College of Computer Studies – Department of Computer Technology

NSEMBED S12

Mini Project #1: Sleep Monitoring System

Created by:

Marquus Joss R. Azevedo

Jose Mari L. Del Castillo

06/28/2024 – Version 0.2

Change Control

Version	Date	Comments
0.1	06/28/2024	Initial draft
0.2	07/03/2024	Final

Table of Contents

I. Introduction

The project will focus on developing a Sleep Monitoring System with the use of a passive-infrared sensor, an ESP32, and a computer. It will be used for recording the movements and idle times of the subject throughout the subject's sleep. The students will use a PIR sensor for detection of motions in which it is used in automatic switches and other security systems. It is also expected that the students will gather and analyze the results of the project and present the information acquired from it.

II. Objectives

The general objective of this project is to develop a sleep monitoring system to record the subject's sleep movement. The projects specific objectives are to understand the specifications and requirements of the PIR sensor, interface the PIR sensor to an ESP32, establish a communication between the ESP32 and the computer, make a timestamp of movements using the actual time and day, and testing and verifying the sleep monitoring system with actual sleeping scenarios.

III. Scope and Limitations

The PIR sensor has a part number of HC-SR501 which will be discussed in the "PIR Sensor" chapter of the report. The sensor will be interfaced to the ESP32 board in which when it detects movement, the built-in LED of the ESP32 will turn on as for when movement is not detected, the built-in LED pf the ESP32 will turn off. It is mandatory to connect the ESP32 to the computer to send movement activities through serial communication. The computer would store the data and activities are referenced with the actual time for further analysis.

The system will be tested through a series of simulated actions to make sure that it is functioning. It will be also tested with actual sleeping subjects. The procedures and results will be included in the documentation. These tests will be conducted and collected to be presented in the documentation with the use of tables, graphs, and charts with each of them having a description containing explanations and analysis of the results. The entire process of the project will be documented and written on paper.

IV. HC-SR501 (PIR Sensor)

The HC-SR2501 PIR sensor is used to detect someone or something in motion which is typically used for modern security systems such as alarm systems, intrusion detection systems, automation of light switches, and many more applications in which motion sensing is used [1]. It is based on an infrared technology, automatic control module which has high sensitivity and reliability, and ultra-low voltage operating mode [2]. It is considered "passive" because infrared signals are not emitted by the sensor itself instead, it only detects infrared radiation emitted by object in its field of view.

The sensor has two main parts such as the pyroelectric sensor and the Fresnel lens. Figure 6 shows a pyroelectric sensor which allows the infrared radiation to pass through in which behind the window, two separate infrared sensor electrodes are placed in which one is responsible for producing the positive output while the other produces the negative output. Figure 7 shows the Fresnel lens that increases the range and field view of the sensor that consists of a series of concentric grooves which acts as individual refracting surfaces for gathering parallel light rays at a focal point and each divided into several facet-sections to create a range of areas/zones detection [1].

Configurations of the HC-SR501 PIR sensor has an operating voltage range of 5 to 20VDC, a quiescent current of less than 50 microamperes, a high level output of 3.3 V / low level output of 0V, an L trigger that cannot be repeated and an H repeated trigger, an delay time of 0.3 seconds to 5 minutes, a block time of 2.5 seconds, a board dimension of 32mm*24mm, an angel sensor of less than 110 degrees cone angle, an operation temperature of -15 degrees to positive 70 degrees, and a lens size sensor of 23mm in diameter [2].

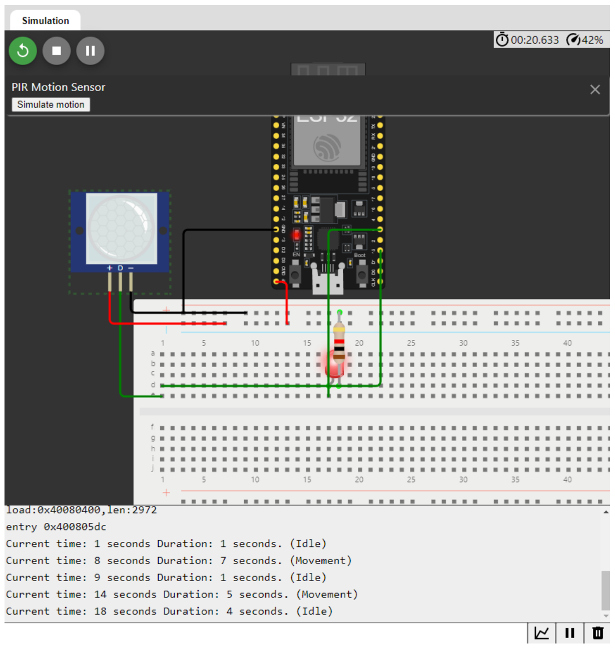
The pins descriptions used in the sensor are VCC, OUT, and GND. The VCC is considered the power supply of the sensor in which if connected an input voltage of 5 to 12V is used. The OUT is the output pin which uses the 3.3V TTL logic output wherein HIGH is when motion is detected and LOW when no motion is detected. Lastly, the GND is the ground pin [1].

The sensor can be adjusted according to its jumper and potentiometer settings. In the jumper settings in Figure 7, there are two types: the retriggering which is in the H position and the non-retriggering which is in the L position [3]. Retriggering means that the output goes HIGH as motion is detected and remains HIGH for a period which is determined by the Time-Delay potentiometer however, further detection is not blocked in which time delay resets whenever motion is detected while non-retriggering means that the output goes HIGH as motion is detected and remains HIGH for a period which is determined by the Time-Delay potentiometer however, further detection is blocked and the output returns to LOW at the end of the time delay. In the potentiometer settings in Figure 8, it can be adjusted according to its sensitivity and time-delay. The sensitivity adjustment part of the potentiometer sets maximum detection range in which it can be adjusted from 3 to 7 meters while for the time-delay adjustment part of the potentiometer sets how long the output will remain HIGH after motion is detected in which it can be adjusted from 1 second to 3 minutes [1].

V. System Overview

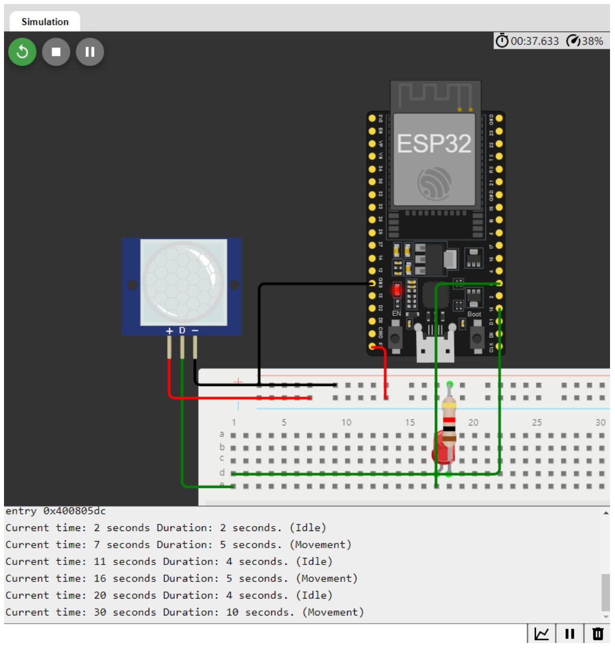
Sleep Monitoring System

The created Sleep Monitoring System is first simulated in Wokwi to test if it detects motion with the LED lighting up if there is movement indicated by the PIR sensor as seen Figure 1 and turning off when there is no movement as seen in Figure 2. Additionally, in the serial monitor, it would also display the total time elapsed in seconds in the "Current time: x seconds" portion of the print statement in which it would be added with the duration of movement or idle time. The duration of the current status is displayed in the "Duration: x seconds. (Status)" portion along with the status (e.g. Idle or Movement) which indicates if the duration of the status displayed is idle (no movement) or movement (indicates motion). Note that for every time it will display the status log, the last status log displayed will be the previous of the current status being recorded as it is still occurring (which is not yet displayed until it stops) in the exception that it is not the first log to be recorded. For example, in starting the actual Arduino program of the Sleep Monitoring System (Figure 3), a user can choose to not do any movement and it will only display the idle time until the user does a movement. Similarly, when the user is moving, it will only display the movement time unless the user stops moving. This behavior is seen in the Arduino program to ensure that when a user stops being idle or stops moving, it would automatically display the previous status and record the current status which is yet to be displayed when it finishes, and the loop will continue. Note that the output in Figure 3 will be the raw data acquired from the PIR sensor. To show the raw data by converting it to a presentable information that one can analyze, the Python program of the Sleep Monitoring System must be used to display the current logs to act as the serial monitor of the Arduino IDE as seen in Figure 4 and to show the logs including the timestamp (time format), status (event type: idle or movement), the duration, total movement time, total idle time, average, median ,mode, maximum and minimum of both movement and idle time (in seconds) in a per row or log basis in which it can be used to show what your current statistics are in a specific timeframe while also getting the final cumulative results in the last log or row of the Excel sheet output of the Python Program as seen in Figure 5. Note that to end the Python program that will run endlessly for as long as the Arduino program is running, a user must press Control then C (Control+C) in the command line as seen in Figure 4.



