“Wack”-a-LED

ECEGR 2000-01

Physical Computing with Python

Brandon Vu and Tony Tran

12/07/2023

Contents

1. Introduction:

In this project we designed a user interactive game. The main objectives we wanted to achieve with this project were to learn more about different components as well as the integration of these components. In this project we used a Raspberry Pi, a breadboard, four LEDs, a buzzer, an LCD display module, and a one by four membrane keypad.

The interaction that we wanted to create was a two-way system where the game prompts the user to do something then the user responds with a corresponding action. With this the game we came up with pulled inspiration from Wack-a-mole hence the name of our project. The objective of the game is to wait for a random LED light up for a random time then press a corresponding button in the given time to "hit" the LED out. This is the main premise of the game. Some other things we added were the LCD to keep a score, guide to user to starting the game, and give the user encouraging comments. The buzzer was added to give an auditory

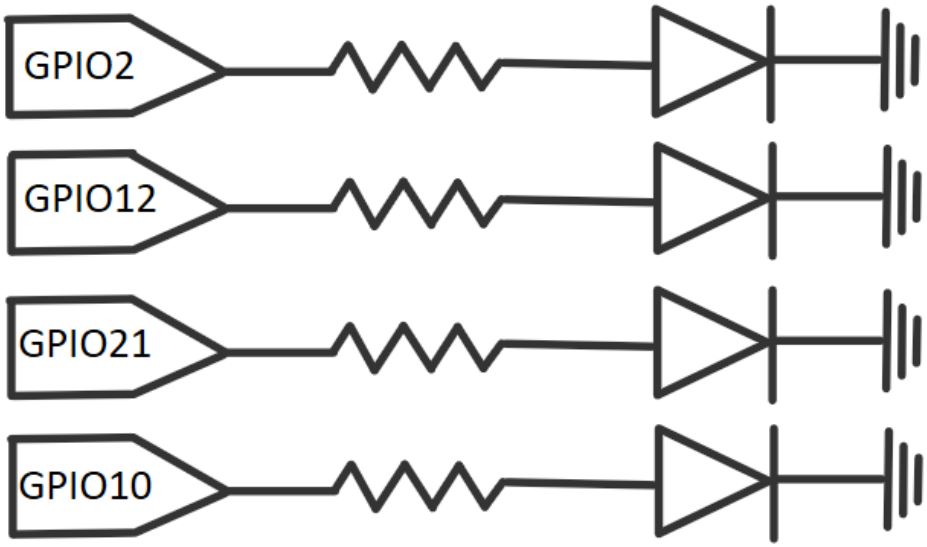
indication of failure or success. The entirety of this project was coded in python with in tandem with a Raspberry Pi.

2. Design:

The components, again, were a Raspberry Pi, LEDs, one by four membrane keypad, a buzzer, and LCD display. We split the work into two groups, one person working on the LCD and the other working on the keypad, LEDs, and buzzer. We worked separately but still collaborated helping each other understand all components. Then tested our components. While testing we learnt a lot about our pi and many of its internal features.

2.1 Hardware:

In this section we will discuss how we designed and built our components on our pi and breadboard. First the LEDs. Each LED was connected to a GPIO output pin then a resistor then the LED to ground. The resistance we used was 330 ohms. The LEDs were read top to bottom top being 1 and bottom being 4. Read left to right on the breadboard, left being 1.

