Contents

Introduction:

The program is designed to go an adventure with the Swift-bot. Start your journey by scanning a QR code to set the zigzag parameters. The program has different functionalities and has different modes. Users can select their own mode and pattern for the movement of the bot. The program also lets you click pictures with the help of a single key on your keyboard. The command line interface is user-friendly, and anyone can understand it without any difficulties. Explore amazing functionalities as you go on a journey with the Swift-Bot!

Requirement Specifications

Functional Requirements:

1) The User should be able to scan a QR code.
2) The Program should be able to decode a QR code.
3) The QR code should have the length of zigzag sections(zigLength)
4) The QR code should have the number of zigzag sections(zigNumber)
5) Length of a zigzag section(zigLength) should be between 15cm to 85cm.
6) The number of zigzag sections(zigNumber) should be even.
7) The number of zigzag sections(zigNumber) should not exceed 12.
8) The program should generate a random speed for the wheels (randomSpeed).
9) The program should ask the user for mode selection.
10) The program should be able to set the mode to selected [Mode]
11) The program should ask the user for Pattern selection.
12) The program should be able to set the pattern to selected [Pattern]
13) The program should ask the user for colour inputs.
14) The program should be able to set the first underlights to [First Colour Input]
15) The program should record current time (startTime)
16) The program should be able to move the Swiftbot forward for the specified zigzag length(zigLength)
17) The bot should be able to stop for 1second after completion of each zigzag length(zigLength)
18) The bot should take a turn after stopping.
19) The program should be able to set the second underlights to [Second Colour Input]
20) The program should be able to move the Swiftbot forward for the specified zigzag length(zigLength)
21) The program should be able to retrace the movement after completion of number of zigzag section(zigNumber)
22) The program should be able to retrace movement with the selected underlight.
23) The Program should record current time again(stopTime)
24) The Program should compute duration(stopTime-startTime)
25) Store length of each zigzag section(zigLength) into a text file.
26) Store number of zigzag sections(zigNumber) into a text file.
27) Store randomly generated wheel speed(randomSpeed) into a text file.
28) Store length(totalLength) of the traversed path (start to end) excluding travelling back into a text file.
29) Store Duration (duration) in seconds that the robot took to complete the move from Start to End into a text file.
30) Store the distance travelled (straightDis) from the Start to the End of the zigzag into a text file.
31) The program should terminate when button ‘X’ is pressed on the SwiftBot.
32) Display the number of zigzag(zigzagJourney) journeys completed before termination.

Non-functional Requirements:

1) The command-line interface should be user friendly, with clear instructions.
2) The program should respond to user inputs and commands promptly, providing a seamless and efficient user experience.
3) The program should be stable and reliable, with minimal chances of crashes or unexpected terminations.
4) Error messages should be clear and informative, aiding users in understanding and rectifying issues.
5) The code should be well-organized, with clear comments.
6) The randomly generated speed should be between 10 to 100.
7) The program should respond to user inputs and commands promptly, providing a seamless and efficient user experience.
8) The program should be able to handle different patterns and distances without a significant impact on performance.
9) The program should store the captured photos in a folder.

Error Handling:

1) Display an error message if the user provides invalid zigzag section length(zigLength)
2) Display an error message if the user provides invalid zigzag section number(zigNumber)
3) Display an error message if there is an error during data logging (writing to the text file)
4) Display an error message if the Swiftbot fails to terminate when the ‘X’ button is pressed.
5) Display ‘Select pattern’ message again if a different key is pressed when selecting pattern.
6) Display ‘Select Underlight’ message again if a different key is pressed when selecting underlight.
7) Display ‘Select mode’ message again if a different key is pressed when selecting mode.

Additional Functionalities:

1) Allow users to customize the LED colours for sections.
2) Allow users to choose between predefined zigzag patterns (zigzag and Squareloop)
3) An energy-saving mode that disables all the underlight to extend battery life.
4) Allow users to capture photo (Pressing ‘C’) during the journey of the robot.

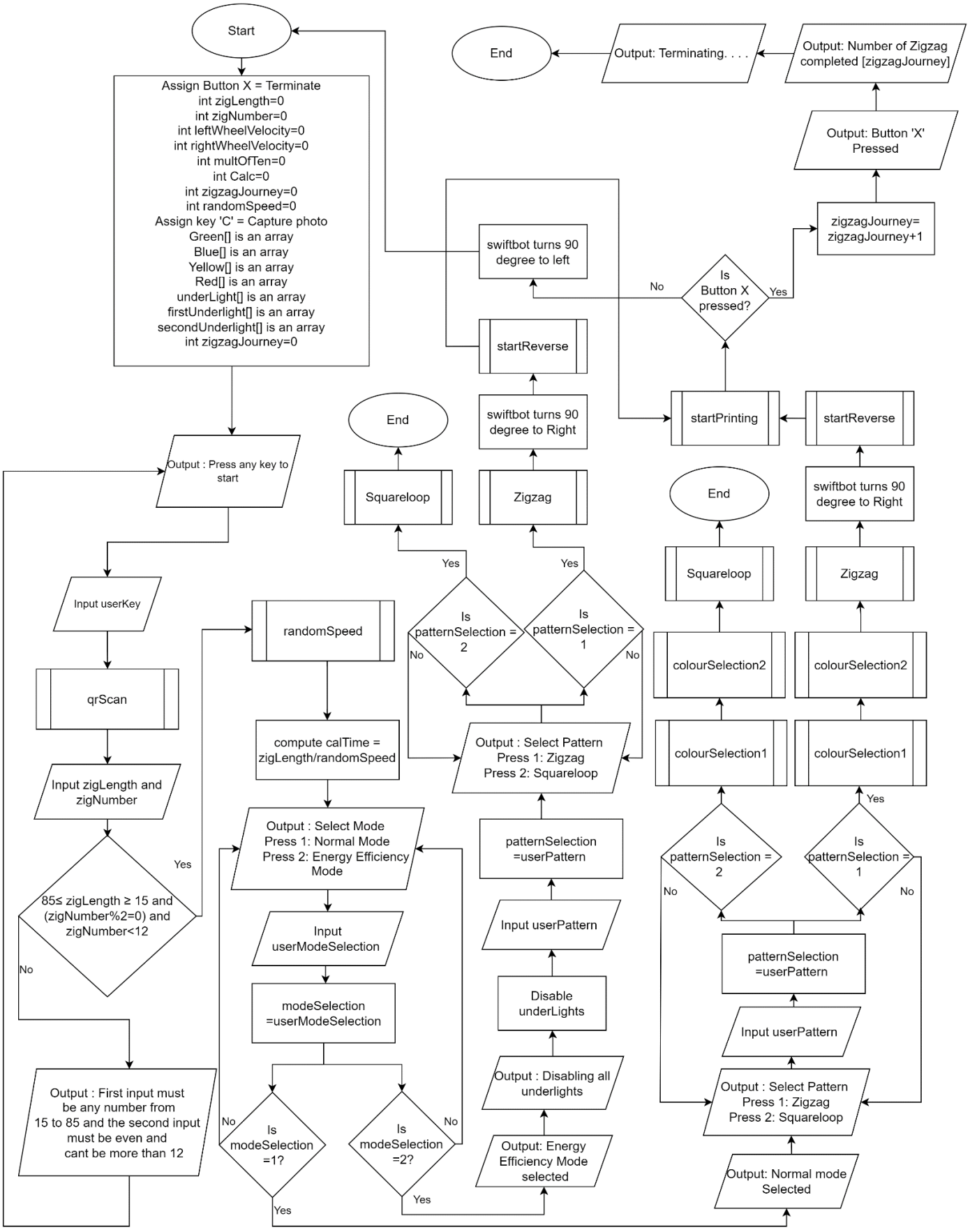
Constraints:

1) The program should adhere to the given specifications for zigzag section length, number of sections, and wheel speed.
2) The Swift-bot’s speed will be only multiple of 10 and not more than 100.
3) For Squareloop pattern, Swift-bot can make a maximum of 2 loops.
4) For custom colour selection, there is only four colours: Red, Green, Yellow and Blue
5) If Squareloop is selected, it cannot be looped again and again.

Flowchart (Algorithm Design)

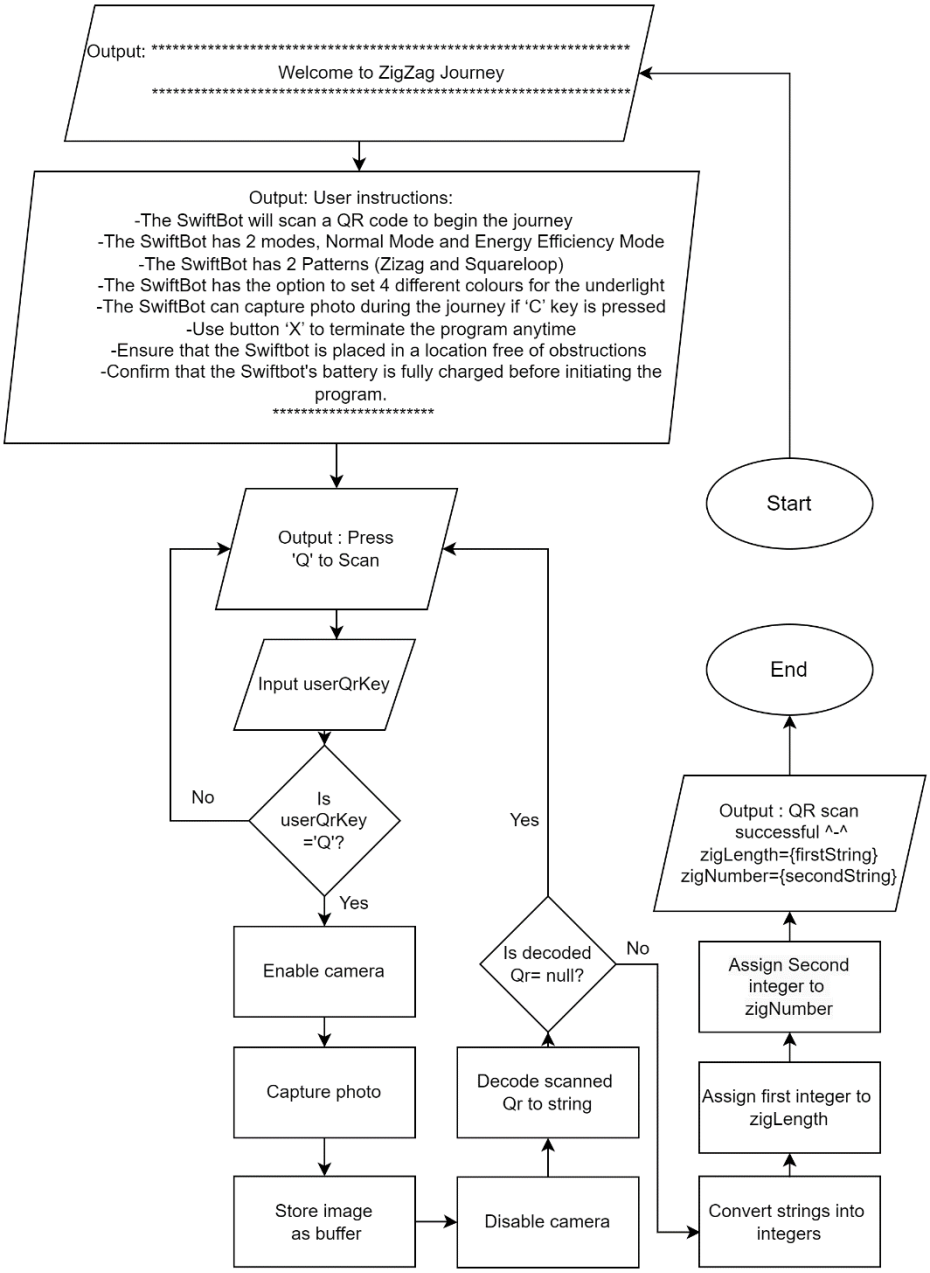
Main sequence:

This is main sequence of the flowchart with small sub processes having different functionalities of the Swift-bot. Main sequence of the flowchart helps us to know how a program is executed.



