Blink 1.0

Final report Engineering Design (4WBB0)

November 1, 2021

Group no: 237	
Name	**Student ID**
Jeroen Beltman	1603442
Lisa Gerrits	1621718
Tom van Liempd	1544098
Thomas Toorenman	1639412
Gijs Vogels	1577514

1 Group effectiveness

To begin, an advantage is that the group contains five group members who all follow a different major. These majors are: Mechanical Engineering, Medical Sciences and Technology, Computer Science, Automotive Technology and Industrial Design. Therefore, there are a lot of different qualities in the group and this can be very useful for a project. Some of the qualities can be used for the design, some for the electrical or programming part and some for making a planning and writing the report.

1.1 First weeks of the project

Coming up with ideas is a strength of the group, because everyone came up with two or three ideas at the beginning of the project and therefore the design goal and an idea was chosen very soon.

Then, a planning for the project progress was made, so that everyone knew all the different phases and deadlines of the project. The second phase of the project was the concept phase. There was a lot of creativity in the group, so the concept was made very soon.

1.2 Designing

After the design goal was chosen and the concept was finished, the designing phase started. Some group members were good in designing, so there were not many designing issues. Sensors and actuators were chosen and then the design was made, so everyone got a good image of how the product is going to look like. Designing is definitely a strength of some group members.

There was also some knowledge about microcontrollers and programming for example, so that made it a bit easier to know which microcontroller was needed for the product and how all the parts should be programmed as well.

1.3 Last weeks of the project

When the detailed design was finished, some parts needed to be 3D printed. One group member already had some experience in 3D printing, so the parts were sent and 3D printed successfully.

There was also some knowledge about electrical equipment. Some parts needed to be soldered and although there was not a lot of experience with soldering, this turned out very well in the end. This is because a quality of the group is that everyone was motivated to learn and help with (new) things.

When the product was realized, the testing phase started. The product is tested during the meetings, but also with other users. The testing phase went quite well.

1.4 Improvements and strengths

Halfway through the project, the group noticed that the same group members kept working on the same tasks, so it was decided to change that. For example, make sure a group member who works almost exclusively on the report also helps with the designing, programming or electrical part. Then, every group member is involved in more than one task in the project. For the next group project, this is something that can be improved. Namely, it should be better that every member is working on more tasks already at the beginning of the project and not only from halfway through the project, because if a group member is doing several different tasks, he/she can learn even more from the project.

When the group ran into a problem, everyone was immediately thinking about solutions and then the group could continue working on the product, when there were problems, the group did not lose much time. Also, every group member followed the planning of the project very well and therefore the deadlines such as the intermediate presentation, the group evaluations and the final presentation were finished on time.

Overall, the group members collaborated very well and were motivated to make a good and original new product. With all the mentioned knowledge of the group, but also with things that could be improved for the next time, the project has been completed successfully.

2 Design goal

Approximately 13 percent of the Dutch population aged 40 years and older have a hearing loss of more than 35 decibels. This means that about 1.2 million people in the Netherlands have hearing impairments [1], 1.2 million is already a large number, but just imagine how big this number is in the whole world. According to estimates of the World Health Organization (WHO), more than 700 million people worldwide would have severe hearing loss in the year 2050 [2]. Noise damage can cause hearing loss at a young age, but some people are born with a hearing impairment or are born deaf due to heredity. Hearing impairments can have a large impact on someone's daily life. Take for example their life at home. People have to deal with many in-house activities and domestic appliances every day, like the washing machine, the dishwasher and the doorbell.

Now imagine you are someone with a hearing impairment. When the doorbell rings, you are likely to not notice the sound, which can result in you missing a package or an appointment. When the timer of the oven goes off you can't hear the alarm which can result in burnt dinner. These are problems that people with hearing impairments have to deal with on the daily basis. Therefore, the design goal is to create an aid for people with a hearing impairment to get notified when a domestic appliance uses sound. This can be facilitated by replacing sound for notifications that require other senses like lighting and vibrations. The sounds from the domestic appliances will be picked up by some sort of sensor. If the design goal is reached, people should be able to recognize the notifications they would otherwise miss, and therefore give them an easier and more comfortable experience at home.

To get a clear and smooth start to the project the first step was to define a specific design question that goes as follows: How can a product be made that helps people with hearing impairment to get notifications when domestic devices make noise?

During the project a design process will be followed to answer this question and to hopefully fulfill our design goal.

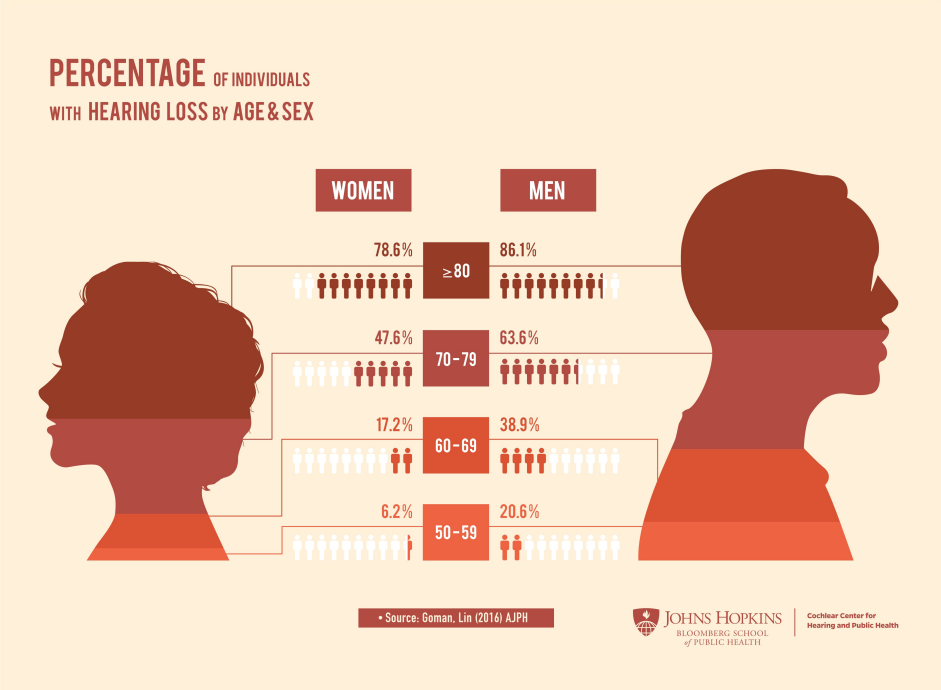

Figure 1: Percentage of elderly with a hearing impairment

3 Functional design and solutions

When designing a new product, there are a lot of specifications to think about. The functional specifications are all the specifications that are needed to create a working product for the user. These specifications are defined in the concept phase of the project.

The MoSCoW method that can be seen below is a method used to clearly state all the functional specifications, and will define what the design must have, should have, could have and won't have.

3.1 MoSCoW Method

The **MoSCoW** method is a good way to classify (functional and technical) specifications.

3.1.1 Must have

These are specifications which you basically have no choice, but to implement them into your product. These are often called constraints. The wristband that will be produced in this product also has some constraints. These constraints will be listed below.

- **The dimensions of the product must be such that it would fit the user's wrist comfortably**
  Solution: To make sure it is comfortable to wear, the product becomes as compact as possible and not too heavy to wear on the wrist.
- **The product must be made with the budget of 70 euros**
  Solution: An inventory of all the parts is made to make sure that 70 euros or less than that is spent.
- **The aid must function autonomously or interactively**
  Solution: A battery will be used to make the product autonomously. The product also interacts with the users. A flowchart is made to visualize the interaction between them:

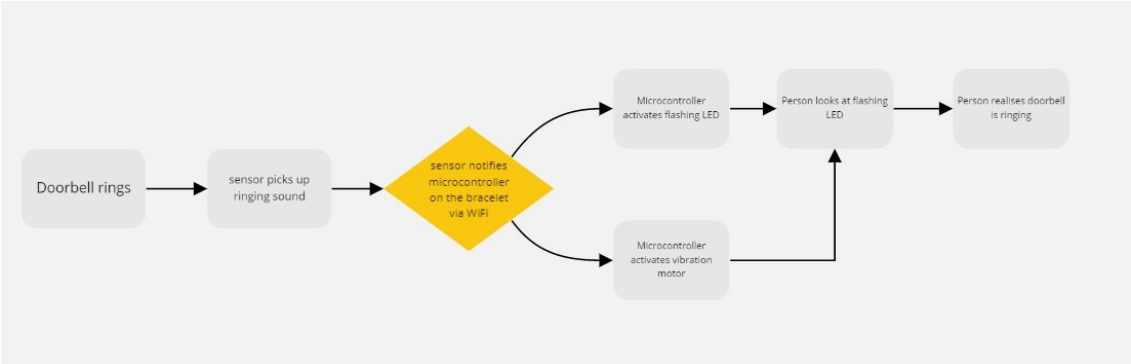

Figure 2: Flowchart

- **The product must have a vibration sensor and a sound sensor as sensor and a light and vibration motor as actuator**
  Solution: There are several sensors, for example sound and vibration sensors, to measure when for example the door bell rings. These sensors will notify the microcontroller via WiFi. Then, the microcontroller will activate the multicolour LED lamp and the vibration motor of the bracelet.

  Sensors: sound sensor at the doorbell and the fire alarm and vibration sensor for the washing machine/dish washer.