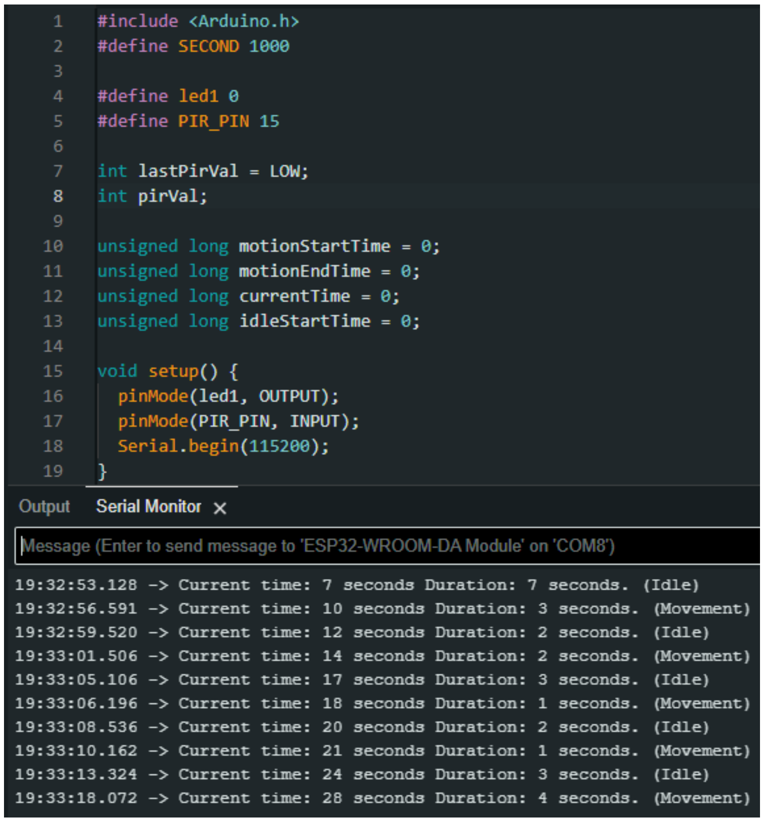
```
load:0x40080400,len:2972
entry 0x400805dc
Current time: 1 seconds Duration: 1 seconds. (Idle)
Current time: 8 seconds Duration: 7 seconds. (Movement)
Current time: 9 seconds Duration: 1 seconds. (Idle)
Current time: 14 seconds Duration: 5 seconds. (Movement)
Current time: 18 seconds Duration: 4 seconds. (Idle)
```

Figure 1. Sleep Monitoring System (Arduino Program) Tested in Wokwi [LED on - Movement]



```
entry 0x400805dc
Current time: 2 seconds Duration: 2 seconds. (Idle)
Current time: 7 seconds Duration: 5 seconds. (Movement)
Current time: 11 seconds Duration: 4 seconds. (Idle)
Current time: 16 seconds Duration: 5 seconds. (Movement)
Current time: 20 seconds Duration: 4 seconds. (Idle)
Current time: 30 seconds Duration: 10 seconds. (Movement)
```

Figure 2. Sleep Monitoring System (Arduino Program) Tested in Wokwi [LED off - Idle]

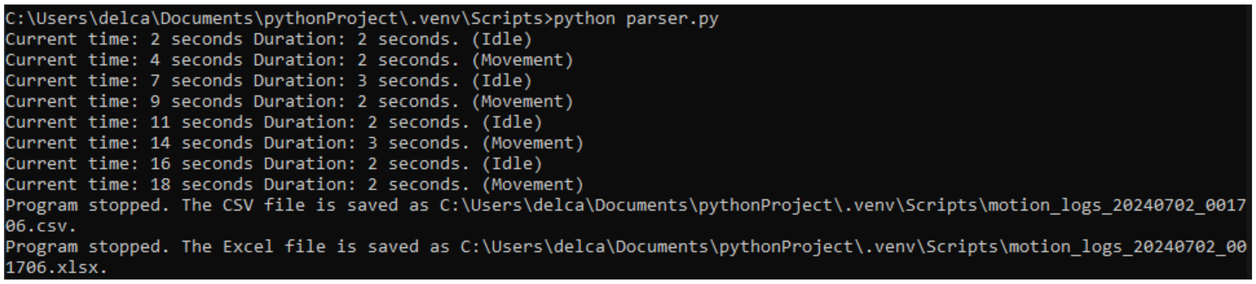


```
#include <Arduino.h>
#define SECOND 1000

#define led1 0
#define PIR_PIN 15

int lastPirVal = LOW;
int pirVal;

unsigned long motionStartTime = 0;
unsigned long motionEndTime = 0;
unsigned long currentTime = 0;
unsigned long idleStartTime = 0;

void setup() {
  pinMode(led1, OUTPUT);
  pinMode(PIR_PIN, INPUT);
  Serial.begin(115200);
}
```

Output
Serial Monitor
Message (Enter to send message to 'ESP32-WROOM-DA Module' on 'COM8')

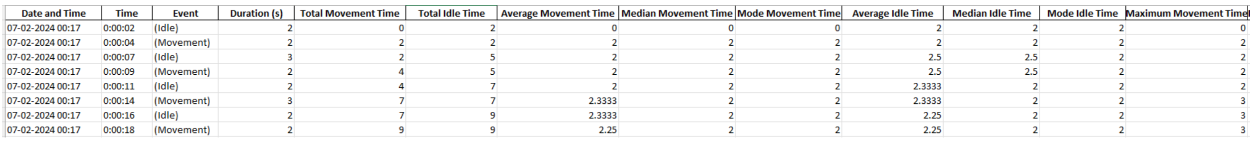
```
19:32:53.128 -> Current time: 7 seconds Duration: 7 seconds. (Idle)
19:32:56.591 -> Current time: 10 seconds Duration: 3 seconds. (Movement)
19:32:59.520 -> Current time: 12 seconds Duration: 2 seconds. (Idle)
19:33:01.506 -> Current time: 14 seconds Duration: 2 seconds. (Movement)
19:33:05.106 -> Current time: 17 seconds Duration: 3 seconds. (Idle)
19:33:06.196 -> Current time: 18 seconds Duration: 1 seconds. (Movement)
19:33:08.536 -> Current time: 20 seconds Duration: 2 seconds. (Idle)
19:33:10.162 -> Current time: 21 seconds Duration: 1 seconds. (Movement)
19:33:13.324 -> Current time: 24 seconds Duration: 3 seconds. (Idle)
19:33:18.072 -> Current time: 28 seconds Duration: 4 seconds. (Movement)
```

Figure 3. Sleep Monitoring System (Arduino Program) Tested in Arduino IDE [Serial Monitor Output]



```
C:\Users\delca\Documents\pythonProject\.venv\Scripts>python parser.py
Current time: 2 seconds Duration: 2 seconds. (Idle)
Current time: 4 seconds Duration: 2 seconds. (Movement)
Current time: 7 seconds Duration: 3 seconds. (Idle)
Current time: 9 seconds Duration: 2 seconds. (Movement)
Current time: 11 seconds Duration: 2 seconds. (Idle)
Current time: 14 seconds Duration: 3 seconds. (Movement)
Current time: 16 seconds Duration: 2 seconds. (Idle)
Current time: 18 seconds Duration: 2 seconds. (Movement)
Program stopped. The CSV file is saved as C:\Users\delca\Documents\pythonProject\.venv\Scripts\motion_logs_20240702_0017
06.csv.
Program stopped. The Excel file is saved as C:\Users\delca\Documents\pythonProject\.venv\Scripts\motion_logs_20240702_00
1706.xlsx.
```

Figure 4. Sleep Monitoring System (Arduino Program & Python Program) [Serial Output]



