REAL-TIME EMBEDDED SYSTEM
PROJECT DESIGN

Dual-Mode Real-Time Cap Sorting System Using Arduino Uno

Name: NAVARRO, ROD GERYK C.

Course/Section: CPE161P-4/C1

Group No.: N/A

Date of Performance: 02/17/2025
Date of Submission: 02/22/2025

CYREL O. MANLISES, PH.D.
Instructor

DISCUSSION

Part I. Discuss in detail your progress here.

This week, I started by connecting all the components to the Arduino R3 and writing the code for the two operating modes of the system. The components I connected include an LCD, a 4x4 keypad, and two servo motors. However, I did not connect the color sensor yet because my main focus for this stage was to develop and test the two modes: automatic and manual. In automatic mode, the system sorts bottle caps into their respective storage areas without user intervention. In manual mode, the user can control the movement of the servo motors to sort the caps as needed. This allows flexibility in operation, depending on whether automation or manual control is preferred.

At the beginning of my experiment, I initialized all the necessary variables to ensure the system runs smoothly. I used the LiquidCrystal_I2C library to control the LCD and defined the keypad layout using a 4x4 matrix. I also specified the pin connections for both the keypad and the servo motors. The servo motors are assigned to pins 11 and 12, while the keypad rows and columns are connected to pins 2 to 9. Additionally, I created a moveServo() function to control servo movement by mapping angles to appropriate duty cycle values. This function ensures that the servos move accurately based on the selected mode. After defining the variables, I proceeded with the hardware setup in the setup() function. The system starts by displaying an initialization message on the LCD, giving visual feedback that it is powering up. I set the servo motor pins as outputs and configured the keypad row pins as outputs while enabling internal pull-up resistors for the column pins. This setup prevents floating values and ensures accurate key detection. Lastly, I displayed a menu on the LCD, prompting the user to select between automatic and manual mode by pressing 'A' or 'B' on the keypad.

While testing the system, I encountered an issue where the keypad input was not consistently detected. I resolved this by modifying the getKey() function. The function scans through each row, sets it to LOW, and then checks all columns for a pressed key. If a key is detected, the function immediately returns its value and resets the row to HIGH. This approach prevents conflicts and ensures reliable key detection. Additionally, I noticed that the servo movement was abrupt, so I added delays between operations to allow smoother transitions between positions. In the loop() function, I implemented logic for both automatic and manual modes. If the user presses 'A,' the system enters automatic mode, and the servos move sequentially to sort the bottle caps. If 'B' is pressed, the system switches to manual mode, allowing the user to control the servos by pressing specific keys. Keys '1' to '3' adjust Servo 1's position, while keys '4' to '6' control Servo 2. This setup gives users flexibility in how they operate the system.

Part II. Provide screenshots as proof of your progress.

