Introduction to Intelligent Robot Control

FINAL PROJECT REPORT

Ryan Jabbour-Laura Rodríguez López-Maxime Hanus

Motivation

In a world where technology seamlessly integrates into every aspect of our lives, our team embarked on a journey to redefine the wake-up experience for users with the creation of a smart alarm clock robot. Our project combines the use of robotics, human robot interaction, and personalized data tracking to offer a truly unique and engaging wake-up solution for people.

Our primary focus is on catering to individuals who face difficulties waking up, whether due to deep sleep patterns, irregular sleep schedules, or a general lack of motivation in the early hours. They tend to cause themselves stress in the morning for them not to be late so they put several consecutive alarms in order for them to wake up. The smart alarm clock robot is designed to be a personalized and engaging companion that addresses the unique needs of these users offering a new approach to their morning habits.

This smart alarm clock robot is not just an ordinary bedside timekeeper; it is a sophisticated piece of technology that interacts with users in an innovative and dynamic manner. Instead of the traditional snooze button, our robot engages users in a stimulating wake-up challenge by emitting sound until a specific pattern, indicated by the LEDs, is input using a set of buttons. This gamified approach not only adds an element of fun to the waking-up process but also promotes mental alertness from the very start of the day.

One of the standout features of our smart alarm clock robot is its ability to keep track of user statistics. Users can access this information through a user-friendly interface using an LCD screen and a joystick in order to navigate through it. With that, users are able to gain a deeper understanding of their sleep habits and make informed decisions to optimize their daily routines.

This final project report delves into the design and construction process of the smart alarm clock robot as well as the data collection and its analysis in order to take out reviews and future suggestions from users. We'll explore the challenges faced, the solutions devised, and the overall impact of our creation.

Design and Construction Process

In this section, we will provide a breakdown of our alarm clock robot's design and construction process. The project is systematically divided into three core components: logical design, programming, and physical construction. The logical design is dedicated to understand and incorporate the user requirements, ensuring our robot meets the needs and preferences of the final users. The programming segment involves the codebase of our alarm robot. We will illustrate the interaction between different elements of our robot, as well as emphasize on the importance of a well-crafted and efficient code. On the physical side, we cover material selection, assembly issues we encountered, and the construction of an aesthetically pleasing and user-friendly robot.

When it comes to **logical design**, the first step is to clearly define the nature of the human-robot interaction. This involves determining actions such as waking up the user with a buzzer sound and displaying a random LED sequence, or monitoring whether the user woke up on time. Essentially, we must identify the fundamental use cases our alarm clock robot aims to handle. For instance, our primary use case involves users successfully waking up. In order to achieve this, end-users interact with the robot by setting an alarm for a specific time. Upon activation, the alarm robot wakes the user with a buzzer sound, showcases a random LED sequence, and awaits the user to press buttons in the displayed order. Once the user turns on the lights, the buzzer ceases, and the robot congratulates them on waking up.

Additionally, our alarm robot considers additional scenarios. What if the user enters the wrong sequence? Is there a time limit for the alarm? What if the user takes too long to wake up? What if they decide not to wake up at all? As a result, the logical design involves establishing each possible case and defining the parameters of the human-robot interaction.

In the **programming** stage, we convert the initial requirements of our robot into actual and functional code. In this phase, it is crucial to define what we can realistically achieve with our current materials. Therefore, as we progress through the programming process, we assess the feasibility of our robot and identify the codebase behind our desired functionalities. The primary requirement of our robot was clear: the alarm system.

In the initial interaction of our robot, users encounter the default screen. This screen displays the current date in DD-MM-YYYY format, along with the current hour and minute in HH:MM (24-hour format). Moreover, users will see the alarm time that they have previously entered, complemented by a clock icon that provides a visual cue to interpret the leftmost number. At first glance, users are able to recognize our robot as an ordinary digital clock, but we would like to go beyond this surface perception. As a result, we added some gamification components. On the default screen, we introduce the concept of the current alarm streak. The streak number represents the consecutive number of times the user has successfully woken up to the alarm. We will get into the specifics later on.

