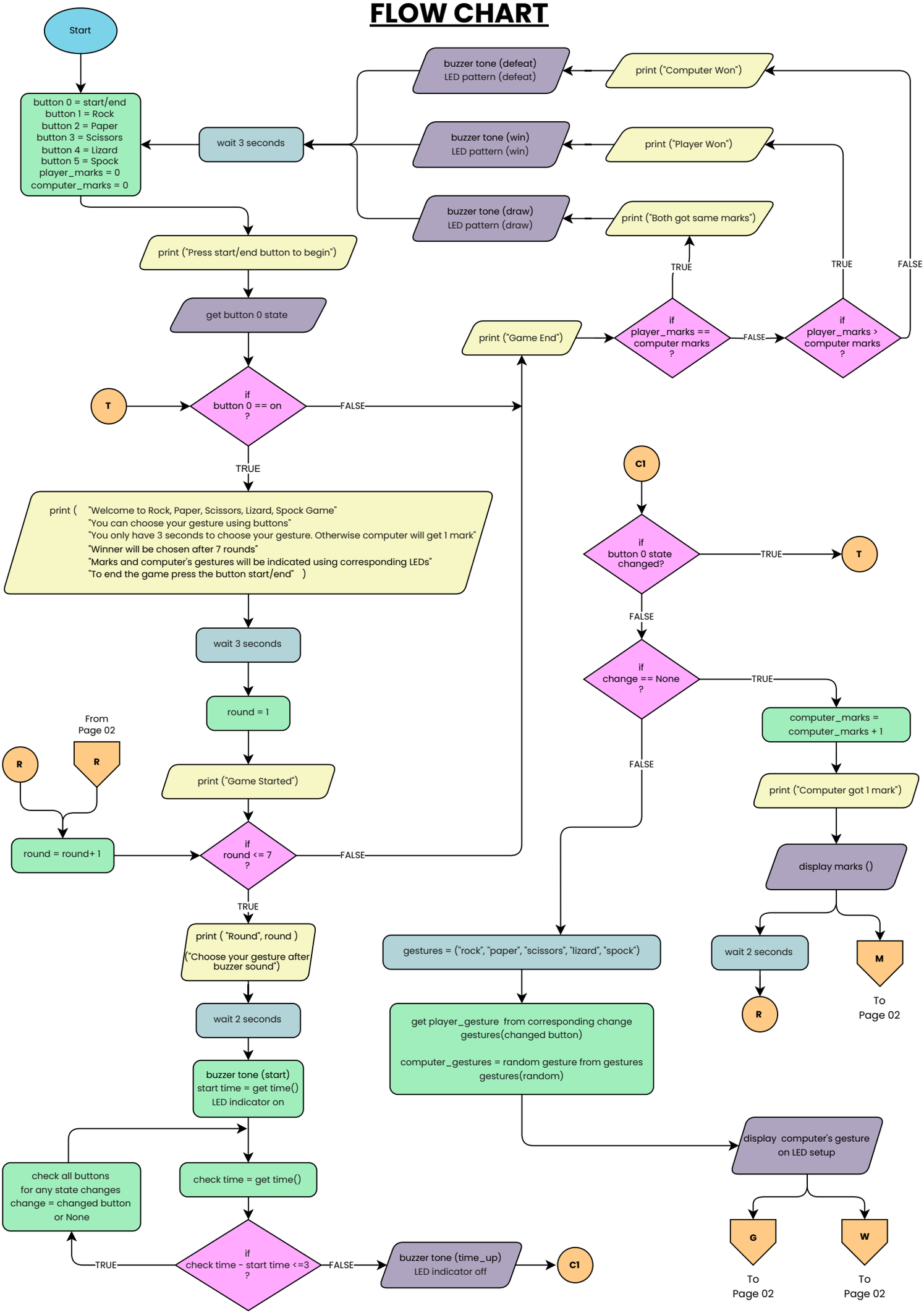
GROUP A4

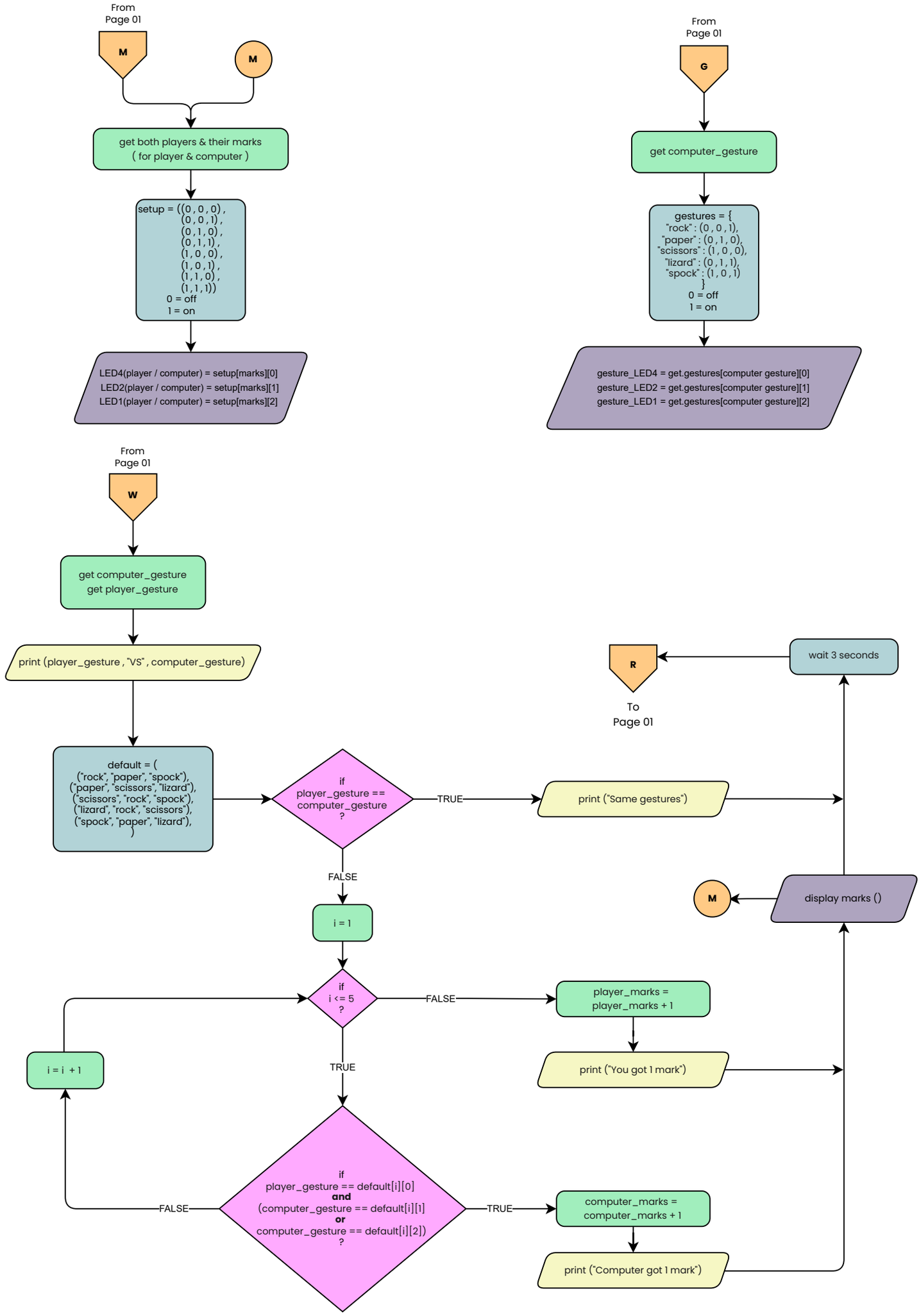
FLOWCHART & DESIGN CHOICE DOCUMENT

Group Members

- **E/21/016 Adams S.L.**
- **E/21/017 Adeesha W.L.T.**
- **E/21/018 Adikaram A.M.P.S.**
- **E/21/019 Adikari A.M.H.S.**
- **E/21/020 Ahamed M.M.H.**

