Medienstation: Dresden - Utopia - Dystopia

Technische Dokumentation

Technische Universität Dresden

Fakultät Informatik
Institut für Software- und Multimediatechnik
Professur für Mediengestaltung

Ansprechpartner/-in:
Laura Dietrich - laura.dietrich@mailbox.tu-dresden.de
Felix Goebel - felix.goebel1@mailbox.tu-dresden.de

Datum:
10. Juni 2024

Inhaltsverzeichnis

1 Einleitung **1**

2 Aufbau der Medienstation **3**
2.1 Tastenfeld 3
2.2 Rechner 4
2.3 Monitor 4

3 Installation und Inbetriebnahme **5**

4 Bedienungsanleitung **7**

5 Technische Spezifikationen **9**
5.1 Tastenfeld Bauteile 9
5.2 Tastenfeld Schaltplan 9
5.3 Arduino Sketch 10
5.4 Applikation der Medienstation 12

6 Fehlersuche und Problemlösung **21**

Abbildungsverzeichnis **i**

Literatur **iii**

1 Einleitung

Im Rahmen mehrerer Veranstaltungen der Professur für Mediengestaltung entstand in Kooperation mit den *Staatlichen Kunstsammlungen Dresden* für die *Kinderbiennale* 2024 die Medienstation *Dresden - Utopia - Dystopia* [SKD24]. Diese Dokumentation enthält Hinweise zum technischen Aufbau, zur Bedienung der Station sowie zu Wartungsmöglichkeiten und generellen Informationen zur Fehlerbehebung.

Abbildung 1.1 – KI Medieninstallation: Dresden - Utopia - Dystopia

2 Aufbau der Medienstation

Die Medienstation setzt sich im Wesentlichen aus drei Hardwarekomponenten zusammen: Tastenfeld, Rechner und Monitor. Alle Elemente sind in einer Staffelei integriert.

2.1 Tastenfeld

Das Tastenfeld besteht aus einem äußeren und einem inneren Gehäuseteil, welche ineinander greifen. Abbildung 2.1 zeigt die sichtbare Außenseite. Die serielle Datenübertragung vom Tastenfeld zum Rechner erfolgt über ein USB-Kabel.

Anzeige- und Bedienelemente

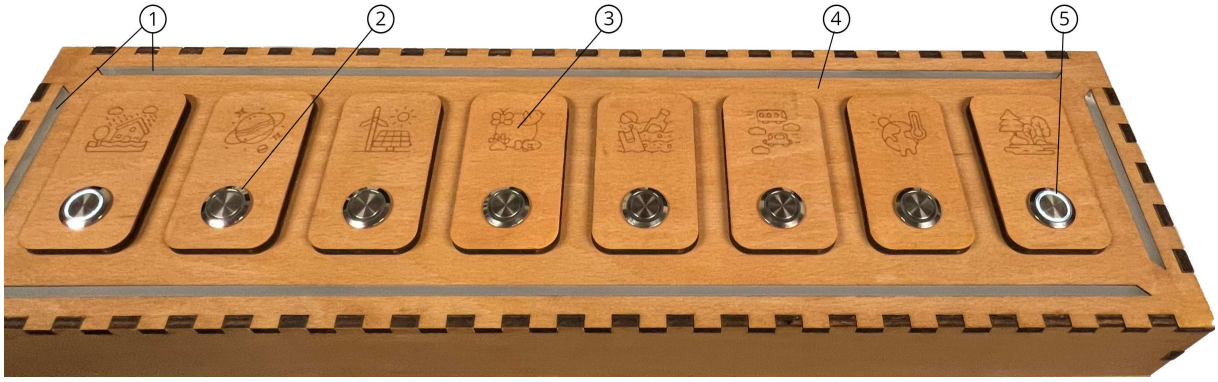

Abbildung 2.1 – Tastenfeld: Legende

1. LED Streifen
2. Taster
3. Piktogramm
4. Äußeres Gehäuse
5. LED Taster

Die LED eines Tasters wird aktiviert, wenn der entsprechende Taster gedrückt wurde. Allgemeine Hinweise zur Bedienung finden sich im Kapitel 4.

Skizze und Maße

Das Tastenfeld ist 160 mm hoch, 576 mm breit und 70 mm tief. Der Kabelausgang befindet sich mittig auf der Gehäuserückseite (vgl. Abbildung 2.2).

