більше 5кОм) послідовно для його налаштування. У цьому експерименті ми використовуємо потенціометр 10кОм. Для його підключення він має 2 методи, а саме високий потенціал і низький потенціал.

3. RS використовується для вибору команд/даних. Коли на роз'єм знаходиться на високому рівні напруги, він знаходиться в режимі читання даних; коли він знаходиться на низькому рівні, він знаходиться в командному режимі.

4. Вивід RW дуже поширений у РК-дисплеях. Це роз'єм вибору для читання/запису. Коли він знаходиться на високому рівні, це значить перебування у режимі читання; коли він знаходиться на низькому рівні — в режимі запису.

5. Вивід E. Зазвичай, коли сигнал у шині стабілізується, він посилає позитивний імпульс, що вимагає операції зчитування. Коли цей пін знаходиться на високому рівні, шина не може мати жодних змін.

6. D0-D7 — це 8-розрядна двонаправлена паралельна шина, яка використовується для передачі команд і даних.

У цьому проекті буде використано можливисть дислпею підключати лише 4 з 8 виходів, а саме D4-D7 з двонаправленої паралельної шини даних. Таким чином буде вдвічі знижена швидкість зв'язку між платою та екраном, але на платі збільшиться кількість вільних цифрових портів для інших цілей.

1.4 Серводвигун

Рис. 4: Вигляд серводвигуна

Серводвигун — це поворотний привід з регулюванням положення. Він складається з корпусу, друкованої плати, двигуна без сердечника, редуктора та датчика положення.

Приймач або MCU виводить сигнал на серводвигун. Двигун має вбудовану опорну схему, яка видає сигнал. Шестерня передає зусилля на вал. Датчик за сигналом зворотного зв'язку визначить, чи досяг він заданого положення.

Серводвигуни використовуються в системах керування, які вимагають наявності та підтримки різних кутів. Коли швидкість двигуна визначена,

шестерня змусить пелюстки обертатися. Коли різниця напруг зменшується до нуля, двигун зупиняється. Діапазон кута повороту становить 0-180 градусів.

Серводвигун має три різнокольорові дроти підключення: коричневий для землі, червоний для плюса живлення, помаранчевий для сигнальної лінії.

У цьому проекті серводвигун нам знадобиться для емітації зпрацювання відкриття замку двері, якщо користувач надасть правильний ключ та вирішить задачу.

1.5 Пасивний зумер

Рис. 5: Вигляд пасивного зумера

Пасивний зумер використовують у тих випадках, коли потрібно створити відтворити будь-який звук чи мелодію: будильники, таймери, інграшки та ін. На пасивний зумер потрібно додаткового подавати звуковий сигнал, що дозволяє

вибирати точну висоту потрібного звуку. Просто під’єднавши до живлення цей зумер звучати не буде (На відміну від активного зумера).

1.6 RGB світлодіод

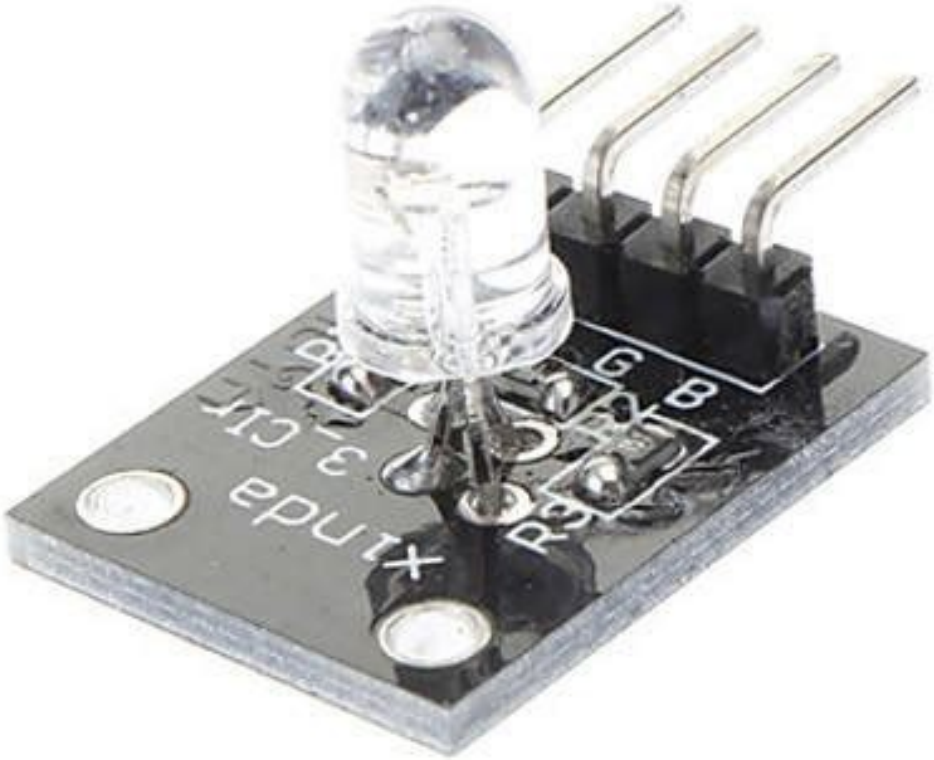

Рис. 6: Вигляд RGB світлодіода

RGB світлодіод поєднує у собі можливість змішувати усі три основні кольори, завдяки чому отримуємо будь який колір з видимого спектру. Модуль має вибудовані 220Ом резистори та один загальний катод, що спрощує під’єднання приладу до макетної плати.

У нашому проекті світлодіод використовується для індикації стану аутентифікації користовача системи.

1.7 Потенціометр

Рис. 7: Вигляд потенціометра з ручкою

Змінний резистор або потенціометр – це електричний пристрій, значення рівня опору якого можна задати у певних межах. Таким чином, ми можемо змінювати параметри електричних схем, гнучко підлаштовуючи їх під певні умови: наприклад, регулювати чутливість датчика або гучність звуку в динаміці. Потенціометри набули широкого поширення в схемах регулювання гучності, напруги, контрастності і т.д., за свою простоту та практичність.

У проекті використано потенціометр номіналом 10КОм. Варто зазначити, що потенціометр використовується у досить незвичному для нього сенсі — для

отримання певного значення на екрані пристрою, для вирішення математичного прикладу чи інше.

1.8 Контактна кнопка

Рис. 8: Вигляд контактної кнопки

Кнопка (або кнопковий перемикач) – найпростіший і найдоступніший з усіх видів датчиків. Натиснувши на неї, контролеру подається сигнал, який потім призводить до якихось дій: вмикаються світлодіоди, видаються звуки, запускаються мотори. У своєму житті ми часто зустрічаємося з різними вимикачами та добре знайомі з цим пристроєм.

Для підключення у будь-якому проекті з кнопкою використовується одночастно резистор номіналом 10кОм, з метою усунення нечіткості контакту. Цей проект не є виключенням, у ньому кнопка використовується для надання підтвердження дій користувача системи.

