HERIOT WATT UNIVERSITY

UK | DUBAI | MALAYSIA

Coursework 2
Systems Programming
Mastermind Application

F28HS

Hardware-Software Interface

2023-24

GROUP - 29

Mohamed Ihsan Fazal - H00452069

mf2056@hw.ac.uk

Sri Sai Vaishnavi Chintha - H00452920

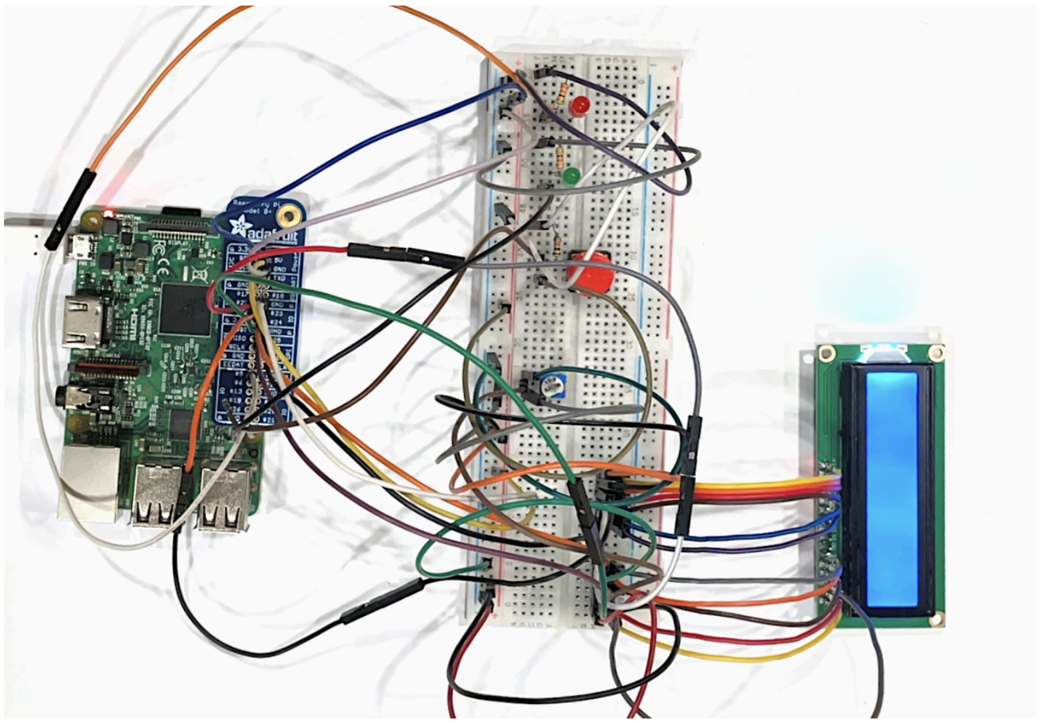
svc2000@hw.ac.uk

Problem specification:

MasterMind is a 2 player game developed as a systems-level application in C and ARM Assembler, running on a Raspberry Pi with attached devices. One player (The application) works as a code keeper while the other works as the codebreaker. The application has a secret code which is supposed to be figured out by the codebreaker. In turns, the codebreaker tries to guess the hidden sequence, using the numbers 1,2,3. In each turn, the application shows how many digits are exact and approximately correct. The game ends when the codebreaker guesses the code or the maximum number of turns is over.

Hardware specification and wiring:

We used 2 LEDs Red and Green connected to GPIO pins 5 and 13. A button is connected to GPIO pin 19. Next LCD control pins are connected to GPIO 25 and 24. The LCD data pins are connected to the GPIO 23, 10, 27, and 22 pins. A potentiometer is connected to LCD3 to control the contrast. 3 power wires are connected from 5v to +ve of the breadboard. 2 Gnd wires are connected to -ve of the breadboard.