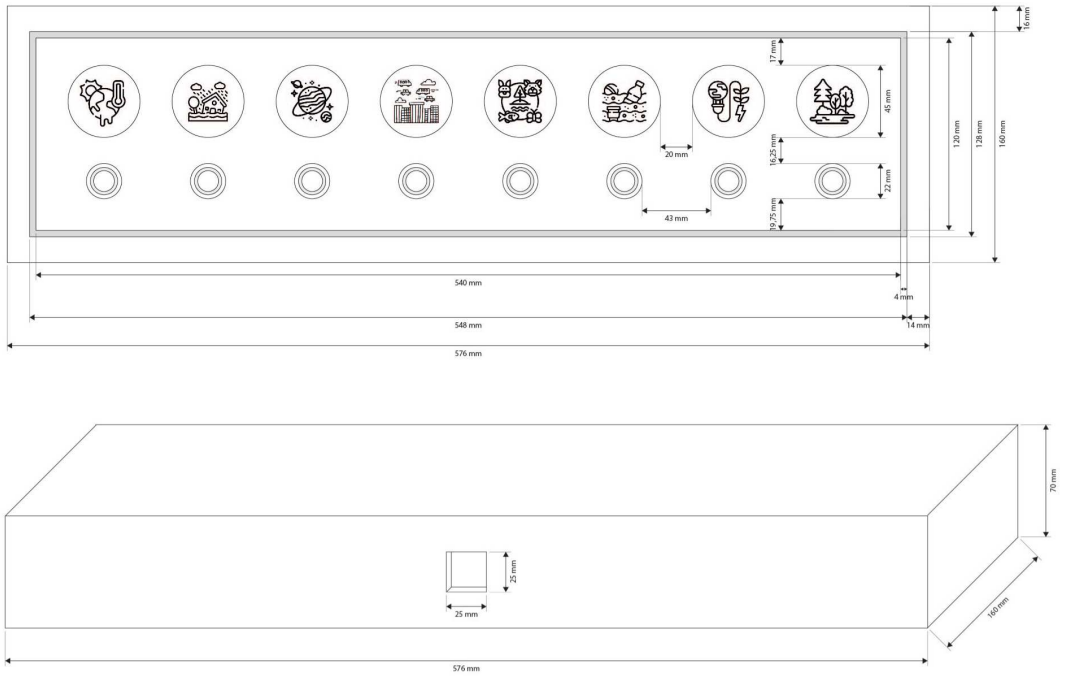

Abbildung 2.2 – Tastenfeld: Skizze

2.2 Rechner

Der verwendete Rechner ist von außen nicht sichtbar, sondern befindet sich in einer zugänglichen schwarzen Aufbewahrungsbox nahe der Staffelei (vgl. Abbildung 1.1). Dieser PC und die darauf laufende Anwendung bilden die Schnittstelle zwischen Monitor und Tastenfeld. Die getätigten Eingaben werden über die serielle Schnittstelle an die Applikation gesendet. Rechner und Tastenfeld können auch unabhängig voneinander ohne serielle Verbindung betrieben werden, dann erfolgt jedoch keine weitere Reaktion außer der optischen Rückmeldung durch die LEDs.

2.3 Monitor

Der Monitor dient lediglich zur visuellen Darstellung der Inhalte der Medienstation. Für den Anschluss an den Rechner ist ein DisplayPort-Kabel erforderlich.

3 Installation und Inbetriebnahme

Bevor die Medienstation in Betrieb genommen werden kann, muss sichergestellt sein, dass das Tastenfeld über ein USB-Kabel mit dem Rechner verbunden und über das mitgelieferte Netzteil mit Strom versorgt wird. Im Anschluss kann der Rechner durch Drücken der *Power* Taste gestartet werden. Ein Skript öffnet nach dem Start von Windows automatisch die Applikation der Medienstation im Vollbildmodus. In diesem Zustand ist auf dem Monitor ein Gemälde von Bernardo Belotto zu erkennen und der LED Streifen des Tastenfeldes leuchtet dauerhaft (vgl. Abbildung 3.1). Die

Abbildung 3.1 – *Dresden vom rechten Elbufer unterhalb der Augustusbrücke*, Bernardo Bellotto, 1748

Medienstation kann jetzt getestet werden. Die generelle Vorgehensweise ist im nachfolgenden Kapitel beschrieben. Sollte das Ergebnis nicht dem zuvor beschriebenen Zustand entsprechen oder ein Test fehlschlagen, gibt das Kapitel 6 Hinweise zur Fehlersuche und Lösungsvorschläge.

4 Bedienungsanleitung

Auf dem Tastenfeld sind acht Piktogramme eingraviert, welche ein konkretes Thema darstellen: Überflutung, Weltraum, Erneuerbare Energien, Tiere, Umweltverschmutzung, Mobilität der Zukunft, Klimaerwärmung und Rückeroberung der Natur (von links nach rechts). Die Auswahl erfolgt durch Drücken des darunterliegenden Tasters. (vgl. Abbildung 4.1)

Abbildung 4.1 – Piktogramme

Um die Aufmerksamkeit zu Beginn auf das Tastenfeld zu lenken, leuchtet der LED-Streifen des Tastenfeldes permanent. Eine erfolgreiche Auswahl wird durch die aktive LED des Tasters angezeigt. Danach signalisiert eine LED-Animation der übrigen Taster, dass eine weitere Auswahl getroffen werden kann. Sobald der zweite Taster gedrückt wird, erlischt der LED-Streifen, um die Aufmerksamkeit vom Tastenfeld wegzulenken.

Anschließend kann das gewählte Szenario auf dem Monitor betrachtet werden. Nach 20 Sekunden kehrt die Medienstation automatisch in den Ausgangszustand zurück und eine neue Auswahl kann erfolgen. Zusätzlich kann der Ablauf anhand des Sequenz- beziehungsweise Zustandsdiagramms nachvollzogen werden. (vgl. 4.2f.)