2 Підключення елементів

2.1 Схематичне з’єднання елементів

У програмному середовищі Kicad зібрано схему підключення елементів проекту. Для створення цієї схеми, було створено новий елемент датчику ID карток, адже цього елементу не було у стандартній бібліотеці Kicad. Усі інші елементи були взяті з стандартної бібліотеки символів.

Через нестачу цифрових виходів для реалізації запланованого функціоналу, було використано можливість плати Arduino UNO використовувати аналогові порти у ролі цифрових. Було використано усі аналогові порти для роботи з пасивним зумером, потенціометром, RGB світлодіодом та кнопки. Схематичне зображення проекту наведене на рис.9.

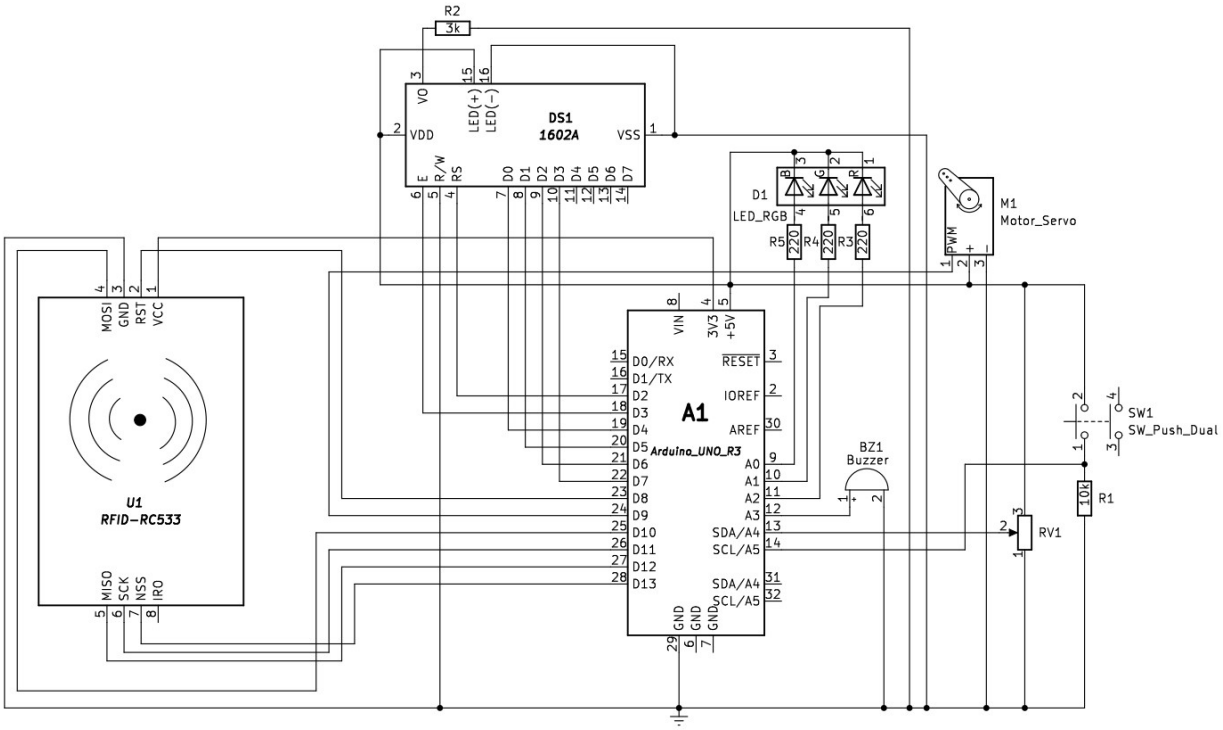

Рис. 9: Схематичне з’єднання проекту

2.2 Підключення елементів на макетній платі

Збірка проекту відбувалася на повнорозмірній макетній платі. Зображення зібраного проекту наведено на рис. 10.

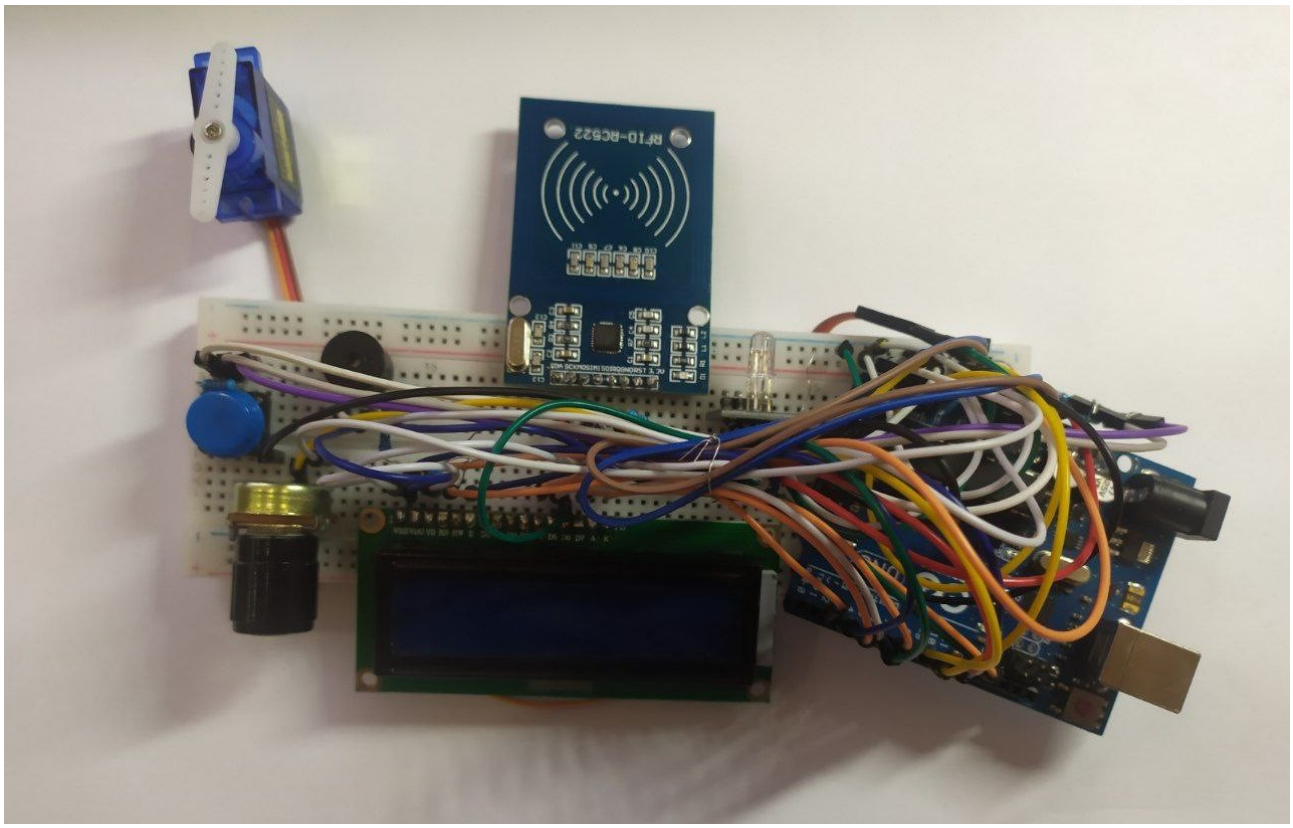

Рис. 10: Проект, зібраний на макетній платі

3 Програмна частина

3.1 Підключення бібліотек

У проекті використовується один заголовний файл — pitches.h, у ньому зберігаються кодові значення нот для мелодій використаних у проекті.

