1111111111111111111111111111

#include "mbed.h"

#include "arm\_book\_lib.h"

// === Alarm LED ===

DigitalOut alarmLed(D10);  // External LED for alarm status

// === 4x4 Keypad Wiring ===

// Rows (outputs)

DigitalOut row1(D2);

DigitalOut row2(D3);

DigitalOut row3(D4);

DigitalOut row4(D5);

// Columns (inputs)

DigitalIn col1(D6);

DigitalIn col2(D7);

DigitalIn col3(D8);

DigitalIn col4(D9);

// Keypad layout

const char KEYS[4][4] = {

    {'1', '2', '3', 'A'},

    {'4', '5', '6', 'B'},

    {'7', '8', '9', 'C'},

    {'\*', '0', '#', 'D'}

};

// Matrix helpers

DigitalOut\* rows[4] = { &row1, &row2, &row3, &row4 };

DigitalIn\* cols[4] = { &col1, &col2, &col3, &col4 };

// Deactivation code

const char correctCode[4] = { '1', '2', '3', '4' };

char enteredCode[4];

int codeIndex = 0;

char scanKeypad() {

    for (int r = 0; r < 4; r++) {

        // Set all rows HIGH, then set current row LOW

        for (int i = 0; i < 4; i++) \*rows[i] = 1;

        \*rows[r] = 0;

        for (int c = 0; c < 4; c++) {

            if (\*cols[c] == 0) {

                ThisThread::sleep\_for(200ms);  // Debounce

                return KEYS[r][c];

            }

        }

    }

    return '\0';  // No key pressed

}

int main() {

    // Set pull-ups on columns

    for (int i = 0; i < 4; i++) {

        cols[i]->mode(PullUp);

    }

    // Alarm initially ON

    alarmLed = 1;

    while (true) {

        char key = scanKeypad();

        if (key != '\0') {

            enteredCode[codeIndex++] = key;

            if (codeIndex == 4) {

                bool correct = true;

                for (int i = 0; i < 4; i++) {

                    if (enteredCode[i] != correctCode[i]) {

                        correct = false;

                        break;

                    }

                }

                if (correct) {

                    alarmLed = 0;  // Turn off alarm

                } else {

                    alarmLed = 1;  // Keep alarm on

                }

                codeIndex = 0;  // Reset for next try

            }

        }

        ThisThread::sleep\_for(100ms);

    }

}

2222222222222222222222222222222222222222222

#include "mbed.h"

#include "arm\_book\_lib.h"

// Alarm LED

DigitalOut alarmLed(D10);

// Keypad rows (outputs)

DigitalOut row1(D2);

DigitalOut row2(D3);

DigitalOut row3(D4);

DigitalOut row4(D5);

// Keypad columns (inputs)

DigitalIn col1(D6);

DigitalIn col2(D7);

DigitalIn col3(D8);

DigitalIn col4(D9);

// Serial setup

BufferedSerial pc(USBTX, USBRX, 9600);

char buffer[64];

// Keypad layout

const char KEYS[4][4] = {

    {'1', '2', '3', 'A'},

    {'4', '5', '6', 'B'},

    {'7', '8', '9', 'C'},

    {'\*', '0', '#', 'D'}

};

DigitalOut\* rows[4] = { &row1, &row2, &row3, &row4 };

DigitalIn\* cols[4] = { &col1, &col2, &col3, &col4 };

// Correct code

const char correctCode[4] = { '1', '2', '3', '4' };

char enteredCode[4];

int codeIndex = 0;

char scanKeypad() {

    for (int r = 0; r < 4; r++) {

        for (int i = 0; i < 4; i++) \*rows[i] = 1;

        \*rows[r] = 0;

        for (int c = 0; c < 4; c++) {

            if (\*cols[c] == 0) {

                ThisThread::sleep\_for(200ms); // Debounce

                return KEYS[r][c];

            }

        }

    }

    return '\0';

}

int main() {

    // Configure pull-ups

    for (int i = 0; i < 4; i++) {

        cols[i]->mode(PullUp);

    }

    alarmLed = 1; // Alarm ON by default

    // Show prompt on serial

    pc.write("Enter Code to Deactivate Alarm\r\n", 33);

    while (true) {

        char key = scanKeypad();

        if (key != '\0') {

            // Optional: show each key press on serial

            int len = sprintf(buffer, "Key Pressed: %c\r\n", key);

            pc.write(buffer, len);

            enteredCode[codeIndex++] = key;

            if (codeIndex == 4) {

                bool correct = true;

                for (int i = 0; i < 4; i++) {

                    if (enteredCode[i] != correctCode[i]) {

                        correct = false;

                        break;

                    }

                }

                if (correct) {

                    alarmLed = 0;

                    pc.write("✅ Alarm Deactivated!\r\n", 24);

                } else {

                    alarmLed = 1;

                    pc.write("❌ Incorrect Code. Try Again.\r\n", 31);

                }

                codeIndex = 0;

