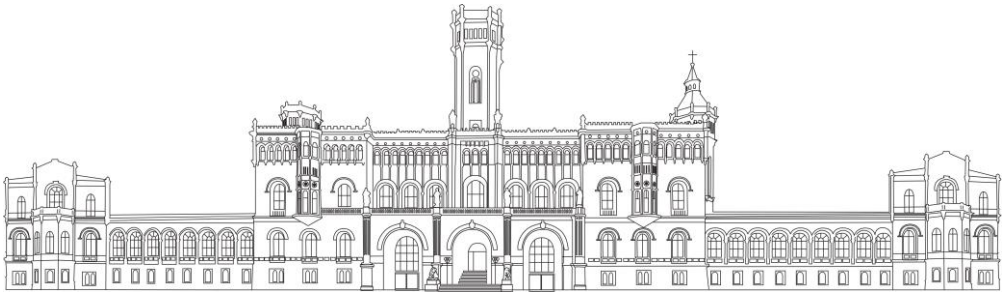


LEIBNIZ UNIVERSITÄT HANNOVER

FACULTY OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

HUMAN-COMPUTER INTERACTION GROUP

AMBIENT NOTIFICATIONS FOR A SHAPE-CHANGING DISPLAY

A Thesis presented for the degree of Bachelor of Science

by

Christian Althaus

Februar 2018

First Examiner:	Prof. Dr. Michael Rohs
Second Examiner:	Prof. Dr. Kurt Schneider
Supervisor:	Tim Dünte, M.Sc.

ABSTRACT

This thesis is about a shape-changing display, which can display information through height adjustment and led lights. An existing system was used to control the shape-changing display. This system will be examined and then redesigned with the objective to make it simpler and more compact, so that a user can take it with home. The developed system should allow visualization of real-time information from web sources. Furthermore, it should display information in an unobtrusive manner.

The displayed scenarios, for example the current or predicted temperature, can be configured in a graphical user interface. It can be specified, how the information will be displayed. This, among others, includes specification of the led colors or different visualization modes. Therefore, a concept for displaying quantitative values is developed. Different animations were implemented to provide better feedback of the developed system and scenarios.

ZUSAMMENFASSUNG

Diese Arbeit behandelt ein Shape-Changing Display, welches benutzt wird um Informationen mit Veränderung der Höhe und mit Led Beleuchtung anzuzeigen. Ein existierendes System wurde eingesetzt um das Shape-Changing Display anzusteuern. Dieses System wird untersucht und anschließend neugestaltet, mit dem Ziel es einfacher und kompakter zu machen, sodass ein Benutzer es mit nach Hause nehmen kann. Das neugestaltete System soll eine Anzeige von Echtzeitinformationen aus dem Internet ermöglichen. Des Weiteren soll es Daten, beziehungsweise Informationen, in einer unaufdringlichen Weise darstellen.

Die angezeigten Szenarios, zum Beispiel die aktuelle oder vorhergesagte Temperatur, kann mit einer graphischen Benutzeroberfläche konfiguriert werden. Es kann angegeben werden, wie die Daten dargestellt werden. Unter anderem, umfasst dies die Led Farbeinstellungen und unterschiedliche Anzeigeverfahren. Daher wird ein Visualisierungskonzept entwickelt. Unterschiedliche implementierte Animationen sollen eine bessere Rückmeldung des entwickelten Systems und der Szenarios ermöglichen.

INHALTSVERZEICHNIS

1 INTRODUCTION

This thesis aims to optimize and simplify an existing shape-changing display prototype, which is used to visualize information.

This display is referred to by "MovingBar", since it is a vertical bar, which can adjust its height. It was built by Andre Lehnert (2016) as part of a master thesis. It has led stripes attached to its sides. These properties are used for ambient notifications, which should inform people about informational states.

1.1 MOTIVATION

Present computer systems are mainly designed for a direct interaction with the user. They usually require full attention, when focusing on the current task. When a person wants to be informed about changing information or waits on an event, this often results in a frequent checking of new updates. An expected e-mail, the current trend of a share on the stock market or the bid price of an online auction are some examples for this. This implicates additional expenditure and also a interruption of other more important tasks.

Ambient interfaces try to solve this problem. This display types should provide unobtrusive feedback of informational states. This not only includes graphical displays, but also further concept like change in movement, sound, light or shape. The peripheral awareness of the user will register the subtle changes of the ambient interface. Therefore, recent events and notifications will be presented in a less distractive way in the environment of the user. New concepts and designs have to be developed to reach these objectives.

1.2 CHALLENGES AND SCOPE

The existing system consists of the shape-changing display, a system controller and the user interface. It allows to control the height and the leds of the MovingBar directly via a graphical user interface or indirect with an outsourced database. The focus here was to enable the control of the system.

The system controller is mainly build to run under the conditions of a laboratory setup. This is, because of the space requirements of the components. The power supply for example is a self-made wooden box and for the setup multiple connection wires are necessary. Also, there are no encasings for the microcontroller and the Raspberry Pi, which are the logic of the system. Besides,

the attachment of the shape-changing display is developed for demonstration purposes, rather than providing usability.

This thesis aims to improve these issues. The new system should provide a better usability and simplify the rather complex architecture of the existing system. In order to display information and the current system state of the MovingBar, visualization concepts will be developed and implemented. Different modes will allow to adapt the information to a corresponding display state. Blink animations should provide feedback of the current system state to indicate warning and error messages. A graphical user interface is developed and allows to specify different scenarios on the MovingBar. A scenario, for example, is the current or predicted temperature of a location. On each side of the MovingBar a scenario can be displayed independently. Therefore, four different scenarios can run simultaneously. The information values are frequently fetched from web sources, which are specified by the user. The user can also specify the color and other important parameters for the illustration.

1.3 DOCUMENT STRUCTURE

Next, a brief overview of the following chapters will be given.

Chapter 2 discusses the related work of this thesis. First, findings of the effect of colors on emotions are described. Then a paper is presented, whose authors researched the characteristics of different light patterns. These findings help to convey information more effectively. The next subsections address the definition and classification of shape-changing interfaces. Finally an overview of various researched and build shape-changing interfaces is given.

Chapter 3 describes the existing system and outlines concepts of the new system. Challenges and objectives are explained in more detail. Then a description of the later used visualization concept is given.

Chapter 4 explains the functionalities of the used hardware and tools. This focuses on the relevant topics for the development and implementation.

Chapter 5 provides a description of the development and implementation of the new system. First the software setup of the ATmega328p microcontroller is explained. Then the design and function of a printed circuit board, which holds multiple components of the system, is described. Next, the software of the Raspberry Pi is explained. Then the functionalities of the graphical user interface are illustrated. Finally, the 3d printed components are described.

Chapter 6 illustrates the results of the development process and given feedback on the final system wil be discussed. Finally an outlook on possible further applications and developments is given.

2

RELATED WORK

In this chapter the effects of colors on emotions are described, since the MovingBar uses color led lights to convey information to a human user. Thus, ambient notifications can purposeful be provided to support a specific, intended atmosphere respectively emotion.

Next, a definition and a classification of shape-changing interfaces is given.

Then different kinds of shape-changing interfaces respectively projects are presented to illustrate the potential and usage of this technology.

2.1 EFFECTS OF COLOR ON EMOTIONS

Valdez and Mehrabian (1994) investigated the effects of color stimuli on human emotions. While previous investigations of this topic mostly used vague description of color and emotions, they tried a more systematic and standardized approach. First they used the standardized Munsell color system, which has the property that it retains the perceived color distance of human color vision within the system.

A color can be described "in terms of hue (i.e. wavelength), brightness or value (i.e. black-to-white quality) and saturation or chroma(i.e. purity [...])". Valdez and Mehrabian (1994) describe the impact of each of this color property on emotions independently.

To describe emotions most correctly, the authors used the PAD model, which stands for pleasure, arousal and dominance. The PAD factors are "analogues of the Evaluation, Activity and Potency factors which [...] may be characterized as the lowest common denominator of cognitive response" (Valdez & Mehrabian, 1994). With these factors emotions can be described, for example bold (.44, .61, .66) or distressed (-.61, .28, -.36), which has low pleasure and dominance values, while arousal is higher. These factors contain values between -1 and 1. The authors derived an approximate mapping of emotions to PAD values from ratings of 240 emotional states (see Table 1). The tendency of the PAD values is shown with plus or minus for positive or negative trends.

P	A	D	Emotions
+	+	+	admired, bold, creative, powerful, vigorous
+	+	–	amazed, awed, fascinated, impressed, infatuated
+	–	+	comfortable, leisurely, relaxed, satisfied, unperturbed
+	–	–	consoled, docile, protected, sleepy, tranquilized
–	+	+	antagonistic, belligerent, cruel, hateful, hostile
–	+	–	bewildered, distressed, humiliated, in pain, upset
–	–	+	disdainful, indifferent, selfish-uninterested, uncaring, unconcerned
–	–	–	bored, depressed, dull, lonely, sad

Table 1: PAD scores for different emotions (Valdez & Mehrabian, 1994)

The studies, among others, include participants, which should look at a colored paper and then complete a questionarie to record their emotions. Ten colors respectively hues were illustrated, with differences in brightness and saturation. Hue, brightness and saturation are the independent and pleasure, arousal and dominace are the dependent variables. With linear and nonlinear regression analyses the authors computed graphs, which should explain the correlation between the mentioned variables.

While the independent variable saturation has no significant square amount of the nonlinear function, three linear equations yield an impression of the correlations.

(1) Pleasure = .69 Brightness + .22 Saturation
(2) Arousal = -.31 Brightness + .60 Saturation
(3) Dominance = -.76 Brightness + .32 Saturation

Equation 1 indicates, that brighter and more saturates colors are more pleasant. Equation 2 and 3 shows, that less bright and more saturated colors provoke higher arousal and dominance.

The authors illustrated the effect of brightness on dominance and arousal with nonlinear functions (see Figure 1, Figure 2). For pleasure, the linear correlation in Equation 1 is sufficient.

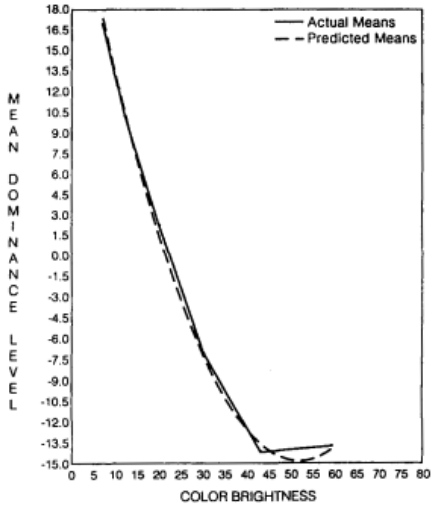


Figure 1: Actual and predicted dominance levels for brightness (Valdez & Mehrabian, 1994)

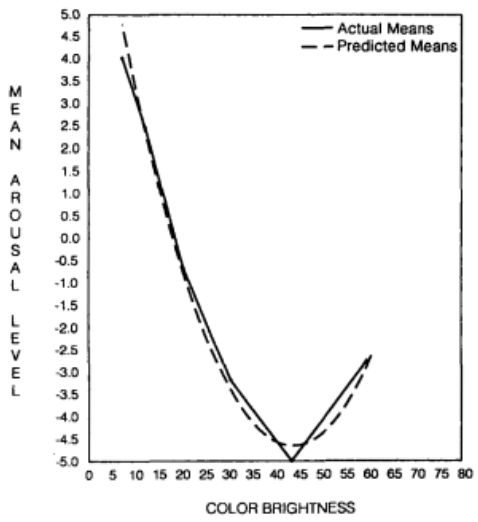


Figure 2: Actual and predicted arousal levels for brightness (Valdez & Mehrabian, 1994)

These findings indicate, that the provoked emotion is not only dependent on hue, but also strong correlates with brightness and saturation. The authors clarify the common error, that for example in many experiments the color red is often associated with arousing emotions, while in fact the high saturation is the reason.

The effects of hue respectively wavelength on pleasure are shown in Figure 3 (see appendix Figure 51, Figure 52 for arousal and dominance).

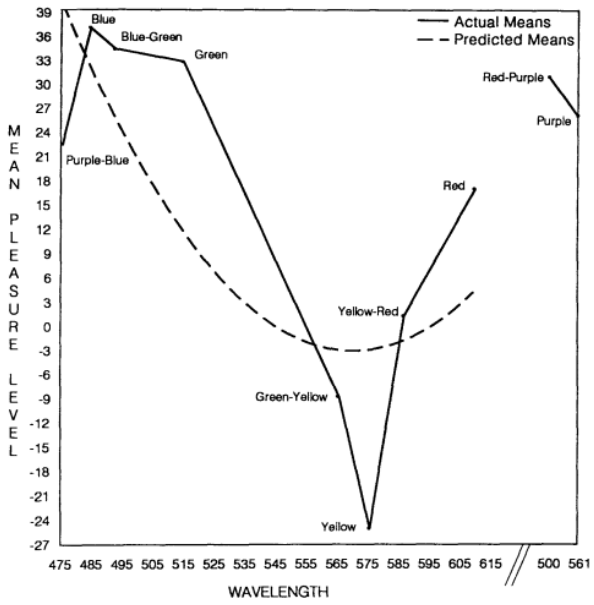


Figure 3. Mean pleasure level plotted against wavelength (Valdez & Mehrabian, 1994)

2.2 LIGHT PATTERNS

In this section light animations are presented, in order to illustrate informational states of a device.

Harrison et al. (2012) investigated light patterns of a single point light in order to enhance the design space of devices, which use leds to convey information. Light patterns are generated by varying the light intensity over time. The kinds of information, that a device intends to convey to a user, the authors named informational states. Informational states, for example, can be the boot process or data transfer of a device. The authors collected various of informational states by a survey and distributed them in five categories, because some states were highly correlated with others. The categories are Notification, Active, Unable, Low-Energy State and Turning On (see Table 2).

Category	Informational States
Notification	The device has an incoming call. …received a message. …event or scheduled item.
Active	…transmitting or receiving data. …thinking, computing, or processing. …active, monitoring, … or progressing.
Unable	…unable to connect. …unable to accept user input or commands
Low-Energy State	…low battery. …sleeping, suspended, or hibernating.
Turning On	…turning on, booting, or warming up.

Table 2: High-level categories (Harrison et al., 2012)

Within the survey they also collected 24 feasible light behaviours respectively light patterns, which, among others, were known by the participants (see Figure 4).

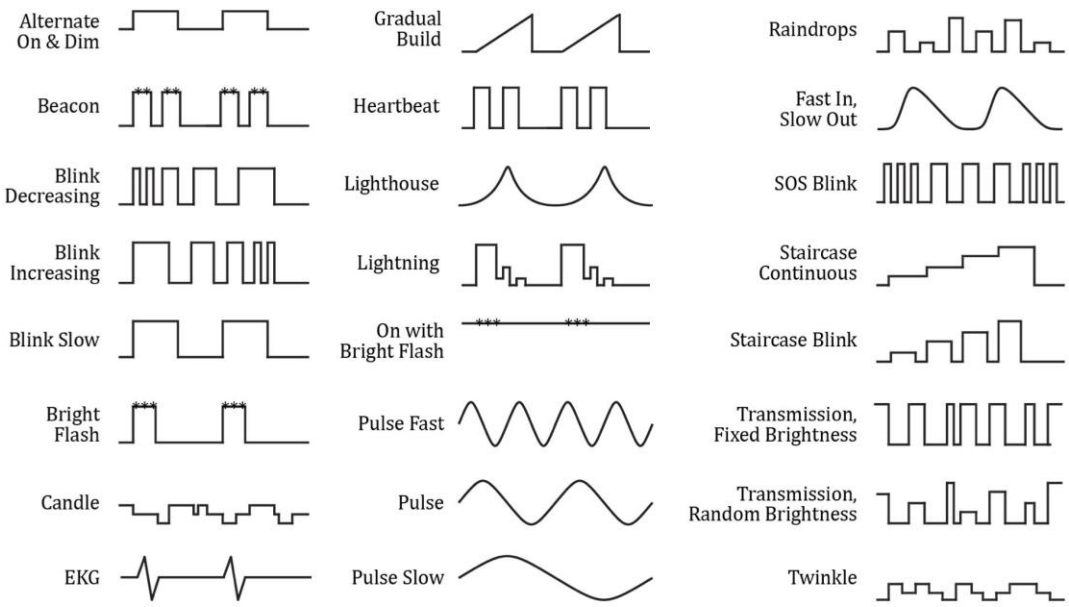


Figure 4: Light patterns (Harrison et al., 2012)

In an additional survey, persons should rate how strongly every light pattern represents each of the five informational states. A five-point likert scale was used for the ratings (see Figure 5, Figure 6 and appendix: Figure 53, Figure 54).