To delve into technical details, the default screen is displayed in the main loop function of our Arduino code. At regular intervals, the loop iterates and checks whether the current hour and minute match the preset alarm hour and minute. If not, the default screen function is invoked, where we have defined all of the elements explained above. Please, see below a snippet of the code:

```
void loop() {
  // get current time
  RtcDateTime now = Rtc.GetDateTime();
  // check if it's alarm time
  if (now.Hour() == alarmHour && now.Minute() == alarmMinute && now.Second() == 0) {
    // alarm code
  } else {
    defaultScreen(now);
  }
  delay(500);
}
```

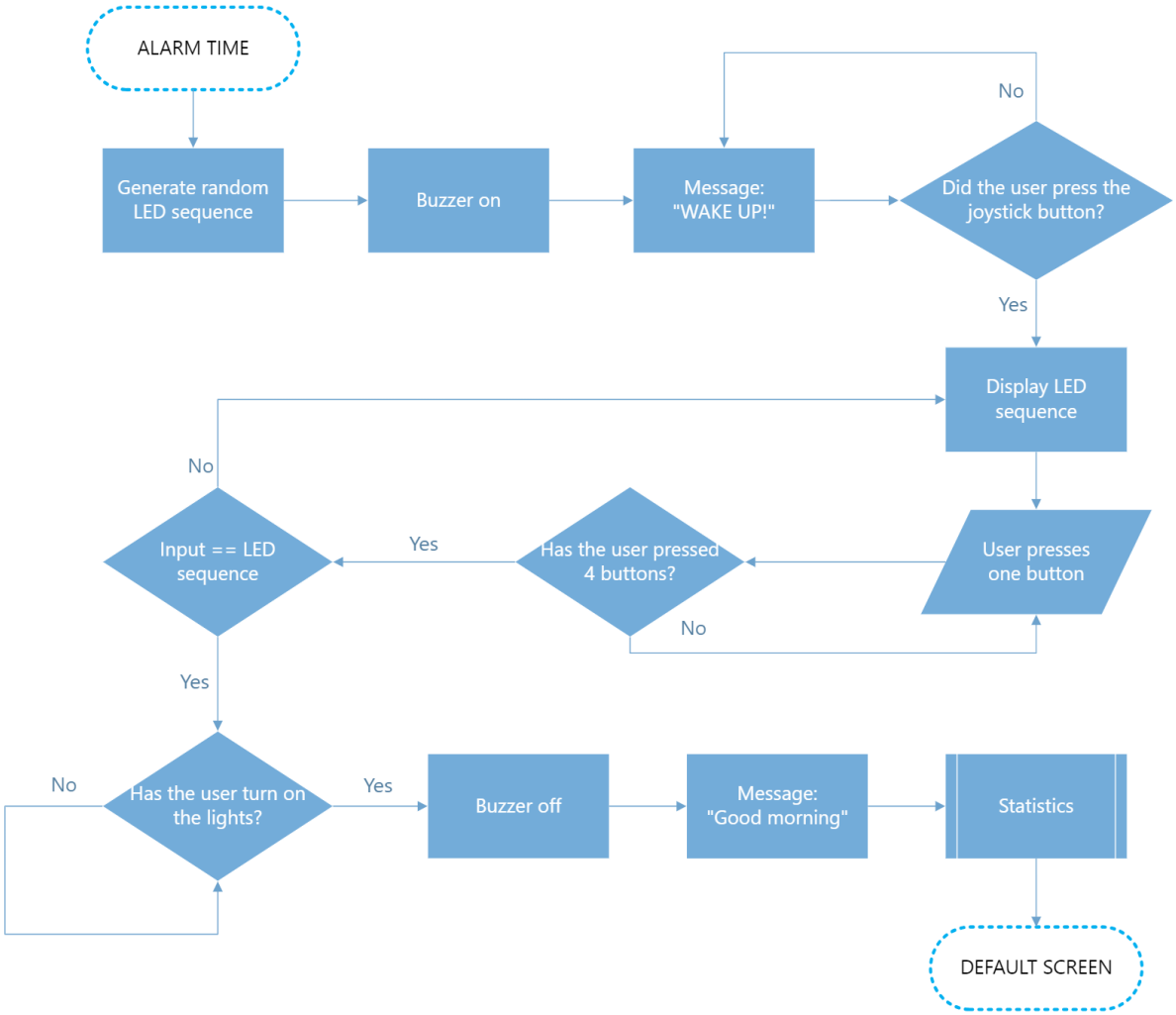
Furthermore, our interest went beyond constructing an interactive robot that engages users not just through sound but also through touch. We accomplished this by integrating a joystick, allowing users to press it for setting an alarm. Within our default screen function, we check whether the joystick button has been pressed. In that case, we transition to a separate function containing the code that allows users to set an alarm. If no movement is detected, the screen continues to display the default information. Please, see below a snippet of the code:

```
//check if user presses joystick
if (joystickButton == 0) {
 // joystick was pressed
 setAlarm();
}
```

