

F28HS  
(Hardware-Software interface)  
Coursework2 - Mastermind  
Group 6  
2023-24

Mufliha Shake Dawood  
(H00420904)  
[ms2176@hw.ac.uk](mailto:ms2176@hw.ac.uk)

Shyam Sundar Velmurugan  
(H00418621)  
[ssv2001@hw.ac.uk](mailto:ssv2001@hw.ac.uk)

# Problem Specification

The problem specification for this task is to develop a simple instance of the Mastermind board game as a systems programming application in C and ARM Assembler.

The application should run on a Raspberry Pi with attached devices such as 2 LEDs (Green & Red), button, LCD display and potentiometer.

The application should generate a random secret sequence of colored pegs, and the user should attempt to guess the sequence by entering sequences of numbers (representing colors) using the button input. The application should provide feedback on the guesses, indicating the number of exact matches (right color and position) and approximate matches (right color but wrong position).

The game continues until the user correctly guesses the secret sequence or a fixed number of turns is reached.

## Hardware Specifications and The Wiring

The hardware systems that has been used in this project are:

- Raspberry Pi 3
- Two LEDs (Red and Green)
- 1 Button
- 1 16x2 LCD display
- 3 Resistors
- 1 Potentiometer
- 1 Breadboard

The button, two LEDs, a potentiometer and an LCD screen are connected in the breadboard to the Raspberry Pi using the jumper wires. The LCD screen is connected in the breadboard itself and provides the required connections with the Raspberry Pi using the male - female wires and the connection of wires from the LCD screen to the other breadboard connections within itself are done with male-male jumper wires.

The potentiometer is being used for adjusting the screen's contrast of the LCD screen so that the output can be read from the screen clearly. The potentiometer consists of 3 legs in which two are connected for the power and the one middle leg is connected to the LCD screen's 3rd pin.

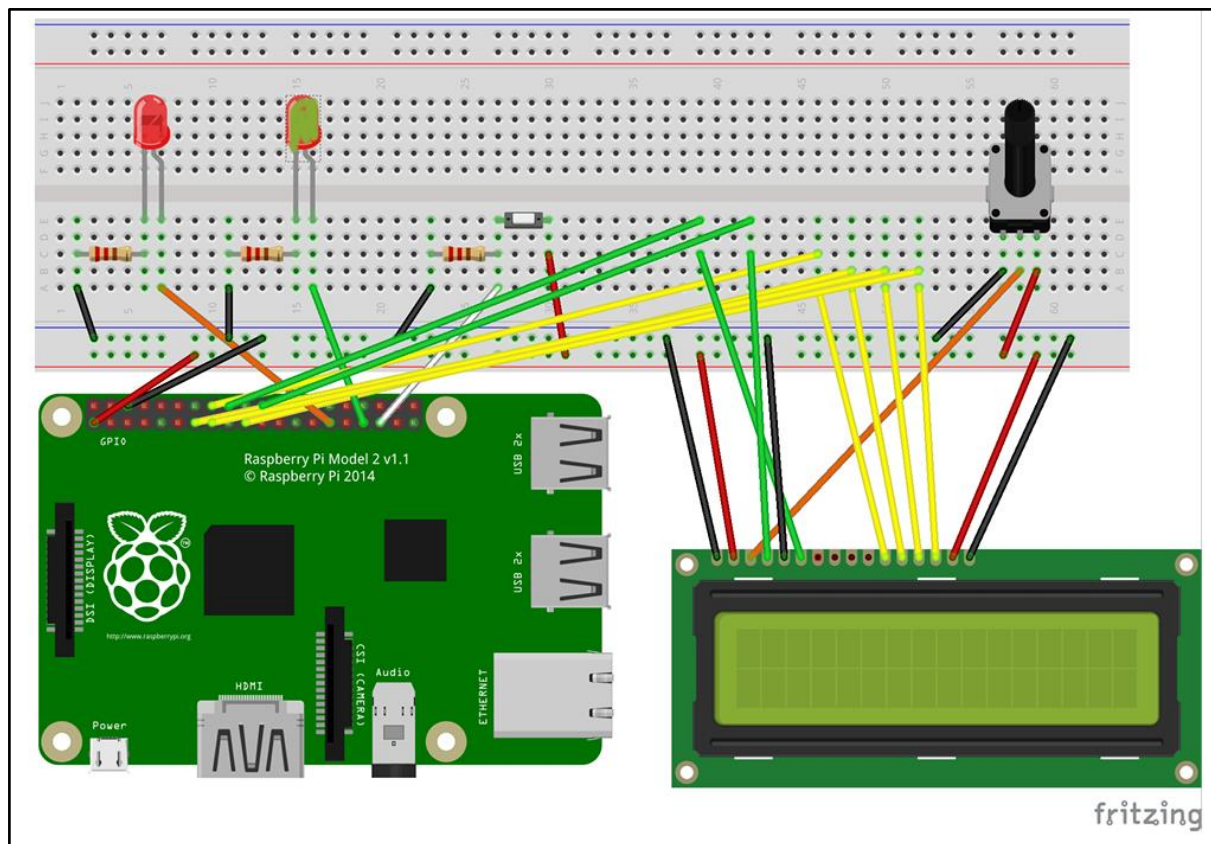
The LCD screen is used for displaying the welcome message, the attempt number, the 3 guesses we have done in each attempt, the exact and approximate guesses after each attempt,