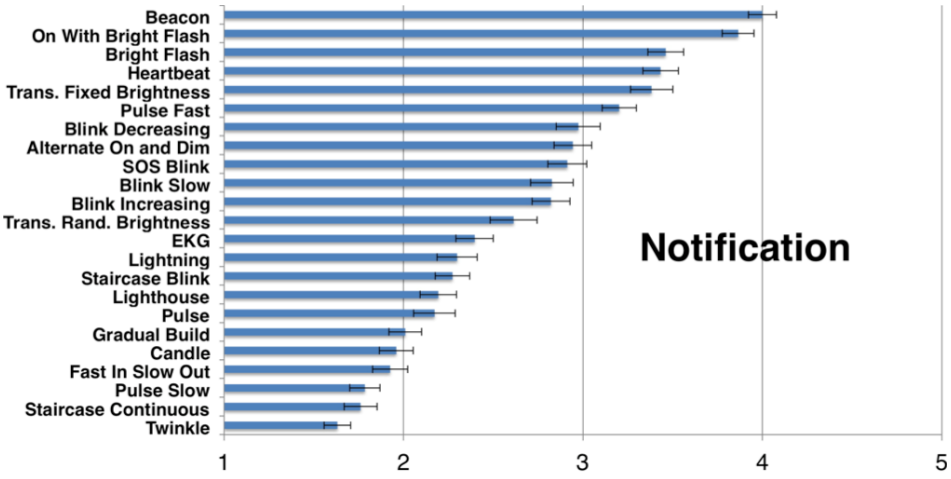


Figure 5: Ratings of light patterns for notifications (Harrison et al., 2012)

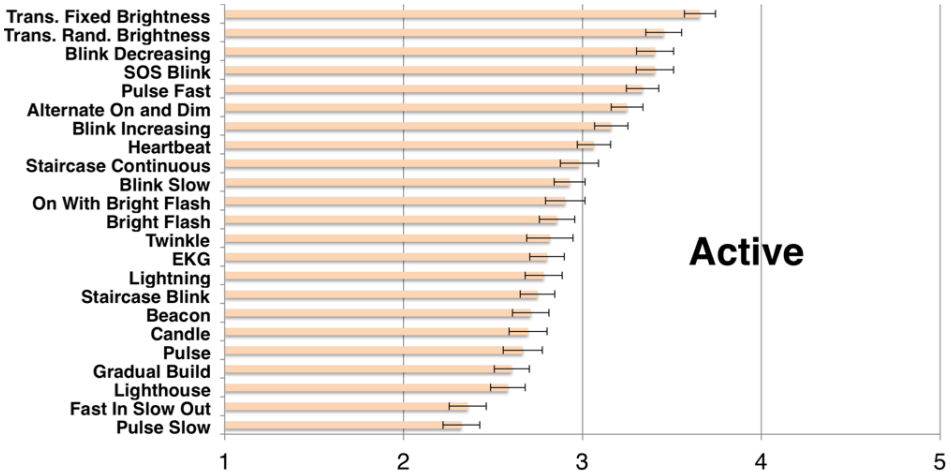


Figure 6: Ratings of light patterns for the active state (Harrison et al., 2012)

Harrison et al. (2012) mentioned, that a good light pattern should have a strong interpretation, indicated by a high Likert rating. Furthermore, it should be iconic, which means that it should represent one information state strongly in comparison to all others. With these criteria, they choose eight recommended light patterns (see Figure 7).

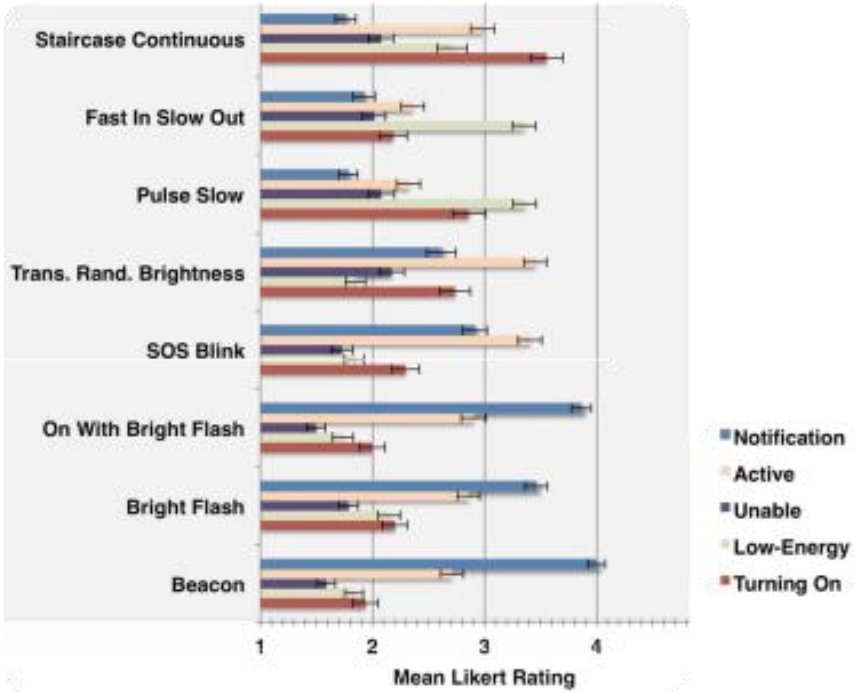


Figure 7: Recommended light patterns (Harrison et al., 2012)

2.3 DEFINITION OF SHAPE-CHANGING INTERFACES

Following, a short description of the term shape-changing interface respectively shape-changing display is given.

Poupyrev et al. (2007) define shape displays as displays, which can „directly create 3D physical shape“. Hardy et al. (2015) describe actuated shape-changing displays as displays, that „transform our interactions with technology by exploiting perceived affordances inherent in physical form“.

2.4 CLASSIFICATION OF SHAPE-CHANGING INTERFACES

First a work is presented, which gives an overview of the foundations of physical shape change. This allows a classification of shape-changing interfaces.

Rasmussen et al. (2012) classified the design space of shape-changing interfaces. They conclude, that there are four important aspects of shape change: the change in shape, the dynamics of change, the interaction and its purpose.

2.4.1 Types of Change in Shape

The authors (Rasmussen et al., 2012) distinguish between eight types of change in shape: orientation, form, volume, texture, viscosity, spatiality, adding/subtracting and permeability (see Figure 8). Following, a short explanation of the terms is given.

Orientation distorts „the original shape through rotations or changes in direction, while preserving the recognizability of the original form". *Form* retains its original volume, while "changing its overall form". *Volume*, whereas, modifies the size of an object, but maintains the overall form. *Texture* changes the surface properties of an object. This adds "visual and tactile properties without affecting the overall form". *Viscosity*, whereas, can be viewed as a specialization of *texture*. Changes in viscosity „can result in both physical shape changes and in the illusion of shape change". *Spatiality* changes the spatial position of objects. This transformation, properly speaking, doesn't modify the physical shape of objects, but through repositioning it evokes a visual illusion.

These transformations are referred to topological equivalent functions respectively homeomorphism. „Two spaces are topologically equivalent if they can be continuously stretched and deformed into another without cutting or joining distinct parts" (Marcelo Coelho & Zigelbaum, 2011). The next two transformations are not topological equivalent.

Adding or Subtracting is described as „transformations that unite or divide elements, while being able to return to the initial shape or shapes". *Permeability* changes are defined "by transformations, where the shape is perforated, but able to return to its initial shape ".

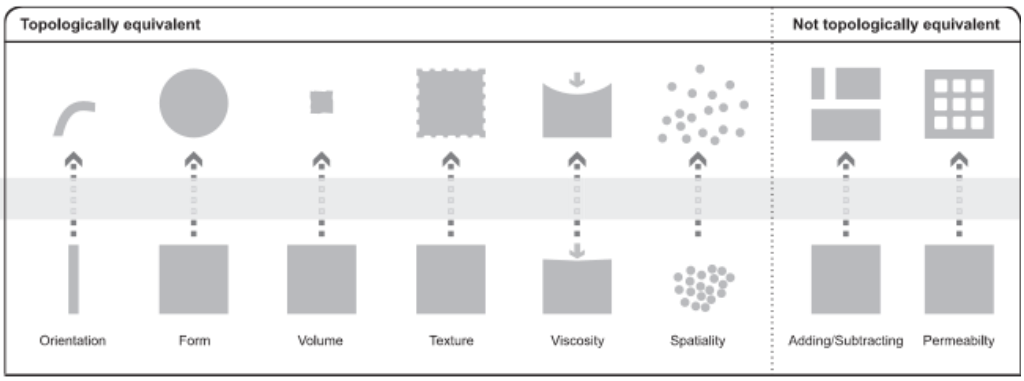


Figure 8: Types of changes in shape (Rasmussen et al., 2012)

2.4.2 Types of Transformations

The shape changes mentioned previous, describe a transformation referred to initial and end state. However, it's also important to define classification parameters between these states. Rasmussen et al. (2012) distinguish between *kinetic* and *expressive parameters*. *Kinetic parameters* should outline the physical changes like acceleration, speed and frequency. Based on that, the *expressive parameters* should reflect the perception of the kinetic parameters.

Kinetic parameters are *velocity*, *path*, *direction* and *space*.

Velocity describes "the speed, acceleration, tempo, vibration and frequency of an object's movement". *Path* illustrates the movement pattern, which can be linear/curved, continuous/intermittent, smooth/jerky or pattern/random. *Direction* describes „the direction in which the object moves "and can be up/down, left/right or forward/backward. *Space* describes „the use of space by the interface, including scale change and form change ".

Expressive parameters are divided into *association* and *adjectives*.

Association should answer the question, how much an interface is perceived as nature-like. This is because many works try to reproduce nature-like motions or properties in the interfaces. Therefore, the interface can be perceived as *organic* or *mechanical*. *Adjectives* describe the *qualities* and emotional/*personal traits*, which are assigned to the interface by the user. *Qualities* are for example soft, pleasant or peaceful. *Personal traits* include emotions like happy, sad or angry.

Rasmussen et al. (2012) mention, that it's important to prove the ascribed, subjective *expressive parameters* of the developers with user studies. They criticize, that it's usually rare to see user studies, who actually test the shape-changing interface.

2.4.3 Interaction

This aspect of shape change defines the interaction between the interface and the user. Shape change can be used as input and/or output. Thus, the authors refer to three types of interaction: *no interaction, indirect interaction* and *direct interaction* (see Figure 9).

No interaction means, that the shape is only used as output. The user doesn't interact with the interface.

Indirect interaction like no interaction uses only shape-changing output, but additional include implict input. The authors define input as implict, when "users may not realize that their actions are being used as input". Examples for implict input can be environmental sensations such as sound, movement and perhaps also personal profiles of the user.

Direct Interaction implies that there's both input and output. The authors describe this as *action and reaction* or *input and output*. Whereas the first term refers to direct feedback, the second refers to not directly related feedback. The output can also be displayed on a remote interface. This can for example allow real-time user communication.

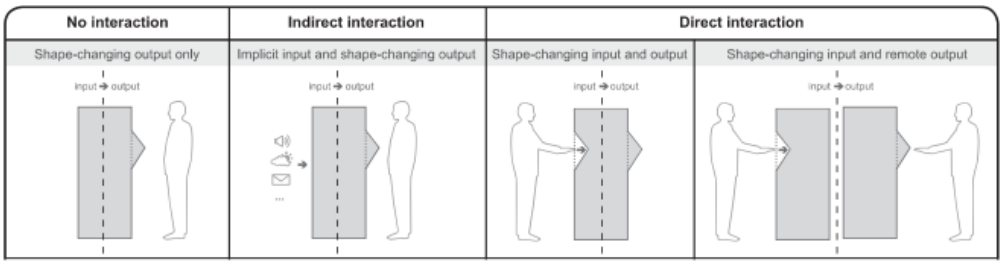


Figure 9: Interaction between the interface and the user (Rasmussen et al., 2012)

2.4.4 Purpose of Shape Change

The purpose or the aims of shape-changing interfaces can be divided into four sub-categories. Namely functional aims, hedonic aims, explorative and toolkits. The authors mention that an interface can have more than one purpose, but for mapping the design space of shape-changing interfaces, each of the purposes is discussed separately.

Functional aims for shape change describe the purpose to communicate information, model dynamic affordances and/or give haptic feedback. Furthermore, the authors name the practical and construction purpose. *Hedonic aims* should represent the non-instrumental goals like aesthetics, emotion and/or stimulation. *Explorative* aims should illustrate the technical aspect of shape change and increase "the understanding of the materials involved in shape change". For example, through usage of new shape-changing materials in simplistic prototypes. *Toolkits* are used for programming shape change through composing several parts.

2.5 SHAPE-CHANGING INTERFACES

In this chapter different works will be presented to illustrate the usage of shape-changing interfaces.

2.5.1 Field of Art

In the field of art shape-changing displays are primarily used for *hedonic aims*. An advantage is, that 3D shapes and real physical forms are clearly more accessible for especially non-technical people.

The Source, for example, is an installation in the Londoner Stock Exchange from an artist group called greyworld (2015, see Figure 10). It is built of balls, representing pixels, that are arranged in a 9x9x9 cube and are attached to metal cables. Within a height of 32 meters the balls can move and form 3D shapes. The installation is used to generate forms from live data, in order to visualize the forces in financial markets. Additional it is used to write messages and visualize forms like an uprising sun in the morning.

Another art project is Protrude, Flow (Kodama & Takeno, 2001) which uses magnetic fluid and electromagnets to form shapes. A microphone can capture surrounding sounds, which influence the forms of the fluid shape. Then there is Aegis Hyposurface (Goulthorpe, 2001), which is a 10 meters wide and 3 meters high wall. It consists of interconnected metal plates, that are actuated by pneumatic pistons. Dependent on environmental stimuli like movement, sound or light, the wall changes its shape based on real-time calculations (see Figure 11).

Figure 10: The Source (Greyworld Artist Group, 2015)

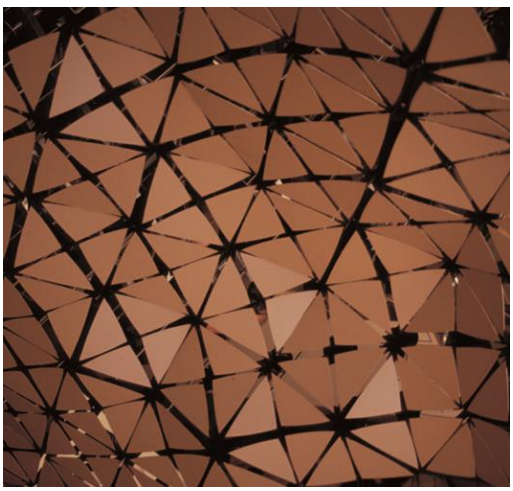


Figure 11: Aegis Hyposurface (Goulthorpe, 2001)

The concept of representing digital information on moving objects, like in The Source, is quite similar to the MovingBars, worked with in this thesis. Both can move horizontal to visualize trends respectively properties of the data. Although the MovingBars can't model 3D object, they use color leds for information representation. Also, the main idea differs, because The Source is intended to show the data in an abstract way and gives an impression of the flow of the data, rather than showing their meaning.

In contrast to the MovingBars, Protrude-Flow and Aegis Hyposurface show a representation of environmental information, rather than inform people about external digital information.

2.5.2 Field of 3D Shape Displays

Another application area of shape-changing interfaces are displays, which are based on a pin architecture. Commonly they're composed of an array of pins, which can modify their height to form 3D shapes.

Iwata et al. (2001) developed such a display, named FEELEX, which can display simple shapes. It's based on an actuator array, which is encased with an elastic rubber, nylon surface. Graphics are projected onto the screen and interaction is possible due to a force sensor on every linear respectively height-adjustable actuator (see Figure 12).

Furthermore, there is Lumen (Poupyrev et al., 2004), which is like FEELEX based on an actuator array, but without a deformable layer. There are rods, which can adjust the height of the rounded, color-changing pixels on the top. Every pixel has a capacitive touch sensor, enable interaction with the shapes on the display (see Figure 13).

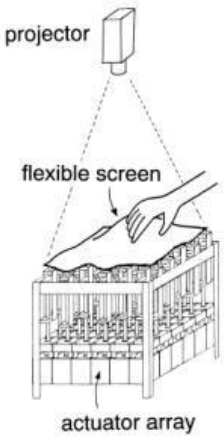
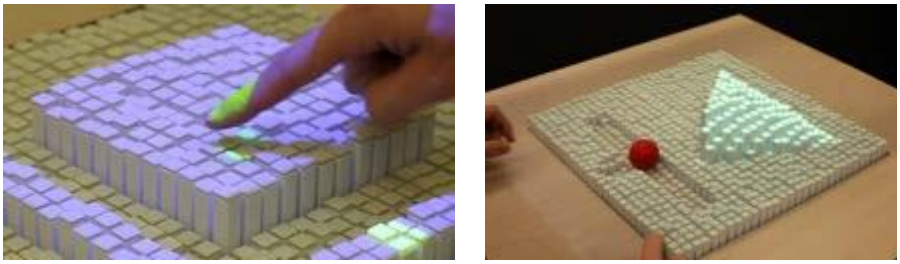


Figure 12: FEELEX setup
(Iwata et al., 2001)

Figure 13: Lumen
(Poupyrev et al., 2004)

Poupyrev et al. (2007) mention, that the main problem with these shape-changing displays is their low resolution. To get realistic textures and effects „the pixel size must shrink at the very least to 1mm“. The authors furthermore suggest to extend the traditional RGB model with the attribute of height. They call it RGBH graphics and mention, that „it can be viewed as the next step in the evolution of a pixel“. They suggest a design property for these RGBH displays, which distinguishes between synchronous and asynchronous modes. These modes should represent the relation between shape and image. In the synchronous mode „the shape extends the graphics, it adds a third dimension to flat images“. The asynchronous mode distinguishes between shape and image, so „shape and image are independent from each other and serve different purposes“.

Follmer at al. (2013) developed a display called inFORM, which uses a 30x30 pin array to model shapes, a camera to sense haptic input and a projector to provide graphics (see Figure 14a). This display has the highest resolution respectively density of pins within the investigated works. The authors showed how this display can be used for 3D model manipulation on the screen. Different tools for modelling, like in a drawing program, can be chosen by putting a ball in corresponding wells. Browsing through a menu or scaling 3D objects can be simulated through moving the ball in an appearing slot (see Figure 14b). It's also possible to deform the model by modify the pins by hand.



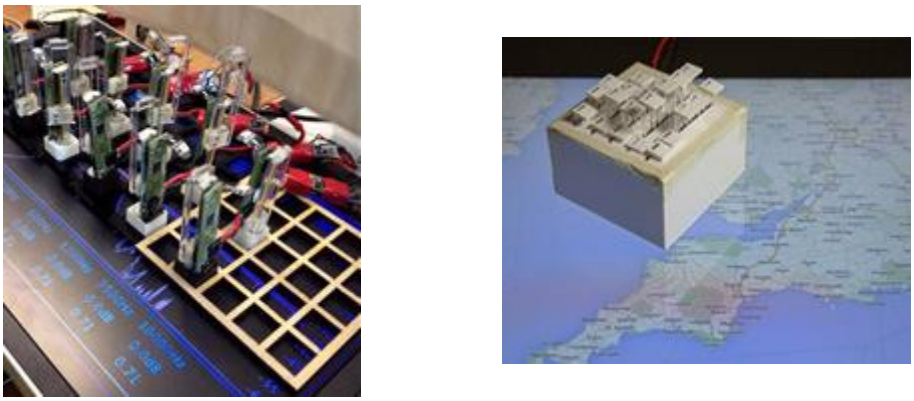
(a) (b)

Figure 14: Interaction (a) and browsing (b) on the inFORM display (Follmer et al., 2013)

The presented works focus on extending an 2D display with an additional physical height dimension to visualize objects. The MovingBars are not supposed to rendering 3D objects, but show information trends by adjusting their height.