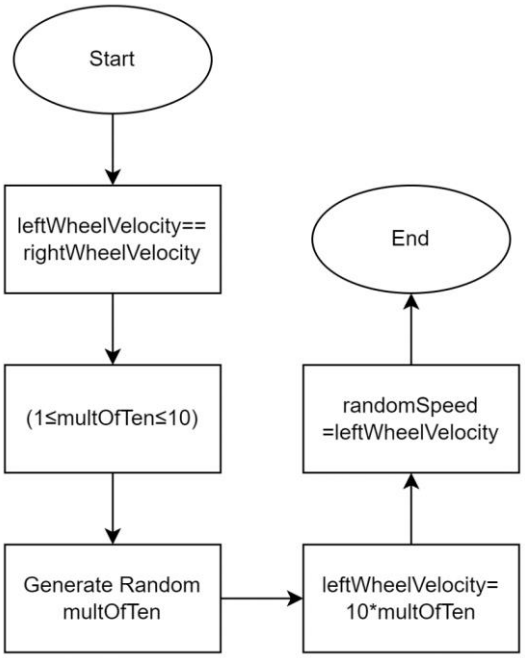
Sub-process: qrCode

This sub-process helps us to decode the QR and store the string values inside two integers and reject null inputs.



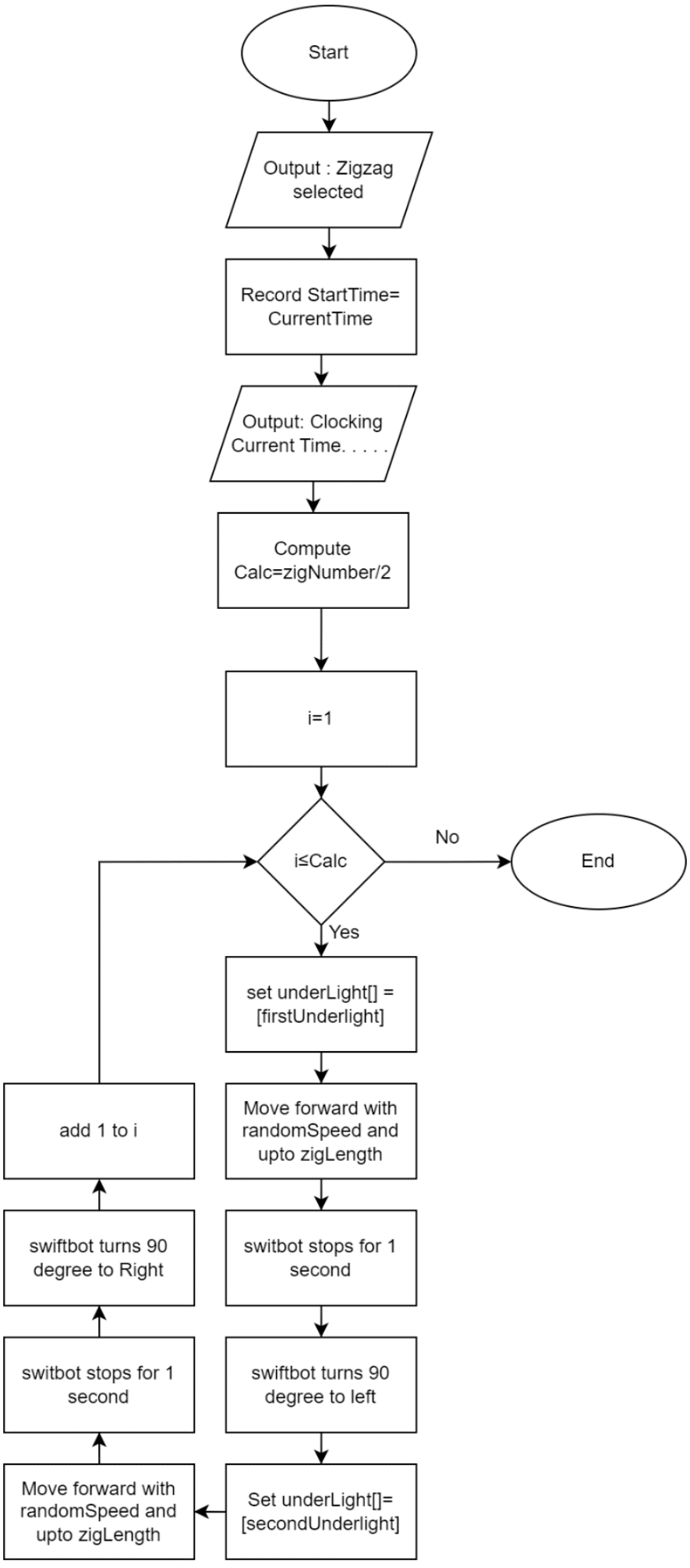
Sub-process: randomSpeed

This sub-process generates random wheel speed. The speeds are multiples of 10. So possible speed values are 10, 20, 30 …….. 100.