Date and Time	Time	Event	Duration (s)	Total Movement Time	Total Idle Time	Average Movement Time	Median Movement Time	Mode Movement Time	Average Idle Time	Median Idle Time	Mode Idle Time	Maximum Movement Time
07-02-2024 00:17	0:00:02	(Idle)	2	0	2	0	0	0	2	2	2	0
07-02-2024 00:17	0:00:04	(Movement)	2	2	2	2	2	2	2	2	2	2
07-02-2024 00:17	0:00:07	(Idle)	3	2	5	2	2	2	2.5	2.5	2	2
07-02-2024 00:17	0:00:09	(Movement)	2	4	5	2	2	2	2.5	2.5	2	2
07-02-2024 00:17	0:00:11	(Idle)	2	4	7	2	2	2	2.3333	2	2	2
07-02-2024 00:17	0:00:14	(Movement)	3	7	7	2.3333	2	2	2.3333	2	2	3
07-02-2024 00:17	0:00:16	(Idle)	2	7	9	2.3333	2	2	2.25	2	2	3
07-02-2024 00:17	0:00:18	(Movement)	2	9	9	2.25	2	2	2.25	2	2	3

Figure 5. Sleep Monitoring System (Arduino Program & Python Program) [Serial Output and Excel File Output]

Components List, Hardware Setup and Sensor Configurations

The component list will be listed below:

- HC-SR501 Human Infrared Sensor Module
- ESP32-WROOM-32D (DevKitC_V4)

- 7 Male-to-Female Cables
- 1 Breadboard
- 1 LED
- 1 Resistor (1k Ω)
- Data and Charging Cable

Sensor Configurations and Setup:

The female end of the 3 cables will be connected to the VCC pin (yellow cable), OUT pin (red cable), and GND pin (black cable) of the HC-SR501 (PIR) sensor as seen in Figure 6, respectively. For the jumper setting of the PIR sensor, it will be set to the H position for it to be set to retriggering mode as seen in Figure 7. In retriggering mode, the sensor will continue to output a high signal for as long as the sensor detects movement. This retriggering mode will ensure that the proper duration times will be printed out for idle and movement times. For the sensitivity setting potentiometer (left potentiometer) of the PIR as seen in Figure 8, it will be set to a sensitivity of 3 meters (turned to left maximum) since the student will be near the PIR sensor when testing. For the delay time potentiometer (right potentiometer) of the PIR as also seen in Figure 8, it will be set to 0.3 seconds (turned to left maximum) to get the least amount of delay for more or less accurate duration times.

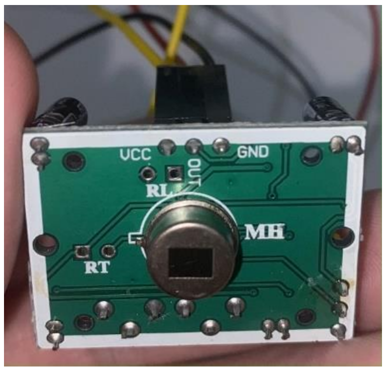


Figure 6. HC-SR501 Sensor and Cables

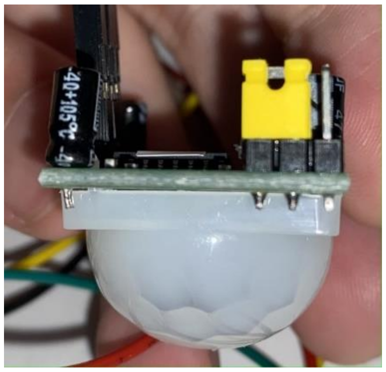


Figure 7. HC-SR501 Jumper Setting

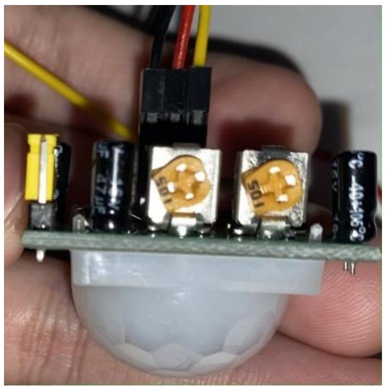


Figure 8. HC-SR501 Potentiometer Settings

ESP32 Setup:

For the ESP32 setup, the female end of the 4 cables will be connected to the 5V (5 volts – yellow cable) pin, GND (Ground – green cable) pin 0 (blue cable) which will be used for lighting up the LED, and pin 15 (red cable) which will serve as the PIR pin as seen in Figure 9. For power and data connection, the USB data cable will be used to connect the ESP32 to the testing machine (laptop of the student).

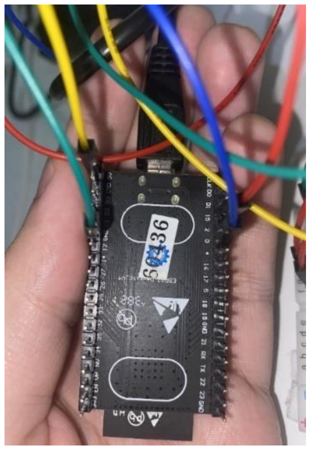


Figure 9. ESP32 Cables

Breadboard Setup:

For the breadboard setup (Figure 10) regarding the cables connected to the PIR sensor, the red cable should be connected to the f1 grid socket of the breadboard, the yellow cable should be connected to the connected to the positive portion (red) of the power rail, and the black cable must be connected to the ground portion (blue) of the power rail.

For the breadboard setup regarding the cables connected to the ESP32, the red cable must be connected to the g1 grid socket next to the red cable from the PIR sensor to be able to send the signals from the PIR sensor to the ESP32. The green cable must be connected to the ground portion (blue) of the power rail as both black (PIR sensor) and green cables are for ground. The yellow cable must be connected to the positive portion of the power rail (red) as both yellow (PIR sensor and ESP32) cables are for positive voltage.

The blue cable from the ESP32 will be connected to the f15 grid socket in which the anode side of the LED will be connected the h15 grid socket to power the LED with the ESP32. Lastly, the resistor will be connected to the ground portion of the power rail while the other end of the resistor will be connected to the cathode side of the LED to complete the circuit. The final result of the setup will be shown in Figure 11.

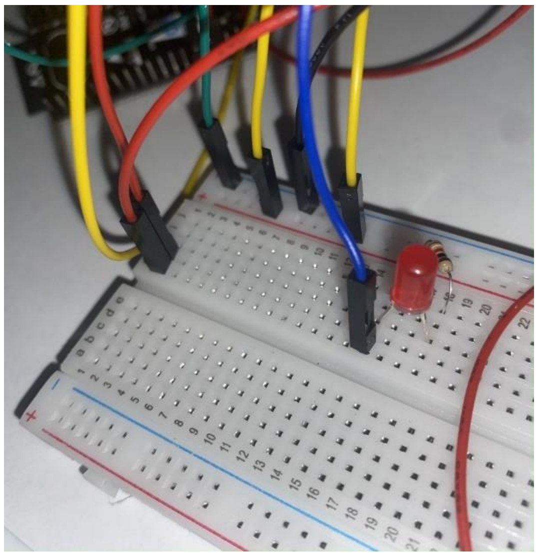


Figure 10. Sleep Monitoring System Breadboard Setup

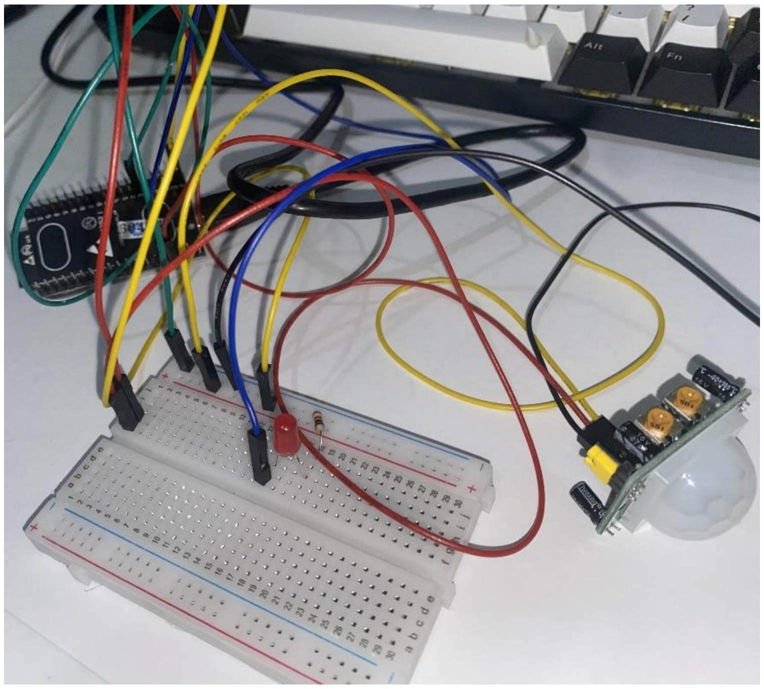


Figure 11. Sleep Monitoring System

Testing and Data Encoding in Python of Raw Data acquired from the Sensor

The initial testing of the Sleep Monitoring System was conducted on the table of one of the students and it was determined that the system is functioning properly as seen in Figure 12, 13, and 14 in which the duration times are accurate with a margin of error of at least 1 or 2 seconds which is most likely due to the delay time setting of the sensor. However, the pattern of logs works as intended as the first log that will display is the idle time, and it will show the movement time next after the movement is done, and it will continue the pattern.