2.5.3 Field of Toolkits

Hardy et al. (2015) developed what they called ShapeClip to build a shape-changing display. Their goal was to shorten the time of the prototyping process with a shape-changing display and lower the complexity to design it. Therefore, it was necessary to make the building parts easy to use for people without the specific technical and programmable knowledge. A ShapeClip can be placed on any display and tracks the change of the screen brightness by using Light Dependent Resistors (LDRs). Dependent on that, the clip modifies its height and output color. Furthermore, touch events can be forwarded from the top of the ShapeClip to the display. ShapeClips can be arranged in an array. With this technology, there can easily be made a shape-changing display, for example a sound equalizer (see Figure 15a) or a terrain map. Even with a single ShapeClip, trends of data can be visualized by moving it over the display (see Figure 15b).



(a) (b)

Figure 15: Shape Clips used as sound equalizer (a) or to reveal trends (b) (Hardy et al., 2015)

While Hardy et al. (2015) use physical information as input and output (touch, light), this thesis works with a moving bar, which visualizes data from a computer system. However, the concept of the z-actuation is also used by the MovingBar.

2.5.4 Field of Ambient Notifications

Pinwheels (Ishii, Ren, & Frei, 2001) is an ambient display, which can illustrate the flow of digital information through „subtle changes in movement and sound". The authors mapped different information onto the pinwheel, for instance the traffic at a subway station captured by a camera or activities in e-mail exchanges (see Figure 17).

Water Lamp (Dahley, Wisneski, & Ishii, 1998) is an ambient fixture, which uses water ripples projected by light to show digital information. The water ripples are evoked by computer-controlled magnets tapping the water surface. The authors mention, they "envisioned that ‚bits' (digital information) falling from cyberspace could create physical water ripples".

Shade Pixel (Kim & Lee, 2008) is an ambient display, which uses only shade to illustrate pixel-graphics. The display is enclosed with a deformable spandex skin, which is attached to a frame with an array of 7x11 holes. In each hole an iron core is attached to the surface layer, which can be adjusted in height by a magnet. This allows to create a sunken-like relief. Thereby graphics can be displayed on the white surface using only the visual differences in shade. Shade Pixel doesn't produce visual noise, like traditional light emissive displays. The display "can be embedded into the surfaces of everyday products such as home appliances, furniture, or other aspects of the everyday environment as an ambient display". Figure 16 shows an application of Shade Pixel as a digital watch.

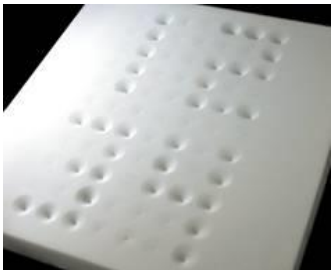


Figure 16: Watch
(Kim & Lee, 2008)

Figure 17: Array of Pinwheels
(Ishii et al., 2001)

Like these works, the Moving Bars stay in background and convey information through light, in common with WaterLamp, allowing the user to focus his main attention to other things. With height movement and light patterns, different information can be mapped onto the leds. The above works, whereas, don't use color light panels.

2.5.5 Field of Everyday Graspable Objects

Coelho and Zigelbaum (2011) build the audio recorder respectively player SpeakCup (see Figure 18), which uses shape change for *direct interaction*. It is a hand-sized silicone disk, allowing to capture and release sound by pressing it into a concave or convex form. An embedded flex sensor is used to detect the deformation. Flashing red leds indicate the recording mode and the system is shut down by flexing it to the flat position.

Shape change is also used for interactive feedback in mobile phones. Hemmert et al. (2010a) developed a box, which can be attached to a mobile phone. This box allows to modify the tapering between the front and backplane of this phone construction. The authors present some applications, which are used for interactive feedback or user notification. In mobile photo browsing the position of the current photo in an album is indicated by the shift of the backplane to the right or left, while „the phone is held in landscape format". Another application is showing the progress of a download.

The authors also developed another box, which uses weight-shift to „support mobile interactions" (2010b). In the box a servo motor is used to rotate a weight, which creates a weight shift in the overall construction. This is used to tactile illustrate a file download or the battery status.

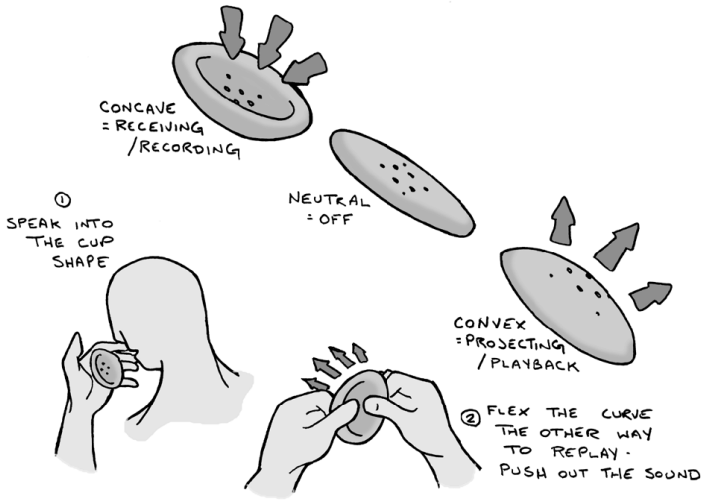


Figure 18: SpeakCup (Zigelbaum et al., 2008)

2.5.6 Field of Conceptual Interfaces

Harrison and Hudson (2009) developed a method to create changeable buttons on a visual display (see Figure 19). They use pneumatic actuation respectively air pressure to shape buttons into a surface latex layer. Previously the shape of the buttons was cut out of a solid backing layer. With fan-based pumps negative and positive pressure is created inside the frame respectively single or multiple chambers, resulting in concave and convex button shapes. The display provides graphics by rear projection and multitouch input, also because the display is made of transparent and translucent materials. This prevents occlusion while user interaction. This technique is "offering some of the flexibility of touch screens, while retaining the beneficial tactile properties of physical interfaces". Within a user study the authors figured out, that the attention to locate and press a button (secondary task) on the mentioned interface is at the same level as physical buttons, while performing a primary task. On the other hand, the authors found out, that flat displays needed much more visual attention than mentioned shape displays.

Surflex (M. Coelho, Ishii, & Maes, 2008) is a programmable and shape-changing surface, made out of foam and shape-memory alloys (see Figure 20). It's intended for "the design and visualization of physical forms". Circuit boards and shape-memory alloys (SMA), which are made out of nickel and titanium, are embedded in the 1" foam. Through electrical heating the SMA wire deforms, allowing the construction to change its form. The authors envisioned, that this can be used for "real-time computer modeling of objects and spaces, and the construction of adaptable interfaces".

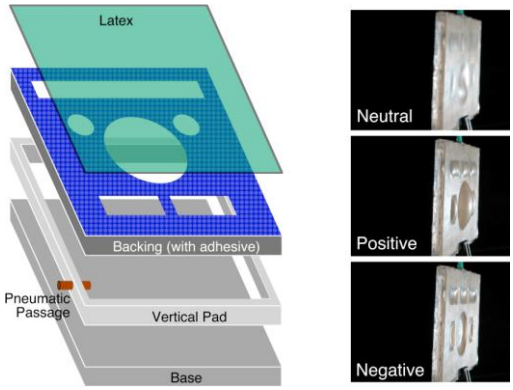


Figure 19: Dynamic Buttons
(Harrison & Hudson, 2009)

Figure 20: Surflex
(M. Coelho et al., 2008)

3

CONCEPT

In this chapter an overview of the previous and the new system will be given. This should underline the objectives of this thesis.

3.1 PREVIOUS SYSTEM

The previous setup was developed by Andre Lehnert as a part of a master thesis (2016). A schematic of the system is shown in Figure 21.

A bar is controlled via two Arduino Nano's, the first controlling the stepper motor and the second controlling the led lights. The Arduinos are connected to a Raspberry Pi with an I2C data bus. This allows to connect several bars to the system (see Figure 22). On the Raspberry Pi an outsourced Salesforce database and a running webserver is used for receiving data. A mobile android application was developed to display monitored notifications, which are feed to the oursourced database. This allows indirect control of the bars. Also a GUI was developed in order for direct control only through the webserver.

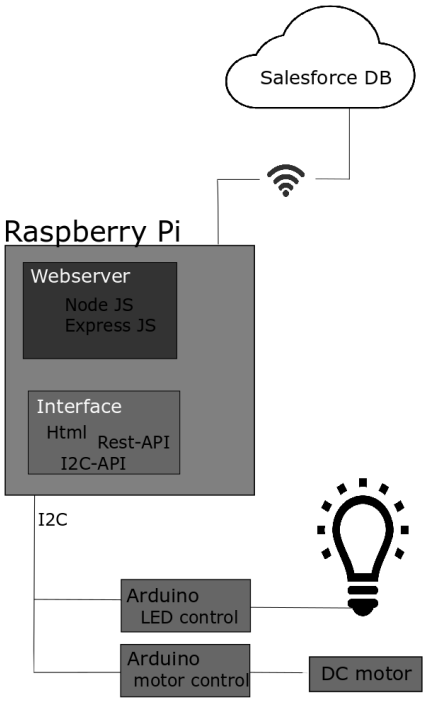


Figure 21: Previous setup schematic

Figure 22: Four Moving Bars (Lehnert, 2016)

The system is designed to run in a laboratory under observation of an expert user.

The used power supply for the system is a self-build box, which is pictured in Figure 23. Here, also the Raspberry Pi is pictured.

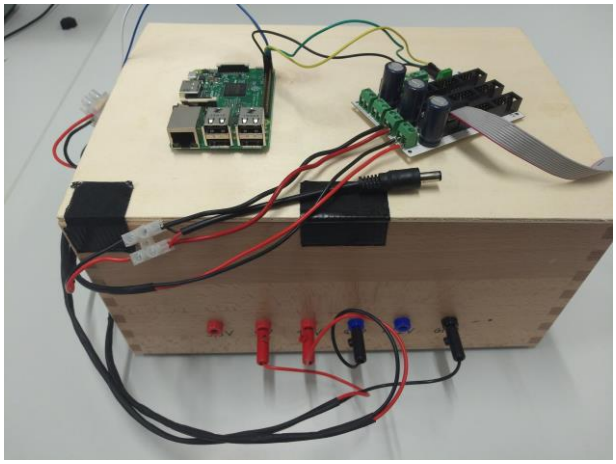


Figure 23: Power supply and Raspberry Pi

The control unit, which among others includes two ATMega328p microcontroller for each MovingBar, is illustrated in Figure 24.

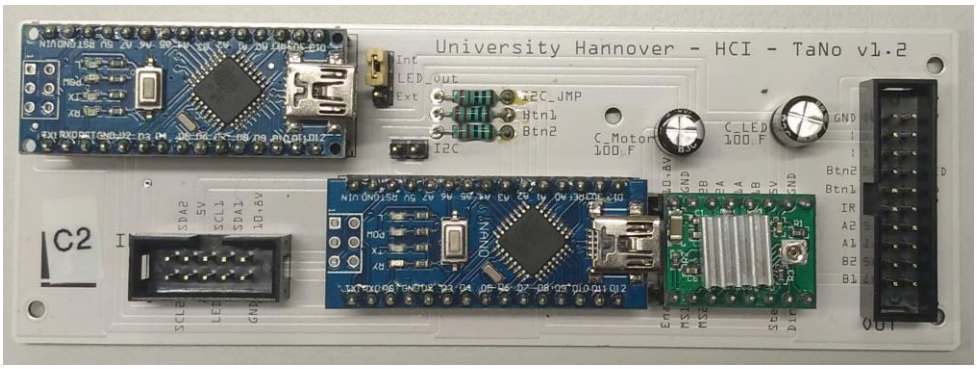


Figure 24: Previous control board

3.2 NEW SYSTEM

Folowing objectives had influence on the redesign of the existing system:

- Simplify of the wiring and power supply
- Making the system more compact, so that a user can take it with home
- Real-time display of information from Web Api's
- Local and simpler control of the MovingBar
- Control the stepper motor and the lighting simultaneously

The various connection cables will be reduced to two visible connection wires. First the power supply and then the signal connector for the MovingBar. This decrease errors and increases usability, when a not experienced user tries to setup the system in his/her environment.

In order to make the system more compact, the existing power supply unit is exchanged by a much smaller supply. An encasing for the system controller components, which will be explained later in more detail, will be fabricated. In the previous setup multiple MovingBars are attached to a metal framework. To improve protection of the components at the bottom and enable a more stable stand, a pedestral is designed.

The system can be configured by an user, so that information respectively quantitative data from web sources can be displayed on the MovingBar. The display of information is done with the led lights on the sides of the MovingBar and height adjustment with the stepper motor. Therefore four different scenarios can run simulatenously. Various illustration properties can be specified, like the led colors or the display mode. A graphical user interface is implemented, which allows the user to set previous mentioned properties.

The led lights and the stepper motor are controlled simultaneously by a single microcontroller. This reduces the error probability of the system, since with less hardware, less errors will be likely.

The schematic of the new setup is illustrated in Figure 25. A Raspberry Pi holds all information to display data and is setup with a GUI via a SSH connection. (A PCB Board is used for holding one Atmega328p, the stepper motor driver A4988 and power supply components.) The led stripes and the DC stepper motor can be controlled simultaneously. The Raspberry Pi and the PCB Board are connected with SPI.

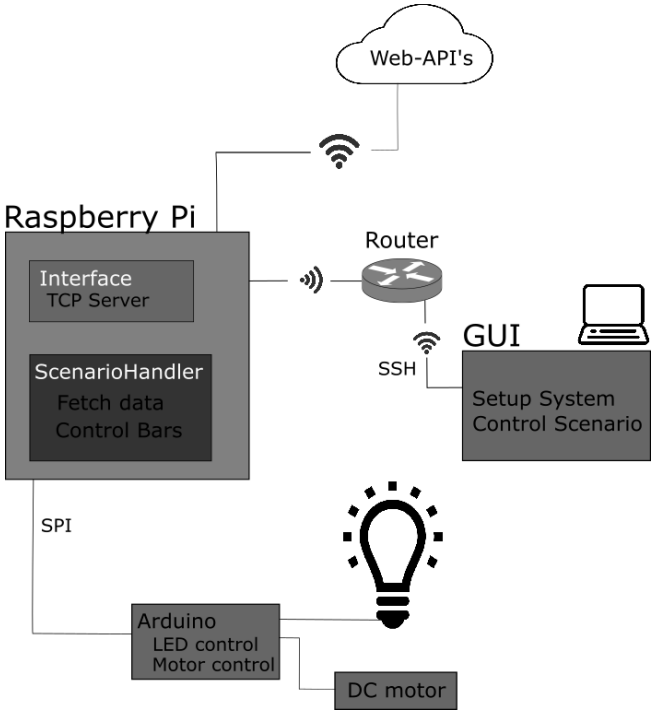


Figure 25: New setup schematic

The complete buildup of the new system is shown in Figure 26. This includes the MovingBar, the controller box and the power supply.

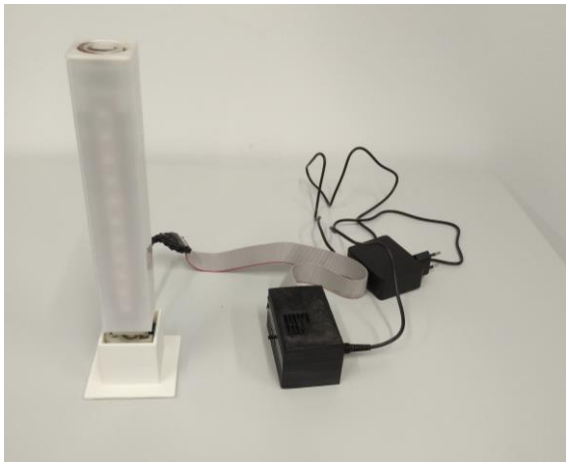


Figure 26: New setup, all components

The components inside the controller box are shown in Figure 27. On the left side is the bottom and on the right side the top part of the encasing. The bottom part holds a printed circuit board with various components. The top part contains the Raspberry Pi Zero W.

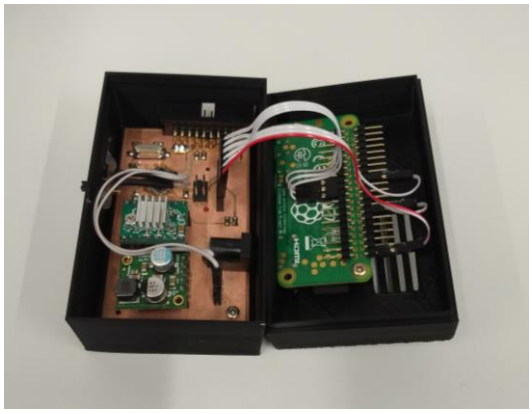


Figure 27: Components in the bottom and top part of the controller box

3.3 VISUALISATION CONCEPT FOR DISPLAYING INFORMATION

For the MovingBar has vertical led stripes and can modify its height, the category of quantitative data respectively information is considered for visualisation.

Therefore, following scenarios give an overview over the various usage applications, which can be displayed with quantitative information:

- Falling or rising stock market values
- Bidding price of an article on an online marketplace
- Number of new e-mails in inbox mail folder
- Number of new notifications of a specific application
- Weather data, for instance temperature, rainfall and sunhours
- Progress of a file download

In order to display information, a reference value has to be chosen. This value is displayed as a single led on the MovingBar, which has a different color than the other leds. So, it can be easily distinguished by the user.

In the later description of the implementation, two display options for quantitative data are developed (see Figure 28). The first displays the data on the led stripe from bottom to top. The reference led, which corresponds to the reference value, is set to the lowest position. This led should indicate, that the displaying of information is still in process. The reference led can also be disabled, to have more leds to display data.