Також у проекті були використані наступні файли.

Файл string.h використовується для забезпечення декількох операцій зі строками копіювання та порівняння.

Бібліотека MFRC522 потрібна для роботи з пристроем зчитування ID карток. LiquidCrystal — використовується для взаємодіх з рідинно кристалічним дисплеем LCD 1602A. Наостанок бібліотека Servo підключена для роботи з сервоприводом.

3.2 Глобальні змінні

У роботі використовується багато глобальних змінних, через специфіку програмування на мові Arduino. Опишемо основні з них. Константи RS, EN, D4-D7 використовуються для визначення пінів для підключення LCD екрану викликом LiquidCrystal lcd(rs, en, d4, d5, d6, d7).

```
const int rs = 2,
      en = 3,
      d4 = 4,
      d5 = 5,
      d6 = 6,
      d7 = 7;
```

У проекті впроваджено функцію додавання карток, для прикладу їх усього 5, але можно збільшити кількість місць для карток. Також створено окрему зміну в одному екзмеплярі названу карткою адміністратора. Ця картка має спеціальні привілегії про які йтиме мова у частині тестування програми.

```
String admin_card = "C0 FE EC 32";
String user_cards[] = {
        "00 00 00 00",
        "00 00 00 00",
        "00 00 00 00",
        "00 00 00 00",
        "00 00 00 00"
};
```

Для музичного супровіду програми впроваджено два цілочисельні масиви для збереження звуків позитивного відгуку та негативного, відповідно: int cheer_sound[] та int alert_sound[].

Останнім іде масив з задачами для простого захисту від дітей.

```
int tasks[10][3] = {
        { 3, 3, 2 },
        { 4, 2, 3 },
        { 1, 5, 3 },
        { 8, 2, 5 },
        { 7, 5, 2 },
        { 2, 9, 1 },
        { 5, 3, 0 },
        { 8, 3, 6 },
        { 9, 5, 6 },
        { 8, 4, 4 }
};
```

Цей масив містить значення які потім будуть прийняті у формі прикладів a+b*c. Наприклад, для першого елементу масиву приклад буде виглядати так: 3+3*2. У відповідь користувач має надати свій результат, якщо він не зійдеться, доступ буде відхилено.

3.3 Опис функції setup()

У цій функції традиційно ініціалізуємо усі необхідні змінні. Для початку починаємо сеанс serial monitor для перегляду деяких значень, у ході перевірки роботи програми.

Далі необхідно запустити інтерфейси для взаємодії з зчитувачем карток, сервоприводом та RGB світлодіодом.

3.4 Опис функції loop()

На мові програмування Arduino, функція loop() призначена для циклічного виконання задач, прописаних у ньому. У проекті до роботи, у цій функції відбуваються наступні дії у такій послідовності.

1. Налаштування змінний на початку кожного циклу;
2. Зчитування значення з ID ключа;
3. Умова перевірки ключа.

Кожний етап займає близько 10 строк коду, завдяки чому цикл є простим і зрозумілим. Згідно основним підходом до процедурного програмування, складна задача розподіляється на багато дрібних підзадач — допоміжних функцій, мова про які йтиметься далі.

3.5 Опис допоміжних функцій

Функції void access_denied() та void access_auth() відповідають за заборону та надання доступу. Програма має функціональність музичного супроводу дій користувача, тому перелічені вище функції викликають відповідні їм мелодії за допомогою void play_alert() та void play_melody(), що грають негативний та позитивний відклік відповідно.

Для переводу сервоприводу у звичайне «закрите» положення, а також для налаштування напису на екрані, після кожної спроби аутентифікації використовується функція void reset_auth();

Але доступ відразу надається тільки по картці адміністратора, утім для звичайних користовачів, доступ отримується тільки у результаті подолання простої перешкоди — арифметичного виразу. Цим займається функція void solve_task(), яка окрім постановки задачі користувачю відповідає за вивід інформації на дисплей та обробку результату користувацької відповіді. Відповідь отримується по натисканню на кнопку.

У випадку використання картки адміністратора, викликається функція void open_admin_settings(), що надає вибір:

- Просто отримати доступ;
- Додати картку;
- Видалити картку.

Якщо для додавання картки — функція void add_card() вже не має вільного місця, програма виведе відповідне повідомлення, у той час як функція видалення картки — void rem_card() завершується завжди успішно.

3.6 Завантаження проекту з використанням USB та програматора

Для завантаження написаного коду проекту до мікроконтролера можна використати інтерфейс USB, це досить невигідний підхід до розподілення ресурсів Arduino, але він найпростіший, а тому широко використовується.

Зазвичай, мікроконтролери Arduino прийнято програмувати за допомогою універсальної шини, як зазначено вище. Але у випадку коли неможливо підключити інтерфейс USB, або його використання зв’язано з втратами у інших параметрах програмованого пристрою.

Програматор наведений на рис.11 використовується для програмування усього сімейства мікроконтролерів AVR, у тому числі й для Arduino, оскільки на платі передбачений спеціальний ISP вивід, для з’єднання з програматором.

Для того, щоб користуватись програматором варто у середовищі Arduino IDE виконати налаштування таким чином, щоб поточний програматор був прийнятий як USBasp, як наведено на рис. 12.

