

스마트 도어락 시스템

김영인 심예영 윤혜진 최현서



국립한경대학교
HANKYONG NATIONAL UNIV.

제출일	2024. 6. 17	전공	융합소프트웨어
과목	IoT프로그래밍	조이름	6조
담당교수	신정호	이름	김영인 심예영 윤혜진 최현서

[목차]

내용

1. 프로젝트 개요.....	3
2. 변경사항.....	3
3. 주요 기능.....	4
4. 사용 설명.....	5
5. 코드 상세 설명.....	6
6. 실행 결과.....	14
7. 결론 및 참고문헌.....	18

1. 프로젝트 개요

본 프로젝트는 스마트도어락 시스템을 설계 및 구현하는 것을 목표로 합니다. 이 시스템은 사용자 인증을 위해 키패드를 사용하고, 성공 및 실패 사운드 신호를 재생하며, 일정 횟수 이상의 실패 시 도어락을 잠그는 잠금 메커니즘을 제공합니다.

2. 변경사항

기존의 방식은 초음파 센서를 활용한 주변 감지 기능이 있었지만 드라이버의 문제로 해당 기능을 제거하기로 결정했습니다.

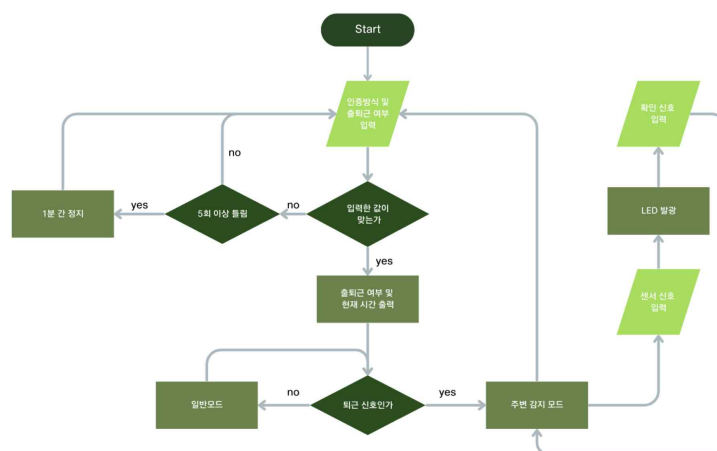


그림 변경 전 Flow Chart

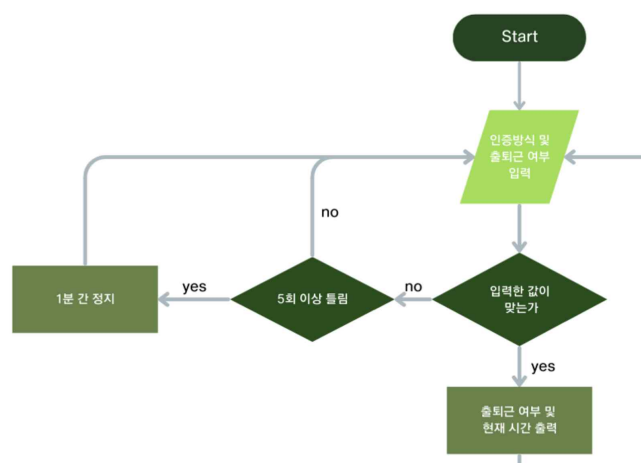


그림 변경 후 Flow Chart

대신에 일반적으로 사용되는 도어락 소리를 추가하여 사용자 경험을 향상시켰습니다. 소리 효과는 다음과 같은 기능을 합니다.

잠금 해제 확인 : 도어락이 해제될 때 사운드를 통해 사용자에게 확인 신호를 제공합니다.

오류 알림 : 오류 발생 시 경고음을 통해 사용자에게 알려줍니다.

3. 주요 기능

사용자 인증 : 사용자는 키패드를 통해 4자리 비밀번호를 입력하여 도어락을 해제할 수 있습니다.

