REAL-TIME EMBEDDED SYSTEM
EXPERIMENT NO. 7

Real-Time Vault System

Name: NAVARRO, ROD GERYK C.

Course/Section: CPE161P-4/C1

Group No.: N/A

Date of Performance: 02/01/2025
Date of Submission: 02/08/2025

CYREL O. MANLISES, PH.D.
Instructor

DISCUSSION

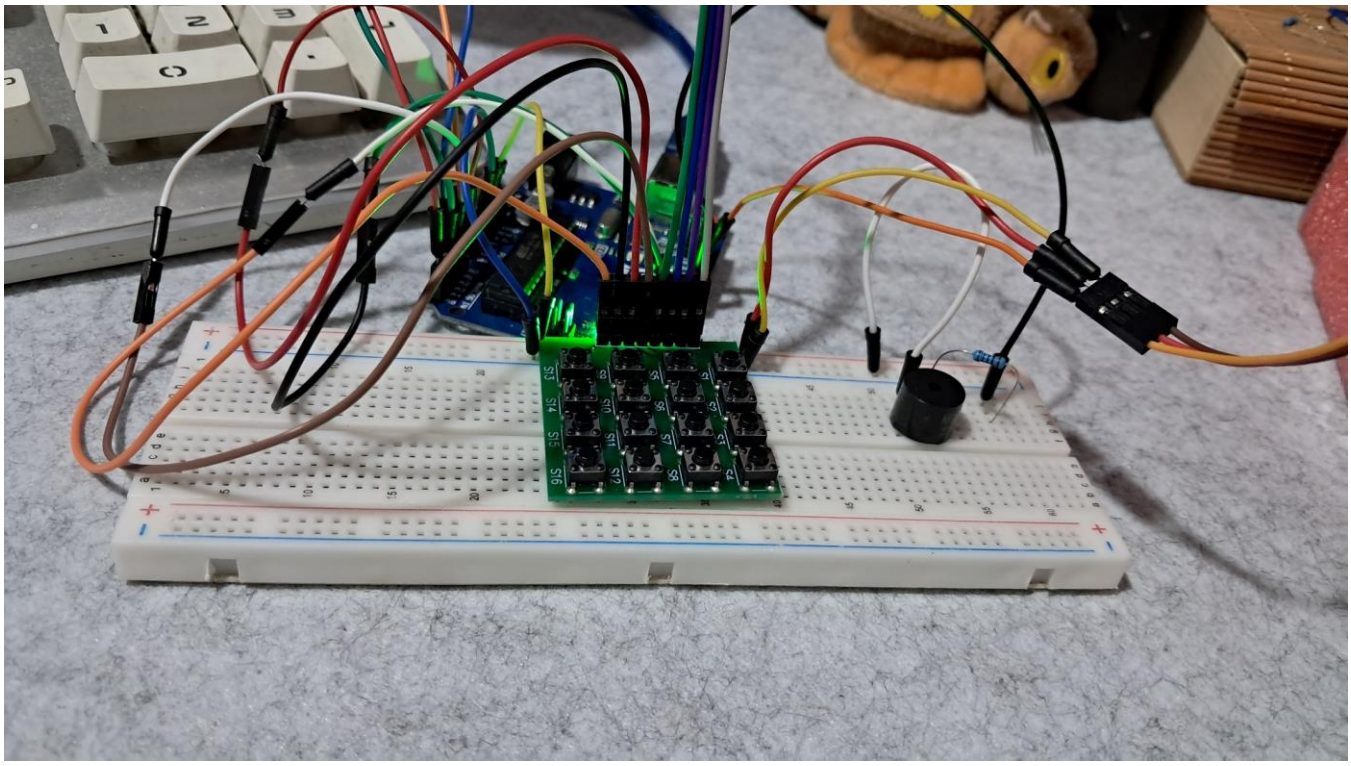


Figure 1: The Connection of the 4x4 Keypad, Servo Motor, and Buzzer

This experiment focuses on building a real-time vault system. At the start of the experiment, I had already set up the circuit because I reused the previous experiment s circuit. I only removed the extra components that were not needed for this project. In most vaults, there is a built-in keypad for entering a passcode. In this experiment, I used a 4x4 keypad to function as the vault s keypad. The keypad includes an "Enter" key to confirm the password after typing. Additionally, there is a feature that allows the user to extend the vault door s open time by 5 more seconds. Normally, the door remains open for only 20 seconds before automatically closing.

To enhance security, I added a buzzer. If the user enters the wrong password, the buzzer will beep five times as an alert. If the password is correct, the buzzer will beep only once. A servo motor is used to control the vault door s opening and closing mechanism. All the components are connected to an Arduino Uno, which serves as the main controller of the system.