  Actuators: multicolour LED lamp and a vibration motor
- **The product must have a battery that complies with the five safety instructions**
  These safety instructions are the following:

  1. Do not make any modifications to the battery
  2. Use a voltage protection

3. Install a fuse in the circuit
4. Use the correct power supply when loading
5. Make sure all wiring is insulated

Solution: The product needs a battery in order to be wireless, so there is no other way than to follow the five safety instructions to make sure a battery can be used.

3.1.2 Should have

These are specifications you have to meet to really fulfill the customer's needs. These are often called requirements. The wristband that will be produced in this project also has some important requirements. These requirements can be seen below.

- **The bracelet should have an adjustable strap**
  Solution: A strap that is adjustable must be chosen/made, otherwise not all people can wear the bracelet.
- **The user should be able to turn off the actuator**
  Solution: If the user has noticed the notification because of the light and vibration, the user can turn it off by pressing a button on the bracelet.
- **The battery should be rechargeable**
  Solution: The bracelet should have a USB port in which a micro USB cable can be attached. This cable is used to charge the battery.
- **The product should be user-friendly**
  Solution: The product should be easy to use, so everyone can use it, also elderly. That's why a bracelet is chosen instead of just an app on smartphones. The product should be as compact as possible and easy to understand.

3.1.3 Could have

These are specifications which are nice to have, but functionality of the product is not compromised if they are not met. These are often called preferences. Also in this project, there will be some preferences and they will be listed below.

- **The product could have a feature to let the user configure the device**
  Solution: Design an app for the smartphone. The app should control the wristband. In the app you can set which colour the light will be when for example the washing machine is finished.
- **The product could give each object an unique vibration**
  Solution: It could be programmed that when the dishwasher is finished, the bracelet will vibrate 1 time, when the washing machine is finished, the bracelet will vibrate 2 times and so on for all the objects involved.

3.1.4 Won't have

These are specifications you already know you will not meet, and also do not need to meet. They are just there as a reminder for the future. In this project there are also specifications that will not be put into the design, but that could be implemented in the future. See the list below.

- **The product won't have display with touch screen**
  Solution: When the bracelet would have a touch screen, the multicolour LED lamp is not needed. The programming part needs to be changed when it is just a display with touch screen.
- **The product won't have more lights, instead of one light**
  Solution: The bracelet should be a bit bigger than, because otherwise there is not enough space for more lights. Also, each light should be linked to an object then.

4 Design concepts

Three different design concepts with the same design goal are made. These three different design concepts will be evaluated separately and then the best design concept will be chosen. The three concepts are the following: The smart wristband concept, the smart glasses concept, the hearing box concept. All concepts must have actuator(s) and sensor(s). Several illustrations of these concepts are made, to make clear what the object should look like.

4.1 Smart Wristband Concept

This concept is made for people with an hearing impairment. The smart wristband will help these people to notice when for example the doorbell, dishwasher or washing machine are ready. The actuators of the wristband will be a multicolour LED lamp and a vibration motor. Then, various sensors (sound sensors and vibration sensors) can be placed around the house, to trigger these actuators. The LED lamp and the vibration motor will be triggered when for example the dishwasher is finished. Every object can get its own colour, so the user can easily know which object they need to pay attention to. The vibration of the wristband is an extra tool to make for 100 percent sure the user does not miss the notifications.

To build this wristband, a small programmable microcontroller is needed. The wristband will need to connect to other sensors around the house wirelessly. This can easily be done via a Wi-Fi network that already exists in most homes. The device ESP32 can be programmed and has built-in Wi-Fi support. It has a lot of possibilities to control various things such as LEDs, motors and can connect to a wide range of sensors. This type of microcontroller is very versatile, so it can be used both for the wristband and for all sensors around the house, which simplifies the design and build process. When used for the sensors, it can be powered using a regular USB cable and power adapter. The wristband will need to be powered using a battery.

In the picture the 3D drawing of the sensor housing for the sound sensor can be seen. This design has enough space for the microcontroller, a hole for the power cord for the microcontroller and a bigger circular hole for the sound sensor. On the side two parts stick out, which can be used to drill holes in and put screws through. The lid for the housing has been made separately. For the vibration sensor the housing looks exactly the same, however it does not have a circular hole because this sensor does not have such part.

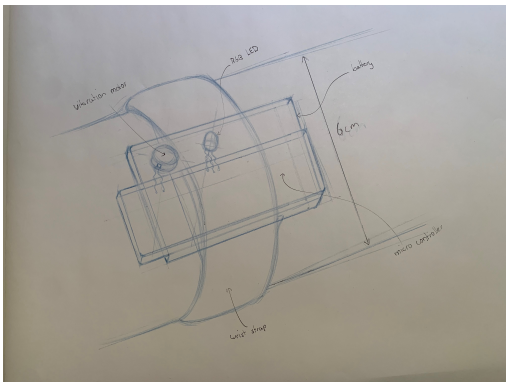

Figure 3: Blueprint

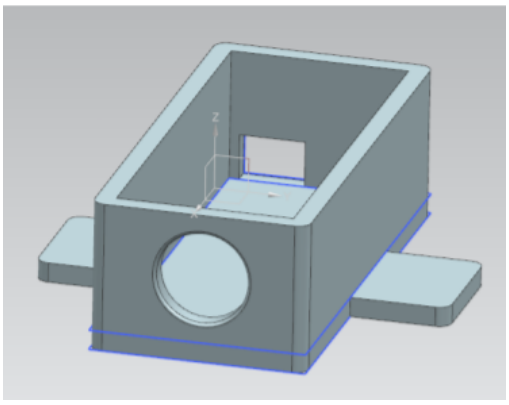

Figure 4: Sensor housing

4.1.1 Pro's and con's

Pro's
- The wristband is easy to use for everyone, also for elderly
- The user will easily notice when for example the washing machine is ready, because of the LED light and the vibration
- Button to turn the notifications off when the user has seen the notification

Con's
- The user must wear it, otherwise there is a big chance the notification will be missed, because the wristband does not make any sound, only flashing light and vibration

4.2 Smart Glasses Concept

This concept is also made for people with hearing impairment. It makes use of the same sensors as the wristband. The sensors are placed at the places close to the preferred devices. When the sensor picks up a sound or vibration it is triggered and sends a signal to the actuator. That is where this design is different than the wristband design.

In Figure 5 a front-view of the smart glasses can be seen. The design makes use of a regular frame of glasses, with a box added to it. The box can be seen on the right. It contains the battery, the microcontroller and an actuator. The actuator will be some sort of motor like a servomotor. When the sensor is triggered, the actuator will start working. It will turn the arm with LED blinker in front of the glasses and thereafter the LED blinker will start blinking, in the desired color. This way the user will be informed that a device is ready.

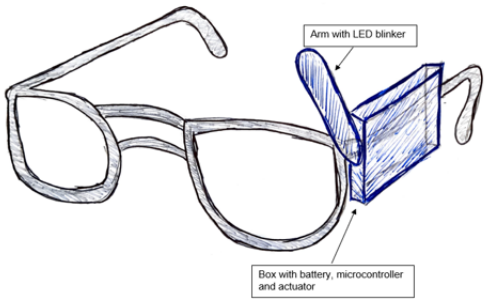

Figure 5: Smart Glasses

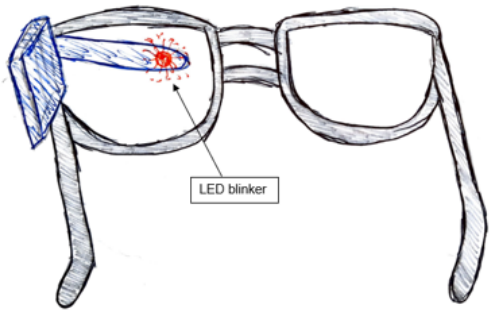

Figure 6: Back view

In Figure 6 the back view of the glasses can be seen when it is activated. It can be seen that the arm is lowered and the blinker is blinking. This is the way the user gets notified.

4.2.1 Pro's and con's

Pro's
- The design is compact and can be placed on regular glasses.
- It will be very noticeable for the user when a device is ready or when a doorbell is rang.
- For people that are already used to wearing glasses, this design will not be too difficult to get used to.

Con's
- The LED light is very close to the eye, which can be quite uncomfortable.
- Even though the arm with LED blinker is not as big as the entire glass, it could still cause dangerous situations when it is lowered, because a bit of vision is lost due to the arm being in front of the eye.
- One side of the glasses will be a bit heavier than the other side, which could maybe feel uncomfortable.

4.3 Hearing box concept

The Hearing Box will be produced to help people with a hearing impairment, who, for example, otherwise may not be able to hear whether the doorbell is ringing or whether their washing machine has finished washing their clothes. In essence it's an USB-wired box with LED lights and text-display for people with a hearing impairment. The product shall be placed on multiple places in the house, such that a person is able to notice the LED lights attached to the box. In addition, multiple sensors need to be placed such that vibrations or sounds of a machine or object could be observed. The design for this product could be changed to any kind of object, however, in order to keep the product as small as possible, a rectangular shape has been chosen.

The microcontroller, powered by an USB-cable, is placed inside this rectangular box. A text-display and two LED lights will be attached to the microcontroller and be positioned on top of the Hearing Box, as depicted in Figure 7.