FLOW CHART


Start
button 0 = start/end
button 1 = Rock
button 2 = Paper
button 3 = Scissors
button 4 = Lizard
button 5 = Spock
player_marks = 0
computer_marks = 0
print ("Press start/end button to begin")
get button 0 state
T
if button 0 == on ?
FALSE
TRUE
print ("Welcome to Rock, Paper, Scissors, Lizard, Spock Game"
"You can choose your gesture using buttons"
"You only have 3 seconds to choose your gesture. Otherwise computer will get 1 mark"
"Winner will be chosen after 7 rounds"
"Marks and computer's gestures will be indicated using corresponding LEDs"
"To end the game press the button start/end")
wait 3 seconds
round = 1
print ("Game Started")
From Page 02
R
R
round = round+ 1
if round <= 7 ?
FALSE
TRUE
print ("Round", round)
("Choose your gesture after buzzer sound")
wait 2 seconds
buzzer tone (start)
start time = get time()
LED indicator on
check time = get time()
if check time - start time <=3 ?
TRUE
FALSE
check all buttons
for any state changes
change = changed button
or None
buzzer tone (time_up)
LED indicator off
C1
print ("Game End")
if player_marks == computer marks ?
TRUE
FALSE
if player_marks > computer marks ?
TRUE
FALSE
buzzer tone (defeat)
LED pattern (defeat)
print ("Computer Won")
buzzer tone (win)
LED pattern (win)
print ("Player Won")
buzzer tone (draw)
LED pattern (draw)
print ("Both got same marks")
wait 3 seconds
C1
if button 0 state changed?
TRUE
T
FALSE
if change == None ?
TRUE
FALSE
computer_marks = computer_marks + 1
print ("Computer got 1 mark")
display marks ()
wait 2 seconds
R
M
To Page 02
gestures = ("rock", "paper", "scissors", "lizard", "spock")
get player_gesture from corresponding change gestures(changed button)
computer_gestures = random gesture from gestures gestures(random)
display computer's gesture on LED setup
G
W
To Page 02
To Page 02


From Page 01
M
M
get both players & their marks (for player & computer)
setup = ((0 , 0 , 0) , (0 , 0 , 1) , (0 , 1 , 0) , (0 , 1 , 1) , (1 , 0 , 0) , (1 , 0 , 1) , (1 , 1 , 0) , (1 , 1 , 1))
0 = off
1 = on
LED4(player / computer) = setup[marks][0]
LED2(player / computer) = setup[marks][1]
LED1(player / computer) = setup[marks][2]
From Page 01
G
get computer_gesture
gestures = { "rock" : (0 , 0 , 1), "paper" : (0 , 1 , 0), "scissors" : (1 , 0 , 0), "lizard" : (0 , 1 , 1), "spock" : (1 , 0 , 1) }
0 = off
1 = on
gesture_LED4 = get.gestures[computer gesture][0]
gesture_LED2 = get.gestures[computer gesture][1]
gesture_LED1 = get.gestures[computer gesture][2]
From Page 01
W
get computer_gesture
get player_gesture
print (player_gesture , "VS" , computer_gesture)
default = (("rock", "paper", "spock"), ("paper", "scissors", "lizard"), ("scissors", "rock", "spock"), ("lizard", "rock", "scissors"), ("spock", "paper", "lizard"),)
if player_gesture == computer_gesture ?
TRUE
print ("Same gestures")
R
To Page 01
wait 3 seconds
M
display marks ()
FALSE
i = 1
if i <= 5 ?
FALSE
player_marks = player_marks + 1
print ("You got 1 mark")
TRUE
i = i + 1
if player_gesture == default[i][0] and (computer_gesture == default[i][1] or computer_gesture == default[i][2]) ?
FALSE
TRUE
computer_marks = computer_marks + 1
print ("Computer got 1 mark")

Design Choice Document for Rock-Paper-Scissors-Lizard-Spock Game Group-A4

TABLE OF CONTENTS

INTRODUCTION

In this document, we will discuss how these buttons and LEDs were designed for user input and interaction in the Rock-Paper-Scissors-Lizard-Spock game.

OBJECTIVE.

