ECE 445

SENIOR DESIGN LABORATORY

FINAL REPORT

AR Sandbox

Team #3

HAOZE GAO (haozeg2@illinois.edu)
HAOWEN ZHENG
(haowenz5@illinois.edu)
QIRAN PAN (qiranp2@illinois.edu)
YIHENG ZHANG
(yihengz5@illinois.edu)

Advisor: Timothy Lee

May 7, 2024

Abstract

Projector-based Augmented Reality (AR) systems are widely used in various applications such as interactive gaming, education, and training. Combined with the conventional sandbox, the AR system provides a new way to visualize and interact with the real world and intuitively understand concepts in geological, hydrological, and topographical sciences. This thesis presents the design and implementation of an AR sandbox system that projects a topographic map onto a sandbox in real time, provides an easier way to calibrate the AR system, and presents a more portable and easy-to-use system. With the implementation of a calibration algorithm and the integration of the projector-camera system, the AR sandbox system can calibrate itself without the need for manual intervention. The calibration result is accurate and robust, and the system can be easily set up and used by users without any technical background.

Key words: Computer Vision, Augmented Reality, Camera Calibration, Projector-Camera System, Ray Casting

Contents

1 Introduction

1.1 Problem Statement

Sandbox is a traditional educational tool that has been widely used in geography education for children. It is a hands-on tool that allows children to learn about topography and geography in a fun and interactive way. With Augmented Reality (AR) technology, sandboxes can be transformed into a more interactive and engaging learning tool. Currently available AR sandboxes [1] [2] are mostly cumbersome and limited to public spaces like activity centers rather than serving as personalized learning tools. For these sandboxes, a significant lag can be noticed when sand is manipulated manually, as it causes contours to be recalculated and re-projected onto the terrain. Meanwhile, contours are flickering on the terrain parts that remain still for the whole time, showing a heavy amount of computational resources are wasted on the unnecessary parts thus restricting more delicate and faster signal processing procedures on the terrain parts that need to be updated as soon as possible. Furthermore, the existing projectors designed for sandboxes exhibit primitive features, characterized by a notably low refresh rate and harsh direct light from the projector that raise safety concerns for children's eyesight.

This AR sandbox uses a different AR module design that can project contour maps in real-time onto the sand surface and can be easily installed to adapt to various kinds of sandboxes, making geography education for children not only informative but also significantly more enjoyable. This solution aims to overcome the limitations of previous AR sandboxes that are bulky and not user-friendly, offering a more accessible and personal learning experience. It is designed to deliver a smoother experience with lower latency that ensures a more comfortable and engaging educational tool for children.

1.2 Solution Overview

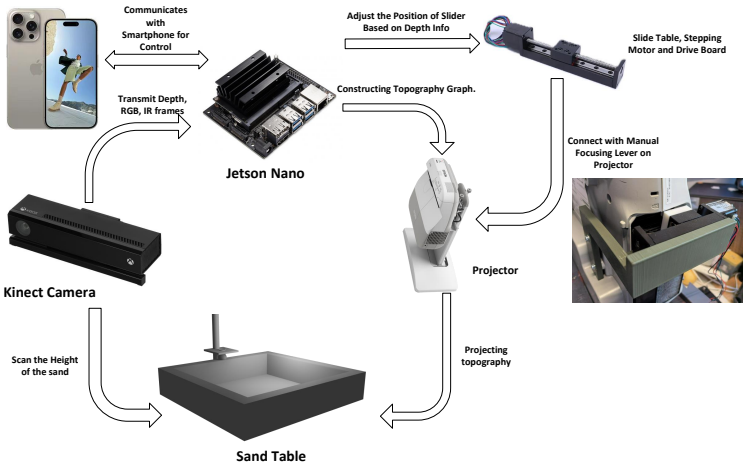


Figure 1: Visual Aid for AR Sandbox Project

The basic criteria for this sandbox are safe, easy to use, and low latency. The overall

structure will be designed to be portable while ensuring both high load-bearing capacity and stability. The AR module, including a depth camera, processing unit, and projector, can be removed from the sandbox and installed on other sand tables. In general, this project will ensure a latency of less than $1\,\mathrm{s}$, and the contours can be accessed wirelessly from portable mobile devices. The projector can be auto-focused to ensure display clarity. The depth camera will be used to detect the height of the sand and collect RGB and IR images. The projector will be of high luminance to display clearly on the sand and short throw to match the camera's field of view and lower the center of mass for safety.

