

LC CS Coursework Report – Mar 2024

Preface – Reflections on the Project (Jun 2025)

Introduction

This project was developed in March 2024 as part of my final Leaving Certificate Computer Science coursework. This reflection and the original report are provided in this document. Now studying information management at university, I wanted to revisit the project with a critical lens, exploring its strengths and limitations.

Reflection

Looking back on this project — completed as part of my Leaving Certificate Computer Science coursework — I now view it as an important first leap into data analytics and data science. It was a product of a 12-week sprint, framed by specific guidelines and requirements, and driven by an interest in understanding sleep patterns through data.

Since then, through university modules in business intelligence, market research, and practical data analytics learning, I've developed a more refined understanding of data quality, modelling, and design. While I'm still proud of this project, I want to highlight a few key limitations and lessons I would take into a revised version.

Simulating 100 Nights of Sleep from a Single Night: A Flawed Foundation

At the time, I collected data from a micro:bit for a single night and used this data to simulate 100 nights of sleep using statistical modelling (mean and standard deviation). This allowed me to generate a larger dataset to experiment with OOP and basic analytics.

In hindsight, this is a flawed approach. Modelling long-term trends from a single night of data introduces serious risks. Sleep is highly variable — affected by environment, stress, diet, routine, and more — and modern sleep tech like the Apple Watch or Oura Ring accounts for this by **continuously collecting data and aggregating results over time**. These devices often require a week or more of baseline data to generate insights.

As I've learned: "Garbage in, garbage out." No matter how sophisticated your model is, if your input data is flawed, limited or unrepresentative, the insights produced are unlikely to be reliable.

Poor Survey Question Design & Data Visualisation

Surveying was a requirement of the LC project, and I gathered responses from 34 students — a solid sample size for high-school level research, considering a sample of 30 is the minimum sample size for which the Central Limit Theorem is valid.

However:

- The question design was weak.
 - For example, one question asked: “Does excessive light affect your sleep?” — with a response scale from “Strongly Agree” to “Strongly Disagree.”
 - This phrasing lacks clarity and neutrality.
 - A better phrasing would have been: “To what extent do you agree with the statement: Excessive light affects your ability to fall and stay asleep?”
- Inappropriate visualisations were used.
 - I used pie charts to display Likert scale, ordinal data, which obscures the progression of agreement.
 - A bar chart would have been clearer and more appropriate, especially for categorical and ordinal responses — something I now appreciate from both my Business Intelligence module and Scott Berinato’s “Good Charts”.
- Data presentation could have been cleaner.
 - Exporting survey results to Excel and using better formatting (rather than screenshotting from Google Forms) would’ve allowed me to present findings more formally and professionally.

Data Validation: From Rule-Based Filters to Smarter Preparation

One of the key stages of the data analysis lifecycle is **preparation**, where raw data is cleaned, validated, and made analysis-ready. In Google’s Data Analytics Professional Certificate (which I’ve since completed), this stage is strongly emphasised.

In my original project, I used a simple rule-based approach: removing empty strings from the CSV data and stripping out sound values over 80 dB. At the time, this was reasonable, and it showed early awareness that outliers and anomalies could skew results. For example, I recognised that sound levels over 80 dB likely reflected sensor glitches rather than meaningful sleep disturbances.

However, by modern standards, this method lacks robustness. More sophisticated approaches might include:

- Outlier detection using **interquartile range (IQR)** or **Z-scores**
- Smarter thresholds that account for **distribution context**, not just absolute limits
- Checks for **multi-sensor consistency** (e.g., missing values across columns in a time slice)

This critique doesn’t take away from what I achieved — it simply highlights how much more there is to learn about preparing data responsibly and confidently. Still, even then, my choice to flag 80 db readings as suspect was a decent step in the right direction.

Additional Critiques

- Graphical outputs were too basic.
 - While I used matplotlib to visualise data trends, the outputs lacked visual clarity and storytelling.

- Modern libraries like seaborn or plotly could have helped produce more engaging, informative charts with less code and better aesthetics.
- No feedback loop from users.
 - While the system offers advice, it doesn't ask users how they feel about their sleep.
 - Adding some sort of feedback loop, even a basic prompt like "did you feel rested?" – could have introduced a human layer to the analysis, and helped build the framework for personalised insights in a future iteration.
- Simulated data was static, not evolving.
 - Each of the 100 simulated nights was generated independently, using static distributions.
 - A more realistic dataset could've simulated gradual changes over time, such as improving or worsening sleep quality, to mimic real-life variability better.

Note on Code Differences

Please note: the Python code referenced in this report reflects the original version submitted in 2024. Since then, all .py files have been fully refactored for clarity, performance, and professionalism.

Key improvements include:

- Functions are wrapped in main() blocks to improve modularity and reusability.
- Removed noisy initial sensor readings, enhancing data quality checks.
- Added docstrings, improved inline comments, and more intuitive logic.
- Adopted Python f-strings for cleaner string formatting.
- Code now conforms to PEP 8 standards (naming conventions, structure, spacing)
- File structure reorganised for clarity (e.g. src/ folder, relative paths)

This updated and improved source code is available in the GitHub repository linked alongside this report.

Final Thoughts

This project was my first real attempt at using data to solve a personal, real-world challenge. Despite its flaws, I still stand by it as a meaningful milestone in my development. Completing it under a 12-week timeframe, balancing hardware integration, data collection, analysis, and visualisation — all at high school-level — was a challenge I'm proud to have taken on.

Looking back now, I can see where this project could be stronger: better data validation, more thoughtful survey design, smarter simulation logic, and more effective visual storytelling.

But most importantly, this project **sparked my interest in data analytics**. It taught me how to think like an analyst: to ask better questions, interrogate assumptions, and extract insight from raw data.

If I were to revisit this project today, I would focus more on data integrity, iterative feedback loops, and statistical reliability — but I wouldn't erase what I originally built. It's a solid foundation and marks the beginning of something bigger.