Our main aim was to create a hardware-based version of the Rock, Paper, Scissors, Lizard, and Spock game. It should be simple enough for two players (User and Computer) to compete with it physically yet provide clear feedback through buzzers as well as LED lights and ensure that the game logic is unbiased.

SYSTEM OVERVIEW.

The following components make up our system:

- **Input**: Five buttons for the player's choices (Rock, Paper, Scissors, Lizard, Spock), and an extra button to start or end the game.
- **Output**: 3 LEDs to show the player’s marks,
  3 LEDs to show the computer’s marks,

3 LEDs to show the computer's choices,
1 LED as an indicator,
and a buzzer for sound feedback.

- **Microcontrolle**r: This part processes inputs and controls outputs according to game logic.

DESIGN CHOICES

Choosing How to Input.

For each of the choices (Rock, Paper, Scissors, Lizard, and Spock) physical buttons have been used as well as an additional button that starts or ends the game. In choosing them, however, we settled on buttons since they are easy and reliable for users.

Buttons:

- Button 0: Start/ End Game

- Button 1: Rock

- Button 2: Paper

- Button 3: Scissors

- Button 4: Lizard

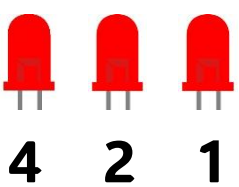
- Button 5: Spock

Determining How to Output

We use LEDs for output. They show the choices made by computer and marks of both players in a binary representation way. We have also added a buzzer which make it more fun because each time a round starts / ends or game ends, a sound will be made.

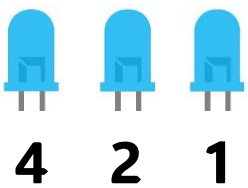
How the binary representation works?

For Marks



Player needs to get the addition of values under turned on LEDs.

For Computer's Choice



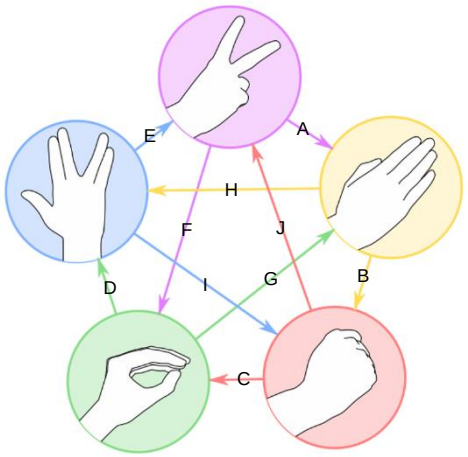
Player needs to get the addition of values under turned on LEDs. And check for gesture.

If
1 = "rock"
2 = "paper"
3 = "lizard"
4 = "scissors"
5 = "spock"

Game Logic

The microcontroller is responsible for handling the game logic. It determines the winner based on the rules of Rock-Paper-Scissors-Lizard-Spock, which is an extension of the traditional Rock-Paper-Scissors game. The rules are as follows.

A. Scissors cuts paper
B. Paper covers rock
C. Rock crushes lizard
D. Lizard poisons Spock
E. Spock smashes scissors
F. Scissors decapitate Lizard
G. Lizard eats paper
H. Paper disproves Spock
I. Spock vaporizes rock
J. Rock crushes scissors



Game Flow

1. Start Game:

- To start the game the player simply presses Button 0.
- The system initializes. It starts round 1.
- The game's start is signaled by a buzzing sound. The LED indicators light up.

2. Player Choice:

- Players make their choices. Rock Paper, Scissors Lizard, Spock. Buttons 1 to 5 are used. (Choices need to be confirmed within 3 seconds.)
- Computer’s choice is randomly generated.
- If no selection is made by the player within the time frame computer gets 1 mark.

3. Round Evaluation:

- The system checks the button states.
- It compares the choices made by player and computer and determines the round's winner.
- The scores are updated. The result is displayed using LEDs.

4. End Game:

- After 7 rounds the game comes to an end.
- The final scores are shown.
- To display the winner feedback pattern is played. This includes a buzzer and LED.

- Pressing Button 0 allows you to reset and start a new game.

This combination of system prompts, and user actions aim to make it easy to ther user to play the game.