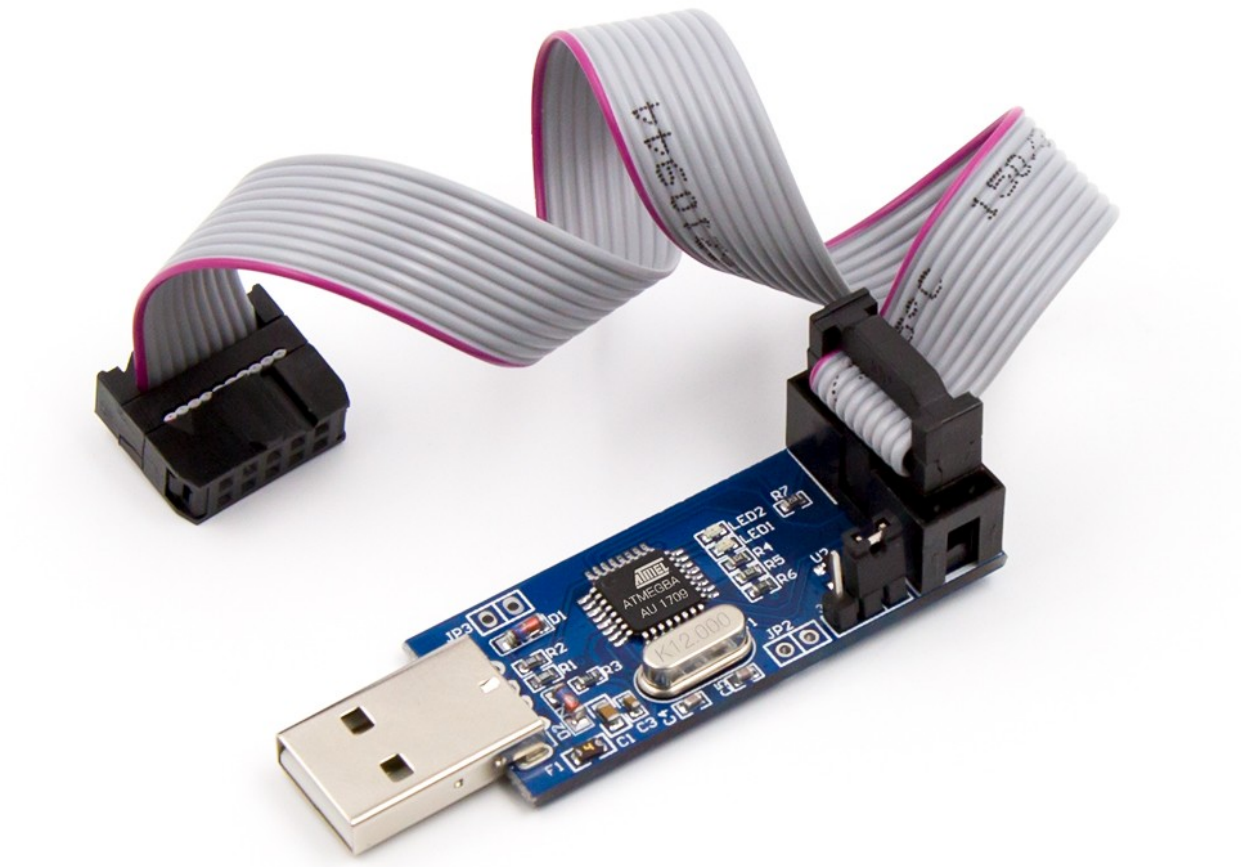

Рис. 11: Зовнішній вигляд програматора USBasp

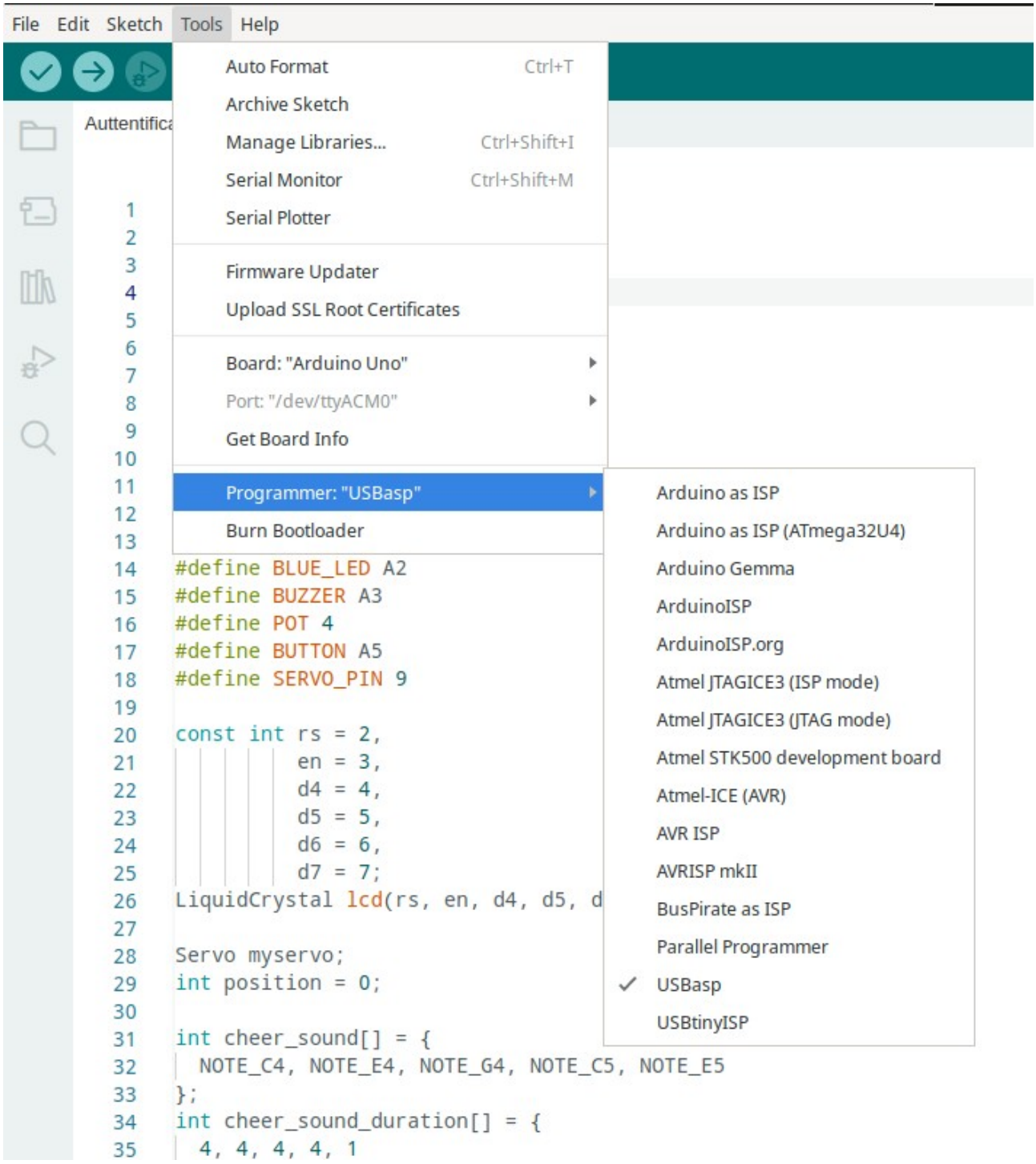

Рис. 12: Налаштування Arduino IDE для використання програматору

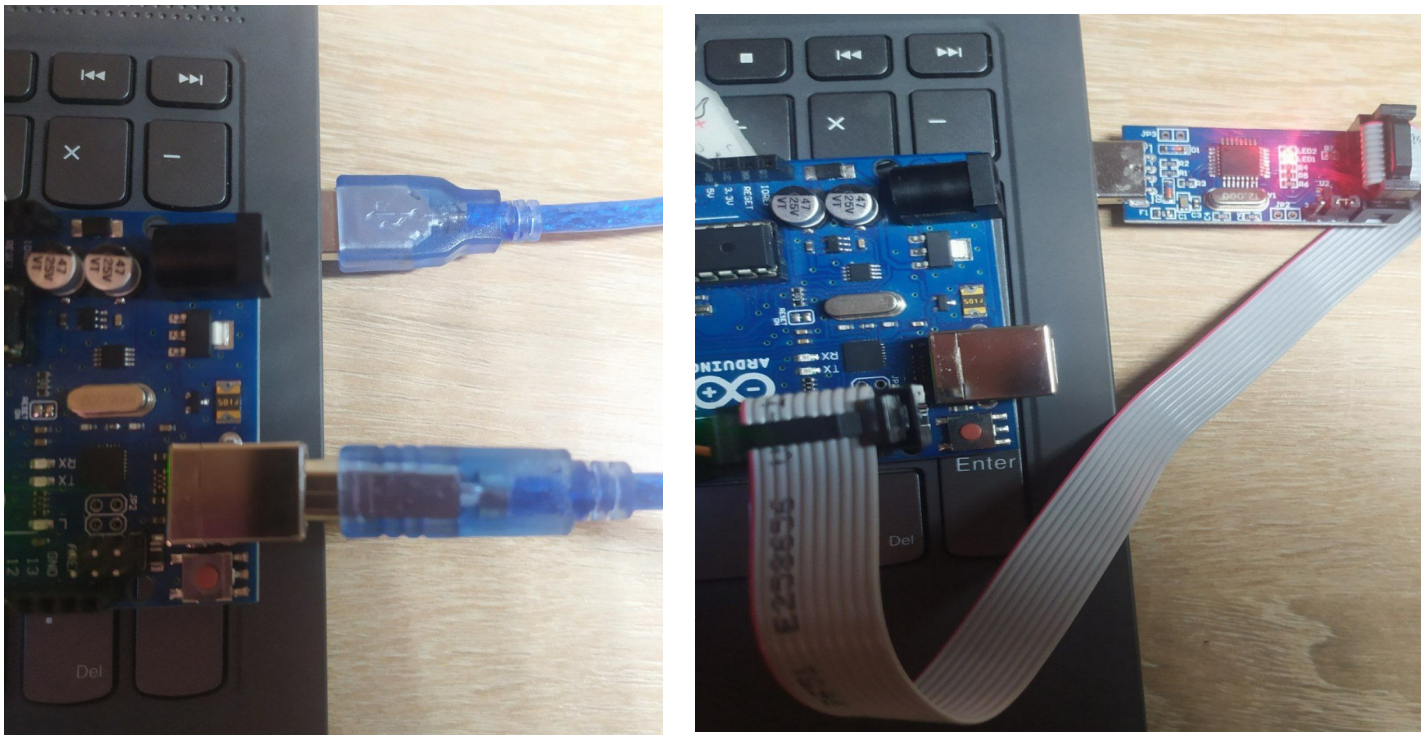

Рис. 13: Підключення плати Arduino за домогою USB та програматора

4 Перевірка роботи

4.1 Перевірка функціоналу користувача

Картка користувача формально може вийти лише на два результати — надання або не надання доступу до системи. Але на шляху до надання доступу можливо не пройти перевірку, а тому, не