Front Page

Mîț Alexia Teodora

Bachelor's Computer Science Program

2024-2025 Academic Year

NHL Stenden University of Applied Sciences

Personal email: alexia2207@yahoo.com

School email: alexia.mit@student.nhlstenden.com

Number of EC's: 0/180

Table of Contents

Front Page 1

Table of Contents 1

Introduction 4

Week 1 5

- Computer Science 5
  - Smart Cafe Helper 5
  - Tip Calculator + Video 10
  - Tip Calculator peer feedback 9
- Embedded Systems 12
  - Blink with external reset 12
  - Mario Pico 15
- Self-reflection 16

Which hat am I? 17
Week 2 19
Computer Science 19
Vanity Plates 19
Vanity Plates goes Flask 23
"How can I use Python to create a program that..." 28
Embedded Systems 29
7-segments voltmeter (0... 10V) 29
Self-reflection 34
Week 3 36
Computer Science 36
Group mindmap 36
My mindmap 37
Survival Simulator 38
Embedded Systems 52
Realtime clock and temperature on LCD-display 52
Self-reflection 57
Week 4 59
Computer Science 59
Just setting up my twttr 59
Testing my twttr 61
Back to the bank 62
Re-requesting a Vanity Plate 66
Nutrition Facts 68
Refueling 72
Convert Celsius to Fahrenheit 76
Check if a number is even or odd 77
Make a factorial calculator 78
Reverse a string 79
Prime number checker 80

Embedded Systems 81
Dino cheater (HID) 81
Analog joystick (HID) 82
Week 5 89
Computer Science 89
Scavenger hunt 89
CS50 Shirt 92
Embedded Systems 95
Wi-Fi scanner 95
Controlling stuff (Webserver) 96
Week 6 103
Computer Science 103
Scavenger hunt Pt II 103
Week 7 108
Computer Science 108
Product goes OOP 108
CS50 Shirtificate 110
Week 6 and 7 113
Embedded Systems 113
Attendance (RFID, RTC, web and LCD-display) 113

Introduction

My name is Mîț Alexia Teodora, and I am 19 years old. I moved from Romania to The Netherlands by myself so I can follow the Computer Science program at NHL Stenden University of Applied Sciences. I chose to come here, because it has a different way of teaching than the Universities from my country. In my opinion, the decision that I made will give me more opportunities to pursue my future career.

This portfolio will show in detail my progress and the skills I gained while following the chosen program.

In addition, my GitHub name is Alexia220700.

Week 1

Computer Science

Smart Cafe Helper

Smart Café Helper

Create a program that assists customers in a café.

Functionality:

1. **Order Handling:**
   Customers type in their orders, and the program responds appropriately regardless of how the order is entered. For example:
   - Input: COFFFF
   - Output: Okay, I will prepare coffee.
2. **Welcome Message:**
   When the program starts, it plays a customizable welcome message. For instance, the default message might be:
   - Welcome to Smart Café!
     The program should allow the user to transform this message by inserting ... between every character, producing:
   - W...e...l...c...o...m...e... t...o... S...m...a...r...t... C...a...f...e...!
3. **Emojis for Keywords:**
   The helper detects certain keywords in the customer's order and responds with corresponding emojis:
   - coffee → ☕
   - tea → 🍵
   - cake → 🍰
     Add two more examples of keywords and their corresponding emojis.
4. **Energy Calculation:**
   The program calculates energy using the formula ($E = mc^2$), based on the weight (in grams) of the food or drink. The customer inputs the weight, and the helper calculates the energy accordingly.
5. **Order Total with Tip:**
   The program calculates the total price of the customer's order, including a tip. The customer selects a tip percentage (e.g., 10%, 15%, 20%), and the program provides the final amount.

Requirements:

1. **Use of Functions:**
   Each feature must be implemented in a separate function for modularity.
2. **Loops and Conditionals:**
   Ensure the program can handle multiple orders in one session. Customers should be able to place orders repeatedly until they choose to exit.

This is the code for the assignment:

```
# define the main function of the code
def main():
    # first part
    # welcoming the customer and asking if they want to customize the text
    # or not
    print("Welcome to our Cafe!")
    string = ("Welcome to our Cafe!")
    answer1 = input("Would you like to customize this message? Answer with yes or no. ")

    # customizing the string if customer asks to do it
    if(answer1.lower() == "yes"):
        # checking if it's a character, not space etc
        for char in string:
            if char.isalpha():
                # end used to remove the newline at the end of each ...
                print(char, "...", end="");
            # i might not need this
            else:
                print(char, end="");



    # second part
    # asking customer for order
    order1 = input(" What would you like to order? ")

    # call writeOrder and get the price
    totalAmount = writeOrder(order1)

    # ask if the customer wants to order more items
    totalAmount += moreThanOneOrder()  # add more orders to the total

    # ask for tip percentage and calculate the total amount with tip
    totalPriceWithTip(totalAmount)

# making a function for taking the order more easily
# order1 is the argument of this function, bc it's defined in another function (the main
function)
def writeOrder(order1):
    # make the letters lowercase to check the order
    # assigned the input string to a new variable, because I made it lowercase
    order = order1.lower();
```

```
    # matching the order to the appropriate emoji
    if(order =="coffee"):
        emoji = '☕';
    elif(order == "tea"):
        emoji ='🫖';
    elif(order == "cake"):
        emoji ='🍰';
    elif(order == "croissant"):
         emoji ='🥐';
    elif(order == "orange juice"):
         emoji ='🍊';
    else:
        print("Sorry, we don't have that item.");
        # exit the function with 0 if the item isn't recognized
        return 0

    # showing the order the customer made
    print("Okay, I will make " +order +emoji+ " for you. ")
    # I can rewrite it using (f"Okay, I will make {order} {emoji} for you.")

    # third part
    # calculating the energy level
    answer2 = input("Would you like to know the energy level of your order? ")

    if(answer2.lower() == "yes"):
        # convert string into an integer for the calculation to work
        weight = int(input("Input the weight, in grams, of the order you made. "))

        # in Python, ** used for exponentiation, instead of ^
        # Energy = mass * speed_of_light^2
        energyLevel = weight * (3 * 10**8)**2

        # rounding the number to three digits
# FIX THIS
        print(f"The energy level of your order is {round(energyLevel, 3)} Joules.")

    # fourth part
    # order total with tip

    # setting different prices for each drink or pastry
    if(order=="coffee"):
        # no need to put $ in Python
```

```
        price = 3;
    elif(order== "tea"):
        price = 2;
    elif(order== "cake"):
        price = 5;
    elif(order== "croissant"):
         price = 6;
    elif(order== "orange juice"):
         price = 8;

    # show the price of the specified item
    print("This item costs ", end="")
    print(price, end="")
    print("$.")
    # or change it with print(f"This item costs {price}$.")

    # return the price of the current order
    return price


def moreThanOneOrder():
    # check if customer wants to order something else or not
    totalAmount = 0

    # while customers responds with yes, the loop repeats
    while True:
        order2 = input("Would you like something else? ")
        if order2.lower() == "yes":
            # ask for new input
            order1 = input("What would you like to order? ")

            # calculate total cost
            # add the price of the new order
            totalAmount += writeOrder(order1)
        else:
            # exit the loop if the customer doesn't want anything else
            break

    # return the total of all additional orders
    return totalAmount
```

```python
# fifth part
# ask for tip percentage and output the total amount for the customer

def totalPriceWithTip(totalAmount):

    # tip percentage
    tipPercentage = int(input("Choose a tip percentage: 10, 15 or 20. "))

    tip = (tipPercentage / 100) * totalAmount

    # add tip to the total amount
    finalPrice = totalAmount + tip

    print("Your total is: ")
    print(finalPrice, end="")
    print("$")
    print("Thank you for visiting our Cafe!")

main()
```

Tip Calculator + Video

Well, we've written most of a tip calculator for you. Unfortunately, we didn't have time to implement two functions:

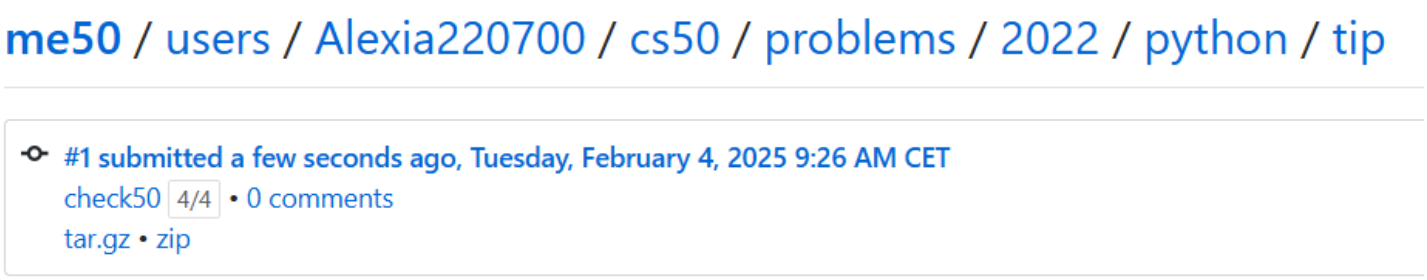
dollars_to_float, which should accept a str as input (formatted as $##.##, wherein each # is a decimal digit), remove the leading $, and return the amount as a float. For instance, given $50.00 as input, it should return 50.0.

percent_to_float, which should accept a str as input (formatted as ##%, wherein each # is a decimal digit), remove the trailing %, and return the percentage as a float. For instance, given 15% as input, it should return 0.15.

This is the code for the assignment:

```
def main():
    dollars = dollars_to_float(input("How much was the meal? "))
    percent = percent_to_float(input("What percentage would you like to tip? "))
    tip = dollars * percent
    print(f"Leave ${tip:.2f}")


def dollars_to_float(d):
    # getting rid of the dollar sign
    d  = d.replace("$", "")
    # returning a float value
    return float(d)

def percent_to_float(p):
    #getting rid of the percentage sign
    p = p.replace("%", "")
    # returning a float value without the last two digits
    p = float(p) / 100
    return float(p)

main()
```

Check50 score and link:



https://submit.cs50.io/check50/acfe5d1ace301275fe5835ccf46c0a7a5cda9116

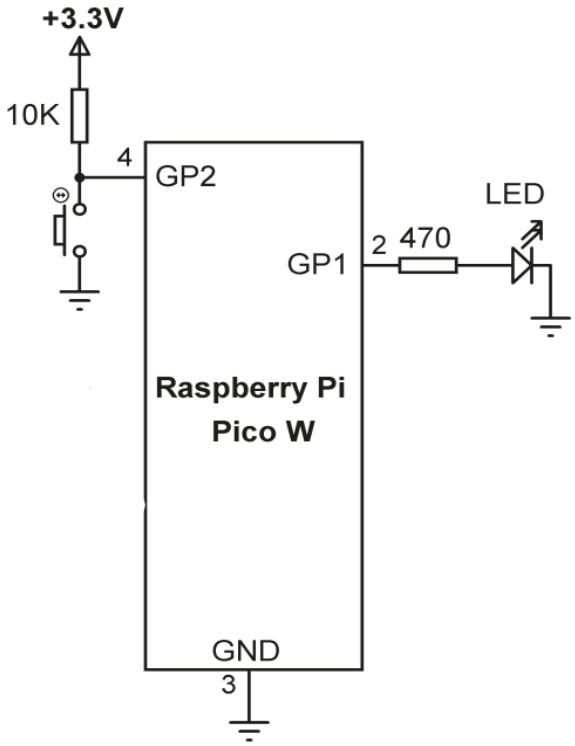
Embedded Systems

Blink with external reset

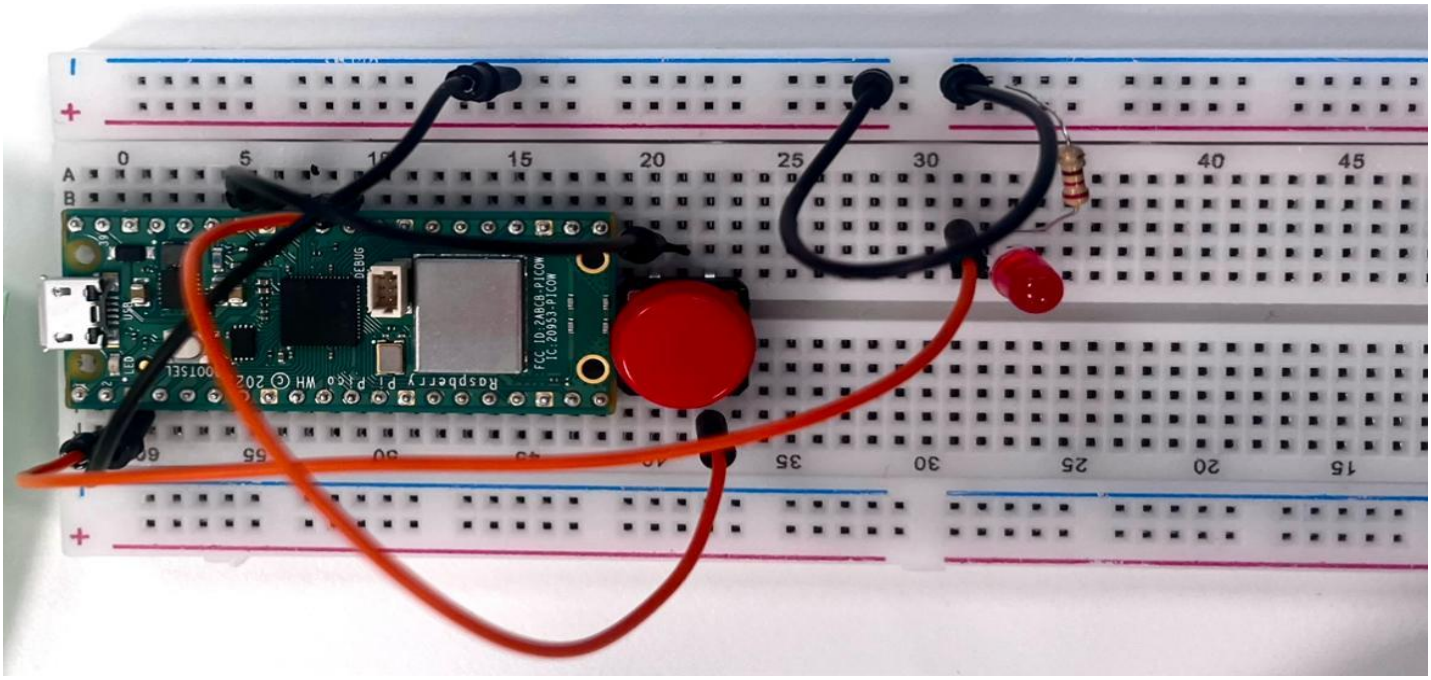
The program had to make the Raspberry Pi Pico's LED blink ON and OFF. Besides, it needed a reset button.

```
from machine import Pin
import utime

# pins for led and button
LED = Pin(1, Pin.OUT)  # LED at GP1

# variable to control the blinking
blink_enabled = True

# loop
while True:
    # control the LED based on the blinking state
    if blink_enabled:
        LED.value(1)  # LED ON
        utime.sleep(1)  # Wait 1 second
        LED.value(0)  # LED OFF
        utime.sleep(1)  # Wait 1 second
    else:
        LED.value(0)  # ensure LED is off when blinking is disabled
```

Schematic:



My project:



Mario Pico

The project had to create a pyramid like the ones in Mario games based on the input number.

```
# ask for correct input height
height = int(input("Choose a height between 1-8: "))

while (height < 1 or height > 8):
    height = int(input("Choose a height between 1-8: "))

i = 0
j = 0

# i goes through rows
for i in range(height):
    # j goes through collumns
    for j in range(height - i - 1):
        # prevents the newline and instead specify a different string to be appended at the
end of the output
        # used to print multiple items on the same line
        print(" ", end="")
```

Example:

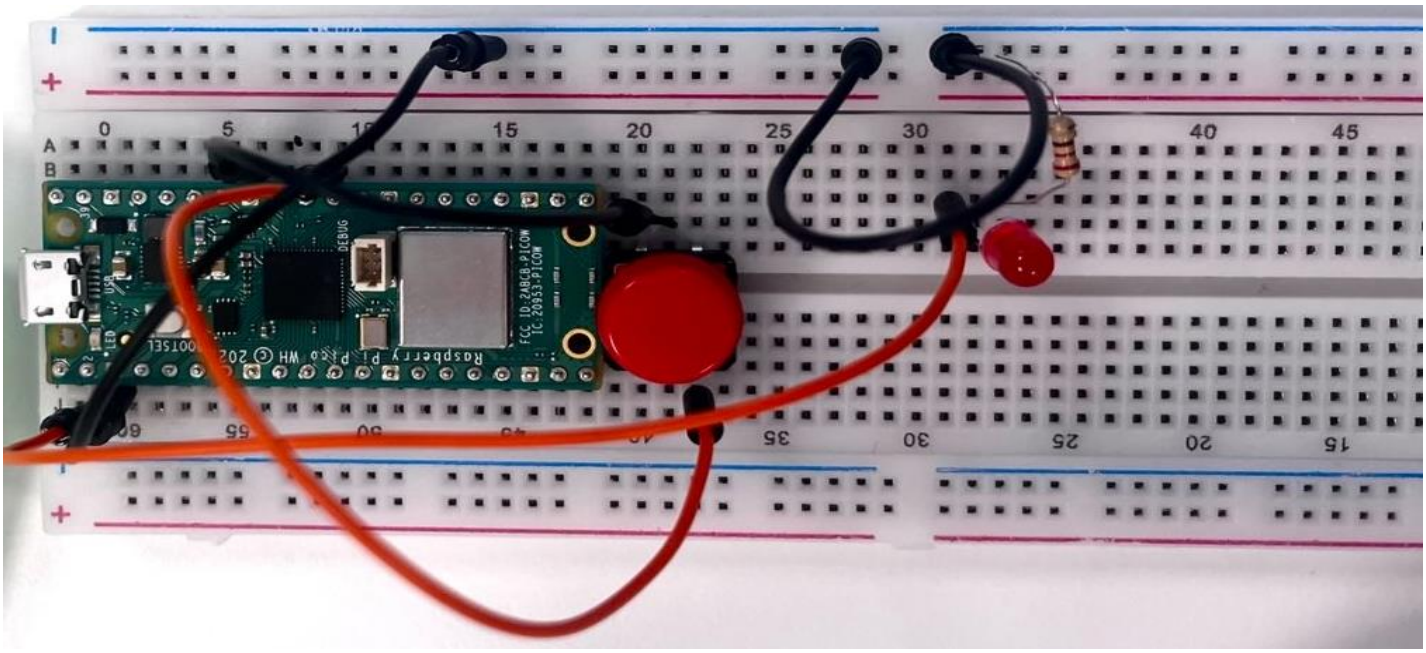
Self-reflection

In the first week of Period 3 we started again with Python. For Computer Science we had two easy assignments. The first one was a Tip Calculator, that shows customers how much they have to pay based on their order. The second assignment was the Smart Cafe Helper. I really liked this project, firstly I made some notes about what I had to do and what I wanted to add to my program.

For Embedded Systems we started working with the Raspberry Pi Pico. I installed Thonny on my laptop, as the book said, and started programming with Micro Python. The first assignment was making an LED Blink and using a button to reset it. The second project was Mario More, that took me 5 minutes to make. I already had the code for this in Python, so I only had to change three lines of code to make it work properly on Thonny.

In plus, we created the group for our next team project. Our team is called Armada. Each letter represents a person from our group, Andrei, Raffael, Matin, Adriana, Diogo, Alexia. During this week, we struggled to find a unique idea to create.

This is my Blinking LED project:



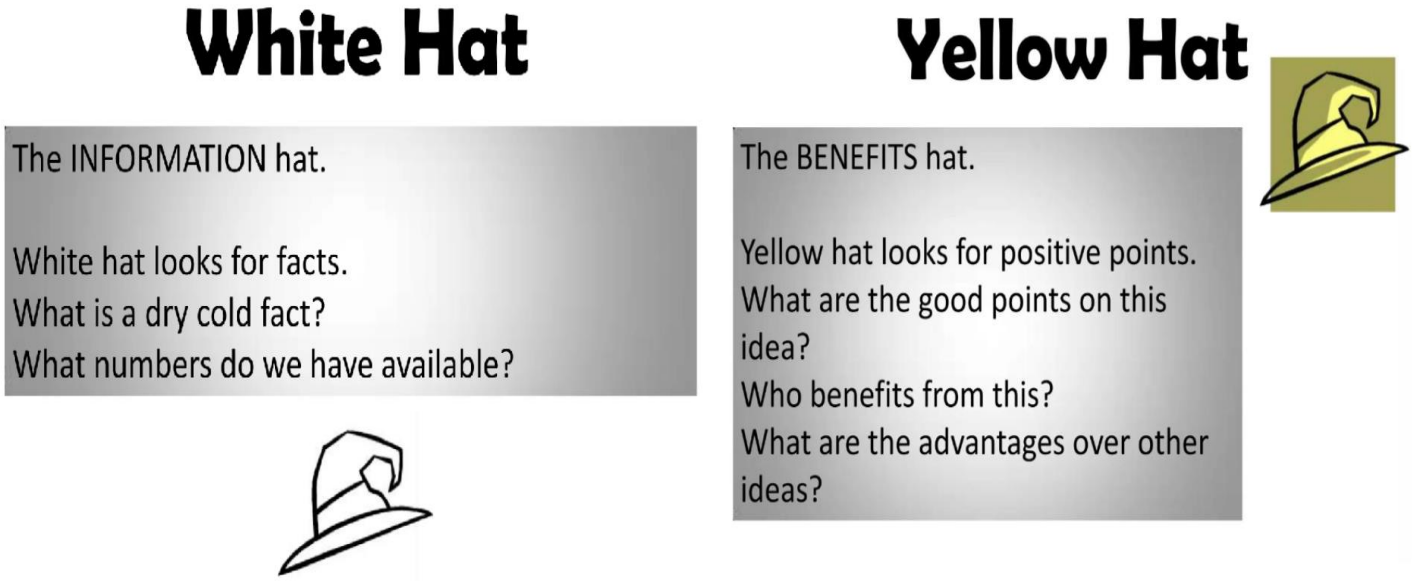
Which hat am I?

In my opinion, I am a combination of the Yellow Hat and the White Hat.

Firstly, the White Hat represents objective thinking, focused on gathering data and facts. The Information Hat reflects my ability to think objectively.

Secondly, the Yellow Hat, also known as the Benefits Hat, represents optimism and positive thinking. When you "wear" the Yellow Hat, you focus on the benefits, value, and possibilities of an idea or situation.

As a conclusion, I am mostly an Information Hat, thinking in a logical way, but I feel like a small part of me is a Yellow Hat, trying to be positive about different situations and see the bright side in our project. Besides, in my opinion, I am also a green hat, especially for my team, since I am creative.