Navarro_EXP7_CPE161P.ino

```
const char rows = 4;
const char cols = 4;
const int buzzAlarm = 9;
const int servoPin = 12;

const char correctPassword[] = "637921";
char enteredPassword[7];
int passwordIndex = 0;
bool vaultOpen = false;
unsigned long vaultCloseTime = 0;

const char keys[rows][cols] = {
  {'D', '*', '0','#'},
  {'C', '9', '8','7'},
  {'B', '6', '5','4'},
  {'A', '3', '2','1'}
};

char rowPins[rows] = {A5, A4, A3, A2};
char colPins[cols] = {A1, A0, 2, 3};

void setup() {
  Serial.begin(9600);
  pinMode(servoPin, OUTPUT);
  pinMode(buzzAlarm, OUTPUT);

for(char r = 0; r < rows; r++) {
  pinMode(rowPins[r], INPUT);
  digitalWrite(rowPins[r], HIGH);
}
```

Figure 2: *The Initialization of Variables and Components*

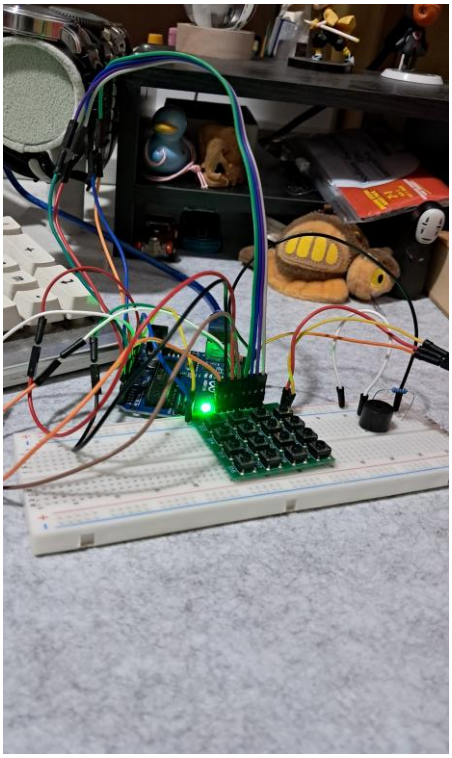
In this experiment, I created a real-time vault system using an Arduino Uno, a 4x4 keypad, a servo motor, and a buzzer. To start, I initialized several variables that are essential for the system. I set the number of rows and columns for the keypad (rows = 4, cols = 4) and defined the pins for the buzzer and servo motor (buzzAlarm = 9, servoPin = 12). I also stored the correct password (637921) in an array and created an empty array (enteredPassword[]) to store the user's input. To track whether the vault is open, I used a boolean variable (vaultOpen), and to manage the vault's closing time, I used vaultCloseTime. Additionally, I mapped the keypad buttons to specific characters so the Arduino could recognize the user's input when a key is pressed.

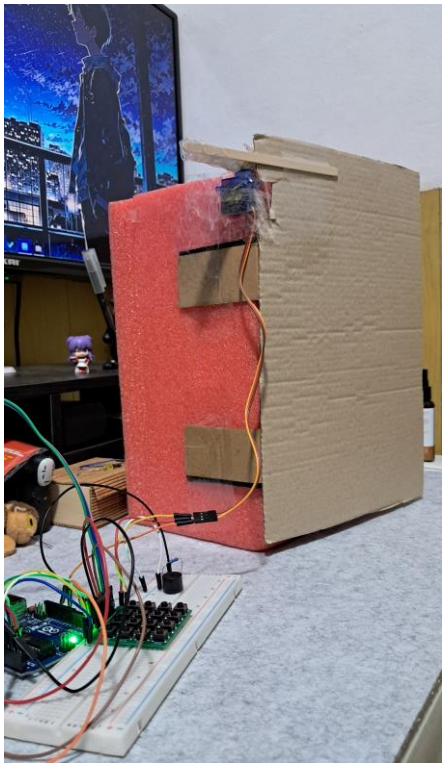
Navarro_EXP7_CPE161P.ino

```
22  void setup() {
23    Serial.begin(9600);
24    pinMode(servoPin, OUTPUT);
25    pinMode(buzzAlarm, OUTPUT);
26
27  for(char r = 0; r < rows; r++) {
28    pinMode(rowPins[r], INPUT);
29    digitalWrite(rowPins[r], HIGH);
30  }
31
32
33  for(char c = 0; c < cols; c++){
34    pinMode(colPins[c], OUTPUT);
35  }
36   closeVault();
37  }
38
39  void loop() {
40   char key = getKey();
41   if(key !=0){
42    Serial.println(key);
43    handleKeyPress(key);
44   }
45
46   if(vaultOpen && millis() >= vaultCloseTime) {
47    closeVault();
48   }
49  }
50
51  char getKey(){
```

