TinkerTech Labs Assignment Documentation

Name: Raj Vishal Turaga

Project Timeline

- June 09, 2021
  - GitHub repository created and started working on the project.
  - Went through the fundamental concepts and working of SSD1306.
  - Used a simulation of the system to get accustomed to working with the display device.

- June 10, 2021
  - Required components were procured to start working on.
  - Basic text displaying code was written.
  - A code to scroll the text was written where we use a for loop to change the cursor position continuously.
  - Encountered **Problem 1**.
  - Problem resolved and the code has been updated.
  - Encountered **Problem 2**.
  - Video for demonstration made and uploaded on GitHub Repository

- June 11, 2021
  - Documentation and GitHub repository submitted.
  - Encountered **Problem 3**

- June 19, 2021
  - Problem 3 was resolved.
  - Text wrapping was replaced with word wrapping.
  - Auto text scrolling was yet to be implemented.

- June 22, 2021
  - Auto text scrolling was implemented although the scrolling was not efficient as the one done on June 11, 2021.
  - Detected major issues in the scrolling algorithm (**Problem 4**)

- June 24, 2021
  - Bluetooth module HC-05 was introduced to the system and the input was taken from a mobile device which transmitted data to the Arduino Uno through HC-05 and displayed it from the SSD-1306 peripheral.
  - Problem 4 was yet to be resolved.

- June 28, 2021
  - Scrolling problem was made better. Problem 4 is resolved.
  - However, for long texts, text wrapping fails to be compatible with scrolling.

Bug Report

- **Problem 1:** Till when to scroll the screen.
    - The problem faced was that the screen was scroll down even after the text has completely disappeared and this is a problem as it is supposed to stop scrolling once the end of the text was reached.
    - To solve this problem a formula was devised to find the end value for the **for** loop used in the program.
    - It was observed that at max, every line on the SSD1306 can accommodate 20 characters and at once it can display 4 lines.
    - Since the display is a 128*64 screen and the screen can display 4 lines, each line takes 16 lines pixels (64/4 = 16).
    - We can find the length of the string and calculate the number of lines required to display the code. Let us store this value in the variable **b**.
    - Since the first lines are displayed at the start, the number of lines yet to be displayed are **b – 4**.
    - Now in order to display the complete text, the screen has to scroll the pixels (**b-4)*16** times. Let us store this value in the variable **c**.
    - Therefore, we initialize the value of the **for** loop from 0 to -c.
        - for(int i = 0; i <= -1*c; i++){
            ----Code----
            Display.setCursor(0,i);
            ----Code----}
        - This will decrement that value of y axis from 0 to -c.

- **Problem 2:** Limitations in the amount of text a variable can store.
    - Input from the serial monitor was being stored in the SRAM which has limited memory space. Hence only up to 800 characters of text is being able stored.
    - To solve this problem, the string was initially stored in PROGMEM (Program Memory). However, this failed as program memory is a read-only memory and hence writing the stored variable is not possible.
    - The next step was to store the variable in EEPROM memory and then check if it is working.
    - **Problem yet to be resolved**
- **Problem 3:** Text wrapping was implemented in the place of word wrapping. Hence, words were being cut in the middle.
    - An array of length of words in the text was stored and was compared with the available space in the current line. If the length of word is greater than the space, the cursor was moved to the text line
- **Problem 4:** Scrolling of the text was improper. The text was not scrolling up and was randomly scrolling and hence was overlapping with existing text.
    - A new method was implemented where once the screen prints 4 lines of the text, it refreshes the screen and displays the remaining content.