The second option should allow a relative illustration of data, which reference value isn't located at the zero position. As a result, negative values can be displayed, related to the reference value. Therefore, the reference led is located in the middle of the led stripe. When the current value is lower than the reference value, the quantity is displayed from top to bottom, starting below the reference led.

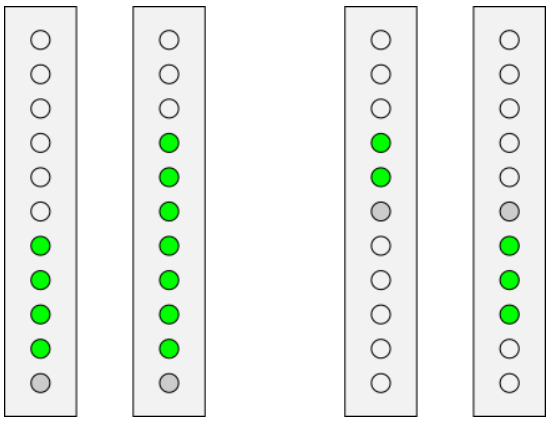


Figure 28: Bottom to top (left) and relative (right) information representation

Furthermore, the display range of the bottom to top mode is enlarged by including height movement. This allows a display range of nineteen, rather than eleven leds. In this mode, only one scenario can be displayed on the MovingBar, because the height movement falsifies the values on the other sides.

Figure 29 illustrates this display mode. From left to right the value respectively the number of leds increases. The reference led disappears, when the current value exceeds the corresponding value range for the led at the top. This is shown in the transition between the two MovingBar displays in the middle. When the number of leds further increases, the MovingBar will adjust the height, which is shown with the last display. The height movement step size is chosen as a multiple of the distance between two consecutive leds. This provides a better association between the height and the not displayed number of leds.

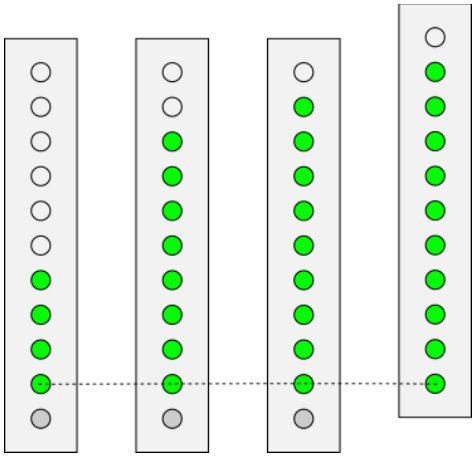


Figure 29: Bottom to top with height movement

Blink animations are used, to inform the user of different notifications. The MovingBar supports two types of animations, which implementation will be explained later. First the overall animations, which run on all sides simultaneously and thus clears the quantitative led stripes. This animations can be used in a waiting state, when no scenario is running. For example, when the system is set up for the first time, to indicate that the system is connected.

Then there are blink animations, which address single led lights. When the system is in normal operation mode, the visible feedback of the current system state relies only on the height movement and the lightning of the leds. So, when there's an error or warning in the system, the MovingBar has to indicate this to the user. Therefore, simple and intuitive notification types have to be chosen. Table 3 gives an overview over the implemented feedback notifications.

Event	Blink pattern	Led number	Color
Led stripe overfilled	Beacon	11(highest)	display color
URL unreachable	Trans. fixed brightness	11(highest)	yellow
XML/JSON parse error	Trans. fixed brightness	11(highest)	orange
WLAN unreachable	Pulse Slow	11(highest)	red

Table 3: Feedback notifications

The supported blink patterns are beacon, transmission fixed brightness, staircase and pulse slow (see chapter 2, Figure 4). All feedback notifications will be displayed on the highest led, to maintain a coherent reference point.

The first event, led stripe overfilled, is shown in the same color as the display leds for the quantitative data. This is, because it's no error notification and shouldn't distract the user. It should rather indicate, that the chosen step size should be increased. The blink pattern beacon is chosen, because it has a high ranking for notification feedback and also a high ranking for the active property (see chapter 2.2). This is appropriate, because when the led stripe is overfilled, it's a warning and no error. This should indicate, that the bar is active respectively is working normally.

The events "URL unreachable" and "XML/JSON parse error" are displayed in the transmission fixed brightness blink pattern in the color yellow and orange. This blink pattern has also a strong interpretation for notification feedback and the active property. The colors yellow and orange are chosen, because of their low mean pleasure level (see chapter 2.1, Figure 3). This low pleasure level is connected to mostly negative emotions (see chapter 2.1,Table 1), which should represent the error event. To reinforce the lower arousal level of these colors a high saturation is chosen. This is, because a higher saturation correlates with a higher arousal level.

Previously the blink patterns for the led stripe overfilled and the just mentioned error events were swapped. This was done, because the beacon blink animation has a lower active interpretation than the transmission fixed brightness animation. A lower active ranking for the error and warning events should indicate that something isn't working properly. However, it was discovered, that the transmission fixed brightness animation was to obtrusive for the relative frequent overfilled event.

When the WLAN is not properly connected or not available, the pulse slow blink pattern is used. This pattern indicates primarily a low-energy state, which represents the low availability of the WLAN. The color red is used, because it's the most common color for error representation. To reinforce the arousal level, the brightness is reduced, and a high saturation is selected.

4

HARDWARE & TOOLS

This chapter explains the used hardware and tools with reference to the topics, which are relevant for the later implementation process.

4.1 USED HARDWARE

In this section the specifications and the functionalities of the used hardware components are presented.

First the data transfer interface SPI will be explained. This is used to allow message transfer between the microcontroller ATmega328p and the Raspberry Pi Zero W, which will be specified next. Then the functionality of the led stripes will be explained .Then the stepper motor driver board A4988 is presented, which is used to adjst the height of the MovingBar with a stepper motor. Finally, the setup of the power supply system will be explained.

4.1.1 SPI

The Serial Peripheral Interface (SPI) is a synchronous communication interface, which allows a master device to send and receive data from multiple slave devices. For this, four ports are necessary:

- SCLK is the clock signal, which is evoked by the master device. This is used to synchronize the data transmission. The frequency should not be higher than the speed of the slave device.

- MOSI (Master Out Slave In) is used to send data from master to a slave device.

- MISO (Master In Slave Out) is used to receive data from the slave device.

- SS (Slave Select) specifies the slave device, with which the master wants to communicate. When it is low, the device is activated.

The data transmission is word based, so that which each send word on MOSI, the slave puts one word on MISO. The word length are often a multiple of 8 bit. For sending and receiving, shift registers are arranged in a virtual ring topology. Since there's no official standard, devices sometimes implement SPI slightly different (Arduino LLC, 2017).

4.1.2 Led Stripes

The led stripes are build of single RGB leds of the type WS2812B[1]. These leds are arranged in a daisy chain. Every led has an integrated controller, which allows to forward data to the next led. In Figure 30 the WS2812B pinout is illustrated. DIN is used for the data input and DOUT for the data output. The pin VDD is the supply power and GND is the ground potential pin. The supply power of 5V and ground should be applied at the first led in the linkage, since all following leds are connected with each other.

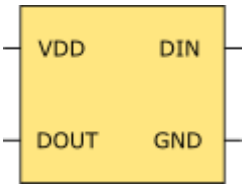


Figure 30: WS2812B

Each led component holds 24 bits for the color, since the red, green and blue component are each encoded with 8bit.

To update one led, a message of pulses, which decode logic high and low, is sent down the dataline. Each led mask out its color and writes it to an internal buffer. Then it sends the remaining message to it's successor. When the dataline is hold low for more than 50 µs, it indicates that a new message will be send next.

Furthermore, the internal buffer respectively the last message is copied into an additional register, which controls the lighting.

4.1.3 Atmega328p

The Atmega328p is a microcontroller from Atmel and, among others, is used in the Arduino Uno. It has 32KB ISP flash memory to store program code, 1kB EEPROM to store setting data and 2KiB of RAM. It has two 8-bit and one 16-bit timer/counter, which will be explained in the next paragraph in more detail. The microcontroller supports external and internal interrupts. It has 23 GPIO pins and 32 general purpose working registers. Furthermore, it supports various communication interfaces, namely SPI, I2C and UART. The maximum operating frequency is 20MHz.

[1] WS2812B datasheet. https://cdn-shop.adafruit.com/datasheets/WS2812B.pdf, visited 04-02-2018

The timers respectively counters of the Atmega328p work independent of the CPU. Therefore, it is possible to enable limited types of multithreading, so that tasks can be executed in parallel.

The timers are counting with the corresponding register, up from zero. When the register overflows, an interrupt can be triggered. Since the processor clock is 16MHz, the 8 bit respectively 16 bit register throw an overflow interrupt in a few milliseconds. To provide longer overflow periods, a prescaler is used to specify that the register will increment only with every n clock cycle. Furthermore the initial value of the register can be set. This allows a more precise setting.

An overview of the 8bit timer is shown in Figure 31 .The schematic of the 16bit timer can be found in the appendix (see Figure 55). In the next paragraph the most important functionalities and registers of the timers are explained.

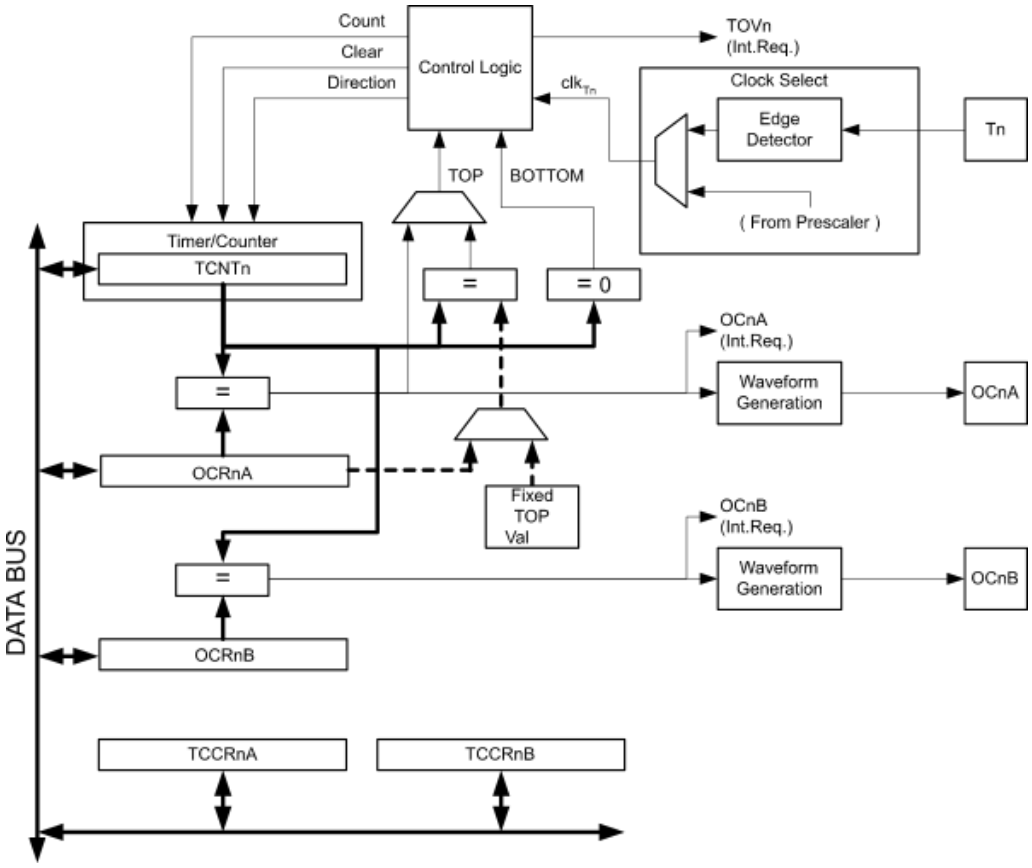


Figure 31: Schematic of one of the Atmega328p's 8-bit timer
(ATmega328p Datasheet, 2016)

There are multiple registers, which are necessary to implement previously named settings. The x in the next section specifies the number of the timer.

First there is the timer/counter register TCNTx, which is incremented continiously. Each timer has two output compare registers OCRxA and OCRxB. They are compared with the timer register and are used to generate a periodic signal on the output compare pins OcxA and OcxB. These compare pins are equal to I/O pins of the ATmega328p, but have to be activated on these I/O pins. Activation and choosing if the compare match output will be high or low, can be set in the register TCCRxA. Also the counter behaviour can be set in this register. This specifies, if the counter value should be cleared on a compare match (CTC mode) or not be cleared on a compare match (normal mode).

The prescaler is set through choosing a proper clock source (TCCRxB). This is done with setting the clock selection bits in this register (CSx, see Table 4).

CSx2	**CSx1**	**CSx0**	**Description**
0	0	0	No clock source
0	d	1	Prescaler = 1, therefore no prescaler
0	1	0	Prescaler = 8
0	1	1	Prescaler = 64
1	0	0	Prescaler = 256
1	0	1	Prescaler = 1024
1	1	0	External clock source, clock on falling edge
1	1	1	External clock source, clock on rising edge

Table 4: Clock selection bits description

Furthermore a flag/bit (OCFxA, OCFxB) can be set in the interrupt flag register TIFRx, to trigger an interrupt service routine. The interrupt service routine respectively the interrupt vector can be specified in the interrupt mask register TIMSKx. A complete list of the supported interrupts of the ATmega328 can be found in the appendix (see Table 9).

4.1.4 Stepper Motor Driver Board

The MovingBar can adjust its height with a stepper motor, which is addressed by the A4988 stepper motor driver carrier. This allows more simple control of the stepper motor, because direction and step signal have a pin interface. Besides there is a thermal overheat shutdown protection and an under-voltage lockout. This breakout board supports different step modes respectively speeds (full, half, quarter, eigthth, sixteenth). Each step mode specifies the covered distance of one step differently. The covered distance of a step in full mode corresponds to the distance of two steps in half mode.

The basic wiring, recommended by the manufacturer, is shown in Figure 32. The step modes are configured with the pins MS1, MS2 and MS3. This allows different speed modes, due to intermediate voltage levels on the motor power supply. The stepper motor is driven by the pins 2B, 2A, 1A and 1B. The STEP input is used to apply a pulse to the board. With every low-to-high transition, the motor propagates one microstep. The minimal period length duration of the STEP input, which can still be detected, amounts approximately 2µs. The DIR pin represents the direction of the stepper motor and is dependent on the wiring of the motor. The ENABLE pin enables (logic low) and disables (logic high) the motor. The RESET and SLEEP pins are not used and are connected to prevent a floating RESET pin.

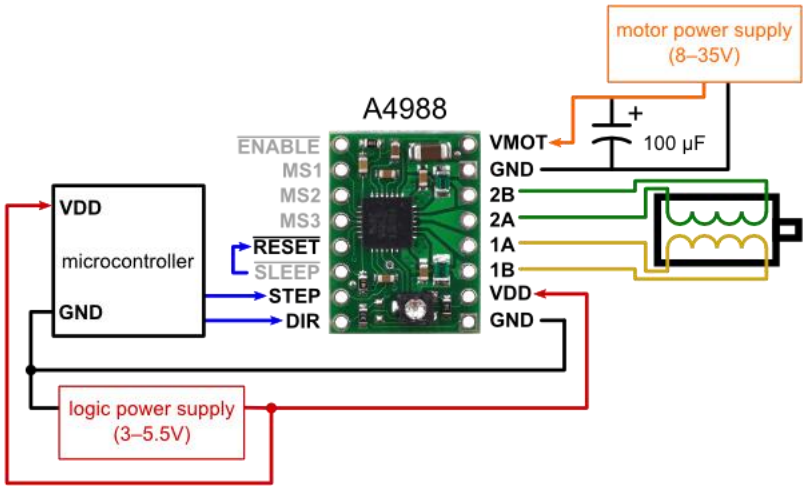


Figure 32: Basic wiring of the A4988
(Stepper Motor Driver Board, User Manual)

4.1.5 Power Supply

The system is powered with two voltage levels. The led lights, the ATmega328p and the logic of the A4988 require 5V input voltage. Whereas the motor respectively the A4988 motor supply needs 12V.

A power adapter (see Figure 33) provides the necessary 12V with a current of 3A. The input should be alternating voltage of 100-240V and 50/60 Hz, so it can be plugged in normal wall sockets.

To reduce the 12V to 5V, the Pololu 5V,5A step-down power regulator[1] is used (see Figure 34). VIN has to be connected to the input voltage 12V and VOUT to the output voltage 5V. The enable pin (EN) can be used to disable the board. Enable unconnected, the board will be active by default, due to a 100kΩ pull-up resistor.

Figure 33:Power adapter

Figure 34:Step-down voltage regulator

4.1.6 Raspberry Pi

A Raspberry Pi Zero W is used, because it has enough processing power and due to the space requirements. The Raspberry Pi Zero W has a width of 30 mm and a length of 65mm. It has a microHDMI output, one microUSB port for power and one USB On-the-Go (OTG) port for data transfer. Also it supports SPI and I2C. Furthermore, it has a build-in bluetooth and wifi module.

4.2 USED TOOLS

In this section, the used tools and equiqment will be presented.

4.2.1 AVR Programming Stick

The smartUSBLight AVR ISP programming stick (see Figure 35) by myavr is used to program the ATmega328p. The In-Sytem-Programmer (ISP) can be connected to a computer via the USB interface. The microcontroller should be equipped

[1] Pololu 5V, 5A Step-Down Voltage Regulator. https://www.pololu.com/product/2851, visited 04-02-2018

with a SPI interface, which is used for data transfer. The programmer can supply the microcontroller with 5V or 3,3V. Furthermore, the clock frequency of the programming stick is reduced automatically, until it matches the clock frequency of the microprocessor.

Figure 35: AVR programmer

4.2.2 3D Printing

For printing an enclosure and a pedestal, the Lulzbot Taz 5 3D printer[1] is used (see Figure 36). It has a print area of 298mm x 275mm x 250mm, which is sufficient for the printed components. The printed material, which is usually a synthetic material, is build layer by layer on the ground plate. To prevent an early solidification, when still printing, the ground plate can be heated up to a temperature of 120°C. The print tolerance can be set from 0.05mm to 0.35mm.