When a sensor has been triggered, a signal will be send to the microcontroller via Wi-Fi in which case it will activate the LED lights and the display. The LED lights will start to blink and a specific message, corresponding to the sound which has been observed, will be send via the display of the product. Once the user has seen the notification, a button can be pressed to stop the LED's blinking.

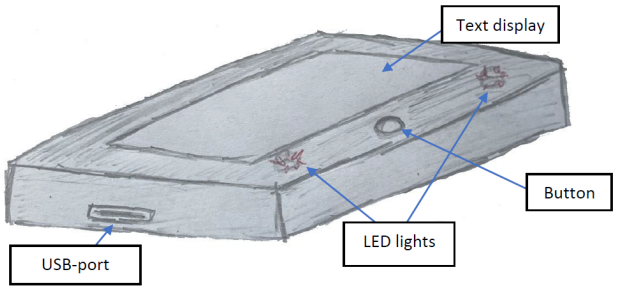

Figure 7: Quick sketch of the hearing box

4.3.1 Pro's and con's

Pro's
- Due to the fact that it is a stationary object, it does not have to be wireless. Hence, a lower budget is needed and safety is improved, since it does not need to use a battery.
- The product does not have to be carried around the house.
- In contrast to the wristband it is possible to use a text-display to display where the sensors have been triggered.

Con's
- It needs to be placed in a lot of places, otherwise the LED lights will not be visible at any time.
- The message on the display may be unreadable for some people with bad eyesight.

5 Final design concept

There are several ways of fulfilling the requirements mentioned in chapter 3. When comparing the design concepts from chapter 4, it is clear that the wristband has the edge over the Smart Glasses and the Hearing Box.

Whereas the wristband is located on the wrist of the user, the Hearing Box would need to be placed in several different places to be visible at all times. This would make the Hearing Box extortionate compared to the wristband since every box needs its own, separate, vital components. The Hearing Box itself will, therefore, cost around 25 euros per box. This excludes any sensors, which will also be around 10 euros per sensor. When making it fully functional in a house, the costs would, therefore, easily exceed the budget of 70 euros. Moreover, when someone covers the Hearing Box with, for example, a coat, someone would not be able to notice the LED or the text display. The Smart Glasses may be a little uncomfortable to wear since the LED is close to the eyes and due to the imbalance in weight. It may also not be as user-friendly as the wristband. Whereas someone can wear the wristband during the day, someone with the Smart Glasses may want to switch glasses when he or she needs to use for example reading glasses. There is also no way of turning off the notification on the glasses, hence someone will need to wait some time for the notification to go away.

The wristband which is called Blink 1.0, does meet all of the functional specifications though. It is very user-friendly, due to the fact that everyone can easily use it. It should be comfortable to wear since the size of the strap can be adjusted in such a way that it will perfectly fit someone's wrist. There is, in contrast to the Smart Glasses, also a possibility to turn off the notification with a button and a user should be able to either see the notification, using the LED or, for example, when the wristband has been covered by their sleeve, to feel it, using the vibration motor. One of the most challenging aspects was the size of Blink 1.0, because the electronic parts need a substantial amount of space. Even though it may look big as could be seen in image 1, it does not feel like it, neither does it feel restraining. In Figure 8, a render of the final design concept can be seen. This figure has the button and light as described before. In Figure 9, a picture of the mockup can be seen. This mockup was made to give a better view on what the design should look like.

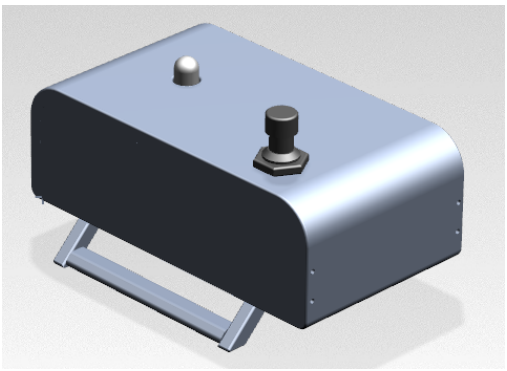

Figure 8: Render Final concept

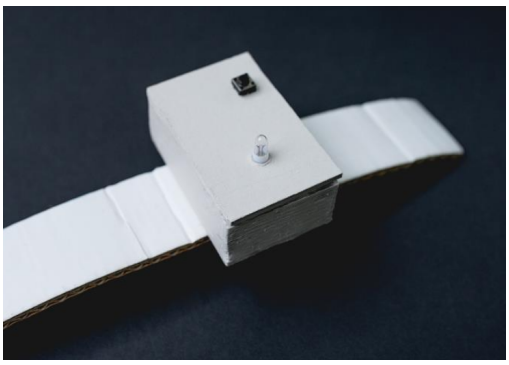

Figure 9: Preliminary design

It is somewhat innovative, since this is the only product that solely focuses on notifying a user when specific sounds or vibrations have been noticed. There are already some smartwatches that can notify users when something happens via sensors connecting with Smart Home. Moreover, there are wristbands for hearing impaired, however, they vibrate whenever they notify a non-specific sound. Also, Blink is a lot cheaper and simpler than for example a smartwatch, which is an advantage for the user.

The smartwatch can be difficult to understand for elderly people, because it has a lot of functions and a display. Blink should be a lot easier to understand, because it does not have a touch screen display and less functions than a smartwatch. When Blink is installed, the user can easily start using Blink.

6 Technical specification

The final design concept also has some technical specifications. Technical specifications are specifications the product needs to do or have to be able to operate. The technical specifications will be mostly quantitative, as it describes the different parts or functions that are needed to fulfill the functional part of the product. Below, the technical specifications can be seen. First the requirements are stated, after which the chosen solution for this requirement can be seen.

- **The aid must function autonomously and interactively**
  In order to work autonomously a battery will be used. The battery used is a Lithium Ion Battery, 3.7V and 1200 mAh. This battery will be connected to a microcontroller, which is a Wemos LOLIN D32. This microcontroller has the ability to send and receive both analog and digital signals. Therefore it is suitable to connect the sensors, button, vibration motor and LED directly to it. These microcontrollers will be connected to each other via WiFi network.
- **The product must have a vibration sensor and a sound sensor and the actuators are a vibration motor and light**
  The wristband uses the following sensors and actuators:
  - Vibration sensor: SW-420, operating voltage range: 3.3V - 5V, digital output, sensitivity adjustable using potentiometer
  - Sound sensor: KY-038, operating voltage range: 3.3V - 5V, sound detection range 50Hz - 20kHz, digital and analog output, sensitivity adjustable using potentiometer for digital output
  - Vibration motor: Adafruit 1201, operating voltage range 2.5V - 4.5V, current draw $\sim$90mA at 4.5V
  - Light: RGB LED, 5mm Diffuse, common cathode
- **The bracelet should have an adjustable strap**
  To make sure that the strap is adjustable, a Velcro strap will be used. This Velcro strap has length 300 mm and width 20 mm. This way the strap can be adjusted to any preferred diameter with a maximum of 85 mm.
- **The bracelet should be able to be used for a few hours without any problems**
  To make sure the product can be used for a while, there is a battery fitted in the product. The battery has a capacity of 1200 mAh. The microcontroller consumes approximately 100-150 mAh. Therefore, the user can use the bracelet for approximately 10 hours and then the user has to recharge the bracelet.
- **The battery must comply with the safety regulations**
  To make sure the battery is safe, some extra parts are needed:
  - The battery has voltage protection to protect it from undervoltage
  - The microcontroller provides current-limiting and short-circuit protection
  - The microcontroller manages the charging of the battery, and protects it from overcharging
- **The user should be able to turn off the actuators**
  To be able to turn off the RGB light and the vibration motor, a button is used. The button used is a 7 mm black button. The button is programmed such that the user can turn off the motor and LED by pressing the button shortly. A long press (5 seconds) will turn the device off completely.

7 Detailing

When a concept is finalized, the next step is to start fabricating it. The detailing phase will start. In this phase the main components will be made and will be detailed. The detailing phase is important because in this step all the parts for the final design are made. The detailing phase of Blink can be divided into three main sections. These are the following: the 3D printed parts, the wristband and the sensors. In this chapter the detailing of these sections will be described.

7.1 Detailing the 3D printed parts

After thinking about the best material for the housing of the wristband, it was decided to 3D print these parts. The advantage of 3D-printing is that a design can be made in which all requirements of the part can be met. The first step of designing a 3D print is therefore to make the design. This was done in Siemens NX12. There were two parts that had to be printed. The top and bottom of the housing. The bottom was made first and there were some key components that had to be designed. The most important components of the bottom are the legs through which the band can be put, the case in which the vibration sensor can be fitted and the walls of the housing itself. The top part is a rather simple part. It was designed as a cap to exactly fit on top of the bottom part. It has three holes: one for the light, for the button and one for the micro-usb port to charge the battery.

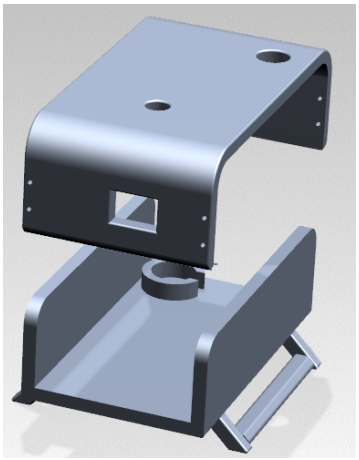

Figure 10: CAD Drawings

After the parts were designed, the files could be sent to the place where they would be printed. Before sending the parts, it had to be checked if the printing time would not exceed the maximum time of 5 hours. This could be done in the program Ultimaker Cura.

