UNIVERSITY OF BRIGHTON

MODULE TUTOR: - Saeed Malekshahi

MODULE CODE: - CI514 Embedded Systems Sem – 2 Learning journal

STUDENT ID: - 21839630

SUBMISSION DATE: - 18/08/2023

Contents

[Introduction 3](#_Toc143261179)

[Task1 3](#_Toc143261180)

[Single light 3](#_Toc143261181)

[Using IDE 3](#_Toc143261182)

[Flow chart 3](#_Toc143261183)

[Evidence 6](#_Toc143261184)

[Two way traffic Light 6](#_Toc143261185)

[Flow chart 6](#_Toc143261186)

[Evidence 9](#_Toc143261187)

[Two Traffic Light with Pedestrian Traffic Light 9](#_Toc143261188)

[Flow chart 10](#_Toc143261189)

[Evidence 12](#_Toc143261190)

[Bubble Sort 12](#_Toc143261191)

[Flow chart 13](#_Toc143261192)

[Conclusion 14](#_Toc143261193)

# Introduction

This report presents a comprehensive overview of tasks involving assembly language programming and hardware control, showcasing the implementation of various systems. Through a combination of theoretical understanding and practical application, this report delves into the creation of traffic light control systems and the execution of sorting algorithms using assembly language. By employing the Arduino platform and inline assembly coding, this report demonstrates the ability to manipulate hardware components and program logic to achieve desired outcomes.

The tasks covered encompass a one-way traffic light system, a two-way traffic light system, and the implementation of bubble sort using assembly language. These tasks not only highlight the understanding of programming languages and hardware configurations but also showcase problem-solving skills and innovative approaches to achieving functional outcomes. The report employs visual aids such as flowcharts, code evidence, and simulations to enhance the clarity and authenticity of the presented concepts.

Throughout the report, the reader will gain insights into the practical applications of assembly language, the integration of hardware systems, and the creative problem-solving techniques employed to create functional systems. By exploring each task's execution process, modifications, and outcomes, this report offers a comprehensive perspective on the intersection of programming, hardware control, and practical implementation.

# Task1

## Single light

### Using IDE

For this undertaking, we employed the Arduino IDE to code inline assembly for a one-way traffic light system. Our chosen file format was the ".ino" file. The process involves defining functions such as "void setup" to establish the initial traffic light setup, and "void loop" to continuously execute the program logic. The IDE conveniently offers a "verify" button to compile the code and identify potential syntax or other errors. Subsequently, the "upload" button facilitates transferring the compiled code to our Arduino Mega board.

### Flow chart

This is illustration of flow chart for one-way traffic light. The execution of the program commences at the beginning, proceeding to the "setup" section where the initialization of the "r18" register takes place. Subsequently, the program advances to the "loop" section, initiating the one-way traffic light sequence. Within this loop, there are four sequential stages represented in the flow chart: "traffic light green," "traffic light amber," "traffic light red," and "red and amber traffic light." Each of these stages incorporates a function call for the "delay" function, introducing time intervals. The flow chart progression concludes at the "END" point, prompting a return to the main loop for continual operation.

A diagram of a traffic light

Description automatically generated

A diagram of a process

Description automatically generated

I've programmed the one-way traffic light control using the "setup" and "loop" functions. In "setup," I configured port 4 as an output and set its value to 0xE0. In the "loop," I divided the sequence into four phases: green light, red light, amber light, and red-amber light.

During the green light phase, I set port 5 with an output of 0x80, causing pin 13 on the Arduino to go high while others went low, illuminating the green light. A delay function follows, transitioning to the amber light phase.

In the amber light phase, I set port 5 with an output of 0x40, causing pin 14 to go high and others to go low, turning on the amber light. Another delay follows, leading to the red light phase.

For the red light phase, I set port 5 with an output of 0x20, signalling the red light to turn on, before moving to the red-amber light phase.

In the red-amber light phase, I set the output to 0x20, lighting up both the red and amber lights while keeping others off.

The "delay" is a basic nested loop designed to introduce time intervals for each phase.

### Evidence

### A circuit board with wires and a red light Description automatically generated

## Two way traffic Light

### Flow chart

The flowchart represents a two-way traffic signal system. The modifications mainly involve changes in the ports and the introduction of new signal phases. Instead of the previous signal phases like "traffic green light," "traffic amber light," "traffic red light," and "traffic red-amber light," the revised flowchart features "traffic green1 and red2 lights," "traffic 1amber and 2amber lights," "traffic red1 and green2 lights," and "traffic 1redamber and 2redamber lights." These changes accommodate the two-way traffic flow.

These adjustments also involve altering the corresponding port configurations to suit each unique process within the traffic signal sequence.

A diagram of a traffic light

Description automatically generated

A diagram of a flowchart

Description automatically generated

The two-way traffic light system is an advancement from the one-way traffic light. In the "void setup" function, we expanded to 0xFC, providing six outputs compared to the three in the one-way system. This facilitates simultaneous output of traffic lights in the loop, with sequences for both traffic lights 1 and traffic lights 2.