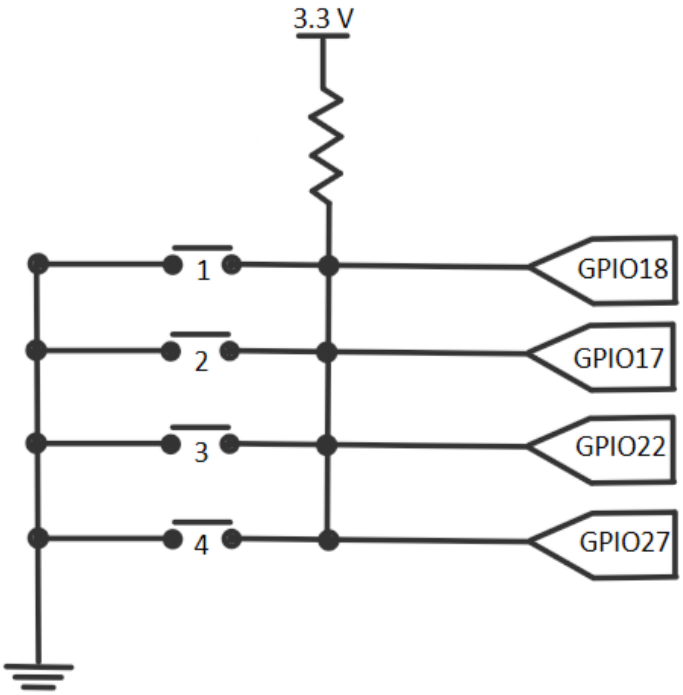


Circuit diagram LEDs.

Next is the buzzer, this was plugged into a GPIO pin then to ground.

Circuit diagram of buzzer.

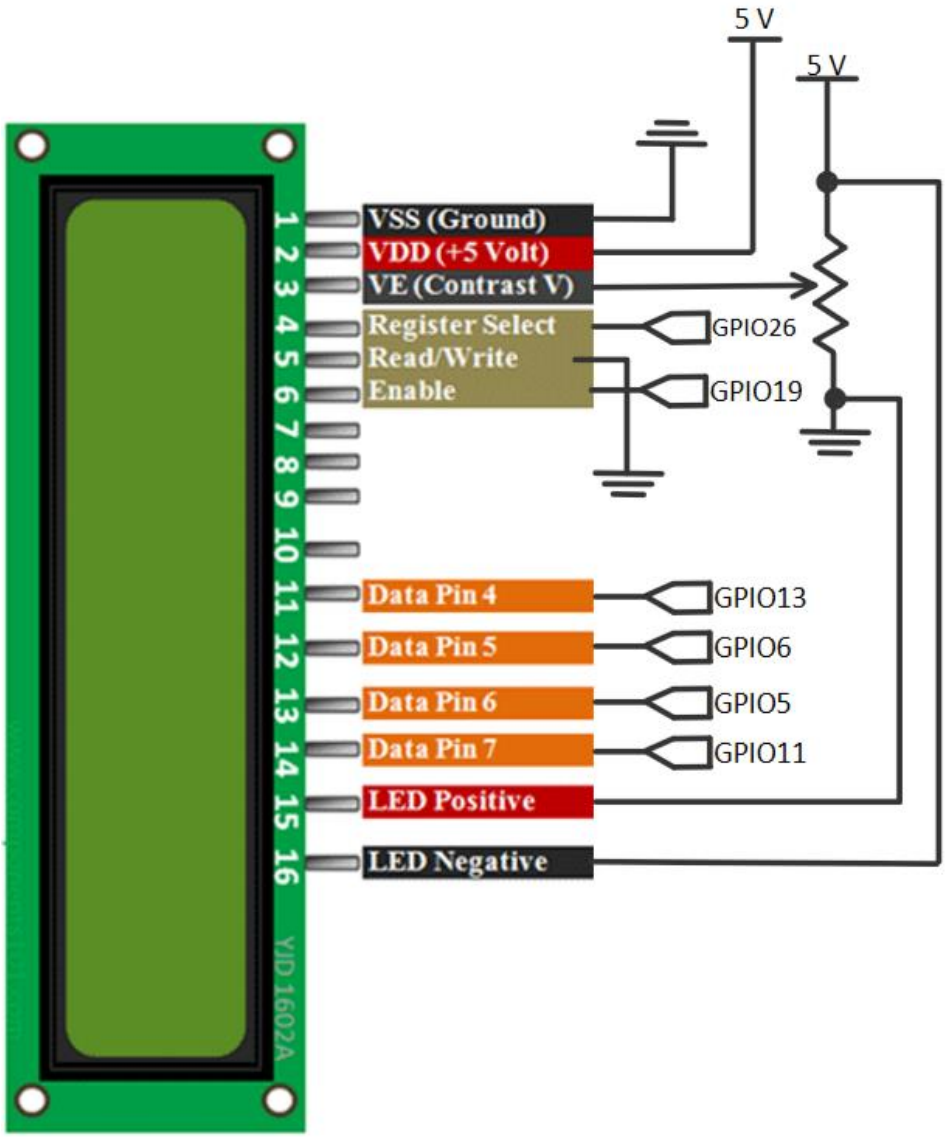
The buttons were set up in a one by four array in a pull up setup where 1 is the is the input read when the circuit is closed and 0 when the circuit is open. This was done with GPIO.PUD_UP in GPIO pin setup. In the picture on page 6 the keypad has five female connections from right to left the pin plugged into them are GND, GPIO17, GPIO18, GPIO27, and GPIO22. This is not represented in the circuit diagram.



