Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

MICROPROCESSOR SYSTEMS
ASSIGNMENT-2
GROUP REPORT

Group - 18
Report made by Ritwik Kumar and Shruti Kathuria as mentioned in the tasks.

GROUP MEMBERS:

SERIAL NUMBER	NAME	TCD ID
1.	CHI CHUN KO	19336516
2.	SWETANG KRISHNA	21355087
3.	RITWIK KUMAR	21355092
4.	BRAHMJOT KAUR	21355071
5.	SHRUTI KATHURIA	21355061

LINKS:

TASKS LINK	AssignmentTasks (1).pdf
GITLAB ASSIGNMENT 2 LINK	Link

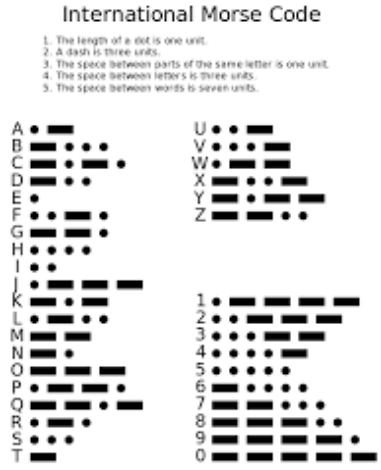
GROUP FUNCTIONALITY:

GROUP LEADER	CHI CHUN KO
GITLAB WORKFLOW OWNER	CHI CHUN KO
PROJECT DEMONSTRATION OWNER	BRAHMJOT KAUR
PROJECT WORKFLOW OWNER	SHRUTI KATHURIA
PROJECT DOCUMENTATION OWNER	RITWIK KUMAR
PROJECT CODE OWNER	SWETANG KRISHNA

INTRODUCTION:

For the Assignment 2 we as a team have worked on the Morse game as defined in the pdf assignment.

Morse code is a system of electronic communication that uses dashes, dots and spaces to represent letters, numbers and punctuation. These dots, dashes and spaces are arranged to spell out a message, slashes are used as spaces to separate words.



The task is to use a mixture of C code and ARM assembly to build a simple game that will teach a player MORSE CODE. The player should interact with the game by pressing the GP21 button on the MAKER-PI-PICO board for a short duration to input a Morse “dot” and a longer duration to input a Morse “dash”. If no new input is entered for at least 1 second after one-or-more “dots” or “dashes' ' have been input, then it should be considered a “space” character. If no new input is entered for at least another 1 second after that, then the sequence should be considered complete and passed to the game portion of the code for pattern matching. The game should have a minimum of two levels. The first level for matching individual characters (with the expected Morse pattern provided) and the second for matching individual characters (without the expected Morse pattern provided).

WORKFLOW:

Project Demonstration Owner: (BRAHMJOT KAUR)

- As a project Demonstration owner, I was responsible for running the implemented code on a raspberry pi board and further creating a demonstration video of 90 seconds of the code running on raspberry pi.

Project Workflow Owner:(SHRUTI KATHURIA)

- As a Project workflow owner, I was responsible for the overall working of the project. Chi divided everybody with one task for the code so that everybody could merge the code later. We later had meetings to discuss the codes too.
- In the meeting everybody voluntarily decided on what parts to work on till the very end. As a Project workflow owner, I kept in mind for everybody to communicate well on the WhatsApp group that was made and everybody used to update on whatever he/she was completing at that time.
- Also attended the labs to discuss the project with the team and also explained the overall progress to the professor in the lab too.
- Met with the team leader in the lab and also discussed the overall progress of the project as a whole. Created targets for the functions to be added in the code and everybody completed it within the given timespan.
- The codes were tested by the leader (Chi) and merged with the main branch and all the issues were updated on the Gitlab issues.
- Tracked the issues on GitLab and checked the functionalities.

