REAL-TIME EMBEDDED SYSTEM
EXPERIMENT NO. 4

Appliance Controller

Name: NAVARRO, ROD GERYK C.

Course/Section: CPE161P-4/C1

Group No.: N/A

Date of Performance: 01/09/2025
Date of Submission: 02/03/2025

CYREL O. MANLISES, PH.D.
Instructor

DISCUSSION

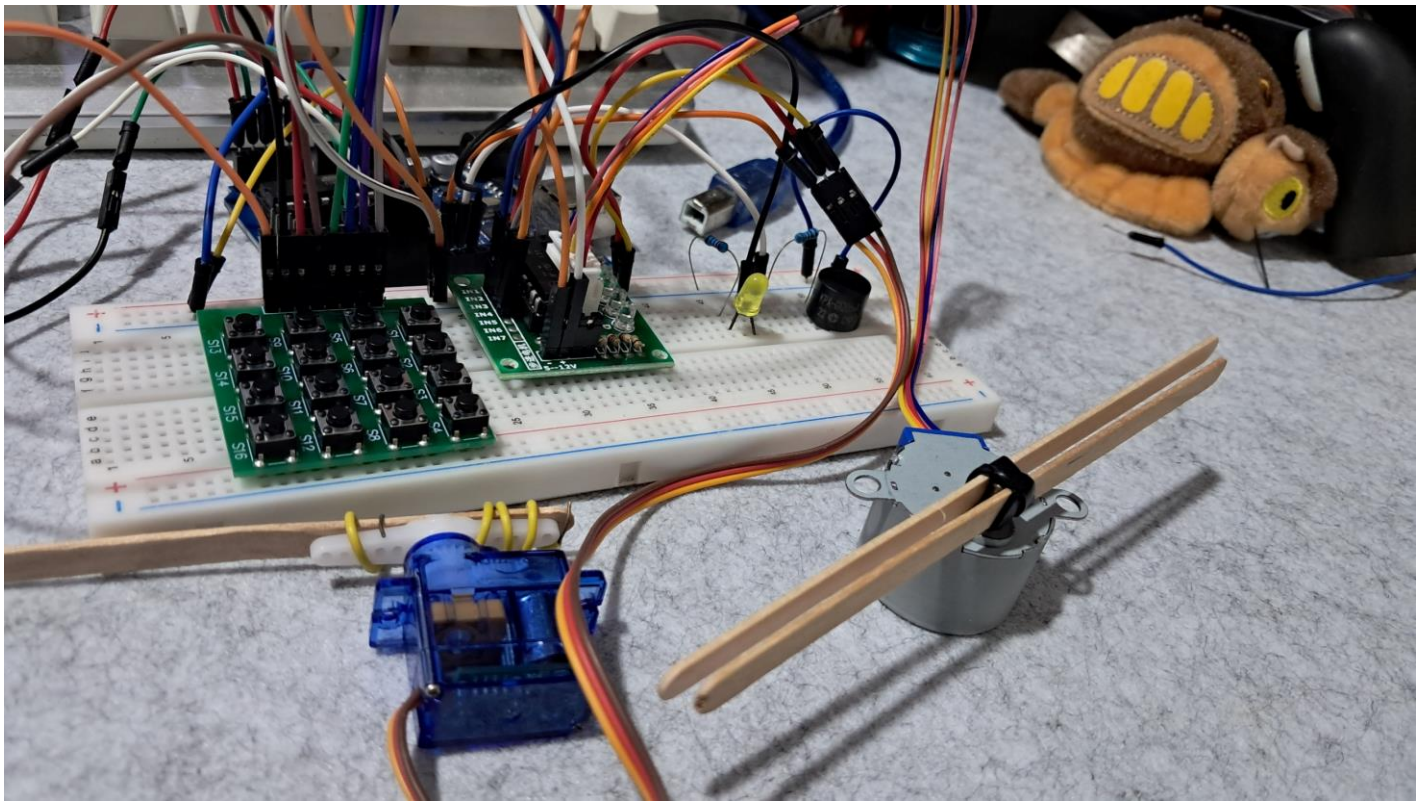

Figure 1: *The Connection of the 4x4 Keypad, Stepper Motor, Servo Motor, LED, and Buzzer*

This experiment is about controlling appliances in real-time. To do this, we need a controller or remote. In my case, I used a 4x4 keypad, although I also had the option to use a Bluetooth remote. I chose the keypad because it will be more useful for me in my next experiment. Figure 1 shows the complete connection of all the components used in this setup. In this experiment, I used two types of motors: a stepper motor and a servo motor. The stepper motor works like a fan, while the servo motor functions like a glass wiper. I also included a buzzer and an LED. The buzzer serves as an alarm, and the LED works as a light. These components represent the appliances I am controlling. The 4x4 keypad acts as my remote or

controller, which allows me to manage and operate all of these appliances.