User Interface

As the basic milestone of our project, we are not intended to develop a GUI (Graphical User Interface)

Each player has a set of LED lights showing their marks while LEDs light up when mark is added.

To indicate the 3 second timeout, there is a separate LED allocated.

A buzzer beep at the end of each round, whenever the game starts or stops.

The choices made by computer is displayed using 3 LEDs. And winners are indicated using a LED blink pattern and a buzzer pattern.

DETAILED EXPLANATION OF THE FLOW CHART DESIGN

Page 01

1. Initialization (Start)

- Start node initializes variables:
  - button 0 = start/end button
  - button 1 = rock
  - button 2 = paper
  - button 3 = scissors
  - button 4 = lizard
  - button 5 = spock
  - player_marks = 0
  - computer_marks = 0

We begin by initializing the buttons with relevant gestures to ensure clarity in the diagram's flow. Additionally, we set up the player and computer marks. The game can be started by pressing button 0.

2. Initial Wait

- wait 3 seconds

> We've introduced a waiting time into the program to allow players to read instructions and better understand the game's flow.

3. Game Start / End Check

- Button 0 state check
  - If button 0 is OFF state, print "Game End" and move to the end.

> We monitor the state of button 0 since it serves as the toggle button in our game. If button 0 enters the ON state, the flow chart execution continues.

4. Game Introduction

- Print game instructions.
- Wait 3 seconds.
- Start round = 1.

5. Round Start

- Print "Game Started".
- If round <= 7, print round instructions.
- Wait 2 seconds & continue.

6. Gesture Input

> In the game, players have 3 seconds to input their gesture. To signal the start of this time window, a specialized buzzer tone (referred to as 'buzzer tone (start)' in the flow chart) is activated. The get_time() function records the current time, and a separate indicator light illuminates to indicate that the player can choose a gesture. The difference between the current time and the start time is then calculated to determine whether the 3-second limit has been reached.
> Also, here we use a variable called change to track the state changes of any button.

- Check all buttons (for gesture input) within a time limit (3 seconds).
- Capture gestures: rock, paper, scissors, lizard, spock.
- Randomly generate computer gesture.
- Computer gesture is indicated (continue in page 2 through connector G).

7. Gesture Comparison

In the flowchart, the gesture comparison process is linked to page 2 via an off-page connector labelled ‘W’

- Compare player gesture with computer gesture.
- Print results.
- Update marks accordingly.
- Display marks and wait 2 seconds.

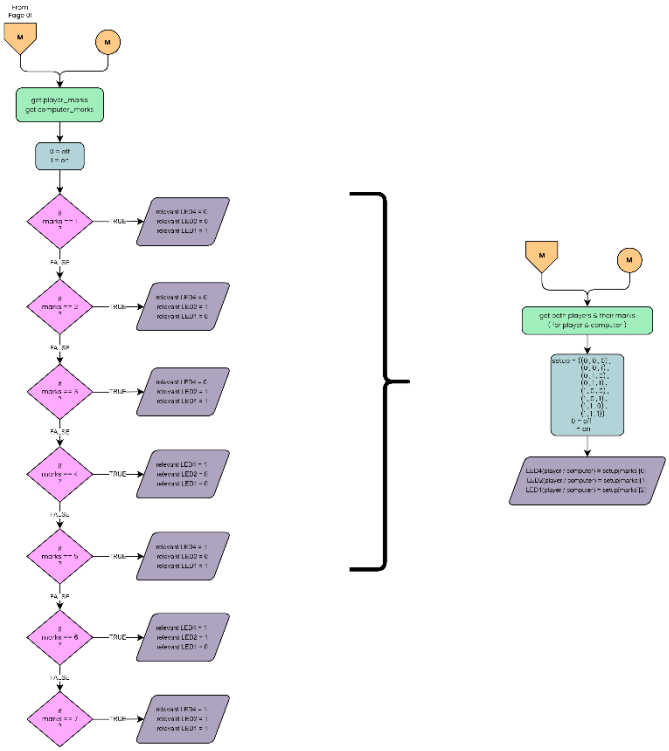
8. Round Update
- Increment round.
- Loop back for next round or end game.

