**CSC 453 Course Project:**

**Sleep Tracker**

CSC 453 - Spring 2021

Due: 4/25/2021

**Group Members:**

Shengdong Chen

Alex Grisham

Matt McLaughlin

Bilal Mohamad

Yizhuo Wu

## Group Member Contribution Breakdown

| Component | Component Weight | Student 1:  Shengdong | Student 2:  Alex | Student 3:  Matt | Student 4:  Bilal | Student 5:  Yizhuo |
| --- | --- | --- | --- | --- | --- | --- |
| Hardware connection and testing | 0.15 | 35.5% | 0% | 17% | 12% | 35.5% |
| Backend coding(Pi) | 0.35 | 45% | 0% | 5% | 5% | 45% |
| Frontend coding(Interface) | 0.35 | 0% | 10% | 45% | 45% | 0% |
| Report Writing | 0.15 | 3% | 89% | 0% | 5% | 3% |
| Per student aggregate contributions | | (35.5 \* .15 ) + (45 \* .35)  + (0 \* .35)  + (3 \* .15) = 21.525 | (0 \* .15 ) + (0 \* .35)  + (10 \* .35)  + (89 \* .15) = 16.85 | (17 \* .15 ) + (5 \* .35)  + (45 \* .35)  + (0 \* .15) = 20.05 | (12 \* .15 ) + (5 \* .35)  + (45 \* .35)  + (5 \* .15) = 20.05 | (35.5 \* .15 ) + (45 \* .35)  + (3 \* .15) = 21.525 |

### 

### Introduction

For this project, we chose to create a sleep tracker. We picked this project because in our research, this project seemed like a good way to try out components that we hadn’t used previously, as well as build upon the things that we’ve learned so far this semester. The goal of this sleep tracker is for users to be able to activate the device before going to sleep so that the device can measure the amount of noise and motion in the room. By providing these pieces of information through a display, the user will have an idea of how active they are during sleep.

### Design

SleepTracker is using a Raspberry Pi with motion and sound sensors to detect users’ actions to then publish the responses to the broker. Then the interface as a subscriber which subscribed the required topics from the local broker can display the message received to users. Diagram 1 shows the high level architecture of Sleep Tracker.