дивлячись на те, що картка зареєстрована у системі, не отримати доступ до неї. Дії користувача зображені на рис. 14 та рис.15.

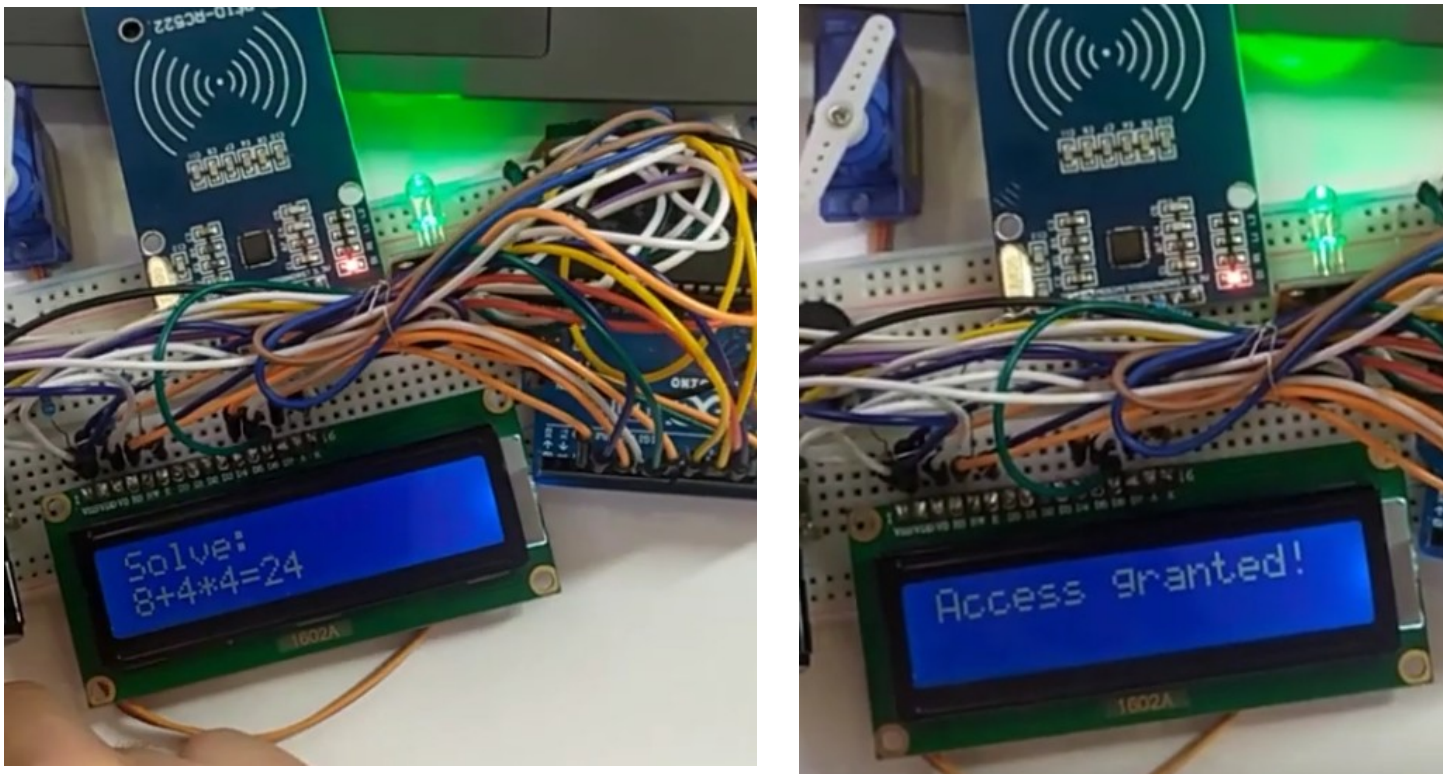

Рис. 14: Дія надання доступу

Рис. 15: Дія не надання доступу

4.2 Первірка функціоналу адміністратора

Адміністратор може отримати доступ у будь-якому разі, не проходячи ніяких додаткових перевірок, відповідно у адміністратора немає дії не надання доступу, але присутній функціонал у вигляді додавання та видалення карток доступу, що зображено на рис 16 та рис.17.

