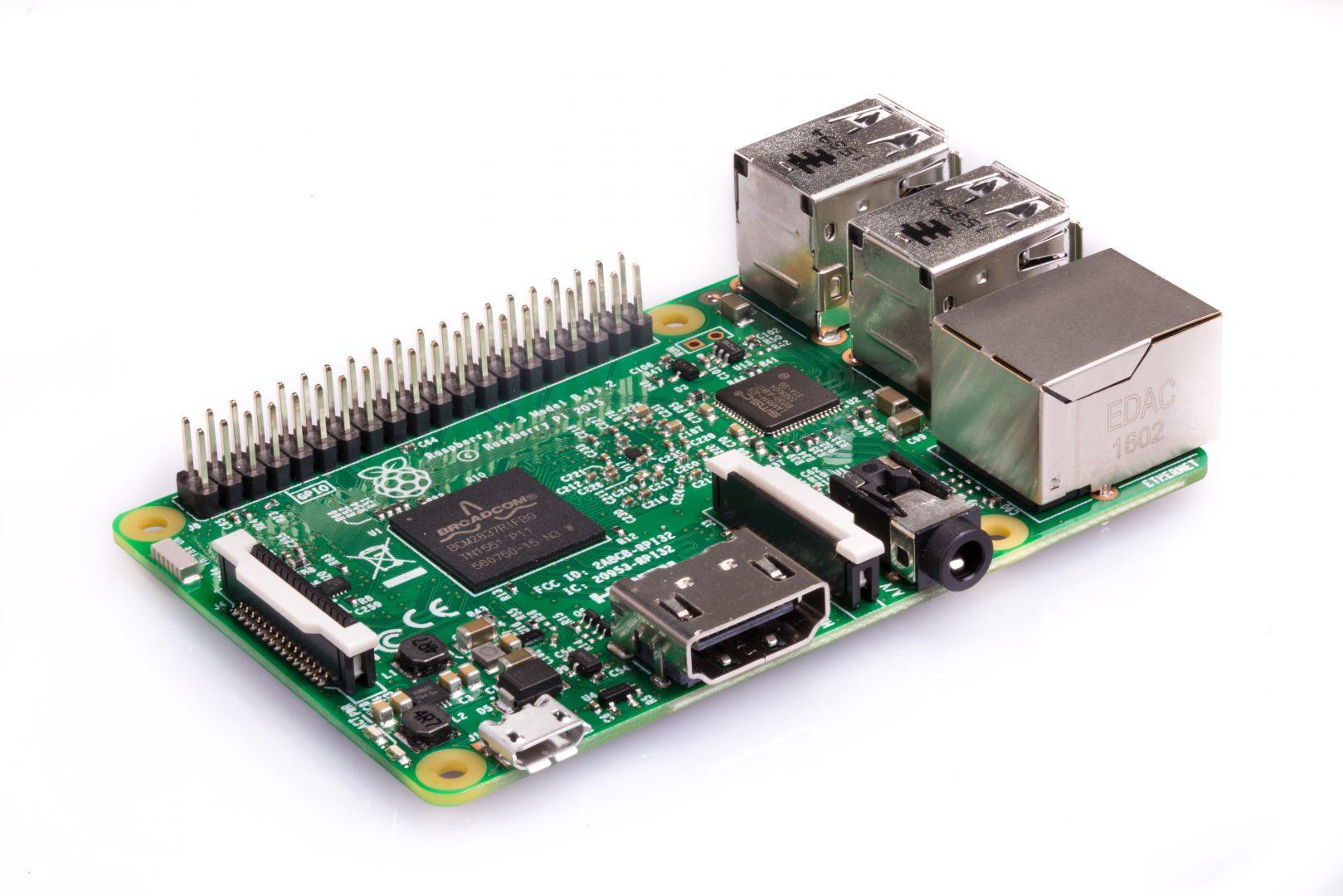
HARDWARE AND SOFTWARE INTERFACE – COURSEWORK 2



FAAIZA SIRAJ

MUHAMMAD TAHA AHSAN

**Code Structure and functionality**

This coursework was based on the mastermind game. We had to implement this concept on the raspberry PI using C language and ARM Assembly, as well as hardware components such as LEDs, a button, LCD, a breadboard, a potentiometer and various wires. We mainly did this in C, however we did use inline assembly for a few functions. We had to focus on reading the user’s input from a button, blinking the LED as well as displaying information on the LCD. Not only did we have to read and output information, we had to compare and repeat processes based on the input received, which made this coursework more challenging and fun to complete at the same time.