사운드 알림 : 비밀번호가 성공적으로 입력되면 성공 사운드가 재생되고, 5회 이상 틀렸을 시에는 경고음이 재생됩니다.

잠금 메커니즘 : 5번 이상 비밀번호를 잘못 입력하면 60초 동안 도어락이 잠깁니다.

대기 화면 : 사용자 입력이 없으면 "Standby screen" 메시지가 CLCD 디스플레이에 표시됩니다.

현재 시각 출력 : 잠금/해제 시에 현재 시각이 출력됩니다.

4. 사용 설명



그림 키패드 모델

텍스트 스위치에 위 키패드 모델을 대입하여 수행합니다.

지정된 비밀번호를 입력하고 *를 누르면 출근, #를 누르면 퇴근으로 처리됩니다.

누른 숫자 문자열의 마지막 네글자가 지정된 비밀번호면 잠금이 해제됩니다.

5. 코드 상세 설명

5.1 SecurityKey 클래스

5.1.1 변수명

```
42 class SecurityKey {
43 private:
44     char storedPassword[5]; // 저장된 비밀번호 (최대 4자리 + null 종단 문자)
45     vector<char> inputPassword; // 입력된 비밀번호
46     int attemptCount; // 비밀번호 시도 횟수
47     int inputCount; // 입력 시도 횟수
48     bool error; // 에러 플래그
49     bool lockout; // 잠금 플래그
50     bool inputReceived; // 입력 수신 플래그
51     time_t lockoutStartTime; // 잠금 시작 시간
52     time_t lastInputTime; // 마지막 입력 시간
53
54 public:
55     SecurityKey() {
56         strcpy(storedPassword, "1234");
57         attemptCount = 0;
58         inputCount = 0;
59         error = false;
60         lockout = false;
61         inputReceived = false;
62         lastInputTime = time(NULL);
63         std::cout << "SecurityKey initialized." << std::endl;
64     }
65 }
```

storedPassword : 도어락 해제를 위한 올바른 비밀번호를 보관합니다.

inputPassword : 사용자가 키패드를 누르면 입력한 비밀번호를 저장합니다.

attemptCount : 잘못된 비밀번호 시도 횟수를 추적합니다.

inputCount : 현재 시도에서 입력된 숫자의 개수를 추적합니다.

error : 오류가 발생했는지 여부를 나타냅니다.

lockout : 실패 시도로 인해 도어락이 잠겨 있는지 여부를 나타냅니다.

inputReceived : 사용자가 현재 시도에서 숫자를 입력했는지 여부를 나타냅니다.

lockoutStartTime : 잠금이 시작된 시간을 기록합니다.

lastInputTime : 마지막 키패드 누름 시간을 기록합니다.