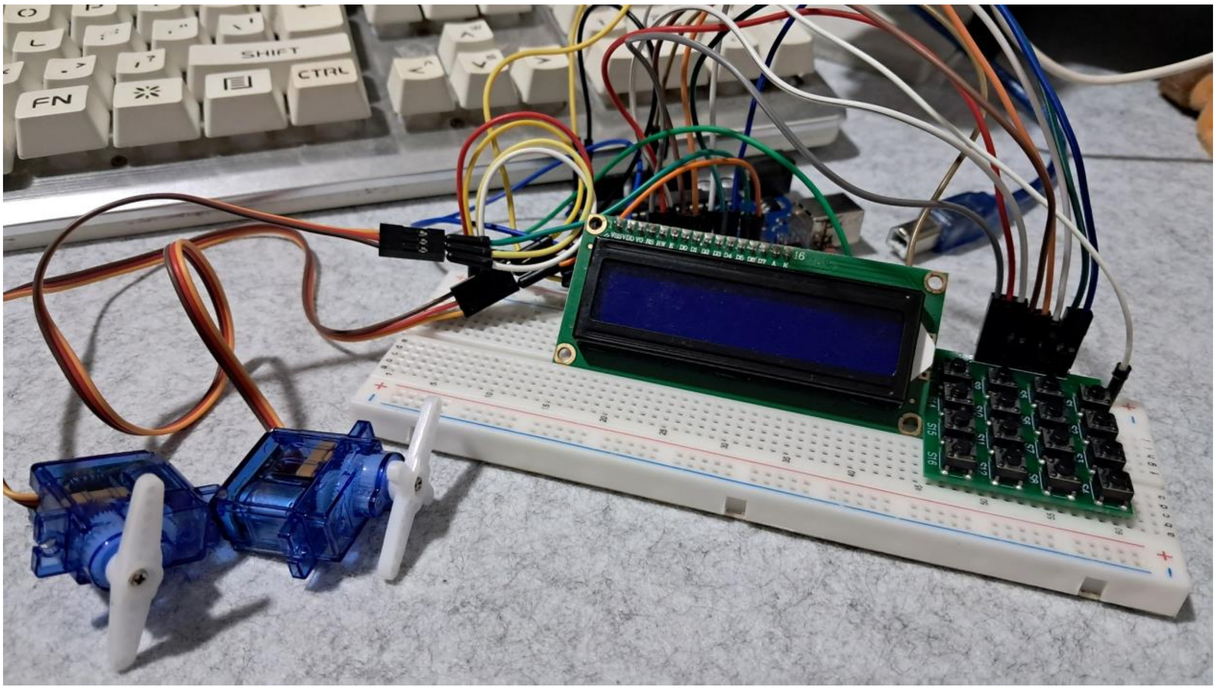
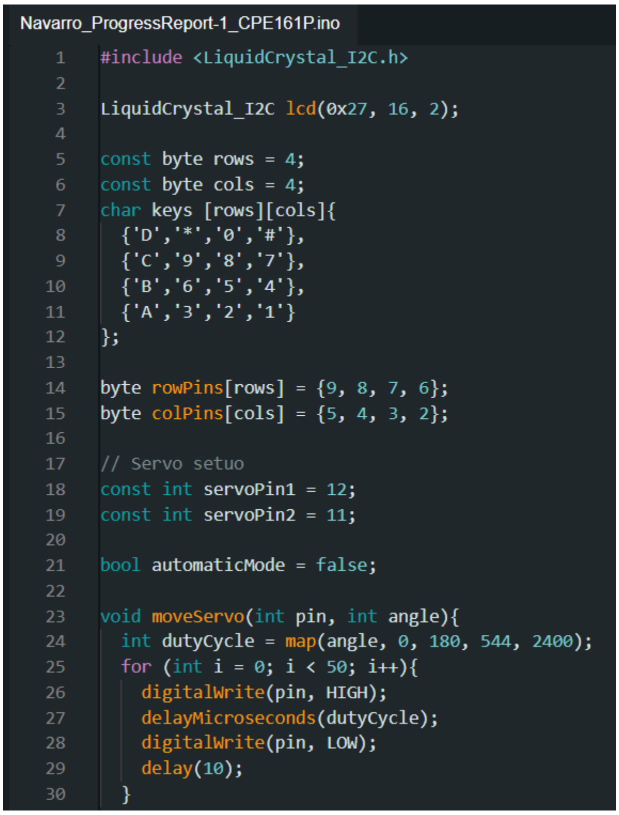
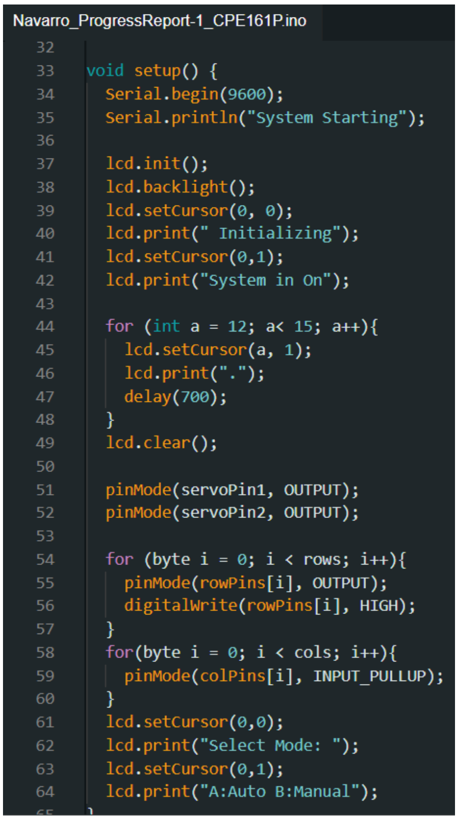