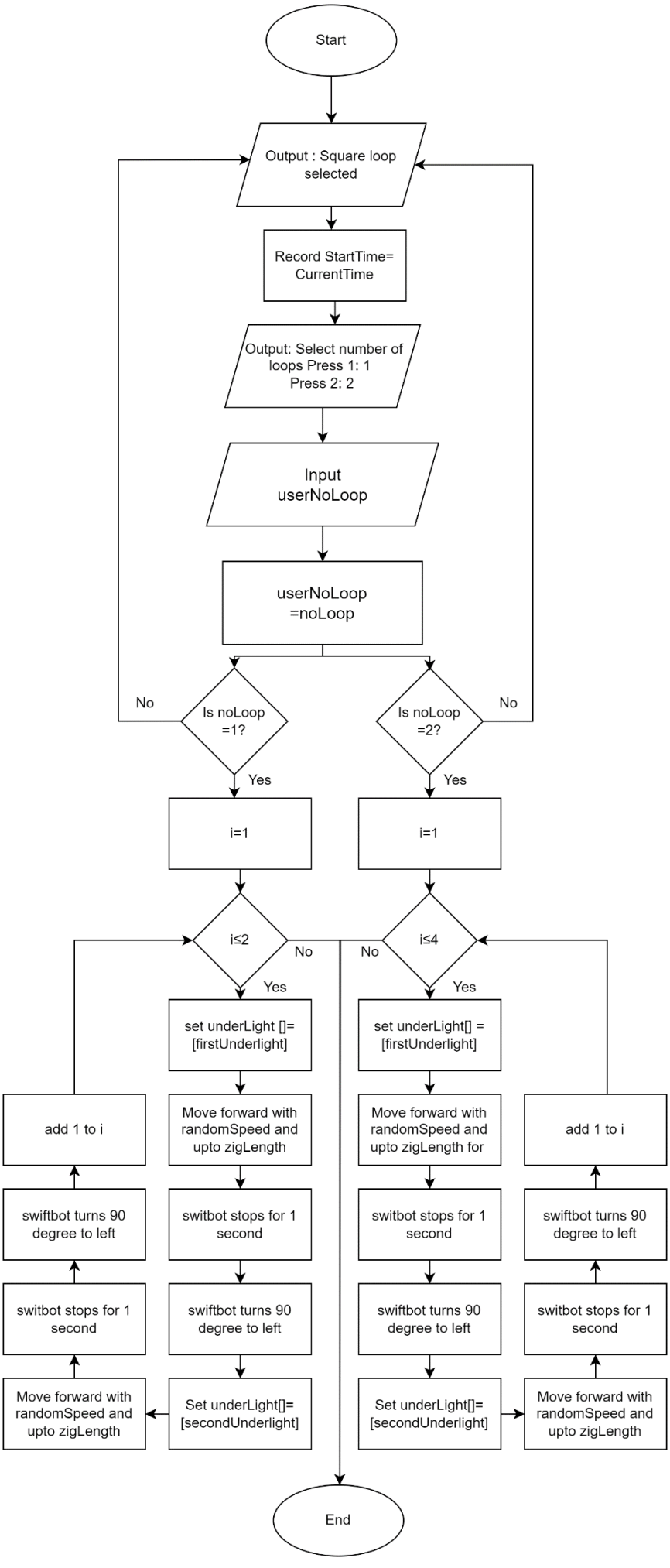
Sub-process: Zigzag

This sub-process helps us to understand the movement of the bot when zigzag pattern is selected.



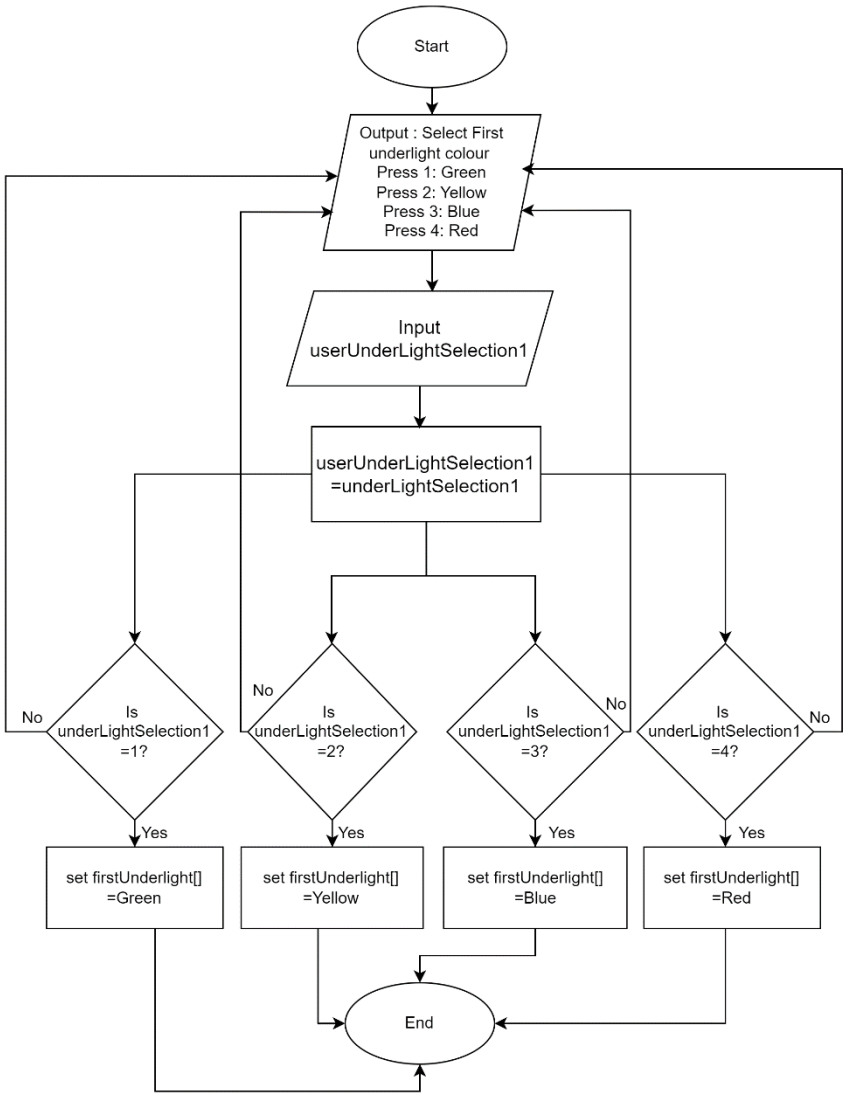
Sub-Process: Squareloop

This sub-process helps us to understand the movement of the bot when Squareloop pattern is selected.