5.1.2 메서드

```
54  ∨ public:
55  ∨     SecurityKey() {
56      |         strcpy(storedPassword, "1234");
57      |         attemptCount = 0;
58      |         inputCount = 0;
59      |         error = false;
60      |         lockout = false;
61      |         inputReceived = false;
62      |         lastInputTime = time(NULL);
63      |         std::cout << "SecurityKey initialized." << std::endl;
64      |     }
65
66  ∨     bool getError() const {
67      |         return error;
68      |     }
69
70  ∨     void setError(bool value) {
71      |         error = value;
72      |     }
73
74  ∨     void resetInput() {
75      |         inputPassword.clear();
76      |         inputReceived = false;
77      |     }
78
79  ∨     void resetInputCount() {
80      |         inputCount = 0;
81      |     }
82
83  ∨     void resetAttemptCount() {
84      |         attemptCount = 0;
85      |     }
86
```

```

87     void displayMessage(const string& message, int duration) {
88         writeCLCD(message);
89         printf("CLCD Output: %s\n", message.c_str());
90         sleep(duration);
91         writeCLCD("Standby screen");
92         printf("CLCD Output: Standby screen\n");
93     }
94
95     bool isPasswordCorrect() {
96         if (inputPassword.size() < 4) return false;
97         string enteredPassword(inputPassword.end() - 4, inputPassword.end());
98         return enteredPassword == storedPassword;
99     }
100
101     void checkPassword(char modeChar) {
102         std::cout << "Checking password..." << std::endl;
103         if (isPasswordCorrect()) {
104             time_t now = time(0);
105             tm *ltm = localtime(&now);
106             char timeStr[20];
107             sprintf(timeStr, "%02d:%02d:%02d", ltm->tm_hour, ltm->tm_min, ltm->tm_sec);
108
109             std::string modeStr = (modeChar == '*') ? "go to work" : "leave work";
110             std::string fullMessage = std::string(timeStr) + "\n" + modeStr;
111
112             writeCLCD(fullMessage);
113             printf("CLCD Output: %s\n", fullMessage.c_str());
114             playSound(OPEN_SOUND_FILE); // 비밀번호가 맞으면 성공 사운드 재생
115             sleep(DISPLAY_DURATION); // 메시지 5초간 표시
116             writeCLCD("Standby screen");
117             printf("CLCD Output: Standby screen\n");
118             resetInputCount(); // 비밀번호 맞추면 입력 count 초기화
119             resetAttemptCount(); // 비밀번호 맞추면 시도 횟수 초기화
120         } else {
121             attemptCount++;
122             if (attemptCount >= 5) {
123                 std::string message = "Incorrect password. Alarm triggered!";
124                 std::cout << message << std::endl;
125                 playSound(ALARM_SOUND_FILE); // 비밀번호 5번 실패 시 실패 사운드 재생
126                 displayMessage(message, ALARM_DURATION); // 알람 메시지 10초간 표시
127                 lockout = true;
128                 lockoutStartTime = time(NULL);
129                 writeCLCD("Lockout: 60 seconds");
130                 printf("CLCD Output: Lockout: 60 seconds\n");
131                 sleep(1); // 1초 대기 후 다시 확인
132                 return;
133             } else {
134                 std::string message = "Incorrect password.";
135                 std::cout << message << std::endl;
136                 displayMessage(message, DISPLAY_DURATION);
137             }
138         }
139         resetInput(); // 입력 초기화
140     }

```



```
142     char getKeypadInput() {
143         if (lockout) {
144             // 잠금 상태에서 남은 시간을 CLCD에 표시
145             time_t currentTime = time(NULL);
146             int elapsed = difftime(currentTime, lockoutStartTime);
147             int remaining = LOCKOUT_DURATION - elapsed;
148
149             if (remaining <= 0) {
150                 lockout = false;
151                 resetAttemptCount(); // 입력 정지 상태가 끝나면 입력 실패 카운트 초기화
152                 writeCLCD("Standby screen");
153                 printf("CLCD Output: Standby screen\n");
154                 resetInput(); // 입력 초기화
155                 return '\0';
156             } else {
157                 char lockoutMessage[32];
158                 sprintf(lockoutMessage, "Lockout: %d seconds", remaining);
159                 writeCLCD(lockoutMessage);
160                 printf("CLCD Output: %s\n", lockoutMessage);
161                 sleep(1); // 1초 대기 후 다시 확인
162                 return '\0';
163             }
164         }
165
166         unsigned char b;
167         int tactswFd = open(TACT, O_RDONLY);
168         if (tactswFd < 0) {
169             std::perror("tact device error");
170             return '\0';
171         }
172
173         if (read(tactswFd, &b, sizeof(b)) < 0) {
174             std::perror("read error");
175             close(tactswFd);
176             return '\0';
177         }
178
179         close(tactswFd);
180
181         switch (b) {
182             case 1: return '1';
183             case 2: return '2';
184             case 3: return '3';
185             case 4: return '4';
186             case 5: return '5';
187             case 6: return '6';
188             case 7: return '7';
189             case 8: return '8';
190             case 9: return '9';
191             case 10: return '*';
192             case 11: return '0';
193             case 12: return '#';
194             default: return '\0';
195         }
196     }
```

getError() : 오류 플래그 값을 반환합니다.