1 - Meeting the Brief

Video Link: <https://www.youtube.com/watch?v=a6n9EpFGZ3A>

Transcript:

I've developed a micro:bit to meet the basic requirements to track sleeping habits using digital inputs like light, sound, temperature, and movement, as well as analog inputs like the time of each measurement. I use two micro:bits — one's a sender and one's a receiver — and they're linked together using the radio functions, as shown here on the screen. We could vary the temperature, light, and sound, and the data is recorded onto the receiver micro:bit's flash drive.

So when this is done in real life, we can export data that's stored on the micro:bit's flash drive onto a CSV file with columns for time, light, sound, temperature, and movement.

In order to clean up and validate this data, we go to the data validation file to meet basic requirement two, which is to validate and store the data gathered from the embedded system. I've used the CSV module and populated the times and each of the lists for light, sound, temperature, and movement. As well as that, I've also changed numbers to integers and floats, and removed the empty strings present in the CSV file. I've also done a helper function to remove invalid sound readings over 80 dB and subtracted 3° from all readings of the temperature monitor.

Moving on to the data analysis file, I've imported the light, sound, temperature, and movement dictionaries. I've also imported NumPy and the Matplotlib library. I run this file here — we should see that running this file gives us the statistics and a graph for each of light, temperature, sound, movement, and time. The data is printed to the terminal here. As well as that, we can also see the graphs shown by Matplotlib.

The functions that are shown in this file are important because we can use them later to meet the advanced requirements.

In the advanced requirements, I have one file that meets all the requirements set out in the advanced section. I created a personal dataset that's generated manually using the data from the analysis file that returns the mean and standard deviation of each of the light, sound, temperature, and movement dictionaries. I've used object-oriented programming to create 100 random night objects with random values for each of the minutes of sleep, as well as the light, sound, temperature, and movement values for each minute of sleep.

So running this file here, we should see the results — the sleep analysis for night one — and the information is also printed to the file. We've also presented the data on a graph, which meets

advanced requirement three. So, night one, night two, and night three — this is all randomly generated data.

We're also prompted to enter our name so that we can address the user for the "What if" questions, which are: "Am I getting enough sleep?" and "Is my sleep environment good enough for quality sleep?"

So, "What is your name?" — we'll put in "Test" — and 8 hours. We can see here if the user has met their recommended sleep duration. As well as that, if they don't meet their sleep requirements, we've also printed a message to consider going to bed earlier to improve sleep quality.

For question two, "Is my sleep environment good enough?", we can ask the user which night they would want to analyse. So, for example, we analyse night one. We're also using the functions that we've imported earlier from the data analysis file in order to show a graph of their light, sound, temperature, and movement over time, as well as the information printed to the file. And we're also given the choice to analyse another night. Otherwise, we can end the program.

2 - Investigation

My computer science project focuses on the tracking and analysis of sleeping patterns in teenagers and young adults. The benefits of good sleep are multifold. According to Mental Health Ireland (1), getting between 7-9 hours of sleep helps us to:

- Mind our mental and physical health
- Cope better and regulate your emotions
- Concentrate and motivate yourselves.

As a Leaving Certificate student, I understand the benefits of a good sleep schedule to maximise academic and personal success. However, I sometimes find difficulty in sticking to a regular schedule, due to problems like digital distractions, exam stress, and mental health issues.

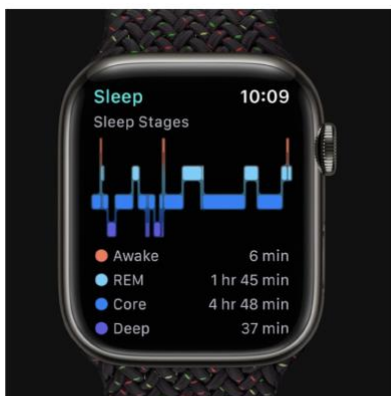
Some factors also affect my ability to fall asleep, such as excessive light and sound as well as waking up in the middle of the night. These could lead to feeling unrested on waking up, affecting my performance in school and my ability to concentrate throughout the day.

I carried out some research on some existing solutions to track and improve sleep. All existing solutions track sleep in different categories, such as REM, core and deep sleep, presenting insights on the user's sleep patterns, as well as showing changing sleep trends.

Smart watches like the Apple Watch Series 9 (2) are excellent solutions as they offer a suite of health tracking features, such as fitness, sleep, and mindfulness. However, a drawback would be that due to its short battery life, the watch would need to be charged at night.

The Oura Smart Ring (3) is also able to measure a wide range of health metrics, including sleep. However, potential clients might be turned away by its high price as well as the need to purchase a subscription.

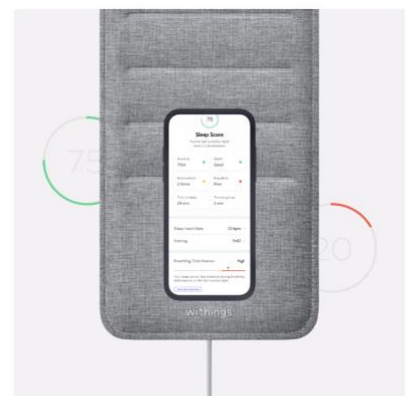
The Withings Under-Mattress Sleep Tracker (4) is a great option for those who don't want to be distracted by a smart device at night, however, if you lie on your bed without sleeping, the tracker could record inaccurate data. It also solely focuses on tracking and analysing sleep.