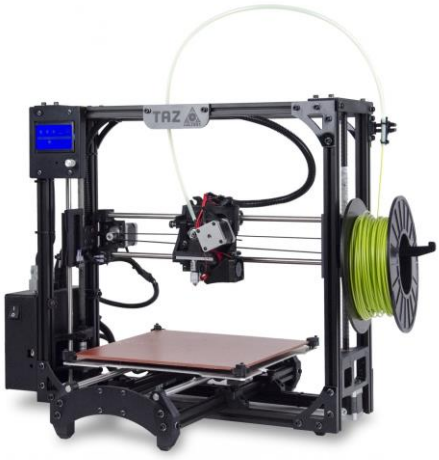


Figure 36: Lulzbot Taz 5 3D printer

For modelling the parts, the free software OpenSCAD[2] is used. This software allows to design complex 3D objects out of simple shapes, like cuboids, cylinders or spheres. For this, a script file has to be written, which is compiled to a graphic

[1] Lulzbot Taz 5 Manual. http://download.lulzbot.com/TAZ/5.0_0.5noz/documentation/Manual/Manual.pdf, visited 21-01-2018

[2] OpenSCAD User Manual. http://www.openscad.org/documentation.html, visited 21-01-2018

representation. The finished model has to be rendered and then can be written to a STL file.

The software CURA[1] allows to control the 3D printer. All necessary parameters can be set, like the accuracy or the head temperature. The STL model file can be imported and printed on the 3D printer.

[1] CURA Lulzbot Edition. https://www.lulzbot.com/cura, visited 21-01-2018

5

DEVELOPMENT & IMPLEMENTATION

In this chapter the development of the different components and the implementation of necessary programs will be explained.

First the implementation and the setup of the ATMega328p is described. The main new functionalities like the SPI connection and the simultaneous control of the leds along with the stepper motor will be specified. Furthermore the realization of blink animations will be explained.

The design of a printed circuit board, which ,among others, holds the ATMega328p, the stepper motor controller and the peripheral connectors, will be explained next. Then the pin mapping and the corresponding functionalities of the different signal headers, which are attached to the circuit board, is described. Then the most important programs, which run on the Raspberry Pi, are specified along with an explanation.

A graphical user interface allows the creation of scenarios, which refer to informational needs, like for example display temperature or rain. A walkthrough of the different windows and views explains the functionalities of the user interface.

Finally, the design and fabrication of the encasing and the pedestal with a 3D printer will be described.

5.1 SOFTWARE SETUP OF THE ATMEGA MICROCONTROLLER

In this section the software setup, along with an explanation of the most important functionalities, of the ATmega328p is provided.

This includes implementation details of the SPI connection and the blink animations. Furthermore, the control of the stepper motor, with the assistance of the previous mentioned timers of the ATmega (see 4.1.3), will be explained. Then the used message protocol will be described.

5.1.1 Implementation of the SPI Connection

The SPI connection should be provided between the Raspberry Pi and the ATmega328P microcontroller. A voltage divider is used on the MISO connection to reduce the output voltage of the ATmega from 5V to 3,3V (see Figure 37).

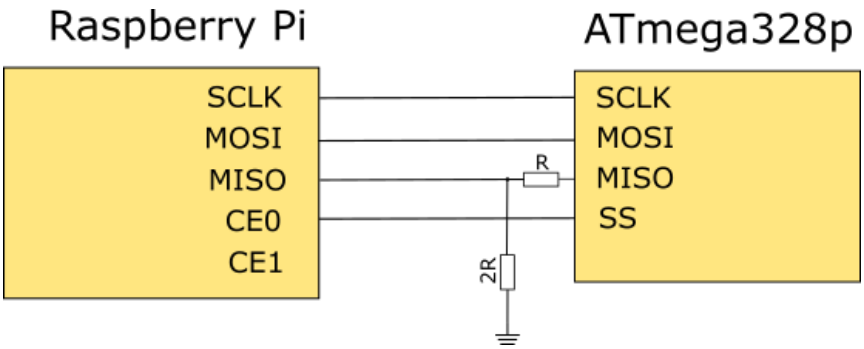


Figure 37: SPI connection with voltage divider.

For the Raspberry Pi, a module named *spidev*[1] is used to enable *SPI* (see Table 5). The files */dev/spidev0.0* and */dev/spidev0.1* represent the connections on CE0 and CE1.

Method	**Description**
`spidev.SpiDev()`	Creates and returns a SPI object
`spi.open(bus,device)`	Opens the SPI bus with the corresponding device
`spi.close()`	Disconnects from the SPI device
`spi.xfer2(list of values)`	Sends a list of values and receives response from SPI device, holds CEx permanent active

Table 5: Used *spidev* methods

In order to send a message to the ATmega328p, a byte list with appended new line character is created and spi.xfer2 called (see Listing 1).

[1] Spidev module for python. https://pypi.python.org/pypi/spidev, visited 21-01-2018

```
1  def send_message(self,message):
2      data=[]
3      for c in message:
4          data.append(c)
5      data.append(ord('\n'))
6
7      resp = self.spi.xfer2(list(data))
8      time.sleep(0.01)
```

Listing 1: Send message via SPI

The ATmega328p microcontroller runs a modified version of a source code by Nick Gammon[1] for enabling SPI. With an interrupt service routine (ISR), data traffic on the MOSI link is monitored.

The interrupt service routine is a special function, which doesn't return a value or takes parameters. Data exchange between the main program and the interrupt service routine should be implemented with global variables. For these variables the datatype "*volatile*" should be used. This tells the compiler that the variables can be changed from other routines or threads, which are not in the control scope of the declaring section. The ATmega328p supports interrupts on all signal pins (14 digital I/O pins, 6 analog input pins).

Next the source code[2] for enabling SPI is explained. Listing 2 shows the initialization of SPI on the ATmega328p. For this, first the MISO pin is defined as output port, to send data to the Raspberry Pi. In lines 6 and 9 the SPE-bit and the SPIE-bit of the SPI Control Register (SPCR) is set to logic high. This enables SPI in slave mode and enables SPI interrupt. The _BV(n) method stands for "bit value" and is equal to a left-shift of 1 n-times.

[1] SPI source code by Nick Gammon. http://www.gammon.com.au/forum/?id=10892, visited 21-01-2018

[2] Arduino source code can be found on Github. https://git.hci.uni-hannover.de/ChristianA/bachelor-thesis/tree/master/motor_lighting_controller, visited 30-01-2018

```
 1  void setup(){
 2      //have to send on master in, *slave out*
 3      pinMode(MISO, OUTPUT);
 4
 5      // turn on SPI in slave mode
 6      SPCR |= _BV(SPE);
 7
 8      // turn on interrupts
 9      SPCR |= _BV(SPIE);
10  }
```

Listing 2: Initialization of SPI

In Listing 3 the interrupt service is specified. The argument SPI_STC_vect defines, that the interrupt is triggered, when the SPI serial transfer is completed. In comparison, there are other interrupts featured by the ATmega328p like a timer compare match or timer overflow (see appendix, Table 9).

The SPDR register is used to store the received byte message. Furthermore, it can be used to send a message back, by modify its value in the interrupt service routine. With the next interrupt triggered, it will be fully transmitted to the master.

If a new line character is received, the message is completed and a global volatile-processing flag is set (line 8 and line 9). Next the message will be parsed.

```
 1  // SPI interrupt routine
 2  ISR (SPI_STC_vect){
 3      byte c = SPDR;  // grab byte from SPI Data Register
 4      // add to buffer if room
 5      if (pos < sizeof message){
 6          message [pos++] = c;
 7
 8          if (c == '\n')
 9              process_it = true;
10  }
```

Listing 3: Interrupt service

5.1.2 Stepper Motor Control

Since one objective is to enable simultaneous regulation of the stepper motor and the led stripes, this section presents a method for this.

As mentioned previous (see 4.1.3), the ATmega328p doesn't support multithreading. Therefore timers/counters are used to generate periodic pulse signals with a specific frequency and duration. This allows to control the stepper motor, because it expects a periodic signal to adjust the height. Every pulse is interpreted as a microstep, so it is possible to set the speed with the frequency and the number of microsteps with the duration.

Listing 4 shows the method, which is used to control the stepper motor. This source code was developed by Björn Fiedler. The method interconnects two timers. The second timer generates the pulses, which address the stepper motor driver A4988. The first timer counts the pulses and disables the motor, when the requested number of microsteps is reached.

The method takes the number of microsteps as an argument. In the first line the time duration between a rising and falling edge is set to 500µs. Therefore, the signal has a periodic length of twice this value, namely 1ms.

```
1  #define PERIODIC_TIME 500 //micro seconds
2
3  void startPulses(unsigned int count) {  //don't call with 0 as argument
4
5    TCCR1B = 0; //disable timer1
6    TIFR1 = (1<<OCF1A); //clear register
7    TCNT1 = 0;
8    TCCR1A = (1<<COM1A1)|(0<<COM1A0); //clear enable pin on next compare match
9    TCCR1C = (1<<FOC1A);              //Force compare match to clean enable
10
11
12   //Timer1 counting the pulses
13   TCCR1A = (1<<COM1A1)|(1<<COM1A0); //set enable pin (low active)
14   OCR1A = count-1;
15   TCCR1B = (1<<CS12)|(1<<CS11)|(1<<CS10); //Clock T1 falling, normal mode
16
17
18   //Timer2 generate pulses
19   DDRD = (1<<PD3);      //set pin 3
20   TCCR2A = (1<<COM2B0)| //Toggle output PD3 on compare match
21            (1<<WGM21);
22   TCCR2B = (1<<CS22)|(1<<CS21)|(0<<CS20); //Prescaler 256 --> 62.5 kHz
23   OCR2A = PERIODIC_TIME/16-1;             //62.5 kHz = 16 micro seconds
24   OCR2B = 0;
25 }
```

Listing 4: Stepper motor control with zero cpu load

In line 8 the compare output mode is specified, so that if a timer overflow occurs the output signal on OC1A will be logic low. OC1A rescpectively the I/O pin PB1 is connected with the enable pin of the stepper motor driver. With a forced compare match in line 9, the enable pin will be set to logic low and the stepper motor is enabled.

Since the first timer should count the pulse number of an external signal, this signal is applied as an external clock. This is specified with setting the clock select bits in the timer control register of timer one (line 15, see also 4.1.3, Table 4). The output compare register OCR1A is set to the number of pulses minus one, since the timer is starting from zero (line 14). When a timer overflow occurs, the register OC1A rescpectively the I/O pin PB1 is set to logic high. Compare match with logic high is set in line 13. The pin PB1 is equivalent to the pin 9 and is connected to the enable pin of the stepper motor controller. Therefore, when the timer overflow occurs, the enable pin will be set to logic high and the stepper motor will be disabled.

Now it is time to setup the second timer, which generates the step pulses. First the output driver of the I/O pin PD3 respectively pin 3 has to be activated. This is done in the data direction register DDR in line 19. Since the pulse signal should alternate between logic high and low, the signal output OC2A should behave equally. Therefore, the toggle output mode is set in line 20. Furthermore, the signal should be repeated continuously. For that, the clear timer on compare match mode (CTC mode) is activated in line 21. As mentioned above, the output signal should have a period length of 1ms. So an appropriate output compare register value has to be chosen dependent on the frequency of the timer clock. In line 22 an prescaler of 256 is specified. With the following formula the value of the output compare register can be computed.

$$OCRxA = duration * \frac{frequency_{cpu}}{prescale}$$

The second term specifies the new clock speed, used to increment the timer/counter register. With a cpu frequency of 16 MHz and a prescaler of 256, the frequency is 62.5kHz. This is equal to a period length of 16 µs. Therefore, in line 23 the time duration is divided by 16. This results in an OCR2A value of 32.

Since the OC2A pin corresponds with the MOSI pin, needed for SPI data transfer, OC2B on pin PD3 is used. In line 24 OC2B is set to zero, so when the counter register is cleared, the output signal is toggled. This results in one clock cycle delay, which is substracted in line 23.

The pinout of the ATmega328p, along with the previous explained pin identifiers, is shown in Figure 38.

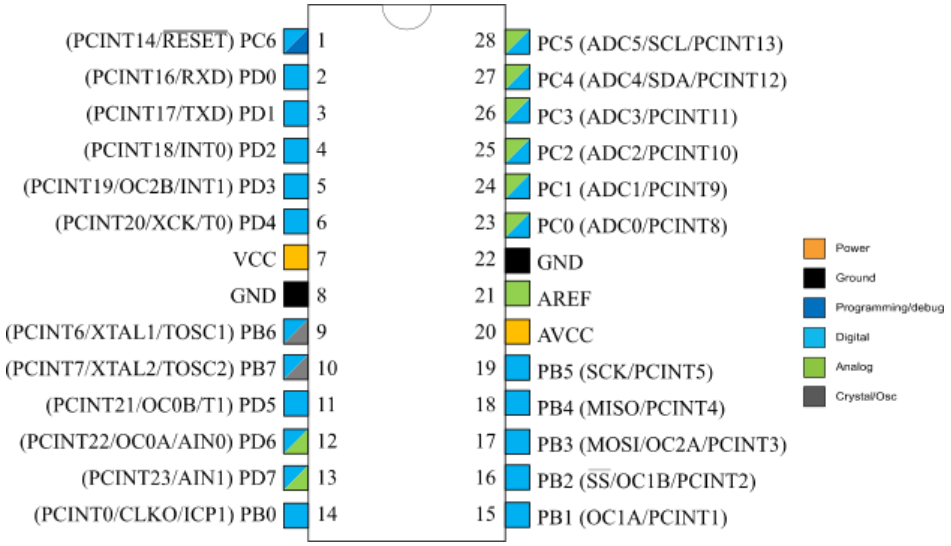


Figure 38: ATmega328p pinout (ATmega328p Datasheet, 2016)

5.1.3 Blink Animations

Next, the implementation and the functionalities of the blink animations will be described.

To control the leds of the MovingBar, a modified version of the arduino animation library by Andre Lehnert is used. This previous animation library is a memory efficient modification of an initially created animation library[1].

The used animation library internally uses an array to store the colors of each led. The MovingBar's led display consists of four cascaded led stripes, which represent each side. Therefore, 44 leds need to be addressed. To update the leds, the data in the array is send down the dataline (see led type specification, section 4.1.2).

The library distinguishes between the running of an animation and the setting of single leds on the stripe.

For the first purpose, the computation and sending of the animation data has to be executed periodically. The computation of the led values is done in appropriate functions, which are called via reference. Here, the computed values are written to an internal buffer, which will then be applied to the led stripe. The update speed can be specified and is chosen to 200ms, which is sufficient for the smooth display of the implemented animations. When there are no running animations,

[1] Led Animation Library. http://yaab-arduino.blogspot.de/p/ala.html, visited 25-01-2018

the refresh loop will be disabled in order to minimize the cpu utilization. Single leds are set and updated independently of the animations. This mode is used to display the quantitative information.

The implemented animations are intended for single leds. Different light patterns, like beacon, staircase continuous, pulse slow and transmission fixed brightness were created (see chapter 2.2, Figure 4). For each animation, an array of sample values is initialized, which represents its characteristic pattern (see Listing 5).

```
1   //intervall: even entries=active, uneven entries=inactive
2   int beaconIntervalls[]={3,2,3,8,3,2,3,8};
3   uint8_t beaconSampleValues[NUMBER_SAMPLE_VALUES]={};
```

Listing 5: Beacon light pattern sample values

This sample values will be interpreted to a led state in the corresponding function (see Listing 6). Therefore, the index of the current necessary value has to be determined (line 5). This is done by modulo division of the elapsed time by the specified duration of one animation cycle. Then this time value is mapped to the corresponding index.

```
 1   void AlaLedLite::beaconLed(){
 2       unsigned long time= millis()-animStartTime;
 3       long duration = time%durationOneCycle;
 4
 5       int index = map(duration, 0, durationOneCycle, 0,
 6                       NUMBER_SAMPLE_VALUES);
 7
 8       if(beaconSampleValues[index]==1){
 9           setLedsOn(singleAnimationColors);
10       }else if(beaconSampleValues[index]==0){
11           setLedsOff();
12       }
13   }
```

Listing 6: Beacon blink animation function

The specified light pattern can be run on each side. On each side, the number of the led and the color can be specified. This allows every side to display notifications.

5.1.4 Message Protocol

The ATmega328p receives and parses messages from the Raspberry Pi Zero W. The messages are encoded to reduce data length and redundance. The first respectively most significant byte of every message specifies the mode.

In Table 6 the message protocol for the control of the stepper motor is illustrated. The first mode is used to adjust the bar to a position. The position lies between 0 and 100%, while 0% is the lowest and 100% the highest position. The speed can be chosen from full, half, quarter and eigthth step mode.

The next two modes are used to move the bar up and down. Here, the step number is passed and the applied step mode is half step by default. The last instruction allows to calibrate the bar to a consistent zero position.

MSB	**1byte**	**1byte**
MOVE	Position	Speed
UP	Steps	Steps
DOWN	Steps	Steps
INIT	Calibrate	

Table 6: Messages for the Stepper Motor

The messages for the control of the led display is shown in Table 7.

The first instruction is used to set a single led. For this, the side, the led number and the color have to be specified. Furthermore, a led can be disabled and a side can be cleared. The operation field allows to choose between these three functionalities.

An entire level respectively all leds, which are on the same height, can be set with the next message.

The animation instruction distinguishes between light pattern for notifications or animations, which run on the entire bar. This is done with the type-field. The duration of one animation cycle in milliseconds is specified with the speed parameter. When set to overall animation, the side and led number, are not considered.

The last message allows to modify the brightness of the bar. A value of 0 deactivates the lighting, whereas a value 255 represents the maximum brightness.