Navarro_EXP4_CPE161P.ino

```
1   const char rows = 4;
2   const char cols = 4;
3   const int lightLed = 10;
4   const int buzzAlarm = 9;
5   int switching = 0;
6   int buttonState = 0;
7   int servoPin = 8;
8   bool motorState = false;
9   bool servoState = false;
10
11  int portIN1 = 4;
12  int portIN2 = 5;
13  int portIN3 = 6;
14  int portIN4 = 7;
15
16  const char keys[rows][cols] = {
17    {'D','*','0','#'},
18    {'C','9','8','7'},
19    {'B','6','5','4'},
20    {'A','3','2','1'}
21  };
22
23  char rowPins[rows]={A5,A4,A3,A2};
24  char colPins[cols]={A1,A0,2,3};
25
```

Figure 2: The Initialization of Variables and Components

In this experiment, I started by initializing the variables that control the different components. I declared constants for the keypad dimensions (rows = 4 and cols = 4), and assigned specific pins for the LED light (lightLed = 10), buzzer (buzzAlarm = 9), and servo motor (servoPin = 8). I also defined variables to track the state of the appliances, like motorState and servoState, which are both set to false at the start. These help in turning the motors on and off later in the code. For the stepper motor, I assigned pins portIN1 to portIN4 (pins 4, 5, 6, and 7). Finally, I created a 2D array called keys to map the keypad buttons, which allows me to assign specific actions to each key.

Navarro_EXP4_CPE161P.ino

```
24  char colPins[cols]={A1,A0,2,3};
25
26  void setup() {
27    Serial.begin(9600);
28    pinMode(servoPin, OUTPUT);
29    pinMode(lightLed, OUTPUT);
30    pinMode(buzzAlarm, OUTPUT);
31
32  for(char r = 0; r < rows; r++){
33    pinMode(rowPins[r], INPUT);
34    digitalWrite(rowPins[r], HIGH);
35  }
36
37  for(char c = 0; c < cols; c++){
38    pinMode(colPins[c], OUTPUT);
39    }
40  pinMode(portIN1, OUTPUT);
41  pinMode(portIN2, OUTPUT);
42  pinMode(portIN3, OUTPUT);
43  pinMode(portIN4, OUTPUT);
44
45  }
```

Figure 3: *The Setup of the Components*

In the setup() function, I initialized the components by setting the pin modes. For outputs like the servo, LED, and buzzer, I used pinMode(pin, OUTPUT). This tells the Arduino that these pins will send signals to the appliances. For the keypad, the rows are set as inputs, and I used digitalWrite(rowPins[r], HIGH) to enable pull-up resistors, ensuring the buttons are ready to detect key presses. The columns of the keypad are set as outputs, allowing the Arduino to scan for key presses by sending signals through the columns and reading the rows. I also set the pins for the stepper motor as outputs, as they will control the motor's rotation. This setup prepares the system to read button inputs and control the appliances.

```
char key = getKey();
if(key != 0){
  Serial.println(key);
 }

  buttonState = digitalRead(switching);

 if (key == '7') {

   if (buttonState == LOW) {
     digitalWrite(lightLed, !digitalRead(lightLed));
   }
   delay(50);
 }

 switching = buttonState;

 if (key == '4') {

   if (buttonState == LOW) {
     digitalWrite(buzzAlarm, !digitalRead(buzzAlarm));
   }
   delay(50);
 }
```