Apple Watch Series 9



Oura Ring

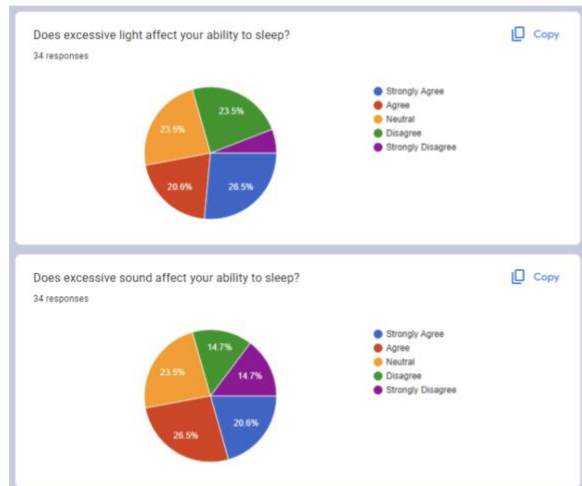


Withings Sleep Analyzer

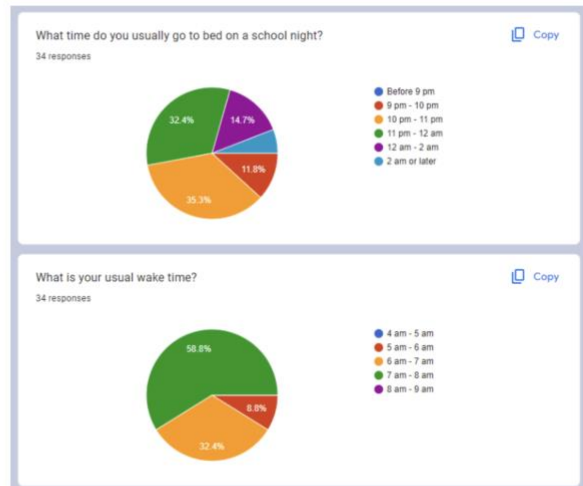
As part of my investigation, I carried out a survey to analyse sleeping trends of students in my school. From the results that I've collected, a large portion of the students report that they strongly agree with the statement that excessive light and sound affects their sleep.

Also relevant to my survey is the amount of sleep that students get. From the results in the survey, however, a significant proportion of students go to sleep between 11 pm - 12 am. This is an issue that

users have that can be fixed by my device, which would encourage users to go to bed at an earlier time.



Survey results: does excessive light/sound affect sleep?



Survey results: how long do participants sleep for each night?

After carrying out research on present solutions for a sleep tracker and analysis app, as well as conducting research on the student body, I feel that this project is worth looking into.

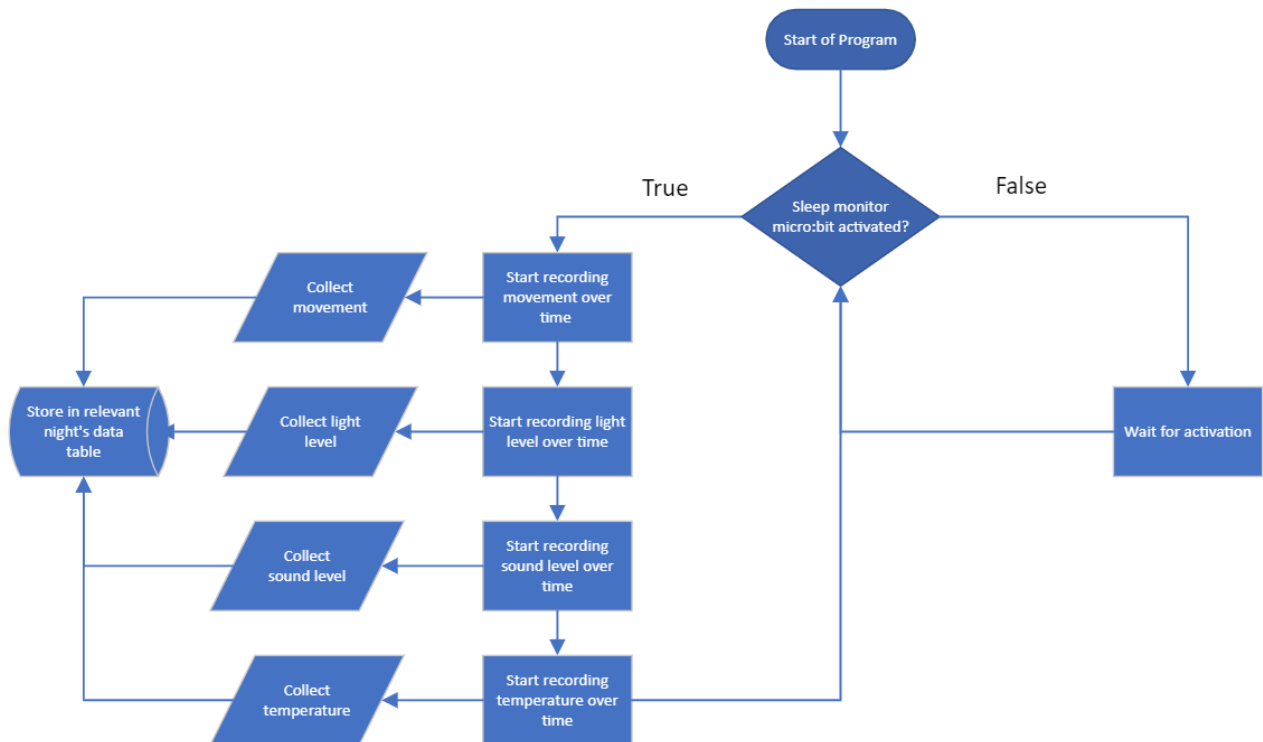
I've concluded that my end users, secondary school students, are looking for a device that tracks their sleep durations and suggests how they can improve their sleep environment.