1.3 Block Diagram

The AR sandbox project consists of six subsystems: sensor, processing, projection, powering, structure, and wireless display. The sensor subsystem acquires depth information from the sand surface. The processing subsystem is responsible for processing the depth information and projecting the correct topography graph onto the sand. The projection subsystem is responsible for projecting the topography map onto the sand surface. The powering subsystem provides power to all subsystems. The structure subsystem is responsible for storing the sand and connecting it with all other parts as a whole AR module. The wireless display subsystem provides a web page to view contours and control the sandbox. The block diagram is shown in Figure 2.

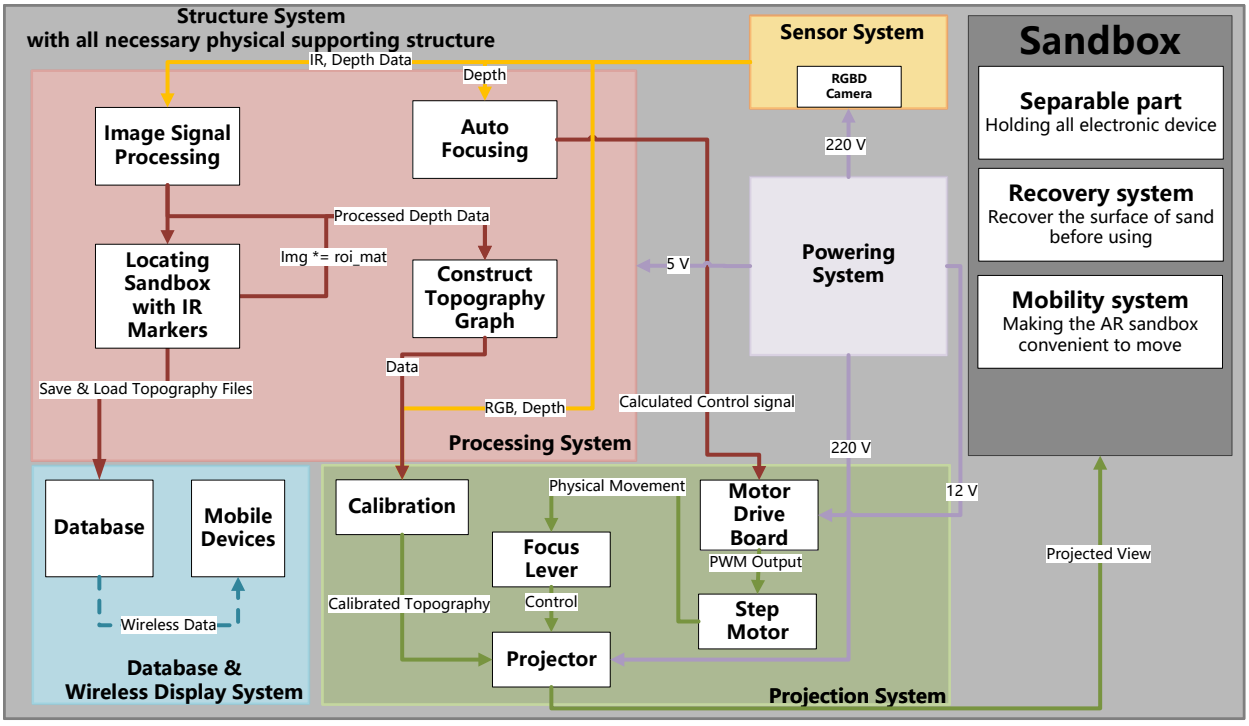


Figure 2: Block Diagram for AR Sandbox Project

2 Design

2.1 Sensor Subsystem

The sensor subsystem in the AR sandbox project refers to the RGB-D camera that is responsible for acquiring depth information from the sand surface and collecting RGB and IR images for detecting the shape of the sandbox.

For depth information acquisition, Microsoft Kinect V2 is chosen for its capability of providing depth information in low light levels, providing RGB and IR frames, and is color and texture invariant. It has a color sensor with a $1920 \times 1080\,\mathrm{p}$ixels resolution and a depth sensor with a $512 \times 424\,\mathrm{p}$ixels resolution. The operational area is delineated by a depth span from 0.5 to 4.5 meters, with a viewing angle of 70° horizontally and 60° vertically [3]. When detecting objects at a depth range from $0.5\,\mathrm{m}$ to $2\,\mathrm{m}$, the average depth accuracy error is less than $4\,\mathrm{mm}$ according to Yang et al. [4]. The Kinect V2 camera is connected to the processing unit through a USB 3.0 cable. The sensor communicates with the processing unit through USB protocol. An open-source library called "libfreenect2" [5] is used to access the RGB, IR, and depth information from the Kinect V2 camera.