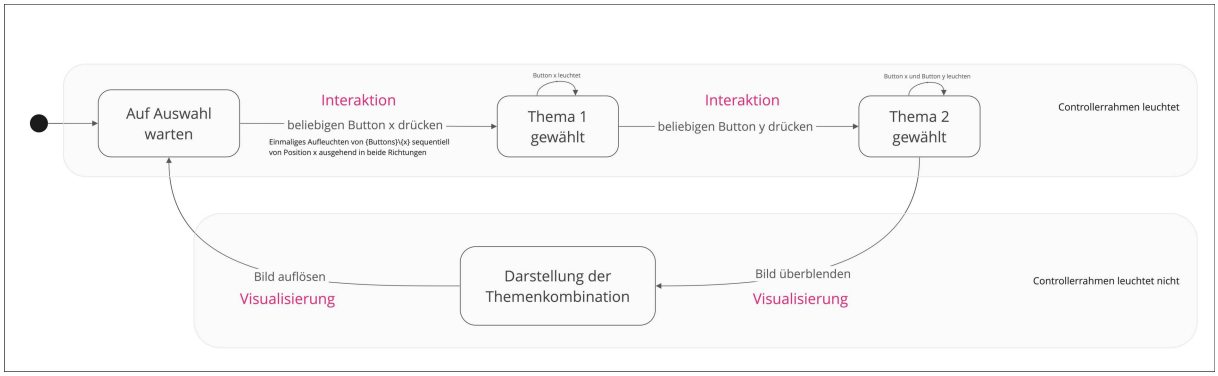

Abbildung 4.2 – Zustandsdiagramm der Medienstation

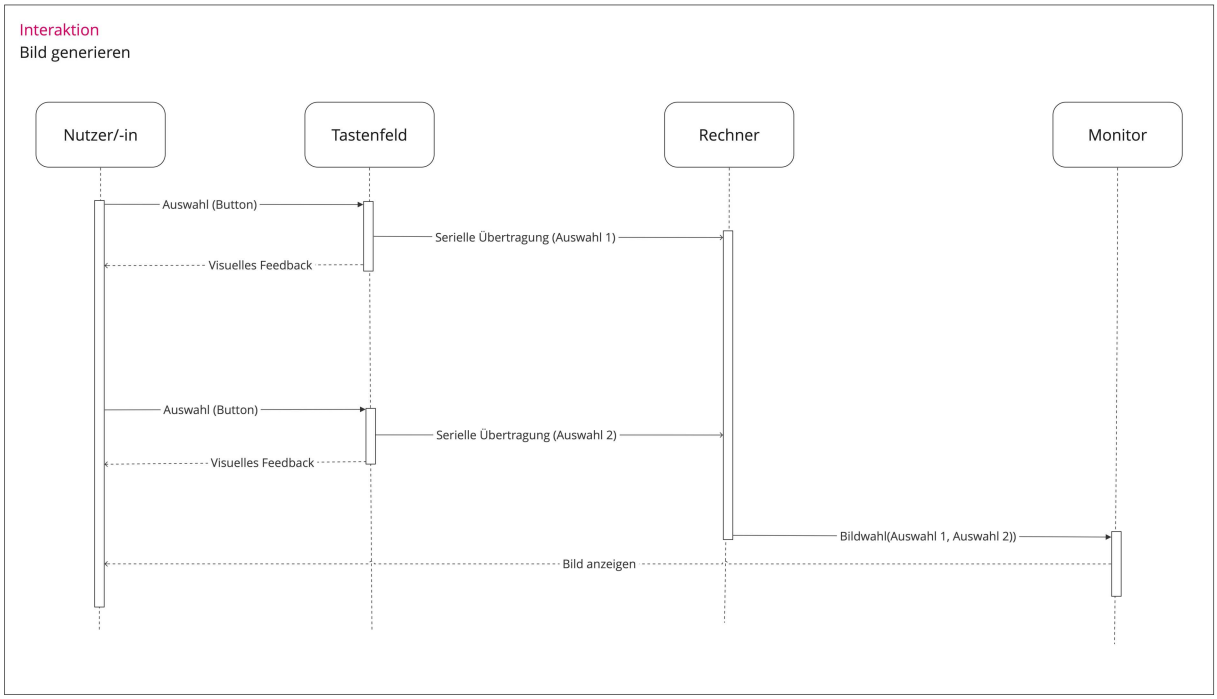

Abbildung 4.3 – Sequenzdiagramm der Medienstation

5 Technische Spezifikationen

Dieses Kapitel listet alle verwendeten Komponenten, skizziert den Schaltplan und beschreibt die verwendete Software.

5.1 Tastenfeld Bauteile

Bauteil	Anzahl	Beschreibung
Allgemein		
Mikrocontroller	1	Arduino Mega 2560 Rev3
Perfboard	1	Breadboard Design
Taster		
Drucktaster	8	3 V, 19 mm, verchromt und mit weißem LED-Ring
Widerstand	1	1 kΩ
Isolierter Kupferschaltdraht	1	∅0.5 mm, 1-adrig
LED-Streifen		
LED-Strip	1	WS2812B, ca. 1 m
Widerstand	1	470 Ω
Isolierter Kupferschaltdraht	1	∅0.5 mm, 1-adrig
Kondensator	1	1000 µF
Adapter Schraubkontakt zu Hohlstecker Buchse	1	5.5 x 2.5 mm
Netzteil	1	5 V, 6 A