Circuit diagram for one by four membrane keypad.

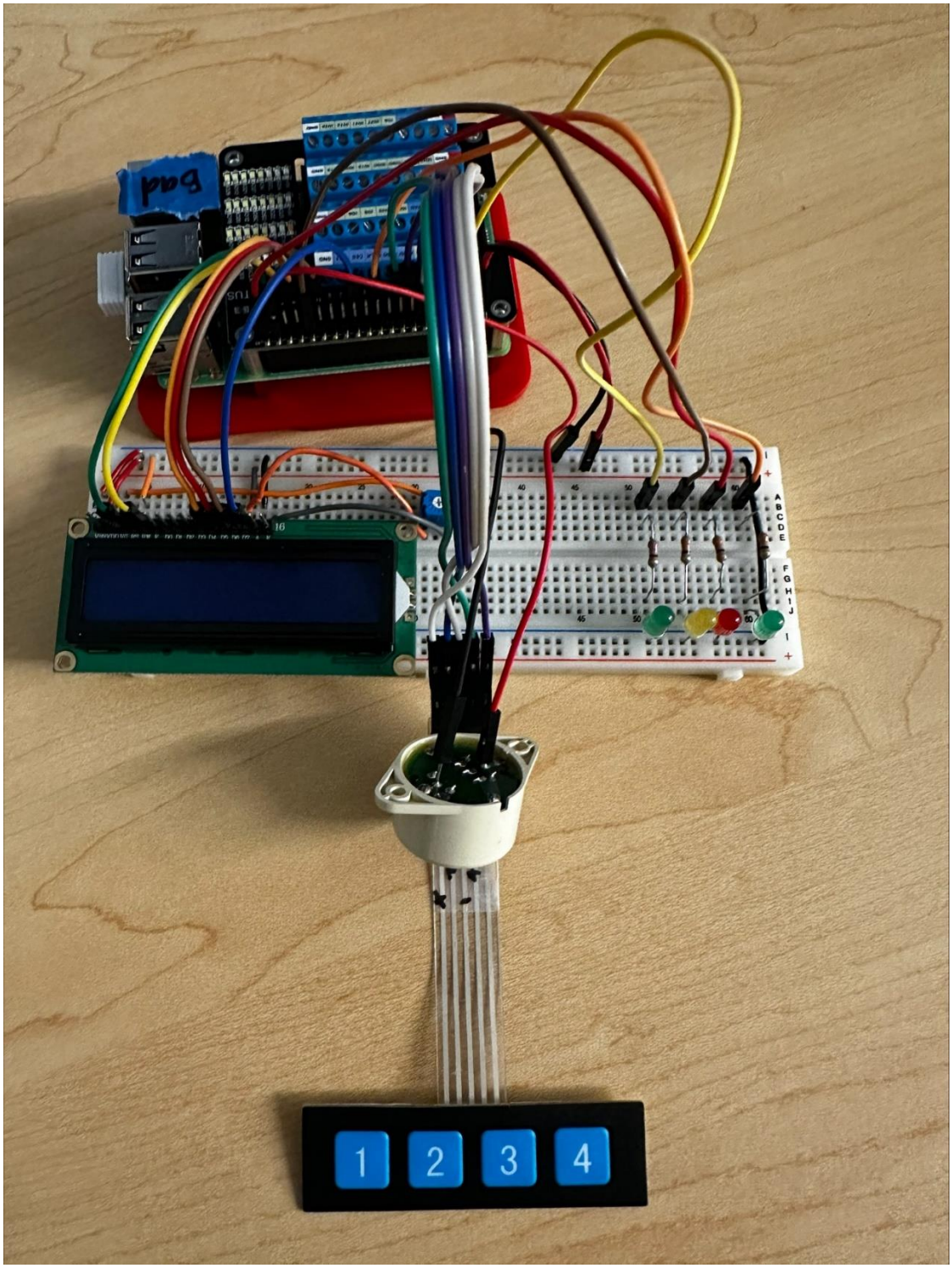
The last component we used was the LCD display. Below is a table containing what GPIO pins were used for the pin on the LCD module and where other pins on the module were connected to.