Timetable

Monday	I made the Tip Calculator.
Tuesday	I started working on the Smart Cafe Helper. I made the introduction and the first writeOrder function.
Wednesday	I added the prices and emojis for the orders. I also added a function in case the customer wants to order more than one item.
Thursday	I implemented tip calculator into my code.
Friday	I made Mario Pico, which was fast since I didn't have to change many things.
Saturday	I made Blink with external reset.
Sunday	

Week 2

Computer Science

Vanity Plates

- “All vanity plates must start with at least two letters.”
- “... vanity plates may contain a maximum of 6 characters (letters or numbers) and a minimum of 2 characters.”
- “Numbers cannot be used in the middle of a plate; they must come at the end. For example, AAA222 would be an acceptable ... vanity plate; AAA22A would not be acceptable. The first number used cannot be a ‘0’.”
- “No periods, spaces, or punctuation marks are allowed.”

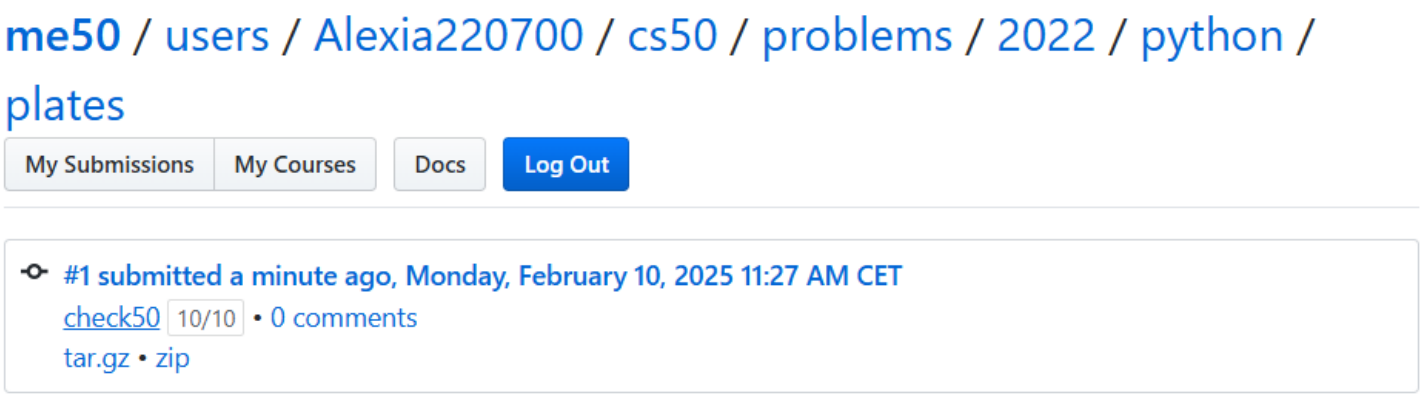
This is the code for the assignment:

```
def main():
    # prompt user for input car plate
    plate = input("Plate: ")
    # check if the car plate is valid or not and return the correct output
    if is_valid(plate):
        print("Valid")
    else:
        print("Invalid")


def is_valid(s):

    # check if the car plate has the correct number of characters
    if len(s) < 2 or len(s) > 6:
        return False

    # use a for loop to check the first 2 characters
    for char in s:
        # check if the first character is a letter and uppercase
        # s is str in Python
        if not(s[0].isalpha() and s[0].isupper()):
            return False
        # check if the second character is a letter and uppercase
        if not(s[1].isalpha() and s[1].isupper()):
            return False
```

```
# check if all characters are letters
nrLetters = 0
nrCharacters = 0

for i, char in enumerate(s):
    nrCharacters += 1
    if (char.isalpha()):
        nrLetters += 1
    # if there is stg different than letters, stop the program
    else:
        break
# if the letters are equal to characters, valid license plate
if(nrCharacters == nrLetters):
    return True

# check if the numbers come at the end of the license plate
# enumerate() function used for looping over a sequence
for i, char in enumerate(s):
    # check if character is a number
    if(s[i].isnumeric()):
        # aux will remember the index of the first number in the license plate
        aux = i
        # check if the first number is 0
        if (s[aux] == '0'):
            return False

        # break so it doesn't continue to check the characters
        break


# enumerate() function used for looping over a sequence
# enumerate(s[aux + 1:]) used to start loop from aux + 1
# can be written also like for i in range(aux + 1, len(s))
for i, char in enumerate(s[aux + 1:]):
    # if the characters after the first number are letters, return false
    if not(char.isnumeric()):
        return False

# if all checks are passed
return True
```

```
main()
```

This is the screenshot from CS50 and the link:



https://submit.cs50.io/check50/c583571356ac6c913cb89b43023bf56144c486bd

Vanity Plates goes Flask

Add Flask to Vanity Plates assignment.

This is the code for the assignment:

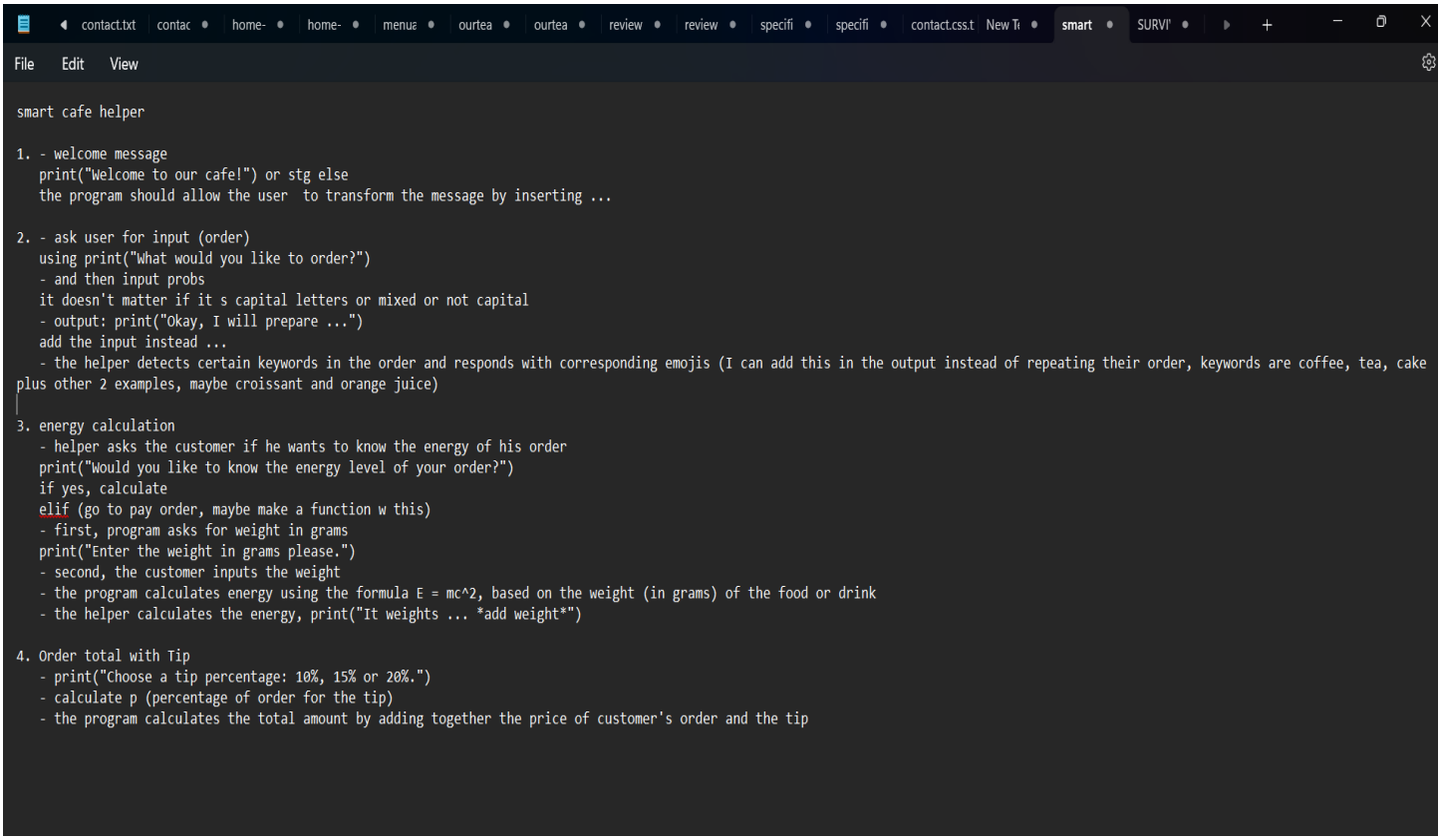
```
from flask import Flask, request, render_template_string

# initialize flask app
app = Flask(__name__)

# function to validate car license plate
def is_valid(s):
    # Check if the car plate has the correct number of characters
    if len(s) < 2 or len(s) > 6:
        return False

    # check if the first two characters are letters and uppercase
    if not (s[0].isalpha() and s[0].isupper()):
        return False
    if not (s[1].isalpha() and s[1].isupper()):
        return False

    # check if all characters are letters
    nrLetters = 0
    nrCharacters = 0

    # iterate through each character in the string
    for i, char in enumerate(s):
        nrCharacters += 1
        if char.isalpha():
            nrLetters += 1
        else:
            break

    # if all characters are letters, it's a valid license plate
    if nrCharacters == nrLetters:
        return True

    # check if the numbers come at the end of the license plate
    for i, char in enumerate(s):
        if s[i].isnumeric():
```

```
            # store the index of the first numeric character
            aux = i
            # if the first number is '0', it's invalid
            if s[aux] == '0':
                return False
            break

    # check if the characters after the first number are all numbers
    # starting from the first character after the first number
    for i, char in enumerate(s[aux + 1:]):
        if not char.isnumeric():
            return False

    # If all checks are passed
    return True



# HTML template for the form and result display
HTML_TEMPLATE = '''
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <!-- ensure proper rendering and touch zooming on mobile devices -->
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Car License Plate Validator</title>
</head>

<body>
    <h1>Car Plate Validator</h1>

    <!-- form to input the car license plate -->
    <!-- using the POST method to send data to the server -->
    <form method="POST">

        <!-- Label for the input field -->
        <label for="plate">Enter car plate:</label>

        <!-- the "id" attribute links the input to the label -->
        <!-- the "name" attribute is used to identify the input data when the form is submitted
-->
```

```
        <!-- the "required" attribute ensures the user cannot submit the form without filling
this field -->
        <input type="text" id="plate" name="plate" required>

        <button type="submit">Validate plate</button>
    </form>

    <!-- display the validation result if available -->
    <!-- tag used for control structures like loops, conditionals -->
    <!-- Python, None is a special constant that represents the absence of a value or a null
value -->
    {% if result is not none %}



        <p>Result: {{ result }}</p>
    {% endif %}
</body>

</html>
'''


# route for the main page, handles both GET and POST requests
@app.route("/", methods=["GET", "POST"])
def index():
    # initialize the result as None
    result = None
    if request.method == "POST":

        # get the car plate from the form input
        plate = request.form.get("plate", "").strip()

        # validate the car plate using the is_valid function
        if is_valid(plate):
            result = "Valid"
        else:
            result = "Invalid"

    # render the HTML template with the result
    return render_template_string(HTML_TEMPLATE, result=result)

# run the Flask application
if __name__ == "__main__":
    app.run(debug=True)
```

[redacted]

These are the notes I made in Notepad before starting to work on the assignment:



"How can I use Python to create a program that..."

Describe a problem definition that can be solved by making a program in Python.

This is my idea:

I always had issues with my lack of sleep, especially when I get overwhelmed with all the assignments we get for University, so I decided to do something about it. A good solution, in my opinion, is a sleep tracking website. It monitors how many hours of sleep you get, the time you go to sleep or wake up, the quality of it and offers a lot of useful advice.

Embedded Systems

7-segments voltmeter (0... 10V)

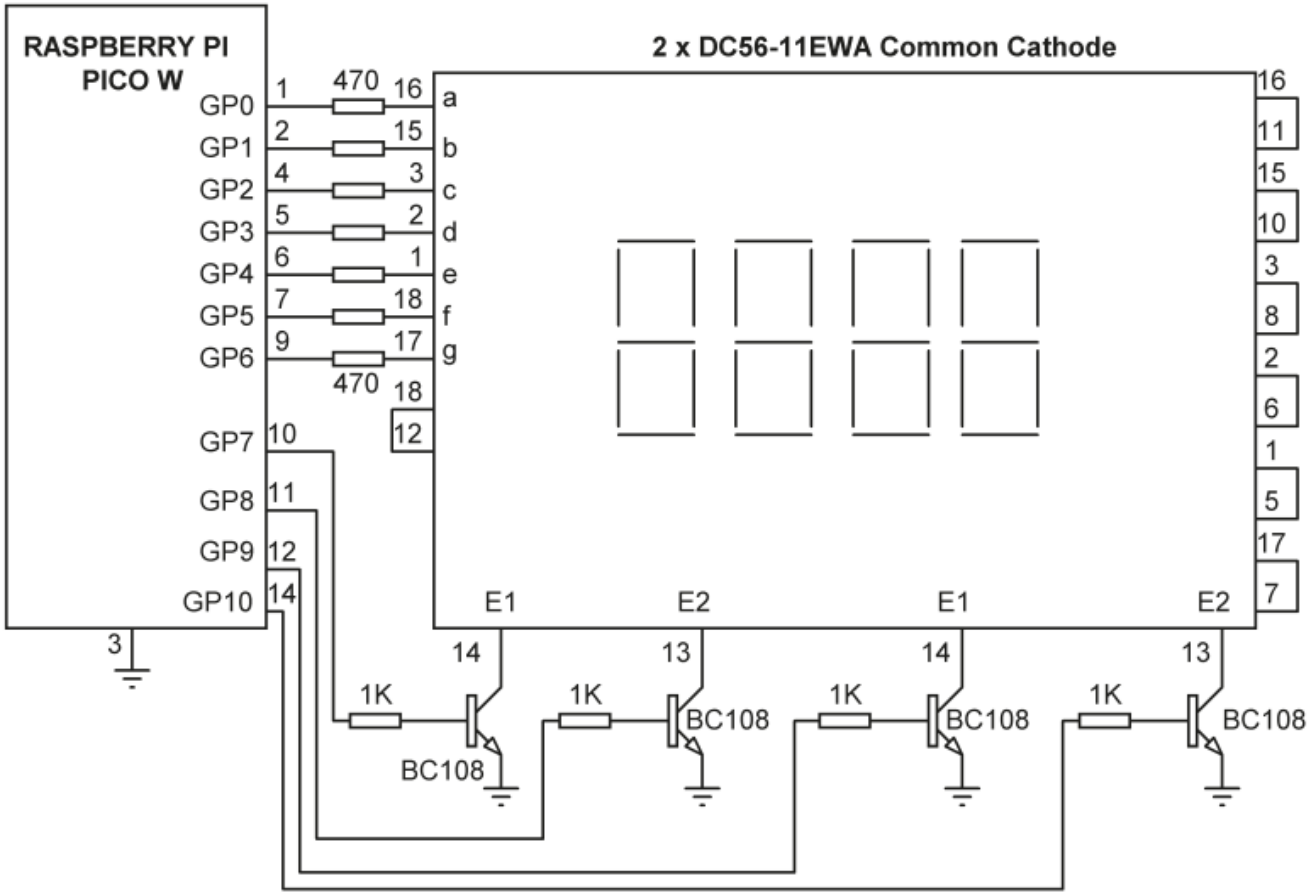
This project had to measure the voltage and show it on the 4-digit 7-segment display. The digits were displayed one after another, not all of them at the same time.

```
from machine import Pin, ADC
import time

# Define segment pins (A-G)
LED = [
    Pin(0, Pin.OUT),  # A
    Pin(1, Pin.OUT),  # B
    Pin(2, Pin.OUT),  # C
    Pin(3, Pin.OUT),  # D
    Pin(4, Pin.OUT),  # E
    Pin(5, Pin.OUT),  # F
    Pin(6, Pin.OUT)   # G
]

# Define digit pins (DIG1-DIG4)
DIGITS = [
    Pin(10, Pin.OUT),  # DIG1 (leftmost)
    Pin(11, Pin.OUT),  # DIG2
    Pin(12, Pin.OUT),  # DIG3
    Pin(13, Pin.OUT)   # DIG4 (rightmost)
]

# Define decimal point pin (shared across all digits)
DECIMAL_POINT = Pin(14, Pin.OUT)

# Segment patterns for 0-9 (A-G)
segment_states = [
    [1, 1, 1, 1, 1, 1, 0],  # 0
    [0, 1, 1, 0, 0, 0, 0],  # 1
    [1, 1, 0, 1, 1, 0, 1],  # 2
    [1, 1, 1, 1, 0, 0, 1],  # 3
    [0, 1, 1, 0, 0, 1, 1],  # 4
    [1, 0, 1, 1, 0, 1, 1],  # 5
    [1, 0, 1, 1, 1, 1, 1],  # 6
    [1, 1, 1, 0, 0, 0, 0],  # 7
    [1, 1, 1, 1, 1, 1, 1],  # 8
    [1, 1, 1, 1, 0, 1, 1]   # 9
```

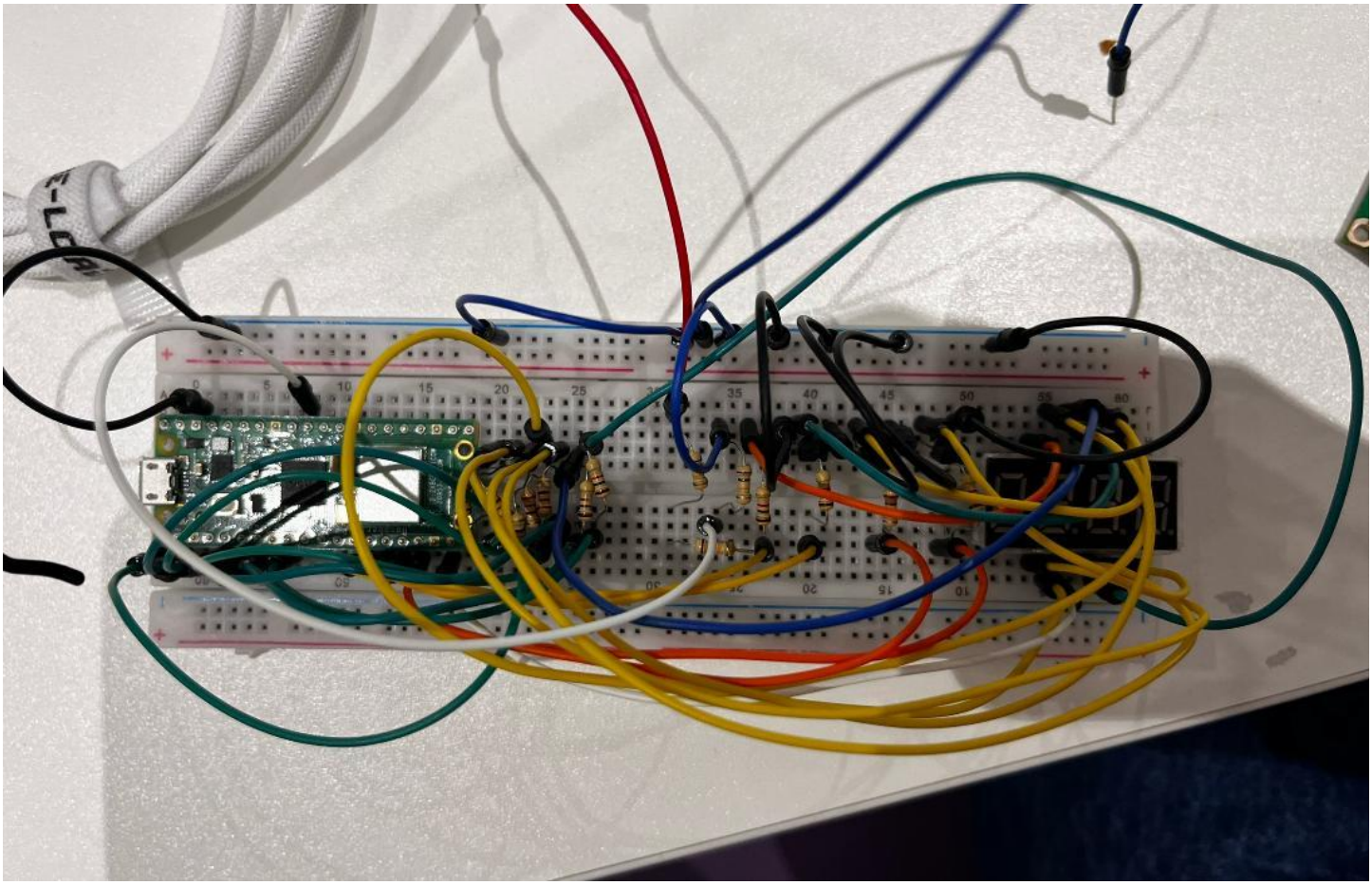
```
]

# Initialize ADC for voltage measurement
adc = ADC(Pin(26))  # Use GP26 for ADC0

def display_digit(digit, pattern, decimal=False):
    # Display a single digit with optional decimal point
    # First turn off all digits
    for d in DIGITS:
        d.value(0)

    # Set segments
    for i in range(7):
        LED[i].value(pattern[i])

    # Set decimal point (only for DIG1)
    DECIMAL_POINT.value(decimal and digit == 0)  # DIG1 is index 0

    # Turn on the specific digit
    DIGITS[digit].value(1)

def display_voltage(voltage_mv):
    # Display a voltage value with all digits visible through multiplexing
    voltage_v = voltage_mv / 1000  # Convert millivolts to volts
    digits = [
        int(voltage_v),              # DIG1 (leftmost)
        int((voltage_v * 10) % 10),  # DIG2
        int((voltage_v * 100) % 10), # DIG3
        int((voltage_v * 1000) % 10) # DIG4 (rightmost)
    ]

    # Display each digit briefly in rapid succession
    display_digit(0, segment_states[digits[0]], decimal=True)  # DIG1 (with DP)
    time.sleep_ms(2)
    display_digit(1, segment_states[digits[1]], decimal=False) # DIG2
    time.sleep_ms(2)
    display_digit(2, segment_states[digits[2]], decimal=False) # DIG3
    time.sleep_ms(2)
    display_digit(3, segment_states[digits[3]], decimal=False) # DIG4
    time.sleep_ms(2)

def read_voltage():
    # Read the voltage from the ADC and convert it to a 4-digit number
```

```
    adc_value = adc.read_u16()  # Read ADC value (0-65535)
    voltage = (adc_value * 3.3) / 65535  # Convert to voltage (0-3.3V)
    return int(voltage * 1000 * 3)   # Convert to millivolts (e.g., 3.3V → 3300)

def main():
    while True:
        voltage_mv = read_voltage()  # Read voltage in millivolts
        # Update the display continuously
        for _ in range(100):  # Update for 100 cycles before reading voltage again
            display_voltage(voltage_mv)

if __name__ == "__main__":
    main()
```

Schematic:



My project:



Self-reflection

This week, for Computer Science we had to write another program in Python. This time, we needed a program that checks different car license plates and tells if it's a valid one or invalid by checking different criteria. I enjoyed this assignment, it wasn't hard to make, it took me two days, because I had a small bug that I had to rethink. After finishing the assignment, I added Flask to it.

