CS12020 Morse Assignment Report

Ethan Swain | ets6@aber.ac.uk

1 Introduction

Encoding and decoding the SACode to and from ASCII laid the groundwork for the other parts of the assignment, therefore this section was essential to complete to a high standard to minimise risks of complications when developing further code. This was because translation between each character set was required in both parts; for transmission to LED in part 2 and responding to command messages in part 3. Specifically, the translation of numbers from SACode to the ASCII equivalent, whilst responding to command messages in part 3, was a trivial task. Therefore, the use of defensive programming throughout was necessary.

2 Design

The design of my program relies on the ability to encode and decode SACode, therefore two arrays with a fixed length were required, one to store ASCII characters and one to store SACode characters. These arrays needed to be accessed throughout the program and must be correctly implemented as the assignment required only translated input to be output to the serial monitor.

As the code had to endlessly loop to listen to the next input to process from the serial port, I was unable to use many constant variables as the values had to be updated throughout the program and then reset when looping back to the beginning. This meant that I had to prioritise defensive programming throughout to ensure variables were set to correct values during the program's loops.

3 Implementation

3.1 Loops

The first line of code receives the user input from the serial port. Part 3 of the assignment brief states that command messages must be entered in SACode, therefore I implemented a nested 'if' statement which checks the user's input against the 4 potential command messages that could be entered. The code validates each character of the user input and if the combination of a command is entered, then the loop calls the function for that specific command. If no commands are entered, then the code will translate the input accordingly and output the result to the serial port and either the green or blue LED pin.

3.2 LED Transmission

To translate SACode to LED, I defined whether the user had entered ASCII or SACode so that the correct LED pin was utilised. I then read the values from the potentiometer sensor to define the time interval between LED flashes (basic potentiometer connection is shown in Figure 1). I implemented this using an 'if' statement for retrieving the correct value for a linear increase between 20ms and 400ms between 0 and 1023 on the potentiometer. As each character that makes up SACode has different timing for LED transmission, I set each character to their corresponding time unit using a 'for' loop with an 'if' statement nested inside.

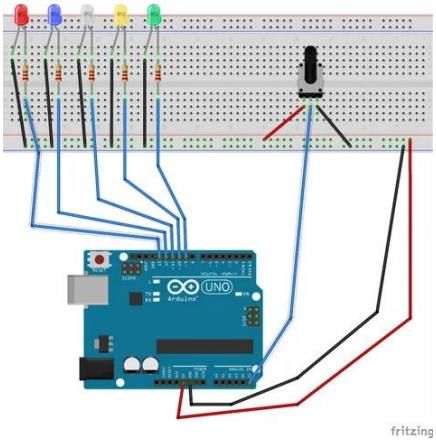

Figure 1: Outline of potentiometer connection to LEDs

I used a similar approach when coding the ‘RA’ command message, however, it utilised the potentiometer to control the brightness of the yellow LED. To define the brightness of the LED, I implemented a series of nested ‘if’ statements inside a ‘for’ loop to retrieve the 3-decimal digit number entered by the user. This integer value then determined the brightness, equivalent to the potentiometer value read. Similar to the initial transmission of SACode to LED flashes, the implementation of the ‘VF’ command meant that a number input in SACode would translate to the number of flashes on the LEDs. However, the complicated part was outputting simultaneous flashes of each LED depending on the number input (see Section 5).

4 Testing

Throughout the development of my code, I frequently tested each section to ensure that it worked as intended; this includes the use of debugging messages (which were removed in the final delivered program). I also output relevant variables to the serial port when testing specific sections. This frequent testing allowed me to have working code after the completion of every section of the assignment, however, when running through the final testing of all components, I encountered a few bugs. This final testing allowed me to look back through my code and resolve these bugs.

5 Complications Encountered

Whilst developing part 1 of the assignment, I initially created a switch case statement for converted characters into SACode. However, this method proved highly impractical and inefficient as every single character required its own case, therefore, I replaced it using a ‘for’ loop that searched through the array.

The other main complication I encountered, was the response to the ‘VF’ command. This was because I struggled to output the LED flashes simultaneously. After attempting to design and implement a variety of different approaches towards this, my final approach involved a ‘for’ loop with an ‘if’ statement nested inside. The ‘for’ loop repeated nine times until each value was equal to zero, in which it then removed the assigned pin of the specific LED the value was assigned to. This meant that each light would flash simultaneously.

6 Tasks Completed

- Encoding and decoding SACode.
- Transmission of SACode on the green LED when ASCII is entered.
- Transmission of SACode on the blue LED when SACode is entered.
- Control of LED transmission speed by the potentiometer.
- Output of IR Receiver value[2]
- Response to the ‘RA’ command using the yellow LED and potentiometer.
- Response to the ‘RIR’ command using the IR receiver.
- Response to the ‘A’ command using the green, yellow, orange, and red LEDs.
- Response to the ‘VF’ command flashing the green, orange, yellow, and red LEDs simultaneously.
- Frequent testing and final testing of code, including necessary bug fixes.

7 Conclusion

Overall, I spent an extensive time developing this assignment to ensure that each section worked correctly. This resulted in a fully working program as the outcome which followed the functional requirements specified in the assignment. However, I believe that sections of code could be made more efficient or tidier. Based on the assessment criteria, I would award myself a mark of 75% as I believe I have completed what is specified.

References

[1] *Using LEDs With a Potentiometer*. From Instructables. https://www.instructables.com/LEDS-With-a-Potentiometer/. Last access 10/12/2022.

[2] Manzoor, M. (Aug 15, 2020). *Arduino with IR Sensor*. From Medium. https://medium.com/illumination/arduino-with-infrared-sensor-48ad4415f320.