A video of the short demonstration of the Sleep Monitoring System will be seen in: Sleep Monitoring System Initial Test. The output for the Python program can be seen in Figure 15, and the Excel sheet that will be produced is seen in Figure 16 where it gives the correct data for each cell. The excel sheet can be accessed in: motion_logs_20240702_010259

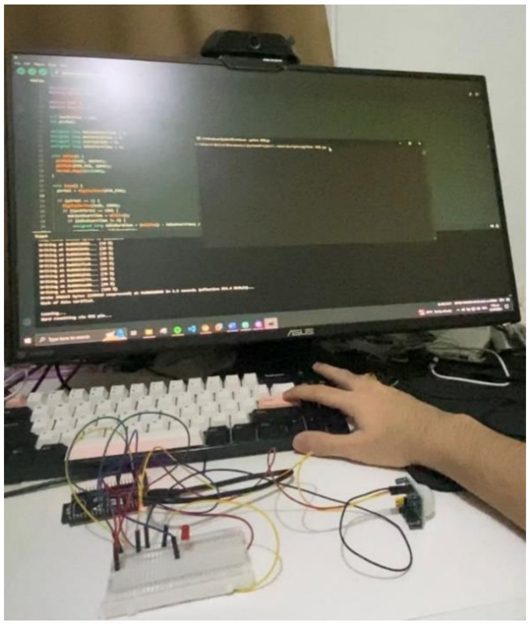


Figure 12. Sleep Monitoring System (Initial Test) [Idle]

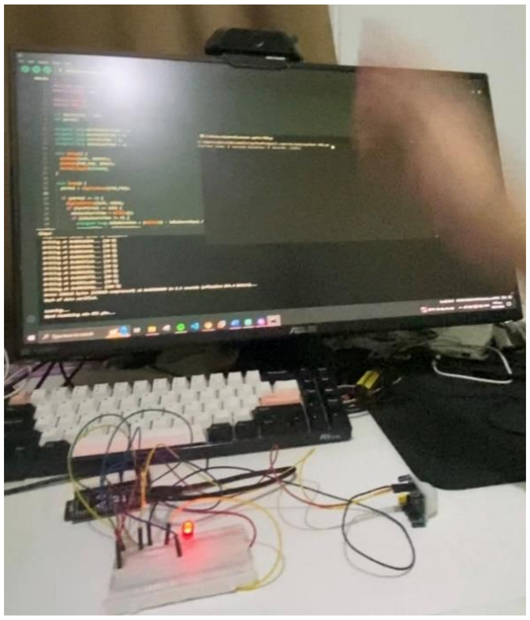
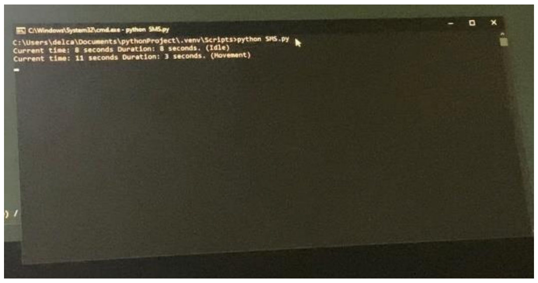
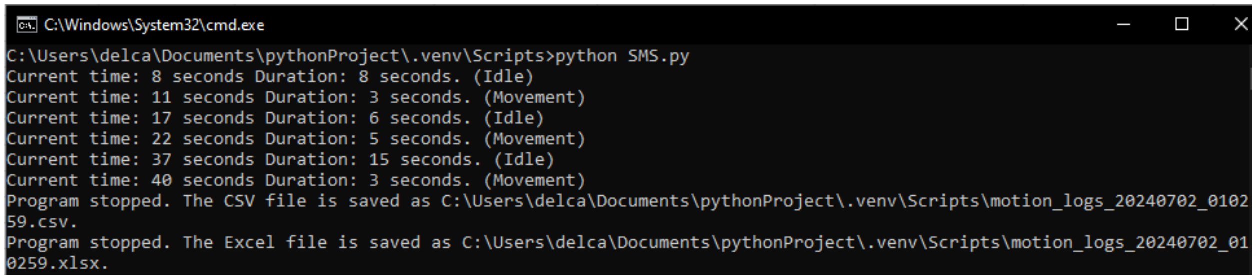


Figure 13. Sleep Monitoring System (Initial Test) [Movement]



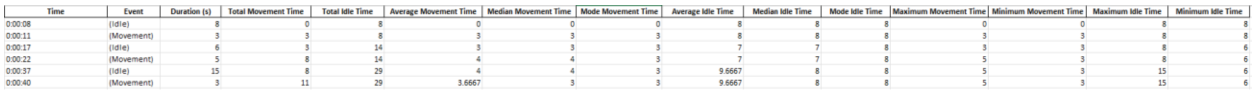
```
C:\Windows\System32\cmd.exe - python SMS.py
C:\Users\delca\Documents\pythonProject\.venv\Scripts>python SMS.py
Current time: 8 seconds Duration: 8 seconds. (Idle)
Current time: 11 seconds Duration: 3 seconds. (Movement)
```

Figure 14. Sleep Monitoring System (Python Program) [Output]



```
C:\Users\delca\Documents\pythonProject\.venv\Scripts>python SMS.py
Current time: 8 seconds Duration: 8 seconds. (Idle)
Current time: 11 seconds Duration: 3 seconds. (Movement)
Current time: 17 seconds Duration: 6 seconds. (Idle)
Current time: 22 seconds Duration: 5 seconds. (Movement)
Current time: 37 seconds Duration: 15 seconds. (Idle)
Current time: 40 seconds Duration: 3 seconds. (Movement)
Program stopped. The CSV file is saved as C:\Users\delca\Documents\pythonProject\.venv\Scripts\motion_logs_20240702_0102
59.csv.
Program stopped. The Excel file is saved as C:\Users\delca\Documents\pythonProject\.venv\Scripts\motion_logs_20240702_01
0259.xlsx.
```

Figure 15. Sleep Monitoring System (Python Program) [Whole Output]



Time	Event	Duration (s)	Total Movement Time	Total Idle Time	Average Movement Time	Median Movement Time	Mode Movement Time	Average Idle Time	Median Idle Time	Mode Idle Time	Maximum Movement Time	Minimum Movement Time	Maximum Idle Time	Minimum Idle Time
0:00:08	(Idle)	8	0	8	0	0	0	8	8	8	0	0	8	8
0:00:11	(Movement)	3	3	8	3	3	3	8	8	8	3	3	8	8
0:00:17	(Idle)	6	3	14	3	3	3	7	7	8	3	3	8	6
0:00:22	(Movement)	5	8	14	4	4	3	7	7	8	5	3	8	6
0:00:37	(Idle)	15	8	29	4	4	3	9.6667	8	8	5	3	15	6
0:00:40	(Movement)	3	11	29	3.6667	3	3	9.6667	8	8	5	3	15	6

Figure 16. Sleep Monitoring System (Python Program Excel File Output)

The final testing of the Sleep Monitoring System will be conducted in the room of one of the students as seen in the setup on the first night of testing that will be used for the succeeding nights as seen in Figure 17.