**SETTING HARDWARE**

The code starts off with the variables and PINs being defined. The next segment is setting the PIN mode that are required for the specific devices (The LEDs, the button, and the LCD), which we first did in c and then changed to a function in inline assembly.

**GAME LOGIC**

The game starts off by giving the user an option to choose which mode they want to play, 1 plays with a randomly generated sequence, 2 allows a user to enter the secret, essentially making it a two player game. After the user selects the mode, they enter the length of the secret that they want, and then how many colors they want in the game, all three of these values are received using scanf. In the normal mode, the secret will be randomly generated, and only displayed if the user started the game in debug mode, the secret is generated using srand and seeding it with the time, followed by modding the value by the number of colors the user entered, this is done as many times as the user chose for the length of the secret. There is a slight delay to give the user time to prepare, and then the game starts.

**Program**

The code starts off with the variables and PINs being defined. The next segment is setting the PIN and the mode of that PIN, which was written in assembly and implemented using inline assembler. We used registers to store the PIN value and then set the mode to 001 since it’s the LED and is an output. The same thing was done for the button PIN, however since the button is an input, the mode was set to 000. Delays were used in our code to improve the efficiency as well as accuracy of our program, since it allows the user’s input to be read properly with the given amount of time, and it also gives users enough time to think; thus, making our program efficient for both the users and the program itself. After that, we included all the code related to the LCD; this consists of the way in which the message will be displayed on the LCD as well as the actual acceptance and setting of the message. We created our own functions to blink the 2 LEDs; red and green, it was first done in C however we changed it to assembly during the completion since it made it more challenging for us but produces an interesting output at the same time. We used features such as delay and shifting for this function. The next function is the input from the button, which is one of the main functions of our program, to complete this part we used delay, shifting, the lcd struct and many more features. The process in which the results of the guess is achieved starts off by looping through the secret and the guess and compares the value of the same index, if they match then the exact counter gets incremented, else, it moves on to the next index. If there is a match then the value in the secret array is changed to a different value so that it doesn’t affect the following comparisons giving incorrect results. The second part of the comparison starts with the first index of the secret and loops through the guess array checking if there are any matches, once it is done then it moves onto the next index of the secret, this is done until all the values have been compared thus producing results for the user.

We map the memory using C and set the mode of the pins in the main using assembly, the register value is set based on the PIN and the shift is the left integer of the PIN multiplied by 3, this is because it uses 3 bits per PIN. After this we provided the user with an option, they can either play the game based on a randomly generated secret, or they could enter the secret themselves and test anyone else. The secret will be read from the button and stored in an array, the main parts of the game runs in the exact same way no matter which option is selected by the user.