3 - Design

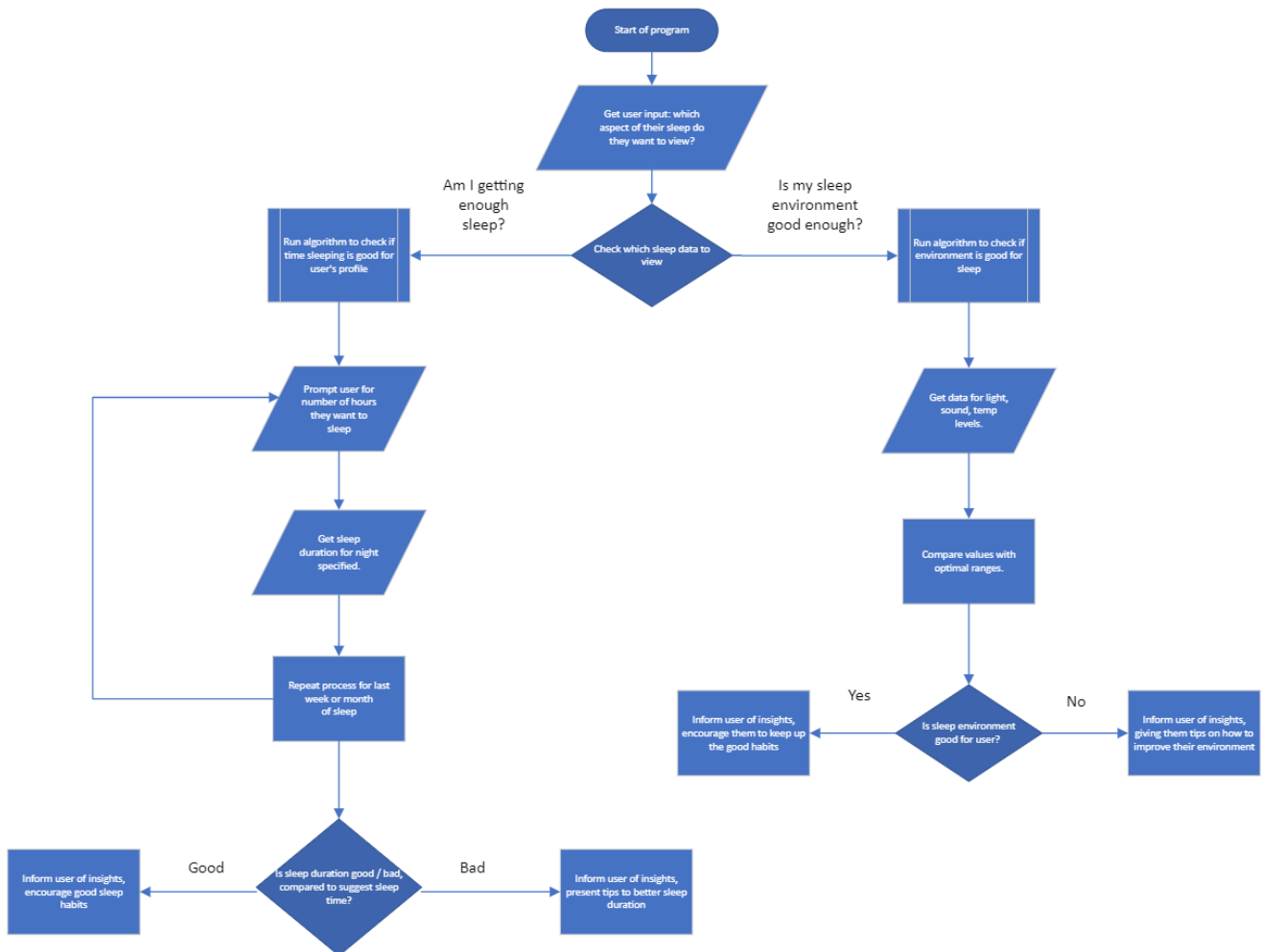
I plan on dividing the project into two distinct stages.

1. Using a micro:bit to collect sleep data, and exporting this data to a computer
2. Using Python and relevant data analysis libraries to analyse this sleep data, discovering and presenting insights.

Below are two flowcharts presenting the planned architecture for my project.



The flowchart for the micro:bit program.



The flowchart for the Python program.

Meeting the basic requirements

The micro:bit will feature the code used to program the embedded system, meeting the requirements of basic requirement 1. I plan on using two micro:bits together.

One micro:bit, labelled “sender”, will collect data with its light, sound, temperature and movement monitors. It will transfer this data using the Radio functions to a micro:bit labelled “receiver”, which will store the data using the Data Logger functions.

By connecting the “receiver” micro:bit to a computer, I plan on exporting the data stored on the micro:bit’s flash memory to a computer.

Here, I aim to download the data as a CSV file, and to use Python to validate and store the data in accordance with basic requirement 2. This involves matching time values with their respective readings and converting values from strings to integers or floating-point numbers.

microbit_data_5

File Home Insert Share Page Layout Formulas Data Review View Automate

Calibri (Body) 11 A[^] A_v **B** *I* U ~~ab~~ D

H5392 X ✓ fx

	A	B	C	D	E	F	G	H	I	J
1	Time (secs)	light	sound	temp	movement					
2	2.64	0								
3	2.65		15							
4	2.66			23						
5	2.67				829					
6	2.78	139								
7	2.94		15							
8	3.04			23						
9	3.14				1029					
10	3.26	137								
11	3.42		112							
12	3.52			23						
13	3.62				1074					
14	3.74	108								
15	3.91		33							

With this validated data, I'll be able to analyse each of the light, sound, temperature and movement metrics with respect to time, seeing how they change and vary over time.

Dictionaries can come in handy here. By creating key:value pairs between time and metric values, it becomes easier to infer information from the dataset, or plot graphs to view this information in a visual manner, meeting basic requirement 3.

Meeting the advanced requirements

I plan on building my own dataset based on the information collected from the micro:bit. This involves creating 100 "simulated" nights of data with very similar information collected in real life.

Since the dataset will contain similar values of light, sound, temperature and movement, as well as containing sleep durations, the dataset will "contain multiple descriptive features of wellbeing" and will be able to answer a variety of 'what if' questions. (advanced requirement 1)

Using this dataset of randomly simulated night data, I plan on answering two 'what if' questions: (advanced requirement 2)

- Am I getting enough sleep?
 - I plan on finding the length of sleep for each night in the dataset.
 - Users will be able to input their desired hours of sleep and see if their sleep duration meets their requirements.
 - Users can see how their sleep duration changes over a time interval, like a week or a month of sleep.
- Is my sleep environment good enough?