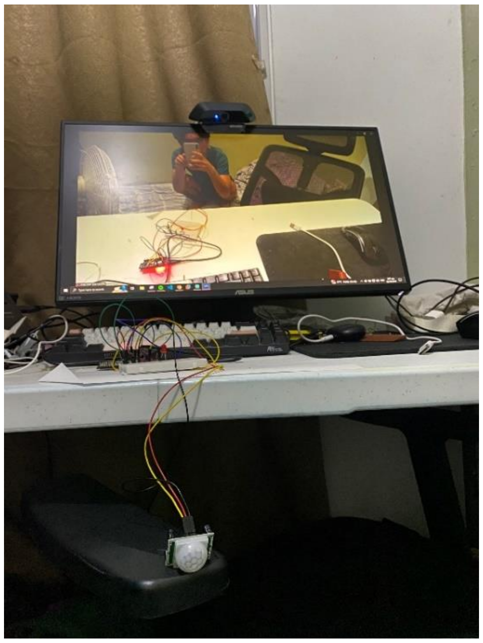
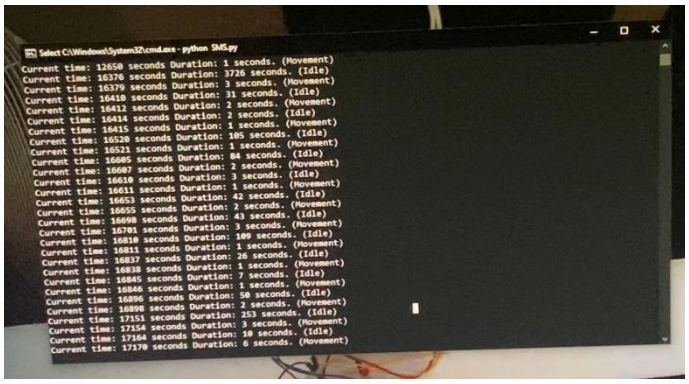


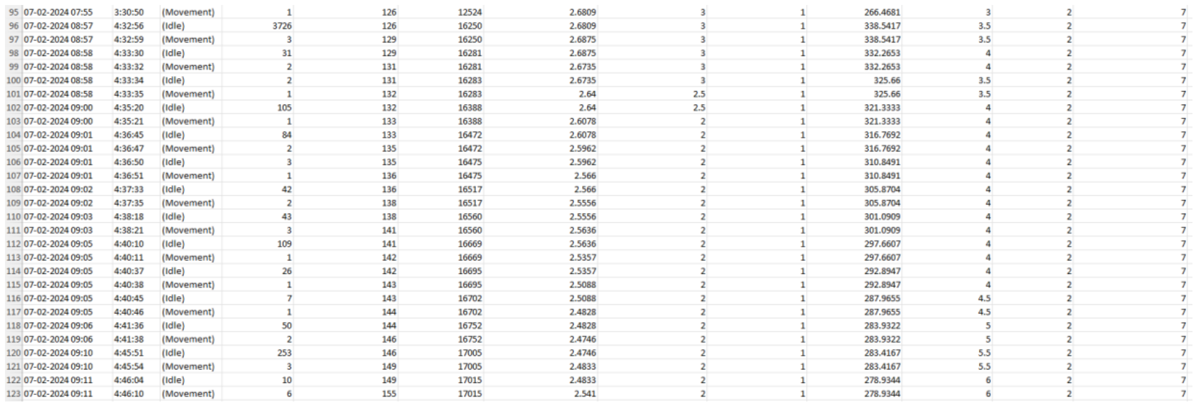
Figure 17. Sleep Monitoring System (Real Time Testing Setup – First Night)

As seen in Figure 18, the Python program was able to record and collect data for long periods of time. The logs are also the same in the Excel sheet output as seen Figure 19. It can be observed that the data regarding the current time, duration, and status is the same in the Excel sheet output of the Python program. In Figure 20, it can be observed that the first movement log present in Figure 19 is the same log that occurred. It was verified as the time in the VLC player was advanced by 3m (Original time: 3:30:50, VLC time: 3:34:12). The process will then be repeated for the next night as the data will be analyzed for each day while comparing all the results.



```
Select C:\Windows\System32\cmd.exe - python SMS.py
Current time: 12650 seconds Duration: 1 seconds. (Movement)
Current time: 16376 seconds Duration: 3726 seconds. (Idle)
Current time: 16379 seconds Duration: 3 seconds. (Movement)
Current time: 16410 seconds Duration: 31 seconds. (Idle)
Current time: 16412 seconds Duration: 2 seconds. (Movement)
Current time: 16414 seconds Duration: 2 seconds. (Idle)
Current time: 16415 seconds Duration: 1 seconds. (Movement)
Current time: 16520 seconds Duration: 105 seconds. (Idle)
Current time: 16521 seconds Duration: 1 seconds. (Movement)
Current time: 16605 seconds Duration: 84 seconds. (Idle)
Current time: 16607 seconds Duration: 2 seconds. (Movement)
Current time: 16610 seconds Duration: 3 seconds. (Idle)
Current time: 16611 seconds Duration: 1 seconds. (Movement)
Current time: 16653 seconds Duration: 42 seconds. (Idle)
Current time: 16655 seconds Duration: 2 seconds. (Movement)
Current time: 16698 seconds Duration: 43 seconds. (Idle)
Current time: 16701 seconds Duration: 3 seconds. (Movement)
Current time: 16810 seconds Duration: 109 seconds. (Idle)
Current time: 16811 seconds Duration: 1 seconds. (Movement)
Current time: 16837 seconds Duration: 26 seconds. (Idle)
Current time: 16838 seconds Duration: 1 seconds. (Movement)
Current time: 16845 seconds Duration: 7 seconds. (Idle)
Current time: 16846 seconds Duration: 1 seconds. (Movement)
Current time: 16896 seconds Duration: 50 seconds. (Idle)
Current time: 16898 seconds Duration: 2 seconds. (Movement)
Current time: 17151 seconds Duration: 253 seconds. (Idle)
Current time: 17154 seconds Duration: 3 seconds. (Movement)
Current time: 17164 seconds Duration: 10 seconds. (Idle)
Current time: 17170 seconds Duration: 6 seconds. (Movement)
```

Figure 18. Sleep Monitoring System (First Night) [Python Program output snippet]



95	07-02-2024 07:55	3:30:50	(Movement)	1	126	12524	2.6809	3	1	266.4681	3	2	7
96	07-02-2024 08:57	4:32:56	(Idle)	3726	126	16250	2.6809	3	1	338.5417	3.5	2	7
97	07-02-2024 08:57	4:32:59	(Movement)	3	129	16250	2.6875	3	1	338.5417	3.5	2	7
98	07-02-2024 08:58	4:33:30	(Idle)	31	129	16281	2.6875	3	1	332.2653	4	2	7
99	07-02-2024 08:58	4:33:32	(Movement)	2	131	16281	2.6735	3	1	332.2653	4	2	7
100	07-02-2024 08:58	4:33:34	(Idle)	2	131	16283	2.6735	3	1	325.66	3.5	2	7
101	07-02-2024 08:58	4:33:35	(Movement)	1	132	16283	2.64	2.5	1	325.66	3.5	2	7
102	07-02-2024 09:00	4:35:20	(Idle)	105	132	16388	2.64	2.5	1	321.3333	4	2	7
103	07-02-2024 09:00	4:35:21	(Movement)	1	133	16388	2.6078	2	1	321.3333	4	2	7
104	07-02-2024 09:01	4:36:45	(Idle)	84	133	16472	2.6078	2	1	316.7692	4	2	7
105	07-02-2024 09:01	4:36:47	(Movement)	2	135	16472	2.5962	2	1	316.7692	4	2	7
106	07-02-2024 09:01	4:36:50	(Idle)	3	135	16475	2.5962	2	1	310.8491	4	2	7
107	07-02-2024 09:01	4:36:51	(Movement)	1	136	16475	2.566	2	1	310.8491	4	2	7
108	07-02-2024 09:02	4:37:33	(Idle)	42	136	16517	2.566	2	1	305.8704	4	2	7
109	07-02-2024 09:02	4:37:35	(Movement)	2	138	16517	2.5556	2	1	305.8704	4	2	7
110	07-02-2024 09:03	4:38:18	(Idle)	43	138	16560	2.5556	2	1	301.0909	4	2	7
111	07-02-2024 09:03	4:38:21	(Movement)	3	141	16560	2.5636	2	1	301.0909	4	2	7
112	07-02-2024 09:05	4:40:10	(Idle)	109	141	16669	2.5636	2	1	297.6607	4	2	7
113	07-02-2024 09:05	4:40:11	(Movement)	1	142	16669	2.5357	2	1	297.6607	4	2	7
114	07-02-2024 09:05	4:40:37	(Idle)	26	142	16695	2.5357	2	1	292.8947	4	2	7
115	07-02-2024 09:05	4:40:38	(Movement)	1	143	16695	2.5088	2	1	292.8947	4	2	7
116	07-02-2024 09:05	4:40:45	(Idle)	7	143	16702	2.5088	2	1	287.9655	4.5	2	7
117	07-02-2024 09:05	4:40:46	(Movement)	1	144	16702	2.4828	2	1	287.9655	4.5	2	7
118	07-02-2024 09:06	4:41:36	(Idle)	50	144	16752	2.4828	2	1	283.9322	5	2	7
119	07-02-2024 09:06	4:41:38	(Movement)	2	146	16752	2.4746	2	1	283.9322	5	2	7
120	07-02-2024 09:10	4:45:51	(Idle)	253	146	17005	2.4746	2	1	283.4167	5.5	2	7
121	07-02-2024 09:10	4:45:54	(Movement)	3	149	17005	2.4833	2	1	283.4167	5.5	2	7
122	07-02-2024 09:11	4:46:04	(Idle)	10	149	17015	2.4833	2	1	278.9344	6	2	7
123	07-02-2024 09:11	4:46:10	(Movement)	6	155	17015	2.541	2	1	278.9344	6	2	7

