ENGI1020 Lab Project Implementation Logbook

Reaction Game – Shair Yousuf Jahin

ENGI 1020, Winter 2021

Introduction

The game will serve as a way to improve one's reaction time to certain visual cues. The game will collect data on the performance of the player and after the end of a session, use the collected data to show a graph of the players reaction time and the number of correct answers they gave. This could be a part of a much larger game program that has many more small games using different input and output components. This can also be used by people who do work that requires them to have quick reflexes like professional athletes and professional gamers to warmup their reflexes before an event.

Final Design

This game will display the name of the color in text in the lcd screen while the screen will have a color, if the screen color and text are same then he will press the button meaning he thinks it is correct and if the text and the color is different, he will press the touch sensor meaning he thinks it is wrong. The reaction time of his answering is calculated and stored, which is later given back as a performance analysis in the form of a graph.

The overall design of my script can be found in Figure 1 below. The details of design for the different components are described within the notes about their implementation. As part of the overall design, one loop was used with to ask questions for a preset amount of time, another loop inside the previous one was used to check if input was given or not and then conditional statements were used to check if the user gave the correct answer or not. The reaction time of the user was stored in a list and the number of correct answers were stored in a variable. A function was used to create the graph of reaction time. The flowchart of my implementation is given below:

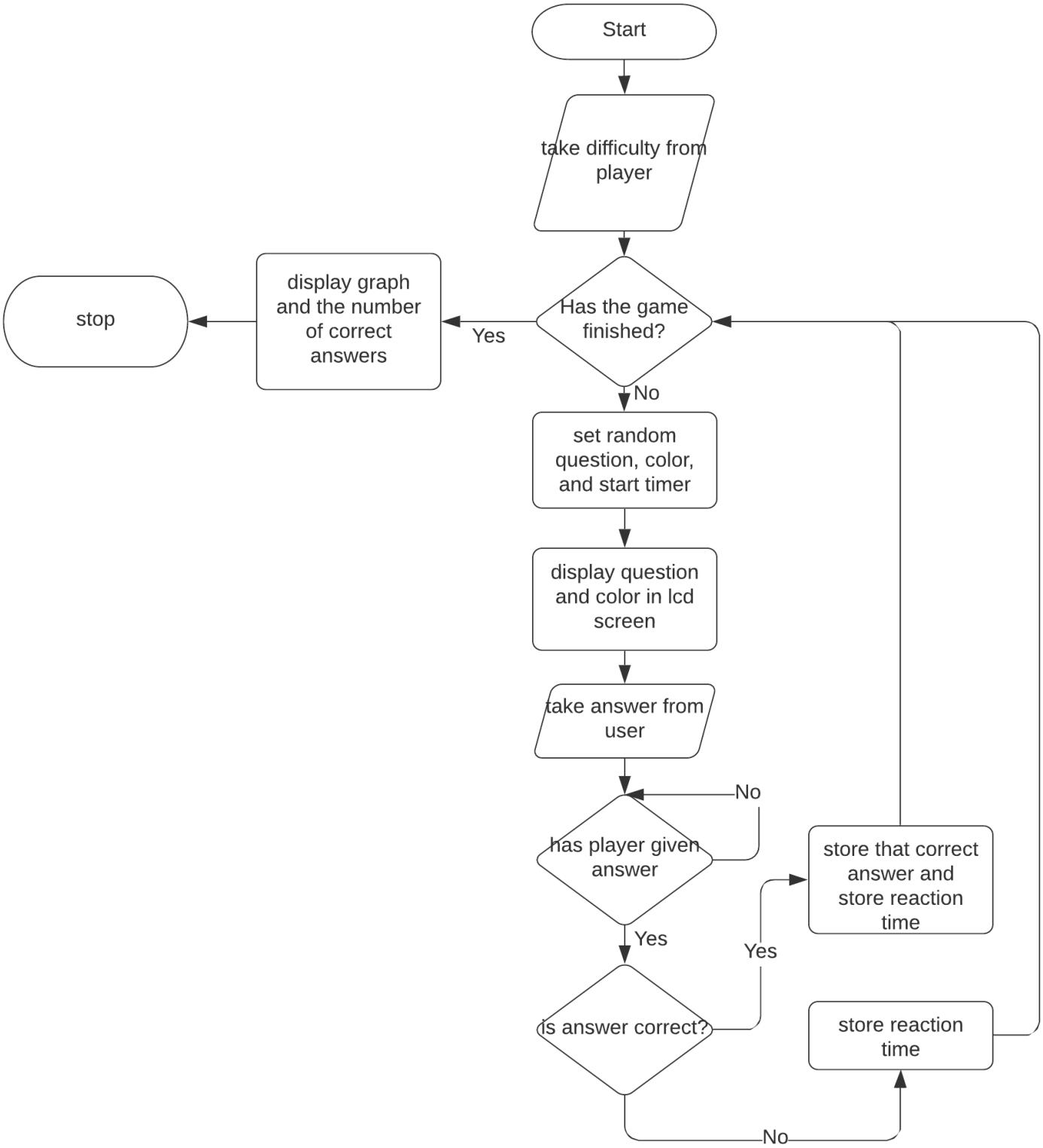


Figure 1: Flow chart of overall design

Implementation

The code used for my program is given below with explanations:

- from engi1020.arduino import *
- import time
- import matplotlib.pyplot as plt
- import random
- from time import sleep
-
- from builtins import input
    - The required modules were imported, engi1020.arduino to work with the sensors and Arduino, time to calculate reaction time and use the sleep function. The random module to make random number for question and color selection and the matplotlib.pyplot to plot the graph.
- def makegraph(xlist,ylist,xlabel,ylabel,title):
- plt.plot(xlist, ylist)
- plt.xlabel(xlabel)
- plt.ylabel(ylabel)
- plt.axis([0, max(xlist), 0, max(ylist)+1])
- plt.suptitle(title)
- plt.show()
    - This makegraph function was made to plot the graph at the end of game.it uses the number of questions and the list containing the reaction time, the labels for the x and y axis and the title of the graph as inputs.
- buttonpin=4
- touchpin=2
    - The pin number for the button and touch sensor is set in a variable.
- questions =[["Green",0.25],["Blue",0.5],["Red",0]]
    - This list is used to store the three questions used in the game, the first values are the text to be displayed in the lcd screen while the second values are the hue values for the corresponding colors. I got these values of hue from my experience with lab 6 where I implemented the changing of lcd screen color depending on the value from the temperature sensor.
- difficulty=float(input ("please enter the difficulty from 0.5, 1 and 2. The lower the number the harder it is: "))
    - this difficulty variable is used in the sleep function later on, basically it is used to set how quickly each question comes on the screen after answering.
- amount=int(input("please enter the number of questions you want to play for: "))
    - this amount variable stores the number of questions for which the user wants to play the game.
- correct=0
    - this stores the number of times the user got the correct answer.
- timelist=[]
- index=[]
    - The timelist list stores the reaction times of the user and the index list is a list that has the number of questions for the graph.
- for i in range(amount):

- a loop to keep looping until all question have been asked.

- question=random.randint(0,2)
- color=random.randint(0,2)
- sleep(difficulty)
  - Random number between 0 to 2 is set to question and color. The sleep function is used to set how quickly the question is asked.
- index.append(i)
- lcd_print(questions[question][0])
- lcd_hsv(questions[color][1],1,255)
- then=time.time()
- now=0
  - The questions and color are given in the lcd screen and the time is stored
- touch=0
- button=0
  - These variables are set to zero, they will store the input values from button and touch sensor.
- input=False
  - Variable storing a Boolean value of if input is received from any of the inputs.
- while input==False:
- touch=digital_read(touchpin)
- button=digital_read(buttonpin)
- if touch!=0 or button!=0:
- input=True
  - A loop that keeps running until an input is received from the button or touch sensor
- if ((questions[question][0]=="Green" and questions[color][1]==0.25) or (questions[question][0]=="Blue" and questions[color][1]==0.5) or (questions[question][0]=="Red" and questions[color][1]==0)) and button==1:
- correct+=1
- now=time.time()
  - Conditional statement to check if the color and the text match and if the user thinks it is correct by pressing the button. The time is stored in a variable.
- elif ((questions[question][0]=="Green" and questions[color][1]==0.25) or (questions[question][0]=="Blue" and questions[color][1]==0.5) or (questions[question][0]=="Red" and questions[color][1]==0)) and touch==1:
- now=time.time()
  - Conditional statement to check if the color and the text match and if the user thinks it is wrong by pressing the touch sensor. The time is stored in a variable.
- elif ((questions[question][0]=="Green" and questions[color][1]!=0.25) or (questions[question][0]=="Blue" and questions[color][1]!=0.5) or (questions[question][0]=="Red" and questions[color][1]!=0)) and button==1:
- now=time.time()
  - Conditional statement to check if the color and the text do not match and if the user thinks it is correct by pressing the button. The time is stored in a variable.
- elif ((questions[question][0]=="Green" and questions[color][1]!=0.25) or (questions[question][0]=="Blue" and questions[color][1]!=0.5) or (questions[question][0]=="Red" and questions[color][1]!=0)) and touch==1:
- correct+=1
- now=time.time()

    - Conditional statement to check if the color and the text do not match and if the user thinks it is wrong by pressing the touch. The time is stored in a variable.
- lcd_clear()
- reactiontime=now-then
- timelist.append(reactiontime)
    - The reaction time is calculated and stored in the list.
- then=0
- makegraph(index,timelist,'question','reaction time','Performance')
    - the graph is made.
- print(correct)