Pin 1 – GND	Pin 4 - GPIO26	Pin 11 – GPIO13	Pin 14 – GPIO14
Pin 2 – 5v	Pin 5 – GND	Pin 12 – GPIO6	Pin 15 – 5v to potentiometer
Pin 3 – potentiometer	Pin 6 – GPIO19	Pin 13 – GPIO5	Pin 16 – GND to potentiometer



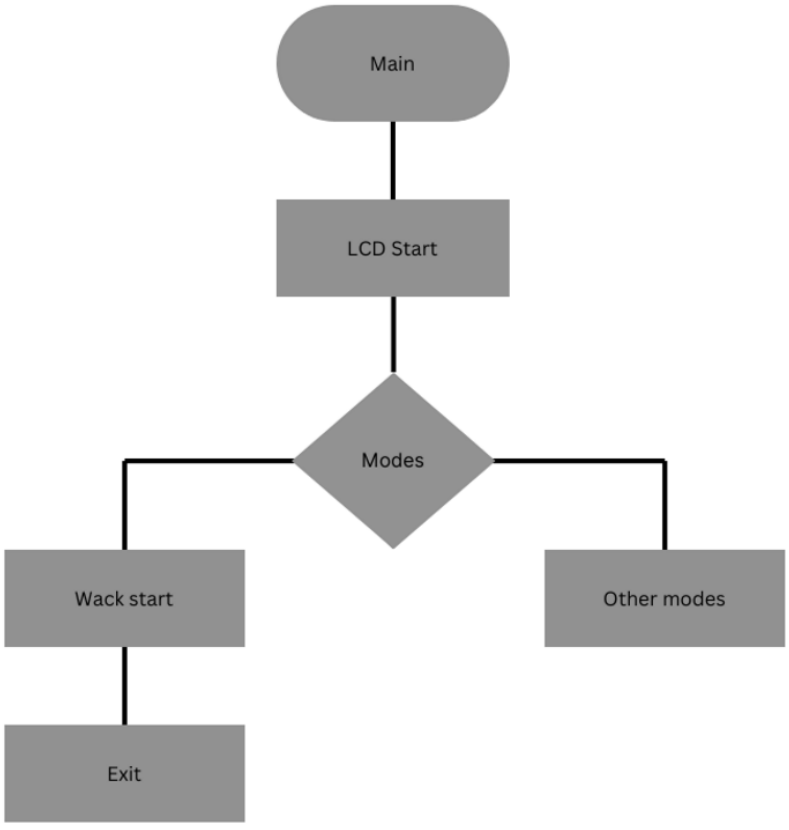
Circuit diagram of LCD display module.

The main pieces of hardware that allowed us to put this altogether was a bread board and a raspberry pi 3. All together our setup of the hardware looked like this in the picture below.