Figure 19. Sleep Monitoring System (First Night) [Python Program Excel output snippet]

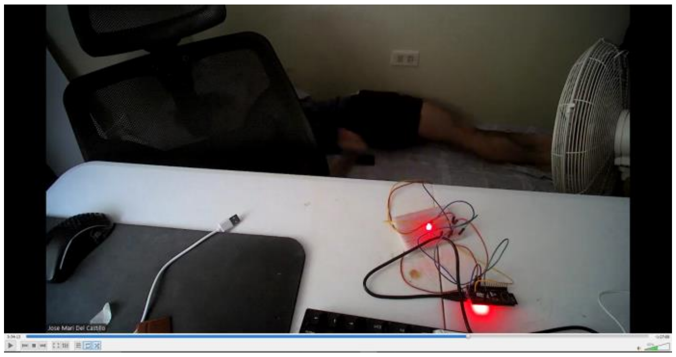
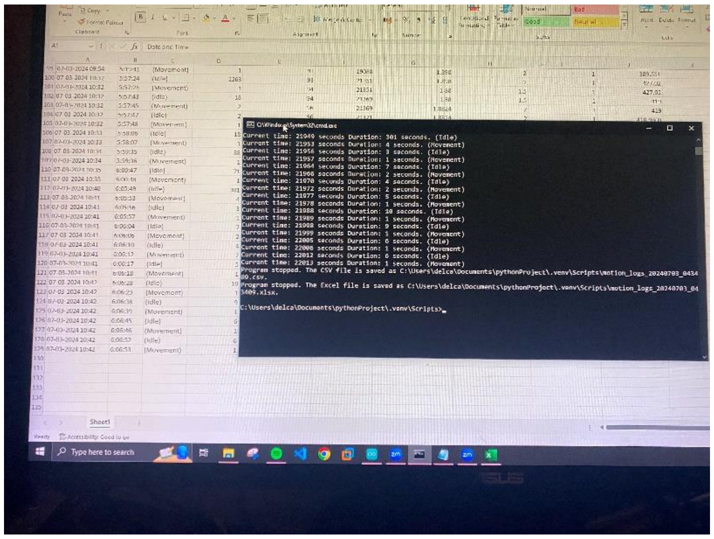


Figure 20. Sleep Monitoring System (First Night) [Motion Detected]

As seen in Figure 21, the Sleep Monitoring System was tested for the second night and the results for each night will be compared.

The video recording for the verification of the data gathered from the Sleep Monitoring System and the Excel output logs for the two nights will be seen in: [Sleep Monitoring]

System Video Recordings and Logs (will only be accessible with DLSU Microsoft OneDrive Email)

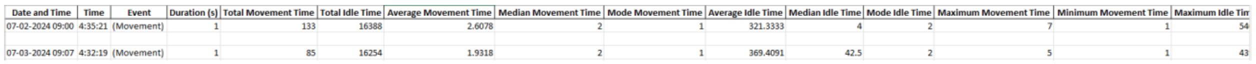


```
C:\Windows\System32\cmd.exe
Current time: 21940 seconds Duration: 301 seconds. (Idle)
Current time: 21953 seconds Duration: 4 seconds. (Movement)
Current time: 21956 seconds Duration: 3 seconds. (Idle)
Current time: 21957 seconds Duration: 1 seconds. (Movement)
Current time: 21964 seconds Duration: 7 seconds. (Idle)
Current time: 21966 seconds Duration: 2 seconds. (Movement)
Current time: 21970 seconds Duration: 4 seconds. (Idle)
Current time: 21972 seconds Duration: 2 seconds. (Movement)
Current time: 21977 seconds Duration: 5 seconds. (Idle)
Current time: 21978 seconds Duration: 1 seconds. (Movement)
Current time: 21988 seconds Duration: 10 seconds. (Idle)
Current time: 21989 seconds Duration: 1 seconds. (Movement)
Current time: 21998 seconds Duration: 9 seconds. (Idle)
Current time: 21999 seconds Duration: 1 seconds. (Movement)
Current time: 22005 seconds Duration: 6 seconds. (Idle)
Current time: 22006 seconds Duration: 1 seconds. (Movement)
Current time: 22012 seconds Duration: 6 seconds. (Idle)
Current time: 22013 seconds Duration: 1 seconds. (Movement)
Program stopped. The CSV file is saved as C:\Users\delca\Documents\pythonProject\.venv\Scripts\motion_logs_20240703_0434
09.csv.
Program stopped. The Excel file is saved as C:\Users\delca\Documents\pythonProject\.venv\Scripts\motion_logs_20240703_04
3409.xlsx.

C:\Users\delca\Documents\pythonProject\.venv\Scripts>_
```

Figure 21. Sleep Monitoring System (Second Night) [Python Program output snippet and Excel sheet]

VI. Results and Analysis

For the two nights of testing in the Sleep Monitoring System Python programs outputs, one row will be selected in each log file for comparison. As seen in Figure 22, the first row will be selected as it is the log that the student will wake up for the first night of testing (student had an alarm set at 9 AM). For the third row in Figure 22 for the second night of testing, the third row will be selected as it is the closest time to the first row with a gap of 2 minutes and 58 seconds. This is to ensure that the comparison would be fair enough as considering the time that the student woke up for the second night of testing in which the total time elapsed is about 6 hours would possibly make a bigger difference in the information gathered.



Date and Time	Time	Event	Duration (s)	Total Movement Time	Total Idle Time	Average Movement Time	Median Movement Time	Mode Movement Time	Average Idle Time	Median Idle Time	Mode Idle Time	Maximum Movement Time	Minimum Movement Time	Maximum Idle Tim
07-02-2024 09:00	4:35:21	(Movement)	1	133	16388	2.6078	2	1	321.3333	4	2	7	1	54
07-03-2024 09:07	4:32:19	(Movement)	1	85	16254	1.9318	2	1	369.4091	42.5	2	5	1	43

Figure 22

Night 1 and Night 2 Results and Comparisons:

It was found that the ratio of movement and being idle is similar in the first and second night as seen in Figure 23 and 24. It can be said that the student does not move that much when he is sleeping.

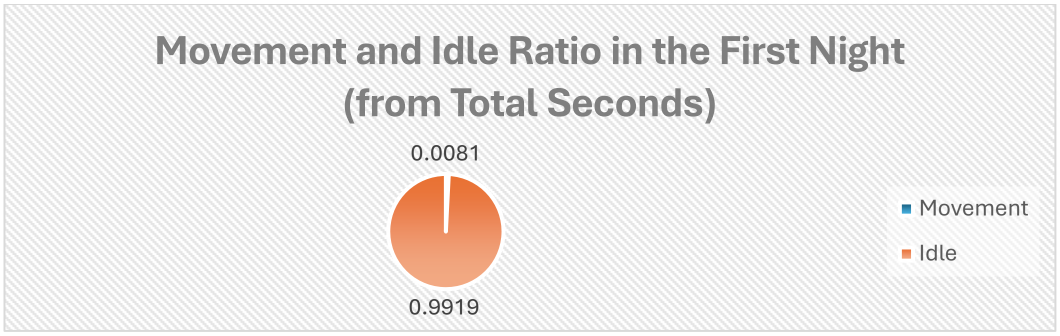


Figure 23

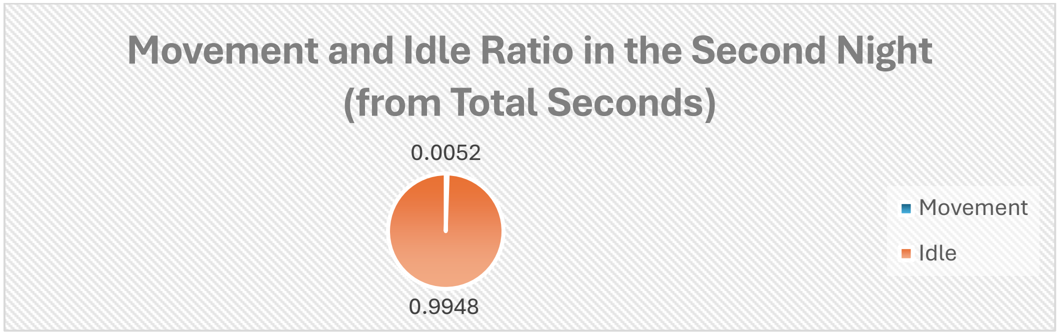


Figure 24

For the average movement times as seen in Figure 25, it could be observed that the student moved a lot more in his sleep in the first night compared to the second night. As expected, the idle time for the first night will be less than the idle time in the second night as the student moved more in the first night. The median movement time is the same for the first two nights. Surprisingly, the median idle time for the second night is significantly larger than the median idle time in the first night. This indicates that a lot of the idle times gathered in the second night have high values which can be seen as well in Figure 26.