Демонстрацію роботи проекту можна побачити за посиланням: https://www.youtube.com/watch?v=zjn_BsDgOvI

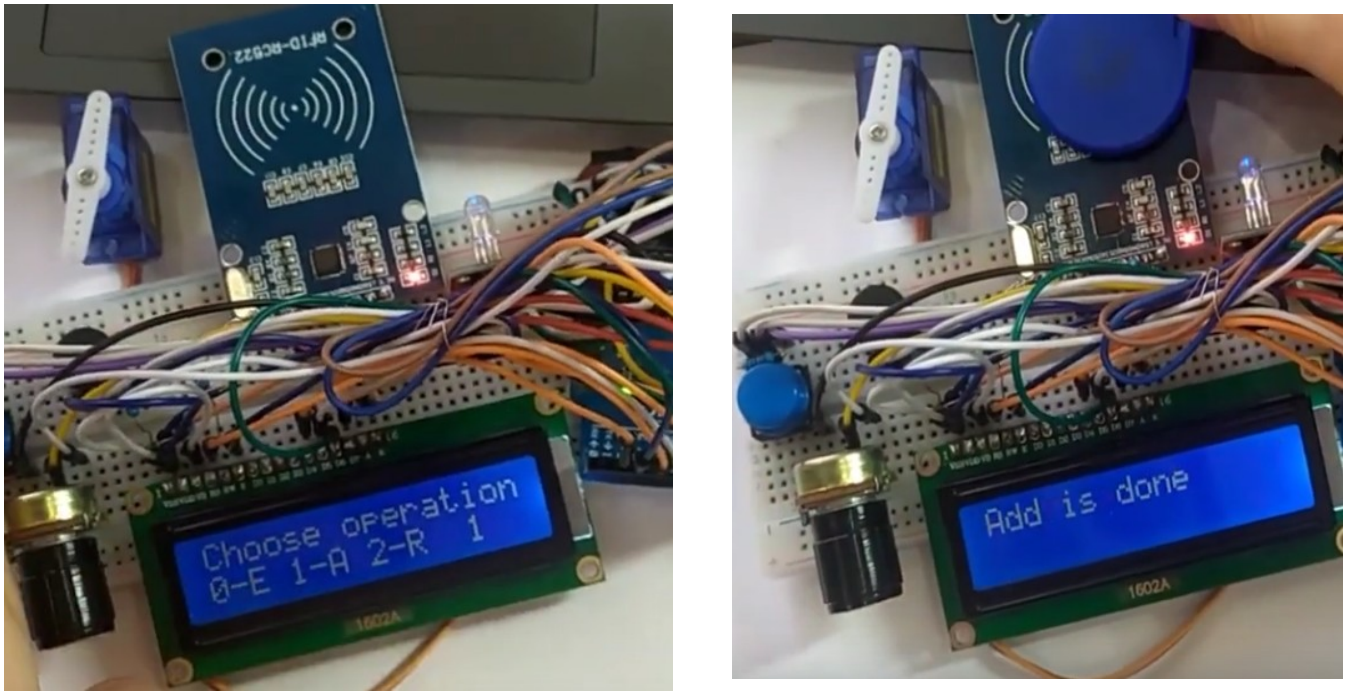

Рис. 16: Дія додавання картки

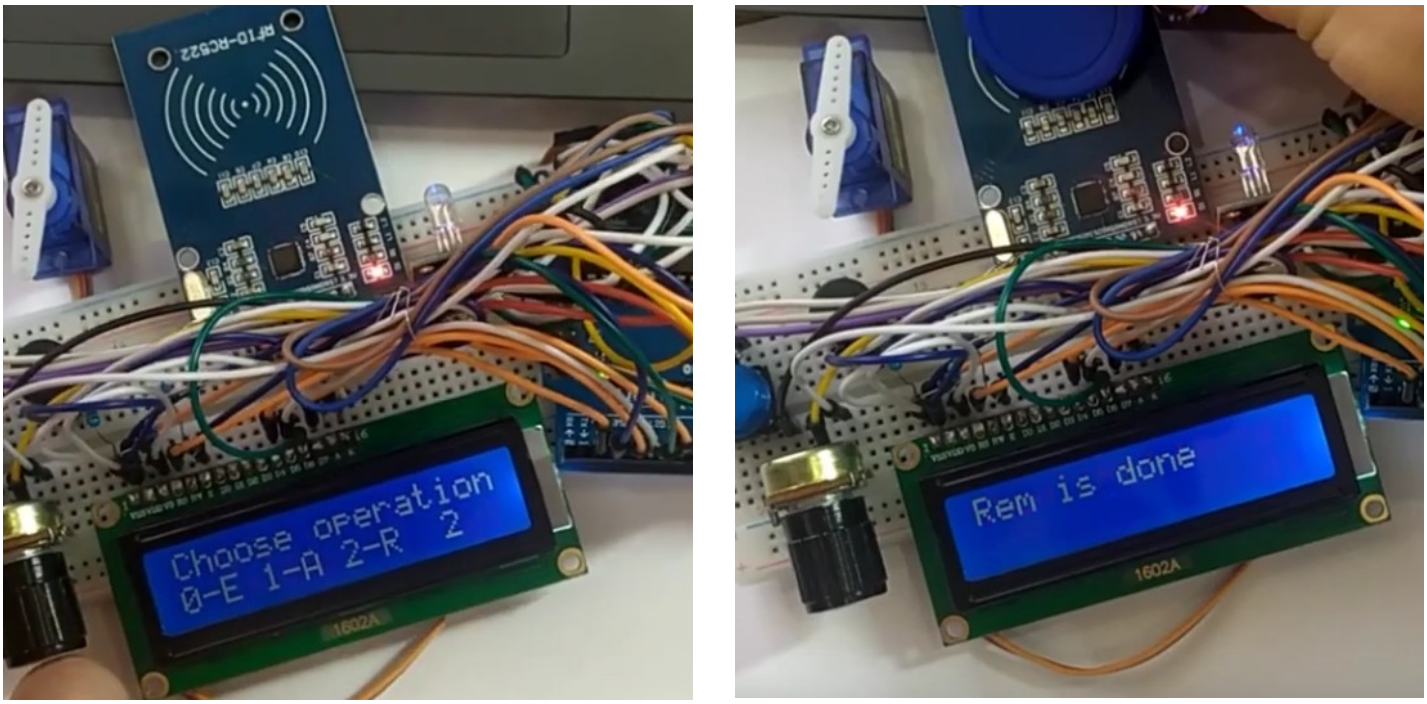

Рис. 17: Дія видалення картки

Висновок

У курсовому проекті було реалізовано пристрій аутентифікації по ID карткам. Було впроваджено основні функції, як для користувача, так і для адміністратора системи. Проект мав на меті підсумувати та систематизувати отримані у ході вивчення дисципліни знання й навички, скомбінувати роботу плати Arduino, та модулів, що розглядалися під час виконання лабораторних робіт.