Project Code Owner:(SWETANG KRISHNA)

- As the project code owner, I was responsible for ensuring that the overall application functions as expected.
- Everyone in the team contributed to implementing the code required for the application, however as the other team members have additional roles to perform I was responsible to undertake a slightly large share of coding tasks.
- The complete code was divided into parts and each was given to the respected team member. As a project code owner I had to take responsibility to divide the coding part evenly amongst the team members considering their individual tasks, so that no one is overburdened.
- The team members were very co-operative and this contributed to the smooth working of the project.

Project Documentation owner:(RITWIK KUMAR)

- As the project documentation owner, I am responsible for compiling the required documentation and final project report from my team.
- Included Headers and title for the documentation of the project.
- I am responsible for setting up the initial template and structure of the report in a location that other team members can access.
- I am responsible for ensuring the final report reads as a single unified document when complete.
- I made sure the report was well made and everybody's contribution was included well in the report.

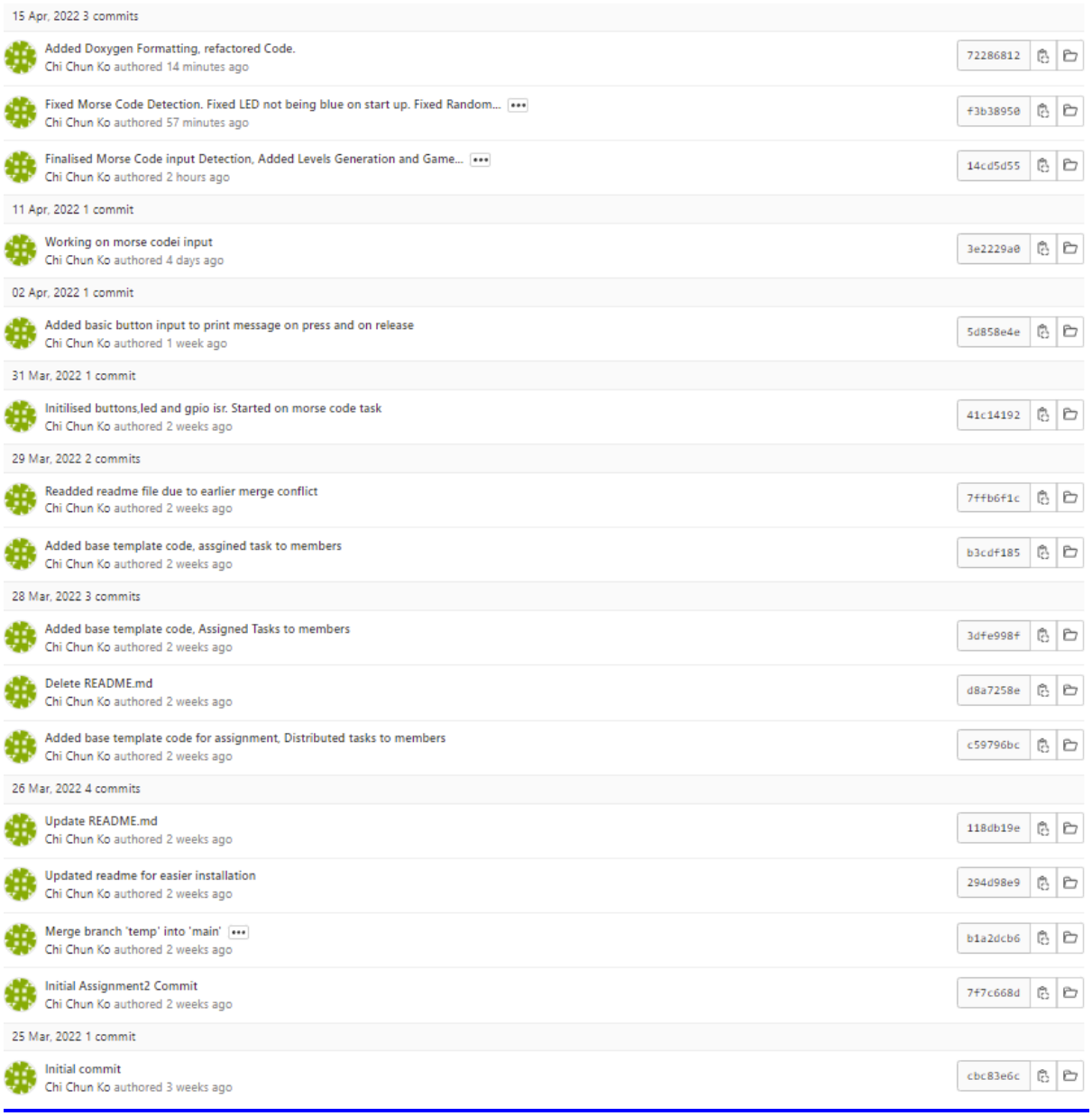
Group Leader and Gitlab Workflow owner :(CHI CHUN KO)