When the parts were printed, they were put in the locker. The parts were how they were designed and the electronics fitted, however they were not nicely finished. The last step of detailing the 3D printed parts was therefore to sand down sharp edges, remove printing layers and supports, and to make sure that all electronics fitted. The holes of the button and light had to be made bigger. After these steps the 3D printed parts were ready for use.

7.2 Detailing the wristband

It was decided that for the wristband an Arduino-based microcontroller would be the best choice, since they are relatively cheap and can be easily programmed. Because the components should communicate over Wi-Fi, the microcontroller should have integrated Wi-Fi capabilities, instead of an external Wi-Fi module which would be less compact. Lastly, a battery-charger was needed for the wristband. Again, for compactness, a microcontroller with this functionality integrated would be optimal.

There are two types of Arduino-based microcontrollers with integrated Wi-Fi: the ESP8266 and ESP32. For Blink, the ESP32 was used, since it is the newer and offers more functionality in general. There are various different variants available of the ESP32 microcontroller. Blink uses the WEMOS D32, as this board also has an integrated battery-charger.

From a software perspective, the devices should be programmed to communicate with each other. In addition, the user should be able to configure the wristband to fit their needs. To fit these requirements two main options were considered:

- **Local network communication**:
  The devices would communicate over the local network, without using the internet. When a device starts-up, it scans the network for other Blink devices and connects to them, ready to exchange information.
  An app on a smartphone could then look for these devices on the network and configure them.
- **Communication over the internet** (to a server):
  The devices would all connect to a central server and communicate with it. The server would handle communication with the devices and could relay information from one device to another.

  This server could also provide a website/web-app to allow for the configuration of the devices.

In the end, the second option was chosen as it would be simpler. The server software can handle both the device communication and the configuration options. For the first option, these requirements would need to handled completely separately.

The software package that runs on the server is ThingsBoard. This is an open-source IoT platform that provides a strong base-layer for the functionality needed for Blink. On top of this base-layer, the specific functionality needed for Blink could easily be programmed. An Arduino SDK (Software Development Kit) already exists for ThingsBoard [3], and hence this was used as the primary method for communication.

ThingsBoard communicates using remote procedure calls (RPC's) which triggers some functionality on the device. For example, in Appendix B, on line 315 the function `rpc_notification` can be found, which is triggered when a notification is received.

7.3 Detailing the sensors

For simplicity, the WEMOS D32 microcontroller is used in both the wristband and the sensors, even though the sensors do not use the battery-charger. The types of sensors used are compatible with Arduino-like boards and were hence easy to connect and read data from. For the two sensors, the following logic is used to read their values and send a notification:

- **Sound sensor**: The sound sensor is read using analog signals. This means that the microcontroller can read a value for how loud the sound was. The microcontroller repeatedly reads the sensor's value and stores the highest value it received. Then, every 500 milliseconds, it compares this value with the threshold that was set and sends a notification if the threshold was reached.
- **Vibration sensor**: The vibration sensor only sends a digital signal to the microcontroller. Hence, the microcontroller can only read if there was a vibration, and cannot read the intensity of this vibration. The logic that the microcontroller applies for the vibrations is the following: it will first smooth out the vibration signal by checking that the sensor is vibrating for at least 20 seconds, after which the sensor should not vibrate for at least 20 seconds. If these conditions are met, then a notification is sent.

The function that implements this logic is `loop_sensor` that can be found on line 403 in Appendix B.

8 Realization

Once all parts provided on the bill of materials have arrived, one should start by soldering the pin headers to the micro-controller. This provides a better electrical connection between the pin headers and the micro-controller. In order to check whether all parts work, all components should be tested. Testing happens after soldering the pin headers, otherwise there could, for example,be faulty measurements with sensors or parts such as the LED would not be able to work properly.

8.1 Testing Electronics

8.1.1 LED RGB

It is possible to test the LED by connecting it to the micro-controller as shown in Figure 11. The pins of the common cathode RGB LED will have to be connected to the bread-board and the micro-controller such that the longest pin is connected to the ground (GND). When the GND-pin of the LED is on the second position (counting from left), the pin which controls the red light, the one on the first position, will have to be connected to an 82 Ohm resistor and has to be connected to an analog pin of the micro-controller. Whereas the other two pins, which control the green and blue lighting, located respectively on the third and fourth position, will have to be connected to the 12 Ohm resistors and should also be connected to different analog pins of the micro-controller. The 3 analog pins that are used on the micro-controller in Figure 11 are numbers 32, 33 and 25. By setting the color-value of the LED RGB in the code of the micro-controller to various values the LED could be tested. After all tests have been successful, it is possible to solder the pins of the LED.

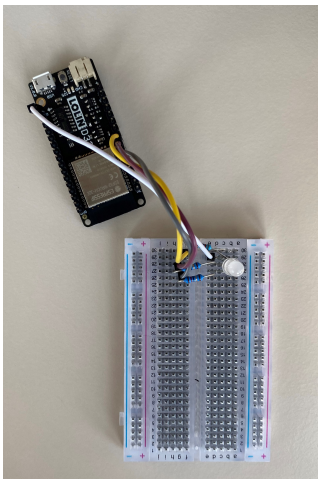

Figure 11: Circuit used for testing LED

8.1.2 Button

One side of the button should be connected to the 3V3, where the other side should be connected to the 10k Ohm resistor and another pin, in this case of Blink 1.0 pin 15 has been chosen, while the other side of the resistor should be connected to the GND.

8.1.3 Vibration motor

The vibration motor could be tested by connecting it in the way shown in Figure 12. The red cable of the vibration motor (+) is connected to the 3V3 pin of the micro-controller, whereas the blue cable (-) is connected to the left pin of the flat side of the NPN transistor as well as to the black side of the diode. The diode should be connected to the 3V3 port with its grey side. The middle pin of the NPN transistor is connected to an 1K Ohm resistor, which is then connected to another analog pin of the micro-controller. In the image to the left it is connected to pin 14. The right pin of the transistor will be connected to the GND. After connecting the vibration motor and micro-controller to the breadboard it is possible to test the vibration motor once the code has been uploaded to the micro-controller. The vibration motor was connected to the breadboard by using jumper wires (instead of plugging it into the breadboard itself), since the wires of the vibration motor were so small, that it was not possible to connect them to the breadboard itself. Soldering could happen only after the vibration motor has been tested and fit- tested.

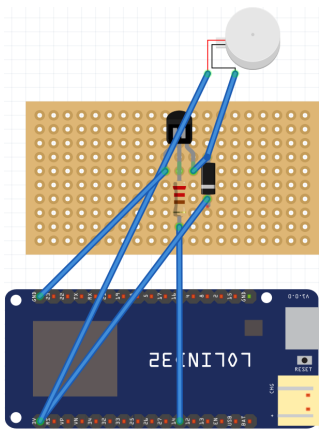

Figure 12: Schematic used for testing the vi-bration motor

After connecting all electronics to a breadboard and to the micro-controller used for the wristband it will look as follows:

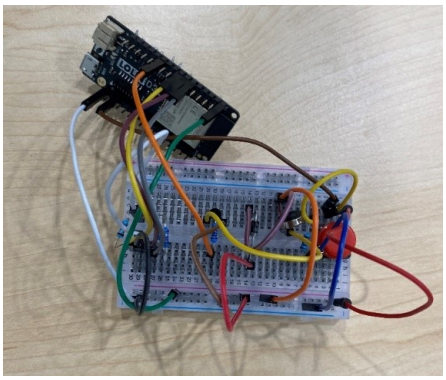

Figure 13: Circuit used for testing of the wristband

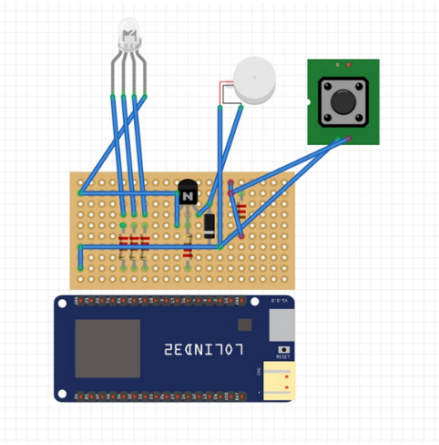

Figure 14: Schematic of the electronics for the wristband

8.1.4 Sensors

The sound sensor and the vibration sensor should also be tested. The sound sensor has 4 pins: The A0-pin is an analog pin to transfer an analog signal, hence, when used, it will need to be connected to any analog pin of the micro-controller. The G-pin is a pin which connects the Ground of the sound sensor to the ground of the micro-controller, hence it will need to be connected to the GND pin. The +-pin is a pin for the operation voltage, hence it will need to be connected to the 3V3 of the micro-controller. The D0-pin is the pin for digital output based on a predefined threshold through the potentiometer and the operation voltage of the micro-controller, hence, when used, it will need to be connected to a digital pin of the micro-controller. The sound sensor has a potentiometer to define a threshold for the digital output pin. In order to set this threshold, it will need to be tested and calibrated extensively. The vibration sensor has 3 pins: The I/O-pin has to be connected to an analog pin on the micro-controller. The GND-pin has to be connected to the ground of the micro-controller. The VCC-pin has to be soldered to the 3V3.The vibration sensor also has an adjuster for the sensitivity. Once again, in order to set this to a certain limit, the product will need to be tested and calibrated extensively.