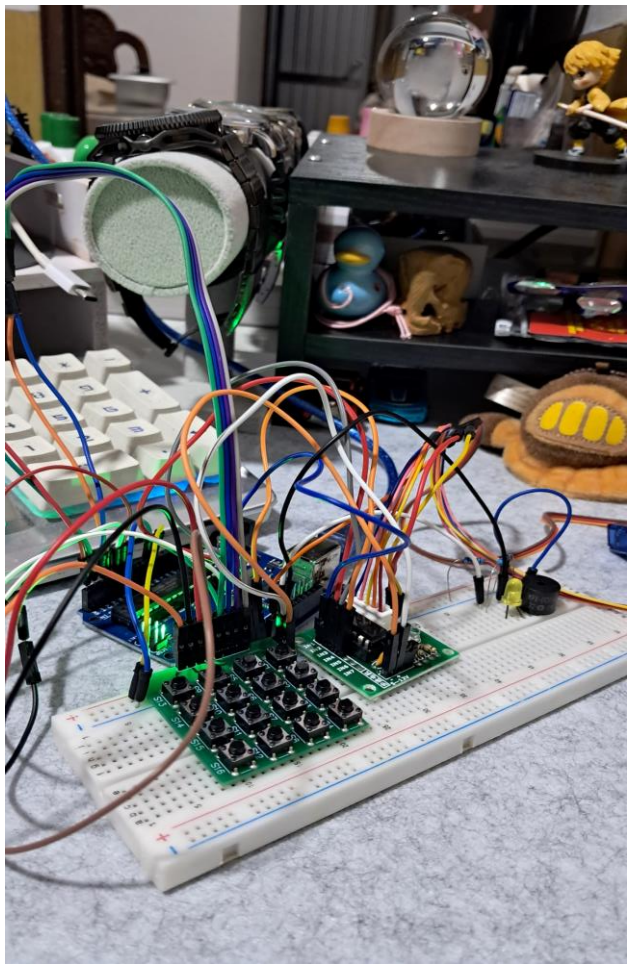

Figure 4: Fixing the Errors

While testing the code, I encountered an issue with the appliances not responding correctly to the keypad inputs. The problem was in the way the button state was being read and updated. I noticed that the variable switching was not assigned a proper pin to read from, which caused the conditions checking buttonState to fail. To fix this, I realized that switching should either be assigned to a specific pin connected to an external switch or removed entirely if not needed. After correcting this, the appliances started responding properly to the keypad inputs. Another issue was with the debounce delay; without it, the appliances would sometimes toggle multiple times with a single press. Adding a small delay of 50 milliseconds after each button press solved this problem.

Navarro_EXP4_CPE161P.ino

```
46
47  void loop() {
48  char key = getKey();
49   if(key != 0){
50      Serial.println(key);
51    }
52
53     buttonState = digitalRead(switching);
54
55    if (key == '7') {
56
57      if (buttonState == LOW) {
58        digitalWrite(lightLed, !digitalRead(lightLed));
59      }
60      delay(50);
61    }
62
63    switching = buttonState;
64
65    if (key == '4') {
66
67      if (buttonState == LOW) {
68        digitalWrite(buzzAlarm, !digitalRead(buzzAlarm));
69      }
70      delay(50);
71    }
72
73    if (key == '#') {
74
75      resetAll();
76
77    }
78      if (key == '1') {
```

Navarro_EXP4_CPE161P.ino

```
84        stepSequence(1,0,0,0);
85        delay(2);
86        stepSequence(0,1,0,0);
87        delay(2);
88        stepSequence(0,0,1,0);
89        delay(2);
90        stepSequence(0,0,0,1);
91        delay(2);
92      }else {
93        stepSequence(0, 0, 0, 0); // Turn off all coils when motor is off
94      }
95
96      if (key == '2') {
97        servoState = !servoState; // Toggle motor state
98       }
99
100     if(servoState){
101      for(int angle = 0; angle <= 180; angle ++){
102        int pulseWidth = map(angle, 0, 180, 500, 2500);
103        digitalWrite(servoPin, HIGH);
104        delayMicroseconds(pulseWidth);
105        digitalWrite(servoPin, LOW);
106        delay(20 - pulseWidth / 1000);
107      }  for(int angle = 180; angle >= 0; angle --){
108        int pulseWidth = map(angle, 0, 180, 500, 2500);
109        digitalWrite(servoPin, HIGH);
110        delayMicroseconds(pulseWidth);
111        digitalWrite(servoPin, LOW);
112        delay(20 - pulseWidth / 1000);
113      }
114    }
115  }
```

Figure 5: Loop Function

In the loop() function, I handled the main operations of the appliances. The getKey() function reads the pressed key from the keypad. For example, pressing '7' toggles the LED light, and '4' toggles the buzzer. The '#' key resets everything, turning off the LED, buzzer, and motors. For the motors, pressing '1' toggles the stepper motor (acting like a fan), and pressing '2' toggles the servo motor (acting like a glass wiper). When the stepper motor is on, the stepSequence() function runs to create the correct pattern for the motor to rotate. For the servo motor, I used a loop to move the motor from 0 to 180 degrees and back, simulating a wiper motion. This loop continuously checks for keypad inputs and updates the appliances in real time, allowing me to control them easily with the keypad.