Figure 1: The Connection of the 4x4 Keypad, Servo Motors, and the LCD



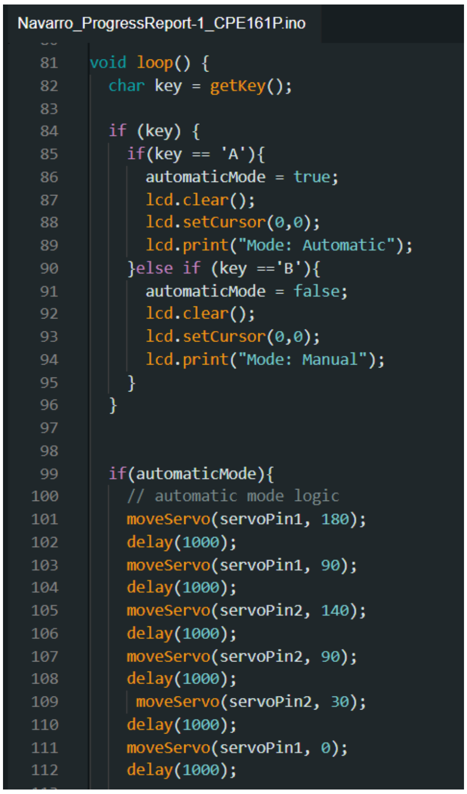
```
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 16, 2);

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

byte rowPins[rows] = {9, 8, 7, 6};
byte colPins[cols] = {5, 4, 3, 2};

// Servo setuo
const int servoPin1 = 12;
const int servoPin2 = 11;

bool automaticMode = false;

void moveServo(int pin, int angle){
  int dutyCycle = map(angle, 0, 180, 544, 2400);
  for (int i = 0; i < 50; i++){
    digitalWrite(pin, HIGH);
    delayMicroseconds(dutyCycle);
    digitalWrite(pin, LOW);
    delay(10);
  }
```

Figure 2: The Initialization of Variables and Components



```
void setup() {
  Serial.begin(9600);
  Serial.println("System Starting");

  lcd.init();
  lcd.backlight();
  lcd.setCursor(0, 0);
  lcd.print(" Initializing");
  lcd.setCursor(0,1);
  lcd.print("System in On");

  for (int a = 12; a< 15; a++){
    lcd.setCursor(a, 1);
    lcd.print(".");
    delay(700);
  }
  lcd.clear();

  pinMode(servoPin1, OUTPUT);
  pinMode(servoPin2, OUTPUT);

  for (byte i = 0; i < rows; i++){
    pinMode(rowPins[i], OUTPUT);
    digitalWrite(rowPins[i], HIGH);
  }
  for(byte i = 0; i < cols; i++){
    pinMode(colPins[i], INPUT_PULLUP);
  }
  lcd.setCursor(0,0);
  lcd.print("Select Mode: ");
  lcd.setCursor(0,1);
  lcd.print("A:Auto B:Manual");
```

Figure 3: The Setup of the Components



```
void loop() {
  char key = getKey();

  if (key) {
    if(key == 'A'){
      automaticMode = true;
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print("Mode: Automatic");
    }else if (key =='B'){
      automaticMode = false;
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print("Mode: Manual");
    }
  }


  if(automaticMode){
    // automatic mode logic
    moveServo(servoPin1, 180);
    delay(1000);
    moveServo(servoPin1, 90);
    delay(1000);
    moveServo(servoPin2, 140);
    delay(1000);
    moveServo(servoPin2, 90);
    delay(1000);
     moveServo(servoPin2, 30);
    delay(1000);
    moveServo(servoPin1, 0);
    delay(1000);
```