setError(bool value) : 오류 플래그를 지정된 값으로 설정합니다.

resetInput() : 입력 비밀번호를 지우고 관련 플래그를 재설정합니다.

resetInputCount() : 입력 횟수를 0으로 재설정합니다.

resetAttemptCount() : 시도 횟수를 0으로 재설정합니다.

displayMessage(const string& message, int duration) : 지정된 시간 동안 CLCD에 지정된 메시지를 표시합니다.

isPasswordCorrect() : 입력된 비밀번호가 저장된 비밀번호와 일치하는지 확인합니다.

checkPassword(char modeChar) : 입력된 비밀번호를 유효성 검사하고, 사운드 신호를 재생하며, 잠금 상태를 업데이트합니다.

getKeypadInput() : 키패드로부터 입력을 읽고 해당 숫자 또는 심볼을 반환합니다.

handleKeyPress() : 사용자 입력, 키패드 누름 및 잠금 확인을 포함하여 사용자 입력을 처리합니다.

5.2 메인 함수

```
248 int main() {
249     std::cout << "Security Key Program Started" << std::endl;
250     SecurityKey key;
251
252     writeCLCD("Standby screen");
253     printf("CLCD Output: Standby screen\n");
254
255     while (true) {
256         key.handleKeyPress();
257         if (key.getError()) {
258             std::cout << "Error occurred. Resetting..." << std::endl;
259             key.setError(false);
260             sleep(1); // 간단한 지연
261         } else {
262             usleep(100000); // 반복 주기 조정
263         }
264     }
265
266     return 0;
267 }
```

- SecurityKey클래스의 인스턴스를 생성합니다.
- CLCD 디스플레이를 "Standby screen" 메시지로 초기화합니다.
- SecurityKey객체의 handleKeyPress()메서드를 지속적으로 호출하는 무한 루프에 들어가 사용자 입력을 처리하고 도어락 상태를 관리합니다.

5.3 handleKeyPress() 메서드 상세 설명

```

198 void handleKeyPress() {
199     char keyInput = getKeyPadInput();
200     if (keyInput) {
201         inputReceived = true;
202         lastInputTime = time(NULL); // 입력 시간 업데이트
203         std::cout << "Handling key press..." << std::endl;
204         printf("Key pressed: %c\n", keyInput);
205         if (keyInput == '*' || keyInput == '#') {
206             if (inputPassword.size() >= 4) {
207                 checkPassword(keyInput);
208             } else {
209                 std::string message = "Please enter a password";
210                 std::cout << message << std::endl;
211                 displayMessage(message, DISPLAY_DURATION);
212                 attemptCount++; // 실패 카운트 증가
213                 if (attemptCount >= 5) {
214                     lockout = true;
215                     lockoutStartTime = time(NULL);
216                     std::string lockoutMessage = "Too many attempts. Locking out.";
217                     std::cout << lockoutMessage << std::endl;
218                     playSound(ALARM_SOUND_FILE); // 비밀번호 5번 실패 시 실패 사운드 재생
219                     writeCLCD(lockoutMessage); // 잠금 메시지 표시
220                     printf("CLCD Output: %s\n", lockoutMessage.c_str());
221                     return;
222                 }
223             }
224             resetInput(); // 입력 초기화
225         } else {
226             inputPassword.push_back(keyInput);
227             printf("Current input: ");
228             for (size_t i = 0; i < inputPassword.size(); ++i) {
229                 printf("%c", inputPassword[i]);
230             }
231             printf("\n");
232             writeCLCD("Inputting...");
233         }
234     }
235 } else if (!lockout) { // 잠금 상태가 아닌 경우에만 대기 화면 출력
236     // 입력이 없으면 대기화면 출력
237     time_t currentTime = time(NULL);
238     if (!inputReceived && difftime(currentTime, lastInputTime) >= WAITING_SCREEN_INTERVAL) {
239         std::string message = "Standby screen";
240         writeCLCD(message);
241         printf("CLCD Output: %s\n", message.c_str());
242         lastInputTime = currentTime; // 대기 화면 출력 시간 업데이트
243     }
244 }
245 }
246 };