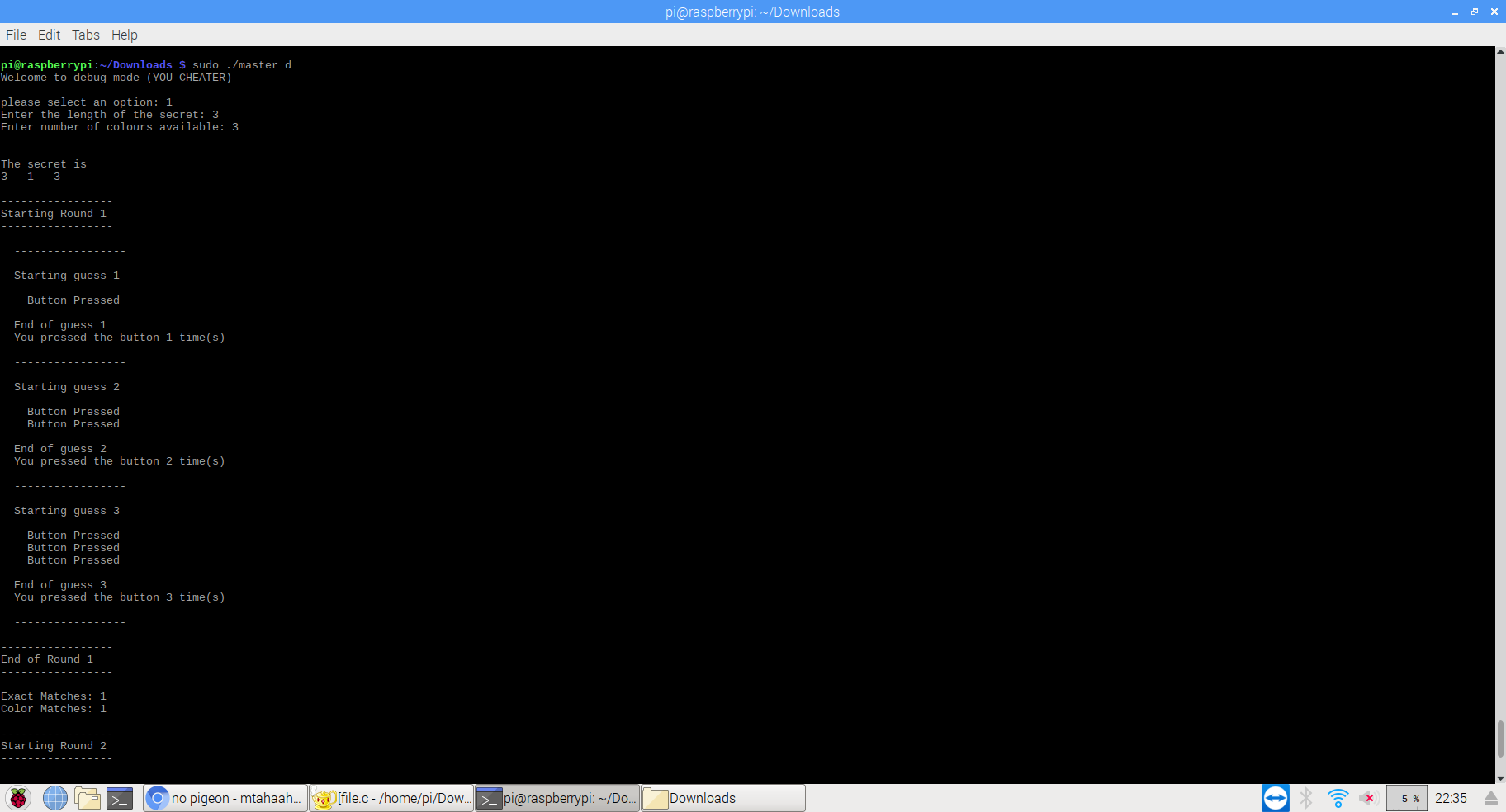
We created a function that sets the pin’s mode called pinMode, in this function, you provide the pin and the mode you want to set it to (input or output). We then get the fSel and the shift for the respective pin so that we can use it in the assembly code. Our assembly for this function starts off by loading the GPIO into R1, and then adds the fSel to the GPIO to access the right register. The next line then reads the current value of the register. We follow this by moving 0b111(#7) into R2, and then we shift this left by the shift value, and then we AND R1 with the negative of R2 so that it clears the selected pins. Next is moving the value of mode into R2, and then shifts it left by the value of shift from the pin. Finally, we OR R1 with R2, setting the pins to the desired mode. In order to use the PINs later on in the program, we simply call the function and pass in the required arguments.

The digitalWrite function is used to clear and set the lcd based on the PIN and the value provided (1 being set0, 0 being clr0). So we first check what value is provided, and then get the corresponding register. Following this, we load the GPIO and add the offset, this is stored in R0. Next is moving #1 into R2, We then move the PIN value into R1, and then AND it with 31, this is the same as 1 <<(pin & 31) in C. Then we shift R2 (#1) left by R1 (PIN) and then store R2 into R0(GPIO).

