F28HS – Coursework 2

MasterMind Game

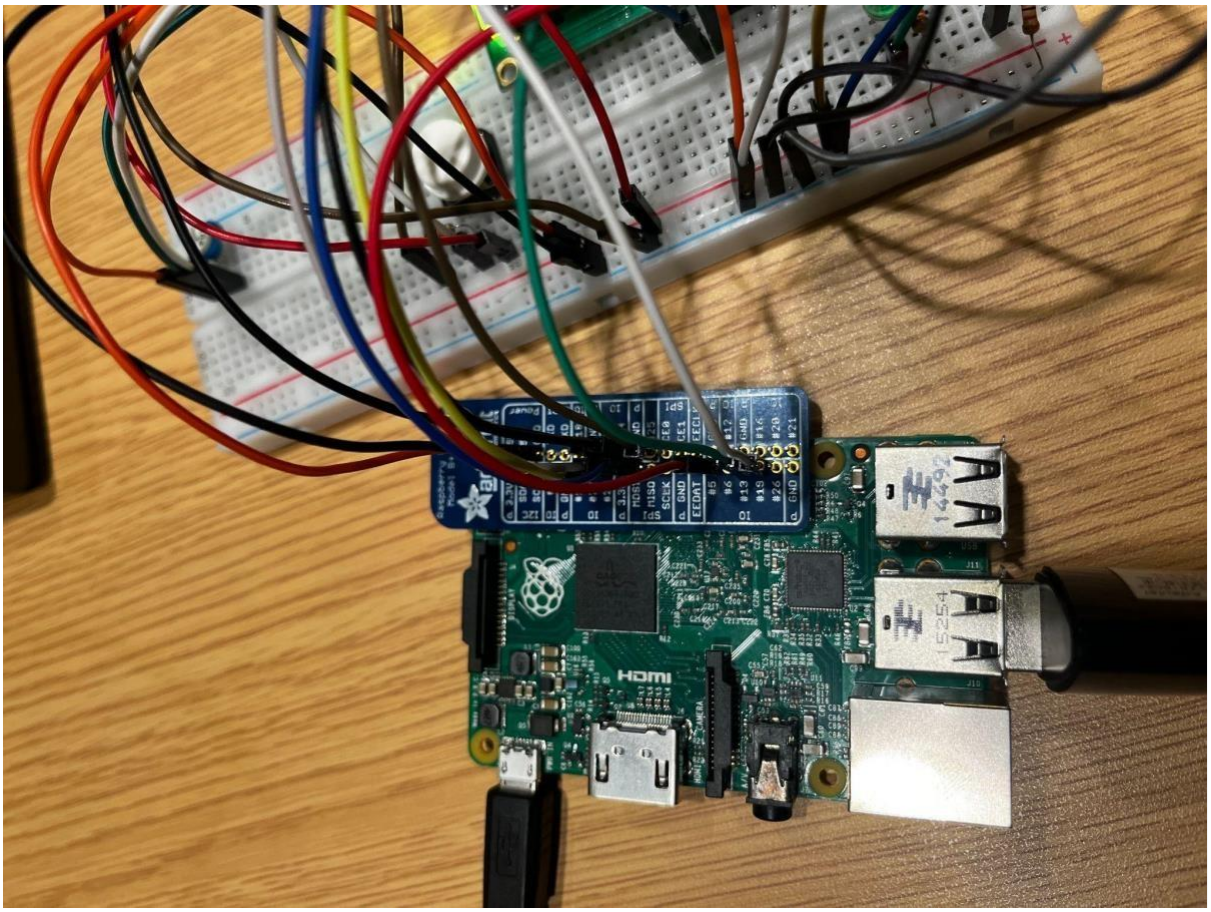
April 5 2024

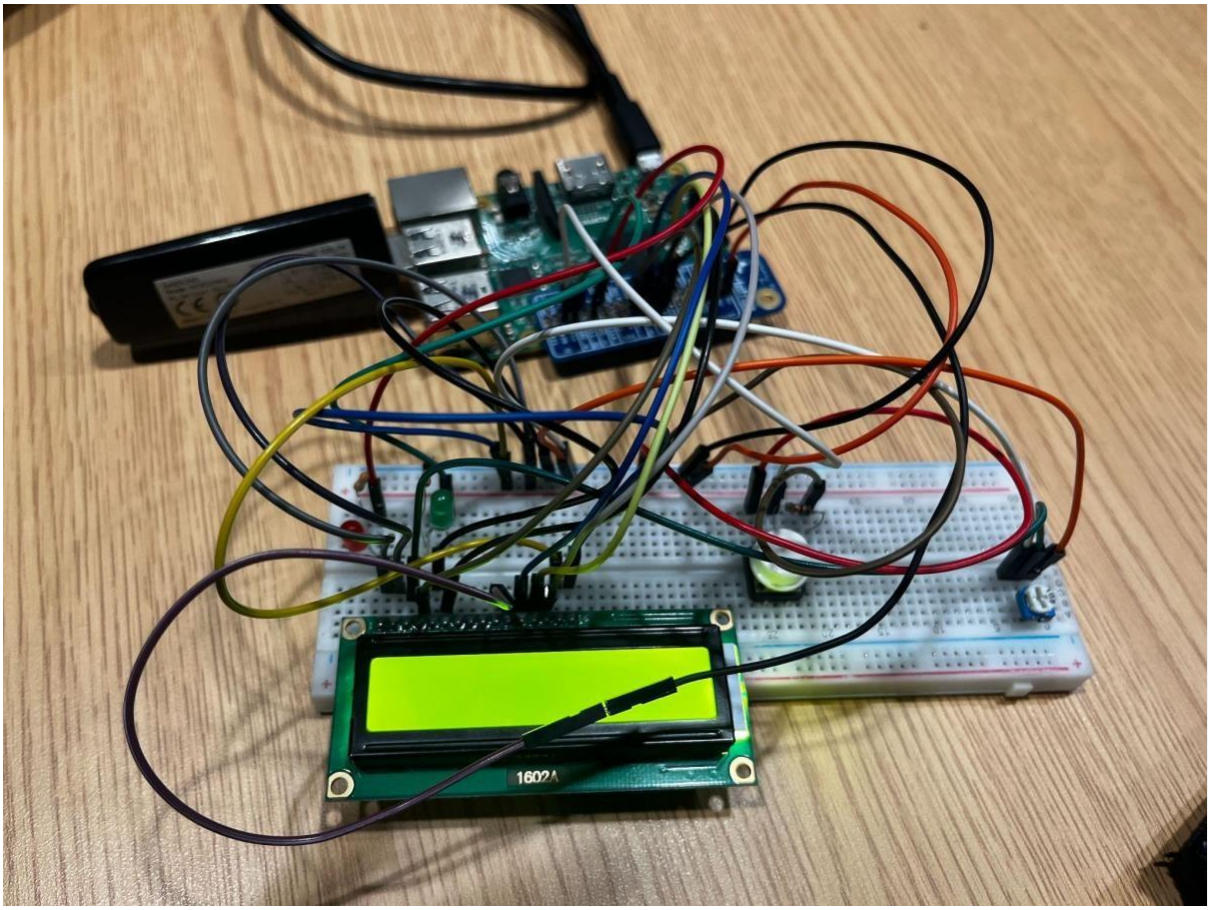
Joseph Cherian Kariampally (H00417153)

1.A short problem specification

This project aims to develop a MasterMind game application that runs on our Raspberry Pi 2 and is written in C and ARM Assembler. The application functions as the code keeper, generating a random sequence for the player (codebreaker) to predict. The player's interaction with the game is aided by hardware peripherals such as LEDs, a button, and an LCD display.

2.The hardware specification and wiring that is used as hardware platform





We have used 2 LEDs: a green and red one which are connected to GPIO pins 13 and 5 respectively. The button is used as an input device to take in the number of values from 1 to 3 which is connected to GPIO pin 19. The LCD is the output device which displays the information of the game. The LCD 1 and LCD 16 are connected to ground and LCD 2 and LCD 15 are connected to power. We have used 5 volts of power instead of 3.3 volts as our LCD isn't displaying with the said number of volts. LCD 4 and 6 connected to GPIO pins 25 and 24 respectively are used for control. LCD 11 to 14 is used for data transmission. LCD 11 is connected to GPIO pin 23, LCD 12 is connected to GPIO pin 10, LCD 13 is connected to GPIO pin 27 and LCD 14 is connected to GPIO pin 22. Additionally, there is a potentiometer as well which controls the contrast of the characters in the LCD. The potentiometer has 2 wires connected to power and ground and a wire in the middle needs to be connected to LCD 3. All these are connected to the breadboard.

3. Functionality of the main functions.

digitalWrite() - is used to set the state whether HIGH or LOW of a GPIO pin.
pinMode() - sets the mode of the GPIO pin as INPUT or OUTPUT. writeLED()
– sends a signal of HIGH or LOW to specific GPIO pins.
readButton() - read the state of button value (ON or OFF) connected to a specific GPIO pin.
waitForButton() - waits for the button press on that specific GPIO pin.
readSeq() - takes an integer value and parses it and parses it into a list of values storing each of these digits in an array called Seq.
showSeq() - display the sequence on the terminal window countMatches() - This function takes two arrays(Seq1 and Seq2) and compares their elements to count matches. initSeq() - initialises the secret sequence and by default it will be a random sequence. showMatches() - It displays the results of countMatches.
readNum() - It reads a sequence of integer from standard input and stores them in an array
blinkN() - Blinking the LED on the pin.
timer_handler() - this is a part of the timer functions and it counts the number of times a timer expires, computes the time gap between start and stop times, outputs the count and interval.
initialTimer() - this initialises time-stamps, set up an interval timer, and install the timer_handler callback.