5.2 Tastenfeld Schaltplan

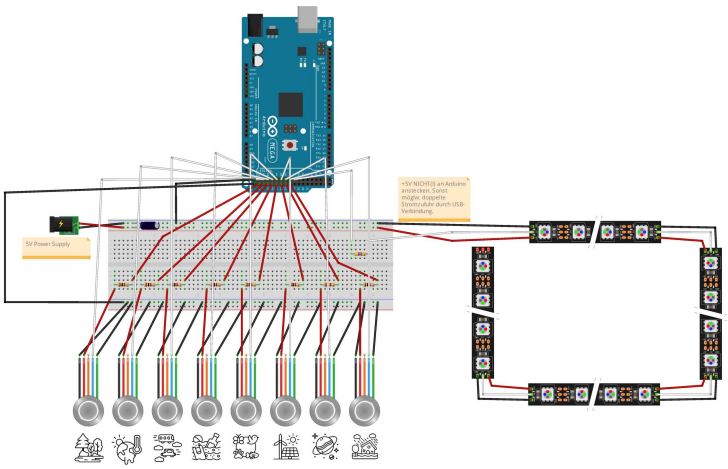

Abbildung 5.1 – Schaltplan

5.3 Arduino Sketch

Falls der Arduino defekt ist und ausgetauscht werden muss, so kann mit Hilfe des Quellcodes 5.1 der Auslieferungszustand wiederhergestellt werden. Die Software *Arduino IDE* ist bereits auf dem Rechner installiert, sodass der Arduino einfach per USB-Kabel angeschlossen und ein neuer Sketch erstellt werden kann. Abschließend erfolgt eine Validierung und der Upload des Sketches [HS24]. Sollte ein Austausch oder eine Neuinstallation des Rechners erforderlich sein, kann die erforderliche Software, wie Entwicklungsumgebung [Söd24] und verwendete Bibliothek [BR24], neu installiert werden.

```
#include <Adafruit NeoPixel.h> // LED-strip library
#include <stdio.h>

// Setup digital pins for all buttons
#define BTN_1_PIN 38
// ... every 2 pins (38, 40, ...)
#define BTN_8_PIN 52

// Setup digital pins for all LEDs (buttons and strip)
#define LED_1_PIN 39
// ... every 2 pins (39, 41, ...)
#define LED_8_PIN 53

#define LED_STRIP_PIN 36
#define LED_STRIP_COUNT 83 // Number of LEDs (current strip length)

struct Button {
    const uint8_t btnPin;
    const uint8_t ledPin;
    bool isPressed;
}

struct Button buttons[] = {
    {BTN_1_PIN, LED_1_PIN, false},
    // ...
    {BTN_8_PIN, LED_8_PIN, false}
};
// Array for the flash animation
uint8_t ledList[] = (
    LED_1_PIN,
    // ...
    LED_8_PIN
);

// strip initialization depends on the type
Adafruit_NeoPixel strip(LED_STRIP_COUNT, LED_STRIP_PIN, NEO_GRB + NEO_KHZ800);

const size_t NUMBTNS = sizeof(buttons) / sizeof(buttons[0]);
const uint8_t BRIGHTNESS = 75;
const uint32_t WHITE = strip.Color(255, 255, 255);

bool flashed = false; // LED flash animation ran on first button pressed?
uint8_t sequenceIndex = 0; // How many buttons were pressed already?

```

```
45 void setup() {
46     Serial.begin(115200);
47     initStrip();
48     for(uint8_t i = 0; i < NUMBTNS; i++) {
49         pinMode(buttons[i].btnPin, INPUT_PULLUP);
50         pinMode(buttons[i].ledPin, OUTPUT;
51     }
52 }
53 // check the number of buttons pressed:
54 // 1 - flash animation, 2 -> send data
55 void loop() {
56     uint8_t i;
57     if (sequenceIndex < 2) {
58         for (i = 0; i < NUMBTNS; i++) {
59             if (digitalRead(buttons[i].btnPin) == LOW && !buttons[i].isPressed) {
60                 buttons[i].isPressed = true;
61                 digitalWrite(buttons[i].ledPin, HIGH);
62                 Serial.write(i+1); // scenario representative
63                 delay(100);
64                 sequenceIndex++;
65                 break;
66             }
67         }
68         if (sequenceIndex == 1 && !flashed) {
69             for (i = 0; i < NUMBTNS; i++) {
70                 sequentialFlash(i); // flash animation
71                 flashed = true;
72                 break;
73             }
74         }
75     }
76     else if (sequenceIndex > 1) {
77         strip.clear(); // turn off led-strip
78         strip.show(); // make changes
79         delay(20000); // during this time the AI image is shown
80         reset(); // Initial state, i.e. deactivate all LED-buttons, activate LED-strip, ...
81     }
82     delay(100);
83 }
84 void reset() {
85     sequenceIndex = 0;
86     flashed = false;
87     for (uint8_t i = 0; i < NUMBTNS; i++) {
88         buttons[i].isPressed = false;
89         digitalWrite(buttons[i].ledPin, LOW);
90     }
91     initStrip();
92 }
93
94 void initStrip() {
95     strip.begin();
96     strip.setBrightness(BRIGHTNESS);
97     strip.fill(WHITE);
98     strip.show();
99 }
```

```
100
101 void sequentialFlash(uint8_t idx) {
102     uint8_t i = idx + 1;
103     uint8_t j = idx - 1;
104     while (true) {
105         if (i < 8) {
106             digitalWrite(ledList[i], HIGH);
107         }
108         if (j >= 0) {
109             digitalWrite, HIGH);
110         }
111         delay(150); // adjust to make changes to the animation speed
112         if (i < 8) {
113             digitalWrite(ledList[i], LOW);
114         }
115         if (j >= 0) {
116             digitalWrite, LOW);
117         }
118         if (i >= 7 && j <= 0) {
119             break;
120         }
121         else if (j > 0 || i < 7) {
122             i++;
123             j--;
124         }
125     }
126 }
```

Quellcode 5.1 – Sketch

5.4 Applikation der Medienstation

Ein Skript startet die Anwendung automatisch bei Systemstart. Dazu wird eine virtuelle Entwicklungsumgebung mit den vorab installierten Bibliotheken aktiviert. (vgl. Quellcode 5.2)

```
1 ./skd-tudd/Scripts/activate
2 python ./src/main.py
```

Quellcode 5.2 – bash

Anforderungen

Die Anwendung wurde mit Python in der Version 3.12.0 entwickelt und benötigt die folgenden Bibliotheken:

1. future==1.0.0
2. iso8601==2.1.0
3. pillow==10.3.0
4. pyserial==3.5
5. PyYAML==6.0.1
6. serial==0.0.97

Implementation

Die Applikation verteilt sich auf fünf Skripte: *app.py*, *controller.py*, *log.py*, *main.py* und *scenario.py*.

```python
import time

from app import App
from controller import Controller
from log import logging
from serial import SerialException
from scenario import *

# MAIN PROGRAMM
if __name__ == '__main__':

    # App & GUI setup
    min_width = 640
    min_height = 480

    app = App(min_width, min_height)
    port_controller = Controller()

    # Sets ignore duplicates (sometimes happens on button press)
    combination = set()

    while app.running:
        app.update()
        try:
            # Convert received bytes and add to the combination set
            if port_controller.arduino.in_waiting > 0:
                selection = ord(port_controller.arduino.readline())
                logging.info(f'Data received -> {selection}')
                combination.add(selection)

            # If a combination of two scenarios has been selected, display AI image
            if len(combination) == 2:
                selected = Scenario.select(combination)
                app.choose_topic(selected)
                combination.clear()

        # Serial Disconnection Handling
        except SerialException:
            # Only log the first disconnection and reset the image
            if port_controller.arduino.is_open:
                logging.error("The Arduino disconnected.")
                combination.clear()
                port_controller.arduino.close()
            # otherwise just try to reconnect
            port_controller.reconnect()

        # Debugging
        except:
            logging.error(f"Something else went wrong.")

        # Optional, but recommended since other threads can run during this time
        time.sleep(.01)
```