Overall, we have created an efficient and accurate program that represents the Mastermind game using the hardware provided, and we used both C and assembly to code the game.

**Functions directly accessing the hardware**

* digitalWrite(C language and Assembly)
* pinMode(Assembly)
* AssemblyInput (Assembly)
* lcdPutCommand (C language)
* lcdPut4Command (C language)
* lcdClear (C language)
* lcdPosition (C language)
* lcdDisplay (C language)
* lcdPutChar (C language)
* lcdPuts (C language)
* blinkRed (C language) and blinkRedAssembly (Assembly)
* blinkYellow (C language) and blinkYellowAssembly (Assembly)
* colorInput (C language)
* game (C language and Assembly)

**Execution of the program in debug mode**

Since the guess was wrong, the 2nd round starts; giving the user another attempt at guessing the secret.

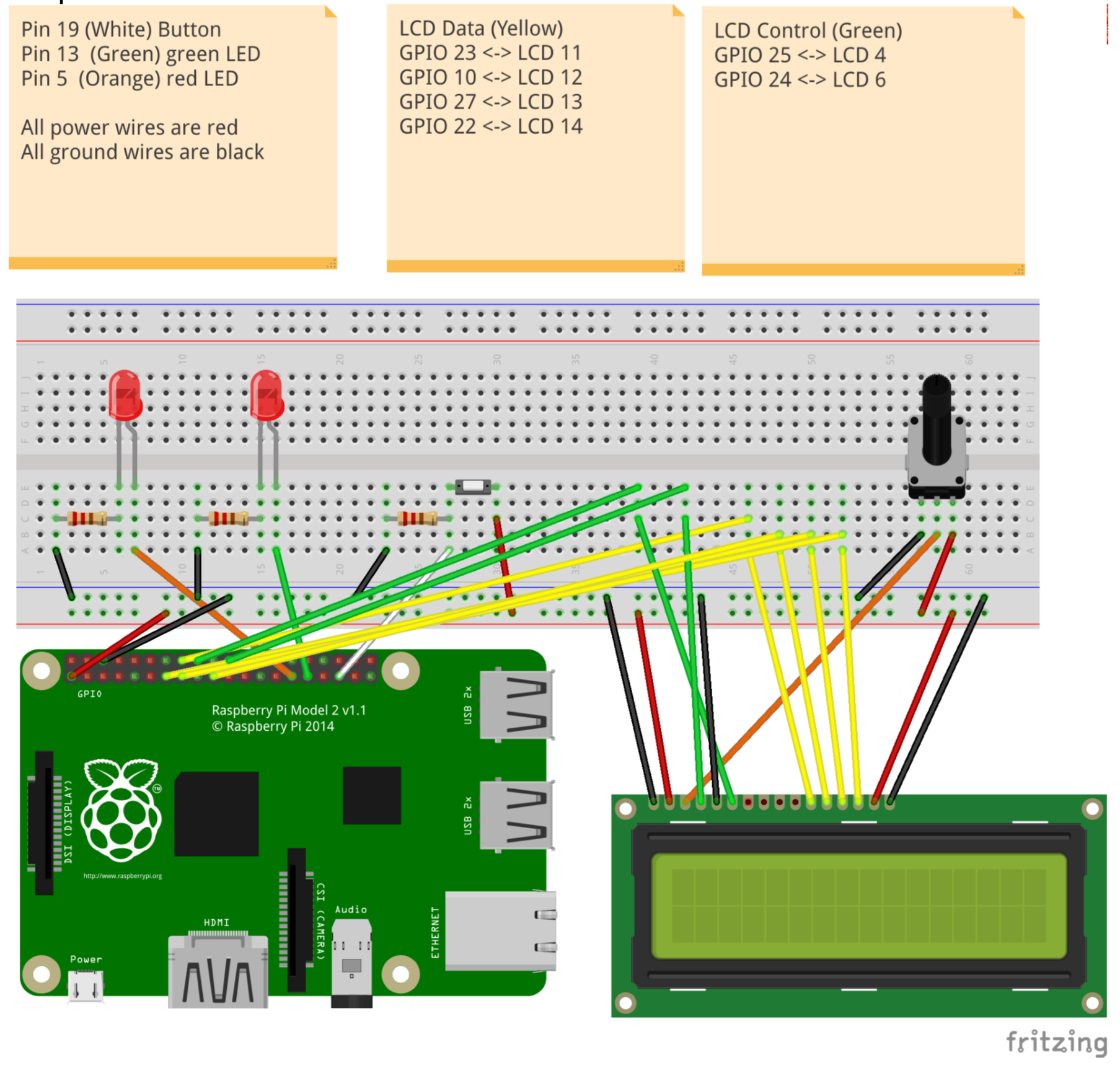
**Summary of the coursework**

We were able to complete the tasks that were set for this coursework and it was done using both C language and inline assembly. The tasks that we were able to complete included randomly generating the secret for the user, allowing the user to input their guesses using the button, comparing the guess to the secret generated in order to collect information on their exact and colour matches, making both LEDs blink when required and the specified number of times; for example blink the red LED once to accept the input and then blink the green LED the number of times the button was pressed to echo the input value, we included a delay to separate the input values in order to make it a neat and successful program, the program will repeat a maximum of 3 times giving the user 3 attempts, however it will end as soon as the guess matches the secret, finally once the results have been calculated it will be displayed on the LCD to inform the users; if the user’s guess matches the secret then “SUCCESS” will be displayed on the LCD after the results.

We included a few extra features into our program, the first feature is that if the user presses the button for a long time before letting it go, it will take that input as 1 and not based on how long he held the button for, this avoids any incorrect input, another feature is the efficiency of the inputs, the user might press the button 2 times really quickly and the program will still