- Users can see the statistics of the data collected for their sleep, which includes metrics on light, sound and temperature, the main factors for sleeping environments.
- Users will be able to see the “optimal” range for each metric, as well as when they have exceeded this range.
- The program will encourage users and give them tips on how to improve their sleeping environment.

By plotting graphs to display the information for each ‘what if’ question, we can both provide users with insights on their sleeping habits and meet advanced requirement 3.

4 – Create

Over the course of twelve weeks, I have carried out an investigation into the project brief, chose to carry out my project on sleep analysis, and developed a program to analyse user's sleep patterns using micro:bit and Python.

The progress log for my project is shown below.

Week	Description
Week 1	Discussed project brief and criteria.
Week 2	Researched ideas for the project
Week 3	Decided on a plan for a sleep tracker and analysis app. Reviewed plan with teacher. Completed Coursework Report Section 2 - Investigation.
Week 4	Collected data from end users through a survey. Designed a flowchart for the project. Completed Coursework Report Section 3 - Plan and Design.
Week 5	Completed micro:bit code. (Basic 1) Experimented with data validation functions on Python.
Week 6	Processed and validated data from CSV file using NumPy arrays. (Basic 2) Analysed and displayed data using matplotlib plotting functions. (Basic 3)
Week 7	No work done due to Mocks.
Week 8	No work done due to Mocks.
Week 9	Encounter problem with validation program (Basic 2), improved validation with a new algorithm employing dictionaries. Brought home a micro:bit to collect realistic sleep data. Used this data for analysis and for Advanced requirements.
Week 10	Used classes, objects and NumPy normal distributions to build a dataset of randomly generated sleep data based on mean and standard deviation of realistic data collected from Week 9. (Advanced 1) Used dataset to complete What If Q1: "Am I getting enough sleep?" Completed What If Q2 ("Is my sleeping environment good enough?") by using analysis code from Basic 3. (Advanced 2)
Week 11	Displayed results from What If Questions to the user via a matplotlib graph. (Advanced 3) Reviewed and refactored code to check for bugs. Carried out unit testing.
Week 12	Completed Coursework Reports Sections 4 and 5 - Create and Evaluation Completed Coursework Report, recorded video for Section 1. Created a website, and copied over the relevant sections to it. Organised work into correct file structure for submission.

Unit testing

I used unit testing to test the inputs for the three input functions in advanced_requirements.py

For example, the user is prompted to enter an integer value between 1 to 24 when asked the number of hours of sleep they want to get.

As the input only accepts integer values between 1 to 24, I wrote code to repeatedly prompt the user to enter their desired hours of sleep in the correct format.

A table of tests for one input is shown below.

```
advanced_requirements.py ×
153
154 # -----
155
156 # TASK - TO MEET ADVANCED REQUIREMENT 2
157 # Each 'what if' question must use a minimum of three validated parameters (using at least two d
158 # and, based on the information provided, offer the user insights in relation to some aspect of
159
160
161
162 # WHAT IF QUESTION 1: "Am I getting enough sleep?"
163
164 username = input("What is your name? ")
165
166 target_hours = input("Hi {a}! How many hours of sleep do you want to get every night (1-24)? ".f
167 if not target_hours.isdigit() or int(target_hours) > 24 or int(target_hours) < 1:
168     while not target_hours.isdigit() or int(target_hours) > 24 or int(target_hours) < 1:
169         target_hours = input("Invalid entry. Please enter an integer between 1-24. ")
170
171 target_hours = int(target_hours) # convert to integer
172
173 # get last night's sleep - see if it meets the target
174 # get last week's sleep duration average and check if it exceeds the target
175 # get last month's sleep duration average, check if it exceeds the target
176
177 def enough_sleep(name, target_hours, nights_data):
178     print("\n----- What If Q1: Am I getting enough sleep? -----\\n")
179
```

Shell ×

```
On night 3, you slept 8 hours and 8 minute(s).
Mean light level: 7.81 lx. Standard deviation of light level: 5.34 lx.
Mean sound level: 37.1 dB. Standard deviation of sound level: 21.35 dB.
Mean temperature: 22.0 °C. Standard deviation of temperature: 2.11 °C.
Mean movement level: 1048.23 mg. Standard deviation of movement level: 50.65 mg.

What is your name? user
Hi User! How many hours of sleep do you want to get every night (1-24)? 0.001
Invalid entry. Please enter an integer between 1-24. 1000000
Invalid entry. Please enter an integer between 1-24. 0
Invalid entry. Please enter an integer between 1-24. 7.5
Invalid entry. Please enter an integer between 1-24. eight
Invalid entry. Please enter an integer between 1-24. 8

----- What If Q1: Am I getting enough sleep? -----

Hi User. Lets see if you're getting enough sleep.
```

	A	B	C	D	E	F	G
1	Test ID	Description	Test data	Expected result	Actual result	Pass (Y/N)	
2	1	To test an integer value. (1-24)	8	target_hours = 8	target_hours = 8	Y	
3	2	To test a small integer value. (1-24)	1	target_hours = 1	target_hours = 1	Y	
4	3	To test a large integer value. (1-24)	24	target_hours = 24	target_hours = 24	Y	
5	4	To test a very small integer value.	0.001	"Invalid entry..."	"Invalid entry..."	Y	
6	5	To test a very large integer value.	1000000	"Invalid entry..."	"Invalid entry..."	Y	
7	6	To test the value of 0.	0	"Invalid entry..."	"Invalid entry..."	Y	
8	7	To test a floating point number.	7.5	"Invalid entry..."	"Invalid entry..."	Y	
9	8	To test a string.	"eight"	"Invalid entry..."	"Invalid entry..."	Y	
10							

Overcoming a problem

One of the main problems that I encountered during the implementation was the validation of data stored on the “receiver” micro:bit’s flash drive.