In the alarm setting function, the human-robot interaction becomes more dynamic as we aim for the user to input data and our robot to process it. Initially, we provide on-screen

messages to guide the user on the expected input and explain how the data will be processed. To set the alarm, we establish a default alarm time at 00:00. The user is then required to manipulate the joystick: moving it upwards increases the hour/minute, and moving it downwards decreases it. Pressing the joystick button confirms the desired value. Once the user successfully inputs the alarm minute, the alarm clock robot stores this information in the EEPROM memory of the Arduino Uno. This step is crucial as it enables the robot to retain the alarm time even when powered off. When the smart alarm robot is powered on, it retrieves the stored alarm time from the EEPROM memory and displays it on the screen. However, It's important to note that while these technical details are integral to the functionality, the end user is unaware of them. For the users, the interaction involves a simple message notifying the alarm time has been successfully saved. After this step, the alarm clock robot reverts to the default screen.

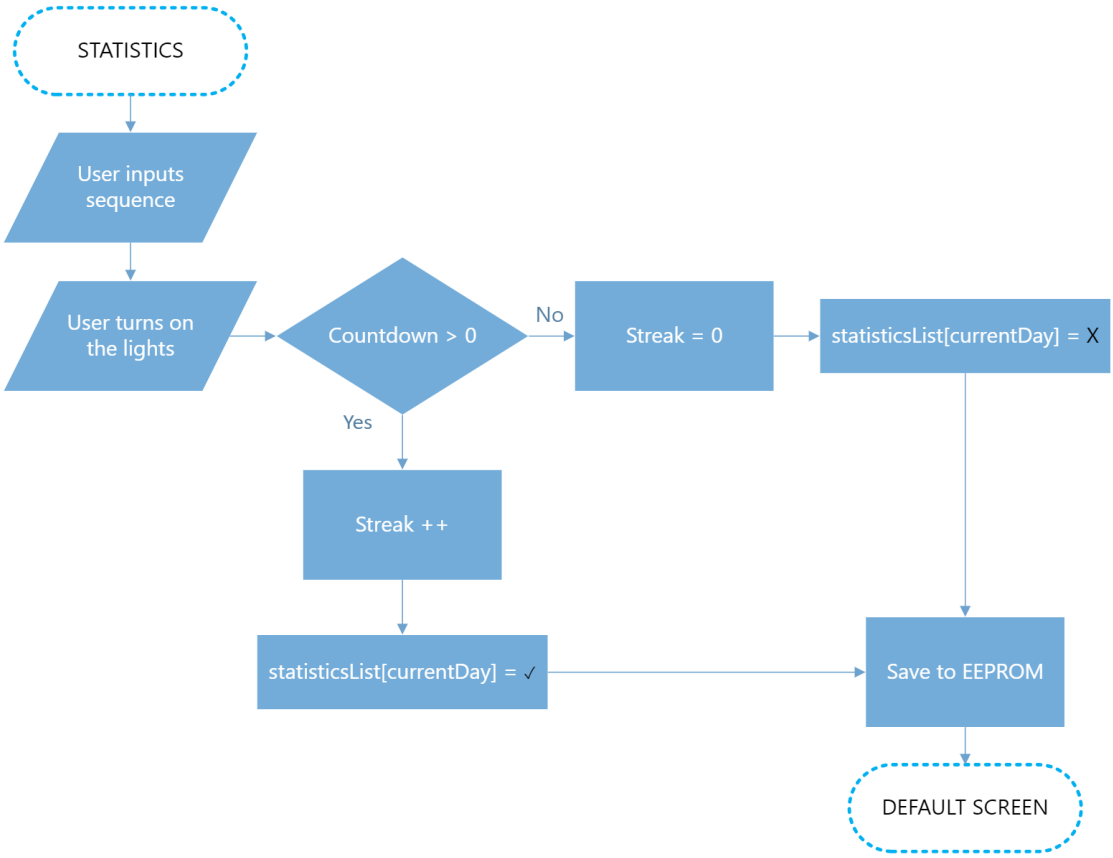
Up to this point, we have covered the default screen and the process of setting an alarm. Now, let’s delve into our robot’s most important functionality: the alarm itself. When it’s time for the alarm to go off, the robot executes the code defined in our alarm function. The steps can be found in the flow diagram below.