```

- 잠금 확인

- lockout플래그가 설정된 경우 남은 잠금 시간을 CLCD에 표시합니다.
- 잠금이 만료된 경우 lockout및 attemptCount플래그를 재설정하고, 입력 비밀번호를 지우고, "Standby screen"으로 돌아갑니다.

- 키패드 입력 읽기

- getKeypadInput()메서드를 호출하여 키패드로부터 입력을 읽습니다.

- 유효한 키가 누르진 경우

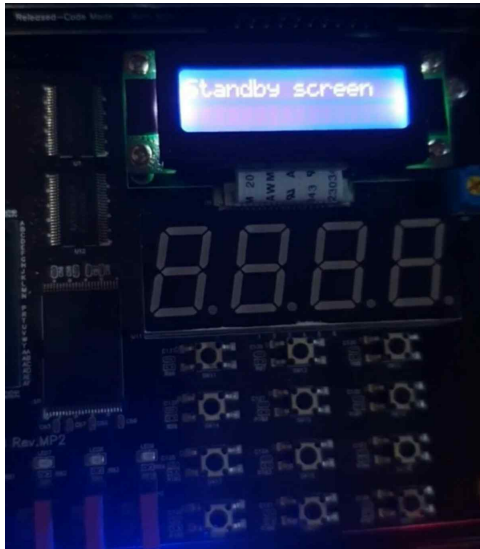
- inputReceived플래그를 true로 설정합니다.
- lastInputTime을 현재 시간으로 업데이트합니다.
- 누른 키를 콘솔에 출력합니다.

- 비밀번호 입력 처리:

- 누른 키가 * 또는 #인 경우:
- 입력 비밀번호가 4자리 이상인 경우 checkPassword()를 호출하여 비밀번호를 유효성 검사합니다.
- 입력 비밀번호가 4자리 미만인 경우 "Enter password" 메시지를 표시하고 attemptCount를 증가시킵니다

6. 실행 결과

1) 초기 화면



프로그램을 실행시키면 CLCD에 **Standby screen** 문구가 출력되고 아래 텍스트 스위치로 입력을 받습니다.

2) 입력 중



입력을 시작하면 **Inputting...**이 출력됩니다

3) 비밀번호가 틀린 경우



Incorrect password가 출력됩니다.

4) 입력된 문자열의 뒷 4자리가 지정된 비밀번호이고, *을 누른 경우



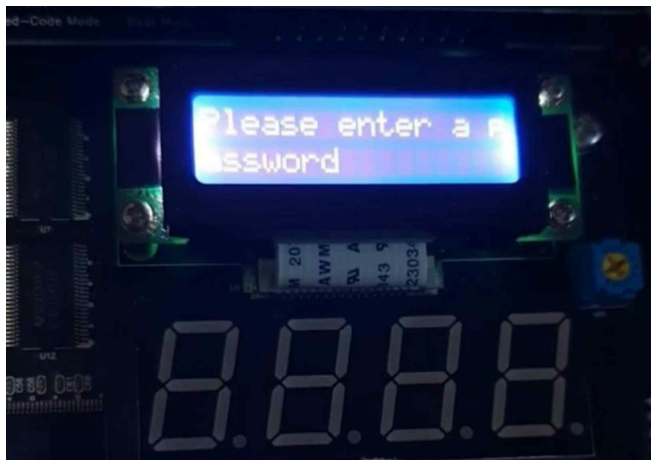
잠금이 해제되어 현재 시각과 go to work 문구가 출력되고 소리가 나옵니다.