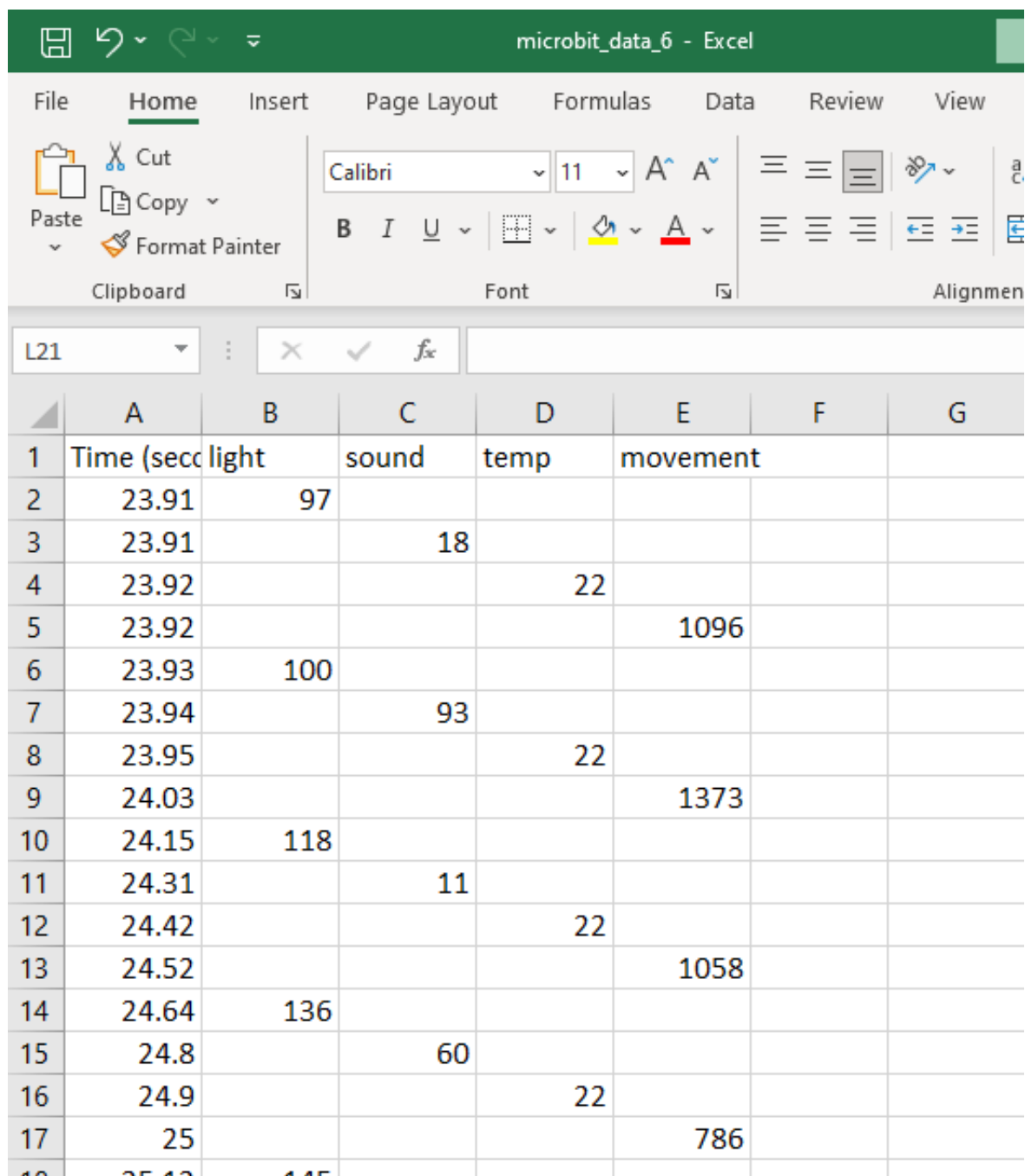
While collecting data, the micro:bit would usually collect data from the micro:bit’s sensors with relation to time. Each metric is usually recorded once for every four time values, so a “staircase” pattern can result as shown here.

microbit_data_6 - Excel							
File	Home	Insert	Page Layout	Formulas	Data	Review	View
<div> <div> <div>Cut</div> <div>Copy</div> <div>Paste</div> <div>Format Painter</div> </div> <div> <div>Calibri</div> <div>11</div> <div>A^A</div> <div>B</div> <div>I</div> <div>U</div> <div></div> <div></div> <div></div> </div> <div> <div>Clipboard</div> <div>Font</div> <div>Alignmen</div> </div> </div>							
L21							
	A	B	C	D	E	F	G
1	Time (secs)	light	sound	temp	movement		
2	23.91	97					
3	23.91		18				
4	23.92			22			
5	23.92				1096		
6	23.93	100					
7	23.94		93				
8	23.95			22			
9	24.03				1373		
10	24.15	118					
11	24.31		11				
12	24.42			22			
13	24.52				1058		
14	24.64	136					
15	24.8		60				
16	24.9			22			
17	25				786		
18	25.12	145					

Overcoming a problem

One of the main problems that I encountered during the implementation was the validation of data stored on the “receiver” micro:bit’s flash drive.

While collecting data, the micro:bit would usually collect data from the micro:bit’s sensors with relation to time. Each metric is usually recorded once for every four time values, so a “staircase” pattern can result as shown here.



	A	B	C	D	E	F	G
1	Time (secs)	light	sound	temp	movement		
2	23.91	97					
3	23.91		18				
4	23.92			22			
5	23.92				1096		
6	23.93	100					
7	23.94		93				
8	23.95			22			
9	24.03				1373		
10	24.15	118					
11	24.31		11				
12	24.42			22			
13	24.52				1058		
14	24.64	136					
15	24.8		60				
16	24.9			22			
17	25				786		
18	25.12	145					

Originally to validate the data I planned on using a for loop to pair time values with their recorded values. However, looking further down the database I found that the micro:bit’s recording didn’t follow this pattern all the time. This may have been due to the “sender” moving outside the range of the “receiver”.

microbit_data_6 - Excel							
File Home Insert Page Layout Formulas Data Review View							
Clipboard		Font		Alignment			
L21							
	A	B	C	D	E	F	G
2302	318.97		37				
2303	319.07			23			
2304	319.17				1087		
2305	319.46		30				
2306	319.56			23			
2307	319.66				1059		
2308	319.78	0					
2309	320.04			23			
2310	320.14				1070		
2311	320.52			23			
2312	321.01			23			
2313	321.49			23			
2314	321.59				1075		
2315	321.71	0					
2316	321.87		7				
2317	321.98			23			
2318	322.08				1059		
2319	322.2	0					

To overcome this issue, I used the csv module to read in the data from the CSV file. I transferred the values in each column to their respective lists - "times", "light", "sound", "temp", and "movement".