Moreover, as previously mentioned, we have incorporated some gamification components to enhance user engagement. Initially, users have 60 seconds to input the correct sequence, and this countdown is visibly displayed on the screen at the time of the alarm. While it may seem like the robot is concurrently managing the LED sequence and the countdown, these sections of the code do not run in parallel due to constraints with our Arduino Uno. To create the illusion of simultaneous execution, we made use of the millis() function. In addition, another gamification element is the streak variable. The logic behind the streak is straightforward: users must correctly input the sequence and turn on the lights **before** the 60-second countdown expires. If successful, the alarm robot increments the streak variable. If, on the other hand, the user fails to complete the tasks in time, the streak resets to 0. These gamification components establish a deeper connection between

the user and the robot, with the primary goal of encouraging consistent use of the alarm clock as part of the daily morning routine. Striking a balance between an enjoyable interaction through gamification and maintaining a level of challenge is crucial.

In our previous flow diagram, we introduced the concept of statistics. To clarify, when we refer to statistics, we are addressing a specific requirement for our alarm clock robot, the integration of statistical data. One of the elements contributing to these statistics is the streak variable, previously discussed in the context of gamification. However, in addition to the streak variable, we have incorporated an additional set of statistics. With our alarm robot, users can assess their wake-up efficiency. The mechanism works as follows: when users successfully turn off the alarm (input the correct sequence and turn on the lights) **before** the countdown reaches zero, the robot records this achievement in a dedicated statistics variable. If, on the other hand, the user fails to wake up on time, the robot also registers this occurrence. From a technical perspective, the alarm robot utilizes a statisticsList variable to manage this statistical data. The logic behind this process can be found in the flowchart below.



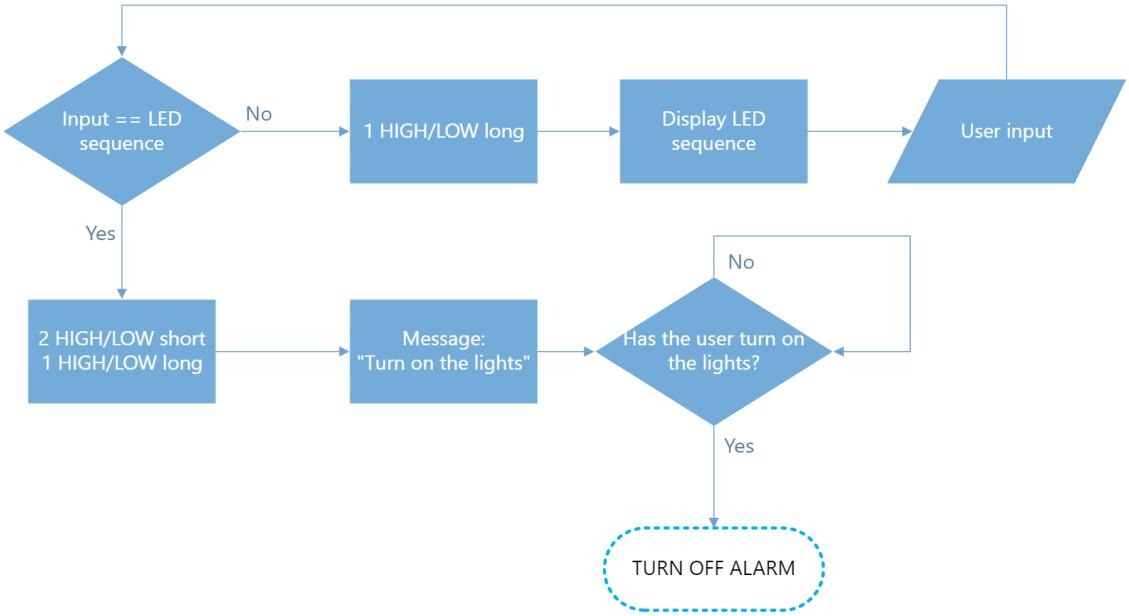
In the statisticsList variable, three possible values exist: no alarm set (O), waking up on time (✓), or waking up late (X). This variable contains seven entries, each corresponding to a day of the week. The default value is O. Upon setting an alarm, the robot follows the previous logic to determine whether the user woke up on time (✓) or not (X). Leveraging the Real Time Clock (RTC), the alarm clock robot identifies the current day of the week and updates the corresponding value in the statisticsList. It's crucial to note that this list is stored in the EEPROM memory of the Arduino Uno board, ensuring that values persist upon powering on the alarm clock robot.