Figure 4: Loop Function

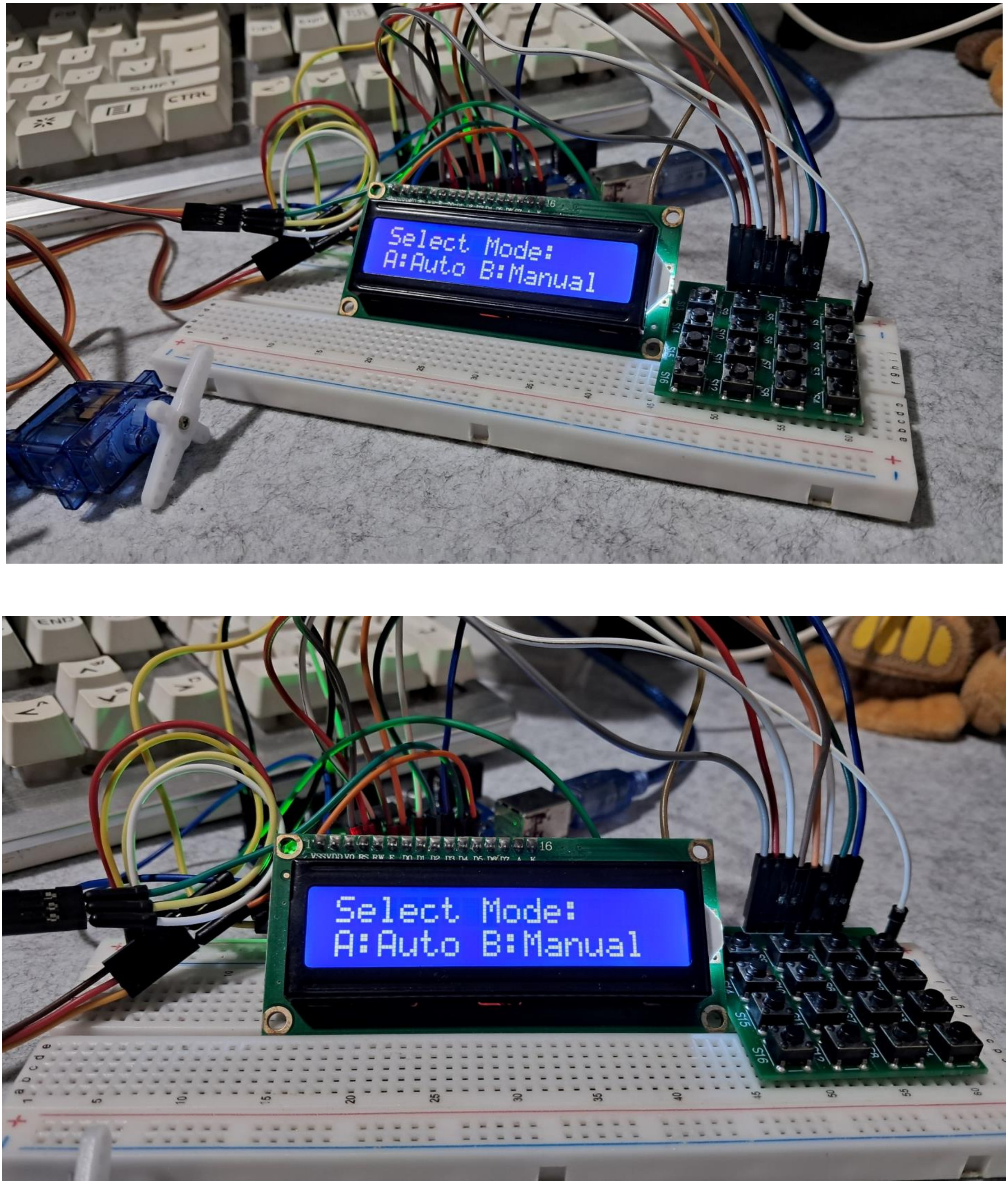


Figure 5: Working Prototype

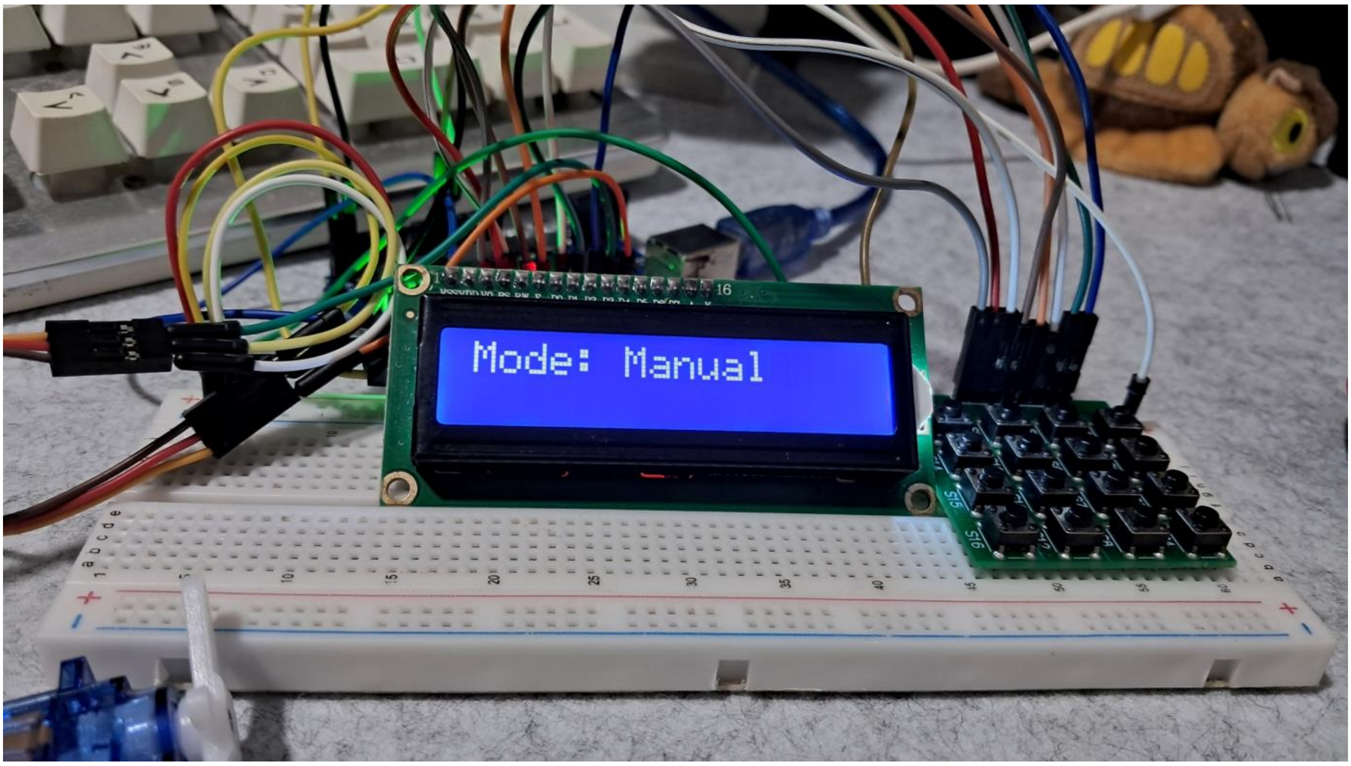


Figure 6: Manual Mode

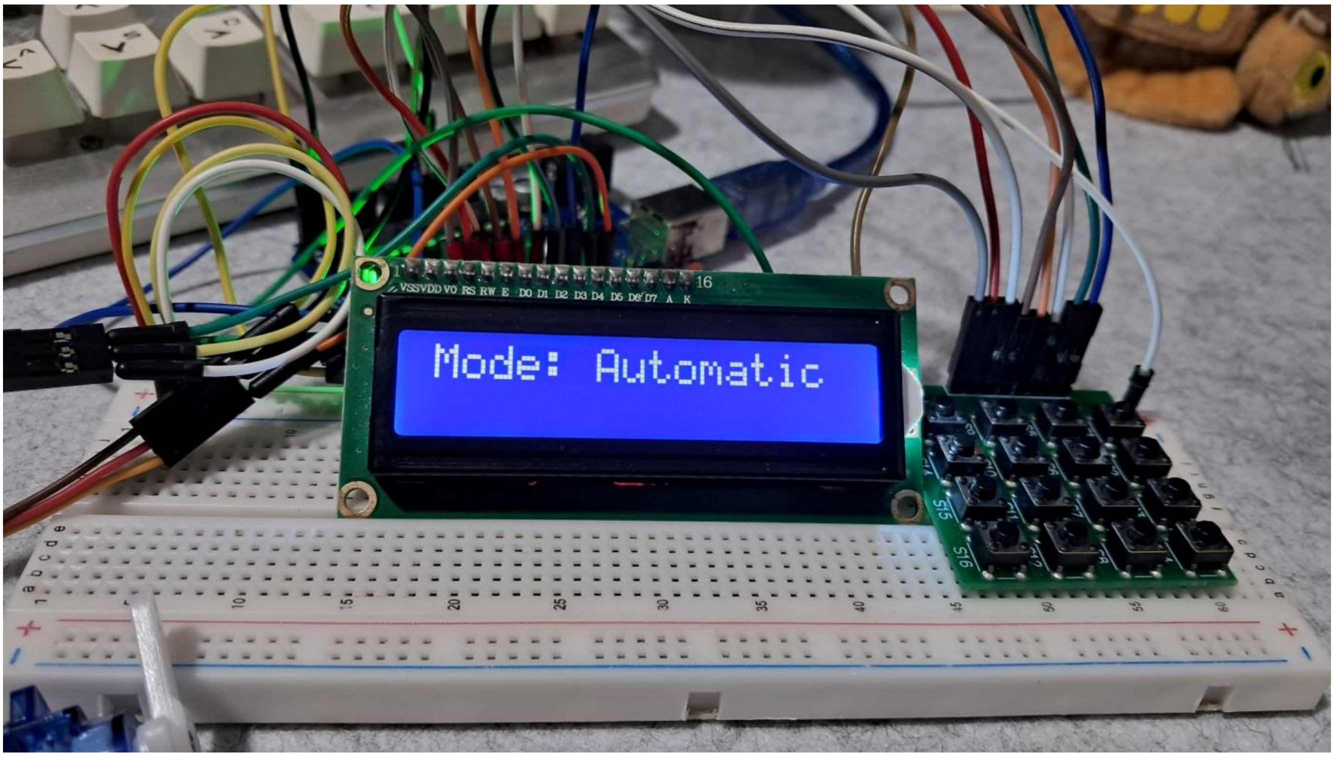


Figure 7: Automatic Mode

Part III. List down your work load

- Made sure all the components are connected correctly.
- Created the two modes, the automatic and manual mode.
- Configured the movement of the servo motors.
- Listed the pin connections for each component.
- Fix any issues in the connections.

Part IV. Gantt chart

Tasks	**02/17/2025**		**02/23/2025**		**03/2/2025**		**03/9/2025**	
Connection of the components and the creation of the two modes	🟩	🟩						
Integrate the color sensor in the system, as well as modifying the code with the color sensor.			🟨	🟨				
Create the mechanism for the whole system.			🟨	🟨				
Debugging			🟨	🟨				
Polishing the working prototype.					🟪	🟪		
Defense/Presentation							🟦	🟦