be able to calculate the input as 2 making it quite efficient. The last extra feature that we implemented is an extension to the coursework requirement. We provided the user with an option at the start, they can either allow the computer to randomly generate a secret, or they can input the secret themselves and test another user. The secret will be inputted using the button, and then the program will continue as normal.

We have learnt a lot from this coursework. Whilst creating this game, we learnt a variety of things; first of which was how the breadboards work, where the wires are supposed to go, where to get power from, which wires will read and set values, and where to ground the circuit. We also learnt about how the GPIO pins of a raspberry pi work, and the many registers involved in controlling and reading them, followed by how to actually get them to do what we want it to do in C and ARM Assembly, this involved setting and modifying the modes, reading from a button, turning on and off an LED, and displaying data on the LCD. From this, the greatest thing we learnt was how to control the GPIO pins through C. We were impressed by C’s robust capabilities along with its direct access to the GPIO and memory. It was a fun and successful coursework overall, we enjoyed solving it as well as learning many new things throughout the coursework.

**Hardware Specification**



**References**

LED

<http://www.macs.hw.ac.uk/~hwloidl/Courses/F28HS/srcs/tut_led.c>

Button

<http://www.macs.hw.ac.uk/~hwloidl/Courses/F28HS/srcs/tut_button.c>

LCD

<http://www.macs.hw.ac.uk/~hwloidl/Courses/F28HS/srcs/lcd-hello.c>

Assebmly

<http://www.macs.hw.ac.uk/~hwloidl/Courses/F28HS/srcs/tinkerHaWo33.c>

<http://www.macs.hw.ac.uk/~hwloidl/Courses/F28HS/handouts_SysPrg_Tut5.pdf>

Coursework Specification

<https://learn-eu-central-1-prod-fleet01-xythos.s3-eu-central-1.amazonaws.com/5b44cfad90f2e/2112728?response-content-disposition=inline%3B%20filename%2A%3DUTF-8%27%27Coursework_F28HS%25285%2529.pdf&response-content-type=application%2Fpdf&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Date=20190324T174333Z&X-Amz-SignedHeaders=host&X-Amz-Expires=21600&X-Amz-Credential=AKIAIZ3QX2YUHH4EOO3A%2F20190324%2Feu-central-1%2Fs3%2Faws4_request&X-Amz-Signature=b597df3cfe04deba33c592e035e64b9ae9ed89ee37def89a56531684fb428c71>