Furthermore, how do users access their data? When the alarm clock displays the default screen, it not only checks if users press the joystick button to set an alarm, but also if the user moves the joystick downward. Once movement is detected, a function is called to display the statistic screen. In this screen, users can easily view the list values in a table-like format, providing an immediate overview of their data. By incorporating this feature, we enhance human-robot interaction, fostering a stronger connection between users and the robot. Apart from viewing the tracked data, there is not much else to do on this screen. To return to the default screen, users simply move the joystick upwards.

Let's address one more question. What if the user decides they do not want to wake up? In this scenario, the user does not really have an alternative. That is, the buzzer will only cease once the user successfully inputs the correct sequence and turns on the lights. The reasoning behind this design choice stems from our primary objective: encourage users to wake up. Our robot is specifically targeted towards individuals who struggle waking up, and thus, we have chosen a very strict boundary that will force users to wake up, even if they might not feel like it. As a final recourse, the alarm will invariably stop if we don't power the robot!

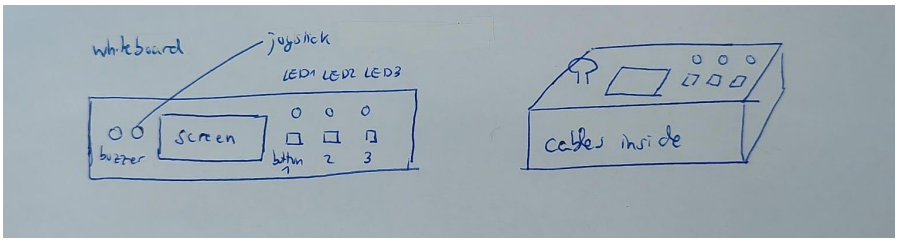
When it comes to **testing** the alarm code, we had to perform numerous iterations to ensure that the alarm robot accurately responded to all proposed use cases. By adopting the role of users, we aimed to pinpoint the appropriate delay values in the code and redefine how certain information was displayed. For instance, a notable challenge surfaced

during unit testing, specifically concerning the LED display. When showcasing the LED sequence, we noticed situations where it was not clear what followed after the user input a sequence using the buttons. If there was a match (generated sequence = input sequence), the robot waited for the user to turn on the lights without prompting them to do so. On the other hand, when there was not a match, the alarm robot quickly displayed the generated LED sequence again, leading to confusion for the user. To address this issue, we redefined the behavior of the LED display in the following way:

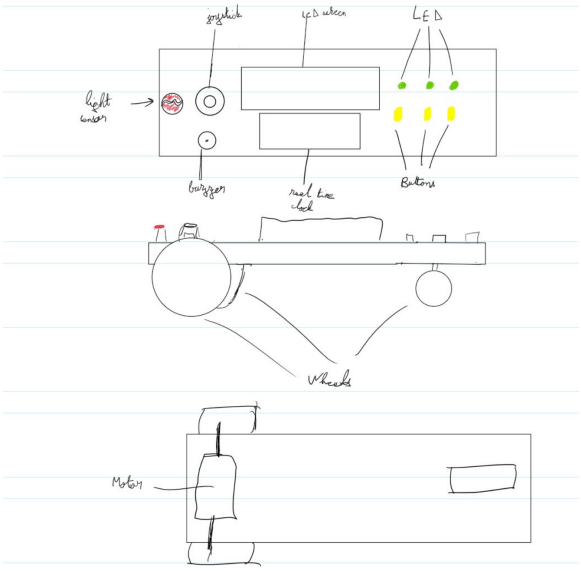


We performed many more use cases, and once we were satisfied with the alarm clock robot behavior, we focused on the **physical design and construction**.