the success message if we have guessed all 3 of them correctly or the sequence not found/better luck next time message if the user has exhausted all their attempts.

| LCD    | GPIO            | LCD        | GPIO       |
|--------|-----------------|------------|------------|
| 1      | (GND)           | 9          | (unused)   |
| 2      | (3v Power)      | 10         | (unused)   |
| 3      | (Potentiometer) | 11 (DATA4) | 23         |
| 4 (RS) | 25              | 12 (DATA5) | 10         |
| 5 (RW) | (GND)           | 13 (DATA6) | 27         |
| 6 (EN) | 24              | 14 (DATA7) | 22         |
| 7      | (unused)        | 15 (LED+)  | (3v Power) |
| 8      | (unused)        | 16 (LED-)  | (GND)      |

*Figure 1. Wiring of LCD to GPIO*

The Raspberry Pi is used as a powerhouse for all the connections made in the breadboard. The Raspberry Pi has been virtually connected using the ssh host in Visual Studio Code, by getting the IP address of Wi-Fi the Raspberry Pi is connected to. We have used RealVNC Viewer as an alternative to duplicate the Raspberry Pi display.



*Figure 2. Overall setup of the Raspberry Pi, LCD screen and the breadboard.*

The button and two LED's, connected with capacitors are used for getting the input from the user and guiding through each process during the whole game respectively. Both LEDs work as follows:

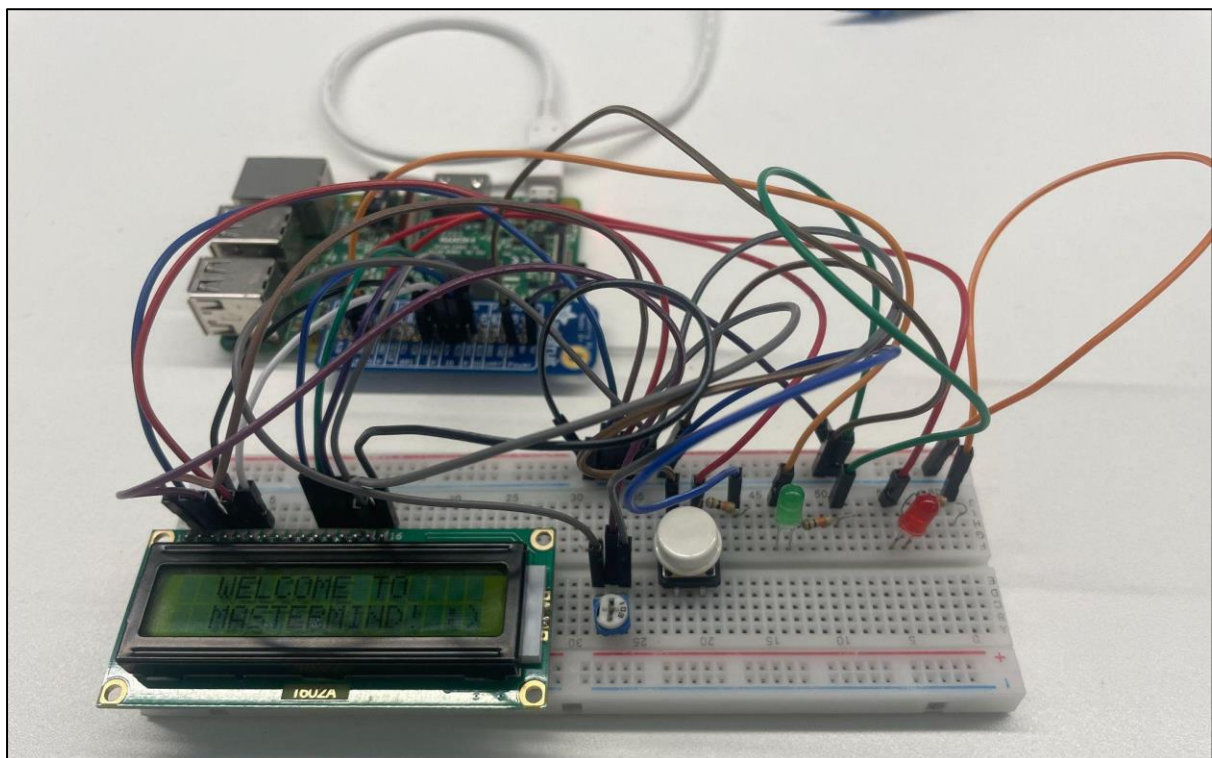
| Hardware Component | GPIO   |
|--------------------|--------|
| Green LED          | Pin 13 |
| Red LED            | Pin 5  |
| Button             | Pin 19 |

### THE PROCESS OF RECEIVING THE GUESSES FROM THE USER

After each button click, the red LED blinks once as a signal for the user to stop, then the green LED blinks according to the number of clicks, marking the user's guess. Each round begins with the red LED blinking three times.

### THE PROCESS OF SHOWING THE OUTPUTS

Initially, the green LED blinks for exact guesses and then blinks again for approximate guesses after a single blink from the red LED. If the combination is correct, both green LEDs blink three times, while the red LED remains illuminated.



*Figure 2. Overall setup of the Raspberry Pi, LCD screen and the breadboard.*

## CODE STRUCTURE

### initSeq

- Initialises the secret sequence of 3 colors, allocating memory dynamically for the sequence and srand has been and inbuilt functions present in the C library.
- In the function, it sets a random number generator with the help of using current time as the seed value, which assists in generating distinct random numbers.
- For iterating over the sequence array and assigning the randomly generating int to each other, a for loop has been implemented.

### showSeq

- It hides the integer assigned and displays the random sequence of colors on the terminal.
- For iterating over the sequence and printing each element, a for loop has been implemented.

### countMatches

- This function is used for counting the number of exact and approximate matches in which are being generated between the sequence guessed by the user and the randomly generated sequence by the Raspberry Pi.
- A pointer is being returned as an integer after it accepts two int arrays as an argument.
- Exact match: for iterating over both arrays until either of them or both of them to be completed, a while loop has been implemented. Inside the while loop, if the current element of the  $seq1 = seq2$ , it's considered as an exact match.
- Approximate match: If  $seq1 \neq seq2$ , a nested while loop is being used for searching a matching element in the  $seq2$  that has not been checked yet, and if that match is true, it will be considered as an approximate match.