2.2 Processing Subsystem

2.2.1 Processor Selection

The chosen processor for our project is the Jetson Nano, a compact yet powerful computer designed specifically for embedded applications and AI IoT [6]. The Jetson Nano delivers the performance and capabilities required to run modern AI workloads efficiently.

The primary consideration for selecting the Jetson Nano was its balance of cost and performance. Our project involves considerable Computer Vision (CV) tasks that require robust GPU support. The specifications of the Jetson Nano are as follows:

- GPU: NVIDIA Maxwell architecture with 128 NVIDIA CUDA cores
- CPU: Quad-core ARM Cortex-A57 MPCore processor
- Memory: 4 GB 64-bit LPDDR4, 25.6 GB/s
- Operating System: Built-in Linux, facilitating development

These features and the price provide a compelling option for our needs.

2.2.2 IR Markers for Sandbox Boundary Detection

To achieve robustness and scalability, adaptable to sandboxes of various shapes, infrared reflective markers were selected for this task, utilizing the Kinect v2 camera's ability to capture infrared (IR) information. These markers were strategically placed at each corner of the sandbox, enabling the camera to capture these markers with minimal interference. The markers are easily distinguishable from other objects in the IR spectrum, which fa-

cilitates using OpenCV to identify these markers and ultimately determine the sandbox boundaries.

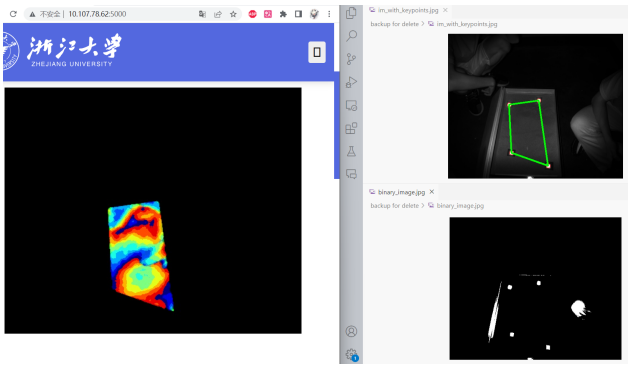


Figure 3: Processed binary IR image(bottom right) and Detected Sandbox Edge after doing blob detection(upper right)

Algorithm for IR Marker Boundary Configuration The pseudocode for IR marker detection in Appendix E shows a structured framework for implementing IR marker detection by leveraging the Kinect v2 camera alongside image processing capabilities provided by OpenCV. Details are as follows:

1. **Preprocess IR Information:** Convert the grayscale image to a binary image using appropriate thresholds. This step isolates the relevant high-intensity areas for further analysis.
2. **Blob Detection:** Utilize OpenCV's blob detection feature [7]. Configure the BlobDetector to detect white blobs (high-intensity areas) and set parameters to filter blobs by area and circularity, focusing on the near-square shape of IR markers. These settings help ignore non-marker objects in the scene.
3. **Obtain Region of Interest Matrix and Apply on Image:** After detecting sandbox boundaries, create a region of interest matrix (*roi_mat*). Set the sandbox area to 1 and other areas to 0. Use this matrix to filter the processed colored contours. Multiply the filtered image with *roi_mat* to isolate and display only the sandbox area in the final image. This step enhances the visual output's relevance and clarity.

2.2.3 Display – Generating Colored Contours from Depth Information

The purpose of this functionality is to visualize depth information more intuitively by applying color mapping to depth data. This approach helps in identifying features at various depths within a scene by representing them with different colors. The process involves several key steps: clamping the depth values to a specified range, normalizing these values, and then mapping them to a set of colors. Details are as follows:

1. **Normalizing Depth Values:** The depth values are first clamped to ensure they fall within a user-specified range. This step is crucial as it filters out depth values that

are either too close or too far, which may not be relevant to the analysis.