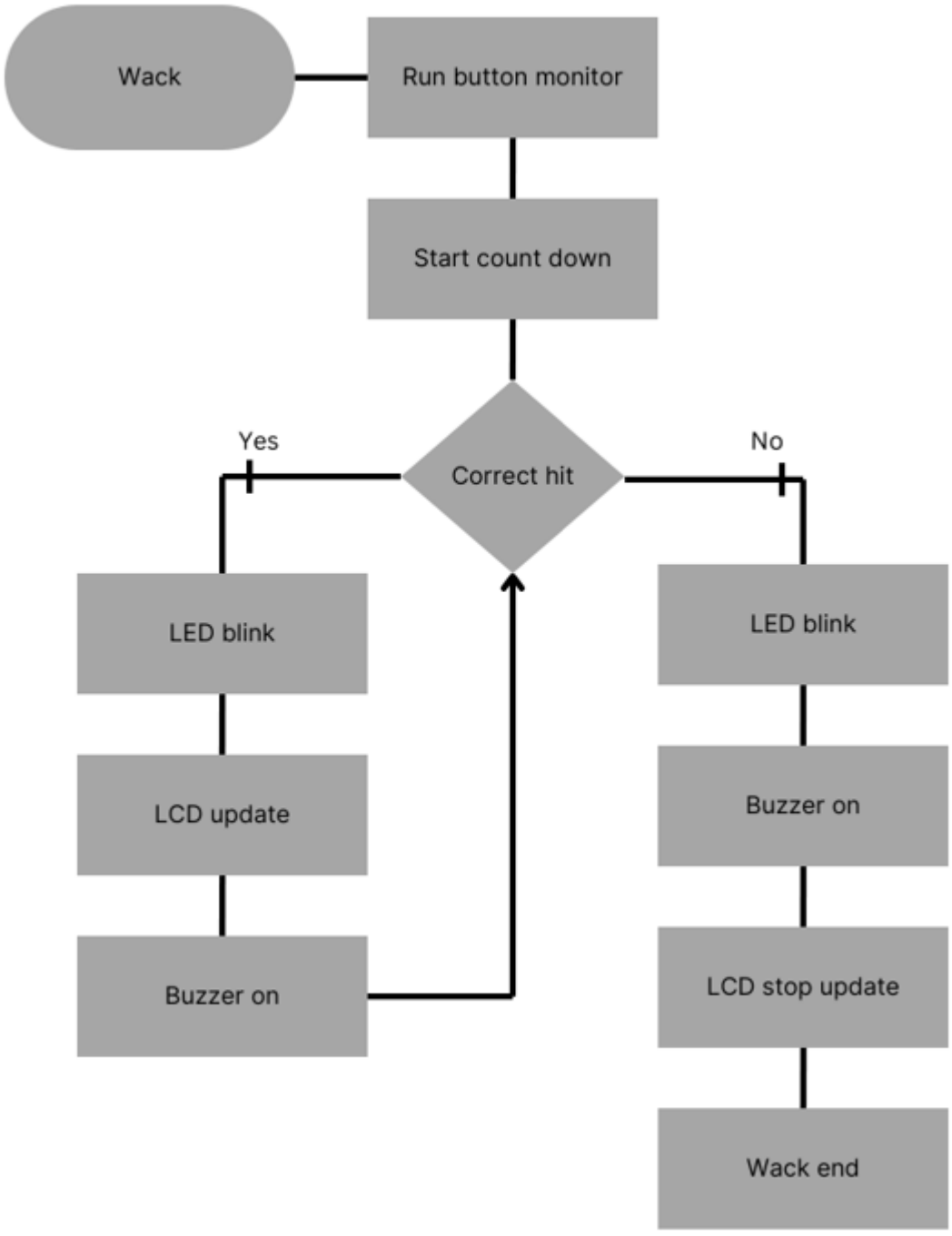
2.2 Software:

In this section we discuss what we used to integrate our components together and what we used to test some of our components. The main software we interacted with the most was WinSCP. We used this because working on the Ras Pi was tedious, and we could move and create files and directories relatively quickly. To share code with each other we used GitLab. We also used git lab to pull, push, and merge code because we were working on our own Pi but later, we would just use Brandon's pi near the end. We used notepad++ to edit code directly on the pi. Lastly, we used putty to interact with the pi. Although we were only working on one mode, four buttons meant a possible four different modes to be created so we wanted the game to be a child class of main called Wack.