For Embedded Systems, I showed the two assignments from the first week on Thursday, since I was a bit sick and missed two days, Tuesday and Wednesday. Also, on Thursday I started working on the new assignment for this week, the 7-segment display that shows a clock.

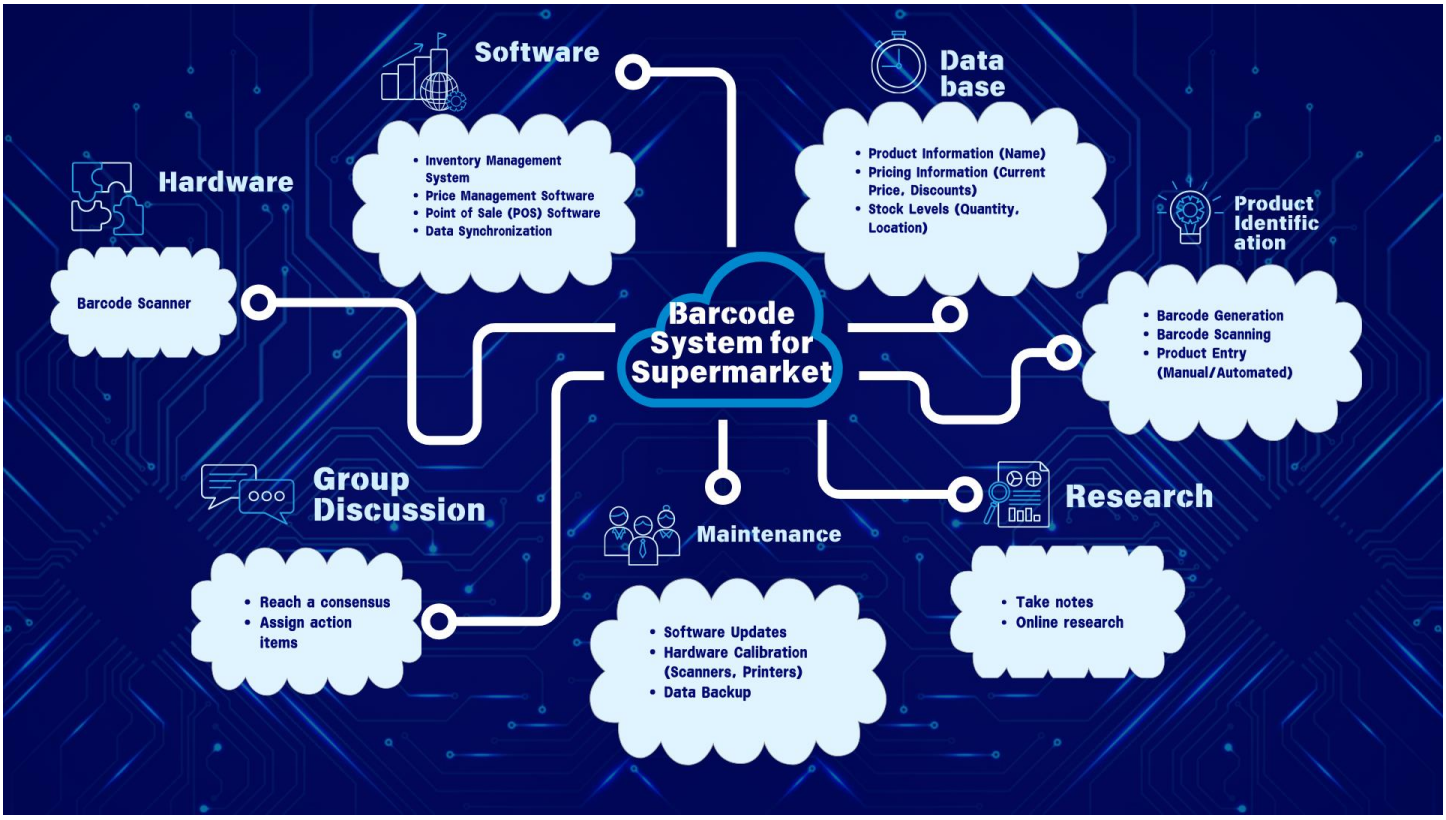
Timetable

Monday	I started working on Vanity Plates, tried to make the is_valid function. After that, I started thinking about my own problem and how I can use Python to fix it for me.
Tuesday	I had some issues with validating the car plate based on the criteria, so I tried to get rid of the bugs. After that, I made the hardware part for the Embedded Systems assignment, the 7-segment voltmeter.
Wednesday	I managed to finish Vanity Plates. I started working on the Flask implementation. The first thing I added was create a basic website for it.
Thursday	I initialized Flask, added the route to the page and finished the assignment.
Friday	I tried the code from the Raspberry Pi book to check if my hardware is working correctly, but it wasn't. After rewiring the breadboard twice, I realized the code was the problem, not the hardware. I translated the code from Arduino to Micro Python for the 7-segment 4-digits clock I made in Period 1. This time everything worked as expected.
Saturday	I focused on making the voltmeter.
Sunday	

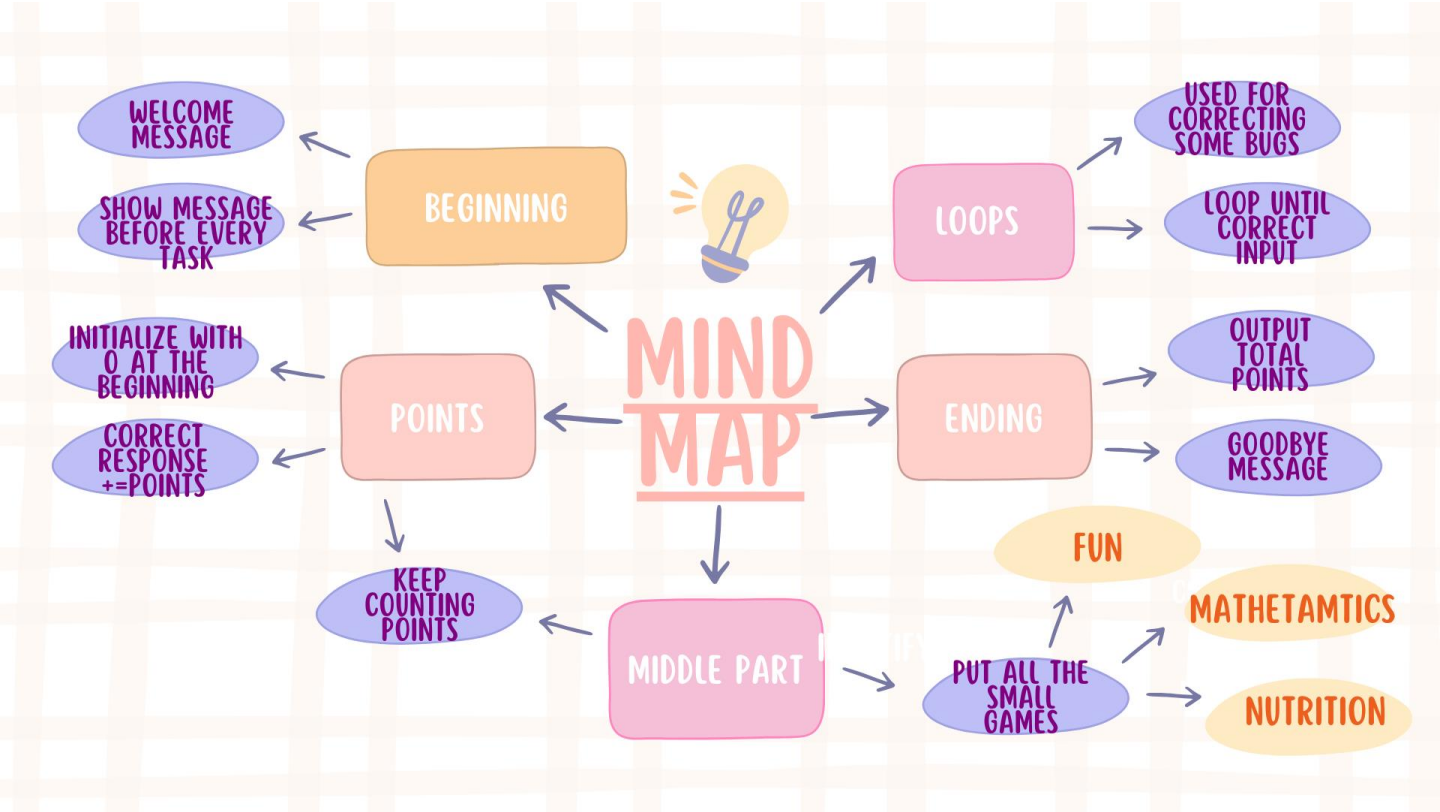
Week 3

Computer Science

Group mind map



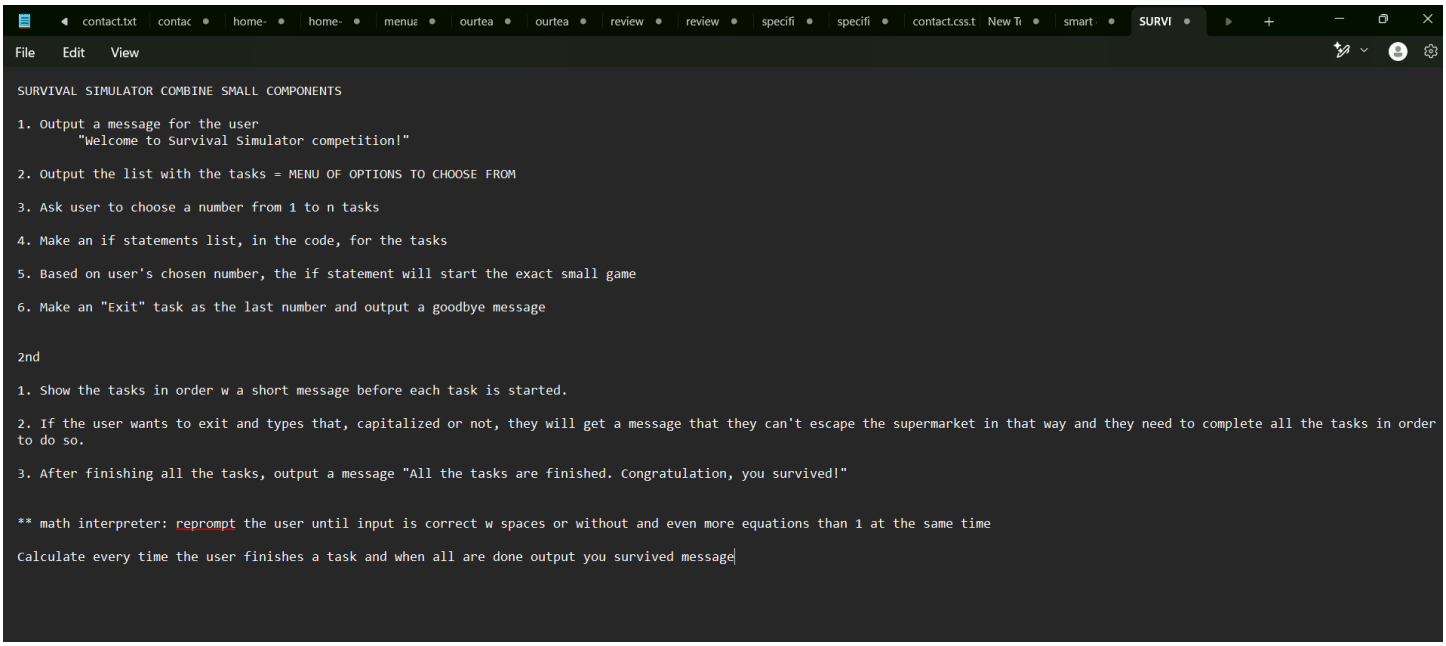
My mind map


MIND MAP
BEGINNING
WELCOME MESSAGE
SHOW MESSAGE BEFORE EVERY TASK
LOOPS
USED FOR CORRECTING SOME BUGS
LOOP UNTIL CORRECT INPUT
POINTS
INITIALIZE WITH 0 AT THE BEGINNING
CORRECT RESPONSE +=POINTS
ENDING
OUTPUT TOTAL POINTS
GOODBYE MESSAGE
MIDDLE PART
KEEP COUNTING POINTS
PUT ALL THE SMALL GAMES
FUN
MATHETAMTICS
NUTRITION

Survival Simulator

Create a program that is a survival simulator for the modern world. In this simulator, the user must solve various challenges with knowledge of file formats, nutrition, mathematics, programming, and general logic. The simulator will be a combination of puzzles and interactive choices.

The notes for the assignment:



The code for the assignment:

```
# BEGINNING #

# initialize points with 0
points = 0



# output a message for the user before starting the tasks
print("Welcome to the Supermarket Survivor!")
# use this to print an empty line
print()

# make the list with the tasks that need to be done in order to escape the simulator
list = ["1. File Extensions",
        "2. Math Interpreter",
```

```
        "3. Meal Time",
        "4. Nutrition Facts",
        "5. Coke Machine",
        "6. CamelCase",
        "7. Outdated",
        "8. Guessing Game",
        "9. Bitcoin Index",
        "10. Emojize",
        "11. Exit"
        ]

print("You have to complete all of these tasks in order to escape the supermarket simulator:")
# use a for loop to print the list in order
for i in range (11):
    print(list[i])

# use this to print an empty line
print()



#
# 1ST TASK: FILE EXTENSIONS #
#

print("TASK NUMBER 1: FILE EXTENSIONS")
# ask for input to check the file
file = input("Input file name: ")
# make it lower and without spaces
file = file.lower().strip()

# check the end of the string and output the type of the file it is
if (file.endswith(".gif")):
    print("image/gif")
elif(file.endswith(".jpg") or file.endswith(".jpeg")):
    print("image/jpeg")
elif(file.endswith(".png")):
    print("image/png")
elif(file.endswith(".pdf")):
    print("application/pdf")
elif(file.endswith(".txt")):
    print("text/plain")
elif(file.endswith(".zip")):
    print("application/zip")
# if the file type doesn't match, show this
```

```
else:
    print("application/octet-stream")
    # if the type of file is not on the list, points = -1

# if the type of file was not on the list, the points become 0 again
# if it was on the list, the points become 1
# it's an easier approach
points += 1

print(f"Points: {points}")
print()



#
# 2ND TASK: MATH INTERPRETER #
#



import random

print("You encounter a vault with a mathematical puzzle. Solve it to unlock the vault.")

levels = [
    ("2 + 3 * 4", 14),
    ("(5 + 3) / 2", 4),
    ("10 - 2 * 3 + 1", 5),
    ("2 ** 3 + 4", 12),
]

for expression, answer in levels:
    print(f"Solve: {expression}")

    # loop used to constantly check if the answer is a number or not
    while True:
        user_answer = input("Your answer: ")

        # check if the input is a valid number
        if user_answer.replace(".", "").isdigit():  # allows floats
            user_answer = float(user_answer)
            break  # exit the loop if the input is valid

        else:
            print("Invalid input. Please enter a number.")
```

```
    # Check if the user's answer is correct
    if user_answer == answer:
        points += 1


print(f"Points: {points}")
print()




#
# 3RD TASK: MEAL TIME #
#


print("TASK NUMBER 3: MEAL TIME")

# implement a program that prompts the user for a time and outputs whether it’s breakfast time, lunch time, or dinner time

def main():

    #loop to constantly check if the time is entered correctly
    while True:
        # ask for input hour
        time = input("What time is it? (HH:MM) ")

        # validate the input format
        if ":" in time and len(time.split(":")) == 2:
            hours, minutes = time.split(":")
            if hours.isdigit() and minutes.isdigit():
                hours = int(hours)
                minutes = int(minutes)
                if 0 <= hours < 24 and 0 <= minutes < 60:
                    # convert the input time to a float representing hours
                    final_time = convert(time)

                    # check for the correct hours and print the corresponding meal time
                    if 7.0 <= final_time <= 8.0:
                        print("breakfast time")
                        points += 1
```

```
                    elif 12.0 <= final_time <= 13.0:
                        print("lunch time")
                        points += 1
                    elif 18.0 <= final_time <= 19.0:
                        print("dinner time")
                        points += 1
                    else:
                        print("It's not meal time.")
                    break  # exit the loop after processing the valid input

                # if the input is invalid, reprompt the user
                print("Invalid time format. Please enter a valid time in HH:MM format.")



def convert(time):
    # split the time into hours and minutes
    hours, minutes = time.split(":")

    # convert hours and minutes to integers
    hours = int(hours)
    minutes = int(minutes)

    # used to convert the time into a single float value
    # dividing the minutes by 60 converts them into a fractional part of an hour
    # 2 hours and 30 minutes becomes 2 + (30 / 60) = 2 + 0.5 = 2.5 hours
    final_time = hours + (minutes / 60)

    # returning the float number so I can use it in def main()
    return final_time

if __name__ == "__main__":
    main()

print(f"Points: {points}")
print()



#
# 4TH TASK:  NUTRITION FACTS #
#
```

```python
print("TASK NUMBER 4: NUTRITION FACTS")

# prompt user for input fruit
# make it lowercase
fruit = input("Enter a fruit: ").lower()

# dictionary of fruits and their calorie counts (per FDA poster)
fruit_calories = {
    'apple': 130,
    'avocado': 50,
    'banana': 110,
    'cantaloupe': 50,
    'grapefruit': 60,
    'grapes': 90,
    'honeydew melon': 50,
    'kiwifruit': 90,
    'lemon': 15,
    'lime': 20,
    'nectarine': 60,
    'orange': 80,
    'peach': 60,
    'pear': 100,
    'pineapple': 50,
    'plums': 70,
    'strawberries': 50,
    'sweet cherries': 100,
    'tangerine': 50,
    'watermelon': 80
}


# check if the fruit is in the dictionary
if fruit in fruit_calories:
    # output its calories
    print(f"Calories: {fruit_calories[fruit]}")
    # another way is to concatenate strings and variables using the + operator
    # print("Calories: " + str(fruit_calories[fruit]))
else:
    # make a loop that continously prompts user for input until it s a correct one
    while fruit not in fruit_calories:
        fruit = input("Enter a fruit: ").lower()
```

```
points += 1

# another method of solving it is by making 20 different booleans
print(f"Points: {points}")
print()




#
# 5TH TASK: COKE MACHINE #
#



print("TASK NUMBER 5: COKE MACHINE")

# inform customer about the amount due
print("Amount Due: 50 Cents.")

# accepted coin denominations
accepted_coins = [25, 10, 5]

# initialize the total amount paid
total_paid = 0

# keep asking for coins until the amount due is met or exceeded
while total_paid < 50:
    # calculate the remaining amount due
    amount_due = 50 - total_paid

    # inform the user of the amount due
    print(f"Amount Due: {amount_due}")
    # another way of writing this is w two print()
    # first one w the message
    # 2nd one with the amount due as an int

    # prompt user to insert a coin
    new_coin = int(input("Insert Coin: "))

    # check if the coin is valid, from the list
    if new_coin in accepted_coins:
        # add the coin to the total amount paid
        total_paid += new_coin
```

```
        # every time user enters a correct coin, he earns a points
        points += 1
    else:
        # if the coin is invalid, ignore it and inform the user
        print("Invalid coin. Please insert a valid coin (25, 10, or 5 cents).")


# if the total paid is 50 cents or more, calculate change owed
if total_paid >= 50:
    change_owed = total_paid - 50
    print(f"Change Owed: {change_owed}")

print(f"Points: {points}")
print()




#
# 6TH TASK: CAMELCASE"
#



print("TASK NUMBER 6: CAMELCASE")

# prompt user for input
camelCase = input("camelCase: ")

# strings in Python are immutable
# build a new string
# making function for snake_case
def snake_case(camelCase):
    # create an empty string to build the result
    snake_case = ""

    # iterate over each character in the string
    for i in camelCase:
        if (i.isupper()):
            # if it finds a character that is uppercase, add the underline, then the word
lowercase
            snake_case += "_" + i.lower()
        else:
            # if character is lowercase, keep it like that
            snake_case += i
```

```
    return snake_case

# output the result
# pass in the camelCase string as an argument
print("snake_case:", snake_case(camelCase))
points += 1



print(f"Points: {points}")
print()



#
# 7TH TASK: OUTDATED #
#

from datetime import datetime

def get_valid_date():
    months = [
        "January",
        "February",
        "March",
        "April",
        "May",
        "June",
        "July",
        "August",
        "September",
        "October",
        "November",
        "December"
    ]



    while True:
        # prompt the user until a valid date is entered
        # remove leading/trailing spaces by using .strip()
        date = input("Date (month-day-year order): ").strip()

        # check if the input contains slashes (MM/DD/YYYY format)
        if "/" in date:
            # split the input into month, day, and year using "/"
```

```
            parts = date.split("/")
            # check if month, day, and year are all digits
            if len(parts) == 3:
                month, day, year = parts
                if month.isdigit() and day.isdigit() and year.isdigit():
                    # if yes, turn them into integers
                    month = int(month)
                    day = int(day)
                    year = int(year)
                    # validate the month, day, and year ranges
                    if (1 <= month <= 12 and 1 <= day <= 31 and 1 <= year <= 9999):
                        # return the date in YYYY-MM-DD format
                        return f"{year:04}-{month:02}-{day:02}"

        # check if the input contains a space and a comma (Month Day, Year format)
        elif " " in date and "," in date:
            # split the input into month_day and year using the comma
            month_day, year = date.split(",")
            # remove any leading/trailing spaces from month_day and year
            month_day = month_day.strip()
            year = year.strip()
            # check if the first part of month_day is a valid month name
            if month_day.split()[0] in months:
                # split month_day into month and day using the space
                month, day = month_day.split()
                # check if day and year are digits
                if day.isdigit() and year.isdigit():
                    day = int(day)
                    year = int(year)
                    # get the index of the month in the months list and add 1 (since list
indices start at 0)
                    month_index = months.index(month) + 1
                    if (1 <= day <= 31 and 1 <= year <= 9999):
                        # return the date in YYYY-MM-DD format
                        return f"{year:04}-{month_index:02}-{day:02}"

        # if the input doesn't match any valid format
        print("Invalid date format. Please use month-day-year or month/day/year.")



# make a function for certain expired items
def filter_expired_groceries():
    # make a list for the groceries
    groceries = [
```

```
        ("milk", "2023-10-01"),
        ("bread", "2023-09-15"),
        ("eggs", "2023-10-10"),
        ("cheese", "2023-09-20"),
    ]

    today = datetime.today().date()

    for item, expiry in groceries:
        expiry_date = datetime.strptime(expiry, "%Y-%m-%d").date()
        if expiry_date < today:
            print(f"{item} is expired.")
        else:
            print(f"{item} is still good.")

    print("You filtered out the expired items and found a clue to the next area.")



def main():
    print("TASK NUMBER 7: OUTDATED")
    print("You need to validate a date and filter expired groceries to proceed.")

    # Validate the date
    valid_date = get_valid_date()
    print(f"Validated Date: {valid_date}")

    # Filter expired groceries
    filter_expired_groceries()



if __name__ == "__main__":
    main()

points += 1

print(f"Points: {points}")
print()




#
# 8TH TASK: GUESSING GAME" #
```

```
#

print("TASK NUMBER 8: GUESSING GAME")
print("Choose a positive number.")

import random

# prompt user for input level, n
level = input("Level: ")

# check if level is not a digit and reprompt
while not (level.isdigit()):
    level = input("Level: ")

# convert level input into integer after checking
level = int(level)

# check if level is negative and reprompt user
while(level < 0):
    level = int(input("Enter a level: "))



# randomly generate an integer between 1 and n (inclusive n), using the random module
# integer from 0 to level (n) inclusive
# when using functions from a module, prefix the function with the module name
random_integer = random.randrange(level+1)

# check if the random integer is between 1 and n
while(random_integer < 1 or random_integer > level):
    random_integer = random.randrange(level+1)

# prompt user to guess the random integer
guess = input("Guess: ")
# check if guess is not a digit or is less than 1, and reprompt
while not guess.isdigit() or int(guess) < 1:
    guess = input("Guess: ")
guess = int(guess)



# if the guess is not correct
while not(guess == random_integer):
    if(guess < random_integer):
        print("Too small!")
```

```
        guess = int(input("Guess: "))
    elif(guess > random_integer):
        print("Too large!")
        guess = int(input("Guess: "))


# if the guess is correct
if(guess == random_integer):
    print("Just right!")
    points += 10


print(f"Points: {points}")
print()




#
# 9TH TASK: BITCOIN INDEX #
#



import requests  # import the library to make HTTP requests to the API

print("TASK NUMBER 9: BITCOIN INDEX")
print("You find a sign that says 'Buy Bitcoin!'")

# prompt the user to input the number of Bitcoins they want to buy and convert it to a float
bitcoins = float(input("How many Bitcoins would you like to buy? "))

try:
    # send a GET request to the CoinCap API to fetch Bitcoin data
    response = requests.get("https://api.coincap.io/v2/assets/bitcoin")

    # parse the JSON response into a Python dictionary
    data = response.json()

    # extract the Bitcoin price in USD from the JSON response and convert it to a float
    price_usd = float(data["data"]["priceUsd"])
```

```
    # calculate the total cost of the Bitcoins by multiplying the number of Bitcoins by the
price
    total_cost = bitcoins * price_usd

    # display the total cost in USD with proper formatting (2 decimal places and commas)
    print(f"The cost of {bitcoins} Bitcoins is ${total_cost:,.2f}.")

    # ask the user if they want to proceed with the purchase and convert their input to lowercase
    decision = input("Would you like to proceed with the purchase? (yes/no): ").lower()

    # check the user's decision
    if decision == "yes":
        # if the user says "yes", confirm the purchase and allow them to proceed
        print("You bought Bitcoin and gained access to the next area.")
        points += 1
    else:
        # if the user says "no", inform them that the door remains locked
        print("You decided not to buy Bitcoin.")
except:
    # handle any errors that occur during the API request or data processing
    print("Failed to retrieve Bitcoin price.")



print(f"Points: {points}")
# print an empty row
print()



#
# 10TH TASK: EMOJIZE #
#

print("TASK NUMBER 10: EMOJIZE")

# List of emoji clues and their decoded meanings
clues = {
    "🍪": "cookies",  # Find fruit and cookies
    "🔑🚪": "find the key to the door",  # Find the key to the door
    "💡🔍": "see the light",  # Look for a light and search
}
```

```
print("You will find a series of emojis now. Decode the message to proceed and exit the
supermarket.")

# loop through each emoji clue
for emoji_clue, decoded_meaning in clues.items():
    print(f"\nClue: {emoji_clue}")

    # prompt the user to guess the meaning of the emoji clue
    user_guess = input("What do you think this clue means? ").strip().lower()

    # check if the user's guess matches the correct meaning
    if user_guess == decoded_meaning:
        print("You decoded the clue.")
        points += 10
    else:
        print(f"The correct answer was: {decoded_meaning}")

print(f"Points: {points}")

#
# ENDING #
#
print("Congratulations! You managed to become a supermarket survivor.")
str = input("Generate a funny farewell speech: ")
```