Figure 3: *The Setup of the Components*

For the setup, I first started the serial communication (Serial.begin(9600)) to allow debugging. Then, I set the buzzer and servo motor as output devices (pinMode(servoPin, OUTPUT); pinMode(buzzAlarm, OUTPUT);). For the keypad, I configured the row pins as inputs with pull-ups and the column pins as outputs. Lastly, I called the closeVault() function to make sure the vault door starts in a locked position when the system powers on. Since I reused the circuit from a previous experiment, I only had to remove unnecessary components, making the setup process faster.





Navarro_EXP7_CPE161P.ino

```
38
39  void loop() {
40   char key = getKey();
41   if(key !=0){
42    Serial.println(key);
43    handleKeyPress(key);
44   }
45
46   if(vaultOpen && millis() >= vaultCloseTime) {
47    closeVault();
48   }
49  }
50
```

```
2
3  void buzz(int times){
4    for(int i = 0; i < times; i++){
5      digitalWrite(buzzAlarm, HIGH);
6      delay(200);
7      digitalWrite(buzzAlarm, LOW);
8      delay(200);
9    }
0  }
```

Figure 4: Fixing the Errors

While testing the system, I encountered an issue where the vault wouldn't close automatically after the set time. To fix this, I checked the loop() function and ensured that it continuously monitors the current time using millis(). If the vault is open and the current time reaches vaultCloseTime, the closeVault() function is called to lock the door. Another issue was with the buzzer sound, which wasn't functioning properly at first. I fixed this by adding the buzz() function, which turns the buzzer on and off for a specific number of times, depending on whether the password is correct (1 beep) or incorrect (5 beeps).

Navarro_EXP7_CPE161P.ino

```
38
39  void loop() {
40   char key = getKey();
41   if(key !=0){
42    Serial.println(key);
43    handleKeyPress(key);
44   }
45
46   if(vaultOpen && millis() >= vaultCloseTime) {
47    closeVault();
48   }
49  }
50
51  char getKey(){
52    char k = 0;
53    for (char c = 0; c < cols; c++) {
54      digitalWrite(colPins[c], LOW);
55      for(char r = 0; r < rows; r++){
56        if(digitalRead(rowPins[r]) == LOW){
57        delay(20);
58        while (digitalRead(rowPins[r]) == LOW);
59        k = keys[r][c];
60      }
61    }
62    digitalWrite(colPins[c], HIGH);
63  }
64    return k;
65  }
66
67
68  void handleKeyPress(char key){
```

Navarro_EXP7_CPE161P.ino

```
87
88  void checkPassword(){
89    if(strcmp(enteredPassword, correctPassword) == 0){
90      Serial.println("Correct Password! Vault Open.");
91      buzz(1);
92      openVault();
93    }else{
94      Serial.println("Incorrect Password");
95      buzz(5);
96    }
97  }
98
99  void openVault(){
100   vaultOpen = true;
101   delay(1000);
102   vaultCloseTime = millis() + 20000;
103   moveServo(0);
104 }
105
106 void closeVault(){
107   vaultOpen = false;
108   moveServo(90);
109   Serial.println("Vault Closed.");
110 }
111
112 void moveServo(int angle){
113   int pulseWidth = map(angle, 0, 180, 500, 2500);
114   for(int i = 0; i < 50; i++){
115     digitalWrite(servoPin, HIGH);
116     delayMicroseconds(pulseWidth);
117     digitalWrite(servoPin, LOW);
```

Figure 5: Loop Function

*The loop() function handles key presses and manages the vault's locking system. When a key is pressed, the getKey() function detects which key was selected. If the user presses #, it resets the password input. If they enter a number, it is stored in enteredPassword[]. When the "D" key is pressed, the checkPassword() function compares the input with the correct password. If it matches, the vault opens, and a timer starts. If the user wants to keep the vault open for longer, they can press *, which adds 5 more seconds. This ensures that the system works efficiently, securing the vault while allowing flexibility in operation.*