Quellcode 5.3 – main.py

```
import glob, os
from enum import Enum

# Enumeration of scenario names for 'better' code readability
class Scenarios(Enum):
    WARMING = 7
    FLOODING = 1
    SPACE = 2
    MOBILITY = 6
    ANIMALS = 4
    POLLUTION = 5
    NATURE = 8
    ENERGIES = 3
# Scenario based folders for AI images
# Keep track of images that were already displayed
class Scenario:
    def __init__(self, scenario) -> None:
        self.name = f"{scenario}"
        self.index = 0

        scenario_names_list = scenario.split("_")
        pre_sorted_list = []
        for s in scenario_names_list:
            pre_sorted_list.append(s[0])
        folder = "".join(sorted(pre_sorted_list))

        self.image_list = glob.glob(f"**/{folder}/*")

    def __repr__(self) -> str:
        return f'\t{self.name}\n\t{os.path.basename(self.image_list[self.index])}'

    # Function: return the corresponding scenario of the selected combination set
    def select(selected_set:set[int]) -> object:
        # transform given selected set into a string, list of sorted numbers
        selection_string = ','.join(map(lambda s: str(s), sorted(selected_set)))
        scenarios = {
            f"{Scenarios.FLOODING.value},{Scenarios.WARMING.value}": warming_flood,
            f"{Scenarios.SPACE.value},{Scenarios.WARMING.value}": warming_space,
            f"{Scenarios.MOBILITY.value},{Scenarios.WARMING.value}": warming_mobility,
            f"{Scenarios.ANIMALS.value},{Scenarios.WARMING.value}": warming_animals,
            f"{Scenarios.POLLUTION.value},{Scenarios.WARMING.value}": warming_pollution,
            f"{Scenarios.ENERGIES.value},{Scenarios.WARMING.value}": warming_energies,
            f"{Scenarios.WARMING.value},{Scenarios.NATURE.value}": warming_nature,
            f"{Scenarios.FLOODING.value},{Scenarios.SPACE.value}": flood_space,
            f"{Scenarios.FLOODING.value},{Scenarios.MOBILITY.value}": flood_mobility,
            f"{Scenarios.FLOODING.value},{Scenarios.ANIMALS.value}": flood_animals,
            f"{Scenarios.FLOODING.value},{Scenarios.POLLUTION.value}": flood_pollution,
            f"{Scenarios.FLOODING.value},{Scenarios.ENERGIES.value}": flood_energies,
            f"{Scenarios.FLOODING.value},{Scenarios.NATURE.value}": flood_nature,
            f"{Scenarios.SPACE.value},{Scenarios.MOBILITY.value}": space_mobility,
            f"{Scenarios.SPACE.value},{Scenarios.ANIMALS.value}": space_animals,
            f"{Scenarios.SPACE.value},{Scenarios.POLLUTION.value}": space_pollution,
            f"{Scenarios.SPACE.value},{Scenarios.ENERGIES.value}": space_energies,
            f"{Scenarios.SPACE.value},{Scenarios.NATURE.value}": space_nature,
```

```
55              f"{Scenarios.ANIMALS.value},{Scenarios.MOBILITY.value}": mobility_animals,
56              f"{Scenarios.POLLUTION.value},{Scenarios.MOBILITY.value}": mobility_pollution,
57              f"{Scenarios.ENERGIES.value},{Scenarios.MOBILITY.value}": mobility_energies,
58              f"{Scenarios.MOBILITY.value},{Scenarios.NATURE.value}": mobility_nature,
59              f"{Scenarios.ANIMALS.value},{Scenarios.POLLUTION.value}": animals_pollution,
60              f"{Scenarios.ENERGIES.value},{Scenarios.ANIMALS.value}": animals_energies,
61              f"{Scenarios.ANIMALS.value},{Scenarios.NATURE.value}": animals_nature,
62              f"{Scenarios.ENERGIES.value},{Scenarios.POLLUTION.value}": pollution_energies,
63              f"{Scenarios.POLLUTION.value},{Scenarios.NATURE.value}": pollution_nature,
64              f"{Scenarios.ENERGIES.value},{Scenarios.NATURE.value}": energies_nature
65          }
66          return scenarios.get(selection_string, None)
67
68  # 28 Categories (Scenarios)
69  warming_flood = Scenario('warming_flood')
70  warming_space = Scenario('warming_space')
71  warming_mobility = Scenario('warming_mobility')
72  warming_animals = Scenario('warming_animals')
73  warming_pollution = Scenario('warming_pollution')
74  warming_energies = Scenario('warming_energies')
75  warming_nature = Scenario('warming_nature')
76  flood_space = Scenario('flood_space')
77  flood_mobility = Scenario('flood_mobility')
78  flood_animals = Scenario('flood_animals')
79  flood_pollution = Scenario('flood_pollution')
80  flood_energies = Scenario('flood_energies')
81  flood_nature = Scenario('flood_nature')
82  space_mobility = Scenario('space_mobility')
83  space_animals = Scenario('space_animals')
84  space_pollution = Scenario('space_pollution')
85  space_energies = Scenario('space_energies')
86  space_nature = Scenario('space_nature')
87  mobility_animals = Scenario('mobility_animals')
88  mobility_pollution = Scenario('mobility_pollution')
89  mobility_energies = Scenario('mobility_energies')
90  mobility_nature = Scenario('mobility_nature')
91  animals_pollution = Scenario('animals_pollution')
92  animals_energies = Scenario('animals_energies')
93  animals_nature = Scenario('animals_nature')
94  pollution_energies = Scenario('pollution_energies')
95  pollution_nature = Scenario('pollution_nature')
96  energies_nature = Scenario('energies_nature')
```