            }

        }

        ThisThread::sleep\_for(100ms);

    }

}

33333333333333333333333333333333333333333333333333

#include "mbed.h"

#include "arm\_book\_lib.h"

#include <ctime>  // For time and date

// === Alarm ===

DigitalOut alarmLed(D10);  // External alarm LED

// === Keypad 4x4 ===

// ROWS (outputs)

DigitalOut row1(D2);

DigitalOut row2(D3);

DigitalOut row3(D4);

DigitalOut row4(D5);

// COLS (inputs)

DigitalIn col1(D6);

DigitalIn col2(D7);

DigitalIn col3(D8);

DigitalIn col4(D9);

// Serial

BufferedSerial pc(USBTX, USBRX, 9600);

// Keypad layout

const char KEYS[4][4] = {

    {'1', '2', '3', 'A'},

    {'4', '5', '6', 'B'},

    {'7', '8', '9', 'C'},

    {'\*', '0', '#', 'D'}

};

DigitalOut\* rows[4] = { &row1, &row2, &row3, &row4 };

DigitalIn\* cols[4]  = { &col1, &col2, &col3, &col4 };

// Code check

const char correctCode[4] = {'1', '2', '3', '4'};

char enteredCode[4];

int codeIndex = 0;

// === Print date/time when alarm is triggered ===

void printCurrentDateTime() {

    time\_t now = time(NULL);

    tm \*t = localtime(&now);

    char datetime[64];

    strftime(datetime, sizeof(datetime), "%Y-%m-%d %H:%M:%S", t);

    pc.write("🚨 Alarm Triggered at: ", 24);

    pc.write(datetime, strlen(datetime));

    pc.write("\r\n", 2);

}

// === Keypad scanning ===

char scanKeypad() {

    for (int r = 0; r < 4; r++) {

        for (int i = 0; i < 4; i++) \*rows[i] = 1;

        \*rows[r] = 0;

        for (int c = 0; c < 4; c++) {

            if (\*cols[c] == 0) {

                ThisThread::sleep\_for(200ms); // Debounce

                return KEYS[r][c];

            }

        }

    }

    return '\0';

}

int main() {

    // Init serial

    pc.write("Enter Code to Deactivate Alarm\r\n", 33);

    // Pull-up setup

    for (int i = 0; i < 4; i++) {

        cols[i]->mode(PullUp);

    }

    // === RTC: Set time only ONCE (comment later) ===

    set\_time(1743648000); // 2025-4-03, 00:00:00

    alarmLed = 1; // Alarm starts ON

    while (true) {

        char key = scanKeypad();

        if (key != '\0') {

            enteredCode[codeIndex++] = key;

            if (codeIndex == 4) {

                bool correct = true;

                for (int i = 0; i < 4; i++) {

                    if (enteredCode[i] != correctCode[i]) {

                        correct = false;

                        break;

                    }

                }

                if (correct) {

                    alarmLed = 0;

                    pc.write("✅ Alarm Deactivated!\r\n", 24);

                } else {

                    alarmLed = 1;

                    pc.write("❌ Wrong Code! ", 16);

                    printCurrentDateTime();  // Record time

                }

                codeIndex = 0;

            }

        }

        ThisThread::sleep\_for(100ms);

    }

}

4444444444444444444444444

#include "mbed.h"

#include "arm\_book\_lib.h"

#include <ctime>

// === Alarm Output ===

DigitalOut alarmLed(D10); // Alarm indicator LED

// === 4x4 Keypad Connections ===

// Rows (outputs)

DigitalOut row1(D2);

DigitalOut row2(D3);

DigitalOut row3(D4);

DigitalOut row4(D5);

// Columns (inputs with pull-up)

DigitalIn col1(D6);

DigitalIn col2(D7);

DigitalIn col3(D8);

DigitalIn col4(D9);

// === Serial Monitor ===

BufferedSerial pc(USBTX, USBRX, 9600);

char buffer[64];

// === Keypad Map ===

const char KEYS[4][4] = {

{'1', '2', '3', 'A'},

{'4', '5', '6', 'B'},

{'7', '8', '9', 'C'},

{'\*', '0', '#', 'D'}

};

DigitalOut\* rows[4] = { &row1, &row2, &row3, &row4 };

DigitalIn\* cols[4] = { &col1, &col2, &col3, &col4 };

// === Deactivation Code ===

const char correctCode[4] = { '1', '2', '3', '4' };

char enteredCode[4];

int codeIndex = 0;

// === Print Timestamp on Alarm Trigger ===

void printCurrentDateTime() {

time\_t now = time(NULL);

tm \*t = localtime(&now);

char datetime[64];

strftime(datetime, sizeof(datetime), "%Y-%m-%d %H:%M:%S", t);

pc.write("\r\n===========================\r\n", 31);

pc.write("🚨 Alarm Triggered at: ", 24);

pc.write(datetime, strlen(datetime));

pc.write("\r\n===========================\r\n", 31);