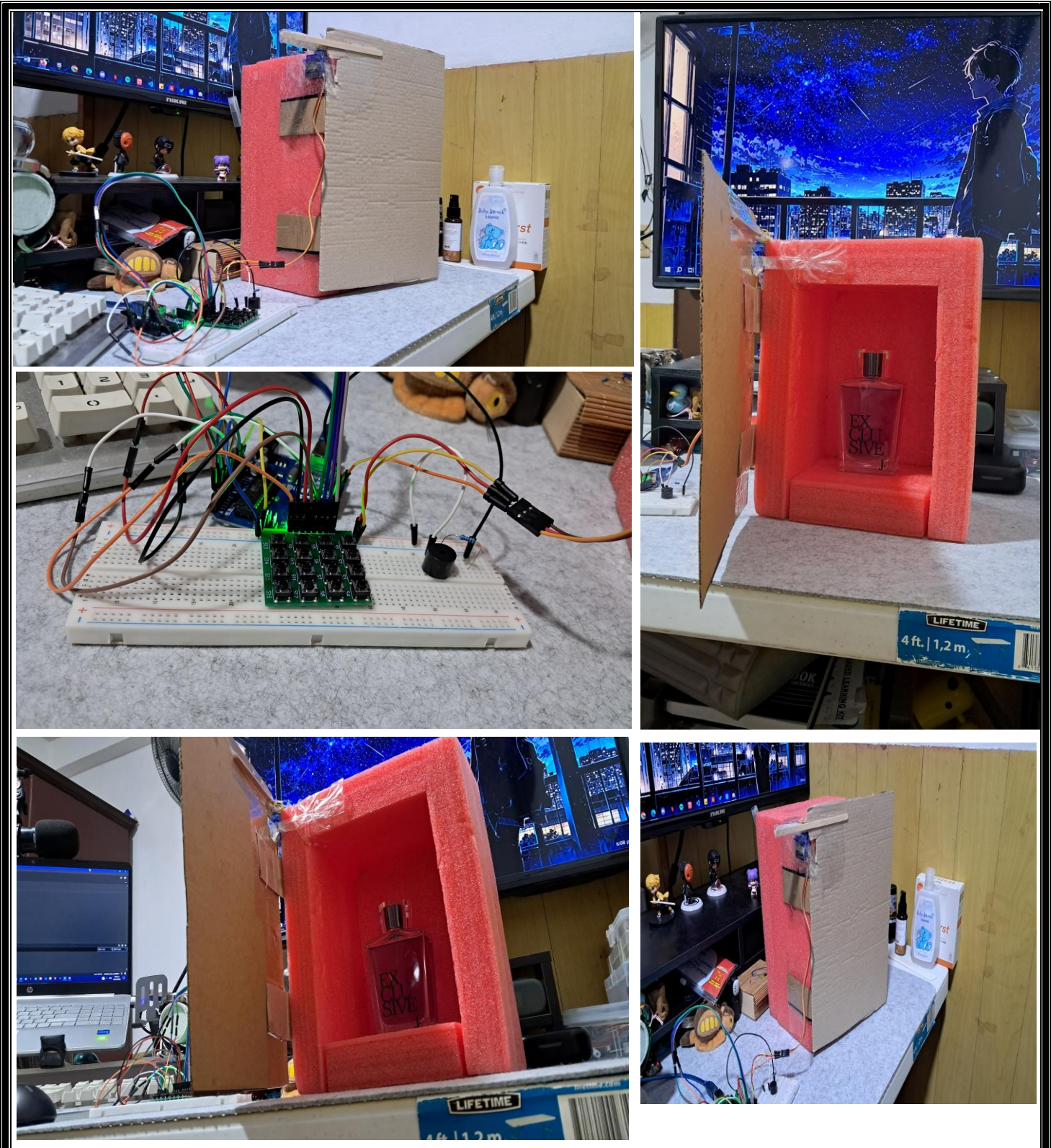


Figure 6: Working Prototype.

This is the final working prototype of the real-time vault system experiment, which successfully meets all the requirements.

CONCLUSION

In this experiment, I successfully built a real-time vault system using an Arduino Uno, a 4x4 keypad, a servo motor, and a buzzer. The system allows users to enter a password using the keypad, and if the correct password is entered, the vault door opens for 20 seconds. A buzzer provides audio feedback, beeping once for a correct password and five times for a wrong password. The experiment also includes a feature that lets users extend the vault's open time by five seconds. To ensure the vault starts in a locked position when powered on, I initialized all components properly and set up serial communication for debugging.

Throughout the testing phase, I encountered and fixed several issues, such as the vault not closing automatically and the buzzer not functioning correctly. By refining the loop() function and implementing a millis()-based timing system, I ensured the vault closes on time. I also improved the buzzer's performance by adjusting the buzz() function. The final working prototype successfully met all requirements, demonstrating a functional and secure vault system. To sum up everything, this experiment provided valuable hands-on experience in integrating multiple components and troubleshooting issues in real-time systems.