4. A discussion of performance-relevant design decisions, and implications on resource consumption

- We have used dynamically allocated arrays in countMatches using the arrays Seq1 and Seq2 which is beneficial not only for the dynamic resizing but also for efficient memory management and reducing memory wastage by not allocating memory for any extra elements.

- We have used constants like MAX_ATTEMPTS in countMatches which will improve code readability and maintainability.

- We implemented mm-matches and lcdBinary directly in assembly language and inline assembler, respectively, to achieve faster execution compared to their counterparts.

5. List of functions directly accessing the hardware (for LEDs, Button, and LCD display) and which parts of the function use assembler and which use C

List of functions implemented in master-mind.c is written in C. These are LCD functions directly accessing the LCD pins to display information on the screen.

lcdPuts() lcdPutChar()
lcdCursor()
lcdCursorBlink()
lcdDisplay() lcdClear()
lcdPutCommand()
lcdPut4Command()

The lcdBinary.c file contains a list of functions accessing hardware implemented using inline assembler

digitalWrite()
readButton()
waitForButton()
pinMode() writeLED()

6.The name and an interface discussion (what are the inputs, what is the output of the sub-routine) of the matching function implemented in ARM Assembler.

It is taking the mm-matches.s file and checking and testing it.

```
raspberry@raspberry:~/Desktop/jk2090/CW2 $ sh ./test.sh
Cmd: ./master-mind -u 123 321
Output:
1 exact
2 approximate
Expected:
1 exact
2 approximate
.. OK
Cmd: ./master-mind -u 121 313
Output:
0 exact
1 approximate
Expected:
0 exact
1 approximate
.. OK
Cmd: ./master-mind -u 132 321
Output:
0 exact
3 approximate
Expected:
0 exact
3 approximate
.. OK
Cmd: ./master-mind -u 123 112
Output:
1 exact
1 approximate
Expected:
1 exact
1 approximate
.. OK
Cmd: ./master-mind -u 112 233
Output:
0 exact
1 approximate
```

```
Cmd: ./master-mind -u 112 233
Output:
0 exact
1 approximate
Expected:
0 exact
1 approximate
.. OK
Cmd: ./master-mind -u 111 333
Output:
0 exact
0 approximate
Expected:
0 exact
0 approximate
.. OK
Cmd: ./master-mind -u 331 223
Output:
0 exact
1 approximate
Expected:
0 exact
1 approximate
.. OK
Cmd: ./master-mind -u 331 232
Output:
1 exact
0 approximate
Expected:
1 exact
0 approximate
.. OK
Cmd: ./master-mind -u 232 331
Output:
1 exact
0 approximate
Expected:
```

```
Output:
1 exact
0 approximate
Expected:
1 exact
0 approximate
.. OK
Cmd: ./master-mind -u 312 312
Output:
3 exact
0 approximate
Expected:
3 exact
0 approximate
.. OK
10 of 10 tests are OK
raspberry@raspberry:~/Desktop/jk2090/CW2 $
```

7. An example set of output from the application running a complete game

The player at the beginning will be asked to press the button, and upon pressing the button, they will be prompted to a message stating "Welcome" along with the number of rounds in which they are playing and the number they will be inputting. As an example, we can take the secret sequence to be 3 3 2. The LCD will display 'number 1,' asking the user to input a number through the button clicks. The red LED blinks, accepting the input, and the green LED blinks as a result of echoing the input value (if the button has been pressed three times, the green LED will blink thrice). The game will continue for a total of 5 rounds. Once all the 3 numbers have been inputted, the green LED first blinks the number of exact matches to the secret code, and the red LED blinks as a separator, and then the green LED blinks again with the number of approximate matches. With each round, the LCD displays the appropriate message, and when they finally succeed, the LCD displays the number of rounds it took them to win.

8. A summary, covering what was achieved (and what not), outstanding features, and what you have learnt from this coursework

Throughout the project, we effectively developed a Mastermind game with the Raspberry Pi 2 and its accompanying components. We programmed the game using C and assembly languages, with the Pi functioning as the code keeper and encouraging players to guess the secret sequence in order to complete the game. Meeting all of the essential standards, we were able to learn about how each component is connected to one another, as well as the fundamental principles related to electricity and wiring. Furthermore, this project gave great experiences with learning in C and assembly code, improving our skills in compiling and merging both languages to create a working game.