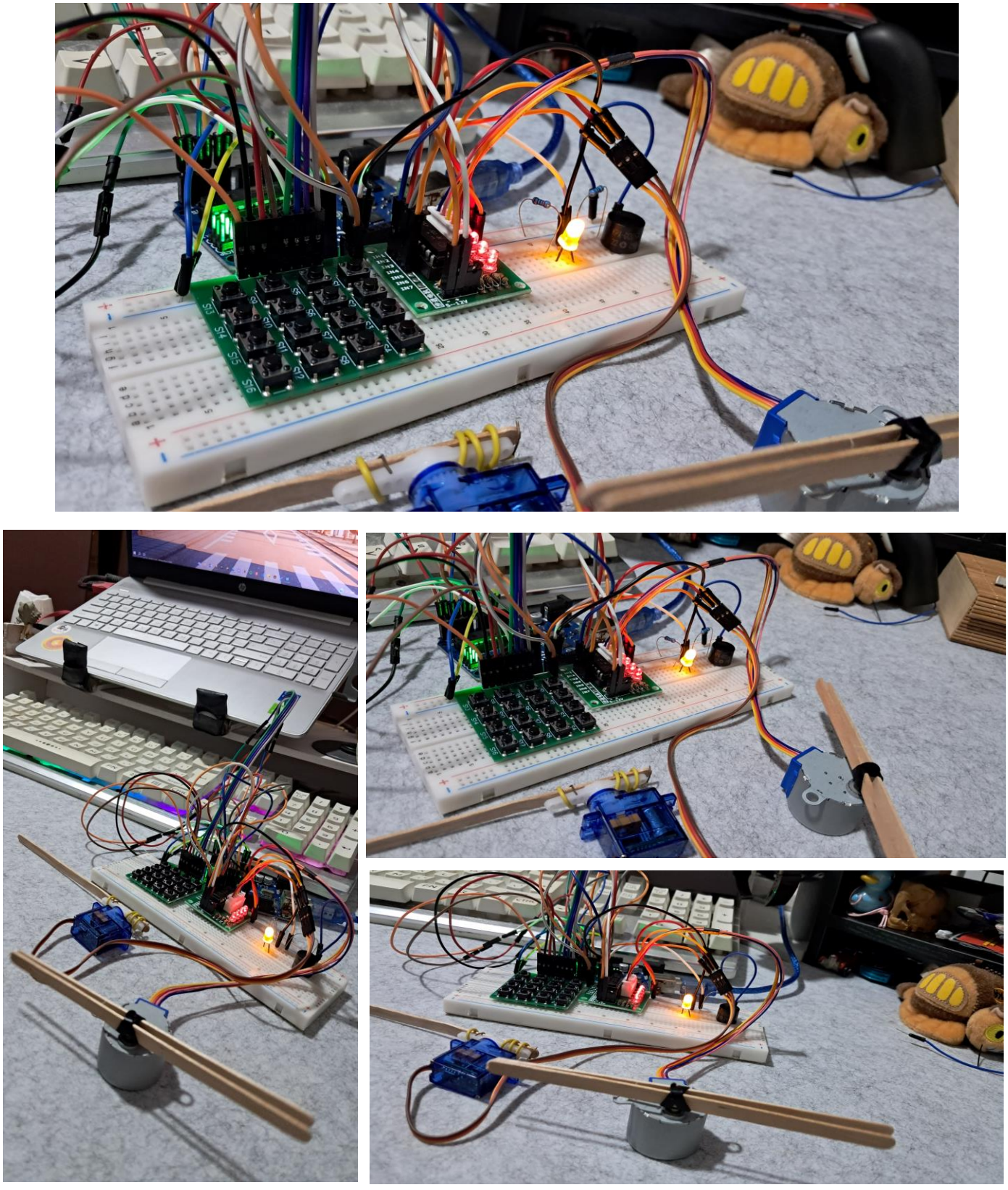

Figure 6: Working Prototype.

This is the final working prototype of the appliance controller experiment, which successfully meets all the requirements.

CONCLUSION

In this experiment, I successfully created a real-time appliance controller using a 4x4 keypad, stepper motor, servo motor, LED, and buzzer. The keypad acted as the main controller, allowing me to turn appliances on and off with specific key presses. For example, pressing '7' toggled the LED light, '4' controlled the buzzer, '1' activated the stepper motor (working like a fan), and '2' controlled the servo motor (moving like a glass wiper). The buzzer served as an alarm, while the LED acted as a simple light. The project also included a reset button ('#') to turn off all appliances at once. This setup helped me understand how to manage multiple devices in real-time using simple keypad inputs.

During the process, I faced a few challenges, such as incorrect button responses and issues with the debounce delay. I fixed the button response problem by assigning the correct pins and adding a debounce delay of 50 milliseconds to prevent multiple toggles from a single press. After troubleshooting, all components responded correctly, and the system worked smoothly. This experiment not only taught me how to control different appliances using Arduino but also improved my problem-solving skills when dealing with hardware and software errors. The project was a success, and it gave me a solid foundation for future experiments involving real-time appliance control.