Errors

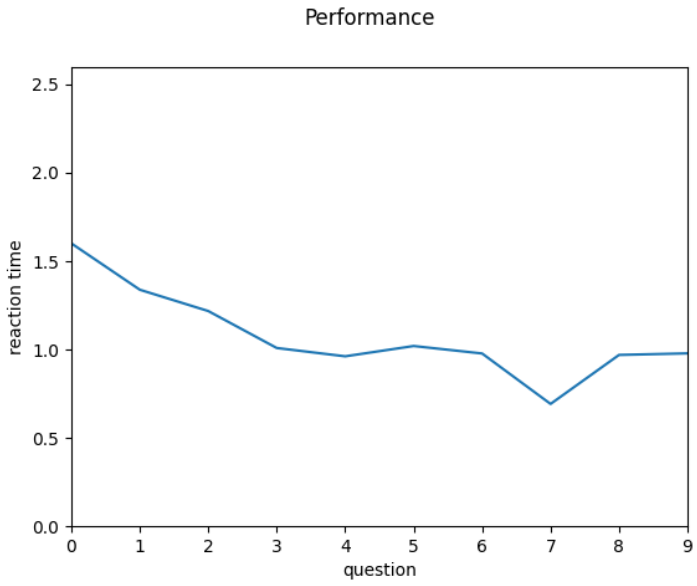
- An error occurred when I tried to implement the loop using if I had received input but did not store in a variable, this resulted in the conditional statements in the loop being ignored. I later changed it and stored the inputs in a variable as shown above. My understanding is that not storing inputs made the input change when the processor reached the conditional statements as finger had been taken away from the sensor.

Lessons learned:

- Storing input values helps the code run better as inputs can change during runtime.
- I previously wanted to implement a sound that sounds if the user answers a correct answer and a sound when he gives the wrong answer. However, later I realized that no particular sound from the buzzer sounded like a victory sound or a sound that indicates loss. Also, producing a sound for a certain duration messes with how the game of the user as if the player is fast, he may finish a question even before the sound of the previous question is finished.
- The time.time() takes time is form in seconds from epoch but subtracting the recent time from the previous one gives the difference in seconds which does not require and conversion.

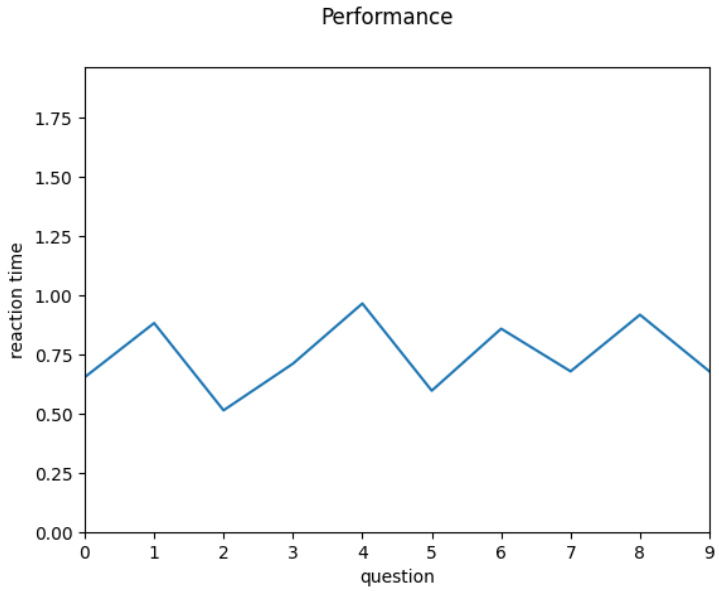
Testing

Test	Manipulation	Output Expected	Observation	Investigation
Test 1	Amount=10 Difficulty=0.5	Graph showing higher reaction time then others as more difficult. The game displays question and colors properly and also shows the number of correct answers properly	As expected,	None needed
Test 2	Amount=10 Difficulty=1	Graph showing lower reaction time then test 1 as less difficult. The game displays question and colors properly and also shows the number of correct answers properly	As expected,	None needed
Test 3	Amount=10 Difficulty=2	Graph showing lowest reaction time then other tests as least difficult. The game displays question and colors properly and also shows the number of correct answers properly	As expected,	None needed