MSB	**1byte**	**3byte**	**1byte**	**1byte**	**1byte**
LIGHT	Side, Operation, Led	Color			
LEVEL	Ledlevel	Color			
ANI	Type, Animation	Color	Speed	Speed	Side, Led
BRIGHT-NESS	Brightness				

Table 7: Messages for the Led Display

5.2 PCB BOARD

A printed circuit board (PCB) is used to hold following components:

- stepper motor driver carrier A4988
- step-down voltage regulator
- DC power socket
- 2x10 pin connector (to connect the MovingBar)
- Atmega328p chip
- a quartz-oscillator with a frequency of 16 MHz
- 1x 4,7kΩ and 1x 2,2kΩ resistor (for SPI transfer)
- 2x220Ω resistors (for buttons)
- 2x 22pF, 1x100nF and 2x100µF condensator
- 1x10kΩ and 1x1kΩ
- 1x led light
- various pin connectors (2x6 pin, 1x5 pin, 1x2 pin)

The layout of the PCB board is displayed in Figure 39. Since the board will later be mounted in a 3D printed encasing, the goal was to make it as small as possible. Therefore a two layer circuit board is designed with vias to connect the top with the bottom side. The meaning of the different labels is explained in Table 8. A complete logic circuit of the board can be found in the appendix (Figure 56).

The dimensions of the board are 46mm in width and 78mm in length. In Figure 39 the conducting paths and the attachment contact fields are illustrated in red ink for the top and blue ink for the bottom layer. The power lines for 12V and 5V are broader, approximately 1mm, because they carry more current. At the top right the power jack with its three contacts for ground and 12V is drawn. The 12V go directly to the step-down voltage regulator and the A4988 stepper motor controller.

Then the voltage regulator provides multiple components with 5V. First, the A4988, to power its logical circuit, and the 2x10 MovingBar connector (at the top left) for the led stripes. Next, the Raspberry Pi is powered over the SPI pin adaptor (at the top in the middle). Finally, the AVR progamming port and the Atmega328p are also connected to the 5V power circuit.

The A4988 controls the stepper motor with four conducting paths (adjacent lines at the top in the middle) over the 2x10 MovingBar connector. Since the stepper motor consumes approximately 10.8V with 0.4A/Phase, these paths are also draft broader.

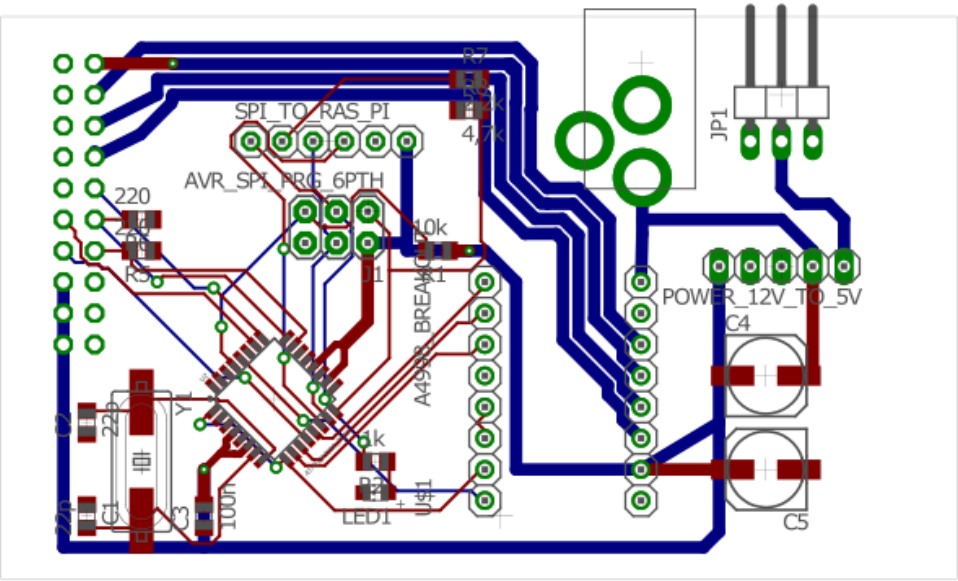


Figure 39: PCB top and bottom layer

A slide switch is connected with the 3pin header in order to power off the system. When the slide switch is closed, the enable pin of the power regulator is pulled to ground. This puts the power regulator in a sleep mode and disables the 5V output voltage. Various capacitors are built in to stabilize the power supply and avoid spikes.

A led beside the Atmega328p (at the bottom left) shows activity on the SPI interface respectively SPI clock connection. The voltage divider (see 5.1.1, Figure 37) uses a 4,7kΩ and a 2,2kΩ resistor.

Notation/Label	Description
POWER_12V_TO_5V	Step-down voltage regulator
A4988_BREAKOUT	Stack-header for the A4988
AVR_SPI_PRG_6PTH	6-pin AVR programming header
SPI_TO_RAS_PI	6-pin SPI interface header
JP1	3-pin header to power system off
Y1	Quartz-oscillator

Table 8: Meaning of the notations

The Atmega328p registers signals from two buttons, one infrared sensor and one identification chip, which are build into the MovingBar. Furthermore, it controls the led stripes.

Figure 40 shows the 2x10 pin header (at the top left in Figure 39), which is used to plug in the MovingBar. Button 1 (BTN1) represents a contact button, which is mounted at the bottom of the bar. It is used to calibrate the bar respectively setting the null position. Button 2 (BTN2) is not used yet. The signal of the infrared sensor is mapped onto IR, the identification chip signal onto ID. The data line of the led stripes is mapped onto the LED pin. B1, B2, A1 and A2 address the two coils of the stepper motor.

In Figure 41 the pin layout of the SPI interface is illustrated. As mentioned above, the 5V pin is used to power the connected Raspberry Pi. The GND line allows the Atmega328p and the Raspberry Pi to share the same ground potential. The remaining pins represent the four connections, which are necessary for the SPI data transfer (see description of SPI, section 4.1.1).

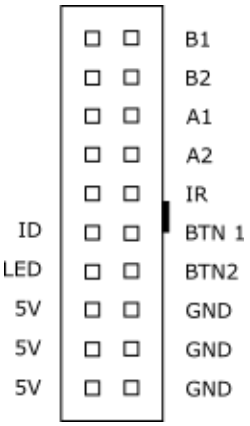


Figure 40: 2x10 pin header

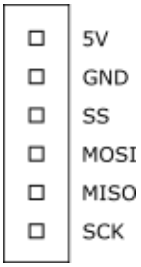


Figure 41: 6 pin SPI header

The Atmega328p's flash and EEPROM memory can be read and be written to over the data lines MISO, MOSI and SCK. Since the SPI connection also uses these pins, the 6pin SPI header should be disconnected or the Raspberry Pi should be turned off. When the RESET pin of the header is pulled low, the microcontroller doesn't treat the data lines as I/O signals (like SPI), but for the purpose of in-system programming.

In order to flash programs on the Atmega328p microcontroller, the interface in Figure 42 is required. This pin layout is called In-System-Programming (ISP).

Figure 42: AVR programming header

For the purpose of programming the microcontroller, the mySmartUSB light programmer by myavr is used.

5.3 SOFTWARE SETUP OF THE RASPBERRY PI

The Raspberry Pi holds all information to display scenarios and animations on the MovingBar. Four scenarios can be displayed simultaneously. Each scenario and its illustration are defined completely with following information:

- Side identifier
- Current (number-) value
- Display color
- Reference led color
- Reference value
- Step size value
- Request web URL to fetch structured data (XML or JSON)
- Path to the value in the XML or JSON file, which should be displayed

The side identifier corresponds to the side of the MovingBar and is used to specify either the front, left, right or back side. The display color is used for the quantitative data representation. Whereas the reference color represents the color of the reference led, described in the visualisation concept chapter (see 3.3). The step size and the reference value are used to compute the height respectively led numbers, which represent the current value. For this, the following formula is used:

$$led_{numbers} = \frac{currentValue - referenceValue}{stepSize}$$

The request URL, as mentioned above, is used to periodically load either XML or JSON data files. These file types consist of items, which form a hierarchical structure. With the path, which is a list of items, the appropriate value can be extracted from the structured file.

Besides the Raspberry Pi also saves following settings, which are relevant for the entire MovingBar:

- Sample time duration between pull requests of scenarios
- Brightness value between 0-100%

When running, the four scenarios are constantly updated with the frequency specified by the sample time. Therefore, a timer thread[1] periodically calls a function (see Listing 7), which is used to request data from a web interface and then displays them. The time delay respectively sample time is preset to one minute, but can also be defined by the user. The scenario control data for each scenario is stored in a list "displays" with the length of four. The entries of the list correspond to the front, left, back and right side of the MovingBar. A new scenario can be inserted into the list with the method in line 23.

```
1  import threading
2
3
4  class displayHandler():
5
6      displays=[None]*4
7
8      def __init__(self,delay):
9          threading.Thread.__init__(self)
10         self.delay=delay    #delay in seconds
11
12
13     def displayData(self):
14         print("display data")
15         for i in range(0,4):
16             if(self.displays[i] is not None):
17                 self.displays[i].display()
18
19         nextCall =threading.Timer(self.delay,self.displayData)
20         nextCall.start()
21
22
23    def addNewValueObject(self,newValueObject,index):
24         self.displays[index]= newValueObject
```

Listing 7: Display handler

Since the scenarios should be set by the user with a graphical user interface, a reliable connection is needed for transferring all necessary information to the Raspberry Pi. For this, a TCP server-client architecture is used. The Raspberry Pi is the TCP server and constantly waiting for an incoming client. A socket is used for reading incoming data. The server is set to a static ip address and port number. The GUI is the TCP client and can send messages to the server. For this, the server and client have to share the same local network.

[1] Raspberry Pi source code can be found on Github. https://git.hci.uni-hannover.de/ChristianA/bachelor-thesis/tree/master/scenarioHandler, visited 30-01-2018

5.3.1 Message Protocol

In Listing 8 the used message protocol of the TCP connection is specified. The messages are sent in plaintext.

The first instruction specifies all necessary parameters to set a scenario. The colors are transferred as hexadecimal representations. The reference and stepsize field are interpreted by the parser method as integers. When a scenario message is parsed, the corresponding side will be reset, and the new scenario will be displayed with the next display cycle.

In order to be able to delete the scenario on a side, the reset instruction is specified.

The sample duration and the brightness of the MovingBar can be set with the next command. The sample duration value represents the update frequency of all current displayed scenarios. The brightness should lie between 0 and 255, for the minimum and maximum value.

Then, the status instruction allows the graphical user interface to request the error and output messages of the running system. On the Raspberry Pi these messages are constantly written to corresponding files. When the status is requested, the files are sent followed by a special character, which specifies the end of data transfer.

The last instruction allows a safe shutdown of the Raspberry Pi.

```
1  DISPLAY:SIDE//DIPLAYCOLOR//REFCOLOR//REFERENCEVALUE//STEPSIZE//
2      MODE//URL//PATHXML//PATHJSON
3  RESETSIDE:SIDE
4  SET:SAMPLEDURATION//BRIGHTNESS
5  STATUS:
6  SHUTDOWNSYSTEM:
```

Listing 8: Message protocol TCP

5.3.2 Initial Deployment

Since the Raspberry Pi should be reachable in a local network via Wifi, an intial configuration has to be done. This includes setting up the Wifi credentials and readout the assigned ip address. In the current system, this has to be done manually by login into the Raspberry Pi via SSH. This has to be done once, when the system is setup in a new location.

5.4 GRAPHICAL USER INTERFACE

The Graphical User Interface (GUI) allows the user to setup and control the MovingBar. As mentioned before, the scenarios and the illustration preferences are necessary informations. With different editors and views, the user can adjust these settings.

When the program[1] is started, the first window shows the options to control each side of the MovingBar independently (see Figure 43). The input fields are used to specify the scenario and its illustration through light and height adjustment. These input options correspond to the names in the previous chapter (see 5.3). The modes can be selected between three options: from bottom to top, from bottom to top with moving height and relative. Default scenarios, which display the temperature or the rainfall in Hanover, can be chosen.

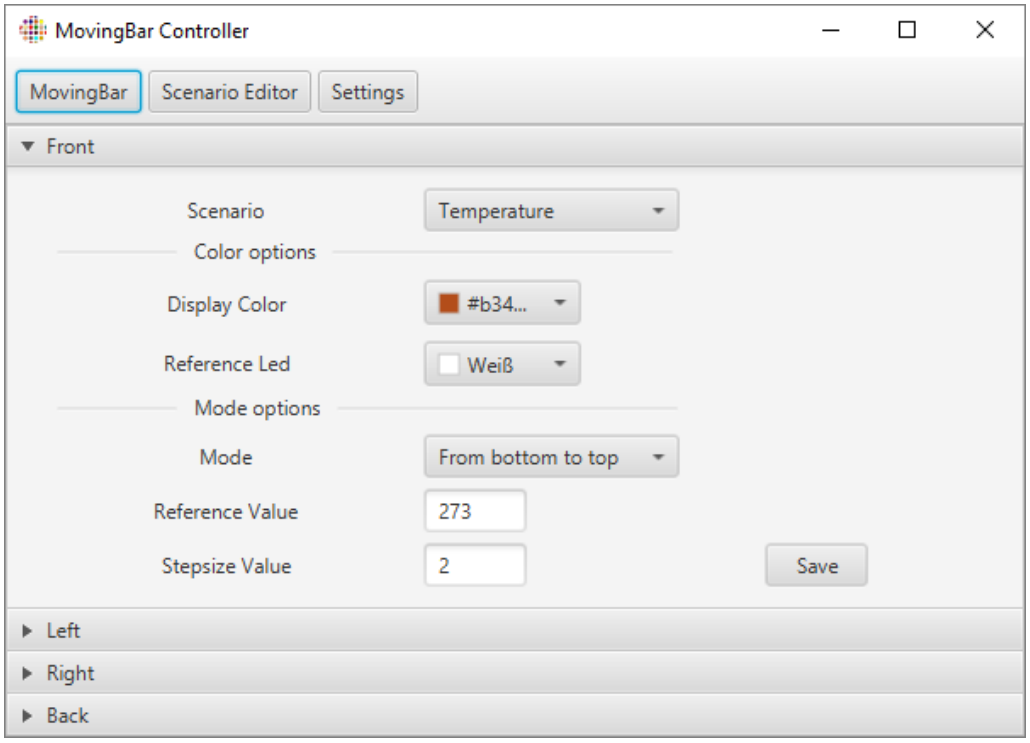


Figure 43: Control each side independly

[1] Graphical User Interface source code can be found on Github. https://git.hci.uni-hannover.de/ChristianA/bachelor-thesis/tree/master/MovingBarGUI

Figure 44 shows the "Scenario Editor", which is used to maintain all created scenarios. A temperature and a rainfall scenario for the city of Hanover is saved by default.

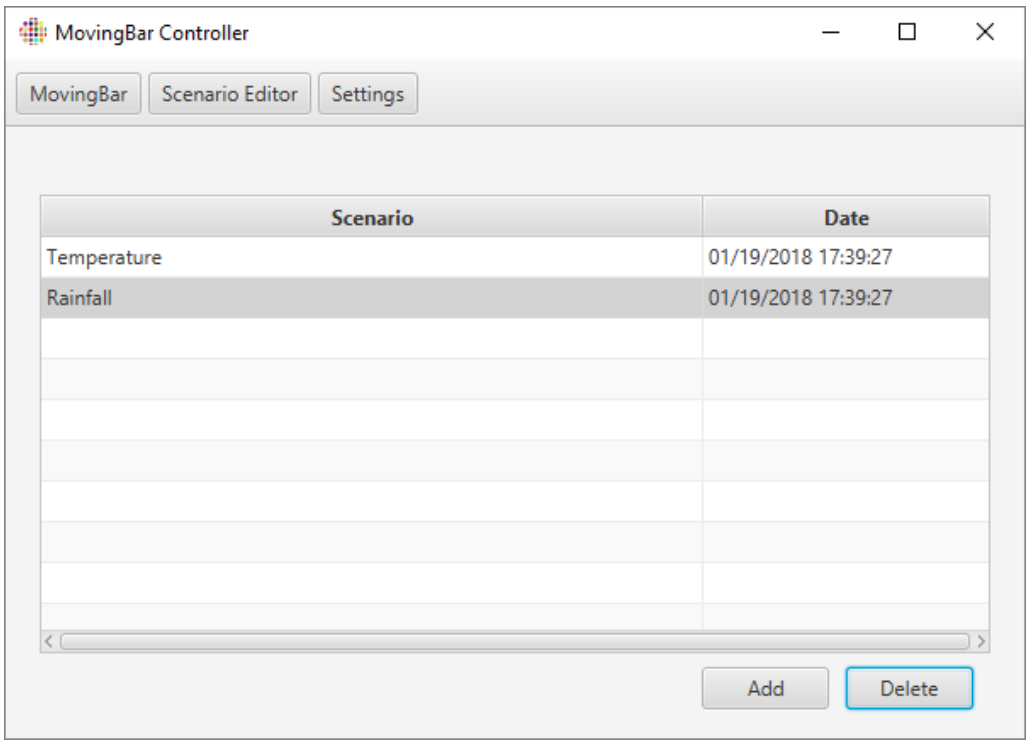


Figure 44: Maintain scenarios

With selecting the add button, the "Scenario Builder" opens (see Figure 45). In this window, the user can enter the request url to the data file and the type of the file (XML or JSON). A hierachical structure view of the file is loaded, by pressing the "Display" button. Here the user can select the field, which should be displayed. This field should either contain an integer or decimal value. With right click on the entry, the option for choosing this field appears. Once a name for the scenario is entered, the "Save" button saves the scenario.