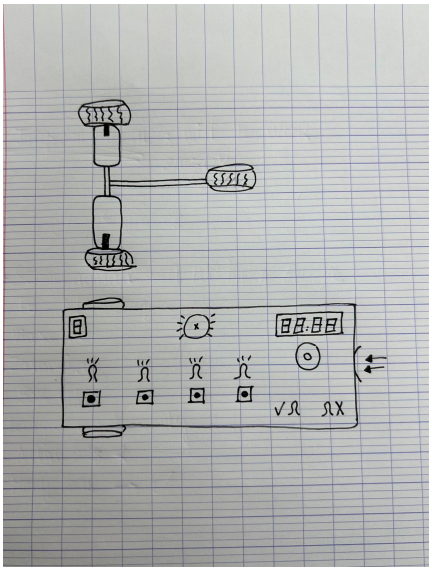
Before starting this process, the group decided that each team member would propose a robot design in order to later on choose the one we like the most. We believed that brainstorming many potential designs could lead to a more creative robot design since we could mix some ideas together.



In the first preliminary sketch, no wheels were taken into account. The robot would take a rectangular shape mainly due to the shape of the breadboard we're using. The idea was to assemble all the circuit inside the robot, and then all the interactive components will be displayed on the interface with the user.



In the second sketch drawn, we still have the basic same design with the same components used but the difference here is the addition of wheels in order to put in place the motion mechanism we initially wanted to implement. Two wheels at the back are operated by one step motor in the middle that coordinates their rotation.

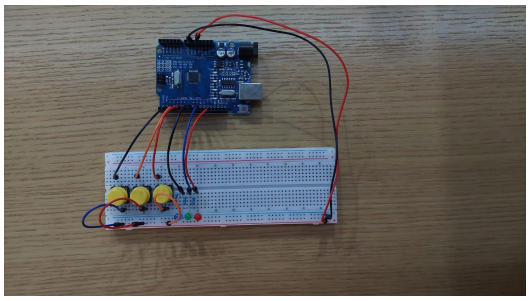


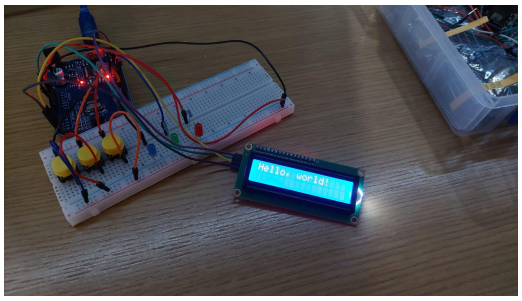
For the third and final sketch, we still had the same design approximately as the other drawings but a different distribution was set for the interactive components on the interface. The wheel structure here is different too: two step motors are used, one for each wheel. Moreover, a new idea came up in this sketch which is to add a screen that could gamify the robot by adding a score and keeping track of a streak in order to measure the robot's efficiency and the user's adaptability to it, considering he is seeking more energetic habits.

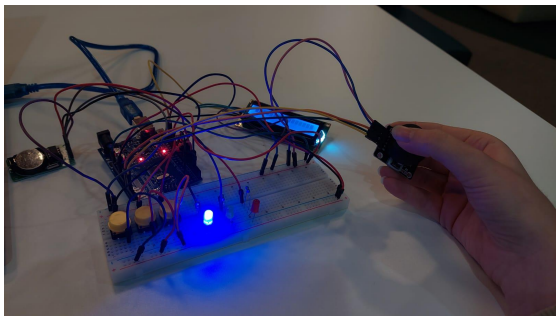
At the end, for the overall design of the robot, we stuck with the idea of the rectangular box helping us fit the breadboard in without having any issues. For the distribution of the components, it depended on our ability to connect them to the breadboard and the arduino using the right ports, so they were on-the-spot decisions.

For the motor and the wheels, we talked about it as a team and have decided to remove this part from our project and put more options and content on the LCD screen, because moving the robot could get complicated afterwards and we didn't find a real use of the movement in this alarm clock.

First step we had to take was to test all the components (buttons, LEDs, LCD screen, joystick) and connect them to the breadboard to check their functionality one by one with our code: having the right cable connections with the arduino, using the right resistors with their respective right resistances... Then, it was combining all of the code together and testing all of the components while connected. When successful with those steps, we moved to the aesthetic part of the construction process.







We started by searching for a box that respected the dimensions we were looking for: not too small in order to fit the breadboard and the cables in, and not too big in order not to exaggerate the robot's design. Then, we made holes around the box for the LCD screen, the USB cable, the joystick, and the LEDs, and fixed each component in their respective place.