- Assigned and distributed tasks to each member of the team based on their strengths and weaknesses
- Arranged meetings (in person and online)
- Set up an alternative discord group channel for organised communication and file sharing.
- Established a gitLab repository as well as handled merge requests.
- Created a guide on how to create and push to a git branch
- Reviewed Code when handling merge requests

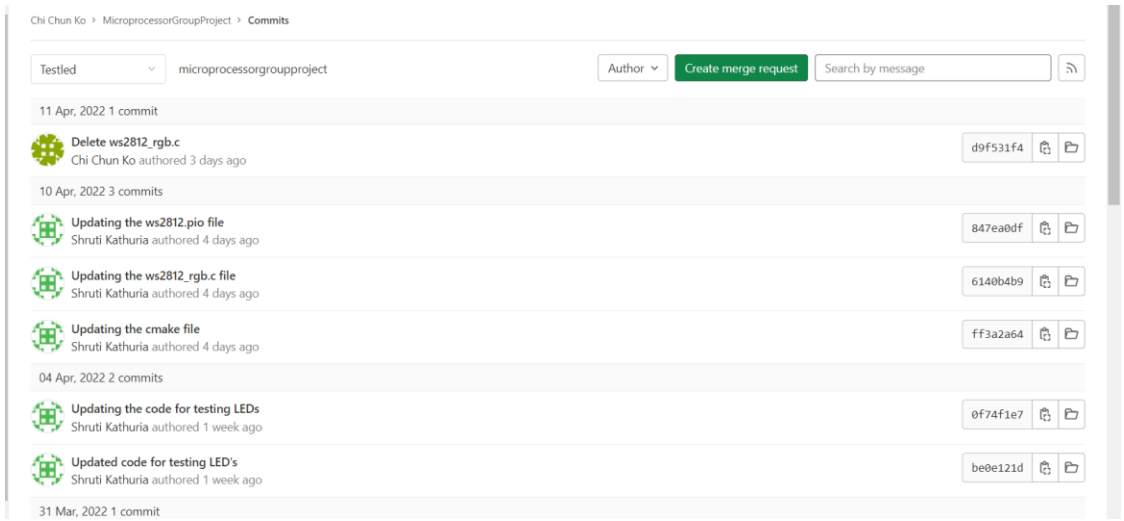
GITLAB:

SCREENSHOT OF THE GITLAB REPOSITORY:

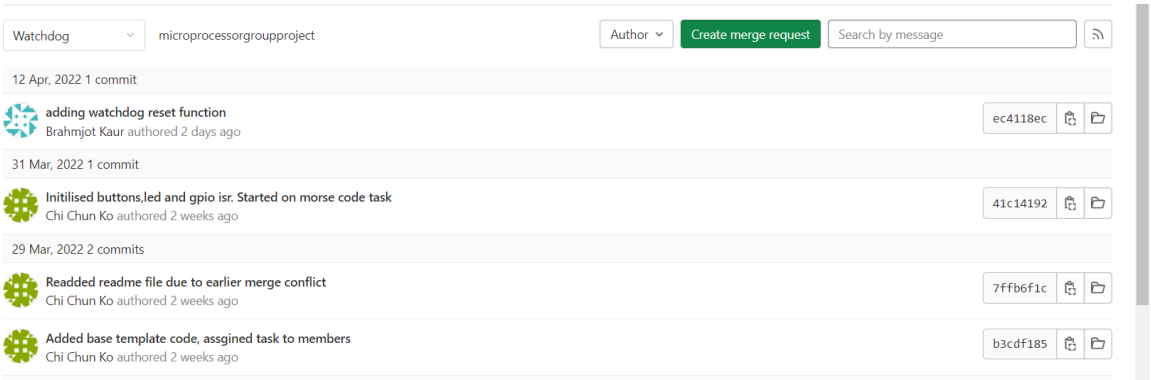
CHI CHUN KO



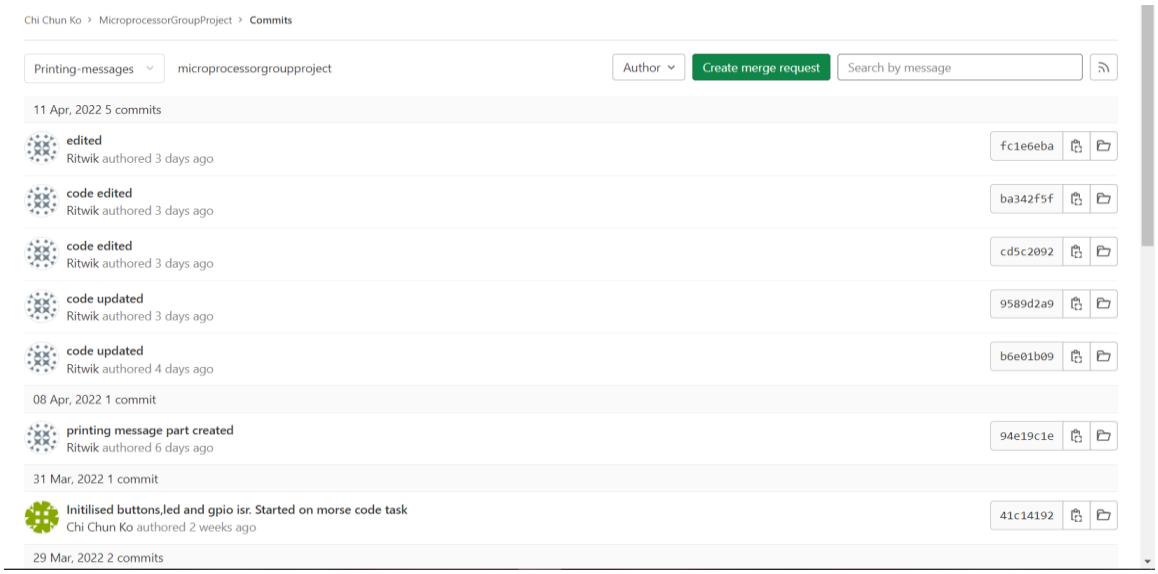
SHRUTI KATHURIA



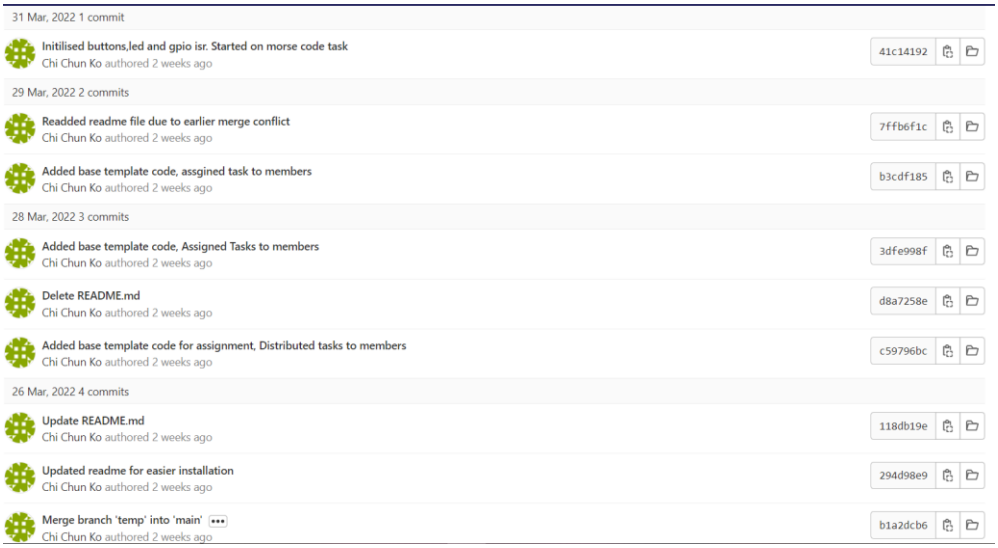
BRAHMJOT KAUR



RITWIK KUMAR



SWETANG KRISHNA



Swetang's terminal on the vs code was not working so swetang's work was pushed by Chi Chun ko.

CODE:

BRAHMJOT KAUR:

Contribution to the code

- I was assigned the task to figure out the use of watchdog timer
- Start with a basic program to print "Hello World" Then implement watch dog timer so that it resets the board
- The "Hello World" statement should be printed as output if done correctly
- The RP2040 Watchdog Timer should be used to trigger a reset of the Raspberry Pi Pico and return it to the application welcome message banner if the game is in a running state and no input is detected for the maximum timeout of the Watchdog Timer (approximately 9 seconds according to the datasheet).

Code Explanation:

- A watchdog timer is a hardware which generates a system reset automatically if the main program does not periodically service it. It is often used to reset an embedded device hanging because if a software or hardware fault.
- Watchdog library from hardware is called in the .c file.
- Watchdog_update function is further defined which updates our watchdog code periodically.
- We call watchdog _reboot in the main function which prints rebooted in case of true.

SHRUTI KATHURIA:

Contribution to the code:

- The tasks were divided by Chi at the very start when assignment 2 was introduced on blackboard.
- I was given a task to work on the RGB lights of the micro pi pico board.
- I used the ws2812_rgb file template as a starting point to write the code for the RGB lights.