Embedded Systems

Realtime clock and temperature on LCD-display

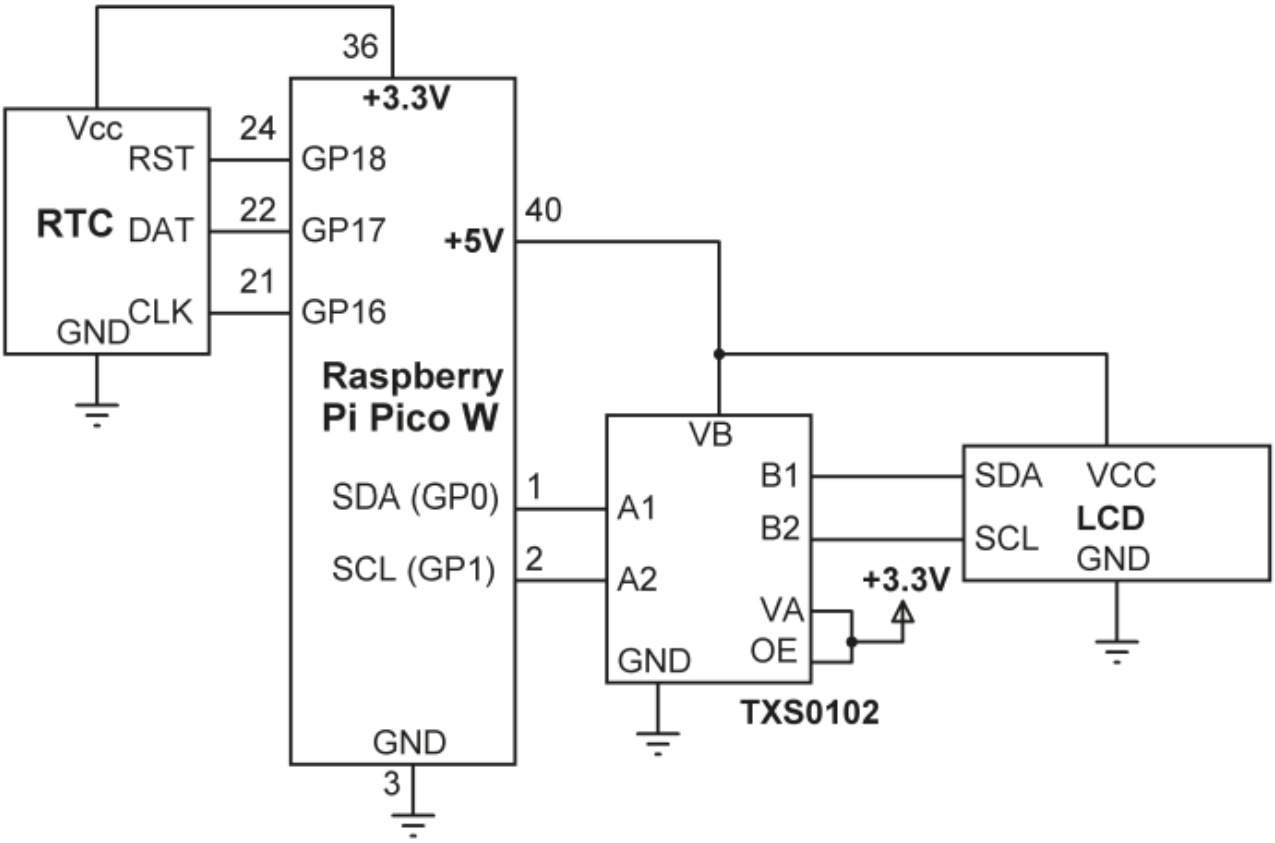
Display the Realtime date and clock, using an RTC. Besides show on the screen the correct temperature.

```
# Import necessary libraries
import machine
from machine import I2C, Pin, ADC  # For hardware interfaces (I2C, GPIO, ADC)
from lcd_api import LcdApi         # LCD API for controlling the display
from pico_i2c_lcd import I2cLcd    # Pico-specific LCD driver
import utime                       # For time-related functions (sleep, etc.)
import ds1302                      # For Real-Time Clock (RTC) functionality

# LCD Configuration
```

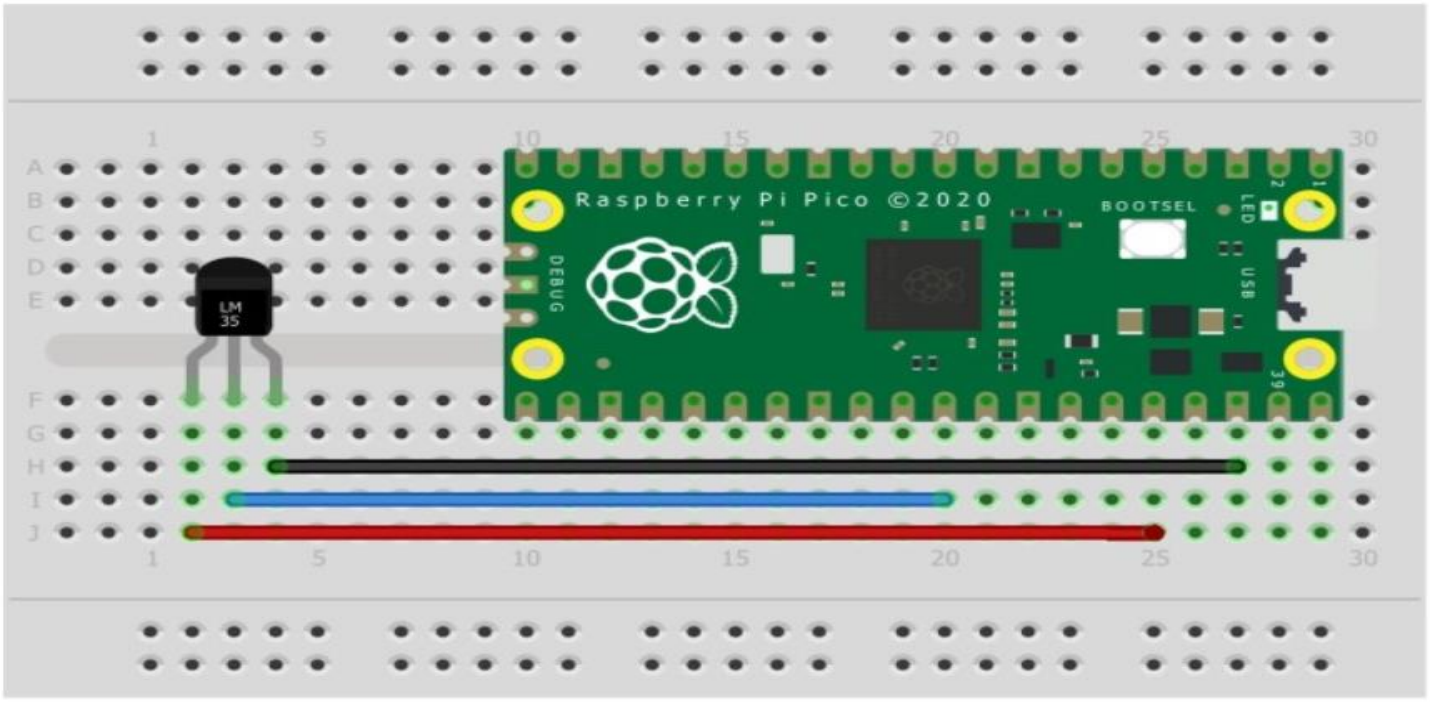
```
I2C_ADDR = 0x27    # Default I2C address for the LCD (use i2c.scan() to verify)
NRows = 2          # Number of rows on the LCD (typically 2 or 4)
NColumns = 16      # Number of columns on the LCD (typically 16 or 20)

# Initialize Hardware Components
i2c = I2C(0, sda=Pin(0), scl=Pin(1), freq=400000)  # I2C bus 0, GPIO0 (SDA), GPIO1 (SCL)
lcd = I2cLcd(i2c, I2C_ADDR, NRows, NColumns)        # Initialize LCD with I2C connection
ds = ds1302.DS1302(Pin(16), Pin(17), Pin(18))       # RTC on GPIO16 (CE), GPIO17 (IO), GPIO18
(SCLK)
lm35 = ADC(Pin(26))                                 # LM35 temperature sensor on GPIO26
(ADC0)

# Add this after LCD initialization (before main loop):
degree_char = bytearray([0x0E, 0x0A, 0x0E, 0x00, 0x00, 0x00, 0x00, 0x00])  # ° symbol pattern
lcd.custom_char(0, degree_char)  # Store in position 0

# Constants
V_REF = 3.3          # Pico's internal voltage reference (3.3V)
ADC_MAX = 65535      # Maximum value for 16-bit ADC (2^16 - 1)
TEMP_SMOOTHING = 10  # Number of samples for temperature averaging

# Variables
temp_samples = []    # List to store temperature samples for smoothing

def read_temp():
    """Read and smooth temperature from LM35 sensor
    Returns: Averaged temperature in Celsius
    """
    adc_value = lm35.read_u16()                     # Read raw ADC value (0-65535)
    voltage = (adc_value / ADC_MAX) * V_REF         # Convert to voltage (0-3.3V)
    temp_c = voltage * 100 / 1.5                    # Convert to Celsius (LM35: 10mV/°C)
    # /1.5 is likely a calibration factor

    # Smoothing using moving average
    temp_samples.append(temp_c)
    if len(temp_samples) > TEMP_SMOOTHING:
        temp_samples.pop(0)  # Remove oldest sample if we have enough

    return sum(temp_samples) / len(temp_samples)  # Return average temperature

# Optional Date/Time Setup
sel = input("Do you wish to set Date/Time? [y/N]: ")
if sel.upper() == "Y":
```

```
    print("Date time format is: Year, Month, Day, Weekday, Hour, Minute, Second, Subsecond")
    Y = int(input("Enter Year (e.g. 2023): "))
    M = int(input("Enter Month (1-12): "))
    D = int(input("Enter Day (1-31): "))
    weekday = int(input("Enter Weekday (0-6, 0=Monday): "))
    h = int(input("Enter Hour (0-23): "))
    m = int(input("Enter Minute (0-59): "))
    s = int(input("Enter Second (0-59): "))
    ss = int(input("Enter Subsecond (0-99): "))
    ds.date_time([Y, M, D, weekday, h, m, s, ss])  # Write settings to RTC

# Main Display Loop
while True:
    # Get current time and date from RTC
    (Y, M, D, weekday, hr, m, s) = ds.date_time()[:7]  # Unpack first 7 values

    # Format values with leading zeros for consistent display
    s = f"{s:02d}"  # Seconds as 2 digits (00-59)
    m = f"{m:02d}"  # Minutes as 2 digits (00-59)
    hr = f"{hr:02d}" # Hours as 2 digits (00-23)
    D = f"{D:02d}"   # Day as 2 digits (01-31)
    M = f"{M:02d}"   # Month as 2 digits (01-12)

    # Read and format temperature (changed from °-C to °C)
    temp = read_temp()
    temp_str = f"{temp:.1f}\x00C"  # \x00 calls our custom character

    # Update LCD Display
    #lcd.clear()  # Clear display before writing new content

    # Line 1: Time (left-aligned) | Temperature (right-aligned)
    lcd.move_to(0, 0)  # Column 0, Row 0
    lcd.putstr(f"{hr}:{m}:{s}")  # Display time in HH:MM:SS format

    # Right-align temperature by calculating position based on string length
    lcd.move_to(NColumns - len(temp_str), 0)  # Right-align temperature
    lcd.putstr(temp_str)  # Display temperature

    # Line 2: Date (left-aligned)
    lcd.move_to(0, 1)  # Column 0, Row 1
    lcd.putstr(f"{D}/{M}/{Y}")  # Display date in DD/MM/YYYY format

    utime.sleep(1)  # Update display every second
```

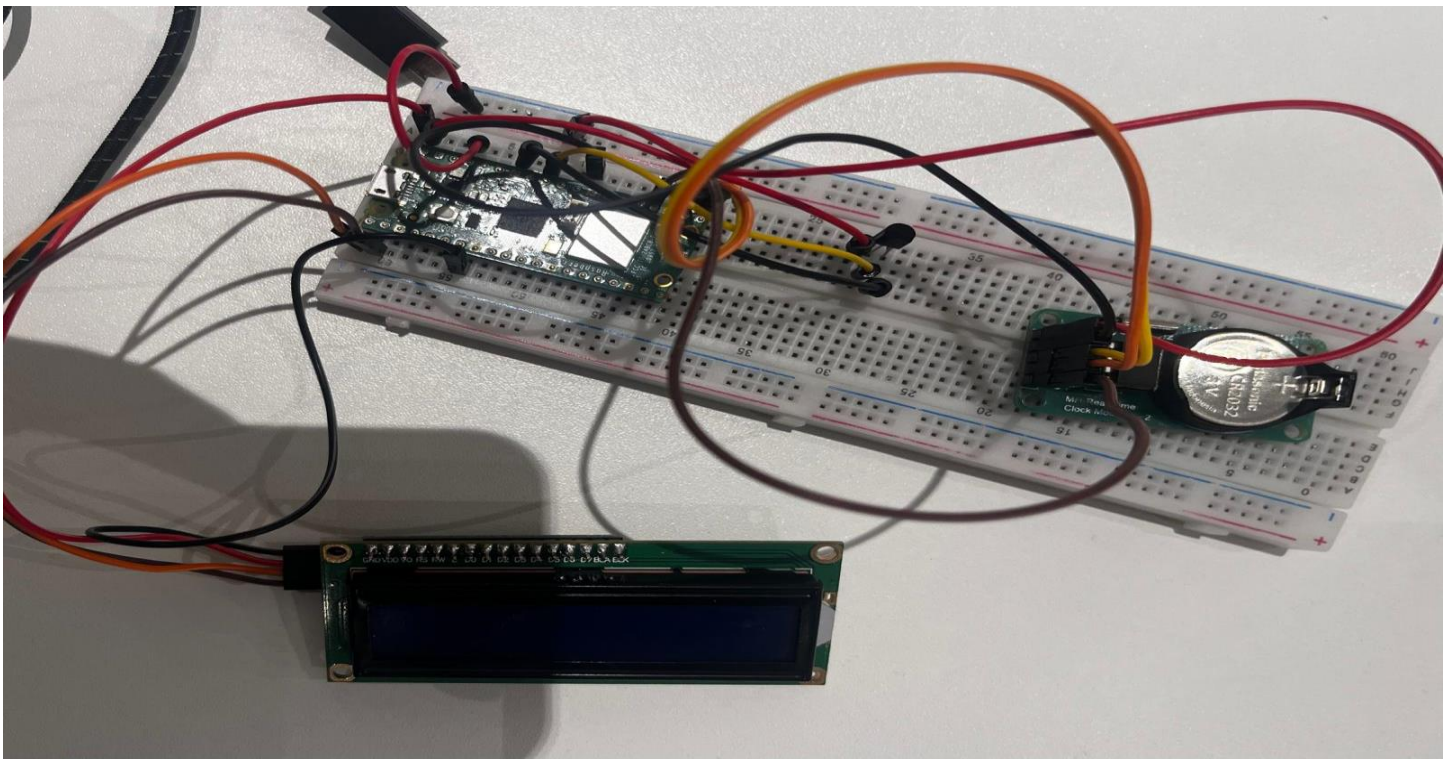
Schematic for the RTC:



Schematic for LM35:



My project:



Self-reflection

For Computer Science we had to create a Supermarket Survivor game. To realize this project, I made the small games from CS50 first. After, I added them together in the big project. I had to add a beginning, an ending. I also chose to add points, so the person knows how many correct responses had. I fixed some bugs, when the user input an incorrect form of the response (letter instead of digit etc.), I added loops so that he would be reprompted until the answer was good.

The Embedded Systems project was a bit hard to realize. At first, I tried to use DHT11 for temperature reading. At some point, I smelled smoke coming from it, so I had to switch to the LM35. Arranging the elements on the LCD's screen was also a bit tough, but I managed to make it. After making this change, it was easier to finalize the assignment properly.

Timetable

Monday	I made the Group Mind Map and my own Mind Map using Canva. After that, I started Survival Simulator.
Tuesday	I started making the short games from CS50, thinking about the big idea and even taking some notes on my laptop.
Wednesday	I continued with the short games.
Thursday	I tried to add all the short games together in one code. I also added points to it so you can get some feedback. The basic idea is that you can't choose your games or get out of the game until you finished every task in the order I want.
Friday	I finished the Survival Simulator.
Saturday	I worked on the Realtime clock and temperature reading.
Sunday	

Week 4

Computer Science

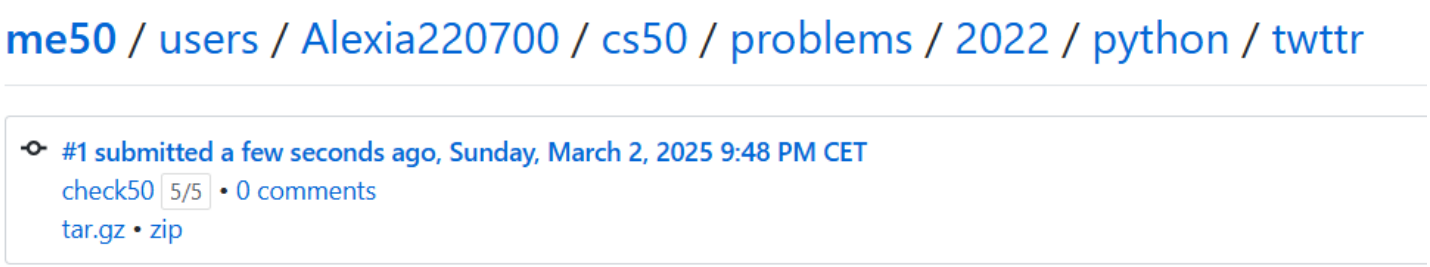
Just setting up my twttr

When texting or tweeting, it's not uncommon to shorten words to save time or space, as by omitting vowels, much like Twitter was originally called twttr. In a file called twttr.py, implement a program that prompts the user for a str of text and then outputs that same text but with all vowels (A, E, I, O, and U) omitted, whether input in uppercase or lowercase.