A problem we encountered was how would we put the buttons on top of the box while still leaving them connected to the breadboard. After throwing a bunch of ideas together, we found the solution: a set of straws that fit perfectly on top of each button and that would go out of the box's higher part where their tips were used as alternative buttons.

I) Data collection

To collect our data, we began by studying a larger survey to see if our tester sample was reliable. Next, we asked a number of people to complete a survey without using our alarm for a week, before submitting the questionnaire to them again, but this time they had to use our alarm.

This approach aims to determine how using our alarm affects the answers provided in the survey.

1) Comparison with a larger survey

a) Importation of the data

In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
df_sleep = pd.read_csv("sleep_dataset/user_survey_2020.csv")
df_sleep = df_sleep[["sleep","amEmotion","amCondition"]]
df_sleep.dropna(inplace = True)
df_sleep.reset_index(inplace = True, drop = True)
df_sleep
```

Out[2]:

	sleep	amEmotion	amCondition
0	4.0	4.0	3.0
1	3.0	3.0	2.0
2	3.0	3.0	3.0
3	4.0	3.0	4.0
4	4.0	4.0	2.0
...	...	...	...
610	4.0	4.0	4.0
611	4.0	3.0	4.0
612	4.0	4.0	4.0
613	2.0	3.0	2.0
614	3.0	2.0	2.0

615 rows × 3 columns

This dataset represents the quality of the sleep. All the number are grades between 0 and 5.

- "sleep" represents the quality of the sleep
- "amEmotion" is the feeling after awakening (A. 1 (Very unpleasant), 2 (Unpleasant), 3 (Moderate), 4 (Pleasant), 5 (Very pleasant))

- "amCondition" is the refreshing feeling at the awakening (A. 1 (Not at all), 2 (Not much), 3 (Moderately), 4 (Fairly), 5 (Fully))

2) Data collection from our testers - without the alarm clock

a) Importation of the data

We asked 3 people to answer our survey when they wake up. Our survey is composed of these questions :

1. **Are you satisfied with your sleep? (Subjective sleep quality score)**
   - A. 1 (Not at all), 2 (Not much), 3 (Moderately), 4 (Fairly), 5 (Fully)
2. **How do you feel after awakening? (Emotional condition)**
   - A. 1 (Very unpleasant), 2 (Unpleasant), 3 (Moderate), 4 (Pleasant), 5 (Very pleasant)
3. **Do you feel refreshed after awakening? (Physical condition)**
   - A. 1 (Not at all), 2 (Not much), 3 (Moderately), 4 (Fairly), 5 (Fully)

Thanks to these we get this data :

In [3]:
```
our_data_no_alarm = pd.read_csv("sleep_dataset/data_without_alarm_clock.csv")
our_data_no_alarm["num_day"] = (our_data_no_alarm.index+1)%7
our_data_no_alarm["num_day"].replace({0:7}, inplace = True)