- Worked on the assign02.c file for the RGB part.

Code Explanation:

- At the very start of the Morse game when the game is not in progress the lights of the RGB lights are blue.
- Used a switch case statement when the game starts the RGB lights are light green with 3 lives.
- Every time when a player inputs a sequence it is not equal to the expected sequence so the number of lives decreases to 2 and the colour updates to light yellow.
- Every time when a player inputs a sequence and it is equal to that of the expected sequence so the number of lives increases (maximum can only be up to 3) and colour changes back to green.
- If the number of lives again decreases from 2 to 1 then the colour of the RGB lights changes to light orange.
- If again the number of lives decreases to 1 to 0 then the colour of the RGB lights changes to light red.
- In the code for accessing the light green light we have put the pixels accordingly. For example, put_pixel (red, green, blue). We write it as put_pixel (0x00,0x7F,0x00. (Rest all are the same representing the pixels of colours in this form.)

SWETANG KRISHNA:

Contribution to the Code:

- Implement a random number generator to select a random alphanumeric character to be used for levels 1 and level 2 of the morse code game.
- The tasks were evenly divided among us and I was given the task to implement a random number generator as well as a random alphabet generator.
- There are two functions for each level that generates a random number and a random alphabet, and as the level increases the random alphabets generated becomes complicated to solve.
- Some of the printing functions were also implemented.