This is the code:

```
def main():
    # ask user for input string
    string = input("Input: ")

    # call the shorten function to remove vowels
    # iterates through each character in the string
    # joins the characters back into a single string and returns it
    shortened_string = shorten(string)

    # output the result
    print("Output:", shortened_string)

def shorten(word):
    # define the vowels to be removed
    vowels = "AEIOUaeiou"

    # construct a new string without vowels
    # [char for char in word if char not in vowels] is a list comprehension in Python
    result = ''.join([char for char in word if char not in vowels])

    return result

# ensures that the main function is called only when the script is run directly, not when
imported as a module
if __name__ == "__main__":
    main()
```

This is the CS50 score with the link:



https://submit.cs50.io/check50/9e9886e0d529cf840070f71002eb81fb3b1f380c

Testing my twttr

Then, in a file called test_twttr.py, implement one or more functions that collectively test your implementation of shorten thoroughly, each of whose names should begin with test_ so that you can execute your tests with: pytest test_twttr.py .

This is the code for the assignment:

```python
import twttr
from twttr import shorten

def test_shorten_mixed_case():
    # test a string with mixed case vowels
    assert shorten("Hello World") == "Hll Wrld"

def test_shorten_numbers_and_symbols():
    # Test a string with numbers and symbols (no vowels)
    assert shorten("123!@#") == "123!@#"

def test_shorten_all_vowels():
    # Test a string with only vowels
    assert shorten("AEIOUaeiou") == ""

def test_shorten_lowercase_vowels():
    # Test a string with lowercase vowels
    assert shorten("banana") == "bnn"
```

This is the CS50 check with the link:

me50 / users / Alexia220700 / cs50 / problems / 2022 / python / tests / twttr

#1 submitted a few seconds ago, Sunday, March 2, 2025 10:02 PM CET
check50 8/8 • 0 comments
tar.gz • zip

https://submit.cs50.io/check50/14f1c0e1e065319a2d37e9eb4b2f5dfbfed3ac98

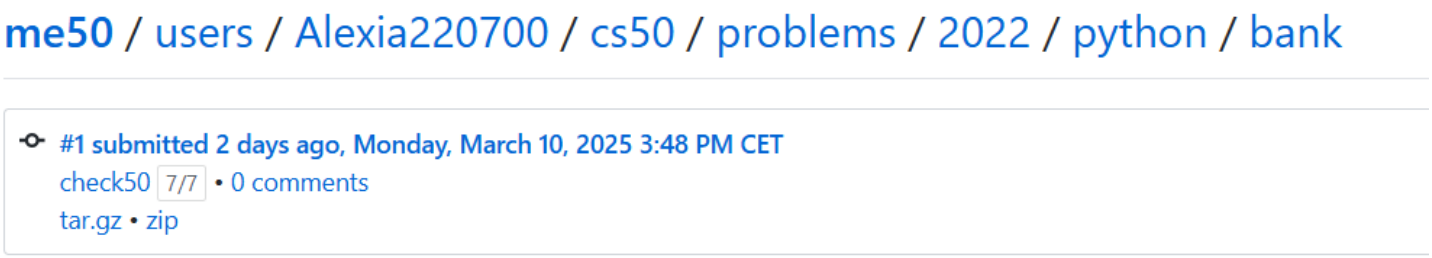
Back to the bank

This is the part for bank.py:

In a file called bank.py, reimplement Home Federal Savings Bank from Problem Set 1, restructuring your code per the below, wherein value expects a str as input and returns an int, namely 0 if that str starts with “hello”, 20 if that str starts with an “h” (but not “hello”), or 100 otherwise, treating the str case-insensitively. You can assume that the string passed to the value function will not contain any leading spaces. Only main should call print.

This is the code for it:

```
def main():
    # Prompt the user for a greeting
    greeting = input("Greeting: ").strip().lower()
    # Call the value function and print the result
    print(f"${value(greeting)}")


def value(greeting):
    # Check the conditions and return the appropriate amount
    if greeting.startswith("hello"):
        return 0
    elif greeting.startswith("h"):
        return 20
    else:
        return 100


if __name__ == "__main__":
    main()
```



https://submit.cs50.io/check50/946253ee3dc5a8857cc5b7c52b9176fd773337c0https://submit.cs50.io/check50/946253ee3dc5a8857cc5b7c52b9176fd773337c0

This is the part for Back to the bank:

Then, in a file called test_bank.py, implement three or more functions that collectively test your implementation of value thoroughly, each of whose names should begin with test_ so that you can execute your tests with: pytest test_bank.py.

This is the code:

```python
from bank import value

def test_hello():
    assert value("hello") == 0

def test_h_but_not_hello():
    assert value("hi") == 20

def test_case_insensitivity():
    # Test case insensitivity
    assert value("HELLO") == 0
    assert value("hELLo") == 0
    assert value("HI") == 20
    assert value("hOw are you") == 20
    assert value("WHATS UP") == 100

def test_whitespace():
    # Test greeting with leading/trailing whitespace
    assert value("  hello") == 0
    assert value("hello  ") == 0
```

This is the score from CS50:

me50 / users / Alexia220700 / cs50 / problems / 2022 / python / tests / bank

#1 submitted 23 days ago, Monday, March 10, 2025 4:31 PM CET
check50 5/5 • 0 comments
tar.gz • zip

https://submit.cs50.io/check50/8bff2fda3089549e15ec38ff969f9cb99984075a

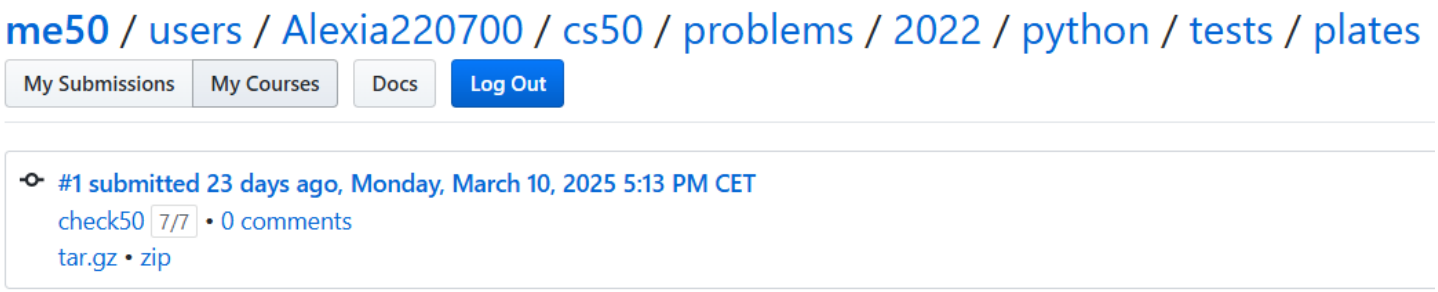
Re-requesting a Vanity Plate

In a file called plates.py, reimplement Vanity Plates from Problem Set 2, restructuring your code per the below, wherein is_valid still expects a str as input and returns True if that str meets all requirements and False if it does not, but main is only called if the value of __name__ is "__main__". Then, in a file called test_plates.py, implement four or more functions that collectively test your implementation of is_valid thoroughly, each of whose names should begin with test_ so that you can execute your tests with: pytest test_plates.py.

This is the code:

```
from plates import is_valid

def test_invalid_length():
    assert is_valid("A") == False
    assert is_valid("ABCDEFG") == False

def test_invalid_characters():
    assert is_valid("AB!23") == False  # contains special character
    assert is_valid("AB 123") == False  # contains space

def test_first_number_zero():
    assert is_valid("AB0123") == False

def test_all_numbers():
    assert is_valid("123456") == False

def test_mixed_valid():
    assert is_valid("AB123C") == False
```

This is the score from CS50:



https://submit.cs50.io/check50/1ec792149d9c682ddba2550ffa71db691f5579e1

Nutrition Facts

In a file called nutrition.py, implement a program that prompts consumers users to input a fruit (case-insensitively) and then outputs the number of calories in one portion of that fruit, per the FDA's poster for fruits, which is also available as text. Capitalization aside, assume that users will input fruits exactly as written on the poster (e.g., strawberries, not strawberry). Ignore any input that isn't a fruit.

This is the code for the assignment:

```
# prompt user for input fruit
# make it lowercase
fruit = input("Enter a fruit: ").lower()

# dictionary of fruits and their calorie counts (per FDA poster)
fruit_calories = {
    'apple': 130,
    'avocado': 50,
    'banana': 110,
    'cantaloupe': 50,
    'grapefruit': 60,
    'grapes': 90,
    'honeydew melon': 50,
    'kiwifruit': 90,
    'lemon': 15,
    'lime': 20,
    'nectarine': 60,
    'orange': 80,
    'peach': 60,
    'pear': 100,
    'pineapple': 50,
    'plums': 70,
    'strawberries': 50,
    'sweet cherries': 100,
    'tangerine': 50,
    'watermelon': 80
}


# check if the fruit is in the dictionary
if fruit in fruit_calories:
    # output its calories
```

```
    print(f"Calories: {fruit_calories[fruit]}")
    # another way is to concatenate strings and variables using the + operator
    # print("Calories: " + str(fruit_calories[fruit]))

# another method of solving it is by making 20 different booleans
```

Here is the check from CS50:

me50 / users / Alexia220700 / cs50 / problems / 2022 / python / nutrition

#1 submitted 12 days ago, Sunday, February 16, 2025 5:38 PM CET
check50 7/7 • 0 comments
tar.gz • zip

https://submit.cs50.io/check50/479aa7a01c33208fe3d9f756403c4f8f09426fa7

Nutrition Unittest

This is the changed code for Nutrition Facts:

```
# dictionary of fruits and their calorie counts
fruit_calories = {
    'apple': 130,
    'avocado': 50,
    'banana': 110,
    'cantaloupe': 50,
    'grapefruit': 60,
    'grapes': 90,
    'honeydew melon': 50,
    'kiwifruit': 90,
    'lemon': 15,
    'lime': 20,
    'nectarine': 60,
    'orange': 80,
    'peach': 60,
    'pear': 100,
    'pineapple': 50,
    'plums': 70,
    'strawberries': 50,
    'sweet cherries': 100,
    'tangerine': 50,
    'watermelon': 80
}

def get_calories(fruit):
    # returns the calories for a given fruit
    # if the fruit is not found, returns None
    return fruit_calories.get(fruit.lower().strip(), None)
```

This is the code for the Unittest:

```
from nutrition import get_calories

def test_apple():
    assert get_calories("apple") == 130

def test_banana():
    assert get_calories("banana") == 110
```

```python
def test_watermelon():
    assert get_calories("watermelon") == 80

def test_case_insensitivity():
    # Test case insensitivity
    assert get_calories("APPLE") == 130
    assert get_calories("BaNaNa") == 110
    assert get_calories("WATERMELON") == 80

def test_unknown():
    # Test unknown fruit
    assert get_calories("unknown") is None

def test_whitespace():
    # Test input with leading/trailing whitespace
    assert get_calories("  apple") == 130
    assert get_calories("banana  ") == 110
    assert get_calories("  watermelon  ") == 80
```

Refueling

In a file called fuel.py, reimplement Fuel Gauge from Problem Set 3, restructuring your code per the below, wherein: convert expects a str in X/Y format as input, wherein each of X and Y is an integer, and returns that fraction as a percentage rounded to the nearest int between 0 and 100, inclusive. If X and/or Y is not an integer, or if X is greater than Y, then convert should raise a ValueError. If Y is 0, then convert should raise a ZeroDivisionError.

Gauge expects an int and returns a str that is:

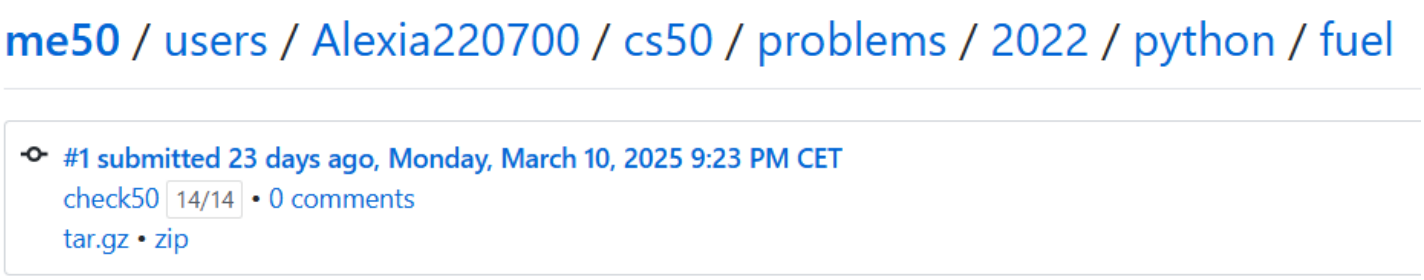
- "E" if that int is less than or equal to 1,
- "F" if that int is greater than or equal to 99,
- and "Z%" otherwise, wherein Z is that same int.

This is the code for fuel:

```
def main():
    fraction = input("Enter a fraction (X/Y format): ")

    try:
        # Pass the user's input (fraction) to the convert function
        percentage = convert(fraction)

        # If no errors occurred, call the gauge function and print the result
        print(gauge(percentage))
    except (ValueError, ZeroDivisionError) as e:
        # Handle errors and print the error message
        print(e)
        fraction = input("Enter a fraction (X/Y format): ")


def convert(fraction):
    # Split the input string
    try:
        X, Y = fraction.split('/')
    except ValueError:
        raise ValueError("Input must be in X/Y format.")

    # Check if X and Y are integers
    try:
        X = int(X)
```

```
        Y = int(Y)
    except ValueError:
        raise ValueError("X and Y must be integers.")



    # Check if X is greater than Y
    if X > Y:
        raise ValueError("X cannot be greater than Y.")
        return False

    # Check if Y is zero
    if Y == 0:
        raise ZeroDivisionError("Y cannot be zero.")
        return False

    # Calculate the percentage
    percentage = round((X / Y) * 100)

    # Ensure the percentage is between 0 and 100
    if percentage < 0:
        percentage = 0
    elif percentage > 100:
        percentage = 100

    return percentage




def gauge(percentage):
    if percentage <= 1:
        return "E"
    elif percentage >= 99:
        return "F"
    else:
        return f"{percentage}%"


if __name__ == "__main__":
    main()
```

This is the CS50 score:



https://submit.cs50.io/check50/cf12d2255941d7aa0981c7d8c97eb3d310d2e10e

Then, in a file called test_fuel.py, implement two or more functions that collectively test your implementations of convert and gauge thoroughly, each of whose names should begin with test_ so that you can execute your tests with: pytest test_fuel.py.

This is the code for testing:

```
import pytest
from fuel import convert, gauge



def test_convert_valid():
    assert convert("3/4") == 75
    assert convert("1/2") == 50

def test_gauge():
    assert gauge(0) == "E"  # edge case
    assert gauge(1) == "E"  # edge case (1% should return "E")
    assert gauge(50) == "50%"  # normal case
    assert gauge(99) == "F"  # edge case

def test_convert_invalid_format():
    # Test for invalid format (non-integer input)
    with pytest.raises(ValueError):
        convert("three/four")
    with pytest.raises(ValueError):
        convert("1.5/4")

def test_convert_invalid_fraction():
    # Test for X > Y
    with pytest.raises(ValueError):
        convert("10/3")
    # Test for Y == 0
    with pytest.raises(ZeroDivisionError):
        convert("100/0")
```

Convert Celsius to Fahrenheit

```python
def main():
    celsius = input("Input temperature (Celsius): ")

    # convert Celsius to Fahrenheit
    fahrenheit = celsius_to_fahrenheit(celsius)

    print(f"{fahrenheit}°F")

def celsius_to_fahrenheit(celsius):
    # replace comma with period for decimal separator
    celsius = celsius.replace(",", ".")

    # check if the input is a valid number (integer or float)
    try:
        celsius = float(celsius)  # convert input to float
    except ValueError:
        celsius = input("Input temperature (Celsius): ")

    celsius = float(celsius)

    # calculate celsius in fahrenheit
    fahrenheit = (9/5) * celsius + 32
    return fahrenheit

main()
```

Check if a number is even or odd

```
number = input("Input: ")

if (number.isnumeric()):
    number = int(number)
else:
    while not number.isnumeric():
        number = input("Input: ")

number = int(number)
if not(number % 2 == 0):
    print("Odd number")
    exit()
else:
    print("Even number")
```

Make a factorial calculator

```
def main():
    factorial_number = input("Enter a number: ")

    factorial_number = int(factorial_number)

    if(factorial_number < 0):
        factorial_number = input("Enter a positive number: ")
        factorial_number = int(factorial_number)

    result = factorial_calculation(factorial_number)

    print(result)

def factorial_calculation(factorial_number):
    if(factorial_number == 0 or factorial_number == 1):
        return 1

    # initialize result integer with 1, because it s a multiplication
    result = 1
    for i in range(1, factorial_number + 1):
        # multiply result with each number that s less or equal to factorial_number input
        result = result * i

    return result

main()
```

Reverse a string

```python
def main():
    mes = input("Input: ")
    print(reverse(mes))

def reverse(mes):
    return mes[::-1]

if __name__ == "__main__":
    main()
```

Prime number checker

```
def main():
    while True:
        prime_number = input("Enter a prime number: ")

        # check if the input is a valid integer greater than 1
        if (prime_number.isnumeric()):
            prime_number = int(prime_number)
            if (prime_number > 1):
                break
            else:
                print("Enter a number greater than 1: ")
        else:
            print("Enter a prime number: ")

    # check if the number is prime
    if check_prime(prime_number):
        print(f"{prime_number} is a prime number.")
    else:
        print(f"{prime_number} is not a prime number.")

def check_prime(prime_number):
    nr_dividing_prime = 0

    # loop from 1 to n (inclusive)
    for i in range(1, prime_number + 1):
        if prime_number % i == 0:
            nr_dividing_prime += 1

    # a prime number has exactly 2 divisors: 1 and itself
    if nr_dividing_prime == 2:
        return True
    else:
        return False

# run the program
main()
```

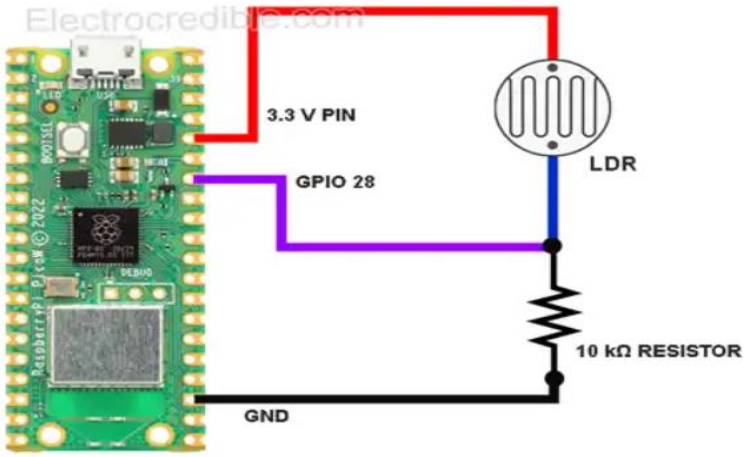
Embedded Systems

Dino cheater (HID)

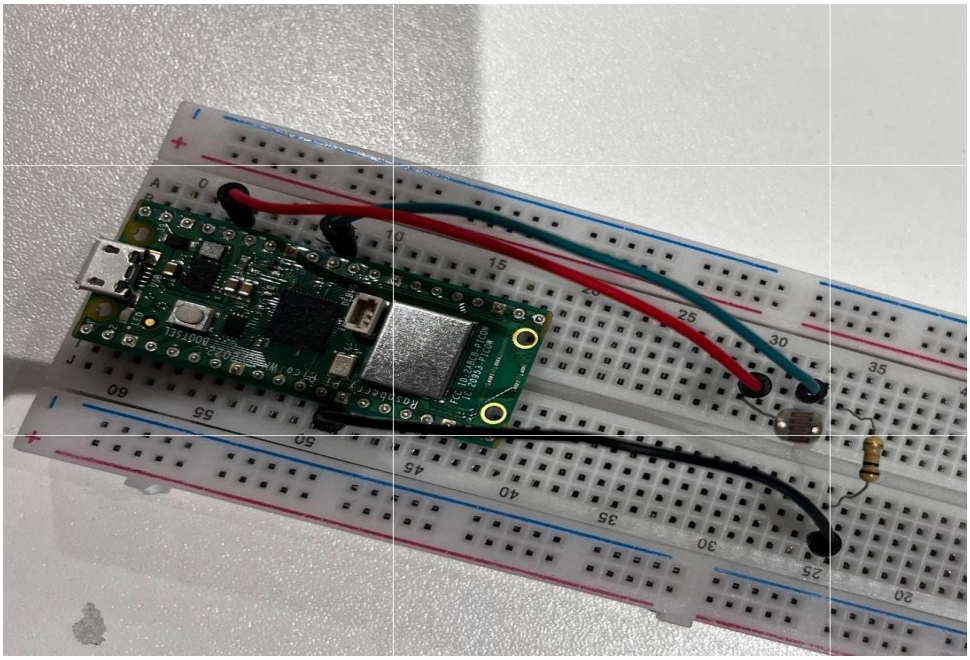
This project creates an automatic "cheater" for Chrome's Dino game using an LDR (Light Dependent Resistor). The LDR detects obstacles on the screen by sensing changes in brightness. When an obstacle (like a cactus) appears, the Arduino processes the drop in light levels and sends a signal to simulate a keypress, making the dino jump automatically.

```
import usb_hid
from adafruit_hid.keyboard import Keyboard
from adafruit_hid.keycode import Keycode
import board
import analogio
import time

# Initialize keyboard
keyboard = Keyboard(usb_hid.devices)

# Initialize LDR on analog pin A0 (GP26)
ldr = analogio.AnalogIn(board.A0)

# Calibration values (adjust these)
WHITE_SCREEN = 3490
BLACK_OBSTACLE = 2750
# This ensures reliable detection even if lighting conditions vary slightly
THRESHOLD = (WHITE_SCREEN + BLACK_OBSTACLE) // 2

def jump():
    keyboard.press(Keycode.SPACE)
    time.sleep(0.07)
    keyboard.release(Keycode.SPACE)

print("CircuitPython Dino Cheater Ready!")

while True:
    ldr_value = ldr.value
    # less light, darker screen, obstacle detected
    if ldr_value < THRESHOLD:  # Obstacle detected
        jump()
    time.sleep(0.01)
```

Schematic:



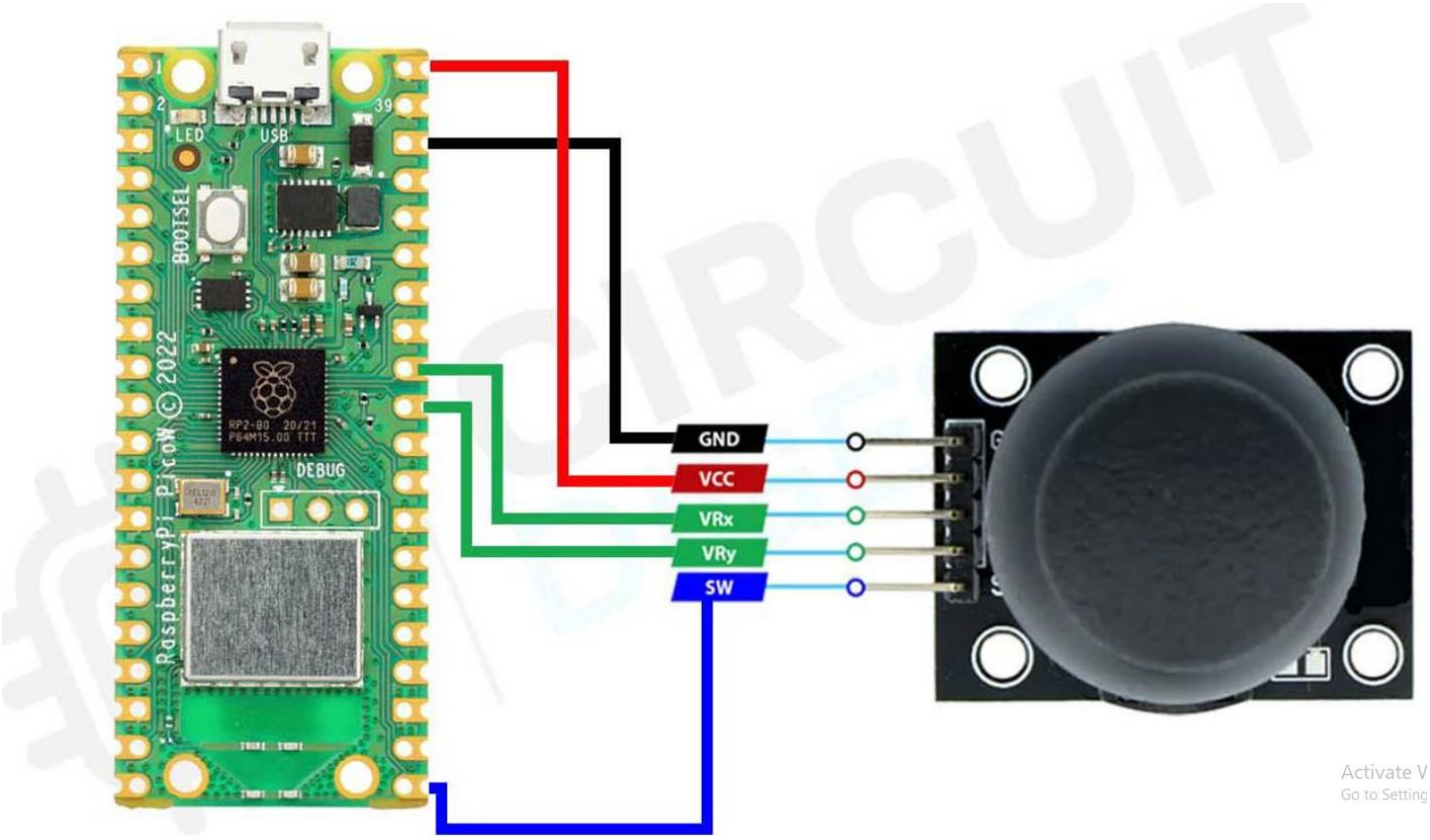
My project:



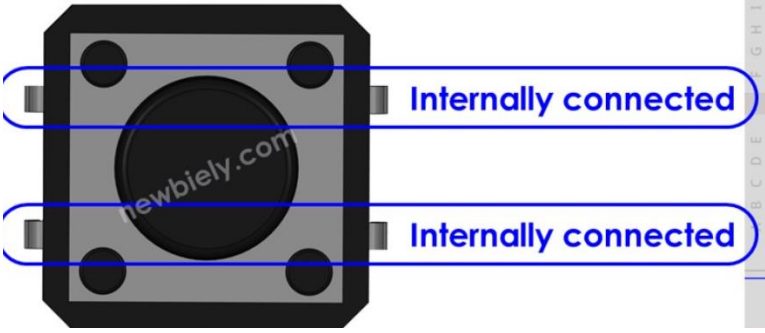
Analog joystick (HID)

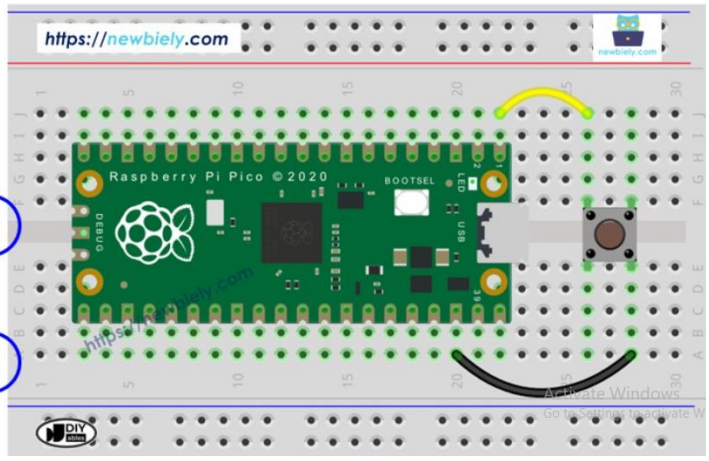
An **analog joystick HID with Raspberry Pi Pico** is a project that transforms a joystick into a USB input device using the Pico's HID (Human Interface Device) capabilities. The joystick's X and Y axes are read via the Pico's ADC (Analog-to-Digital Converter), and button presses are detected through GPIO inputs.

Schematic for Joystick:



Schematic for button:





```
import time
import board
# reads analog signals (variable voltage levels) from pins
import analogio
# enables Analog Input
# analog reads give you a range of values (e.g., 0–65535 on Pico)
import digitalio
#  enables USB Human Interface Device (HID) functionality
import usb_hid
from adafruit_hid.mouse import Mouse

# Joystick setup
x_axis = analogio.AnalogIn(board.GP26)
y_axis = analogio.AnalogIn(board.GP27)
button = digitalio.DigitalInOut(board.GP16)
button.direction = digitalio.Direction.INPUT
button.pull = digitalio.Pull.UP

# creates a virtual mouse
mouse = Mouse(usb_hid.devices)

while True:
    # Read joystick (-1.0 to 1.0 range)
    # convert the raw analog joystick values into a normalized range
    x = (x_axis.value - 32768) / 32768
    y = (y_axis.value - 32768) / 32768

    # Deadzone
    #  helps prevent unintended cursor movement when the joystick is near its center position
    # ignores small movements
    if abs(x) < 0.4: x = 0
    if abs(y) < 0.4: y = 0

    # Move mouse
    # Pushing joystick fully right → x = 1.0 → int(1.0 * 5) = 5 (moves cursor right by 5
pixels)
    # * 5 scales the sensitivity (increase to move faster / decrease to move slower)
    # 0 means no scroll wheel action
    mouse.move(int(x * 5), int(y * 5), 0)

    # Click handling
    if not button.value:
        mouse.click(Mouse.LEFT_BUTTON)
        time.sleep(0.3)
```

```
    time.sleep(0.01)
```

Peer Feedback

Kamyab:

- project is not a necessity
- it is a luxury item
- the price matters and whether it's going to constantly check the products in the refrigerator
- and how we can keep it updated
- if customer eats that item, does he need to keep updating the scanner or is it going to update on its own

Fabiana:

- for hardware, maybe in the fruit section of the fridge, track by weight how much of a product you have
- determine automatically how much product i have, not having to constantly check it with the scanner

Timetable

Monday	I made the first assignment, Just setting up my twttr and the Unittest. Besides, I made the assignment Back to the bank.
Tuesday	I made Re-requesting a Vanity Plate. I already made the Nutrition Facts assignment for the Survival Simulator, so I only had to change it a bit and add the Unittest.
Wednesday	I made the Refueling assignment.
Thursday	I made the short assignments: Convert Celsius to Fahrenheit, Check if a number is even or odd, Make a factorial calculator, Reverse a string, Prime number checkr
Friday	I worked on the Analog Joystick and made it fast since it was an easy project.
Saturday	I worked on the Dino Cheater.
Sunday	

Week 5

Computer Science

Scavenger hunt

```
# Assignment 1:

def is_valid_email(email):
    char = 0
    correct_input = 0

    if not(email.count('@') == 1):
        return False
    else:
        correct_input += 1

    # check for characters before and after '@' character
    local, domain = email.split('@')
    if not local or not domain:
        return False
    else:
        correct_input += 1

    if('.' not in domain):
        return False
    else:
        correct_input += 1

    if(correct_input == 3):
        return True
```

```
def fizzbuzz(n):
    aux = n
    divisible_by_3 = 0
    divisible_by_5 = 0

    if(n % 3 == 0):
        while(n % 3 == 0):
            n = n / 3
            divisible_by_3 += 1
```

```
    if(n % 5 == 0):
        while(aux % 5 == 0):
            aux = aux / 5
            divisible_by_5 += 1

    if not(divisible_by_3 == 0):
        return "Fizz"
    elif not(divisible_by_5 == 0):
        return "Buzz"
    elif(divisible_by_5 == 0 and divisible_by_3 == 0):
        return "FizzBuzz"
    else:
        return n
```

```
def is_palindrome(s):
    if not(s[::-1]):
        return False
    else:
        return True
```

```
def longest_word(sentence):
    longest_word_nr = 0
    nr_letters = 0
    word = ""

    for char in sentence:
        # counts the number of letters until there is a space
        nr_letters += 1

        if(char.isspace):
            if(longest_word_nr <= nr_letters):
                longest_word_nr = nr_letters
                longest_word = word
                word = ""
            # the counter goes back to zero
            nr_letters = 0

        word = word.join(char)
```

```
def add(a, b):
    return a + b

def subtract(a, b):
    return a - b

def multiply(a, b):
    return a * b

def divide(a, b):
    if b == 0:
        return "Error: Division by zero"
    return a / b
```

CS50 Shirt

In a file called shirt.py, implement a program that expects exactly two command-line arguments:

- in sys.argv[1], the name (or path) of a JPEG or PNG to read (i.e., open) as input
- in sys.argv[2], the name (or path) of a JPEG or PNG to write (i.e., save) as output

The program should then overlay shirt.png (which has a transparent background) on the input after resizing and cropping the input to be the same size, saving the result as its output.

Open the input with Image.open, per pillow.readthedocs.io/en/stable/reference/Image.html#PIL.Image.open, resize and crop the input with ImageOps.fit, per pillow.readthedocs.io/en/stable/reference/ImageOps.html#PIL.ImageOps.fit, using default values for method, bleed, and centering, overlay the shirt with Image.paste, per pillow.readthedocs.io/en/stable/reference/Image.html#PIL.Image.Image.paste, and save the result with Image.save, per pillow.readthedocs.io/en/stable/reference/Image.html#PIL.Image.Image.save.

The program should instead exit via sys.exit:

- if the user does not specify exactly two command-line arguments,
- if the input's and output's names do not end in .jpg, .jpeg, or .png, case-insensitively,
- if the input's name does not have the same extension as the output's name, or
- if the specified input does not exist.

Assume that the input will be a photo of someone posing in just the right way, like these demos, so that, when they're resized and cropped, the shirt appears to fit perfectly.

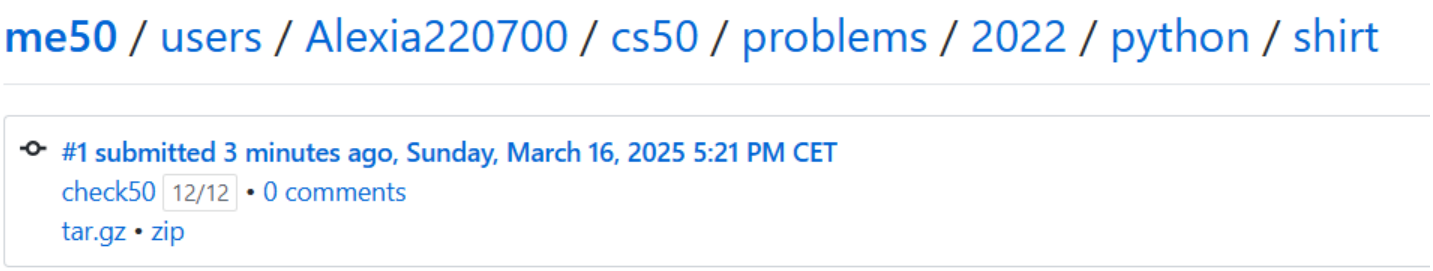
This is the code:

```
import sys
import os
# Pillow package
from PIL import Image, ImageOps
```

```
def main():
    # Check command-line arguments
    if len(sys.argv) != 3:
        sys.exit("Usage: python shirt.py input_image output_image")

    input_path = sys.argv[1]
    output_path = sys.argv[2]

    # check file extensions
    valid_extensions = {".jpg", ".jpeg", ".png"}

    # extracts the file extension from the input file path
    input_ext = os.path.splitext(input_path)[1].lower()
    # same but with the output file
    output_ext = os.path.splitext(output_path)[1].lower()

    # check if both images have the same extension
    if (input_ext not in valid_extensions) or (output_ext not in valid_extensions):
        sys.exit("Input and output must be .jpg, .jpeg, or .png files")

    if (input_ext != output_ext):
        sys.exit("Input and output must have the same extension")

    # check if input file exists
    if not (os.path.exists(input_path)):
        sys.exit(f"Input file {input_path} does not exist")

    try:
        # open input image
        with Image.open(input_path) as input_image:
            # open shirt image
            shirt = Image.open("shirt.png")

            # resize and crop input image to match shirt size
            input_image = ImageOps.fit(input_image, shirt.size)

            # overlay shirt on input image
            input_image.paste(shirt, shirt)

            # save the result
            input_image.save(output_path)
```

```
    # catch-all exception handler
    except Exception as e:
        # terminate the program immediately
        #  f-string that includes the error message (e) in the output
        sys.exit(f"An error occurred: {e}")

# ensures certain code only runs when the script is executed directly
if __name__ == "__main__":
    main()
```

This is the check from CS50:



https://submit.cs50.io/check50/5a63b06a3e0cde65caf473d2d4bc42ada70de37d

Embedded Systems

Wi-Fi scanner

Create a WiFi scanner that detects and analyzes wireless networks in an area. It provides information such as network names (SSIDs), signal strength, security protocols, and channel usage. WiFi scanners help optimize network performance, troubleshoot connectivity issues, and identify overcrowded channels.

```
import network

# Turn on and set up WiFi in station mode
wlan = network.WLAN(network.STA_IF)
wlan.active(True)

# Scan for available networks
networks = wlan.scan()  # Returns tuples: (ssid, bssid, channel, rssi, security, hidden)

# Sort networks by signal strength (RSSI) in descending order
def get_signal_strength(network_tuple):
    return network_tuple[3]  # RSSI is at index 3 (not 1)

networks = sorted(networks, key=get_signal_strength, reverse=True)

# Print header
print(f"{'ID':<4} {'SSID':<25} {'Signal':<8}")
print()

# Print network details
for netid, net in enumerate(networks, start=1):
    ssid = net[0].decode() if net[0] else "[Hidden]"  # SSID (could be hidden)
    signal = net[3]  # Signal strength (RSSI) is at index 3

    print(f"{netid:<4} {ssid:<25} {signal:<8} dBm")
```

For the WiFi scanner I didn't need a schematic since I only used the Raspberry Pi Pico.

Controlling stuff (Webserver)

Create a Webserver that can control an LED, turning it ON or OFF. Besides, read the temperature and show it on the same web page as the LED control part. For this project I used the internal temperature sensor of the Raspberry Pi Pico.

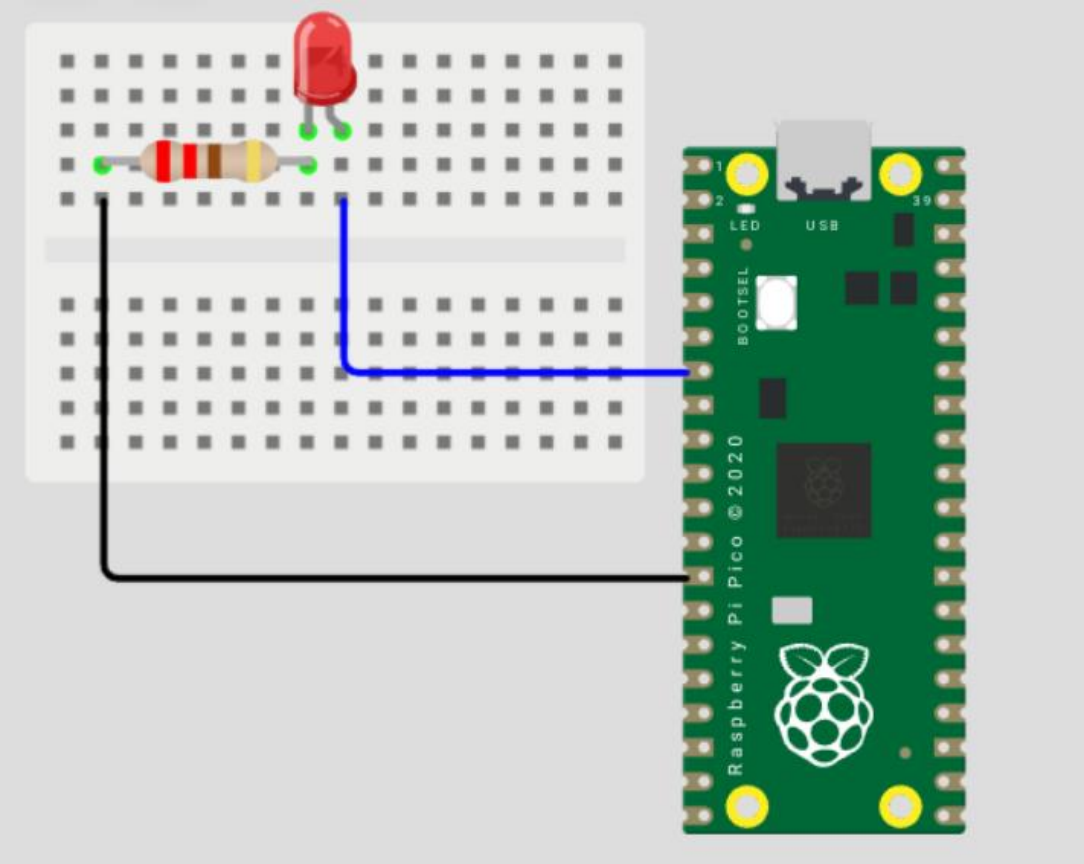
```
# Import necessary libraries
import network
import socket
from time import sleep
import machine

# ========== WiFi Setup ==========
wlan = network.WLAN(network.STA_IF)
wlan.active(True)
wlan.connect('Odido-17E2B7', 'NCS7E9XJD684LUDC')

# Wait for connection
while not wlan.isconnected():
    print('Waiting for connection...')
    sleep(1)

print('Connected!')
print('IP:', wlan.ifconfig()[0])

# ========== Hardware Setup ==========
led = machine.Pin(15, machine.Pin.OUT)  # LED on GPIO15
sensor_temp = machine.ADC(4)  # Internal temperature sensor
conversion_factor = 3.3 / (65535)  # ADC conversion factor

# ========== Web Page Template ==========
html = """<!DOCTYPE html>
<html>
<head>
    <title>Pico Web Server</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <style>
        body {
            font-family: Arial, sans-serif;
            max-width: 600px;
```

```
            margin: 0 auto;
            padding: 20px;
        }
        .container {
            border: 1px solid #ddd;
            border-radius: 8px;
            padding: 20px;
            margin-top: 20px;
        }
        button {
            padding: 10px 20px;
            font-size: 16px;
            background-color: #4CAF50;
            color: white;
            border: none;
            border-radius: 4px;
            cursor: pointer;
            margin: 5px;
        }
        .status {
            margin: 15px 0;
            padding: 10px;
            background-color: #f8f8f8;
            border-radius: 4px;
        }
        a {
            text-decoration: none;
        }
    </style>
</head>
<body>
    <h1>Raspberry Pi Pico Webserver</h1>

    <div class="container">
        <h2>LED Control</h2>
        <a href="/led/on"><button>ON</button></a>
        <a href="/led/off"><button>OFF</button></a>
        <div class="status">LED Status: %s</div>
    </div>

    <div class="container">
        <h2>Temperature</h2>
        <div class="status">Current Temperature: %.1f C</div>
        <!-- I had to add this so i could get rid of the error: showing Â before celsius-->
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
```

```
        <small>(Internal sensor reading)</small>
    </div>

    <div class="container">
        <h2>System Info</h2>
        <div>IP Address: %s</div>
    </div>
</body>
</html>
"""

# ========== Web Server Setup ==========
addr = socket.getaddrinfo('0.0.0.0', 80)[0][-1]
s = socket.socket()
s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
s.bind(addr)
s.listen(1)
print('Listening on', addr)

def read_temperature():
    """Read the internal temperature sensor"""
    try:
        reading = sensor_temp.read_u16() * conversion_factor
        temperature = 27 - (reading - 0.706)/0.001721
        return float(temperature)  # Ensure float return
    except:
        return 0.0  # Return default if error occurs

# ========== Main Server Loop ==========
while True:
    try:
        cl, addr = s.accept()
        print('Client connected from', addr)
        request = cl.recv(1024)
        request = str(request)

        # LED Control
        led_status = "ON" if led.value() else "OFF"
        if '/led/on' in request:
            led.on()
            led_status = "ON"
        elif '/led/off' in request:
            led.off()
            led_status = "OFF"
```

```
        # Read temperature
        temp = read_temperature()

        # Get IP address
        ip_address = wlan.ifconfig()[0]

        # Prepare and send response
        response = html % (led_status, temp, ip_address)
        headers = 'HTTP/1.0 200 OK\r\nContent-type: text/html\r\n\r\n'
        cl.send(headers)
        cl.send(response)
        cl.close()

    except OSError as e:
        cl.close()
        print('Connection closed:', e)
```

Schematic:


LED
USB
BOOTSEL
Raspberry Pi Pico ©2020

Self-reflection

During this week I tested my knowledge of Python programming language during the Scavenger Hunt. Besides, I finished the assignment called CS50 Shirt, where I had to match a shirt to a person from a picture.

Embedded systems gave us the opportunity to see how we can use Raspberry Pi Pico for creating different projects with internet connections. The first one, WiFi scanner, was a short code and it showed me the available networks and the signal strength of them. On the other hand, the Webserver was more complex. Firstly, I had to make the code for controlling the LED connected to the breadboard. After that, I added the part of the code for detecting the temperature, using the internal sensor of the Raspberry Pi Pico. Besides the hardware and Python parts, I also coded a web page using HTML and CSS to give it a neat structure.