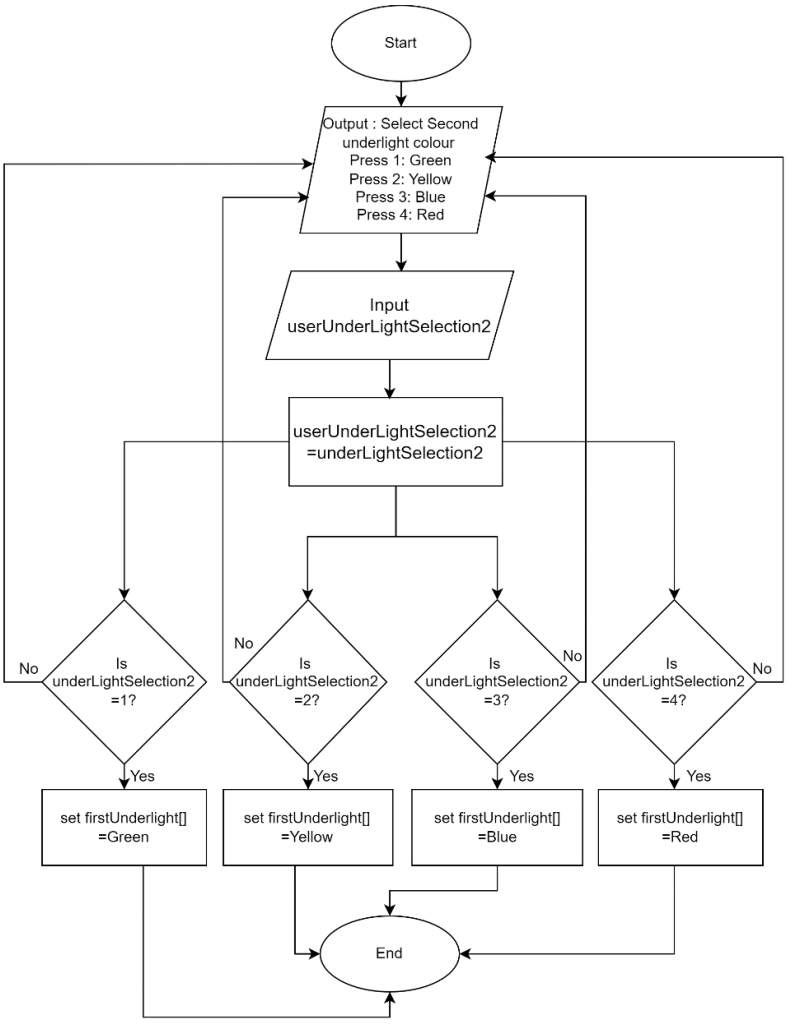
Sub-process: colourSelection1

This sub-process helps us to understand the first colour selection process when normal mode is selected.



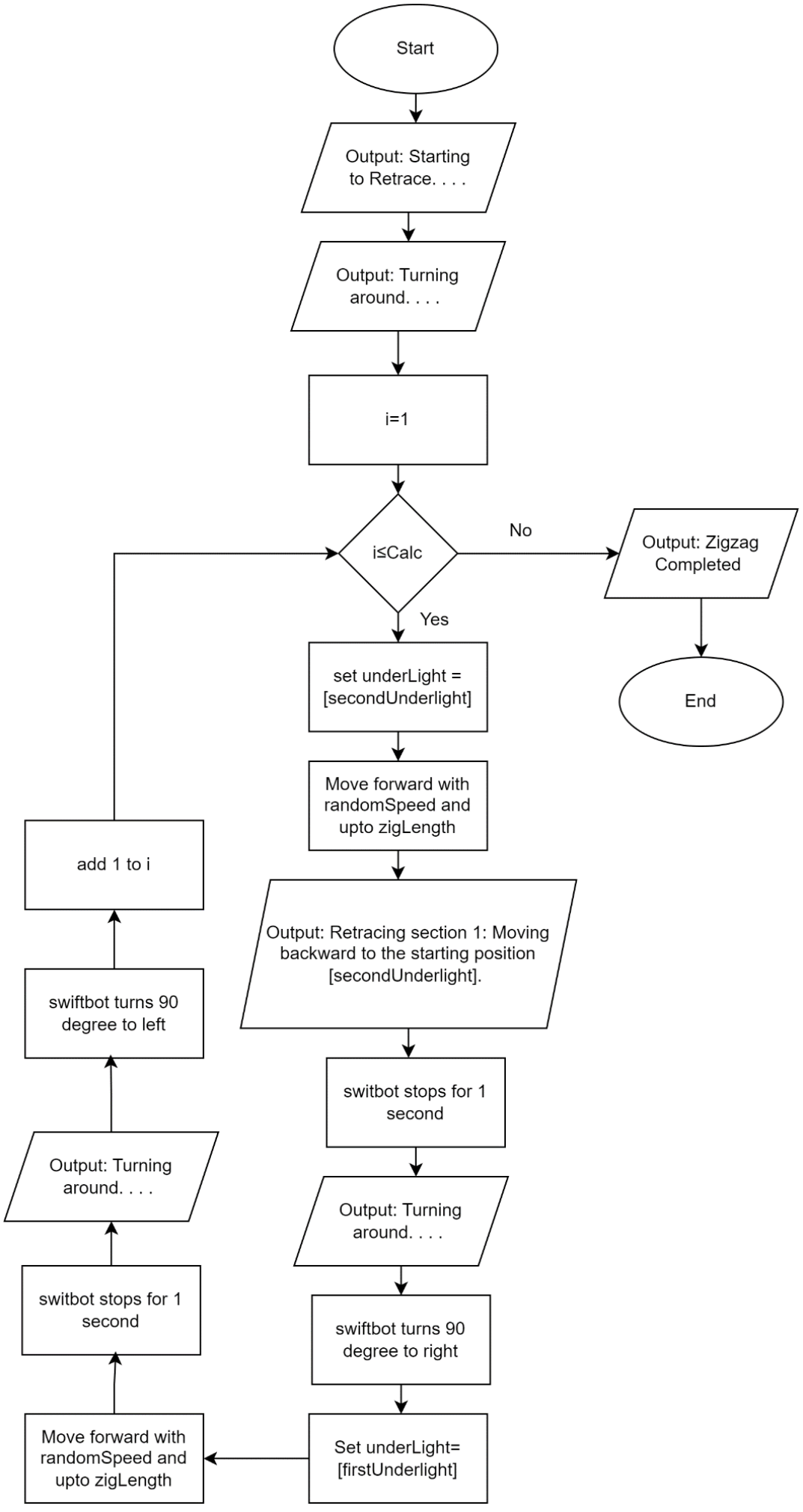
Sub-process: colourSelection2

This sub-process helps us to understand the Second colour selection process when normal mode is selected.