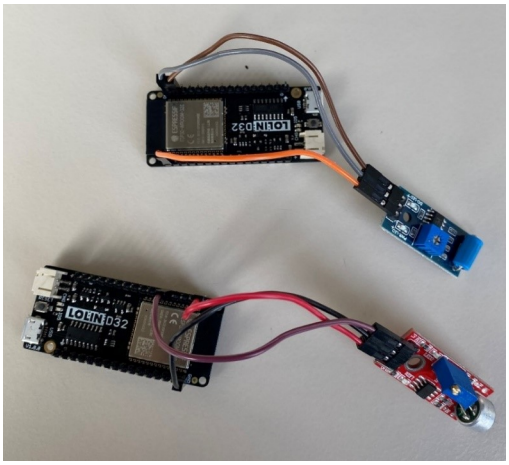

Figure 15: Circuit used for testing sensors

8.2 Fit-testing the electronics

After having soldered everything together the electronics should be fit-tested into the 3D housing. The 3D housing has been printed with 2 holes in it. One for the button and one for the RGB LED, as shown in Figure 16, after fit-testing them, it was clear that they would both easily fit in the housing. There is also a hole for the vibration motor, which at first was not big enough. By using a file (tool) it could be expanded such that the motor would fit, as shown in Figure 17. However once trying to fit all the electronics, it was not possible to close the 3D housing. Hence, white tape has been used to securely close the 3D housing.

Figure 16: Fit-testing the LED and button

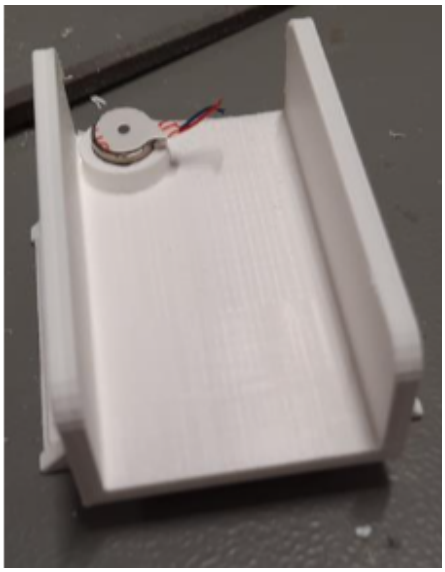

Figure 17: Fit-testing the vibration motor

Two ABS boxes have been used as the sensor-housings. A round hole has been drilled in one of the housings since the sound sensor needs to be in direct contact with its surrounding and the housing may act too much as a damper for the sound. Above that a rectangular hole has been made to fit the USB into the USB port of the micro-controller in order to charge it as could be seen in Figure 18. The other housing only has a rectangular hole for the USB port.

Figure 18: Housing for the sound sensor

8.3 PfP / Soldering

By using the set-up provided in Figure 14, it is possible to test the entire electronics and, again, the parts separately. Once confirmed that all parts work properly, one should start soldering the electronics for the wristband. The pins of the LED should be soldered directly to the resistors and the ground, otherwise, it is not possible to fit the wires in the 3D-housing. In order to solder the vibration motor, one should solder wires to the wires of the vibration motor and solder the electronic parts such as the transistor in the way shown in Figure 14. An experiment PCB has been used such that several inputs can be connected to the ground pin and the 3V3 pin. The ends of the resistors have been cut off and have been used as a line to which the inputs could be connected. Otherwise, it would be very difficult to connect all inputs to a single pin. Such PCB and it's set-up is given in figure Figure 19. The board has been cut to a size such that it will fit all the soldering and electronics and would still fit over the micro-controller and in the housing. Still, the same schematic as Figure 14 has been used. Hence, providing an overview of how everything is placed. After soldering everything together the assembly will look like Figure 20.

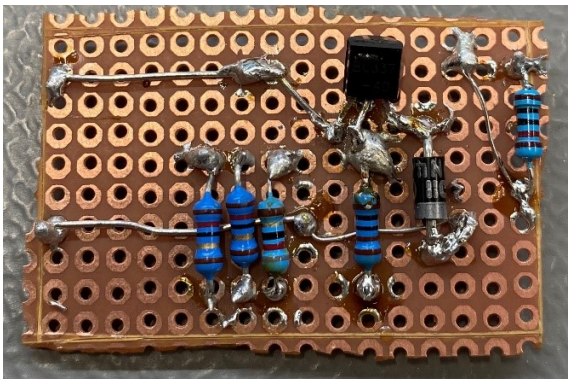

Figure 19: PCB used for this project with several parts of the electronics

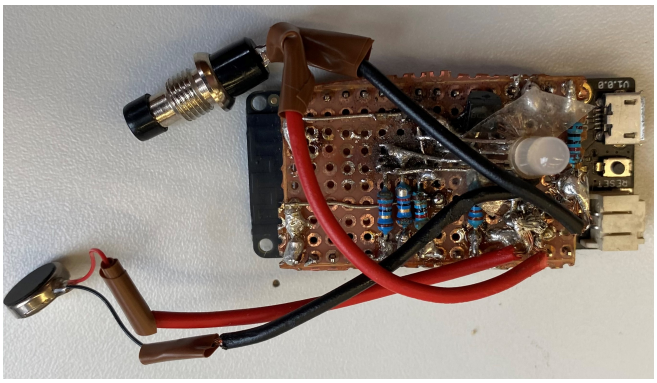

Figure 20: Final result of the soldering

8.4 Bill of materials

Product	**Supplier**	**Price**
3D-printed frame for the wristband	TU/e	€5
3x Wemos Lolin D32, Microcontroller	Tinytronics	€34.50
Lithium Ion Battery 3.7V 1200mAh, Battery	SOS Solutions	€11.99
RGB LED –5mm Common Cathode, Model: RGBLED5MMDIF, LED	Tinytronics	€0.25
82 Ohm Resistor (LED Resistor), Resistor	Tinytronics	€0.05
2x Metal Film Resistor 12.0 Ohm, Resistor, YAGEO	Reichelt	€0.16
Black Button 7mm, Button	Tinytronics	€0.50
10 kOhm Resistor, Resistor	Tinytronics	€0.05
1kOhm Resistor, Resistor	Tinytronics	€0.05
1N4001, Diode, Diodes Incorporated	Kiwi-Electronics	€0.13
Vibration Motor	Ben's Electronics	€0.95
BC33725TA, Transistor, ONSEMI	Farnell	€0.50
KY-037 Sound Detection Sensor Module, Sensor, Electropeak	Otronic	€2.49
SW-420 Vibration Sensor Module, Sensor	HobbyElectronica	€1.95
Velcro Cable Ties 300 x 20 mm, Cable Ties	Onlinekabelshop.nl	€1.20
2x Multi ABS Housing, 85x56x26 mm black, Sensor Housing, Hammond Manufacturing	Reichelt	€8.44
Total costs		**€68.21**

8.5 The final design

In Figure 21 the final design can be seen with all components attached. It can be seen that the device has all parts that were desired. The wristband is fitted with a vibration motor to get the user's attention and a RGB LED to tell the user what kind of signal was received. For example, a blue light can be used for the dishwasher and a red light for the doorbell. After the person has noticed the light, you can simply press the button on the wristband to turn off the LED and the vibration motor. The button and LED light can be seen on top, and the wristband can be strapped around the arm with the strap that can also be seen.

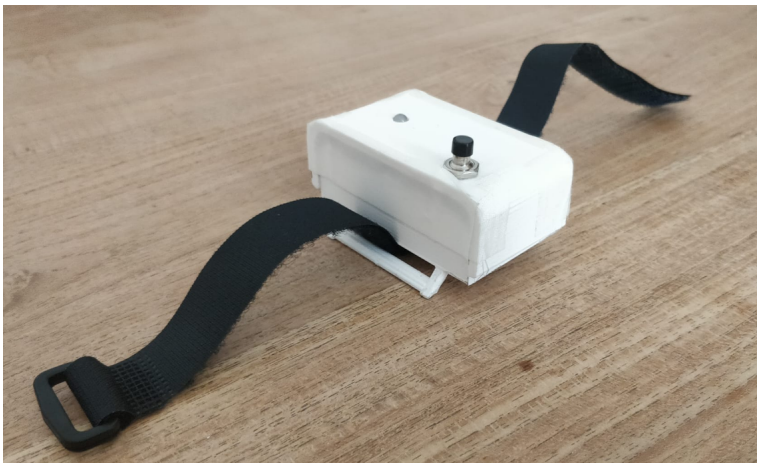

Figure 21: Final Design

9 Test plan and results

Blink 1.0 is an aid in the form of a bracelet for people that suffer from a hearing impairment. To make this product work, Blink 1.0 has a couple of important functions. These functions are:

- The ability to recognize sounds and vibration from domestic appliances
- The ability to send signals from the sound sensor to the bracelet via WiFi
- A vibration motor to notify the user
- A flashing RGB LED to notify the user

A testing plan should be made for all the important functions of Blink 1.0, such that the product can be tested correctly before production.