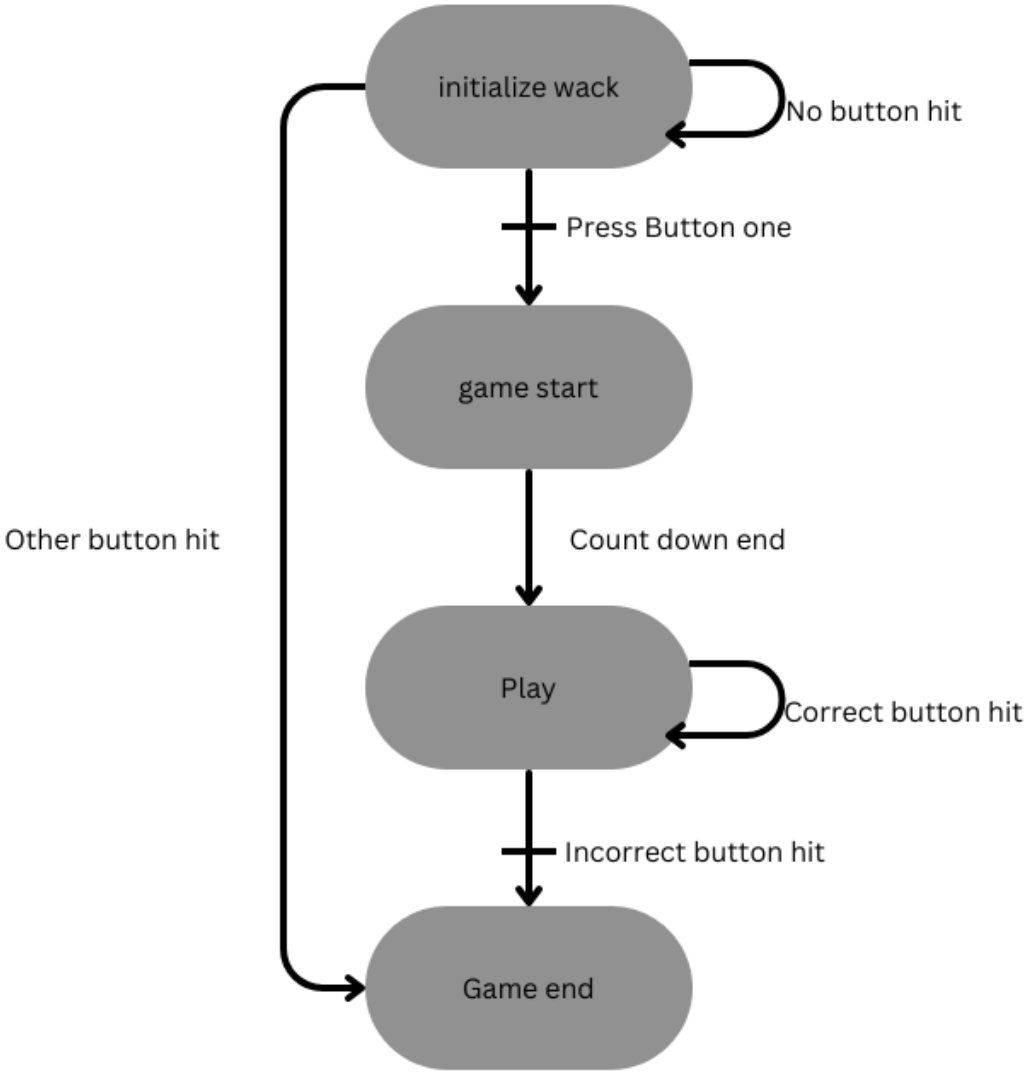
Workflow diagram of main.