Quellcode 5.4 – scenario.py

```
1  import logging
2  # Logging format and configuration
3  logging.basicConfig(
4      level=logging.INFO,
5      format='%(asctime)s %(levelname)s: %(message)s',
6      datefmt='%H:%M:%S',
7  )
```

Quellcode 5.5 – logging.py

```python
import serial, time

from log import logging
from serial.tools import list_ports
from serial import SerialException

class Controller():
    def __init__(self) -> None:
        ports = self.get_ports()
        logging.info(f'Ports -> {ports}')
        arduino_port = self.get_arduino_port(ports)

        try:
            self.arduino = serial.Serial(port=arduino_port, baudrate=115200, timeout=.1)
        except SerialException:
            logging.error(f'Port {arduino_port} can not be found or configured!')
        except ValueError:
            logging.error(f'Parameters of Serial Port are out of range!')
        except:
            logging.error(f'An error occured: Trying to create the port {arduino_port}')

        if self.arduino.port == None:
            self.disconnected = True
            logging.error(f'The Arduino could not be found and/ or configured!')
        else:
            self.disconnected = False
            logging.info(f'The Arduino is listening on port {arduino_port}.')

    # Check for all active ports
    def get_ports(self) -> list[(str, str)]:
        port = list(list_ports.comports())
        ports = []
        for p in port:
            ports.append((p.name, p.manufacturer))
        return ports

    # If arduino is present, return the corresponding port
    def get_arduino_port(self, ports) -> str:
        for port in ports:
            description = port[1].split()
            for d in description:
                if d.lower() == "arduino":
                    return port[0]
        return None

    def reconnect(self):
        was_connected = True
        if self.arduino.port == None:
            was_connected = False

        ports = self.get_ports()
        arduino_port = self.get_arduino_port(ports)

        if arduino_port != None:
```

```
55              self.disconnected = False
56              # It actually helps to wait for other threads,
57              # otherwise the first 1... 10 reconnection attempts fail with the current setup
58              time.sleep(.1)
59
60              try:
61                  self.arduino.port = arduino_port
62                  self.arduino.open()
63                  if was_connected:
64                      logging.info(f"The Arduino reconnected on port {arduino_port}.")
65                  else:
66                      logging.info(f"The Arduino is now listening on port {arduino_port}.")
67
68              except SerialException:
69                  logging.error(f'Can not open Serial Connection on port {arduino_port}')
70                  self.arduino.close()
71                  self.disconnected = True
72              except:
73                  logging.error(f'Something went wrong while trying to reconnect.')
```

Quellcode 5.6 – controller.py

```
 1 import time
 2 import tkinter as tk
 3
 4 from datetime import datetime
 5 from log import logging
 6 from PIL import Image, ImageTk
 7 from scenario import Scenario
 8
 9 class App(tk.Tk):
10     # App constructor with constants and imagelabel as well as keybindings
11     def __init__(self, min_width, min_height):
12         super().__init__()
13         self.running = True
14
15         self.MIN_WIDTH = min_width
16         self.MIN_HEIGHT = min_height
17         self.SCREEN_WIDTH = self.winfo_screenwidth()
18         self.SCREEN_HEIGHT = self.winfo_screenheight()
19
20         self.title("Kinderbiennale 2024")
21         self.geometry(f'{self.MIN_WIDTH}x{self.MIN_HEIGHT}')
22         self.attributes("-fullscreen", True)
23         self.wm_protocol("WM_DELETE_WINDOW", self._quit)
24
25         self.image = self.RESET_IMAGE = Image.open('./images/Canaletto.jpg')
26         resized_image = self.image.resize(
27             (self.SCREEN_WIDTH, self.SCREEN_HEIGHT),
28             Image.Resampling.LANCZOS
29         )
30         self.photo = ImageTk.PhotoImage(resized_image)
31
32         self.label = tk.Label(self, image=self.photo)
33         self.label.pack()
```

```
34
35          self.bind('<Key>', self.on_key_press)
36          self.config(cursor='none')
37
38      # Quit on pressing 'x' in the title bar
39      def _quit(self):
40          logging.info(f'Exiting...')
41          self.running = False
42
43      # Function: Enter/ leave fullscreen mode or exit
44      def on_key_press(self, event) -> None:
45          if event.char == 'f':
46              self.enter_leave_fullscreen()
47              self.resize_Image()
48          elif event.keysym == 'Escape':
49              self._quit()
50
51      # Function: Fit window to screen size with in-built tkinter properties
52      def enter_leave_fullscreen(self) -> None:
53          if self.attributes('-fullscreen'):
54              self.attributes('-fullscreen', False)
55          else:
56              self.attributes('-fullscreen', True)
57
58      # Function: Resizing the window
59      def resize_Image(self) -> None:
60          width, height = self.get_app_size()
61          logging.info(f'Resizing to {width}x{height}')
62
63          img = self.image.resize((width, height), Image.Resampling.LANCZOS)
64          photo = ImageTk.PhotoImage(img)
65          self.label.config(image=photo)
66          self.label.image = photo
67
68      # Choose a topic
69      def choose_topic(self, scenario : Scenario) -> None:
70          logging.info(f'Beginning to crossfade...\n{scenario}')
71          width, height = self.get_app_size()
72          # pick an image at the current index (scenario - image list)
73          image = Image.open(scenario.image_list[scenario.index])
74
75          # resize that image to fit the current screen size
76          resized_image = image.resize((width, height), Image.Resampling.LANCZOS)
77          # resize the currently displayed image
78          resized_current_image = self.RESET_IMAGE.resize(
79              (width, height),
80              Image.Resampling.LANCZOS
81          )
82          # Calculate delta time dynamically (varies with the resolution) to sync with 20s
83          start = datetime.now()
84          self.blend_images(resized_current_image, resized_image)
85          end = datetime.now()
86          delta_time = (end - start).total_seconds()
87          time.sleep(20.0 - delta_time*2) # wait for 20 s then reset
88          self.blend_images(resized_image, resized_current_image)
```

```
89          scenario.index += 1
90          # Reset object/ scenario index, if all images have been displayed
91          # display all images inside the ai folders sequentially
92          # largest interval between identical images
93          if scenario.index >= len(scenario.image_list):
94              scenario.index = 0
95          logging.info(f'Crossfading and reset completed.')
96
97      # Function: Crossfade two AI images
98      def blend_images(self, from_image, to_image) -> None:
99          alpha = .0
100         while alpha < 1.0:
101             img_blended = Image.blend(from_image, to_image, alpha = alpha)
102             photo = ImageTk.PhotoImage(img_blended)
103             self.label.config(image = photo)
104             self.label.image = photo
105             self.label.update()
106             alpha += self.get_animation_speed()
107
108     def get_app_size(self) -> tuple[int, int]:
109         if self.attributes('-fullscreen'):
110             width, height = self.SCREEN_WIDTH, self.SCREEN_HEIGHT
111         else:
112             width, height = self.MIN_WIDTH, self.MIN_HEIGHT,
113         return width, height
114
115     # set animation speed accordingly
116     def get_animation_speed(self) -> float:
117         if self.get_app_size()[0] > 2000:
118             return .02
119         elif self.get_app_size()[0] > 1000:
120             return .01
121         else:
122             return 0.001
```