Sub-process: startReverse

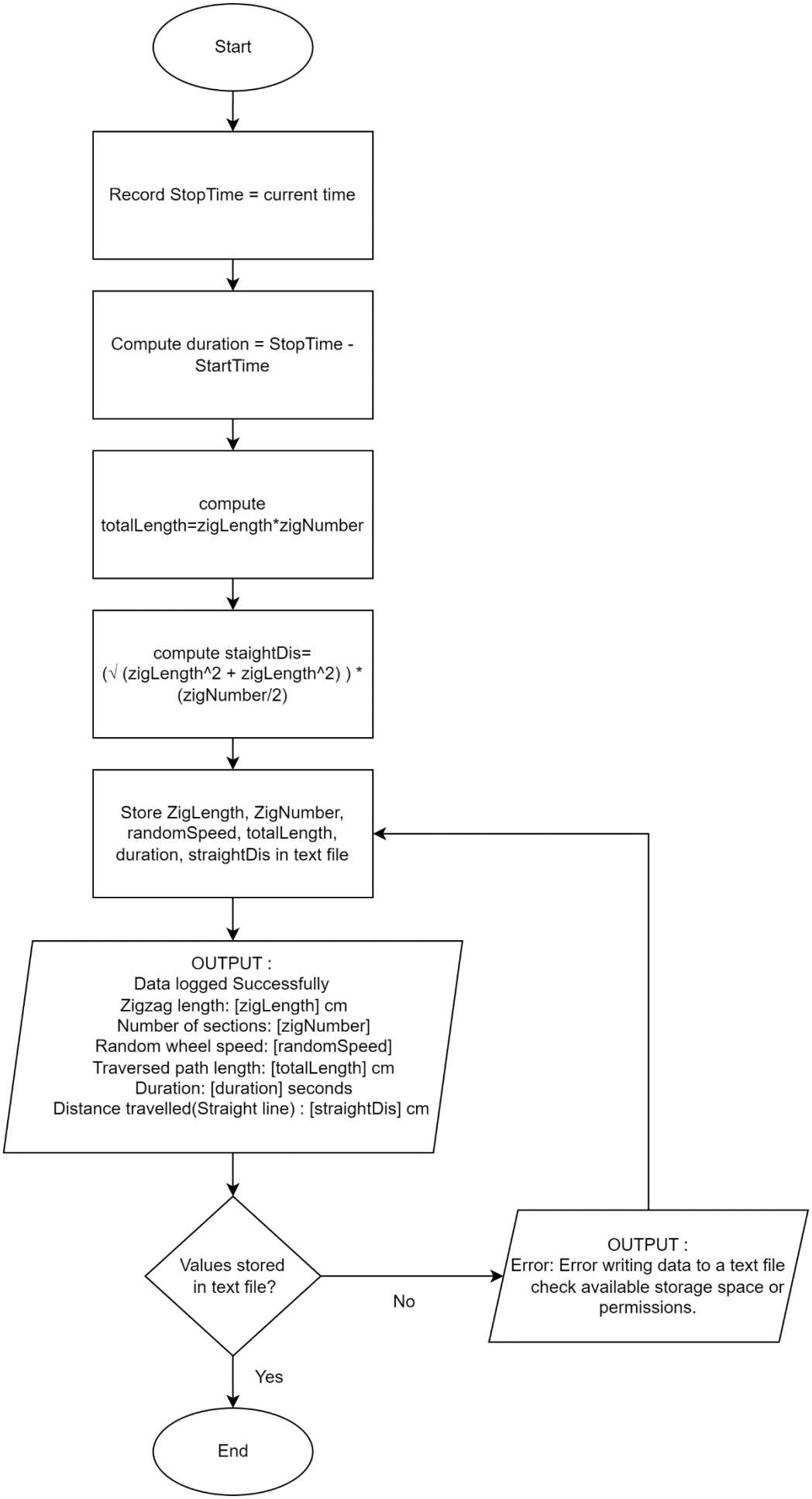
This sub-process helps us to understand the how Swift-bot will trace its way back to the starting position.



Sub-process: startPrinting

This sub-process helps us to understand the how Swift-bot will compute staightDis and totalLength;

This process will also show us how the program will store different variables in a text file.



Command Line Interface Design:

The Command line interface of the Swift-Bot movement is user-friendly; Users can easily give commands using different keys and get real-time feedback. For every invalid output by the user, the program smartly handles the error and guides users on the correct input or suggests appropriate actions to rectify the issue. All the instructions can be found in the very first page of the command line interface.

Swiftbot-Command Line Interface Design

Instruction Page:

The program initiates as soon as a key on keyboard is pressed

```
SwiftBot-Command line Interface

>> Press any key to start
>> 2
                ********************************************************************
                                    Welcome to ZigZag Journey
                ********************************************************************

User instructions:
        -The SwiftBot will scan a QR code to begin the journey
        -The SwiftBot has 2 modes, Normal Mode and Energy Efficiency Mode
        -The SwiftBot has 2 Patterns (Zizag and Squareloop)
        -The SwiftBot has the option to set 4 different colours for the underlight
        -The SwiftBot can capture photo during the journey if 'C' key is pressed
        -Use button 'X' to terminate the program anytime
        -Ensure that the Swiftbot is placed in a location free of obstructions
        -Confirm that the Swiftbot's battery is fully charged before initiating the program.

                                ***********************
```

Welcome to our Command Line Interface (CLI) instruction page, designed to empower users with the essential knowledge and skills needed to navigate and leverage the full potential of command line tools

Here are guidelines to inform users about the limitations of the bot and its capabilities. Additionally, there are instructions on how users can choose their preferred mode, pattern, and color, introducing some extra functionalities.