Timetable

Monday	I did the Scavenger Hunt assignments during class, then I added it to my portfolio. After that, I also started working on CS50 Shirt assignment, I made the checks for the input and output arguments, then the ones for the exceptions.
Tuesday	I made the part for fitting the picture and also the exception.
Wednesday	I worked on the Wifi scanner.
Thursday	I made the code for the Webserver without the html part.
Friday	I finished the webserver by adding the html part of the code.
Saturday	
Sunday	

Week 6

Computer Science

Scavenger hunt Pt II

- For this assignment, I wanted to make a program that solves number-to-letter encryption.
- The letters are in alphabetical order, from A to Z.
- Letter A represents number 1 and so on until the end of the alphabet.

This is the code for the assignment:

```
def simple_number_encrypt(numbers):
    """Encrypts single-digit numbers to letters (1=A, 2=B, ..., 9=I)"""
    encrypted = ""
    for num in numbers:
        if num.isdigit():
            digit = int(num)
            if 1 <= digit <= 9:
                letter = chr(digit + 64)  # A=65 in ASCII
                encrypted += letter
            else:
                encrypted += num  # Keep 0 or other digits as-is
        else:
            encrypted += num  # Keep non-digit characters
    return encrypted
```

This is the Unittest:

```
import unittest
from scavengerhunt import simple_number_encrypt

class TestNumberEncryption(unittest.TestCase):

    # Tests for simple_number_encrypt
    def test_simple_encrypt_basic(self):
        self.assertEqual(simple_number_encrypt("123"), "ABC")
```

```
        self.assertEqual(simple_number_encrypt("85023"), "HE0BC")

    def test_simple_encrypt_with_non_digits(self):
        self.assertEqual(simple_number_encrypt("1a3b5"), "AaCbE")
        self.assertEqual(simple_number_encrypt("1-2-3"), "A-B-C")

    def test_simple_encrypt_edge_cases(self):
        self.assertEqual(simple_number_encrypt("0"), "0")
        self.assertEqual(simple_number_encrypt("9"), "I")
        self.assertEqual(simple_number_encrypt("10"), "A0")


if __name__ == '__main__':
    unittest.main()
```

Regex

```
import re
print(re.findall(r'c.t', 'cat cut c9t ctt'))
# ['cat', 'cut', 'c9t']

print(re.findall(r'ab*', 'a ab abb abbb abc'))
# same thing without the 'c' at the end
# if r'ab+' same thing

print(re.findall(r'colou?r', 'color colour colouur'))
# shows first two words

print(re.findall(r'a{3}', 'aa aaa aaaa aaaaa'))
# aaa aaa aaa

print(re.findall(r'a{2,4}', 'a aa aaa aaaa aaaaa'))
# idk?

print(re.findall(r'^Hello', 'Hello world'))
# prints Hello

print(re.findall(r'world$', 'Hello world/ nworld Hello'))
# it's not correct

print(re.findall(r'[aeiou]', 'hello world'))

print(re.findall(r'[^aeiou]', 'hello world'))
# everything except vowels

print(re.findall(r'\d+', 'abc123xyz'))
# all together 123

print(re.findall(r'\s+', 'Hello world'))
# empty string / whitespace

print(re.findall(r'\S+', 'Hello world'))
# ['Hello', 'world']

print(re.findall(r'w+', 'Hello, world!'))
# w

print(re.findall(r'W+', 'Hello, world!'))
```

```
# empty

print(re.findall(r'^Hello', 'Hello world/nHello again', re.MULTILINE))
# 'Hello'

print(re.match(r'Hello', 'Hello world'))
# <re.Match object; span=(0,5), match='Hello'>
# can be used in the code as a boolean in an "if match ()"

print(re.fullmatch(r'Hello world', 'Hello world'))
# <re.Match...>

print(re.split(r', ', 'apple, banana, cherry'))
# list of the 3 fruit

print(re.findall(r'\d++', 'There are 3 apples and 5 bananas.'))
# shows 3 and 5 in a list
```

Vanity Plates with Regex

import re

def main(): plate = input("Plate: ") print("Valid" if is_valid(plate) else "Invalid")

def is_valid(s): # Regex pattern explanation: # ^ - Start of string # [A-Z]{2,6} - 2 to 6 uppercase letters # (?!0) - Negative lookahead to ensure no leading zero # \d* - Zero or more digits (but first digit can't be 0) # $ - End of string # OR # ^[A-Z]{2,6}$ - Just letters (2-6 characters) pattern = r'^[A-Z]{2,6}((?!0)\d*)?$'

```
# Check if the string matches the pattern
if not re.fullmatch(pattern, s):
    return False

# Additional check that if there are numbers, they must be at the end
# and not mixed with letters
if any(char.isdigit() for char in s):
    # Find where the digits start
    digit_pos = next(i for i, char in enumerate(s) if char.isdigit())
    # Check all remaining characters are digits
    if not all(char.isdigit() for char in s[digit_pos:]):
        return False

return True
```

if **name** == "**main**": main()

Week 7

Computer Science

Product goes OOP

This Python code defines a `Pizza` class with various methods to create and interact with pizza objects.

This is the code for the assignment:

```python
class Pizza:
    def __init__(self, ingredients):
        self.ingredients = ingredients

    def __str__(self):
        return f"Pizza with {', '.join(self.ingredients)}"

    @classmethod
    def margherita(cls):
        return cls(["tomato sauce", "mozzarella", "basil"])

    @classmethod
    def pepperoni(cls):
        return cls(["tomato sauce", "mozzarella", "pepperoni"])

    @staticmethod
    def cooking_time(size):
        if size == "small":
            return 10
        elif size == "medium":
            return 15
        else:
            return 20

# Using class methods as alternative constructors
margherita = Pizza.margherita()
pepperoni = Pizza.pepperoni()
print(margherita)
```

```
print(pepperoni)

# Using static method
print(f"Cooking time: {Pizza.cooking_time('medium')} minutes")
```

CS50 Shirtificate

In a file called shirtificate.py, implement a program that prompts the user for their name and outputs, using fpdf2, a CS50 shirtificate in a file called shirtificate.pdf similar to this one for John Harvard, with these specifications:

- The orientation of the PDF should be Portrait.
- The format of the PDF should be A4, which is 210mm wide by 297mm tall.
- The top of the PDF should say “CS50 Shirtificate” as text, centered horizontally.
- The shirt’s image should be centered horizontally.
- The user’s name should be on top of the shirt, in white text.

All other details we leave to you. You’re even welcome to add borders, colors, and lines. Your shirtificate needn’t match John Harvard’s precisely. And no need to wrap long names across multiple lines.

This is the code:

```
# Import required libraries
from fpdf import FPDF  # For PDF generation
import sys  # For system exit and error handling

def main():
    """
    Main function to generate a customized CS50 shirtificate PDF.
    Handles user input, PDF creation, and error management.
    Returns 0 on success, 1 on failure.
    """
    try:
        # USER INPUT SECTION
        # Prompt user for their name and validate input
        name = input("Enter your name: ").strip()
        if not name:
            raise ValueError("Name cannot be empty")

        #  PDF INITIALIZATION
        # Create PDF document with A4 portrait orientation
        pdf = FPDF(orientation="portrait", format="A4")
        pdf.add_page()  # Add the first (and only) page
```

```
        # Configure PDF to prevent automatic page breaks
        pdf.set_auto_page_break(False, margin=0)

        #  TITLE SECTION
        # Set font for the title and add centered text
        pdf.set_font("Helvetica", "B", 36)  # Bold Helvetica, size 36
        pdf.cell(0, 40, "CS50 Shirtificate", align="C", new_x="LMARGIN", new_y="NEXT")

        # IMAGE PLACEMENT
        # Calculate center position for the shirt image (A4 width = 210mm)
        image_width = 160  # Width of the shirt image in mm
        x_position = (210 - image_width) / 2  # Center calculation

        # Add the shirt image to the PDF with error handling
        try:
            # Place image 80mm from top, centered horizontally
            pdf.image("shirtificate.png", x=x_position, y=80, w=image_width)
        except RuntimeError as e:
            # Convert library error to more descriptive error
            raise FileNotFoundError("Could not find shirtificate.png") from e

        # NAME PLACEMENT ON SHIRT
        # Configure text settings for the name
        pdf.set_font("Helvetica", "B", 24)  # Bold Helvetica, size 24
        pdf.set_text_color(255, 255, 255)  # Set text color to white (RGB)

        # Calculate centered position for the name text
        text_width = pdf.get_string_width(name)  # Get rendered width of name
        text_x = (210 - text_width) / 2  # Center calculation
        pdf.text(x=text_x, y=140, txt=name)  # Place text at calculated position

        # OUTPUT GENERATION
        # Save the PDF file
        pdf.output("shirtificate.pdf")
        print("Successfully created shirtificate.pdf")
        return 0  # Return success code

    except Exception as e:
        # Handle any errors that occur during execution
        print(f"Error: {e}", file=sys.stderr)  # Print to stderr
        return 1  # Return error code

if __name__ == "__main__":
```

```
    # Execute main function and exit with proper status code
    sys.exit(main())
```

This is the CS50 score:

me50 / users / Alexia220700 / cs50 / problems / 2022 / python / shirtificate

#1 submitted a few seconds ago, Tuesday, April 1, 2025 2:31 PM CEST
check50 2/2 • 0 comments
tar.gz • zip

https://submit.cs50.io/check50/53f89a68676588c52700cc20cd388802167cd7b7

Week 6 and 7

Embedded Systems

Attendance (RFID, RTC, web and LCD-display)

This code implements an RFID-based attendance system with:

- Pico W for Wi-Fi connectivity and web interface
- RFID reader to scan authorized tags (stored in members dictionary)
- LCD display showing greetings ("Good Morning/Afternoon/Evening") and usernames
- RTC module for accurate timekeeping and time-based greetings
- Web server showing real-time attendance status (present/absent)
- Visual feedback via RGB LED (green=present, red=absent) and buzzer

```
# Import network module for WiFi connectivity functionality
import network
# Import socket module for creating and managing network sockets
import socket
# Import machine module for hardware-level control of Raspberry Pi Pico
import machine
# Import uasyncio module for asynchronous operations (renamed to asyncio for convenience)
import uasyncio as asyncio
# Import uselect module for socket polling functionality
import uselect
# From machine module, import specific classes needed:
from machine import I2C, Pin, PWM  # I2C for communication, Pin for GPIO, PWM for pulse-width
modulation
# Import LCD API interface class
from lcd_api import LcdApi
# Import I2C LCD driver specifically for Pico
from pico_i2c_lcd import I2cLcd
# Import MFRC522 library for RFID reader functionality
from mfrc522 import MFRC522
# Import DS1302 library for real-time clock module
from ds1302 import DS1302

# Network SSID (name) for WiFi connection
SSID = "Alexia"
# Password for the WiFi network
PASSWORD = "alexia2005"
```

```
# Initialize I2C communication:
# - Using I2C interface 0
# - Frequency set to 400kHz
i2c = I2C(0, sda=Pin(0), scl=Pin(1), freq=400000)

# Initialize LCD display:
# - Device address is 0x27
# - 2 lines, 16 characters per line
lcd = I2cLcd(i2c, 0x27, 2, 16)

# Initialize Real-Time Clock (DS1302):
rtc = DS1302(Pin(2), Pin(3), Pin(4))

# Initialize RFID reader (MFRC522):
reader = MFRC522(sck=Pin(18), mosi=Pin(19), miso=Pin(16), cs=Pin(17), rst=Pin(22))

# Initialize buzzer on GPIO9 as output
buzzer = Pin(9, Pin.OUT)

# Initialize LEDs
red = PWM(Pin(10))
green = PWM(Pin(11))
blue = PWM(Pin(12))

# Set PWM frequency for all LED colors to 1000Hz
red.freq(1000)
green.freq(1000)
blue.freq(1000)

# Initialize relay on GPIO14 as output
RELAY = Pin(14, Pin.OUT)

# Set initial relay state to OFF (0)
RELAY.value(0)

# Set initial RTC date and time:
# Format: [year, month, day, weekday, hour, minute, second]
# Example: April 2, 2025 at 14:35:00 (weekday 0)
rtc.date_time([2025, 4, 2, 0, 14, 35, 0])

# Start the RTC (enable clock counting)
rtc.start()
```

```
# Dictionary of authorized RFID tags:
# Key = tag ID (integer), Value = name (string)
members = {
    327292211: "Sam",  # Member 1
    806623267: "Lisa",  # Member 2
    554433765: "Mark",  # Member 3
    545567664: "Emil",  # Member 4
    455545321: "Oana",  # Member 5
    705905107: "correct one"  # Member 6
}

# Initialize attendance tracking dictionary:
# Creates an entry for each member with:
# - present: boolean (initially False)
# - time: string (initially "-")
attendance = {key: {"present": False, "time": "-"} for key in members.keys()}

def get_greeting():
    """Returns time-appropriate greeting based on current hour"""
    # Get current hour (24-hour format) from RTC
    hour = rtc.date_time()[4]

    # Return greeting based on time of day:
    if 5 <= hour < 12:
        return "Good Morning"
    elif 12 <= hour < 18:
        return "Good Afternoon"
    else:
        return "Good Evening"

async def connect_wifi():
    """Connects to WiFi network and returns IP address"""
    # Create WLAN interface in station mode
    wlan = network.WLAN(network.STA_IF)

    # Activate the interface
    wlan.active(True)

    # Check if already connected
    if wlan.isconnected():
        # Print current IP address if connected
        print("Connected! IP Address:", wlan.ifconfig()[0])
        return wlan.ifconfig()[0]  # Return IP address
```

```
    # Attempt to connect to WiFi
    print(f"Connecting to Wi-Fi: {SSID}...")
    wlan.connect(SSID, PASSWORD)

    # Wait for connection (10 second timeout)
    for _ in range(10):
        if wlan.isconnected():  # Check if connection succeeded
            print("Connected! IP Address:", wlan.ifconfig()[0])
            return wlan.ifconfig()[0]  # Return IP on success
        await asyncio.sleep(1)  # Wait 1 second between checks

    # If connection fails after 10 seconds
    print("Failed to connect to Wi-Fi.")
    return None  # Return None on failure

def set_led_color(red_val, green_val, blue_val):
    """Sets RGB LED color based on boolean inputs (0/1)"""
    # Set red LED: 65535 is max duty cycle (on), 0 is off
    red.duty_u16(65535 if red_val else 0)

    # Set green LED
    green.duty_u16(65535 if green_val else 0)

    # Set blue LED
    blue.duty_u16(65535 if blue_val else 0)

def get_time():
    """Returns formatted current time string from RTC"""
    # Get all time components from RTC
    year, month, day, weekday, hour, minute, second = rtc.date_time()

    # Return formatted string: HH:MM:SS DD/MM/YYYY
    return f"{hour:02}:{minute:02}:{second:02}  {day:02}/{month:02}/{year}"

def generate_html():
    """Generates HTML page showing attendance status"""
    # HTML header with embedded CSS styles
    html = """<!DOCTYPE html><html><head><title>Attendance Records</title>
    <style>body{font-family:Arial,sans-serif;text-align:center;}
    table{width:80%;margin:auto;border-collapse:collapse;}
    th,td{padding:10px;border:1px solid black;}th{background-color:#f2f2f2;}
    .present{background-color:lightgreen;}.absent{background-color:lightcoral;}</style>
```

```
    </head><body><h1>Attendance Status</h1><table>
    <tr><th>Name</th><th>Status</th><th>Time</th></tr>"""

    # Add a table row for each member
    for card_id, name in members.items():
        # Determine status text
        status = 'Present' if attendance[card_id]["present"] else 'Absent'
        # Determine CSS class for row coloring
        row_class = "present" if attendance[card_id]["present"] else "absent"
        # Get timestamp
        time = attendance[card_id]["time"]
        # Add table row
        html += f'<tr
class="{row_class}"><td>{name}</td><td>{status}</td><td>{time}</td></tr>'

    # Close HTML tags
    html += "</table></body></html>"
    return html  # Return complete HTML string

async def start_web_server():
    """Starts async web server on port 80"""
    # Connect to WiFi and get IP
    ip = await connect_wifi()
    if ip is None:  # Check if connection failed
        print("Failed to connect to Wi-Fi. Exiting...")
        return  # Exit if no connection

    # Print server address
    print(f"Starting web server at: http://{ip}")

    # Create TCP socket
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    # Bind to all interfaces on port 80
    server.bind(("0.0.0.0", 80))
    # Allow up to 5 queued connections
    server.listen(5)
    # Allow address reuse (helps with quick restarts)
    server.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    # Set non-blocking mode for async operation
    server.setblocking(False)

    # Create poller for socket events
    poller = uselect.poll()
    # Register server socket for input events
```

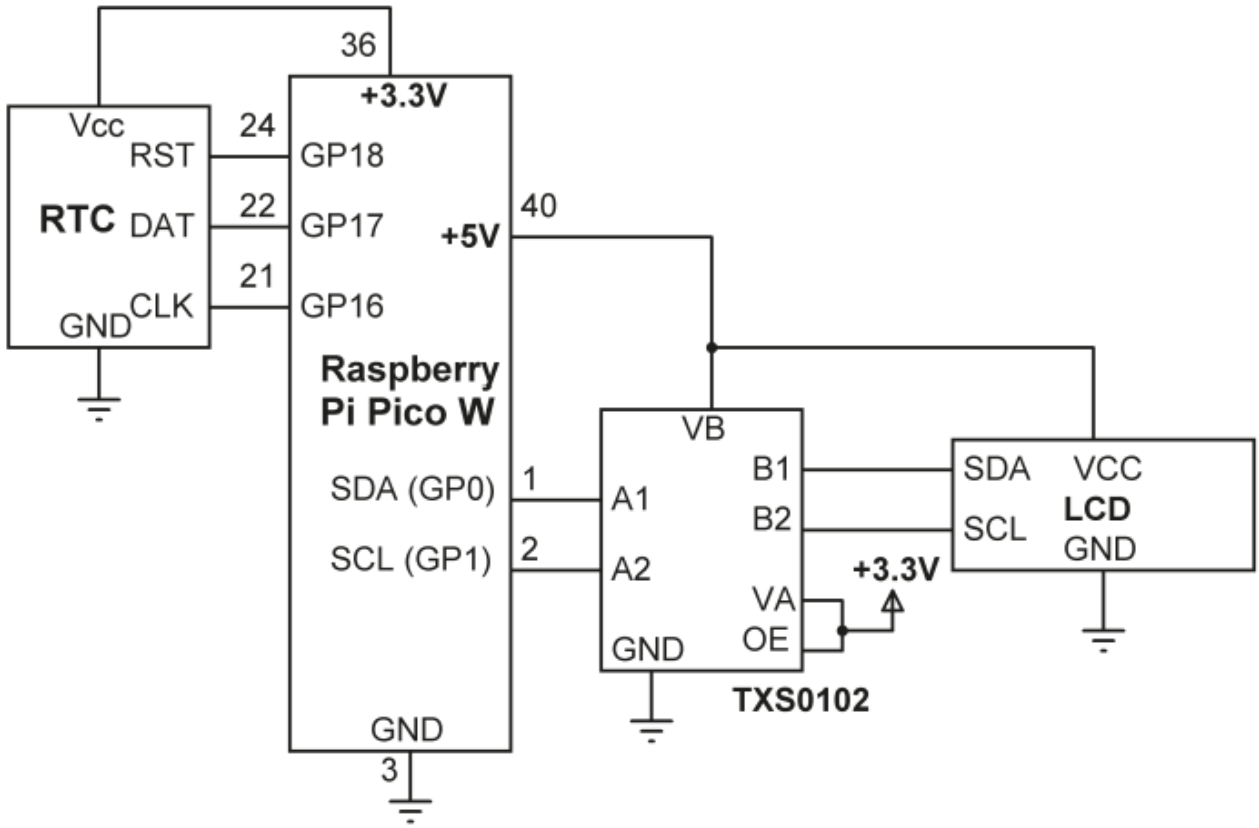
```
    poller.register(server, uselect.POLLIN)

    # Main server loop
    while True:
        try:
            # Wait for events with 1 second timeout
            events = poller.poll(1000)
            # Process all events
            for sock, event in events:
                if event & uselect.POLLIN:  # New connection event
                    # Accept the connection
                    conn, addr = server.accept()
                    # Set non-blocking mode
                    conn.setblocking(False)
                    print(f"Client connected from {addr}")
                    # Handle client request
                    await handle_client(conn)

            # Yield to other tasks
            await asyncio.sleep(0)

        except Exception as e:  # Catch any errors
            print(f"Error in web server: {e}")
            # Small delay before continuing
            await asyncio.sleep(0.1)

async def handle_client(conn):
    """Handles incoming HTTP requests"""
    try:
        # Create poller for this connection
        poller = uselect.poll()
        # Register connection for input events
        poller.register(conn, uselect.POLLIN)

        # Wait for data with 1 second timeout
        events = poller.poll(1000)
        if events:  # If data received
            # Read request (up to 1024 bytes)
            request = conn.recv(1024).decode()
            print("Client request:", request)

            # Create HTTP response
            response = "HTTP/1.1 200 OK\nContent-Type: text/html\n\n" + generate_html()
            # Send response
```

```
            conn.sendall(response.encode())
    except Exception as e:  # Catch any errors
        print(f"Error handling client: {e}")
    finally:
        conn.close()  # Always close connection

async def rfid_task():
    """Main RFID scanning task"""
    print("Starting RFID task...")
    # Clear LCD and show default message
    lcd.clear()
    lcd.putstr("Insert Tag")

    # Main RFID loop
    while True:
        # Initialize RFID reader
        reader.init()
        # Request tag detection
        (status, _) = reader.request(reader.REQIDL)

        if status == reader.OK:  # If tag detected
            # Select the tag and get UID
            (status, uid) = reader.SelectTagSN()
            if status == reader.OK:
                # Convert UID bytes to integer
                card = int.from_bytes(bytes(uid), "little", False)
                # Look up in authorized members
                auth = members.get(card)

                # Prepare LCD display
                lcd.clear()
                lcd.move_to(0, 0)

                if auth is None:  # Unauthorized tag
                    lcd.putstr("Not Authorized")
                    print("Unauthorized card detected!")
                    # Red LED
                    set_led_color(1, 0, 0)
                    # Short beep
                    buzzer.on()
                    await asyncio.sleep(0.2)
                    buzzer.off()
                    await asyncio.sleep(1)
                else:  # Authorized tag
```