Програмно є можливість додавати 5 карток-ключів, але при необхідності можна збільшити розміри масиву карток додавши до нього всю вільну пам’ять мікроконтролера.

Для виконання схеми проекту, по якій відбувалося підключення елементів на макетну плату було використано САПР KiCad версії 7.0.

Проект виконаний у макетному вигляді, на рисунках та відео наочно зображені усі з’єднання, що дублюють підключення елементів на схематичному зображенні.

Було написано програмний код для тестування роботи проекту та забезпечення базового функціоналу приладу. Завершальною частиною роботи стало тестування всього проекту — відладка, перевірка роботи функцій програми.

Список використаних джерел

[1] Horowitz, Hill. «The Art of electronics». 3 Ed. Cambridge university press, 2015.

[2] Blum. «Exploring Arduino». 2 Ed. Wiley, year.

[3] Tanenbaum, Austin. «Structured computer organisation» 6 Ed. Pearson Education, 2013.

[4] Sipser. «Introdaction to the theory of computation». 3 Ed. Cengage Learning. 2013.

[5] Schlaepfer, Oskay. «Open circuits». No starch press. 2023.

[6] Scherz, Monk. «Practical electronics for inventors». 4 Ed. Mc Graw Hill. 2018.

[7] Онлайн документація для мови програмування Arduino — https://www.arduino.cc/reference/en/

[8] Онлайн документація проекту KiCad — https://docs.kicad.org/

[9] Сайт для симуляції пристроїв онлайн — https://wokwi.com/

Додаток А

У цьому додатку приводиться повні лістинги коду програми.

Файл pitches.h:

```
#define NOTE_B0  31
#define NOTE_C1  33
#define NOTE_CS1 35
#define NOTE_D1  37
#define NOTE_DS1 39
#define NOTE_E1  41
#define NOTE_F1  44
#define NOTE_FS1 46
#define NOTE_G1  49
#define NOTE_GS1 52
#define NOTE_A1  55
#define NOTE_AS1 58
#define NOTE_B1  62
#define NOTE_C2  65
#define NOTE_CS2 69
#define NOTE_D2  73
#define NOTE_DS2 78
#define NOTE_E2  82
#define NOTE_F2  87
#define NOTE_FS2 93
#define NOTE_G2  98
#define NOTE_GS2 104
#define NOTE_A2  110
#define NOTE_AS2 117
#define NOTE_B2  123
#define NOTE_C3  131
#define NOTE_CS3 139
#define NOTE_D3  147
#define NOTE_DS3 156
#define NOTE_E3  165
#define NOTE_F3  175
#define NOTE_FS3 185
#define NOTE_G3  196
#define NOTE_GS3 208
#define NOTE_A3  220
#define NOTE_AS3 233
#define NOTE_B3  247
#define NOTE_C4  262
#define NOTE_CS4 277
#define NOTE_D4  294
#define NOTE_DS4 311
#define NOTE_E4  330
#define NOTE_F4  349
#define NOTE_FS4 370
#define NOTE_G4  392
#define NOTE_GS4 415
#define NOTE_A4  440
#define NOTE_AS4 466
#define NOTE_B4  494
#define NOTE_C5  523
#define NOTE_CS5 554
#define NOTE_D5  587
```

```
#define NOTE_DS5 622
#define NOTE_E5  659
#define NOTE_F5  698
#define NOTE_FS5 740
#define NOTE_G5  784
#define NOTE_GS5 831
#define NOTE_A5  880
#define NOTE_AS5 932
#define NOTE_B5  988
#define NOTE_C6  1047
#define NOTE_CS6 1109
#define NOTE_D6  1175
#define NOTE_DS6 1245
#define NOTE_E6  1319
#define NOTE_F6  1397
#define NOTE_FS6 1480
#define NOTE_G6  1568
#define NOTE_GS6 1661
#define NOTE_A6  1760
#define NOTE_AS6 1865
#define NOTE_B6  1976
#define NOTE_C7  2093
#define NOTE_CS7 2217
#define NOTE_D7  2349
#define NOTE_DS7 2489
#define NOTE_E7  2637
#define NOTE_F7  2794
#define NOTE_FS7 2960
#define NOTE_G7  3136
#define NOTE_GS7 3322
#define NOTE_A7  3520
#define NOTE_AS7 3729
#define NOTE_B7  3951
#define NOTE_C8  4186
#define NOTE_CS8 4435
#define NOTE_D8  4699
#define NOTE_DS8 4978
```