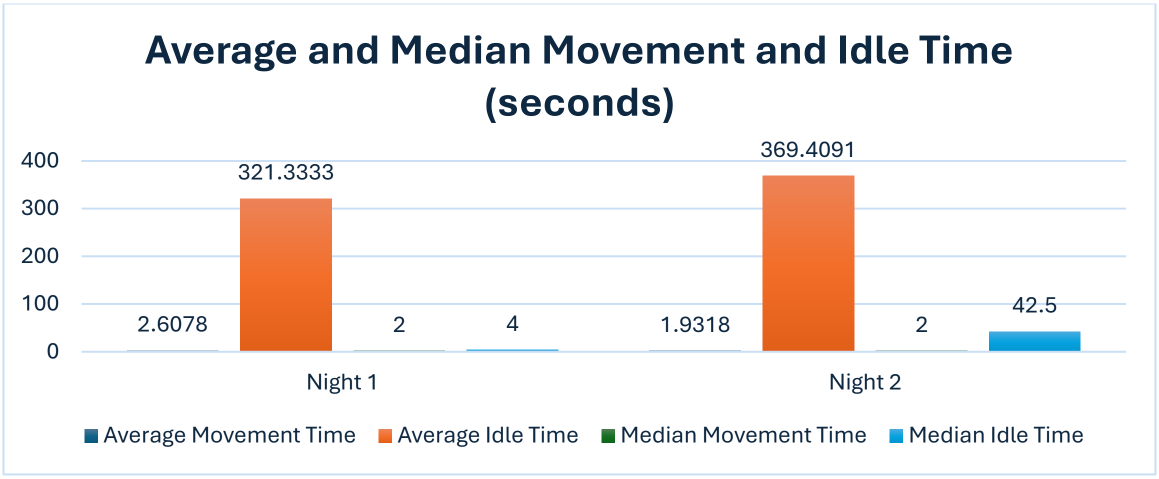
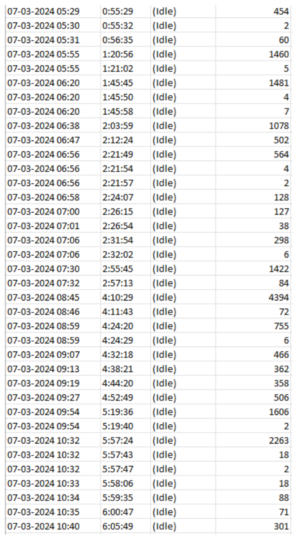


Figure 25



07-03-2024 05:29	0:55:29	(Idle)	454
07-03-2024 05:30	0:55:32	(Idle)	2
07-03-2024 05:31	0:56:35	(Idle)	60
07-03-2024 05:55	1:20:56	(Idle)	1460
07-03-2024 05:55	1:21:02	(Idle)	5
07-03-2024 06:20	1:45:45	(Idle)	1481
07-03-2024 06:20	1:45:50	(Idle)	4
07-03-2024 06:20	1:45:58	(Idle)	7
07-03-2024 06:38	2:03:59	(Idle)	1078
07-03-2024 06:47	2:12:24	(Idle)	502
07-03-2024 06:56	2:21:49	(Idle)	564
07-03-2024 06:56	2:21:54	(Idle)	4
07-03-2024 06:56	2:21:57	(Idle)	2
07-03-2024 06:58	2:24:07	(Idle)	128
07-03-2024 07:00	2:26:15	(Idle)	127
07-03-2024 07:01	2:26:54	(Idle)	38
07-03-2024 07:06	2:31:54	(Idle)	298
07-03-2024 07:06	2:32:02	(Idle)	6
07-03-2024 07:30	2:55:45	(Idle)	1422
07-03-2024 07:32	2:57:13	(Idle)	84
07-03-2024 08:45	4:10:29	(Idle)	4394
07-03-2024 08:46	4:11:43	(Idle)	72
07-03-2024 08:59	4:24:20	(Idle)	755
07-03-2024 08:59	4:24:29	(Idle)	6
07-03-2024 09:07	4:32:18	(Idle)	466
07-03-2024 09:13	4:38:21	(Idle)	362
07-03-2024 09:19	4:44:20	(Idle)	358
07-03-2024 09:27	4:52:49	(Idle)	506
07-03-2024 09:54	5:19:36	(Idle)	1606
07-03-2024 09:54	5:19:40	(Idle)	2
07-03-2024 10:32	5:57:24	(Idle)	2263
07-03-2024 10:32	5:57:43	(Idle)	18
07-03-2024 10:32	5:57:47	(Idle)	2
07-03-2024 10:33	5:58:06	(Idle)	18
07-03-2024 10:34	5:59:35	(Idle)	88
07-03-2024 10:35	6:00:47	(Idle)	71
07-03-2024 10:40	6:05:49	(Idle)	301

Figure 26. Sleep Monitoring System (Excel output – second night sorted by type [idle])

As seen in Figure 27, the maximum movement time of the first night is at 7 seconds while the maximum movement time in the second night is at 5 seconds which corresponds to the average movement time and average idle time as seen Figure 25. Despite the second night having lower maximum movement time, the first night still had a higher maximum idle time than the second night.

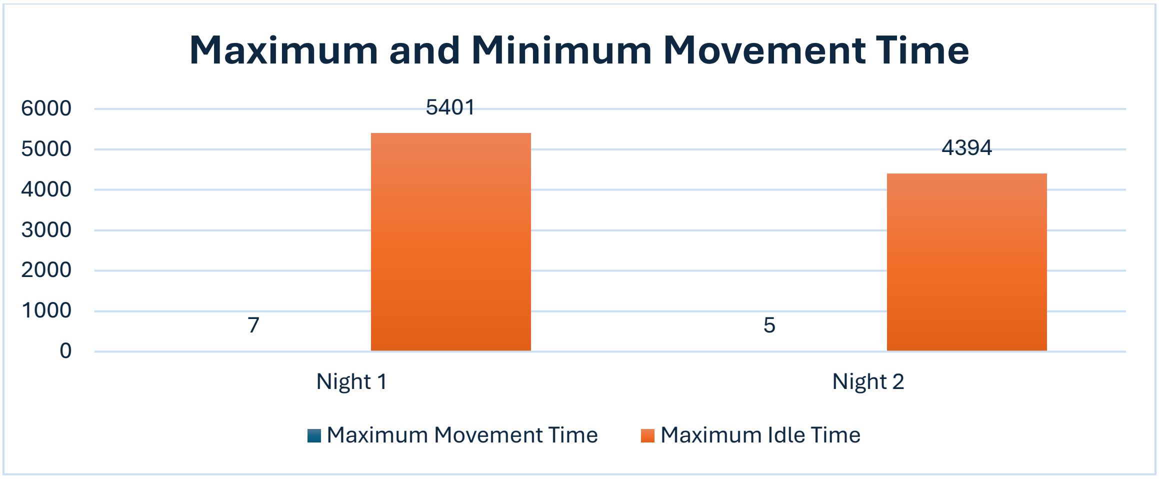


Figure 27

VII. Conclusion

In conclusion, the students were able to successfully make a functionable and useful Sleep Monitoring System using the ESP32, HC-SR501 PIR sensor, a computer, Arduino Program, and Python program that can be used to analyze a person's sleep with regards to his or her movement and idle times. The raw data gathered from the testing was converted into presentable information as we can see the timestamps for every event of idle and movement times, the time elapsed, the duration of each log or event, total movement time, total idle time, average, median, mode, maximum and minimum of both movement and idle time in multiple timeframes of your sleep (per row) for it to be compared with results from the different nights of sleeping as seen in the results and analysis. The system was successfully tested in real-time sleeping scenarios and gained useful and interpretable information from it. The students have also gained appreciation towards such a small and cheap sensor that can be used to gather useful information from anyone's sleep that can be analyzed and interpreted.