Effortlessly interact to scan QR! Simply press 'Q' to activate the scanner. Once activated, observe the cheerful '^_^' confirmation upon a successful scan. Discover details about zigzag sections, including their length in centimeters ([zigLength]) and the total number detected ([zigNumber]). Enjoy seamless QR scanning directly from the command line interface.

Successful Scan:

```
>> Press 'Q' key to scan the QR
>> 'Q' pressed
>> Enabling camera. . . .
>> QR scan successful ^-^
>> Length of zigzag section (cm): [zigLength]
   Number of zigzag sections: [zigNumber]
```

Unsuccessful Scan:

```
>> Press 'Q' key to scan the QR
>> 'R' pressed
>> Press 'Q' key to scan the QR
>> Error: First input must be any number from 15 to 85 and the second input must be even and
cant be more than 12
>> Press 'Q' key to scan the QR
```

Upon pressing 'R' or any wrong key, the system prompts you with instructions to input specific values. If an error occurs due to improper inputs, a clear message guides you: the first input should be a number between 15 and 85, and the second input must be an even number, not exceeding 12.

With these intuitive steps, Normal Mode transforms your journey visually appealing.Uniquely enjoyable experience that aligns with your aesthetic preferences.

Immerse yourself in a streamlined experience tailored to your preferences. Whether you opt for the efficiency of Energy Efficiency Mode or the versatility of Normal Mode, enjoy a focused and personalised Swift-bot journey.

```
Mode Selection
    >> Select Mode
    >> Press 1: Normal Mode
    >> Press 2: Energy Efficiency Mode
    >> [Mode] Selected
[Normal Mode]

  >> Select First underlight colour
  >> Press 1: [Green]
     Press 2: [Yellow]
     Press 3: [Blue]
     Press 4: [Red]
  >> First underlight set to [firstUnderlight]
  >> Select Second underlight colour
  >> Press 1: [Green]
     Press 2: [Yellow]
     Press 3: [Blue]
     Press 4: [Red]
  >> Second underlight set to [secondUnderlight]
```

Personalise your interface in Normal Mode with vibrant underlight colours or conserve energy in Energy Efficiency Mode by disabling colours.

In Normal Mode, the CLI offers a dynamic and visually enriching experience through customizable underlight colors.User is able to pick any 4 colours ; Red, Green , Blue and Yellow

Activate Energy Efficiency Mode to enhance your command line experience. This mode optimises power consumption by systematically turning off all underlights. Enjoy a streamlined and energy-conscious journey, ensuring prolonged battery life without compromising functionality.

```
[Energy Efficiency Mode]
   >> Energy Efficiency Mode selected
   >> Disabling all underlights. . ..
[Pattern Selection]
   >> Select Pattern
      Press 1: Zigzag
      Press 2: Squareloop
[Zigzag]
   >> Zigzag selected. . .
   >> Clocking Current Time. . . . .
   >> Setting underlights to [firstUnderlight]
   >> Zigzag section 1: Moving forward for [zigLength] cm
   >> Setting underlights to [secondUnderlight]
   >> Zigzag section 2: Moving forward for [zigLength] cm.
   >> . . . . .
   >> Zigzag Completed
[Squareloop]
   >> Squareloop selected
   >> Select number of loops
      Press 1: 1
      Press 2: 2
```

Explore two captivating patterns; Zigzag and Squareloop.Zigzag pattern makes the bot travel in a zigzag pattern whereas the Squareloop makes the create a square .Select your preferred pattern using '1' or '2'.Bring a dash of creativity to the Swift-Bot journey with Pattern Selection.

Embark on a dynamic Swift-bot journey with Zigzag. The Swift-bot gracefully moves through sections, each [zigLength] cm forward. Underlights shift seamlessly, creating a visually captivating zigzag pattern. As time is logged, the CLI concludes with a confirmation of Zigzag completion.

As you make your selection, watch the Swift-bot come alive with the captivating Squareloop pattern.Immerse yourself in the Squareloop CLI, where the specified number of loops synchronises seamlessly with underlights, turning your Swift-bot into a visual spectacle.You will be asked to select the number of loops using ‘1’ and ‘2’ key on your keyboard.Select and enjoy the magic of Squareloop with just a press of a key!

Dive into Swift-bot's retracing journey, where every step is carefully followed, leading to a visually dynamic finish of the zigzag pattern. Experience the charm of CLI interactions as it smoothly reflects its own path.

```
[Retracing]
  >> Starting to Retrace. . . .
  >> Turning around. . . .
  >>
  >>
  >>
  >> . . . . .
  >> Zigzag Completed
[Termination]
  >> Button 'X' Pressed
  >> Number of Zigzag completed [zigzagJourney]
  >> Terminating. . . .
  >> Terminated
[Capturing Photo]
  >> 'C' Pressed
  >> Capturing photo. . . .
  >> Photo captured successfully. . . .
```

Engage with the CLI design through the following steps:
Button 'X' Pressed:
Initiate Termination
Zigzag Journey Completion Count:
Track the number of completed zigzag patterns - [zigzagJourney].
Terminated:
Wait for the confirmation as the termination process comes to a close And shows 'Terminated'

Photo Successfully Captured: Receive confirmation that the photo capture has been successfully completed.

Photo Capture in Progress: Observe as the CLI diligently works to capture your photo, indicated by the ongoing process.

'C' Key Pressed: Trigger the photo capture process with a simple press of the 'C' key.