### showMatches

- The variables assigned for the exact and approximate matches are printed on the terminal.

### readSeq

- For processing the command-line with options -s or -u, we use this function which parses an integer value as a list of digits and inserts them into the sequence.

### initTimer

- The function is being used for setting a timer to generate a 'SIGALRM' signal after a specified timeout by calling the 'setitimer' system.

### blinkN

- This function deals with the blinking of LEDs which are connected to the specified GPIO pin for c number of times.
- By using the digitalWrite function, LED working is switched on and off, and a delay function has been defined for the time between its on and off.

### main

- This function begins with all the variable declarations for different settings and choices, LCD configuration, the unit testing options and hardware pins.
- For controlling the timing of button pressing and LEDs blinking for visual feedback, the program uses timers.
- The game starts by displaying the welcome and other contents on the LCD. Then it initializes the sequences. The for loop counts the number of times button is being pressed and a while loop is implemented for setting a timer for the button input and also a condition has been kept that the maximum button click per input is 3.
- The matches are counted, and results are stored in a pointer variable, after receiving inputs from the user in which the red LED is used as a separator for the green LED to print first the

exact matches and later the approximate matches. Both exact and approximate matches are printed and displayed on the LCD.

- After every round, if the exact match is 3, the sequence is considered as a found one and the LCD displays an “SUCCESS!” message along with the number of attempts.
- If the exact match is not 3 and even at the end of maximum 10 attempts, the LCD displays “SEQUENCE NOT FOUND , BETTER LUCK NEXT TIME!” message and game ends.

## **DESIGN CHOICES**

### Assembly

- We have used inline assembly, which has been used in lcdBinary files to make hardware more efficient and accurate, and also gets access to hardware.

### Breadboard connection

- We have attached the LCD display into the bread board itself.
- The connections from Raspberry Pi to the LCD screen are done directly instead of the connection provided in the coursework description.
- A male-male jumper wire has been connected from one side of the breadboard to another so that power can be provided to the LCD display.
- Since our potentiometer could not adjust the contrast of the LCD display at 3.3V, we connected it to 5v and later adjusted the potentiometer to the required contrast in the LCD display.

### Timer

- A timer has been given for the user button inputs and the time interval given between each blinking of LEDs.
- Delay is being used for stopping the code temporarily from running the following line, thereby making it easier for the user to play the game.

### Pointers

- They are being used inside countMatches for storing exact and approximate matches.
- They are also used for theSeq in allocating memory for the sequence.

## **FUNCTIONS ACCESSING THE HARDWARE**

### digitalWrite

- This is responsible for setting specific pin of GPIO port for given value (HIGH or LOW)
- It calculates offset value based on a value parameter:  
If value is LOW, offset is set to 10  
If value is HIGH , offset is set to 7.
- It calculates the address of GPIO that must be modified by adding offset to base address and storing into register R0.
- Then it loads the pin number into register R1 and performs a bitwise AND operation to ensure the pin is in the range of 0-31.
- After that it calculates the bit mask and stores it in register R2. Once that is done, it later stores calculated value into GPIO using the R0 address.
- Finally, it stores the output value into variable output and returned in the format of 1 or 0, which is HIGH or LOW respectively.

### pinMode

- This sets the mode (INPUT or OUTPUT) of specific pin of a GPIO port.
- It first calculates the function select register and then shifts the value based on the pin number.
- If mode is OUTPUT, it loads GPIO base address into register R1, calculates the address of GPIO register that must be modified and performs bitwise operations for setting the pin to output mode.
- If mode is INPUT, it then performs similar operations to set pin to input mode.
- If mode is neither INPUT nor OUTPUT, it prints an error message “Invalid PIN Mode”.
- The output variable is not used in this function as its value is set inside the inline assembly block.