}

// === Scan Keypad ===

char scanKeypad() {

for (int r = 0; r < 4; r++) {

for (int i = 0; i < 4; i++) \*rows[i] = 1;

\*rows[r] = 0;

for (int c = 0; c < 4; c++) {

if (\*cols[c] == 0) {

ThisThread::sleep\_for(200ms); // Debounce

return KEYS[r][c];

}

}

}

return '\0'; // No key pressed

}

int main() {

// Configure column inputs

for (int i = 0; i < 4; i++) {

cols[i]->mode(PullUp);

}

// === Set RTC Time (only once) ===

set\_time(1743648000); // 2025-4-03, 00:00:00

// === Startup State ===

alarmLed = 1;

pc.write("Enter Code to Deactivate Alarm\r\n", 33);

while (true) {

char key = scanKeypad();

if (key != '\0') {

enteredCode[codeIndex++] = key;

if (codeIndex == 4) {

bool correct = true;

for (int i = 0; i < 4; i++) {

if (enteredCode[i] != correctCode[i]) {

correct = false;

break;

}

}

if (correct) {

alarmLed = 0;

pc.write("✅ Alarm Deactivated!\r\n", 24);

} else {

alarmLed = 1;

pc.write("❌ Incorrect Code!\r\n", 23);

printCurrentDateTime(); // ✅ Timestamp on alarm trigger

}

codeIndex = 0; // Reset entry

}

}

ThisThread::sleep\_for(100ms);

}

}

555555555555555555555555555555555555555555

#include "mbed.h"

#include "arm\_book\_lib.h"

#include <ctime>

// === Alarm LED ===

DigitalOut alarmLed(D10);

// === Keypad (4x4) ===

// Rows (outputs)

DigitalOut row1(D2), row2(D3), row3(D4), row4(D5);

// Columns (inputs with pull-up)

DigitalIn col1(D6), col2(D7), col3(D8), col4(D9);

// Serial

BufferedSerial pc(USBTX, USBRX, 9600);

char buffer[64];

// Keypad Map

const char KEYS[4][4] = {

    {'1', '2', '3', 'A'},

    {'4', '5', '6', 'B'},

    {'7', '8', '9', 'C'},

    {'\*', '0', '#', 'D'}

};

DigitalOut\* rows[4] = { &row1, &row2, &row3, &row4 };

DigitalIn\* cols[4] = { &col1, &col2, &col3, &col4 };

// Correct Code

const char correctCode[4] = { '1', '2', '3', '4' };

char enteredCode[4];

int codeIndex = 0;

// === Alarm Timestamp Log ===

char alarmLog[5][64];  // Up to 5 timestamps

int logIndex = 0;

// Store current timestamp in log

void storeCurrentTime() {

    time\_t now = time(NULL);

    tm \*t = localtime(&now);

    strftime(alarmLog[logIndex], sizeof(alarmLog[logIndex]), "%Y-%m-%d %H:%M:%S", t);

    logIndex = (logIndex + 1) % 5;  // Circular buffer (keep last 5)

}

// Display all stored logs

void displayLog() {

    pc.write("\r\n📜 Alarm Log (Most Recent 5):\r\n", 34);

    for (int i = 0; i < 5; i++) {

        if (strlen(alarmLog[i]) > 0) {

            pc.write(" - ", 3);

            pc.write(alarmLog[i], strlen(alarmLog[i]));

            pc.write("\r\n", 2);

        }

    }

    pc.write("\r\n", 2);

}

// Scan keypad

char scanKeypad() {

    for (int r = 0; r < 4; r++) {

        for (int i = 0; i < 4; i++) \*rows[i] = 1;

        \*rows[r] = 0;

        for (int c = 0; c < 4; c++) {

            if (\*cols[c] == 0) {

                ThisThread::sleep\_for(200ms);

                return KEYS[r][c];

            }

        }

    }

    return '\0';

}

int main() {

    // Configure columns

    for (int i = 0; i < 4; i++) {

        cols[i]->mode(PullUp);

    }

    // Set time once

    set\_time(1743648000); // 2025-4-03, 00:00:00

    alarmLed = 1;

    pc.write("Enter Code to Deactivate Alarm\r\n", 33);

    while (true) {

        char key = scanKeypad();

        if (key != '\0') {

            if (key == '#') {

                displayLog();  // ✅ Show log

                continue;

            }

            enteredCode[codeIndex++] = key;

            if (codeIndex == 4) {

                bool correct = true;

                for (int i = 0; i < 4; i++) {

                    if (enteredCode[i] != correctCode[i]) {

                        correct = false;

                        break;

                    }

                }

                if (correct) {

                    alarmLed = 0;

                    pc.write("✅ Alarm Deactivated!\r\n", 24);

                } else {

                    alarmLed = 1;

                    pc.write("❌ Incorrect Code!\r\n", 23);

                    storeCurrentTime();  // ✅ Save timestamp

                }

                codeIndex = 0;

            }

        }

        ThisThread::sleep\_for(100ms);

    }

}