Error Handling:

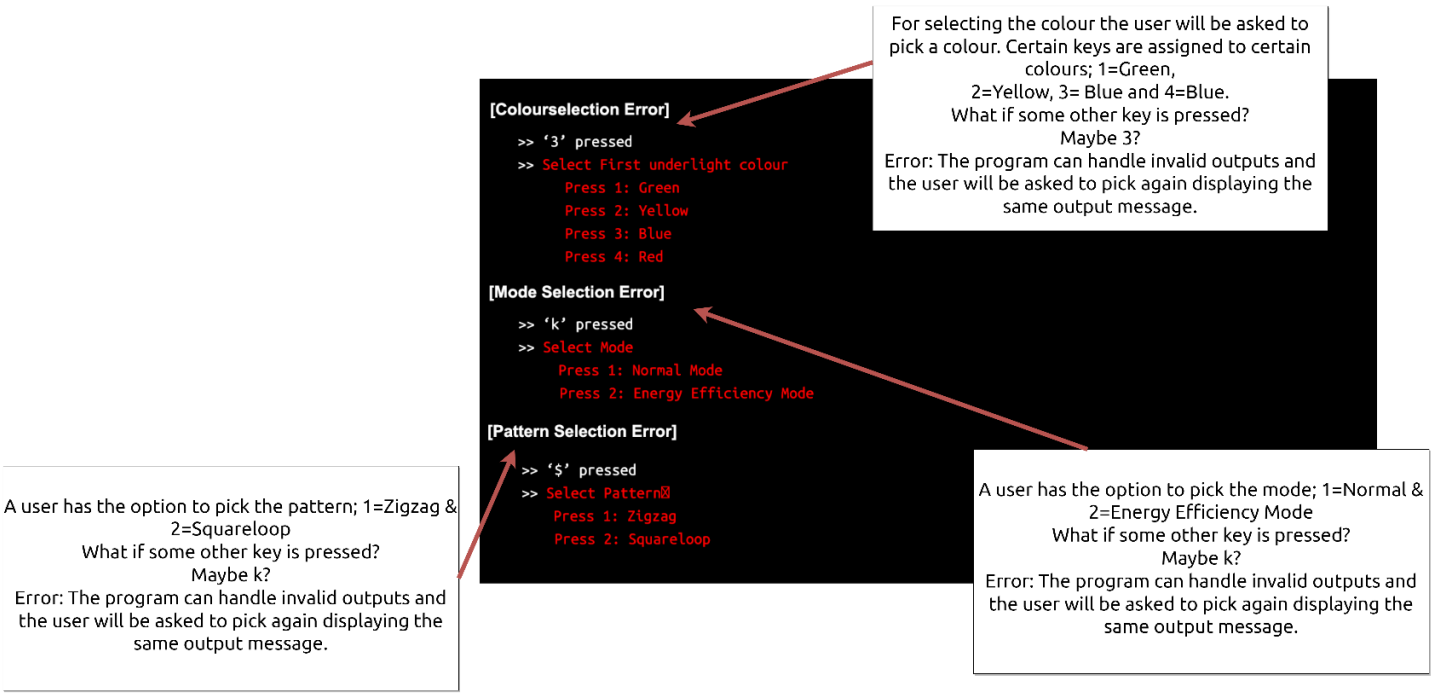
Explore a comprehensive summary of your Swift-bot journey, capturing details on zigzag patterns ([zigLength] cm in [zigNumber] sections), wheel speed ([randomSpeed]), traversal path length ([totalLength] cm), duration ([duration] seconds), and straight-line distance travelled ([straightDis] cm).

```
[Writing data to a text file]
  >> Data logged Successfully
     Zigzag length: [zigLength] cm
     Number of sections: [zigNumber]
     Random wheel speed: [randomSpeed]
     Traversed path length: [totalLength] cm
     Duration: [duration] seconds
     Distance travelled(Straight line) : [straightDis] cm
[Unsuccessful Data logging]
  >> Error: Error writing data to a text file
     check available storage space or permissions.
  >> . . . .
  >> [Trying again]
[colourSection1 Error]
  >> 'Z' is pressed
  >> Select First underlight colour
     Press1: Green
     Press2: Yellow
     Press3: Blue
     Press4: Red
  >> . . . .
```

Prompt: "Z is pressed."
Upon pressing any key other than 1, 2, 3, or 4,' the CLI gracefully reverts to the initial stage, guiding the user to select the first underlight colour:
Error Handling : Invalid inputs are not accepted and asks for the correct input again and again until correct input is received from the user.

Experience a user-friendly error-handling mechanism that acknowledges and rectifies issues during data writing. The program will make a concerted effort to overcome the error and resume its intended functionality.

Error Message: An error message will be shown to the user with certain instructions
Retrying:
The program automatically attempts to write data to the text file again.



Explanation:

The Command line interface of the Swift-Bot movement is user-friendly; Users can easily give commands using different keys and get real-time feedback. For every invalid output by the user, the program smartly handles the error and guides users on the correct input or suggests appropriate actions to rectify the issue. All the instructions can be found in the very first page of the command line interface. Starting from the Landing page, to the very last page, the CLI coordinates with the main flowchart and Software requirements.

Planning and Monitoring progress:

Week	Task to complete	Task completed	Problems Encountered	Solution	Improvement
Week14	Software Requirement	Functional Requirement	Requirement was not atomic and clear	Took feedback from tutor on 8/1/24	Removed all extra lines and made it detailed by addding correct variables
Week15	Software Requirement	Non Functional Requirement	Null	null	null
Week15	Software Requirement	Error handling	Errors handling did not match flowchart	Fixed all the error handling by following the flowchart	Made a separate section for Error handling for better visualization
Week15	Flowchart	Basic Flowchart prepared	Notation problem and bad Error handling with no output message	Took my time to fix all the notation and added output messages	Fixed the fonts size
Week15	Command line interface design	Designed a basic one using microsoftword	Command line was not visually pleasing and without any callout box	Took feedback from tutor on 8/1/24 meeting	Changed everything and developed the whole Command line interface in photoshop using a font similar to Command prompt and added callouts to explain everything.
Week16	Command line interface design	Added error handling	Missed few error handling output text	Added error handling and marked them red	null
Week 17	Flowchart	null	Forgot to add input boxes when dealing with user inputs	Added input boxes where needed with proper variable names	Resized the fonts
Week18	Flowchart	Brushing up Flowchart	Minor problems like missing ‘yes’ or ‘no’ decisions	All the minor problems was identified by the module leader (Yasooda) in the lab and I took my time to fix everything	Resized flowchart for better visualization
Week19	Visual Presentation	Adding introduction, summary,page number,table of contents and conclusion	null	added	null

Summary

In short, Swift-bot follows the instructions of the user and completes its journey without overlapping the main requirements. Have a Safe Journey with your Swift-Bot!