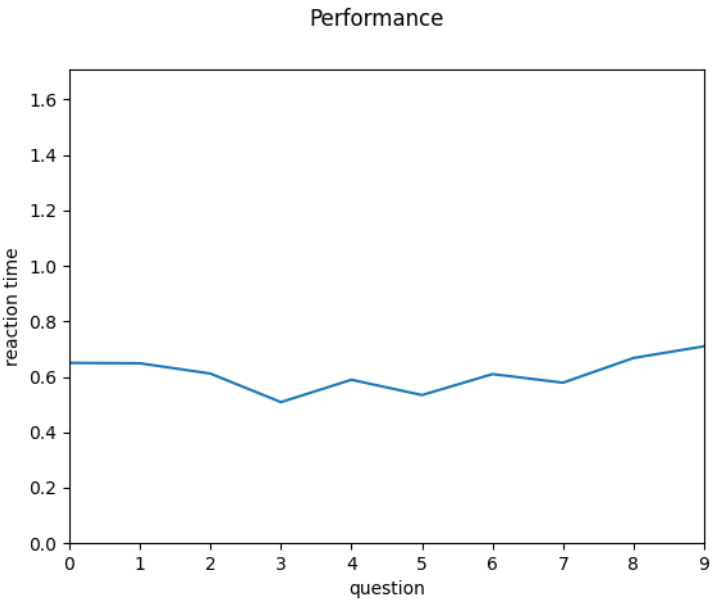

```
please enter the difficulty from 0.5, 1 and 2. The lower the number the harder it is: 0.5
please enter the number of questions you want to play for: 10
no Arduino board detected
Check Windows Device Manager to see which COM the Arduino Uno has been assigned
Please specify COM port (eg. COM10): com5
Arduino board detected
10
```

Fig: Test 1 outputs




```
please enter the difficulty from 0.5, 1 and 2. The lower the number the harder it is: 1
please enter the number of questions you want to play for: 10
no Arduino board detected
Check Windows Device Manager to see which COM the Arduino Uno has been assigned
Please specify COM port (eg. COM10): com5
Arduino board detected
10
```

Fig: Test 2 outputs




```
please enter the difficulty from 0.5, 1 and 2. The lower the number the harder it is: 2
please enter the number of questions you want to play for: 10
no Arduino board detected
Check Windows Device Manager to see which COM the Arduino Uno has been assigned
Please specify COM port (eg. COM10): com5
Arduino board detected
10
```

Fig: Test 3 outputs

Errors

- No errors were encountered as the implementation was done by following proper procedure set in the lab instructions, my familiarity with the syntax and for doing a similar task in previous lab.

Lessons learned:

- Setting difficulty to 0, will cause an error as even before proper value assigning of input the loop may complete a repeat.
- Taking input and storing them in a variable is the safest way to ensure everything runs smoothly.
- For future development implementing validation can be useful to check that input values for difficulty and amount variables are not such that the code will not run.

Reflection Question

How were each of the following used within your project? You can give example statement(s) but also explain why.

- Expression: Expressions were used multiple times throughout the program. For example, the line correct+=1 is an expression that adds one to the value in the variable correct and stores it. This expression allowed me to use the correct variable as a counter of how many times the player got the correct answer.
- Variable: Variables were used to store different types of data in the program. For example, the line button=0 was used to initialize and set the value of the variable button to zero. Variable was used as its value can change and this changed value for this particular variable was later used in conditional statements.
- If statements: If statements were used in the program. For example, the line if touch!=0 or button!=0: was used to check if an input was received or not. If statements allow us to set conditions for certain lines of code to run only when those conditions are met.
- Loops: Both For and while loops were used for example, the lines for i in range(amount): and while input==False: The for loop is used to repeat the codes in it until all questions have been answered and the while loop checks if any input has been received. Loops allow us to repeatedly use the same code without requiring us to write it over and over again.
- Lists: Lists were also used in the program. For example, timelist=[] was used to store the reaction time of the user. Lists are used as they are an effective way to store data an the built in functions of a list allows lot of functionality.
- Functions: I used a function with the declaration of makegraph(xlist,ylist,xlabel,ylabel,title): this function makes the graph at the end of the program. Functions allow larger problems to be cut into smaller bits and focus on those particular bits at a time. They also make code more reusable and editable as making a change to a function does not directly affect the whole program.
- Modules: I had to import many modules to make my program such as import random that allowed me to make random numbers within a range. Modules have many useful functions. They can be used in different programs by importing them and make code cleaner and more editable, as editing or making your own module does not affect the function of your main code directly.

Reflection Question - How can you apply lessons learned in the ENGI1020 lab activities to other parts of engineering or life in general?

The labs have taught me many useful skills. I am now confident in my ability to work with sensors and hardware. This will serve as a strong base for my pursuing of a degree in computer engineering. Learning how to make a project proposal and logbook are fundamental skills of an engineer. The labs reinforced the topics I learned in class such as flow control, loops, modules and more. The labs also taught me how to use mathematical expressions to create relations between input values and output values. I think I will now try and automate some of my daily tasks using code. I can certainly make an alarm clock using the Arduino and buzzer if I ever need. I learned that each input and output device have different types of values they use, some may need to be converted to fit our purpose. Furthermore, I have grown as a better coder due to my practice in the labs. I believe that these will be very useful in my life.