End of Page 01
The flow proceeds to Page 02 as indicated by the connectors.

Page 02

1. Retrieve Player and Computer Marks & Display
- Setup LED outputs for marks.

Instead of using numerous decision blocks, we optimized our diagram by utilizing a tuple-like approach to predefine the LED states for each mark. It is shown in the diagram below.



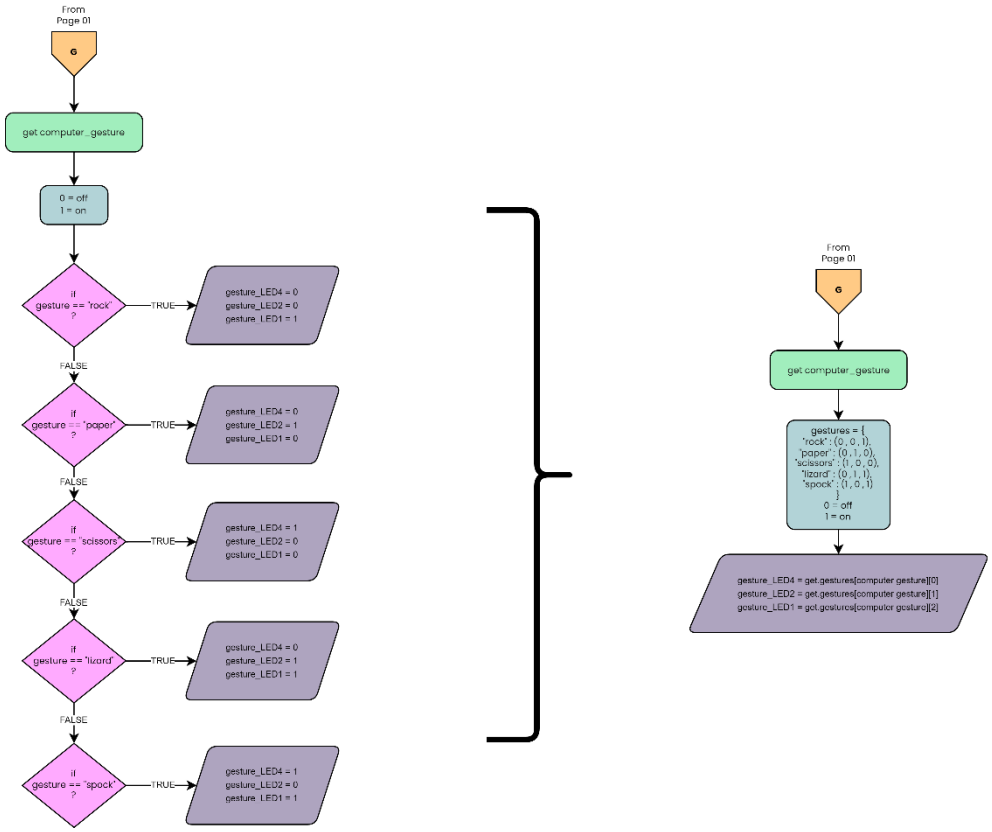
LED4 corresponds to the LED that indicates 4 marks, while LED2 and LED1 represent the LEDs indicating 2 marks and 1 mark, respectively. The process of the diagram is described below using an example.

Example:
Sub-tuple Selection:
When player_marks = 5, we retrieve the sub-tuple corresponding to 5 marks from the setup tuple: setup[5] = (1, 0, 1).
This sub-tuple contains three elements: (1, 0, 1).
LED States:
We apply the states of these sub-elements to the relevant LEDs:
player_LED4 is turned on because setup[5][0] = 1.
player_LED2 is turned off because setup[5][1] = 0.
player_LED1 is turned on because setup[5][2] = 1.

2. Computer's Gesture Indication

- Display the gesture

Here also, instead of using numerous decision blocks, we optimized our diagram by utilizing a dictionary-like approach to predefine the LED states for gesture. It is shown in the diagram below.



Similarly, in binary representation, we use gesture_LED4 to indicate 4, and similarly, gesture_LED2 and gesture_LED1 represent 2 and 1, respectively.