5) 입력된 문자열의 뒷 4자리가 지정된 비밀번호이고, #을 누른 경우



잠금이 되고 현재 시각과 leave work 문구가 출력되며 소리가 나옵니다.

6) 4자리 이상 비밀번호를 누르지 않고, * 또는 #을 누른 경우



Please enter a password가 출력됩니다

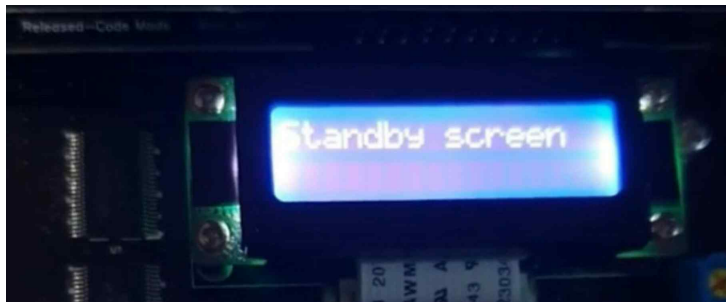
7) 5번 이상 비밀번호를 틀린 경우



Too many attempt. Locking out. 문구가 출력됩니다.



이후 60초 카운트다운이 시작됩니다.



카운트다운이 끝나면 초기화면으로 돌아옵니다.

7. 결론 및 참고문헌

7.1 결론

본 프로젝트를 통해 스마트도어락 시스템을 성공적으로 설계 및 구현했습니다. 이 시스템은 사용자 편의성과 보안성을 모두 고려하여 설계되었으며, 다양한 기능을 제공합니다.

7.2 참고문헌

깃허브 : <https://github.com/lold2424/IOT-Team-Project>

7.2.1 ChatGPT



일정 거리 이하로 한번이라도 들어오면 확인 신호 받기 전까지 led 켜지게 해줘

Smart4412 환경에서 일정 거리 이하로 물체가 감지되면 확인 신호를 받기 전까지 LED가 계속 켜지는 C 언어 코드를 작성해 보겠습니다. 이 경우 확인 신호는 버튼 입력으로 받도록 하겠습니다.

필요한 라이브러리 및 설정

먼저 필요한 라이브러리와 핀 번호를 설정합니다. 예제에서는 GPIO 핀 번호가 가상적이며 실제 하드웨어에 따라 변경해야 할 수 있습니다.

c

코드 복사

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <time.h>

#define TRIG_PIN 67 // 초음파 센서의 Trig 핀
#define ECHO_PIN 68 // 초음파 센서의 Echo 핀
#define LED_PIN 69 // LED 핀
#define BUTTON_PIN 70 // 확인 신호 버튼 핀
```

```
void gpio_export(int pin) {
    char buffer[64];
    int len;
    int fd = open("/sys/class/gpio/export", O_WRONLY);
    if (fd < 0) {
        perror("gpio/export");
        return;
    }
    len = snprintf(buffer, sizeof(buffer), "%d", pin);
    write(fd, buffer, len);
    close(fd);
}

void gpio_unexport(int pin) {
    char buffer[64];
    int len;
    int fd = open("/sys/class/gpio/unexport", O_WRONLY);
    if (fd < 0) {
        perror("gpio/unexport");
        return;
    }
    len = snprintf(buffer, sizeof(buffer), "%d", pin);
    write(fd, buffer, len);
}
```

7.2.1 기타 문헌

사운드 관련 자료 : <https://en.cppreference.com/w/cpp/utility/apply>

시간 관련 자료 : <https://en.cppreference.com/w/cpp/chrono/c/time>

경보음 : <https://gongu.copyright.or.kr/gongu/wrt/wrt/view.do?wrtSn=13252821&menuNo=200020>

잠금 해제 소리 :

<https://gongu.copyright.or.kr/gongu/wrt/wrt/view.do?wrtSn=13242733&menuNo=200020>