VIII. Appendix

A. Arduino Code

```
#include <Arduino.h>
#define SECOND 1000

#define led1 0
#define PIR_PIN 15

int lastPirVal = LOW;
int pirVal;

unsigned long motionStartTime = 0;
unsigned long motionEndTime = 0;
unsigned long currentTime = 0;
unsigned long idleStartTime = 0;

void setup() {
  pinMode(led1, OUTPUT);
  pinMode(PIR_PIN, INPUT);
  Serial.begin(115200);
}

void loop() {
  pirVal = digitalRead(PIR_PIN);

  if (pirVal == 1) {
    digitalWrite(led1, HIGH);
    if (lastPirVal == LOW) {
      motionStartTime = millis();
      if (idleStartTime != 0) {
        unsigned long idleDuration = (millis() - idleStartTime) / SECOND;
        currentTime += idleDuration;
        Serial.print("Current time: ");
        Serial.print(currentTime);
        Serial.print(" seconds Duration: ");
        Serial.print(idleDuration);
        Serial.println(" seconds. (Idle)");
      }
      lastPirVal = HIGH;
    }
  } else {
    digitalWrite(led1, LOW);
    if (lastPirVal == HIGH) {
      unsigned long motionDuration = (millis() - motionStartTime) / SECOND;
      if (motionDuration > 0) {
        currentTime += motionDuration;
        Serial.print("Current time: ");
        Serial.print(currentTime);
        Serial.print(" seconds Duration: ");
        Serial.print(motionDuration);
        Serial.println(" seconds. (Movement)");
      }

      idleStartTime = millis();
      lastPirVal = LOW;
    } else {
      if (idleStartTime == 0) {
        idleStartTime = millis();
      }
    }
  }
}
```

B. Python Code

```
import serial, csv, sys, statistics, os
from datetime import datetime, timedelta
import pandas as pd
from openpyxl import load_workbook

filename = f'motion_logs_{datetime.now().strftime("%Y%m%d_%H%M%S")}.csv'

serport = serial.Serial('COM8', 115200)

with open(filename, 'a', newline='') as csvfile:
    fieldnames = ['Date and Time', 'Time', 'Event', 'Duration (s)',
                  'Total Movement Time', 'Total Idle Time', 'Average Movement Time', 'Median
Movement Time', 'Mode Movement Time',
                  'Average Idle Time', 'Median Idle Time', 'Mode Idle Time','Maximum Movement Time',
'Minimum Movement Time',
                  'Maximum Idle Time', 'Minimum Idle Time']
    writer = csv.DictWriter(csvfile, fieldnames=fieldnames)

    if csvfile.tell() == 0:
        writer.writeheader()

    total_movement_time = 0
    total_idle_time = 0
    countMovement = 0
    countIdle = 0
    average_movement_time = 0
    average_idle_time = 0
    movement_time_list = []
    idle_time_list = []

    try:
        while True:
            buffer = serport.readline().decode().strip()
            print(buffer)
            time_elapsed = buffer.split(' ')[2]
            duration = buffer.split(' ')[5]
            status = buffer.split(' ')[7]

            if status == "(Movement)":
                current_datetime = datetime.now()
                formatted_datetime = current_datetime.strftime("%m-%d-%Y %H:%M")
                total_movement_time += int(duration)
                countMovement += 1
                movement_time_list.append(int(duration))
                try:
                    average_movement_time = round(total_movement_time/countMovement, 4)
                except ZeroDivisionError:
                    average_movement_time = 0
                try:
                    median_movement_time = round(statistics.median(movement_time_list), 4)
                except statistics.StatisticsError:
                    median_movement_time = 0
                try:
                    mode_movement_time = statistics.mode(movement_time_list)
                except statistics.StatisticsError:
                    mode_movement_time = 0

                try:
                    average_idle_time = round(total_idle_time/countIdle, 4)
                except ZeroDivisionError:
                    average_idle_time = 0

                try:
                    median_idle_time = round(statistics.median(idle_time_list), 4)
                except statistics.StatisticsError:
                    median_idle_time = 0

                try:
                    mode_idle_time = statistics.mode(idle_time_list)
                except statistics.StatisticsError:
                    mode_idle_time = 0


                try:
                    max_movement_time = max(movement_time_list)
                except ValueError:
                    max_movement_time = 0

                try:
                    min_movement_time = min(movement_time_list)
                except ValueError:
                    min_movement_time = 0

                try:
                    max_idle_time = max(idle_time_list)
                except ValueError:
                    max_idle_time = 0

                try:
                    min_idle_time = min(idle_time_list)
                except ValueError:
                    min_idle_time = 0
                writer.writerow({
                    'Date and Time': formatted_datetime,
                    'Time': timedelta(seconds = int(time_elapsed)),
                    'Event': status,
                    'Duration (s)': duration,
                    'Total Movement Time': total_movement_time,
                    'Total Idle Time': total_idle_time,
                    'Average Movement Time': average_movement_time,
                    'Median Movement Time': median_movement_time,
                    'Mode Movement Time': mode_movement_time,
                    'Average Idle Time': average_idle_time,
                    'Median Idle Time': median_idle_time,
                    'Mode Idle Time': mode_idle_time,
                    'Maximum Movement Time': max_movement_time,
                    'Minimum Movement Time': min_movement_time,
                    'Maximum Idle Time': max_idle_time,
                    'Minimum Idle Time': min_idle_time
                })
            else:
                current_datetime = datetime.now()
                formatted_datetime = current_datetime.strftime("%m-%d-%Y %H:%M")
                total_idle_time += int(duration)
                countIdle += 1
                idle_time_list.append(int(duration))
                try:
                    average_movement_time = round(total_movement_time / countMovement, 4)
                except ZeroDivisionError:
                    average_movement_time = 0
                try:
                    median_movement_time = round(statistics.median(movement_time_list), 4)
                except statistics.StatisticsError:
                    median_movement_time = 0
                try:
                    mode_movement_time = statistics.mode(movement_time_list)
                except statistics.StatisticsError:
                    mode_movement_time = 0

                try:
                    average_idle_time = round(total_idle_time / countIdle, 4)
                except ZeroDivisionError:
                    average_idle_time = 0

                try:
                    median_idle_time = round(statistics.median(idle_time_list), 4)
                except statistics.StatisticsError:
                    median_idle_time = 0

                try:
                    mode_idle_time = statistics.mode(idle_time_list)
                except statistics.StatisticsError:
                    mode_idle_time = 0

                try:
                    max_movement_time = max(movement_time_list)
                except ValueError:
                    max_movement_time = 0

                try:
                    min_movement_time = min(movement_time_list)
                except ValueError:
                    min_movement_time = 0

                try:
                    max_idle_time = max(idle_time_list)
                except ValueError:
                    max_idle_time = 0

                try:
                    min_idle_time = min(idle_time_list)
                except ValueError:
                    min_idle_time = 0

                writer.writerow({
                    'Date and Time': formatted_datetime,
                    'Time': timedelta(seconds = int(time_elapsed)),
                    'Event': status,
                    'Duration (s)': duration,
                    'Total Movement Time': total_movement_time,
                    'Total Idle Time': total_idle_time,
                    'Average Movement Time': average_movement_time,
                    'Median Movement Time': median_movement_time,
                    'Mode Movement Time': mode_movement_time,
                    'Average Idle Time': average_idle_time,
                    'Median Idle Time': median_idle_time,
                    'Mode Idle Time': mode_idle_time,
                    'Maximum Movement Time': max_movement_time,
                    'Minimum Movement Time': min_movement_time,
                    'Maximum Idle Time': max_idle_time,
                    'Minimum Idle Time': min_idle_time
                })
            csvfile.flush()

    except KeyboardInterrupt:
        pass

    print(f"Program stopped. The CSV file is saved as {os.path.abspath(filename)}.")
    serport.close()

df = pd.read_csv(filename)

excel_filename = filename.replace('.csv', '.xlsx')
df.to_excel(excel_filename, index=False)

wb = load_workbook(excel_filename)
ws = wb.active

for col in ws.columns:
    max_length = 0
    column = col[0].column_letter
    for cell in col:
        try:
            if len(str(cell.value)) > max_length:
                max_length = len(cell.value)
        except:
            pass
    adjusted_width = (max_length + 3)
    ws.column_dimensions[column].width = adjusted_width

wb.save(excel_filename)

print(f"Program stopped. The Excel file is saved as {os.path.abspath(excel_filename)}.")
```

IX. References

[1] "How HC-SR501 PIR Sensor Works & How To Interface It With Arduino," Last Minute Engineers, 3 July 2018. [Online]. Available: https://lastminuteengineers.com/pir-sensor-arduino-tutorial/. [Accessed 2 July 2024].

[2] "HC-SR501 PIR MOTION DETECTOR," [Online]. Available: https://www.mpja.com/download/31227sc.pdf. [Accessed 2 July 2024].

[3] R. a. Dinosaurs, "PIR sensor HC SR501," Youtube, 26 June 2021. [Online]. Available: https://www.youtube.com/watch?v=2dZ4cfluTTU. [Accessed 2 July 2024].