CSE 121: Lab 7 Report

By Daniel Beltran

12/08/2023

Lab 7: Part 1

During the first part of this lab, we are tasked with outputting morse code from an LED connected to our PI4. Forgetting the ESP for the time-being, using the screwdriver in our kit to open up the cannakit and bringing the PI4 board. To understand the pins on the PI4, following the documentation for the schematic helped in locating the ground and GPIO pin for building the breadboard circuit.

When starting the program for part 1, I asked ChatGPT to assist me in creating the encoding of morse language that will light up the LED connected to the PI4 GPIO pin that we send in (in my case GPIO PIN 26). The following is the list of characters (letters/numbers) that correspond to their morse output:

```
const char *MORSE_CODE[] = {
      ".-",    // A
      "-...",  // B
      "-.-.",  // C
      "-..",   // D
```

```
    ".",     // E
    "..-.",  // F
    "--.",   // G
    "....",  // H
    "..",    // I
    ".---",  // J
    "-.-",   // K
    ".-..",  // L
    "--",    // M
    "-.",    // N
    "---",   // O
    ".--.",  // P
    "--.-",  // Q
    ".-.",   // R
    "...",   // S
    "-",     // T
    "..-",   // U
    "...-",  // V
    ".--",   // W
    "-..-",  // X
    "-.--",  // Y
    "--..",  // Z
    "-----", // 0
    ".----", // 1
```

```
    "..---", // 2
    "...--", // 3
    "....-", // 4
    ".....", // 5
    "-....", // 6
    "--...", // 7
    "---..", // 8
    "----."  // 9
};
```

From here, I had asked ChatGPT to provide the template to start initiating the program and one thing that I had noticed is that it provided the use of the header file **wiringPi.h** which was difficult in itself. This header file was from the package of wiringpi that allowed for the configuration of the GPIO pins on the PI4 that would allow for the writing and displaying of power to the pins that would then light up the LED. Simply installing the package was difficult in itself as there were many obstacles that prevented the issuing of the code. Below will provide the URL to the site that allowed the installation of this onto the PI. After all that, the following is code template was provided by ChatGPT:

```
// Function to transmit a single Morse code character
void transmitMorse(char c) {
    if (c == ' ') {
        delay(500); // Space between words
    } else {
        int index = c - 'A'; // Assuming only uppercase letters for
        if (index >= 0 && index < 26) {
            const char *morse = morse_code[index];
            while (*morse) {
                if (*morse == '.') {
                    digitalWrite(LED_PIN, HIGH);
                    delay(200); // Dot duration
                } else if (*morse == '-') {
                    digitalWrite(LED_PIN, HIGH);
                    delay(600); // Dash duration
                }
                digitalWrite(LED_PIN, LOW);
                delay(200); // Inter-element spacing
                morse++;
            }
            delay(400); // Inter-character spacing
        }
    }
}
```

```
// Function to transmit a Morse code message
void transmitMessage(const char *message) {
    for (int i = 0; message[i] != '\0'; i++) {
        transmitMorse(message[i]);
    }
}
```