Example:
When the computer chooses ‘paper’ as its gesture, we retrieve the corresponding tuple from the gestures dictionary. For ‘paper’, the tuple is (0, 1, 0).

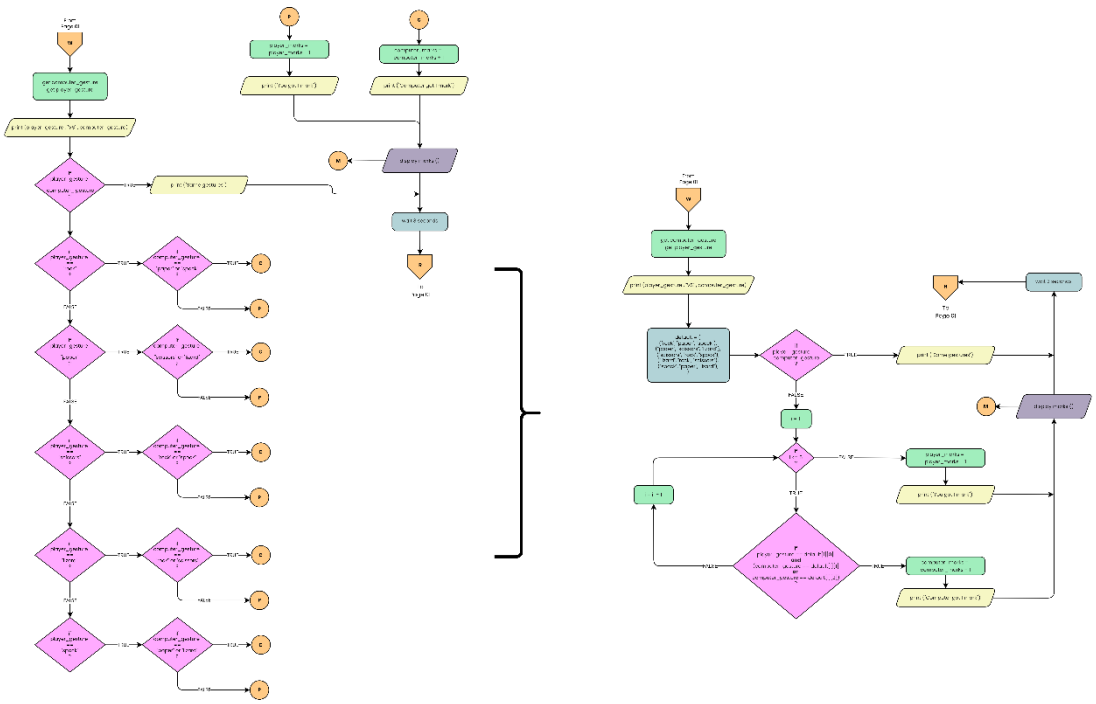
Next, we apply the states relevant to the sub-elements for the LEDs:

gesture_LED4 is turned off because gestures.get("paper")[0] = 0.
gesture_LED2 is turned on because gestures.get("paper")[1] = 1.
gesture_LED1 is turned off because gestures.get("paper")[2] = 0.

3. Gesture Comparison Logic

- Get player and computer gestures from previous inputs
- Check if player gesture equals computer gesture.
- If same, print "same gesture" and loop back.
- If different, determine winner and update scores.

Here is how we optimized our diagram:



In our game, we utilize a tuple to define the default rules. The process checks whether any of these combinations occur during the game. If such a combination occurs, the computer earns one mark according to our logic. Additionally, we compare the player's gesture with the computer's gestures that can defeat the player. If both gestures match, the game proceeds to the next round.

CONCLUSION

This design ensures that the Rock-Paper-Scissors-Lizard-Spock game is not only fun but also interactive for users. The hardware buttons and LEDs give players instant feedback. The microcontroller handles the game logic. It ensures accuracy and fairness. Adding sound feedback enhances overall interactivity. The enjoyment of the game is significantly improved.

IMPORTANT

You can download a clear PDF of the diagram optimizations using the link provided below:

https://drive.google.com/file/d/1ACt8q22IQmhY8ytVsSDMVZyferAktw7B/view?usp=sharing