9.1 Sound detecting

The wristband makes use of sound detecting sensors for loud devices like the doorbell or fire alarm. These sensors should be tested properly to see if they will work in the real world.
The questions before the testing are the following: What kind of experiment is useful to do and also what types of graphs are shown in the results? For the sound sensor it is the most important thing to take a look at the volume of sounds and also at how sensitive it is for the sounds in the surroundings.

The following experiment could therefore be done: The sound sensor is wired and is programmed in the software. Then multiple sounds with different frequencies and tones will be played at different volumes. During those sounds other, lower volume sounds will be played at the background. For example, the volume of the sound of a shutting door. Then, it can be seen how much higher the sound of the doorbell is and a threshold can be chosen to make sure the bracelet is only triggered when the doorbell rings. The graphs will have the time on the x axis and on the y axis the volume of the sounds that the sensor has picked up.

9.1.1 Sound detecting: test results

The sound sensor for detecting the sound of the doorbell is tested and the results of the doorbell sounds are shown in Figure 23.

In the graph, there are a number of peaks. The five highest peaks are the moments where the doorbell was actually pressed. The other peaks are moments when a door was shutting loudly. It stands out that these peaks are much lower than the peaks of the doorbell sound. It was important to measure the volume of another sound, because then a threshold could be determined. By setting the threshold to a value of 2100, the bracelet will be triggered on the right moments and will not be triggered when someone is shutting the door loudly.

9.2 Vibration detecting

The wristband also makes use of vibration detecting sensors for vibrating devices like the washing machine or the dryer and also these sensors should be properly tested.

For the vibration sensor there should also be a testing plan to be able to see if the software works how it is expected to work. However, this device is less difficult to test than the sound sensor, because there are less vibration in the surroundings of the sensors than there are other sounds in the surroundings. Therefore it will occur less that the vibration sensor picks up a sign that it should be triggered. Therefore the plan for this device is to make vibrations on the levels that are comparable to the vibrations that washing machines and dryers make. The goal is to see if the sensors can be detected. When this is the case it can be programmed that the sensor should send a notification to the wristband when the vibration has not been detected for a to be determined period of time.

9.2.1 Vibration detecting: test results

The vibration sensor is tested on a washing machine. A graph is produced by the vibration sensor, while reading the washing machine vibrations.

In Figure 24, the last 20 minutes of a washing cycle is shown. At the start, there are a few short vibrations. These vibrations are filtered out by the microcontroller and will not trigger a notification, this is because there is programmed that Blink will send a notification only when the washing machine is vibrating for 20 seconds and then stops vibrating for 20 seconds. At about 2/3 in the graph, the vibration sensor is vibrating almost constantly.

This indicates the spin-dry part of the washing cycle. The moments where the sensor is not picking up vibrations, will also be filtered out by the microcontroller.

9.3 WiFi Connection

For this feature the main goal is to make sure that the connection between the sound sensor and the bracelet will not be disturbed by any other connections or devices. To test this, the following experiment can be done: Other devices should be put in the room that also use WiFi to connect to each other. A speaker that is connected to a smartphone via WiFi can be put into the same room as the sound sensor or the bracelet. If the sound sensor has difficulties with relaying signals to the bracelet, or cannot connect to the bracelet at all, something should be changed in the programming code. Another test that can be done is checking if all parts with WiFi connection work how they should work at all places in the house.

9.3.1 WiFi Connection: test results

Blink works on WiFi and is tested many times, so the WiFi connection is automatically also tested many times. The WiFi connection is working when there are other devices in the same room as Blink, so there were no problems.

9.4 User friendliness

The product should also be tested on user-friendliness. The product is user friendly if it is comfortable to use and easy to understand. There are a couple of factors which are important for this part. These factors are the following:

- Difficulty of putting on Blink
- Comfort of user when wearing Blink
- Difficulty of learning to understand Blink

For the user it should be easy to put on and put of the device. To test if Blink is difficult in these actions, a testing plan for this is needed. The same holds for the comfort of the user when wearing Blink. Blink will go on your wrist, which is a part of your body that is moved a lot. This means that the device should be comfortable to wear, in order to be successful. The last factor should also be tested, because for users, it is important to be able to understand the device rather quickly.

The best way of testing user-friendliness is to have a group of participants test the device. The results of this group of people can be reflected on a bigger scale to see if Blink will work in real life. In order to have a group of people test the device, first a list of criteria is needed that the product will be tested on. The test participant can indicate on a scale how good Blink operates on these criteria:

- I can easily put on Blink
- I can easily take off Blink
- I can easily adjust the tightness of the band
- Blink is not too heavy to wear on the wrist
- Blink feels smooth on the arm and the wrist
- Blink makes comfortable vibrations
- The LED light is comfortable to look at
- I understand how Blink works after a small period of time

After this the participant is asked if there are any things that could be improved in their opinion. This entire process will be the way of testing how user-friendly Blink is.

9.4.1 User friendliness: test results

The user friendliness of Blink is tested on six users. The users have all filled in a table with the list of criteria which was made before. The users needed to choose between: disagree (1), disagree-average (2), average (3), average-agree (4) or agree (5), for each criterion. All the results can be seen in Table 1. In this table, for each criterion, it is shown how many users have chosen each category (1, 2, 3, 4 and 5).

10 Design evaluation

The design goal for this project was to help people with bad hearing. With the help of Blink, people with a hearing impairment have to be notified when a device in the house makes a sound that needs to be heard. Now that the design has been made, an evaluation can be made to see if the design goal was reached. In this chapter the design will therefore be evaluated and it will be seen if the product that was made is how it was desired to be.

10.1 Critical steps in design procedure

In the procedure of designing the final design and end product, there are some critical steps that were the most important of all the steps. The final design was made, however on some steps a lot of time was needed. These critical steps are the following:

- Precise soldering
- Making sure everything fits in the 3D printed parts

When making such a small product with all these electronics, one of the key steps is the soldering. All electronics should be connected in some way and the soldering should also be done very precisely, as there is not much room for error. Therefore the soldering is a critical step. The soldering was done nicely and all electronics work, however a bit more space was needed than it was foreseen when the 3D printed parts were designed.

The other critical step of the design process was to fit all electronics and soldered parts in the 3D printed parts. The 3D printed parts were designed with a small margin such that there was still a bit room for parts that would stick out a bit more than expected. When all parts were soldered, it became clear that more space was needed. Making sure that everything fits in the 3D printed parts was therefore also a critical step. The 3D printed parts had to be adjusted a bit in some parts to make sure that all electronics could fit in the housing.

10.2 Analysis of end product

The end product can now be analyzed and it can be seen if all requirements were put into the design. As can be seen in Figure 21, all the parts that were desired to be put into the design, also were added to the end product. This means that on that criteria the design was a success. The goal for the design was that it should be creative, innovative but also user-friendly. The final design turned out to be as creative as it was wanted. The end product is quite unique and several components that were added were creative. For instance the Velcro strap. Normally these straps are used for computer wires to have multiple wires together, however in our case it is used as strap for around the wrist. This was done because this strap is easily usable and it can also be easily adjusted to someones wrist. This is one of the examples of why the final design is creative.

The final design also is innovative, as there are not many products with the same functions as Blink. Therefore the design turned out as innovative as the group wanted it to be. The product has most definitely become user-friendly. This was also tested when the product was not finalized yet, such that it could be improved already. The end product has an adjustable strap, such that almost everyone can use the product how it should be used. Also the 3D printed parts are smooth, so the housing of the wristband feels nice on the wrist and does not hurt when putting it on and taking it off. Also the device looks quite big, however it is not too heavy. It is a bit more heavy than the usual watch, however it was not the goal to copy a watch. The device is however pretty big and this could be a pit less pleasant for some people, however all parts that needed to be in the wristband were put in with not much extra room, so it could not have been made a lot smaller. For the LED light it was quite bright at first and during the testing on people, however this was adjusted and is now less bright. Also the vibration motor was not efficiently tied down during testing, and was therefore not making the best vibrations. The final product however has the vibration motor on a place where it is tied down nicely. This way the vibrations are also of good quality.

To conclude, overall the final design was made with a working product as result. This end product was made how it was wanted to be made, and it has become a creative and user-friendly product.

10.3 Possible improvements and Ideal product

When such a design is made there are always possible improvements that would make the product even better and that would have more people interested in the product. After naming these possible improvements an ideal end product could be thought of. Possible improvements are the following:

1. The product could be smaller, such that it looks nicer on your wrist and gives more comfort.
2. The product could have more advanced settings, like multiple lights.
3. The product could have a more advanced display.
4. The product could come with an own doorbell.

For the first possible improvement, a lot of parts should be changed.

This means that the improvement should be implemented very early in the design cycle, at the designing step and detailing step. If the wristband would be made smaller, the product would be more comfortable to wear, and it would look way nicer. This could be solved by using a smaller battery and smaller electronics.

The second improvement also needs some adjustments in parts. Multiple lights should be ordered and more space should be made for these lights in the 3D printed parts. This is an improvement that could be made quite easily with a bit more time.

The third improvement also needs quite some more designing, as a better display not only needs to be designed and made, but it also needs a lot more programming than just a button and LED light. Also this improvement should be implemented quite early in the design cycle, also at the designing and detailing phase.

Figure 22: Ideal product

The fourth improvement would be that a very own doorbell would be included in the package of parts that will be delivered when a client buys Blink. This smart doorbell would already be connected to Blink, such that there is less effort needed to install the sound sensor. This improvement should be implemented in the design cycle at the step where the programming takes place, as the doorbell should be programmed to the wristband.