Main contains the main menu or “LCD start” where the user will choose what mode they want to play choosing button one will run “Wack start” until the user fails the game. The other buttons will be other modes but at this time they exit.



Workflow diagram of Wack.

Wack is the game the project is based on when Wack starts button monitor and the three second count for the game starts. When the count ends the game starts. When correct hits are registered then the LED blink, LCD update (gives comments and score), and buzzer on (beeps for short time). When incorrect hits are registered LED blink, buzzer on, LCD stop update (indicates failure and final score), and Wack end. Once the module reaches Wack end everything ends.



State diagram of starting Wack.

Link to Gitlab code: https://gitlab.com/su-ecegr-tran/shared/game.git

3. User's Manual:

In this we will add on to hardware particularly the setup of it and a more detailed description of how to play the game and how to run the module. Referencing the picture of our hardware from page 6, the LEDs are to the right, LCD to the left, and buttons and buzzer in the middle. This is the orientation we found the most convenient for testing and playing the game. The orientation of the components can be in any order as long as they are connected correctly.

To run the game in a putty session use **python -m Game** in the directory the file is located. This will start the LCD prompting the user to choose a mode. Selecting one will start Wack other selections will exit. Selecting one will start a countdown to begin the game. Once the count ends the game starts. Pushing the button corresponding to the LED continues the game. LEDs are read from left to right, one being to the far left and four being to the far right. When the user chooses the incorrect button then the game ends. Ex. If LED one lights up press button one if the user does not push button one on time the game ends. To play again repeat the steps above.

4. References:

Eddy Ferre. Our professor of this course.

GitLab link: https://gitlab.com/su-ecegr-tran/shared/game.git

Command to run game: **python -m Game**

Links that helped with setting up the LCD:

Code for testing LCD:

https://www.raspberrypi-spy.co.uk/2012/08/16x2-lcd-module-control-with-backlight-switch/

LCD hardware information:

https://www.raspberrypi-spy.co.uk/2012/07/16x2-lcd-module-control-using-python/

LCD general setup:

https://www.circuitbasics.com/raspberry-pi-lcd-set-up-and-programming-in-python

link to hardware design app: https://www.circuito.io/app?components=8655,9443,200000

5. Proposed Future Work:

This was mentioned for a good amount of time though out this report but other modes other than Wack can be made. Other different games can be made. A pattern matching game LEDs light up in a pattern then the user copies that pattern on the button after it is shown. A song game where the user presses the buttons corresponding with the LEDs and the buzzer sounds making a beat. A quality-of-life change could be when Wack ends giving the user an option to choose a new mode or replay Wack.

6. Conclusion:

This was my first time ever working with a raspberry pi and this was very fun, more than I expected. This was also my first time making something out of electrical components other than circuits with just resistors, LEDs, and logic. This was a fun time I had, and I think it was a really good way to end this class culminating in a big project of our own ideas. This has definitely motivated me to find some other project I can do on my own.

In future project planning more will help. What I mean is learning as much as I can about the components I am going to use before even using them. Not only that but making sure all the hardware works before coding, but some code is needed to test. I found that at some points I would just sit confused because I did not know what something was doing or if it was working properly. Having more confidence with what I am working with will let me be more efficient in the future and I will not waste as much time troubleshooting.

Coding was the least fun part of this project, which is why I did not choose computer engineering. Testing with code and integrating, that part was not as difficult as I thought it would be what was, was working on the same code. Brandon and I spent a lot of time trying to figure this out because we wanted to work on the code together, but we never worked on code in class. It was always out of the lab and the lab time was used to work out the kinks of our code. Eventually we just shared code on a shared Google Drive file. We did work out GitLab though and saw how good of a resource it can be. I think I can use it more when I am more confident with how it works.