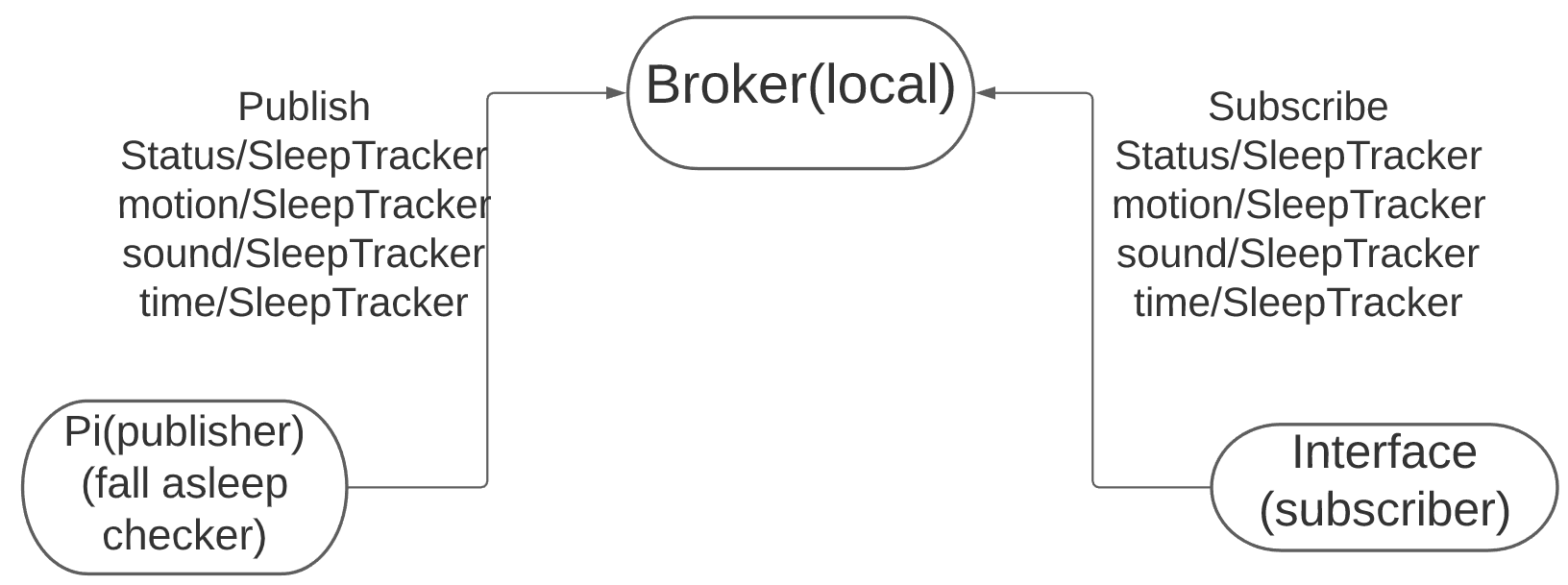


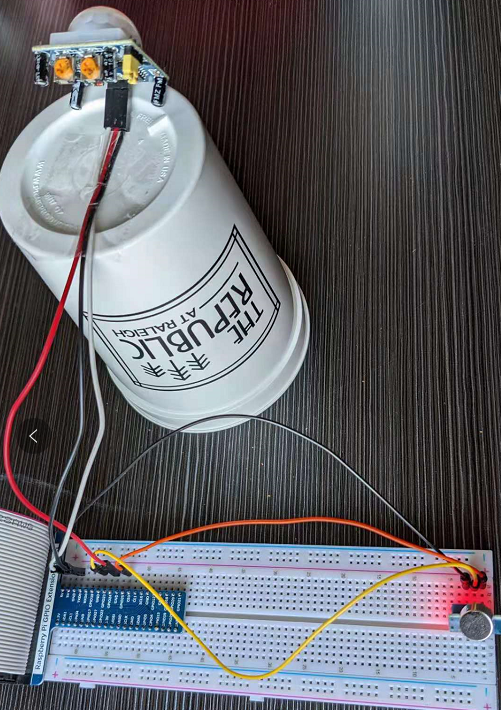
Diagram 1:

( High Level Architecture )

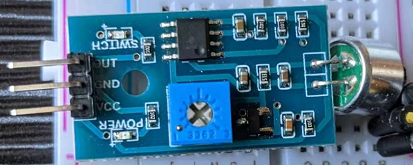
SleepTracker should start whenever the sensor.py file on the Raspberry Pi is running, there is an algorithm that checks if the user has fallen asleep, if so, the timer and counters start. All the data would be published once the sensor.py terminated.

Interface would display the total sleep time and the numbers of motions and sounds during sleeping.

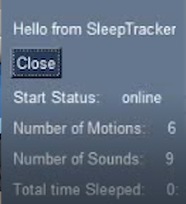
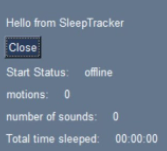
### Implementation

To implement the SleepTracker, we first have built the circuit for Raspberry Pi to run the program for sensors. We used one PIR motion sensor and one high-sensitivity sound sensor to connect with the raspberry pi. Which has shown in the picture. To interact with the broker, we have written the program, sensor.py, to get input from both sensors. In the program, we mainly have three parts to implement our function.

The first part of the sensor.py is the algorithm to detect whether the user has actually fallen asleep. To implement this function, the program has a countdown timer with detection of motion. During the actual usage, the timer would start running a 5-minute loop with the motion sensor when the program started. If no motion was deleted in a 5-minute loop, then the program will start the part to detect and log all sounds and motion. Otherwise, the timer will continue to loop until no motion is deleted, which indicates that the user has fallen asleep.

The second part of the sensor.py is the main program to connect with the broker and get input from sensors. To record every input, we have variables to count the number of sounds and motions detected. During the actual implementation, we also have changed the sensitivity of both sound and motion sensor to get the input in a more accurate way, which has shown in the picture.

The third part of the sensor.py is the ctrl-c handler, which also has the function to send data to the broker. We used paho.mqtt library to help us publish the datas. In the program, we have three variables to send, which are the number of motions, the number of sounds, and the time of total sleeping. In addition, the program will also have two log files of sound and motion sensors with timestamps recorded in the Raspberry Pi. To send all three data to the interface, users need to terminate the program, which indicates that the users have awoken, and then all three variables will be updated in the interface.

For our interface.py, we have mainly implemented it with PySimpleGUI library due to its simplicity. The interface also has the paho.mqtt library to subscribe to all datas, which are status of the sleep tracker, the number of motions, the number of sounds, and the total sleeping time. Thus, once the interface is connected to the broker and the sensor program is running on the raspberry pi, users can go to sleep with all data recorded in the raspberry pi, and after terminating the program, users can access all data in the interface, which can be run in a different machine.

### Results and Discussion

In regards to the experimental results of our project, there is not data that would be sufficient to demonstrate through graphical charts and tables for discussion; however, there were a few interesting observations that can be made about the results of the project overall. The motion sensor connected to the Raspberry Pi uses an infrared sensor to determine motion of the person sleeping, but if a pet, such as a cat or dog, were to move within the range of the motion sensor, the sensor would detect movement as the pet produces heat. This could potentially lead to inaccurate results for anybody that has pets. Similarly, this can apply in terms of the sound sensor such that any foreign noises aside from the noises made by the person sleeping can result in accurate results. One factor that prevented precise measurements of the sleep the user has undergone is the lack of sensors that have more detailed information. The microphone and motion sensor used for this project determine whether a certain threshold has been surpassed or not. They do not provide the level of sound being received or the intensity of the movement. These limitations are what hindered moving forward with the deep sleep aspect of the project, which was removed from the requirements due to the scope of the project. Utilizing mere checks of whether the sensors are being triggered with specific metrics proved insufficient for attempting to measure the fluctuations of deep sleep precisely.

### Related Work and References

During our research and early designing stages, we used a wide variety of websites in order to help us choose this project and come up with the necessary circuitry for it. Some of these websites include motion and sound detector connections from *Raspberry Pi Projects* as well as other websites like *Towards Data Science* to help pick out the project. The links to all references are below:

<https://www.instructables.com/Sound-Sensor-Raspberry-Pi/>

<https://projects.raspberrypi.org/en/projects/parent-detector>

<https://towardsdatascience.com/create-your-own-smart-baby-monitor-with-a-raspberrypi-and-tensorflow-5b25713410ca>

<https://www.hackster.io/kuzma/walabot-sleep-tracker-472740>

### 