Файл Auttentification_system.ino:

```
#include "pitches.h"

#include <string.h>
#include <SPI.h>
#include <MFRC522.h>
#include <LiquidCrystal.h>
#include <Servo.h>

#define SS_PIN 10
#define RST_PIN 8
MFRC522 mfrc522(SS_PIN, RST_PIN);
#define GREEN_LED A0
#define RED_LED A1
#define BLUE_LED A2
#define BUZZER A3
#define POT 4
#define BUTTON A5
#define SERVO_PIN 9

const int rs = 2,
      en = 3,
```

```
      d4 = 4,
      d5 = 5,
      d6 = 6,
      d7 = 7;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

Servo myservo;
int position = 0;

int cheer_sound[] = {
        NOTE_C4, NOTE_E4, NOTE_G4, NOTE_C5, NOTE_E5
};
int cheer_sound_duration[] = {
        4, 4, 4, 4, 1
};
int alert_sound[] = {
        NOTE_E5, NOTE_DS5, NOTE_C5, NOTE_G4, NOTE_C4
};
int alert_sound_duration[] = {
        4, 4, 4, 4, 1
};
bool is_auth = false;

String admin_card = "C0 FE EC 32";
String user_cards[] = {
        "00 00 00 00",
        "00 00 00 00",
        "00 00 00 00",
        "00 00 00 00",
        "00 00 00 00"
};

int tasks[10][3] = {
        { 3, 3, 2 },
        { 4, 2, 3 },
        { 1, 5, 3 },
        { 8, 2, 5 },
        { 7, 5, 2 },
        { 2, 9, 1 },
        { 5, 3, 0 },
        { 8, 3, 6 },
        { 9, 5, 6 },
        { 8, 4, 4 }
};
int tasks_size = 10;
int tasks_counter = 0;
int user_cards_size = 5;

void setup() {
        lcd.begin(16, 2);
        Serial.begin(9600);
        SPI.begin();
        mfrc522.PCD_Init();
        Serial.println("Approximate your card to the reader...");
        Serial.println();
        lcd.print("Entry by cards");
        myservo.attach(SERVO_PIN);
        myservo.write(0);
        pinMode(RED_LED, OUTPUT);
        pinMode(GREEN_LED, OUTPUT);
        pinMode(BUZZER, OUTPUT);
```

```
        pinMode(BUTTON, INPUT);
}

void loop() {
        is_auth = false;
        tasks_counter++;
        if (tasks_counter >= tasks_size)
                tasks_counter = 0;
        digitalWrite(BLUE_LED, HIGH);
        digitalWrite(GREEN_LED, LOW);
        digitalWrite(RED_LED, LOW);
        if (!mfrc522.PICC_IsNewCardPresent()) {
                return;
        }
        if (!mfrc522.PICC_ReadCardSerial()) {
                return;
        }
        Serial.print("UID tag :");
        String content = "";
        byte letter;
        for (byte i = 0; i < mfrc522.uid.size; i++) {
                Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
                Serial.print(mfrc522.uid.uidByte[i], HEX);
                content.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : "
"));
                content.concat(String(mfrc522.uid.uidByte[i], HEX));
        }
        Serial.println();
        Serial.print("Message : ");
        content.toUpperCase();
        for (int i = 0; i < user_cards_size; i++) {
                if (content.substring(1) == user_cards[i]) {
                        solve_task();
                        is_auth = true;
                }
        }
        if (content.substring(1) == admin_card)
                open_admin_settings();
        if (!is_auth)
                access_denied();
}

void access_denied() {
        digitalWrite(BLUE_LED, LOW);
        digitalWrite(RED_LED, HIGH);
        Serial.println("Access denied");
        lcd.clear();
        lcd.print("Access denied!");
        play_alert();
        reset_auth();
}

void access_auth() {
        digitalWrite(BLUE_LED, LOW);
        digitalWrite(GREEN_LED, HIGH);
        Serial.println("Access granted!");
        lcd.clear();
        lcd.print("Access granted!");
        myservo.write(180);
        play_melody();
        reset_auth();
```

```
}

void reset_auth() {
        delay(3000);
        myservo.write(0);
        lcd.clear();
        lcd.print("Waiting for key!");
}

void play_alert() {
        for (int thisNote = 0; thisNote < (sizeof(alert_sound) / sizeof(int));
thisNote++) {

                int noteDuration = 1000 / alert_sound_duration[thisNote];
                tone(BUZZER, alert_sound[thisNote], noteDuration);

                int pauseBetweenNotes = noteDuration * 1;
                delay(pauseBetweenNotes);
                noTone(BUZZER);
        }
}

void play_melody() {
        for (int thisNote = 0; thisNote < (sizeof(cheer_sound) / sizeof(int));
thisNote++) {

                int noteDuration = 1000 / cheer_sound_duration[thisNote];
                tone(BUZZER, cheer_sound[thisNote], noteDuration);

                int pauseBetweenNotes = noteDuration * 1;
                delay(pauseBetweenNotes);
                noTone(BUZZER);
        }
}

void solve_task() {
        lcd.clear();
        int task_result = tasks[tasks_counter][0] + tasks[tasks_counter][1] *
tasks[tasks_counter][2];
        int user_result = 0;
        char task_string[5];
        task_string[0] = '0'+tasks[tasks_counter][0];
        task_string[1] = '+';
        task_string[2] = '0'+tasks[tasks_counter][1];
        task_string[3] = '*';
        task_string[4] = '0'+tasks[tasks_counter][2];
        task_string[5] = '\0';
        int button_state = digitalRead(BUTTON);
        while (button_state == LOW) {
                Serial.print(button_state);
                lcd.clear();
                user_result = analogRead(POT) / 20;
                button_state = digitalRead(BUTTON);
                lcd.print("Solve: ");
                lcd.setCursor(0, 2);
                lcd.print(task_string);
                lcd.print('=');
                lcd.setCursor(6, 2);
                lcd.print(user_result);
                delay(100);
        }
```

```
        if (user_result == task_result) {
                access_auth();
        } else {
                access_denied();
        }
}

void open_admin_settings() {
        int operation = 0;
        int button_state = digitalRead(BUTTON);
        while (button_state == LOW) {
                lcd.clear();
                lcd.print("Choose operation: ");
                lcd.setCursor(0, 2);
                lcd.print("0-E 1-A 2-R");
                lcd.setCursor(13, 2);
                operation = analogRead(POT) / 500;
                button_state = digitalRead(BUTTON);
                lcd.print(operation);
                delay(100);
        }
        switch (operation) {
                case 0:
                        access_auth();
                        is_auth = true;
                        return;
                case 1:
                        lcd.clear();
                        lcd.print("Show card to add");
                        delay(3000);
                        add_card();
                        break;
                case 2:
                        lcd.clear();
                        lcd.print("Show card to rem");
                        delay(3000);
                        rem_card();
                        break;
                default:
                        lcd.clear();
                        lcd.print("Invalid operation");
                        delay(3000);
                        reset_auth();
        }
}

void add_card() {
        lcd.clear();
        if (!mfrc522.PICC_IsNewCardPresent()) {
                return;
        }
        if (!mfrc522.PICC_ReadCardSerial()) {
                return;
        }
        Serial.print("UID tag :");
        String content = "";
        byte letter;
        for (byte i = 0; i < mfrc522.uid.size; i++) {
                Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
                Serial.print(mfrc522.uid.uidByte[i], HEX);
```

```
                content.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : "
"));
                content.concat(String(mfrc522.uid.uidByte[i], HEX));
        }
        Serial.println();
        Serial.print("Message : ");
        String input_card = content.substring(1);
        content.toUpperCase();
        int i;
        for (i = 0; i < user_cards_size; i++) {
                if (input_card != "00 00 00 00") {
                        user_cards[i] = content.substring(1);
                        lcd.print("Add is done");
                        delay(3000);
                        is_auth = true;
                        reset_auth();
                        return;
                }
                if (input_card == admin_card) {
                        open_admin_settings();
                        return;
                }
        }
        lcd.clear();
        lcd.print("No enough space");
        delay(3000);
        reset_auth();
}

void rem_card() {
        lcd.clear();
        if (!mfrc522.PICC_IsNewCardPresent()) {
                return;
        }
        if (!mfrc522.PICC_ReadCardSerial()) {
                return;
        }
        Serial.print("UID tag :");
        String content = "";
        byte letter;
        for (byte i = 0; i < mfrc522.uid.size; i++) {
                Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
                Serial.print(mfrc522.uid.uidByte[i], HEX);
                content.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : "
"));
                content.concat(String(mfrc522.uid.uidByte[i], HEX));
        }
        Serial.println();
        Serial.print("Message : ");
        content.toUpperCase();
        String card_to_remove = content.substring(1);
        for (int i = 0; i < user_cards_size; i++) {
                if (user_cards[i] == card_to_remove) {
                        user_cards[i] = "00 00 00 00";
                        lcd.print("Rem is done");
                        delay(3000);
                        is_auth = true;
                        reset_auth();
                        return;
                }
        }
```

```
        lcd.clear();
        lcd.print("No such card");
        delay(3000);
        reset_auth();
}
```