#### writeLED

- This is responsible for writing a value to the specific LED's pin of a GPIO port.
- It first determines offset value based on parameter:  
If value is LOW, offset is set to 10  
If value is HIGH, offset is set to 7.
- It loads GPIO base addresses into the R1 register.
- Calculates the address of GPIO that must be modified by adding offset to base address and later storing them into R0.
- Loads the LED pin number into register R1 and performs bitwise AND operations to ensure the LED pin number is within the range of 0-31.
- After that it calculates the bit mask and stores it in register R2. Once that is done, it later stores calculated value into GPIO using the R0 address
- This output is not used outside of the function as its only required for inline assembly block.

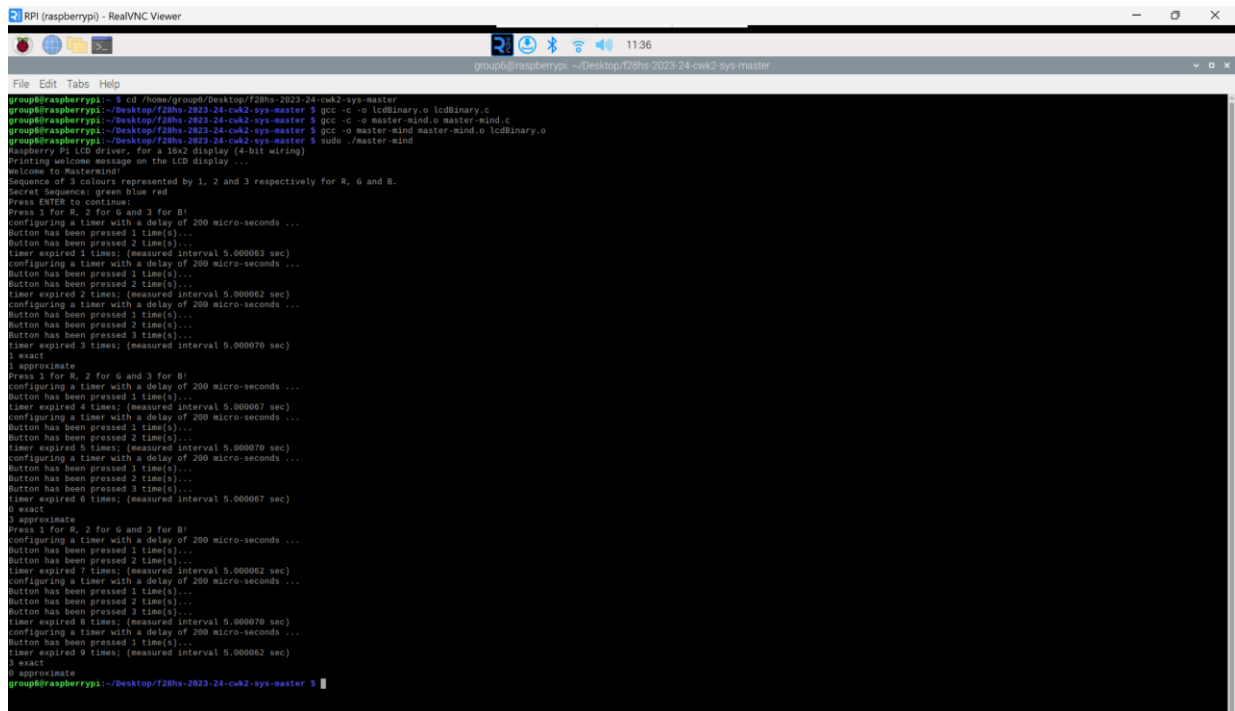
#### readButton

- This function reads the state of the button connected to a GPIO pin.
- First, it initializes a variable 'status' which stores the state of the button.
- It loads values from the GPIO register offset by 0x34 (52 in decimal) into register R2. This offset likely corresponds to the input pin state register.
- After that it calculates the bit mask and stores it in register R4. And then, performs bitwise AND operation between value in R2 and bit mask in register R4, storing result in 'status' variable.
- The output is returned in Boolean type indicating the button is pressed or not.