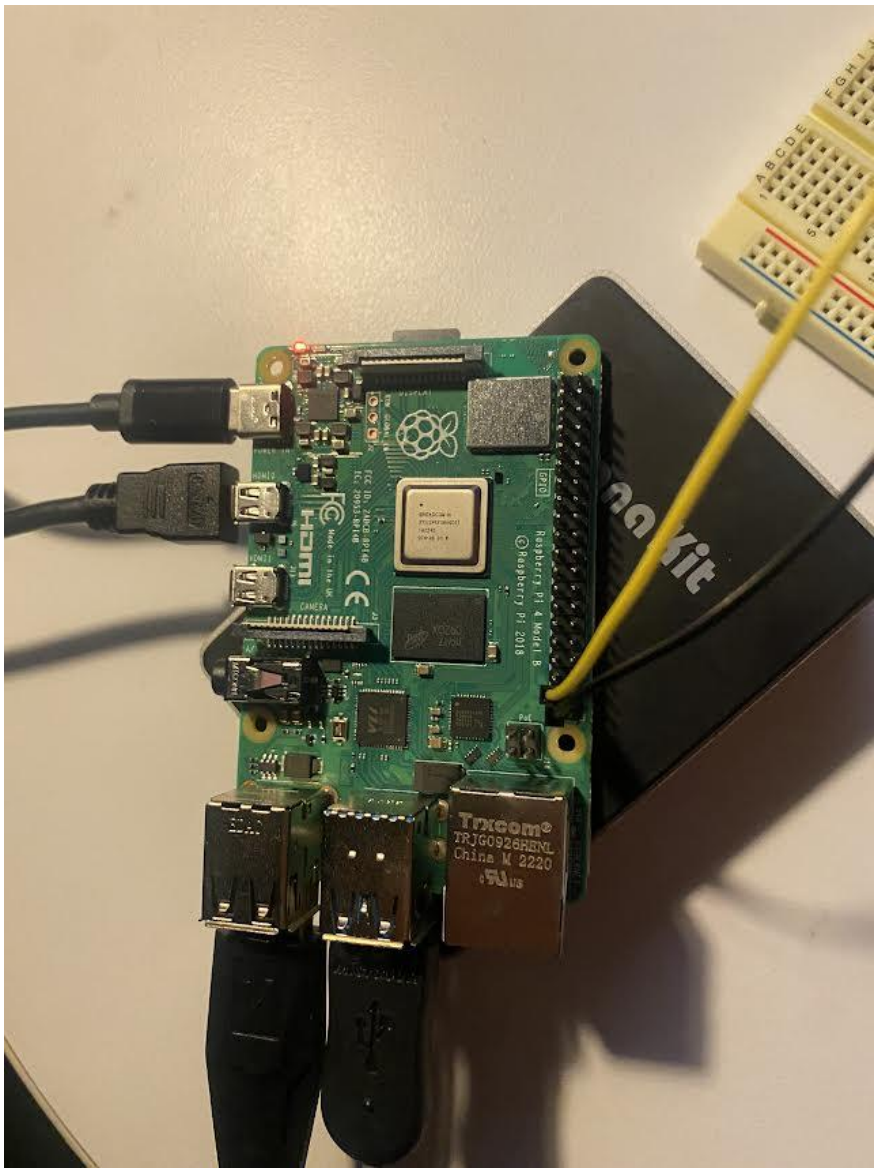
```c
int main(int argc, char *argv[]) {
    // Check if the correct number of arguments is provided
    if (argc != 3) {
        printf("Usage: %s <number_of_times> <message>\n", argv[0]);
        return 1;
    }

    // Initialize wiringPi library
    if (wiringPiSetupGpio() == -1) {
        fprintf(stderr, "Unable to initialize wiringPi\n");
        return 1;
    }

    // Set the LED pin as OUTPUT
    pinMode(LED_PIN, OUTPUT);

    // Get the number of times to send the message
    int times = atoi(argv[1]);

    // Get the message to transmit
    const char *message = argv[2];

    // Transmit the message the specified number of times
    for (int i = 0; i < times; i++) {
        transmitMessage(message);
    }

    // Turn off the LED after tran ssion
    digitalWrite(LED_PIN, LOW);
```