our_data_no_alarm
```

Out[3]:

	user	sleep	amEmotion	amCondition	num_day
0	1	3.0	3.0	2.0	1
1	1	2.0	3.0	4.0	2
2	1	4.0	3.0	4.0	3
3	1	4.0	4.0	2.0	4
4	1	4.0	4.0	4.0	5
5	1	4.0	3.0	2.0	6
6	1	4.0	3.0	3.0	7
7	2	4.0	3.0	3.0	1
8	2	4.0	3.0	2.0	2
9	2	2.0	4.0	4.0	3
10	2	4.0	4.0	4.0	4
11	2	3.0	3.0	2.0	5
12	2	2.0	2.0	2.0	6
13	2	3.0	3.0	2.0	7
14	3	4.0	2.0	3.0	1
15	3	4.0	2.0	4.0	2
16	3	4.0	2.0	2.0	3
17	3	4.0	3.0	2.0	4
18	3	3.0	3.0	2.0	5
19	3	4.0	5.0	4.0	6
20	3	3.0	3.0	2.0	7

The column user represents the id of the user of the alarm clock and num_day is a column representing the number of day the user utilize the alarm clock. The sleep, amEmotion and amCondition columns represent the same thing as the previous dataset.

3) Data collection from our testers - with our alarm clock

a) Importation of the data

We interrogate 3 people and gave them the robot for 3 nights each. We have add questions at the end of their trial :

- Would you keep this alarm clock ? Yes or No
- Would you use it regularly ? Yes or No
- Would you recommend it to someone who has trouble awakening ? Yes or No
- Do you have any feedback ?

We get this data :

In [4]:
```
our_data_alarm = pd.read_csv("sleep_dataset/our_data.csv")
our_data_alarm["num_day"] = (our_data_alarm.index+1)%3
our_data_alarm["num_day"].replace({0:3}, inplace = True)
our_data_alarm
```

Out[4]:

	user	sleep	amEmotion	amCondition	num_day
0	1	4	2	5	1
1	1	2	1	3	2
2	1	4	3	4	3
3	2	5	3	5	1
4	2	3	2	4	2
5	2	2	3	4	3
6	3	3	3	4	1
7	3	4	3	5	2
8	3	5	5	5	3

The end of test question yielded these results :

1. User 1 :
    - Yes
    - No
    - Yes
    - I think this alarm clock is interesting, I could use it but not regularly. I could use it before an exam for example to be mentally awake but not everyday.
2. User 2 :
    - No
    - No
    - Yes
    - I have no difficulties to wake up so this robot just put me in a not really good mood for no particular reason. But I totally understand it can help people with difficulties to wake up.
3. User 3 :
    - Yes
    - Yes

- Yes
- I don't like the sound of the alarm but the robot completed its tasks to wake me up fast and make me feel ready for the day.

II) Analysis of the data collected

1) Analysis of the larger survey

In [5]:
```
pd.df_sleep[["sleep","amEmotion", "amCondition"]].mean()
```

Out[5]:
```
sleep          3.256911
amEmotion      3.058537
amCondition    2.663415
dtype: float64
```

We can see that, on average, people are so so when they wake up, and don't feel much awake.

In [6]:
```
df_sleep.groupby("sleep").mean()
```

Out[6]:

sleep	amEmotion	amCondition
1.0	2.520000	1.560000
2.0	2.633929	2.071429
3.0	2.919192	2.545455
4.0	3.266667	2.941667
5.0	4.025000	3.925000

We can see the mean values of the amEmotion and amCondition by the quality of sleep for people who don't use our alarm clock.

We can now compare these results with the ones we collected to see if our data is representative.

2) Analysis of the data collected

a) Without the alarm clock

In [12]:
```
our_data_no_alarm[["sleep","amEmotion", "amCondition"]].mean()
```

Out[12]:
```
sleep          3.476190
amEmotion      3.095238
amCondition    2.809524
dtype: float64
```

We can see that, on average, people are so so when they wake up, and don't feel much awake like in the bigger survey. This means our data is reliable.

In [8]:
```
our_data_no_alarm[["sleep","amEmotion", "amCondition"]].groupby("sleep").mean()
```

Out[8]:

sleep	amEmotion	amCondition
2.0	3.000000	3.333333
3.0	3.000000	2.000000
4.0	3.153846	3.000000

Here are the average values of amEmotion and amCondition by the quality of their sleep for people we asked to answer our survey without our alarm clock.

b) With the alarm clock

In [13]:
```
our_data_alarm[["sleep","amEmotion", "amCondition"]].mean()
```

Out[13]:
```
sleep          3.555556
amEmotion      2.777778
amCondition    4.333333
dtype: float64
```

We can see that, on average, people feel a bit less good when they wake up, but feel much awake when they use our alarm clock.

In [10]:
```
our_data_alarm.drop(columns = ["user", "num_day"]).groupby("sleep").mean()
```

Out[10]:

sleep	amEmotion	amCondition
2	2.000000	3.500000
3	2.500000	4.000000
4	2.666667	4.666667
5	4.000000	5.000000

We can see people feel more awake thanks to our robot than people who don't use it but they are in a less good mood than them.

In [11]:
```
our_data_alarm.drop(columns = ["user"]).groupby("num_day").mean()
```

Out[11]:

num_day	sleep	amEmotion	amCondition
1	4.000000	2.666667	4.666667
2	3.000000	2.000000	4.000000
3	3.666667	3.666667	4.333333

They also seem less bothered on the third day than the first. They are probably getting used to the alarm clock.

III) Conclusion

To conclude, after this analysis, we can say that our robot's objectives have been achieved. Indeed, we wanted to create something to help people wake up in good conditions to start the day, and the data and user feedback allow us to say that it's a success. However, we can surely still improve on this idea by working on the user experience so that people's good moods are less affected.