In Figure 22 a render of a product that would be ideal if the design would be further developed can be seen. This product has a LED strip which looks neat, and also has a smaller button. The overall wristband in this figure is also much smaller, which would be beneficial. However, it was not feasible to make the ideal product for this project.

References

[1] VeiligheidNL & Erasmus MC. (2020, december). Prevalentie van gehoorverlies in nederland (Nr. 879). VeiligheidNL. https://www.veiligheid.nl/organisatie/publicaties/rapport-prevalentie-van-gehoorverlies-in-nederland

[2] Apple. (2021, 6 augustus). Apple deelt inzichten gehoorgezondheidsonderzoek. Apple Newsroom (Nederland). https://www.apple.com/nl/newsroom/2021/03/apple-hearing-study-shares-new-insights-on-hearing-health/

[3] Thingsboard. (2021). GitHub - thingsboard/thingsboard-arduino-sdk: Arduino libarary to connect with ThingsBoard IoT Platform. https://github.com/thingsboard/thingsboard-arduino-sdk

A Appendix: Test results

A.1 Sound detecting

The results of the test with the sound sensor are shown in the graph below:

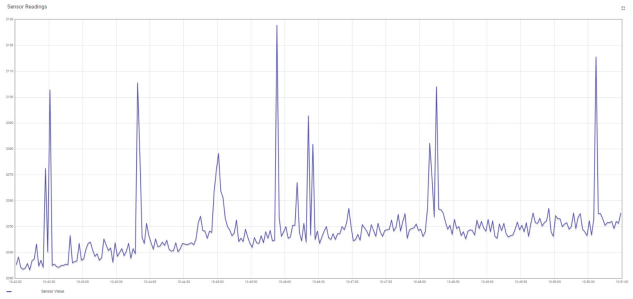

Figure 23: Door bell sounds

A.2 Vibration detecting

The results of the test with the vibration sensor are shown in the graph below:

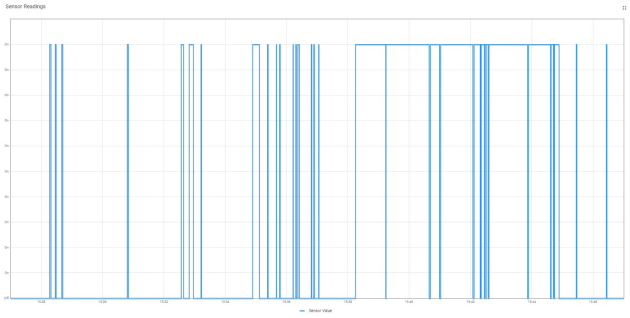

Figure 24: Washing machine vibrations

A.3 User friendliness

Criterion	Disagree (1)	Disagree-average (2)	Average (3)	Average-agree (4)	Agree (5)
I can easily put on Blink	0	0	2	3	1
I can easily take off Blink	0	0	3	2	1
I can easily adjust the tightness of the band	0	0	0	1	5
Blink is not too heavy to wear on the wrist	0	2	1	2	1
Blink feels smooth on the arm and wrist	0	1	3	1	1
Blink makes comfortable vibrations	1	3	0	2	0
The LED light is comfortable to look at	0	1	3	0	2
I understand how Blink works after a small period of time	0	0	1	4	1

Table 1: Test results user friendliness

B Appendix: Programming code

```
/**
 * ----------------------
 * LIBRARY IMPORTS
 * ----------------------
 */
// Choose which device type we are building
#define PROVISION_DEVICE_KEY "m8ngc9rqw9aw60n94ffy"
#define PROVISION_DEVICE_SECRET "b4n95o01ovu5vchs8eqf"

// We need the WiFi library, WiFiManager and ThingsBoard
#include "WiFiManager.h"
#include "ThingsboardConnect.h"

#if defined(ESP32)
#include "analogWrite.h"
#endif

/**
 * ----------------------M
 * GLOBAL VARIABLES
 * ----------------------
 */
// Set current version
#define CURRENT_FIRMWARE_TITLE    "Dev"
#define CURRENT_FIRMWARE_VERSION  "1.1.8"


// Pins for Sensors
#define SOUND_SENSOR GPIO_NUM_34
#define VIBRATION_SENSOR GPIO_NUM_23

// Pins for Wristband
#define VIBRATION_MOTOR GPIO_NUM_14
#define BUTTON GPIO_NUM_2

#define RGB_RED GPIO_NUM_25
#define RGB_GREEN GPIO_NUM_33
#define RGB_BLUE GPIO_NUM_32

#define BATTERY_LEVEL GPIO_NUM_35

// We create the WifiManager
WiFiManager wifiManager;

// Compute a unique device name based on the chip id that has been set by the factory
String getUniqueDeviceName() {
#if defined(ESP8266)
  String deviceId = String(ESP.getChipId(),HEX);
#elif defined(ESP32)
  uint32_t id = 0;
  for(int i=0; i<17; i=i+8) {
    id |= ((ESP.getEfuseMac() >> (40 - i)) & 0xff) << i;
  }
  String deviceId = String(id,HEX);
#endif
  deviceId.toUpperCase();
  return "Blink_" + deviceId;
}

String DEVICE_NAME = getUniqueDeviceName();


ThingsboardConnect tb(CURRENT_FIRMWARE_TITLE, CURRENT_FIRMWARE_VERSION, DEVICE_NAME.c_str(),
     PROVISION_DEVICE_KEY, PROVISION_DEVICE_SECRET);

const size_t rpcHandlersSize = 3;
RPCHandler rpcHandlers[rpcHandlersSize] = {
  { "notification", rpc_notification },
  { "test_notification", rpc_test_notification },
  { "reboot", rpc_reboot }
};