Quellcode 5.7 – app.py

6 Fehlersuche und Problemlösung

Bei Problemen jeglicher Art sollte zunächst versucht werden, den Fehler durch einen Neustart des Rechners zu beheben. Dazu kann die Medienstation zentral vom Strom getrennt und wieder angeschlossen werden. Sollte dies nicht zum Erfolg führen, kann gegebenenfalls eine externe Tastatur und Maus angeschlossen werden. Ist die Anwendung gestartet, kann der Vollbildmodus durch Drücken der Taste *f* verlassen werden und es erscheint die Shell mit Protokollinformation, welche Auskunft über den Systemzustand der Anwendung geben (vgl. Abbildung 6.1). Während

Abbildung 6.1 – Beispiel für Protokollinformationen

der Entwicklung der Medienstation sind Fehler aufgetreten, deren Lösungen in der nachfolgenden Tabelle aufgeführt sind:

Problem	**Lösung**
Nach dem Start der Medienstation erscheint die Befehlszeilenschnittstelle bzw. Shell vor der Anwendung.	Tastatur: Tastenkombination *Alt-Tab* drücken und das Anwendungsfenster auswählen. Maus: Direktes Anklicken des Anwendungsfensters.
Das Tastenfeld gibt Rückmeldungen in Form der LEDs, doch trotz der Auswahl zweier Taster erscheint kein neues KI-Bild.	Möglicherweise ist der PC nicht mit dem Tastenfeld verbunden: USB Kabel neu stecken oder austauschen.
Das Tastenfeld gibt generell Rückmeldung in Form von LEDs, aber einzelne LED-Taster leuchten nicht.	Unter Umständen verursacht ein Wackelkontakt an den digitalen Pins des Arduino die Störung und Kabel haben keinen durchgehenden Kontakt. Das Gehäuse sollte geöffnet und die Schaltung untersucht werden.
Das Tastenfeld gibt Rückmeldung in Form der LED-Taster, aber der LED-Streifen leuchtet nicht.	Eventuell wird der LED-Streifen nicht mit Strom versorgt. In diesem Fall sollte das Netzteil überprüft werden. Außerdem kann die Lebensdauer überschritten sein.
Das angezeigte Bild erscheint verpixelt.	Wahrscheinlich sind die Anzeigeeinstellungen unter Windows nicht korrekt konfiguriert. Eine Änderung der Auflösung oder ein Wechsel des Anzeigezooms auf 100% können zur Besserung führen.
Die Applikation der Medienstation beendet sich unerwartet.	Nach dem Wechsel in das Verzeichnis *C:\Users\admin-lokal\kinderbiennaleapp* kann das Skript *main.ps1* ausgeführt werden.

Das Gehäuse des Tastenfeldes ist lasiert und kann daher mit Wasser gereinigt werden. Falls weitere Probleme auftreten, welche nicht gelöst werden können, besteht die Möglichkeit sich an die Kontaktpersonen der Medienstation zu wenden.

Abbildungsverzeichnis

1.1 KI Medieninstallation: Dresden – Utopia – Dystopia 1

2.1 Tastenfeld: Legende 3
2.2 Tastenfeld: Skizze 4

3.1 *Dresden vom rechten Elbufer unterhalb der Augustusbrücke*, Bernardo Bellotto, 1748 . 5

4.1 Piktogramme 7
4.2 Zustandsdiagramm der Medienstation 7
4.3 Sequenzdiagramm der Medienstation 8

5.1 Schaltplan 9

6.1 Beispiel für Protokollinformationen 21

Literatur

[BR24] Phillip Burgess und Kattni Rembor. *Adafruit NeoPixel Überguide*. Adafruit Learning System. 27. Mai 2024. URL: https://learn.adafruit.com/adafruit-neopixel-uberguide/arduino-library-installation (besucht am 28. 05. 2024) (siehe S. 10).