For traffic lights 1 and 2 sequences, the configurations are as follows:

- Traffic green1 and red2 lights use 0x90.

- Traffic amber1 and amber2 lights use 0x48.

- Traffic red1 and green2 lights use 0x24.

- Traffic redamber1 and redamber2 lights use 0x78.

The output pins on the Arduino are assigned as follows:

- Green1 light is connected to pin 13.

- Amber1 light is connected to pin 12.

- Red1 light is connected to pin 11.

- Red2 light is connected to pin 10.

- Amber2 light is connected to pin 50.

- Green2 light is connected to pin 51.

### Evidence

A circuit board with wires and lights

Description automatically generated

## Two Traffic Light with Pedestrian Traffic Light

In the enhanced version of the two-way traffic signal system featuring a pedestrian cross, the design incorporates 8 additional outputs on the port. Furthermore, an interrupt mechanism has been established through serial communication. Upon receiving a "Start" signal via the serial port, the interrupt is triggered. This transition activates the pedestrian cross light, illuminating it in green. Simultaneously, traffic light 1 and traffic light 2 shift to red, ensuring a secure passage for pedestrians. This integrated system ensures pedestrians can safely navigate the crossing, promoting a more efficient and secure traffic management solution.

The code begins by setting up serial communication at a baud rate of 9600 and initializing some hardware using inline assembly code. This hardware initialization involves loading the value 0xFF into register r18 and then outputting it to a specific port (port 4). This section is outside the loop, executed only once during setup.

Inside the loop, the program continuously checks if there is any serial input available. If input is received, it reads a character from the serial input. If the character is 'start', it sends the message "start" over serial communication and performs a sequence of hardware manipulations using assembly code. These manipulations involve loading the value 0x31 into r18 and outputting it to another port (port 5), followed by multiple calls to a delay subroutine.

If the input character is not 'start', the program sends an "Invalid input" message over serial. Regardless of the input, the program then enters four phases of hardware manipulation, each represented by a set of assembly instructions that load specific values into r18 and output them to port 5. After each phase, a delay subroutine is called to introduce a delay

The delay subroutine uses nested loops to create a delay. It initializes three loop counters (r23, r24, r25) and executes nested loops to decrement these counters. This nested loop structure creates a delay in the program execution.

### Flow chart

A blue rectangular object with white text

Description automatically generated

A diagram of a flowchart

Description automatically generated

### Evidence

A circuit board with wires and wires

Description automatically generated

# Bubble Sort

The second task of the given assessment is running bubble sort using assembly language and since the bubble sort is based upon swapping the original number where some random numbers are generated and then based on the caparison between two consecutive numbers the swapping is done to implement it using assembly language a large bit array is initialized where a number is store in each of the bit a unique pin is assigned to each number where the pin is started from r25 PORTB and all the number are inserted in the same manner they are generated and then a comparison between two pin value is performed and in case the left value is greater the swapping of the number is performed otherwise the loop continues and this function is performed for total O(2\*n) time where the n is the total element in the array generated and an additional function swapping is created and to check whether the sorting is performed as expected the output of the value is also printed in the console or can be re-directed to the LED Screen, but since LED screen has a limited dimension and total 50 random are generate at each iteration therefore the output is shown the IDE itself and in order to apply swapping between two number the r19 and r20 pin are used where the XOR operation is performed and the value are swap based on the XOR value and in order to run the iteration and external interrupt is required which will trigger the algorithm where the external interrupt is the push button which will automatic run the bubble sort iteration be generating some random number, and for the conversion of the number , they are firstly converted to hex value since assembly language can only process heximal number. The process flow diagram of the above mention sorting algorithm and a screenshot of the simulation is shown below along with which the video presentation of the running simulation is also present in the folder Bubble-sort.

## Flow chart

A diagram of a process

Description automatically generated

EVIDENCE

A screenshot of a computer

Description automatically generated

A computer screen shot of a computer

Description automatically generated

# Conclusion

In conclusion, the report showcases successful implementation of various tasks involving assembly language programming and hardware control. It begins with a one-way traffic light system,

demonstrated through Arduino IDE and inline assembly code. The flowchart effectively illustrates the program's structure.

Moving on, a two-way traffic light system is presented, highlighting modifications and expanded configurations for simultaneous traffic control. The allocation of pins and changes in port configurations are clearly explained.

The report then covers a more complex task, bubble sort, implemented in assembly language. The unique approach of using pins for value storage, comparison, and swapping is described, along with an external interrupt mechanism for initiation.

Throughout the report, visual aids such as flowcharts, evidence of implementation, and simulations enhance understanding. These elements contribute to clear communication of technical concepts and practical application. Overall, the report demonstrates adeptness in assembly programming, hardware manipulation, and logical problem-solving. Through concise explanations and visual support, it effectively conveys the successful execution of each task.