I then used dictionaries to pair the "times" list with each of the other lists. Then, iterating through each list, I removed all key:value pairs that have a value of an empty string.

Further validation, like converting strings to integers and floating point numbers, and removing invalid readings, ensured that the data in the CSV file is suitable for analysis.

data_validation_v2.py ×

```
1 # TASK - TO MEET BASIC REQUIREMENT 2 - Validate and store the data gathered from the embedded sy
2 import csv
3
4 # creates empty lists
5 times = []
6 light, sound, temp, movement = [], [], [], []
7
8 # reading from the csv file
9 with open("microbit_data_6.csv", newline='') as data_csv:
10     csv_reader = csv.DictReader(data_csv, delimiter=",")
11
12     # populating times and logger lists
13     for row in csv_reader:
14         times.append(row['Time (seconds)'])
15         light.append(row['light'])
16         sound.append(row['sound'])
17         temp.append(row['temp'])
18         movement.append(row['movement'])
19
20 def int_list_exclude_empty(input_list): # changes values to integers if they arent empty strings
21     return_list = []
22     for val in input_list:
23         if val == '':
24             return_list.append(val) # we will keep the empty strings, then filter the dictionarie
25         else:
26             return_list.append(int(val)) # convert all numbers to integers
27     return return_list
28
29 def remove_pairs(input_dict): # helper function to remove pairs with values that are ''
30     return_dict = {}
31     for key, value in input_dict.items():
32         if value != '':
33             return_dict[key] = value
34     return return_dict
35
```

data_validation_v2.py ×

```
31     for key, value in input_dict.items():
32         if value != '':
33             return_dict[key] = value
34     return return_dict
35
36 def filter_sound_readings(input_dict): # helper function to remove invalid sound readings over 8
37     return_dict = {}
38     for key, value in input_dict.items():
39         if value <= 80: # only include sound readings of over 80 dB, as anything higher is due t
40             return_dict[key] = value
41     return return_dict
42
43
44 times = [float(time) for time in times] # convert string values to float
45 light = int_list_exclude_empty(light)
46 sound = int_list_exclude_empty(sound)
47 temp = int_list_exclude_empty(temp)
48 movement = int_list_exclude_empty(movement)
49
50 # making dictionaries, pairing times with values
51 light_dict = dict(zip(times, light))
52 sound_dict = dict(zip(times, sound))
53 temp_dict = dict(zip(times, temp))
54 movement_dict = dict(zip(times, movement))
55
56 # remove any pairs with a value of a ''
57 light_dict = remove_pairs(light_dict)
58 sound_dict = remove_pairs(sound_dict)
59 temp_dict = remove_pairs(temp_dict)
60 movement_dict = remove_pairs(movement_dict)
61
62 # remove three degrees from all readings due to inaccuracy of thermometer
63 temp_dict = dict(zip(times, [temp - 3 for temp in temp_dict.values()]))
64
65 # remove invalid readings in sound_dict
66 sound_dict = filter_sound_readings(sound_dict)
```

Data analysis - an important piece of code

An important piece of code was the function `analyse_data()` in `data_analysis_v2.py`, which analysed the data in the CSV file. A section of the function is shown below:

```
advanced_requirements.py - data_analysis_v2.py
21
22 def analyse_data(dictionary, measuring, time_interval = "mins"): # inputs dictionary and what is being measured
23     arr = np.array(list(dictionary.values()))
24
25     # central tendency
26     mean_val = np.mean(arr) # mean value, affected by outliers
27     median_val = np.median(arr) # median value, not affected by outliers
28
29     # measure of spread
30     standard_deviation = np.std(arr) # standard deviation
31     first_quarter = np.percentile(arr, 25) # 1/4 of values
32     third_quarter = np.percentile(arr, 75) # 3/4 of values
33     interquartile_range = third_quarter - first_quarter # IQR of values
34
35     # outliers
36     min_value = np.min(arr) # min value
37     max_value = np.max(arr) # max value
38
39     if measuring == "light":
40         unit = "lux"
41         print("\nLet's see if the light levels in your sleeping environment affect your sleep.\n")
42
43         print("Mean is {mean} {unit}, median is {median} {unit}.".format(mean = round(mean_val, 3), median = round(median_val, 3), unit = unit))
44         print("Interquartile range is {iqr} {unit}, standard deviation is {std} {unit}.".format(iqr = round(interquartile_range, 3), std = round(standard_deviation, 3), unit = unit))
45         print("Minimum value is {mini} {unit}, maximum value is {maxi} {unit}.".format(mini = round(min_value, 3), maxi = round(max_value, 3), unit = unit))
46
47         # light has a minimum of 0, and a maximum of 255.
48         # if maximum value is outside of the interquartile range (3rd quarter), print a warning to user.
49         if max_value > third_quarter:
50             # get max value from dict, and its accompanying key.
51             corresponding_time = 0 # the time that the maximum value occurred
52             biggest = -100000
53             # to find the time where the max value occurred
54             for time, light_val in dictionary.items():
55                 if light_val > biggest:
56                     corresponding_time, biggest = time, light_val
57
58             print("\nAt {time} {interval}, light levels reached {value} {unit}, which lies outside the normal range of {q1} {unit} and {q3} {unit}.".format(time = corresponding_time, interval = time_interval, value = max_value, unit = unit))
59             print("Consider moving to a darker sleeping environment for better sleep.")
60
61         plot_graph(dictionary.keys(), dictionary.values(), "Light Level vs Time", "Time ({interval})".format(interval = time_interval), "Light Level (lx)")
62
```

The function takes in a dictionary containing time:value pairs (where value is light, sound, temp or movement). I converted the list of dictionary values to a numpy array so I can take advantage of numpy's statistics functions.