Code Explanation:

- There are a total 4 functions that are used to generate a random number and a random alphabet.
- The first two are used to generate a random number and a random alphabet for level 1 it will only choose a random number from the range 0-100.
- The level 1 alphabet generator will only generate the capital letters.
- For the next level, that is level 2 of the morse code game, the motive is to make the level tougher for the player so the number generated will be in the range 100-10000.
- The function is dependent on a header file time.h so I also had to add the header file to the code.
- Moreover we can use both of the functions to create a random and complicated morse code (ALPHANUMERIC) as the level rises.
- The alphabet generator uses the ASCII value of the alphabets to generate a random letter.(In short it is the same as a number generator but then it converts the number using the ASCII value to the alphabets)

RITWIK KUMAR:

Contribution to the Code:

- The task which I got during task distribution was Printing messages.
- Printing of the group number and the levels for the project.
- Printing the random character for level 1 and level 2.
- Printing message for completion of the level.
- All these were printed in the .c file of the assignment.

Code Explanation:

- For the morse game I printed the Group number that is provided on the blackboard and the level selection.
- The random character taken from the rand function for level 1 and level 2 is also printed.
- Once the game is completed when the person enters 5 right sequences during the current level then the message for completion of the level is printed on the console.
- For the level selection it depended on the input string if it was correct then the level was changed else the game was terminated and a printf message was printed on the console saying that program terminated with a wrong input.
- For different levels the difficulty level is also printed in the printf statements.

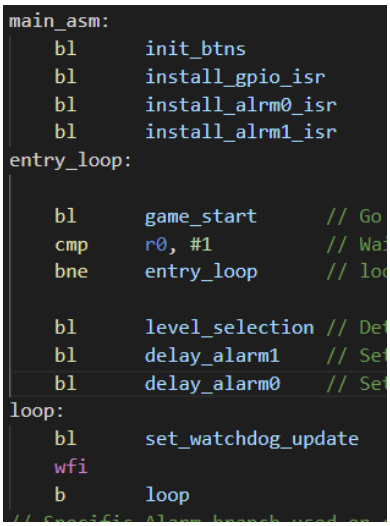
Chi Chun Ko:

Code Contribution

- Handled all assembly language code and majority of the code in C.
- Installed all inputs including gpio button detection, timings and alarms which handled space and user buffer submissions.
- Handled morse code game logic - This includes the input timings for dots and dashes, string detection, life, levels and stage progression logic.
- Designed changing levels and character generation
- Tested and reviewed all code pushed onto the GitLab repository including other team members.
- Modified team members code to suit the assignment.

Implementation

The program is mainly run on ASM Language with additional functionality added with C. There are three main parts to the ASM program. The first is main_asm where all required ISRS and initializations are performed.



In **entry_loop**, the user is required to enter a 5-character morse code string in order to select a level. Until a level is selected the loop will repeat. Alarms are also manually established with longer delays before inputs. This is to prevent the user from having a small window of time to enter an input upon level entry.

In **loop**, the C watchdog function is called every time an input is triggered. This resets the watchdog timer to 9 seconds.

Two alarms are used to handle space and pattern matching. The first alarm, alarm1 triggers once 1.5 seconds have passed since last input. When this alarm is triggered, the alarm ISR occurs during which, the second alarm0 is set with a 2 second delay. Tests were performed with 1 second on alarm1 and 1 second on alarm0, however, the window for input was simply too small and errors could be easily made.

CONCLUSION:

Doing this assignment, we as a group learned a lot of things:

- Collaborated well as a group on GitLab as well as on the WhatsApp group.
- Learned about how to code for Morse and how to add up different functionalities to the same project.
- Learned to use GitLab for the projects and also created merge requests, issues and pushing of the code into our branches.
- Learned how to make a proper game for the hardware kit.
- Learned assembly coding to a great extent.