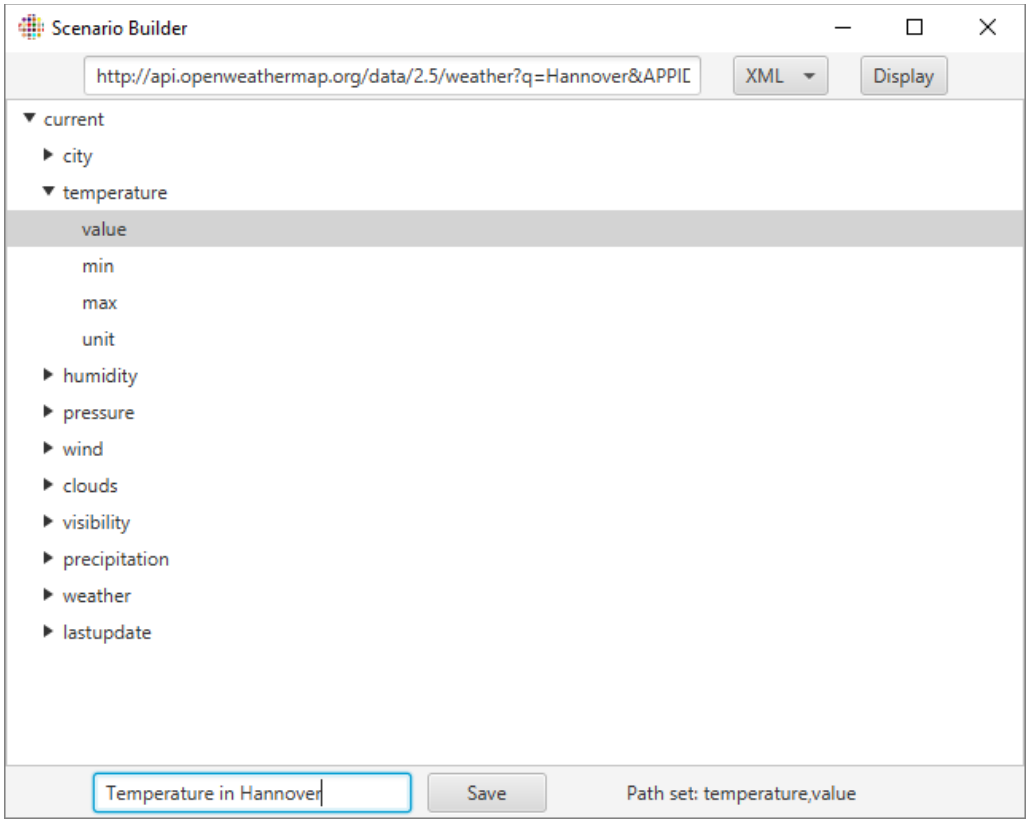


Figure 45: Scenario Editor

Finally, the setting dialog in Figure 46 allows to specify the tcp network parameters and the settings for the entire bar. For the tcp connection the ip address and the port number of the Raspberry Pi have to be entered. With the "Select" button, the connection will be established. Feedback messages, including errors, will be shown in the right corner. The status can be requested and consists of the standart output, followed by the error messages of the Raspberry Pi. Next, the brightness of the bar (0-255) and the sample time in seconds can be specified with the lower fields. The "Shutdown" button allows to safely shutdown the Raspberry Pi.

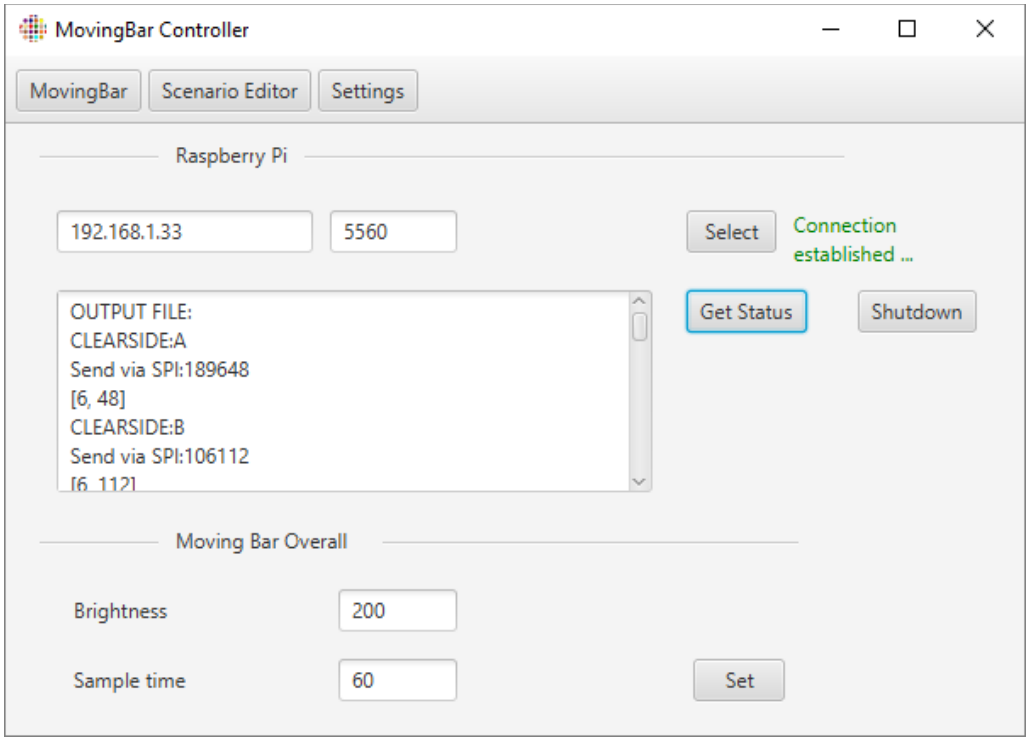


Figure 46: Connected to the Raspberry Pi and get status information

All settings will be saved in a database, when the GUI is closed. Therefore, scenarios or side settings don't have to be recreated.

5.5 ENCASINGS

Since one goal is to make the system more compact, an encasing for the control components and a mounting pedestal is fabricated.

5.5.1 System Controller

The PCB board and the Raspberry Pi are hold in a printed encasing. The model of the top and bottom part of the encasing[1] is shown in

Figure 47 and Figure 48. Multiple holes in the side of the encasing allow to plug in the power supply and connect the MovingBar adaptor. Besides there is a hole for the power switch. On the top, a ventilation hole is recessed to prevent overheating of the Raspberry Pi and the components of the PCB board.

The PCB board can be attached with screws in the bottom part of the encasing. Whereas the Raspberry Pi can be mounted in the top part of the encasing.

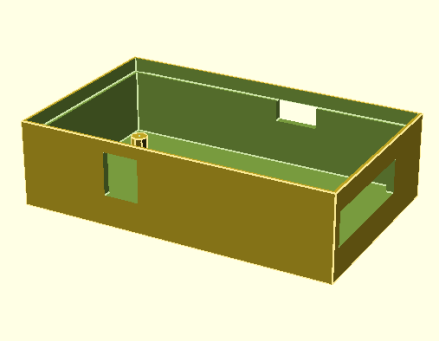


Figure 47: Bottom part of the encasing

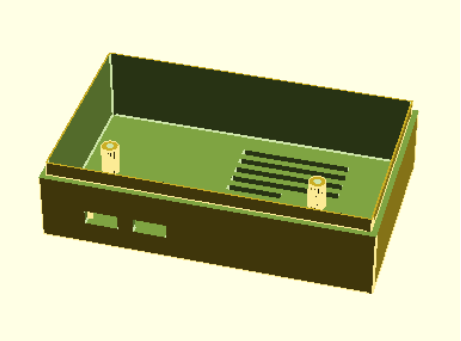


Figure 48: Top part of the encasing

[1] Model files (.stl and .scad) can be found on Github. https://git.hci.uni-hannover.de/ChristianA/bachelor-thesis/tree/master/models, visited 30-01-2018

5.5.2 Mounting Pedestal

The mounting pedestal (see Figure 49) holds the Moving Bar and ensures stability. The connection wires are led through a hole in the back. The Moving Bar is fixed with four nuts on the ground plane. The fabricated model is pictured in Figure 50.

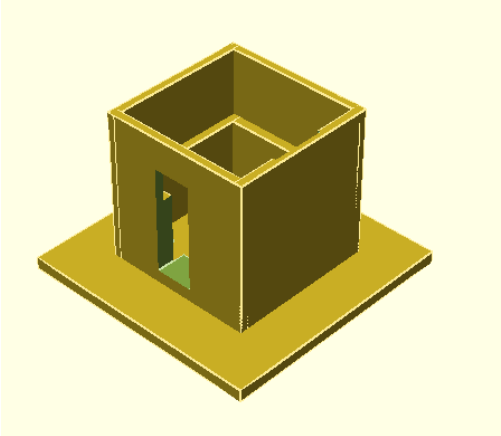


Figure 49: Mounting pedestal model

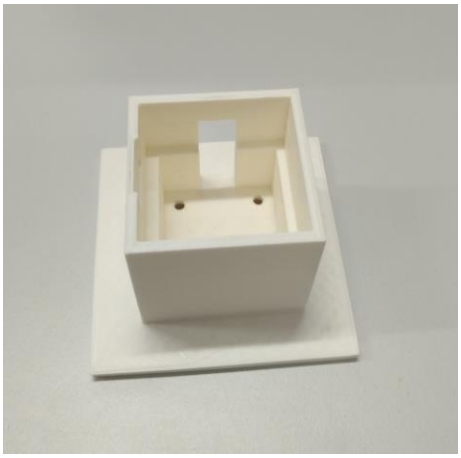


Figure 50: Fabricated model

6 CONCLUSION

In this chapter the results of this thesis will be presented briefly. Then an outlook on possible further developments will be given.

6.1 RESULTS

The developed system allows to control the shape-changing display MovingBar in a more simple and user-friendly manner, opposed to the previous system.

The previous system was, due to the size and complexity, mainly a prototype for a laboratory setup.

The new system includes the MovingBar, a small controller box and the wiring respectively power supply.

A system controller was developed, which, among others, consists of a Raspberry Pi and a microcontroller. The Raspberry Pi holds all necessary information, which represent the scenario and display configurations. It updates the lighting and height of the MovingBar with sending corresponding messages to the microcontroller. The connection is enabled by using the serial peripheral interface. The height and the lighting can now be controlled simultaneously. Multiple blink animations were implemented and are used for notification feedback.

Then, a visualisation concept was created, in order to display information in a readable manner. Here, among others, blink animations were applied to convey the status of the information. Furthermore, the status of the system is displayed with error or warning notifications. Appropriate colors, with reference to emotional effects, were chosen to illustrate these notifications.

A graphical user interface was developed, which is used to control the system. This allows to set the displayed information and to specify parameters like the led color or the display mode. The user can create scenarios, for example showing the temperature or rainfall of a location. The data is frequently fetched from the specified web source and displayed with the declared settings on the MovingBar.

In order to get user feedback, the developed system was set up in a frequently visited place in the human-computer-interaction institut for several days. The compact and easy setup was emphasised by the users. This also revealed, that the users find it difficult to remember the represented value of one led for every side. Also, the exact value cannot be represented, due to a limited number of leds.

However, the display is helpful for showing trends of information. When, for example, the user recognizes that there are more leds on, than previously, a rising trend is indicated. Concretely that can mean, that it is warmer or raining more than a couple of hours before. However, more usage of the MovingBar will very likely increase the accuracy of the display value interpretation.

6.2 OUTLOOK

In the future, the system could be enhanced with following functionalities.

The network addresses and the wifi connection have to be specified manually for the respective setup evironment. This can't be done by a normal user. Therefore, it would improve usability, when this settings could be specified in the graphical user interface. Maybe, in the initial setup the user could configure this settings, by accessing the Raspberry Pi via the USB-to-go port. This allows a SSH connection to setup wifi and read the assigned ip address.

Then, the readability of the MovingBar could be increased. This could be achieved by an independent brightness setting for every led. This enables a greater state space of each led and allows to visualise information on each led with different brightness levels.

Also further display modes can be developed. This could be a relative display mode with height adjustment. Therefore, the display range of the normal relative display mode, would be increased.

7

APPENDIX

7.1 RELATED WORK

7.1.1 Effects of Color on Emotions – Functions

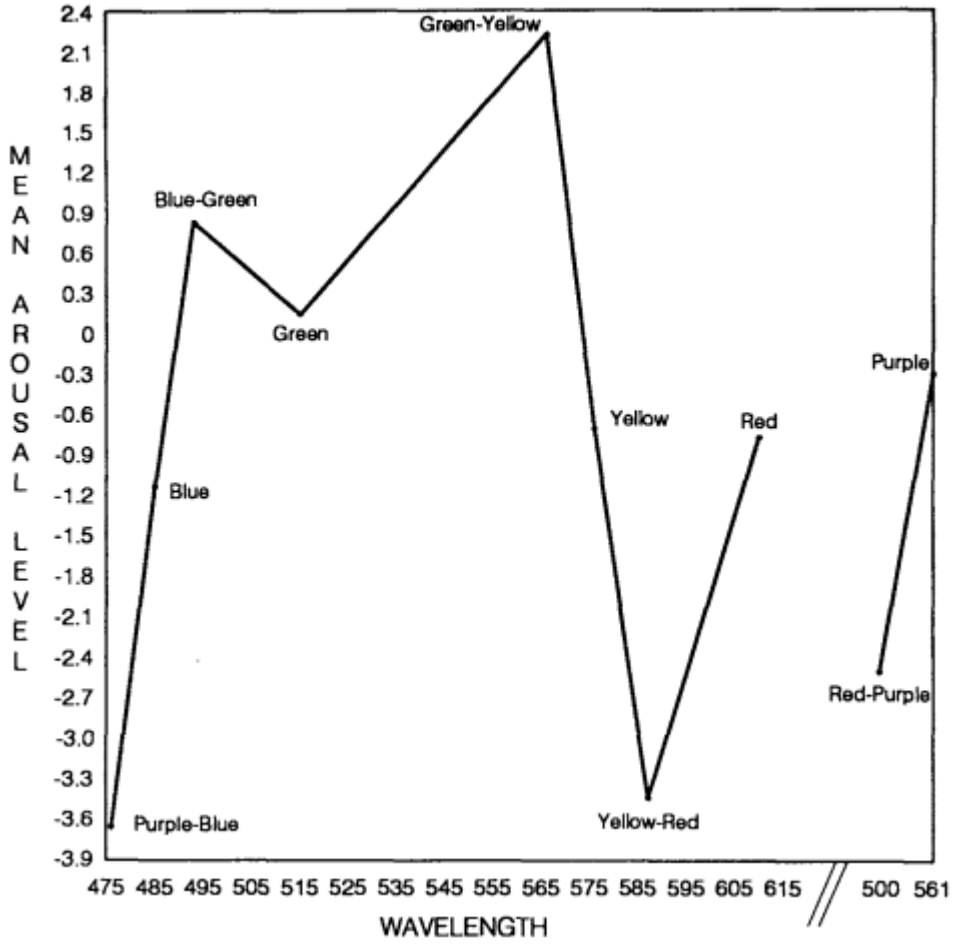


Figure 51: Mean arousal level plotted against wavelength (Valdez & Mehrabian, 1994)

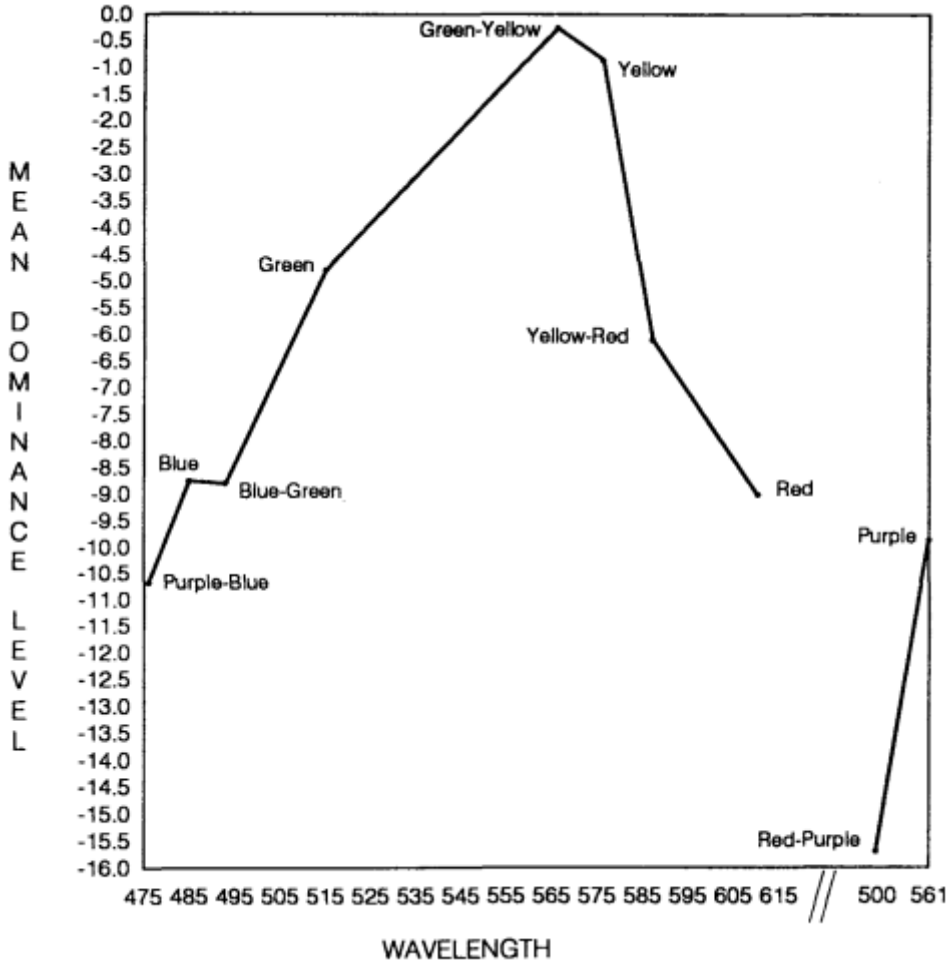


Figure 52: Mean dominance level plotted against wavelength (Valdez & Mehrabian, 1994)