[HS24] Jacob Hylén und Karl Söderby. *How to upload a sketch with the Arduino IDE 2 | Arduino Documentation*. 17. Jan. 2024. URL: https://docs.arduino.cc/software/ide-v2/tutorials/getting-started/ide-v2-uploading-a-sketch/ (besucht am 28. 05. 2024) (siehe S. 10).

[SKD24] Staatliche Kunstsammlungen Dresden SKD. *Kinderbiennale* "PLANET UTOPIA". Japanisches Palais: Kinderbiennale Planet Utopia. 2024. URL: https://japanisches-palais.skd.museum/ausstellungen/kinderbiennale-planet-utopia/ (besucht am 24. 05. 2024) (siehe S. 1).

[Söd24] Karl Söderby. *Downloading and installing the Arduino IDE 2 | Arduino Documentation*. 28. Feb. 2024. URL: https://docs.arduino.cc/software/ide-v2/tutorials/getting-started/ide-v2-downloading-and-installing/ (besucht am 28. 05. 2024) (siehe S. 10).

Abkürzungsverzeichnis

SKD Staatliche Kunstsammlungen Dresden 1
KI Künstliche Intelligenz 3

Abbildungsverzeichnis

1.1 Medienstation: Dresden - Utopia - Dystopia, Foto: Oliver Killig 1

2.1 Entwicklungsprozess eines Schlagwortes mit Piktogramm 5
2.2 Verschiedene Kamerapositionen 6
2.3 KI Bild auf Basis von Abbildung 2.2 7

3.1 Inspirationsbilder für *Rückeroberung der Natur* 10
3.2 Konzeptionelle Repräsentation des Automatic1111 Workflows 11
3.3 Einige beispielhafte Ergebnisse des iterativen Workflows in Automatic1111 14
3.4 Konzeptionelle Repräsentation des ComfyUI Workflows 15
3.5 Beispielresultat des ComfyUI All-in-one-Workflows für die Kategorien Space und Mobilität der Zukunft. 16

4.1 Prototyp 19
4.2 Konstruktion Prototyp Tastenfeld 20
4.3 Schematische Darstellung der Schaltung für den Prototypen 21

6.1 Finale Medienstation im Japanischen Palais 27
6.2 Entwicklung einer Webanwendung für diverse Endgeräte 28

Literatur

[AUT22] AUTOMATIC1111. *Stable Diffusion Web UI*. original-date: 2022-08-22T14:05:26Z. Aug. 2022. URL: https://github.com/AUTOMATIC1111/stable-diffusion-webui (besucht am 23. 07. 2024) (siehe S. 11).

[Autar] Automatic1111. *Features*. en. no year. URL: https://github.com/AUTOMATIC1111/stable-diffusion-webui/wiki/Features (besucht am 30. 07. 2024) (siehe S. 12).

[com23] comfyanonymous. *comfyanonymous/ComfyUI*. original-date: 2023-01-17T03:15:56Z. Jan. 2023. URL: https://github.com/comfyanonymous/ComfyUI (besucht am 04. 07. 2024) (siehe S. 13).

[Dif24] Diffusers. *Inpainting*. 2024. URL: https://huggingface.co/docs/diffusers/using-diffusers/inpaint (besucht am 12. 07. 2024) (siehe S. 12).

[Goe24] Felix Andreas Goebel. *Felkonsky/kinderbiennale-webapp*. original-date: 2024-06-04T14:10:44Z. 12. Juni 2024. URL: https://github.com/Felkonsky/kinderbiennale-webapp (besucht am 20. 06. 2024) (siehe S. 28).

[LD23] Lanea Lilienthal und Laura Dietrich. *KP 23 KiBi Konzeption - Miro*. 2023. URL: https://miro.com/app/board/uXjVMbJWLZs=/ (besucht am 10. 06. 2024) (siehe S. 1).

[Ope22] OpenAI. *DALL·E: Introducing outpainting*. en-US. 2022. URL: https://openai.com/index/dall-e-introducing-outpainting/ (besucht am 30. 07. 2024) (siehe S. 12).

[Pod+23] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna und Robin Rombach. *SDXL: Improving Latent Diffusion Models for High-Resolution Image Synthesis*. en. arXiv:2307.01952 [cs]. Juli 2023. URL: http://arxiv.org/abs/2307.01952 (besucht am 27. 05. 2024) (siehe S. 14).

[Rom+21] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser und Björn Ommer. *High-Resolution Image Synthesis with Latent Diffusion Models*. 2021. arXiv: 2112.10752 [cs.CV] (siehe S. 9).

[SKD24] SKD. *Japanisches Palais: Kinderbiennale Planet Utopia*. 2024. URL: https://japanisches-palais.skd.museum/ausstellungen/kinderbiennale-planet-utopia/ (besucht am 10. 06. 2024) (siehe S. 1).

[ssi23] ssitu. *ssitu/ComfyUI_UltimateSDUpscale*. original-date: 2023-05-16T02:40:34Z. Mai 2023. URL: https://github.com/ssitu/ComfyUI_UltimateSDUpscale (besucht am 06. 08. 2024) (siehe S. 15).

[Ye+23] Hu Ye, Jun Zhang, Sibo Liu, Xiao Han und Wei Yang. *IP-Adapter: Text Compatible Image Prompt Adapter for Text-to-Image Diffusion Models*. en. arXiv:2308.06721 [cs]. Aug. 2023. URL: http://arxiv.org/abs/2308.06721 (besucht am 10. 05. 2024) (siehe S. 12).

[ZRA23] Lvmin Zhang, Anyi Rao und Maneesh Agrawala. *Adding Conditional Control to Text-to-Image Diffusion Models*. arXiv:2302.05543 [cs]. Nov. 2023. DOI: 10.48550/arXiv.2302.05543. URL: http://arxiv.org/abs/2302.05543 (besucht am 06. 05. 2024) (siehe S. 11).