```

```
/**
 * ----------------------
 * PROGRAM SETUP
 * ----------------------
 */

/**
 * Setup function which will run once at start
 */
void setup() {
  // Set-up serial communication which can be used for debugging when connected to a
    computer
  Serial.begin(115200);

  setup_pins();

  // Setup wifi connection using WifiManager
  setup_wifi();

  // When we receive an update from the server that the attributes have changed, the
    process_attributes function will be called
  tb.useAttributes(process_attributes);
  tb.useRPC(rpcHandlers, rpcHandlersSize);
  tb.setAttributes("bl_device_type,bl_sensor_threshold");
}

/**
 * Setup the Wi-Fi connection using Wi-Fi manager
 * When this function finishes, we guaranteed have a Wi-Fi connection,
 * because if Wi-Fi fails, we automatically reboot
 */
void setup_wifi() {

  // We customize the wifi manager menu with only the pages we need
  const char* menu[] = {"wifi","info","sep","restart"};
  wifiManager.setMenu(menu,9);
  wifiManager.setShowInfoUpdate(false);

  wifiManager.setHostname(DEVICE_NAME.c_str());
  wifiManager.setTitle("Blink 1.0");

  // Allow other code to execute while we are awaiting wifi connection.
  wifiManager.setConfigPortalBlocking(false);

  // Set blue color to indicate wifi connecting
  rgb_color(0,0,255);

  // Trigger the logic of WiFiManager to connect to the Wi-Fi network,
  // or if it fails, let the user configure a new Wi-Fi network
  wifiManager.autoConnect(DEVICE_NAME.c_str());

  if(!wifiManager.autoConnect(DEVICE_NAME.c_str())) {
    unsigned long config_start = millis();
    unsigned long button_pressed_time = millis();

    int color = 0;
    int animation = 0;

    while(1){
      // If 3 minutes have passed, try to restart
      if(millis() - config_start > 180000){
        Serial.println("No WiFi connection was established. Rebooting...");
        delay(1000);
        ESP.restart();
      }

      if( animation == 0 )
        color++;
      else
        color--;

      if( color >= 4096 && animation == 0 )
        animation = 1;
      else if( color <= 0 && animation == 1 )
```

```
        animation = 0;

      rgb_color(0,0,color / 16);

      int buttonPressed = digitalRead(BUTTON);
      if(buttonPressed == HIGH && button_pressed_time == 0)
        button_pressed_time = millis();
      else if(buttonPressed == HIGH && button_pressed_time != 0 && millis() -
    button_pressed_time > 5000)
        poweroff();
      else if(buttonPressed == LOW)
        button_pressed_time = 0;

      // Check if wifi was connected succesfully
      if( wifiManager.process() )
        break;

      yield(); // watchdog
    }
  }

  rgb_color(0,0,0);

  // Turn of the wifi acccespoint (which WifiManager should do, but does not always do for
    some reason)
  WiFi.softAPdisconnect(false);
  WiFi.enableAP(false);

  //Start web portal for config access after the device has started
  wifiManager.startWebPortal();
}

void rgb_color(int red, int green, int blue) {
  analogWrite(RGB_RED,map(red,0,255, 0, 64));
  analogWrite(RGB_GREEN,map(green,0,255, 0, 64));
  analogWrite(RGB_BLUE,map(blue,0,255, 0, 64));
}

/**
 * Setup logic for the wristband
 */
void setup_pins() {
  // Setup pins for wristband
  pinMode(RGB_RED, OUTPUT);
  pinMode(RGB_GREEN, OUTPUT);
  pinMode(RGB_BLUE, OUTPUT);

  for(int i = 0; i < 255; i++) {
    rgb_color(0,i,0);
    delay(4);
  }

  rgb_color(0,0,0);

  pinMode(VIBRATION_MOTOR, OUTPUT);
  digitalWrite(VIBRATION_MOTOR, false);

  pinMode(BATTERY_LEVEL, INPUT);
  pinMode(BUTTON, INPUT);

  pinMode(GPIO_NUM_0, INPUT);
  pinMode(GPIO_NUM_2, INPUT);
  pinMode(GPIO_NUM_4, INPUT);
  pinMode(GPIO_NUM_15, INPUT);

  // Setup pins for sound sensor
  pinMode(SOUND_SENSOR, INPUT);

  // Setup pins for vibration sensor
  pinMode(VIBRATION_SENSOR,INPUT);
}

String deviceType;
int sensor_threshold = 0;
```

```
216
217 // We process the attributes (user configuration options) from ThingsBoard
218 void process_attributes(const Shared_Attribute_Data &data) {
219   if( data.containsKey("bl_sensor_threshold") )
220     sensor_threshold = data["bl_sensor_threshold"].as<int>();
221   if( data.containsKey("bl_device_type") )
222     deviceType = data["bl_device_type"].as<String>();
223 }
224
225
226 /**
227  * ----------------------
228  * PROGRAM LOOP
229  * ----------------------
230  */
231
232 bool pending_reboot = false;
233
234 bool thingsboard_disconnected = false;
235
236 unsigned long previous_processing_time = millis();
237
238 /**
239  * Loop function which will be called repeatedly
240  */
241 void loop() {
242   if(pending_reboot) {
243     Serial.println("Rebooting...");
244     delay(1000);
245     ESP.restart();
246   }
247
248   // Call wifimanager loop handle logic
249   wifiManager.process();
250   // Call thingsboard (and WiFi) loop handle logic
251   if(!tb.handle()) {
252     rgb_color(0,0,20);
253     thingsboard_disconnected = true;
254   } else if(thingsboard_disconnected) {
255     rgb_color(0,0,0);
256     thingsboard_disconnected = false;
257   }
258
259   // Call the correct loop function depending on the configured DEVICE_TYPE
260   if( deviceType == "Wristband" )
261     loop_wristband();
262   else if( deviceType == "Vibration Sensor" || deviceType == "Sound Sensor" )
263     loop_sensor();
264 }
265
266
267 unsigned long button_pressed_time = 0;
268 unsigned long button_released_time = 0;
269
270 bool notification_active = false;
271 int battery_percentage = 0;
272
273 /**
274  * Loop logic for the wristband
275  */
276 void loop_wristband() {
277   if (millis() - previous_processing_time >= 500) {
278     previous_processing_time = millis();
279     int battery_level = analogRead(BATTERY_LEVEL);
280     float battery_voltage = battery_level / 4096.0 * 6.9;
281     battery_percentage = int( ( constrain(battery_voltage, 3.7, 4.15) - 3.7 ) / 0.45 * 100 )
      ;
282     tb.sendTelemetry("battery_level",battery_level);
283     tb.sendTelemetry("battery_percentage",battery_percentage);
284     tb.sendTelemetry("battery_voltage",battery_voltage);
285   }
286
287   int buttonPressed = digitalRead(BUTTON);
288   if(buttonPressed == HIGH && button_pressed_time == 0) {
```

```
    button_pressed_time = millis();
  } else if(buttonPressed == LOW && button_pressed_time != 0) {
    button_pressed_time = 0;

    if( millis() - button_released_time > 200 ) {
      if( notification_active )
        rgb_color(0,0,0);
      else {
        if( battery_percentage > 60 )
          rgb_color(0,255,0);
        else if( battery_percentage > 20 )
          rgb_color(255,255,0);
        else
          rgb_color(255,0,0);
        delay(700);
        rgb_color(0,0,0);
      }
      notification_active = false;
    }

    button_released_time = millis();
  } else if(button_pressed_time != 0 && millis() - button_pressed_time > 5000) {
    poweroff();
  }
}

RPC_Response rpc_notification(const RPC_Data &data) {
  Serial.println("Received Notification");

  // Extract rgb colors + vibration pattern from RPC data
  int rgbRed = data["bl_rgb_red"].as<int>();
  int rgbGreen = data["bl_rgb_green"].as<int>();
  int rgbBlue = data["bl_rgb_blue"].as<int>();
  int pattern = data["bl_vibration_pattern"].as<int>();

  // Set the LED to the correct colors
  rgb_color(rgbRed,rgbGreen,rgbBlue);

  // Trigger the vibration motor to vibrate the preprogrammed pattern
  vibrate_pattern(pattern);

  notification_active = true;

  return RPC_Response();
}

void poweroff() {
  Serial.println("Button long-pressed. Shutting down...");

  // Set the LED to RED for 1 second to indicate to the user that the device is turning off
  for(int i = 255; i > 0; i--) {
    rgb_color(i,0,0);
    delay(4);
  }
  rgb_color(0,0,0);

  // Wait for the user to release the button such that the device does not immediately wake
    back up
  delay(3000);

  // Set ESP to wakeup when button is pressed
  esp_sleep_enable_ext0_wakeup(BUTTON, 1);

  // Put device into deep sleep
  esp_deep_sleep_start();
}

void vibrate_pattern(int pattern) {
  if(pattern == 0) {
    digitalWrite(VIBRATION_MOTOR, true);
    delay(1000);
    digitalWrite(VIBRATION_MOTOR, false);
  } else if(pattern == 1) {
    digitalWrite(VIBRATION_MOTOR, true);
```

```
    delay(500);
    digitalWrite(VIBRATION_MOTOR, false);
    delay(100);
    digitalWrite(VIBRATION_MOTOR, true);
    delay(500);
    digitalWrite(VIBRATION_MOTOR, false);
  } else if(pattern == 2) {
    digitalWrite(VIBRATION_MOTOR, true);
    delay(800);
    digitalWrite(VIBRATION_MOTOR, false);
    delay(100);
    digitalWrite(VIBRATION_MOTOR, true);
    delay(100);
    digitalWrite(VIBRATION_MOTOR, false);
  }
}

RPC_Response rpc_test_notification(const RPC_Data &data) {
  tb.sendRPC("notification");
  return RPC_Response();
}

RPC_Response rpc_reboot(const RPC_Data &data) {
  pending_reboot = true;
  return RPC_Response("rebooting", true);
}

bool didVibrate = false;
int maxSoundLevel = 0;

bool didSendNotification = false;

unsigned long startedVibrating = 0;
unsigned long stoppedVibrating = 0;

unsigned long startedVibratingSmooth = 0;
unsigned long stoppedVibratingSmooth = 0;

/**
 * Loop logic for a sensor
 */
void loop_sensor() {
  if( deviceType == "Sound Sensor" ) {
    // Read the sound level from the sensor
    int soundLevel = analogRead(SOUND_SENSOR);

    // We keep track of the maximum sound level that was received
    if( soundLevel > maxSoundLevel )
      maxSoundLevel = soundLevel;

    // We call this processing every 500ms
    if (millis() - previous_processing_time >= 500) {
      previous_processing_time = millis(); // Reset timer

      // Send data back to Thingsboard to be able to show this in a graph to the user
      tb.sendTelemetry("sensor_value",maxSoundLevel);
      tb.sendTelemetry("sensor_threshold",sensor_threshold);

      // Send a notification if the sound level is above the threshold
      if( maxSoundLevel > sensor_threshold && !didSendNotification ) {
        tb.sendRPC("notification");

        // Set this flag to true to make sure we are not sending multiple notifications
    right after eachother
        didSendNotification = true;
      } else
        didSendNotification = false;

      // Reset the max sound level
      maxSoundLevel = 0;
    }
  } else if( deviceType == "Vibration Sensor") {
    // Read the vibration sensor
    bool vibrating = digitalRead(VIBRATION_SENSOR);
```

```
435
436     // Keep track if the sensor vibrated
437     // We will check if it vibrated every 500ms
438     didVibrate = didVibrate || vibrating;
439
440     // We call this processing every 500ms
441     if (millis() - previous_processing_time >= 500) {
442       previous_processing_time = millis(); // Reset timer
443
444       // Send data back to Thingsboard to be able to show this in a graph to the user
445       tb.sendTelemetry("sensor_value",didVibrate);
446
447       /**
448        * The following logic smooths out the vibrations,
449        * making sure a notification is only sent if the vibration sensor vibrated
450        * for at least 20s and then stopped for at least 20s
451        */
452       if( startedVibrating == 0 && didVibrate ) {
453         startedVibrating = millis();
454         stoppedVibrating = 0;
455       } else if( stoppedVibrating == 0 && startedVibrating != 0 && !didVibrate ) {
456         stoppedVibrating = millis();
457         startedVibrating = 0;
458       }
459
460       if( startedVibrating != 0 && millis() - startedVibrating > 20000 &&
    startedVibratingSmooth == 0 ) {
461         startedVibratingSmooth = startedVibrating;
462         stoppedVibratingSmooth = 0;
463       }
464
465       if( stoppedVibrating != 0 && millis() - stoppedVibrating > 20000 &&
    stoppedVibratingSmooth == 0 && startedVibratingSmooth != 0 ) {
466         stoppedVibratingSmooth = stoppedVibrating;
467         startedVibratingSmooth = 0;
468
469         // Send notification if conditions are met
470         tb.sendRPC("notification");
471       }
472
473       didVibrate = false;
474     }
475   }
476
477 }
```