Wiring of the pi

Discussion of the code:

- digitalWrite: This function is used to set a specific GPIO pin to LOW or HIGH state by directly accessing the registers.
- pinMode: This function is used to set the mode of the GPIO pin to INPUT or OUTPUT by accessing the pin's position
- WriteLED: This function is used to turn the LED on or off by clearing the pin based on the value provided (HIGH or LOW)
- readButton: This function is used to read the state of the button, if the button is pressed it returns LOW, or else it returns HIGH.
- waitForButton: This function waits till the button connected is pressed. Then it reads the button state using readButton
- initSeq: This function is used to generate a random sequence in the size of SEQL for the secret code.
- showSeq: This function prints the sequence generated by the initSeq.
- countMatches: This function is used to count the exact and approximate matches between the 2 sequences.
- showMatches: This function is used to print the exact and approximate matches from the countMatches function.
- readSeq: This function is used to read the int digits from the value and stores it in an array.
- readNum: This function is used to read the number inputted by the user as a string and convert it to an int.
- blinkN: This function is used to blink the LED a given number of times with a 0.5sec delay

Performance-relevant design decisions:

- The code we have provided avoids unnecessary abstractions and uses low-level operations which minimizes resource consumption, such as CPU cycles and memory usage.
- The digitalWrite() and pinMode() functions efficiently manipulate the GPIO pins by directly accessing memory-mapped GPIO registers minimizing overhead and reducing resource consumption.
- The initSeq() function uses dynamic memory allocation (malloc()) to allocate memory for the sequence ensuring efficient memory usage.

- lcdClear() function is used in the main function to appropriately clear the LCD display ensuring the display is ready for the new round's information.
- The code includes error-handling mechanisms such as checking for NULL after memory allocation and error checking for the functions used in initTimer()

Functions accessing the hardware:

These functions are implemented using inline assembler
- digitalWrite()
- pinMode()
- writeLED()
- readButton()

These functions are implemented using C
- waitForButton()
- lcdPuts()
- lcdPutsChar()
- lcdCursor()
- lcdCursorBlink()
- lcdDisplay()
- lcdPosition()
- lcdPutCommand()
- lcdPut4Command()
- lcdClear()

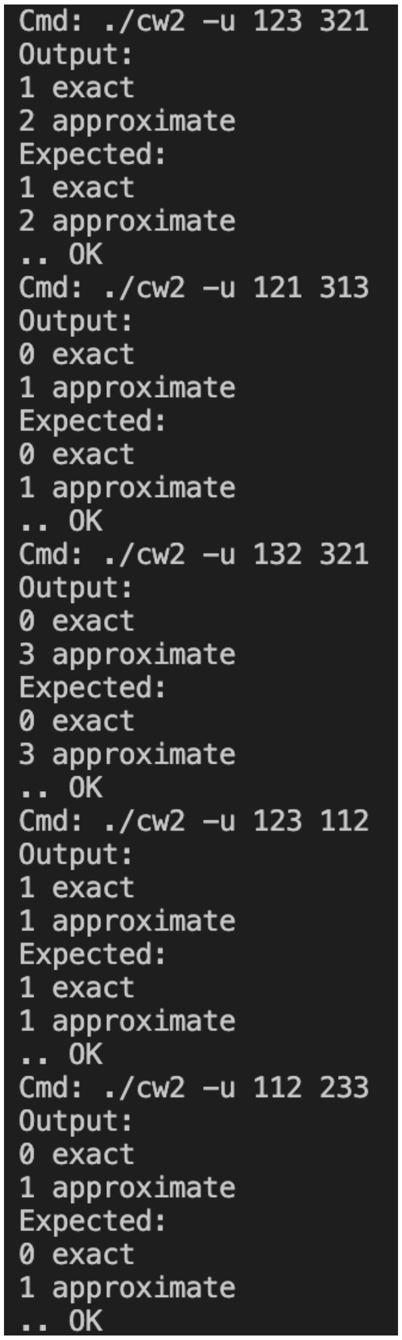
Example set of Output:

The game starts with a welcome message on the LCD and asks the user to input the number. The terminal prints the number of times the button is pressed and the approximate and exact matches. The whole output of the terminal is in the zip file called output.log