7.1.2 Light Patterns – Figures

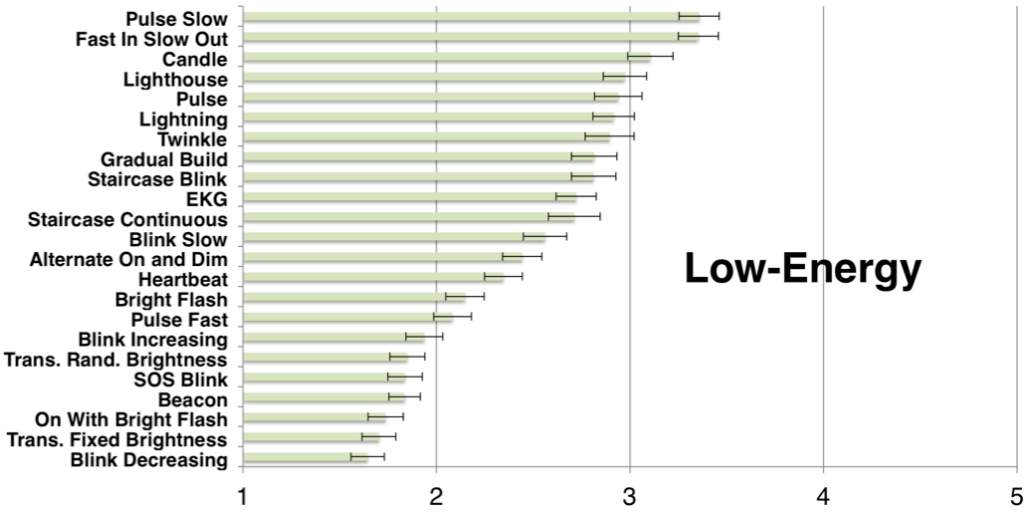


Figure 53: Ratings of light patterns for the low-energy state (Harrison et al., 2012)

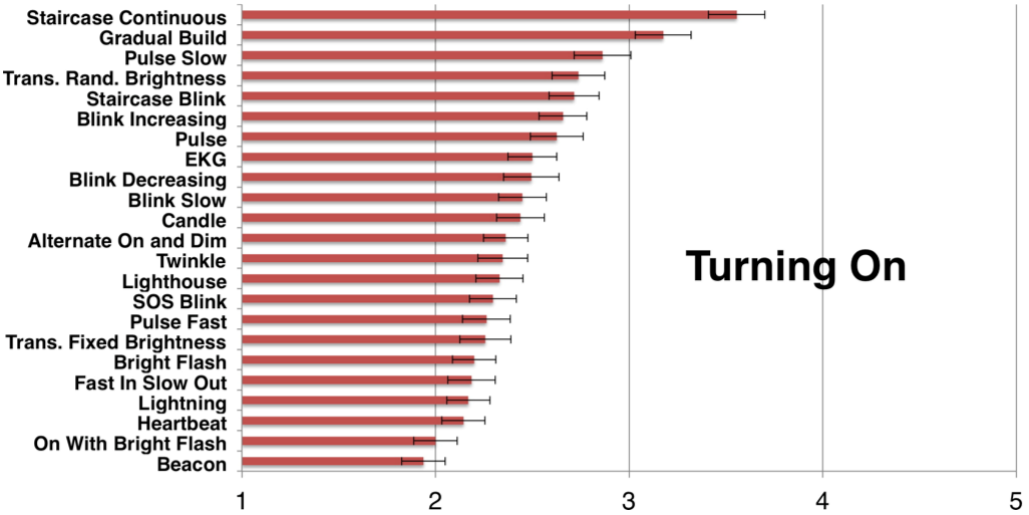


Figure 54: Ratings of light patterns for the turning on state (Harrison et al., 2012)

7.2 USED HARDWARE

7.2.1 ATmega328p 16-bit timer

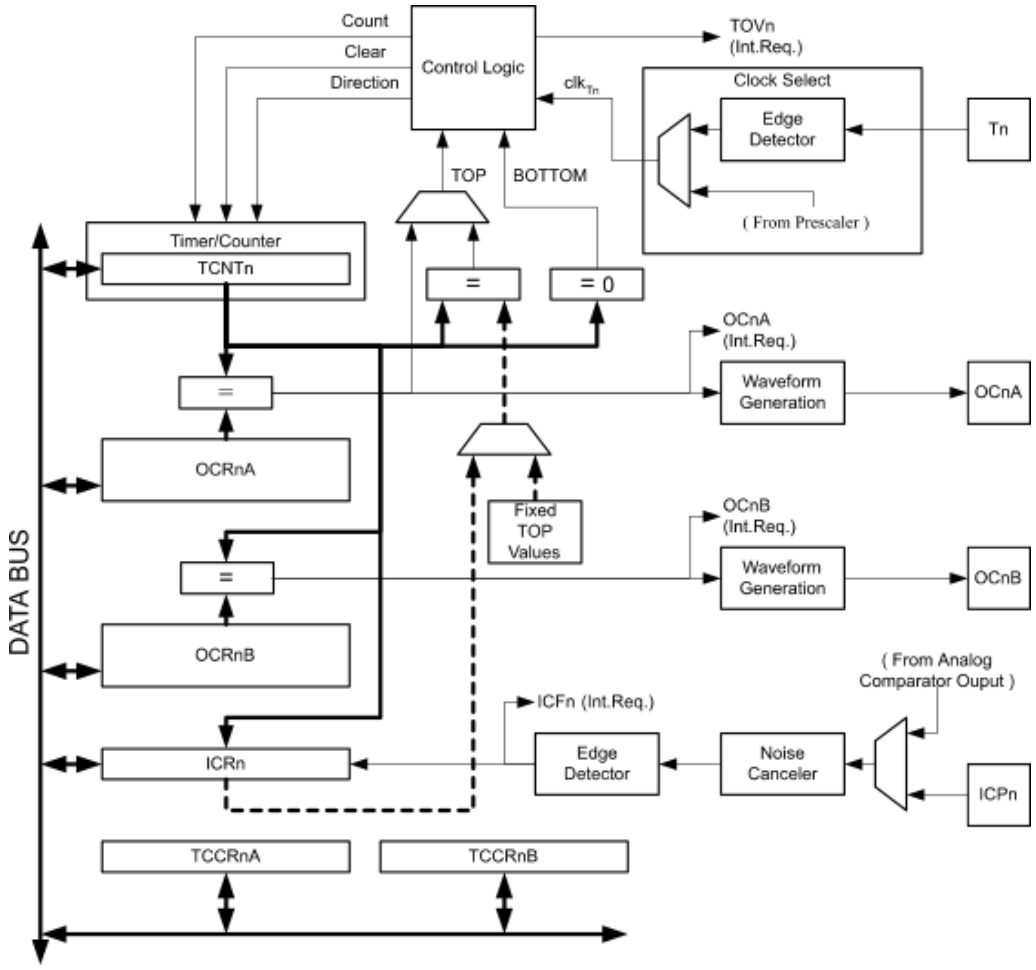


Figure 55: Schematic of the 16-bit timer
(ATmega328p Datasheet, 2016)

7.2.2 List of Internal and External Interrupts

Vector Number	Definition	Source
1	Reset	
2	External Interrupt Request 0 (pin D2)	(INT0_vect)
3	External Interrupt Request 1 (pin D3)	(INT1_vect)
4	Pin Change Interrupt Request 0 (pins D8 to D13)	(PCINT0_vect)
5	Pin Change Interrupt Request 1 (pins A0 to A5)	(PCINT1_vect)
6	Pin Change Interrupt Request 2 (pins D0 to D7)	(PCINT2_vect)
7	Watchdog Time-out Interrupt	(WDT_vect)
8	Timer/Counter2 Compare Match A	(TIMER2_COMPA_vect)
9	Timer/Counter2 Compare Match B	(TIMER2_COMPB_vect)
10	Timer/Counter2 Overflow	(TIMER2_OVF_vect)
11	Timer/Counter1 Capture Event	(TIMER1_CAPT_vect)
12	Timer/Counter1 Compare Match A	(TIMER1_COMPA_vect)
13	Timer/Counter1 Compare Match B	(TIMER1_COMPB_vect)
14	Timer/Counter1 Overflow	(TIMER1_OVF_vect)
15	Timer/Counter0 Compare Match A	(TIMER0_COMPA_vect)
16	Timer/Counter0 Compare Match B	(TIMER0_COMPB_vect)
17	Timer/Counter0 Overflow	(TIMER0_OVF_vect)
18	SPI Serial Transfer Complete	(SPI_STC_vect)
19	USART Rx Complete	(USART_RX_vect)
20	USART, Data Register Empty	(USART_UDRE_vect)
21	USART, Tx Complete	(USART_TX_vect)
22	ADC Conversion Complete	(ADC_vect)
23	EEPROM Ready	(EE_READY_vect)
24	Analog Comparator	(ANALOG_COMP_vect)
25	2-wire Serial Interface(I2C)	(TWI_vect)
26	Store Program Memory Ready	(SPM_READY_vect)

Table 9: Complete interrupt list
of the ATmega328p microcontroller (Gammon, 2012)

7.3 DEVELOPMENT & IMPLEMENTATION

7.3.1 PCB Board

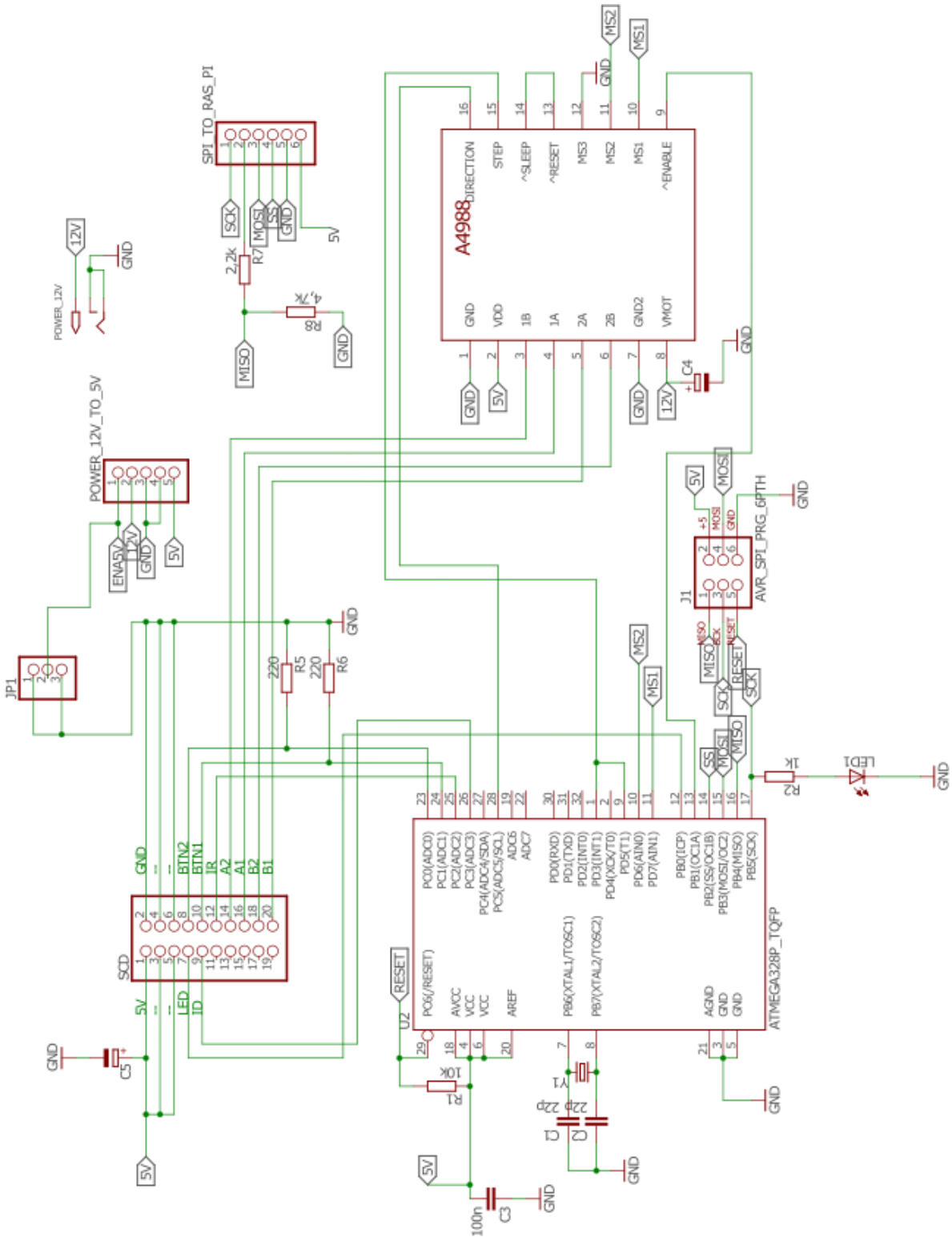


Figure 56: Complete logic circuit of the PCB board

LIST OF TABLES

LIST OF FIGURES

BIBLIOGRAPHY

Arduino LLC. (2017). Arduino LLC. Retrieved from https://www.arduino.cc/en/Reference/SPI

ATmega328p Datasheet. (2016). Retrieved January 29, 2018, from http://www.atmel.com/Images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P_datasheet_Complete.pdf

Coelho, M., Ishii, H., & Maes, P. (2008). Surflex: a programmable surface for the design of tangible interfaces. *CHI '08 Extended Abstracts on Human Factors in Computing Systems*, 3429–3434. https://doi.org/http://doi.acm.org/10.1145/1358628.1358869

Coelho, M., & Zigelbaum, J. (2011). Shape-changing interfaces. *Personal and Ubiquitous Computing*, *15*(2), 161–173. https://doi.org/10.1007/s00779-010-0311-y

Dahley, A., Wisneski, C., & Ishii, H. (1998). Water Lamp and Pinwheels : Ambient Projection Digital Information into Architectural Space. *CHI 98 Conference Summary on Human Factors in Computing Systems*, (April), 269–270. https://doi.org/10.1145/286498.286750

Follmer, S., Leithinger, D., Olwal, A., Hogge, A., & Ishii, H. (2013). inFORM: Dynamic Physical Affordances and Constraints through Shape and Object Actuation. *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology - UIST '13*, 417–426. https://doi.org/10.1145/2501988.2502032

Gammon, N. (2012). Interrupt Description ATmega328p. Retrieved from https://gammon.com.au/interrupts

Goulthorpe, M. (2001). Aegis Hyposurface. Retrieved from https://mcburry.net/aegis-hyposurface/

Greyworld Artist Group. (2015). The Source. Retrieved from http://greyworld.org/archives/31

Hardy, J., Weichel, C., Taher, F., Vidler, J., & Alexander, J. (2015). ShapeClip: Towards Rapid Prototyping with Shape-Changing Displays for Designers. *Proceedings of the ACM CHI'15 Conference on Human Factors in Computing Systems*, *1*, 19–28. https://doi.org/10.1145/2702123.2702599

Harrison, C., Horstman, J., Hsieh, G., & Hudson, S. (2012). Unlocking the expressivity of point lights. *Proceedings of the 2012 ACM Annual Conference on Human Factors in Computing Systems - CHI '12*, 1683–1692. https://doi.org/10.1145/2207676.2208296

Harrison, C., & Hudson, S. E. (2009). Providing Dynamically Changeable Physical Buttons on a Visual Display. *Proceedings of the 27th International Conference on Human Factors in Computing Systems - CHI 09*, 299–308. https://doi.org/10.1145/1518701.1518749

Hemmert, F., Hamann, S., Laboratories, D. T., Zeipelt, J., & Joost, G. (2010a). Shape-Changing Mobiles : Tapering in Two-Dimensional Deformational Displays in Mobile Phones. *Proceedings of the 28th of the International Conference Extended Abstracts on Human Factors in Computing Systems CHI EA 10*, 3075–3079. https://doi.org/http://doi.acm.org/10.1145/1753846.1753920

Hemmert, F., Hamann, S., Laboratories, D. T., Zeipelt, J., & Joost, G. (2010b). Weight-Shifting Mobiles : Two-Dimensional Gravitational Displays in Mobile Phones. *Proc. CHI*, 3087–3091. https://doi.org/10.1145/1753846.1753922

Ishii, H., Ren, S., & Frei, P. (2001). Pinwheels: visualizing information flow in an architectural space. *CHI '01 Extended Abstracts on Human Factors in Computing Systems*, 111–112. https://doi.org/10.1145/634067.634135

Iwata, H., Yano, H., Nakaizumi, F., & Kawamura, R. (2001). Project FEELEX: Adding Haptic Surface to Graphics. *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, (1), 469–476. https://doi.org/10.1145/383259.383314

Kim, H., & Lee, W. (2008). Shade Pixel. *SIGGRAPH Posters*, 34. https://doi.org/10.1145/1400885.1400922

Kodama, S., & Takeno, M. (2001). Sound-responsive magnetic fluid display. *Human-Computer Interaction - Interact'01*, (Figure 2), 737–738.

Lehnert, A. (2016). *A Shape-Changing Display for Ambient Notifications*. Leibniz Universität Hannover.

Poupyrev, I., Nashida, T., Maruyama, S., Rekimoto, J., & Yamaji, Y. (2004). Lumen. *ACM SIGGRAPH 2004 Emerging Technologies on - SIGGRAPH '04*, 17. https://doi.org/10.1145/1186155.1186173

Poupyrev, I., Nashida, T., & Okabe, M. (2007). Actuation and tangible user interfaces: the Vaucanson duck, robots, and shape displays. *Proceedings of TEI 2007*, 205–212. https://doi.org/10.1145/1226969.1227012

Rasmussen, M. K., Pedersen, E. W., Petersen, M. G., & Hornbæk, K. (2012). Shape-Changing Interfaces: A Review of the Design Space and Open Research Questions. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*, 735–744. https://doi.org/10.1145/2207676.2207781

Stepper Motor Driver Board User Manual. (n.d.). Retrieved from https://www.pololu.com/product/1182

Valdez, P., & Mehrabian, A. (1994). Effects of color on emotions. *Journal of Experimental Psychology: General*, *123*(4), 394–409. https://doi.org/10.1037/0096-3445.123.4.394

Zigelbaum, J., Chang, A., Gouldstone, J., Monzen, J. J., & Ishii, H. (2008). SpeakCup : Simplicity , BABL , and Shape Change. *Framework*, 145–146. https://doi.org/10.1145/1347390.1347422

EIDESSTATTLICHE ERKLÄRUNG

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt habe. Ich habe alle Stellen, die ich aus den Quellen wörtlich oder inhaltlich entnommen habe, als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Unterschrift:

Hannover, den 5. Februar 2018