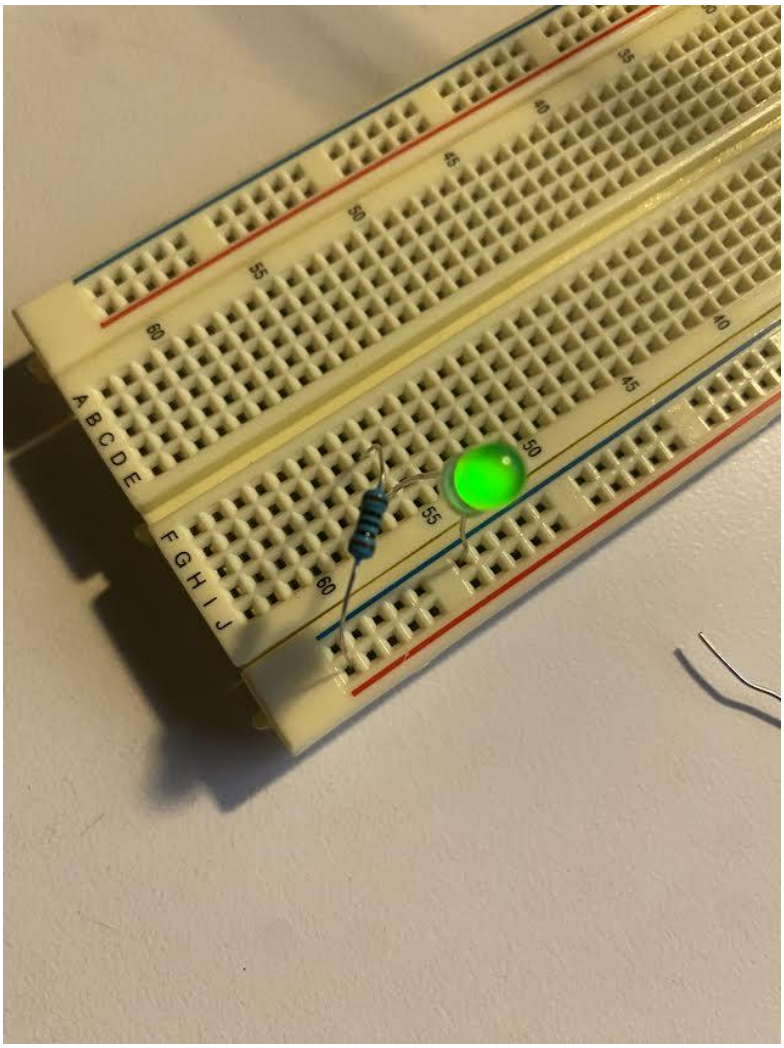
ChatGPT's starting template was really good as the **main()** and **transmit_morse_message()** function did not need to be altered for the end product.

```
53  void transmit_morse_char(char C){
54      C = toupper(C);
55      if(C == ' '){
56         usleep(2000000);
57      }
58      else{
59          int dex; // creates the index from the start of the Alphabet
60          if(isdigit(C)){
61                dex = C - '0' + 26;
62         } else if(isalpha(C)){
63                dex = C - 'A';
64         }else{
65             return ;
66         }
67      if(dex >= 0 && dex < 36){ // Set's the boundary for the alphabet
68         const char *MORSE = MORSE_CODE[dex]; // Setting more to the library
69         while(*MORSE){
70               if(*MORSE == '.'){
71              digitalWrite(LED_GPIO_PIN, HIGH);
72              usleep(500000);
73            }else if(*MORSE == '-'){
74              digitalWrite(LED_GPIO_PIN, HIGH);
75              usleep(750000);
76            }
77            digitalWrite(LED_GPIO_PIN, LOW);
78            usleep(500000);
79            MORSE++;
80         }
81          usleep(1000000);
```

If you notice, the formatting of the function stayed relatively the same. The only difference is that there is the inclusion of numbers as that was assumed to be required. Initially we had the quick if/else conditional blocks but within the else block it held the majority of the logic. Within that block, we check if our character is a digit to which it detects after the 26th character in the MORSE_CODE variable. One thing to note is that line 54 is necessary as I ran into many issues with the LED not lighting up for uppercase or lowercase (trial and error). This ensures that either lowercase or uppercase letters will be recognized as inputs for the program to execute. Other than that, doing the check for the DOT and DASH with the customization of delays to output onto the LED. In order to compile the program, simply run **gcc lab7_1**

-lwiringPi -o lab7_1 and then **./lab7_1 <number_of_iterations> <string_message>.** In order to build the breadboard circuit, the following setup allowed for the project to execute.





PI4 Documentation: https://www.raspberrypi.com/documentation/computers/raspberry-pi.html

WiringPI: http://wiringpi.com/wiringpi-updated-to-2-52-for-the-raspberry-pi-4b/

&

https://roboticsbackend.com/introduction-to-wiringpi-for-raspberry-pi/#WiringPi_on_Raspberry_Pi_OS