#### waitButton

- This code waits for a button press by repeatedly checking the button state using `readButton`.
- It waits for a certain duration defined by `DELAY` between checks. If the button is pressed within 13 iterations, it stops waiting.
- Otherwise, it continues until the maximum number of iterations is reached.

## EXECUTION OF THE GAME



```
group6@raspberrypi:~$ cd /home/group6/Desktop/f28hs-2023-24-cwk2-sys-master
group6@raspberrypi:~/Desktop/f28hs-2023-24-cwk2-sys-master$ gcc -c -o lcdBinary.o lcdBinary.c
group6@raspberrypi:~/Desktop/f28hs-2023-24-cwk2-sys-master$ gcc -c -o master-mind.o master-mind.c
group6@raspberrypi:~/Desktop/f28hs-2023-24-cwk2-sys-master$ gcc -o master-mind master-mind.o lcdBinary.o
group6@raspberrypi:~/Desktop/f28hs-2023-24-cwk2-sys-master$ sudo ./master-mind
Raspberry Pi LCD driver, for a 16x2 display (4-bit wiring)
Printing welcome message on the LCD display ...
Welcome to Mastermind
Sequence of 3 colours represented by 1, 2 and 3 respectively for R, G and B.
Secret Sequence: green blue red
Press ENTER to continue:
Press 1 for R, 2 for G and 3 for B!
configuring a timer with a delay of 200 micro-seconds ...
Button has been pressed 1 time(s)...
Button has been pressed 2 time(s)...
timer expired 1 times: (measured interval 5.00003 sec)
configuring a timer with a delay of 200 micro-seconds ...
Button has been pressed 1 time(s)...
Button has been pressed 2 time(s)...
timer expired 2 times: (measured interval 5.00002 sec)
configuring a timer with a delay of 200 micro-seconds ...
Button has been pressed 1 time(s)...
Button has been pressed 2 time(s)...
timer expired 3 times: (measured interval 5.00070 sec)
1 exact
0 approximate
Press 1 for R, 2 for G and 3 for B!
configuring a timer with a delay of 200 micro-seconds ...
Button has been pressed 1 time(s)...
timer expired 4 times: (measured interval 5.00007 sec)
configuring a timer with a delay of 200 micro-seconds ...
Button has been pressed 1 time(s)...
Button has been pressed 2 time(s)...
timer expired 5 times: (measured interval 5.00070 sec)
configuring a timer with a delay of 200 micro-seconds ...
Button has been pressed 1 time(s)...
Button has been pressed 3 time(s)...
timer expired 6 times: (measured interval 5.00007 sec)
0 exact
3 approximate
Press 1 for R, 2 for G and 3 for B!
configuring a timer with a delay of 200 micro-seconds ...
Button has been pressed 1 time(s)...
Button has been pressed 2 time(s)...
timer expired 7 times: (measured interval 5.00002 sec)
configuring a timer with a delay of 200 micro-seconds ...
Button has been pressed 1 time(s)...
Button has been pressed 2 time(s)...
timer expired 8 times: (measured interval 5.00070 sec)
configuring a timer with a delay of 200 micro-seconds ...
Button has been pressed 1 time(s)...
timer expired 9 times: (measured interval 5.00002 sec)
1 exact
0 approximate
group6@raspberrypi:~/Desktop/f28hs-2023-24-cwk2-sys-master$
```

## SUMMARY AND LEARNING OUTCOMES

We successfully completed the Master-mind game project on the Raspberry Pi, utilizing both C and assembly languages. This project taught us a lot about programming at a low level and how the hardware interacts with the operating system.

By implementing various functions, we gained insights into bit-level programming and learned how different hardware components are connected to specific memory locations. We also had the opportunity to manually connect wires and hardware components, which further deepened our understanding of these concepts.

Overall, this project was a valuable learning experience that enhanced our knowledge of programming and hardware systems.

## VIDEO ONE DRIVE LINK

[Group6 HSI](#)