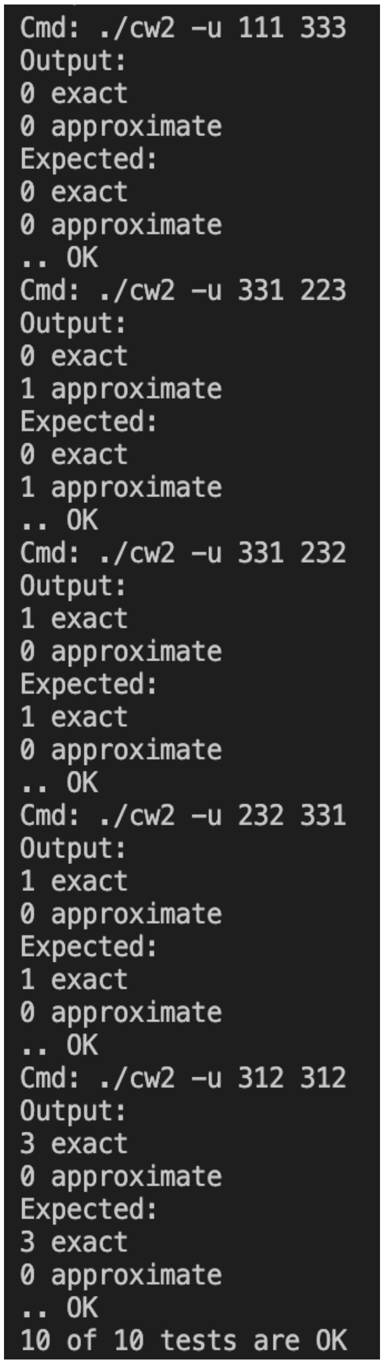
Interface discussion of matching function:

Inputs: This function takes two inputs: a pointer to the secret code and the input sequence. The sequences are stored as arrays.

Outputs: This function returns 2 numbers: exact matches and approximate matches



```
Cmd: ./cw2 -u 123 321
Output:
1 exact
2 approximate
Expected:
1 exact
2 approximate
.. OK
Cmd: ./cw2 -u 121 313
Output:
0 exact
1 approximate
Expected:
0 exact
1 approximate
.. OK
Cmd: ./cw2 -u 132 321
Output:
0 exact
3 approximate
Expected:
0 exact
3 approximate
.. OK
Cmd: ./cw2 -u 123 112
Output:
1 exact
1 approximate
Expected:
1 exact
1 approximate
.. OK
Cmd: ./cw2 -u 112 233
Output:
0 exact
1 approximate
Expected:
0 exact
1 approximate
.. OK
```



```
Cmd: ./cw2 -u 111 333
Output:
0 exact
0 approximate
Expected:
0 exact
0 approximate
.. OK
Cmd: ./cw2 -u 331 223
Output:
0 exact
1 approximate
Expected:
0 exact
1 approximate
.. OK
Cmd: ./cw2 -u 331 232
Output:
1 exact
0 approximate
Expected:
1 exact
0 approximate
.. OK
Cmd: ./cw2 -u 232 331
Output:
1 exact
0 approximate
Expected:
1 exact
0 approximate
.. OK
Cmd: ./cw2 -u 312 312
Output:
3 exact
0 approximate
Expected:
3 exact
0 approximate
.. OK
10 of 10 tests are OK
```

Summary:

Achievements and Features:

The code successfully implements the mastermind game using C, inline assembler, and assembly, effectively interfacing with the hardware components such as LED, LCD, and buttons to do so. LCD displays the necessary information and the button provides a means for the user to facilitate gameplay. Our code also works on sequences of arbitrary lengths allowing the user to play the mastermind game in varying difficulty levels.

Areas of Improvement and Learning Points:

Though the code provides necessary error handling mechanisms, additional error checking and validation could be provided to enhance robustness. Further optimization could also assist in minimizing resource consumption.

The coursework gave us a much deeper understanding of programming in C and assembly and also provided us with insight into hardware manipulation using software.