I displayed all statistics found to the user, such as the mean, median, standard deviation, interquartile range and maximum and minimum values of the list of light values.

In order to meet basic requirement 3, I found the maximum value of the metric being recorded, and compared it with the interquartile range. This will indicate a value that is likely to have interrupted good sleep. If the maximum value lies outside the range, the program will advise the user on how to improve their sleeping environment.

Finally, the function returned the mean and standard deviation of each dictionary. This is used later in the code for the advanced requirements, where I developed my own database of sleep data.

```

21
22 def analyse_data(dictionary, measuring, time_interval = "mins"): # inputs dictionary and what is being measured
23     arr = np.array(list(dictionary.values()))
24
25     # central tendency
26     mean_val = np.mean(arr) # mean value, affected by outliers
27     median_val = np.median(arr) # median value, not affected by outliers
28
29     # measure of spread
30     standard_deviation = np.std(arr) # standard deviation
31     first_quarter = np.percentile(arr, 25) # 1/4 of values
32     third_quarter = np.percentile(arr, 75) # 3/4 of values
33     interquartile_range = third_quarter - first_quarter # IQR of values
34
35     # outliers
36     min_value = np.min(arr) # min value
37     max_value = np.max(arr) # max value
38
39     if measuring == "light":
40         unit = "lux"
41         print("\nLet's see if the light levels in your sleeping environment affect your sleep.\n")
42
43         print("Mean is {mean} {unit}, median is {median} {unit}.".format(mean = round(mean_val, 3), median = round(median_val, 3))
44         print("Interquartile range is {iqr} {unit}, standard deviation is {std} {unit}.".format(iqr = round(interquartile_range,
45         print("Minimum value is {mini} {unit}, maximum value is {maxi} {unit}.".format(mini = round(min_value, 3), maxi = round(m
46
47         # light has a minimum of 0, and a maximum of 255.
48         # if maximum value is outside of the interquartile range (3rd quarter), print a warning to user.
49         if max_value > third_quarter:
50             # get max value from dict, and its accompanying key.
51             corresponding_time = 0 # the time that the maximum value occurred
52             biggest = -1000000
53             # to find the time where the max value occurred
54             for time, light_val in dictionary.items():
55                 if light_val > biggest:
56                     corresponding_time, biggest = time, light_val
57
58             print("\nAt {time} {interval}, light levels reached {value} {unit}, which is lies outside the normal range of {q1} {u
59             print("Consider moving to a darker sleeping environment for better sleep.")
60
61         else:
62             print("\nBased on your sleep data, light levels are falling within an optimal range.")
63             print("This shows that your sleeping environment is excellent for optimal sleep, keep it up!")
64

```

5 – Evaluation

Meeting the project requirements

The final artefact satisfies the requirements set out in the coursework brief.

- Basic 1: I've developed a micro:bit that tracks sleeping habits, which utilises digital inputs (light, sound, temperature and movement) and analogue inputs (time of each measurement). It logs these values and exports them to a computer.
- Basic 2: I used a Python program to validate the data presented in a CSV file, and to store the data in dictionaries.
- Basic 3: I used numpy and matplotlib to analyse the validated data and to calculate key information to advise users on how to improve their sleep.
- Advanced 1: I imported the mean and standard deviation from the data analysis file, and used object oriented programming to develop a dataset of randomly generated sleep data. This dataset is capable of answering the what if questions I've developed.
- Advanced 2: I answered both "what if" questions using three validated parameters of different data types, the user's name (string), the sleep dataset (list), and the number of sleep hours needed OR the ID of the night to analyse (integer).
- Advanced 3: I used matplotlib to display the results of the "what if" questions to users with graphs.

Meeting the requirements of end users

The final artefact meets the requirements of end users set out in the Investigation. As mentioned in section two, my end users, secondary school students, are seeking a device that:

- Tracks their sleep durations and minimises sleep debt.
 - What If Q1, "Am I getting enough sleep?" asks the user to input how many hours of sleep they need every night.
 - The program analyses the user's sleep history and commends them if they meet their sleep goal. Otherwise, it advises them to go to bed earlier.
- Suggests how they can improve their sleep environment for quality sleep.
 - What If Q2, "Is my sleep environment good enough?" reviews the sleep data from a specified night in the user's sleep history.
 - The program gives advice to the user when data collected from their sleep falls outside the optimal range. Otherwise, it congratulates the user and recommends them to maintain the optimal sleep environment.

How the artefact can be improved

Reviewing my program, the main improvement would be to develop a better algorithm to analyse the user's sleep environment and to provide a better solution to What If Q2.

The current algorithm finds the maximum value of light, sound, or temperature and compares it to the optimal range (the interquartile range) of the data collected.

This way, advice is almost always provided to the user, instead of the user seeing a message telling them to maintain their optimal environment.

To further the project I could have considered other What If questions, like analysing how the sleeping environment has changed over time, as well as having the user provide feedback on the program.

For example, the program can ask if the user feels rested or slept well. If not, the program can ask why; did the user not get enough sleep? Was their environment too cold or warm? Or were light and sound levels too high? I can use this information and adjust the optimal range for the user.

Overall, I am satisfied with how I met the requirements of the project, being able to apply my programming skills to improve an aspect of wellbeing.

6 – References

Investigation

1. <https://www.mentalhealthireland.ie/16369-2/>
2. <https://www.apple.com/uk/apple-watch-series-9/>
3. <https://ouraring.com/oura-experience>
4. <https://www.withings.com/be/en/sleep-analyzer>

7 – Summary Word Count

Final Word Count

1. Meeting the Brief: 0
2. Investigation: 492
3. Plan and Design: 531
4. Create: 831
5. Evaluation: 556

Total: 2,410 words