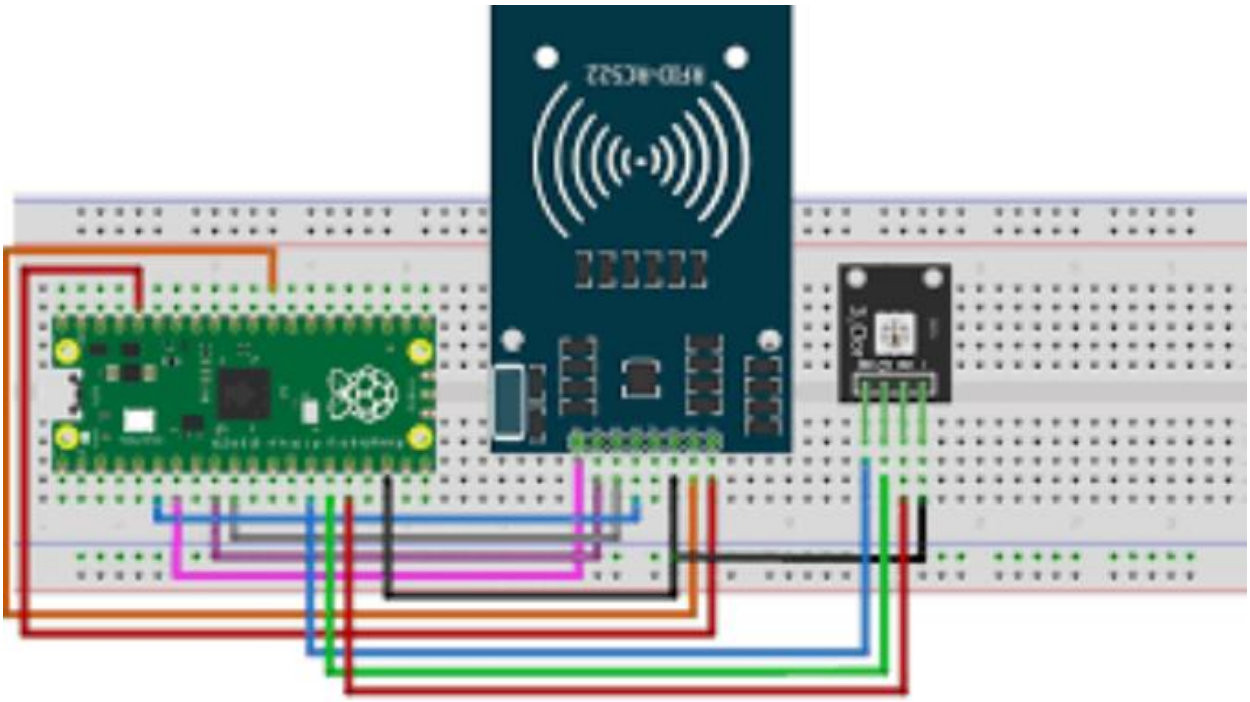
```
                    # Get time-based greeting
                    greeting = get_greeting()
                    # Get current time
                    current_time = get_time()

                    # Display greeting on first line
                    lcd.putstr(f"{greeting}")
                    # Move to second line
                    lcd.move_to(0, 1)
                    # Display name
                    lcd.putstr(f"{auth}")

                    print(f"Access granted: {auth} at {current_time}")

                    # Toggle attendance status
                    attendance[card]["present"] = not attendance[card]["present"]
                    # Update timestamp if present, else clear it
                    attendance[card]["time"] = current_time if attendance[card]["present"]
else "-"

                    # Set LED color: green for present, red for absent
                    set_led_color(0, 1, 0) if attendance[card]["present"] else
set_led_color(1, 0, 0)
                    # Short beep and relay activation
                    buzzer.on()
                    RELAY.on()
                    await asyncio.sleep(0.2)
                    buzzer.off()
                    RELAY.off()

                    # Keep message displayed for 2 seconds
                    await asyncio.sleep(2)
                    # Turn off LED
                    set_led_color(0, 0, 0)

                # Reset to default state
                lcd.clear()
                lcd.putstr("Insert Tag")

        # Small delay between scans
        await asyncio.sleep(0.05)

async def main():
```

```
    """Main async function that runs both tasks"""
    print("Starting main loop...")
    # Run web server and RFID tasks concurrently
    await asyncio.gather(start_web_server(), rfid_task())

# Program entry point
try:
    # Start the async event loop
    asyncio.run(main())
except Exception as e:  # Catch any unhandled exceptions
    print(f"Error in main loop: {e}")
    # Clean up hardware on error
    set_led_color(0, 0, 0)  # Turn off LED
    lcd.clear()  # Clear LCD
    lcd.backlight_off()  # Turn off backlight
```

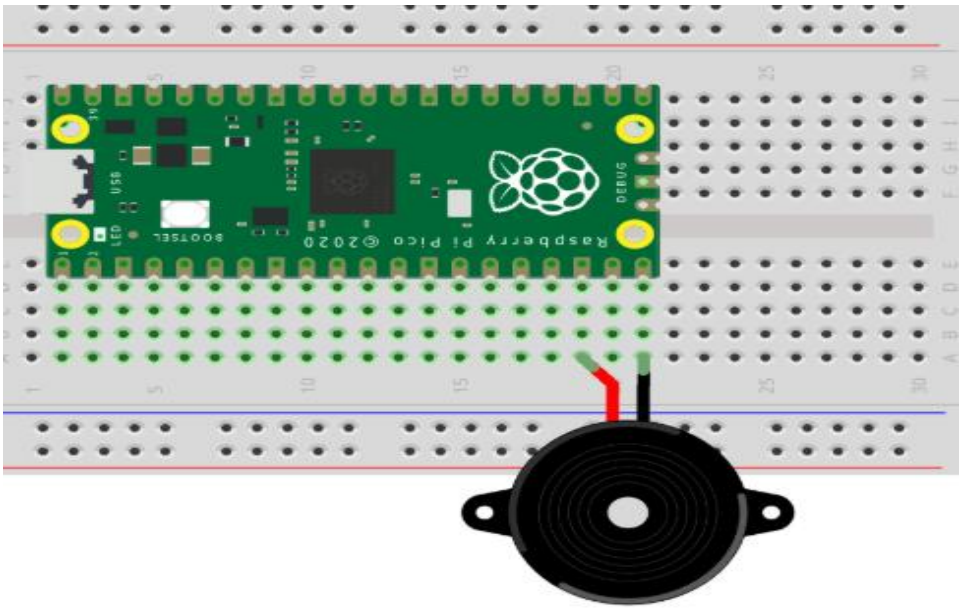
This is the schematic for the RTC and LCD:



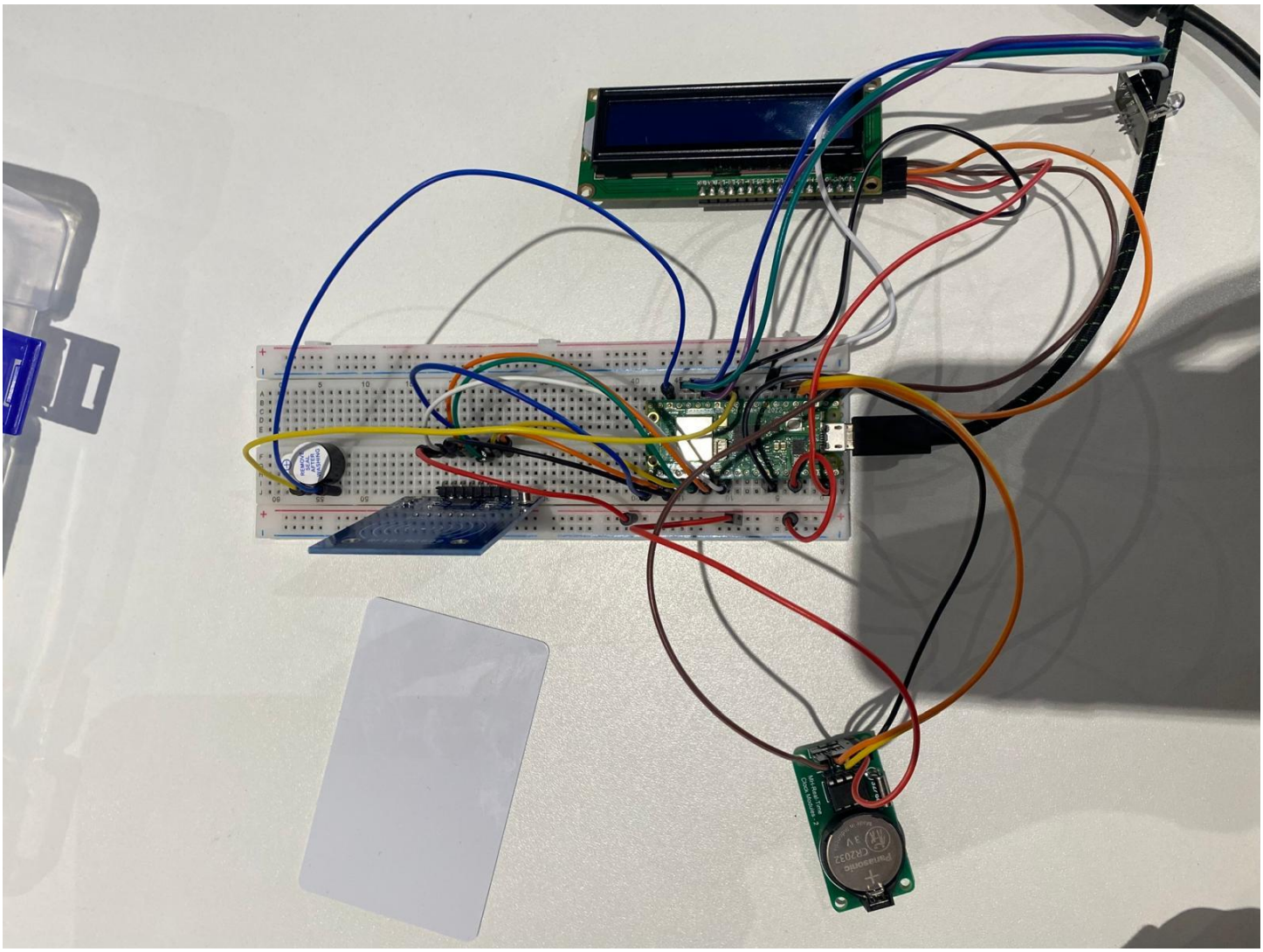
Schematic for RFID and RGB LED:



Schematic for buzzer:



My project:



Self-reflection

For Computer Science, as our first assignment of the week, we had to create our own Scavenger Hunt. I chose to make a short code, that encrypted digits to letters. The letters were in alphabetical order, from A to Z and the digits started from 1. Number zero remained the same, unchanged and also the other characters besides letters.

Besides our Scavenger Hunt assignment, we had to make an OOP project and Vanity Plates, using Regex.

For Embedded Systems we had to create an Attendance project. The Attendance Management System is a digital solution designed to efficiently track and manage the attendance of employees, students, or members of an organization.

Key Components Used:

- RFID Module (RC522) – For scanning employee/student ID cards.
- LCD Display (16x2) – Shows user information and system status.
- Real-Time Clock (RTC - DS3231) – Provides accurate date and time logging.
- Buzzer – Gives audio feedback on successful/failed scans.
- RGB LED – Indicates system status (e.g., green for success, red for error).
- Microcontroller (Raspberry Pi Pico) – Processes data and controls peripherals.
- ID Cards (RFID Tags) – Unique identification for each user.

Professional Skills

6 Thinking Hats:

Matin: Red hat

Andrei: Red hat

Adriana: Blue Hat

Rafael: Black/Blue hat

Diogo: White/Green hat

Alexia: Green/Yellow hat

Roles in the group

Rafael:(Group Leader & Task Manager): Oversees the project, assigns and divides tasks, ensures deadlines are met, and keeps the team organized. Acts as the main point of contact for project progress and decision-making.

Diogo: (Vice Leader & Emergency Support): Acts as the second-in-command, stepping in if Rafael is unavailable. Ensures all tasks are completed on time, taking over last-minute work if needed to keep the project on track.

Adriana: (Documentation & Reporting): Responsible for maintaining clear and detailed documentation of the project, including research, implementation steps, and technical guides. Ensures all work is properly recorded for future reference.

Matin: Barcode Database & API Research: Find out where you can get a barcode database (free or paid), how to integrate it, and what format the data comes in.

Andrei: Hardware & Cost Research: Look into compatible barcode scanners, Raspberry Pi Pico accessories, and pricing for the necessary components.

Alexia: Software & Libraries Research: Find libraries for scanning barcodes, connecting the scanner to the Pi Pico, and displaying data on a website.

The Diamond Model

1. Discover
    a. Overbuying groceries: people purchase unnecessary items, especially when hungry or tired.
    b. Forgetting items: busy individuals and elderly people often forget items they need.
    c. Lack of inventory awareness: confusion about what's already at home.
    d. Food waste: buying too much or not tracking expiration dates.
    e. Financial waste: unnecessary expenses due to redundant purchases.
2. Define
    a. People struggle to effectively manage grocery shopping due to forgetfulness, lack of real-time inventory information, and insufficient tracking of product expiration, resulting in food waste, financial loss, and inconvenience.
3. Execute
    a. Barcode scanning.
    b. Voice-assisted grocery list management.
    c. Inventory tracking with automated notifications.
    d. Shared grocery lists.
    e. AI predictions of needed groceries based on past usage.
4. Deliver
    a. A practical, accessible, and user-friendly mobile app that includes barcode scanning, smart inventory tracking, automated expiry reminders, and shopping list creation. It simplifies grocery management for users of all ages, especially benefiting busy individuals and elderly users.

5 Whys problem-solving framework

Observation: The sensor does not read the expiry date when we scan the barcode

Why do we need to read the expiration date?

Because we have to put it on the website.

Why do we have to put on the website?

Because it must warn the user when the food expires.

Why do we need to warn about the expiration date?

Because we want to reduce the waste.

Why do we want to reduce waste?

Because food waste has a negative impact on the environment and economy.

Why does food waste have a negative impact?

Because it leads to unnecessary resource consumption, higher costs, and environmental harm.

Final Research Question

“How can barcode scanning technology be improved to accurately read expiration dates and help reduce food waste?”

Challenges

1. Barcode Database & Expiry Date Information

Many barcodes do not include expiration dates by default.

Need to find a reliable barcode database or API that provides expiration date information.

2. Hardware Compatibility & Performance

   Ensuring the barcode scanner works smoothly with Raspberry Pi Pico.

   Choosing a scanner that accurately reads various barcode formats, including damaged or faded ones.

3. Software & Library Limitations

   Finding the right libraries to decode barcodes and extract useful information.

   Integrating the barcode scanner with a web-based system for real-time data updates.

Opportunities

1. Reducing Food Waste

   The project directly contributes to sustainability by helping users track expiration dates and reduce waste.

2. Enhancing Inventory Management

   Small businesses and households can efficiently manage their stock and avoid expired products.

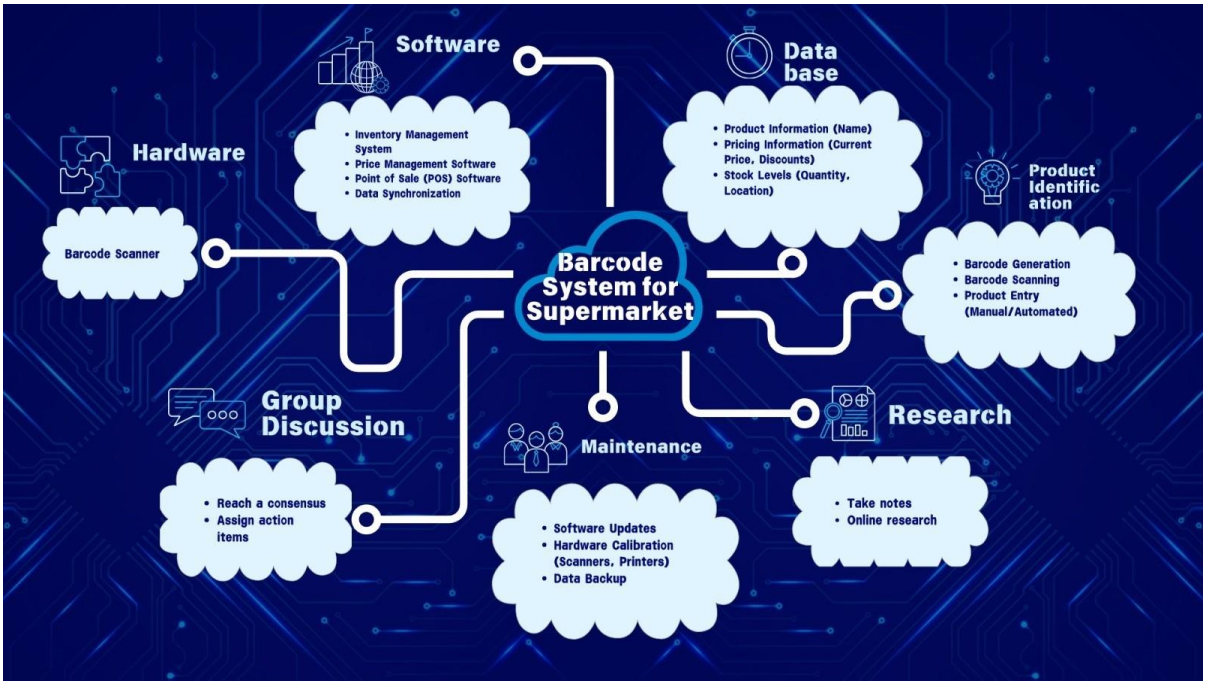
3. Expanding to Other Applications

   The system could be adapted for pharmacies, grocery stores, and warehouses to track perishable items.

4. Learning & Skill Development

   The team gains valuable experience in hardware integration, web development, and problem-solving.

Mind Map for Project Idea



Part of my research

Which Raspberry should we use for our project? Raspberry Pi 4 or Raspberry Pi 5?

Performance:

Pi 4: Quad-core Cortex-A72, 1.5 GHz.

Pi 5: Quad-core Cortex-A76, 2.4 GHz (2-3x faster).

GPU:

Pi 4: VideoCore VI.

Pi 5: VideoCore VII (better for gaming/3D).

RAM:

Pi 4: 2GB/4GB/8GB LPDDR4.

Pi 5: 4GB/8GB LPDDR4X (faster).

Connectivity:

Pi 4: 2x USB 3.0, Gigabit Ethernet, Wi-Fi 5, BT 5.0.

Pi 5: Improved USB 3.0, PoE support, Wi-Fi 5, BT 5.0.

Storage:

Pi 4: MicroSD, USB boot.

Pi 5: MicroSD, PCIe for NVMe SSDs.

Display:

Pi 4: Dual 4K @ 60Hz.

Pi 5: Dual 4K @ 60Hz with HDR.

Power:

Pi 4: USB-C (5V/3A).

Pi 5: USB-C (5V/5A recommended).

Extras:

Pi 5: Onboard RTC, power button, better cooling.

Price:

Pi 4: Starts at $35.

Pi 5: Starts at $60.

Summary:

Pi 5 offers significant upgrades in performance, connectivity, and features, making it ideal for advanced projects, while Pi 4 remains a cost-effective option for basic tasks.

Peer feedback

Kamyab:

- project is not a necessity
- It is a luxury item
- the price matters and whether it's going to constantly check the products in the refrigerator
- how we can keep it updated
- if customer eats an item, does he need to keep updating the scanner or is it going to update on its own

Fabiana:

- for hardware, maybe in the fruit section of the fridge, track by weight how much of a product you have
- determine automatically how much product i have, not having to constantly check it with the scanner

Processing the feedback

I found the feedback from my two colleagues, Fabiana and Kamyab, both from different teams, very helpful. They gave me more ideas, that I discussed with my teammates afterwards in a team meeting.

Fabiana's suggestion about weight-based tracking offers a more precise and automated solution compared to manual piece-counting, especially for perishable items like fruits and vegetables. This approach could significantly improve accuracy while reducing user effort.

Secondly, I mostly agree with Kamyab regarding the fact that our product is a luxury item. For some people, it feels like a necessity to have a list of the products they have in their own fridge at any given moment and check the expiry date. On the other hand, many people won't stop using the classical method of writing their own shopping list on paper.

These two important opinions sparked different question marks. My team will try to find solutions so we can improve our project and do the best we can to give users a qualitative food-tracking app.

P3. Computer Science

Name	Student nr.	Group	Year
ALEXIA MIT	5587832		

The student is responsible for safe keeping of this card

Assignments

Wk	Subject	Date	Signature
3.1	Smart Café Helper	12 MAR 2025	
	Tip Calculator + Video	12 MAR 2025	
	Tip Calculator peer feedback	12 MAR 2025	
3.2	Vanity Plates goes Flask	12 MAR 2025	
3.3	Survival Simulator	12 MAR 2025	
3.4	Testing my twttr	12 MAR 2025	
	Back to the Bank	12 MAR 2025	
	Re-requesting a Vanity Plate with pytest: raise	12 MAR 2025	
	Nutrition Facts unittest	31 MAR 2025	
	Presentatie	12 MAR 2025	
3.5	Scavenger hunt	31 MAR 2025	
	CS50 Shirt	2 APR 2025	
3.6	Scavenger hunt Pt II	31 MAR 2025 2 APR 2025	
3.7	Product goes OOP		2 APR 2025
	CS50 Shirtificate	2 APR 2025	
T3	**Assessment**		

Feedback:

Please every week. Take a look maria goes flask.

This card is proof that the above assignments are properly completed and should be:

- signed off by the appropriate teacher(s) during the guided ateliers
- a digital attachment of your portfolio and
- physically handed over at the beginning of the assessment

NHL STENDEN university of applied sciences

P3 Embedded Systems

Name	Student nr.	Group	Year
ALEXIA MÎȚ	5[illegible]832		

The student is responsible for safe keeping of this card

Assignments

Wk	Ch	Subject	Date	Signature
3.1	1/3	Blink with external reset	31 FEB 2025	M.Bara
	2/3	Mario Pico	13 FEB 2025	M.Bara
3.2	4/5	7-segments voltmeter *(0...10V)*	25 MAR 2025	M.Bara
3.3	5	Realtime clock and temperature on LCD-display	1 APR 2025	M.Bara
3.4	-	Dino cheater *(HID)*	26 MAR 2025	M.Bara
		Analog joystick *(HID)*	26 MAR 2025	M.Bara
3.5	11	Wifi scanner	31 MAR 2025	M.Bara
		Controlling stuff *(Webserver)*	31 MAR 2025	M.Bara
3.6	12	Attendance *(RFID, RTC, web and LCD-display)*	2 APR 2025	M.Bara
3.7				
3.8		**Portfolio**		
T3		**Assessment**		

Feedback:

This card is proof that the above assignments are properly completed and should be:

- signed off by the appropriate teacher(s) during the guided ateliers
- a digital attachment of your portfolio and
- physically handed over at the beginning of the assessment