2. **Mapping to Color Indices:** These normalized values are then mapped to color indices. The mapping translates the normalized depth into a discrete index, which corresponds to a specific color in the colormap.
3. **Creating a Custom Colormap:** A custom colormap is created from the JET colormap [8], which contains a gradient of colors. Only a specific number of colors defined by n are selected to create this colormap, which helps in distinguishing between different depth values more effectively.
4. **Applying the Colormap:** Finally, the custom colormap is applied. This step maps the previously obtained color indices to actual RGB color values, which are then used to produce the final visual representation of the depth data.

The function `plot_depth_contour` demonstrates this process. It takes a depth array and parameters defining the depth range and the colormap specifics, and it returns an image where the depth information has been transformed into a visually comprehensible format using colors.

2.2.4 Concurrent Handling of Web and Display Using Multithreading

To enhance the responsiveness and efficiency of our system, which involves both displaying processed images and serving these images via a web interface, a multithreading approach is employed. This method allows the web server and the image display processes to operate simultaneously without blocking each other, ensuring smooth execution. The key to managing the shared resources, particularly the image frames, is the use of threading with a lock mechanism to handle concurrency safely.

The implementation involves two main threads: one for the Flask web server and another for the image processing task. A global variable, `current_frame`, is used to store the latest processed frame, which is accessed by both threads. This variable is protected by a threading lock, `current_frame_lock`, to prevent concurrent access issues such as race conditions.

1. **Web Server Thread:** Runs the Flask app, including the `/video_feed` route for streaming video frames. The `video_stream` function checks for the latest frame in the `current_frame` variable, locks it for thread safety, copies it, encodes it to JPEG, and yields it in a multipart HTTP response.
2. **Image Processing Thread:** Processes incoming video data to generate color-mapped contours or other outputs. Updates `current_frame` with the new image under the `current_frame_lock` to ensure safe access for the web server thread.

2.3 Projection Subsystem

2.3.1 Projector Selection

The projector in the projection subsystem is responsible for projecting the topography map onto the sand surface. Considering that topography is displayed on sand, the projection subsystem should be able to project the topography map with high luminance to overcome the strong diffuse reflection of sand. To mount the projector lower on one side of the sandbox for easier installation and to match the camera's field of view, the projection subsystem aims for a short-throw projector. For this, this AR sandbox uses a high luminance ultra short throw projector from EPSON with a resolution of 1024×768 pixels and a zoom factor of 0.31 in horizontal direction and 0.41 in vertical. It has a refresh rate of $60\,\mathrm{fps}$ and 2600 ANSI lumens. The projector connects with the processing unit through an HDMI cable, which is compatible with the Jetson development board.

One feature of this projector compared to the projector previously used in the AR sandbox project is that its projection is significantly off-axis, which means that the optical axis of the projector does not coincide with the center of the projected image. This feature is beneficial for the installation of the projector as it can be mounted lower on one side of the sandbox and still project the topography map onto the sand surface. However, this feature also brings challenges to the calibration of the projector-camera system, which will be discussed in the Projector-Camera System Calibration section.

2.3.2 Projector Auto Focusing

As the projection subsystem uses a short throw projector, a small distance change between the projector and the surface it projects to would bring a blurry image and require refocusing with the focusing lever on the projector. To automatically adjust the focus of the projector according to the height of the sand surface as the projector only allows for manual focusing through a lever shown in Figure 4, an auto-focusing subsystem is designed. It uses a stepper motor with a sliding table to control the projector's lever automatically. The stepper motor connects with the focusing lever on the projector through a customed connector on a sliding table shown in Figure 4. The stepper motor would be controlled by the processing unit and connected with the lever to change its position. The processing unit would use the depth information from the sensor subsystem to adjust the focus of the projector according to the height of the sand table.

The autofocusing system uses a 28HD2830 two-phase stepper motor with a 1.8° step angle and a holding torque of $0.07\,\mathrm{N \cdot m}$. It is installed on a linear sliding table with $50\,\mathrm{mm}$ stroke. The stepper motor connects with the processing unit through a drive board. The sliding table will go for $1\,\mathrm{mm}$ per rotation of the stepper motor. The drive board connects with the processing unit through GPIO pins and receives a PWM signal from the processing unit. The Jetson Nano has GPIO pins that can be used as PWM output. The acceleration and deceleration of the stepper motor can be controlled by the PWM signal. Figure 5 shows the relationship between distance and speed of the stepper motor.

For more precise control of the stepper motor's position, the motor is driven by a half-

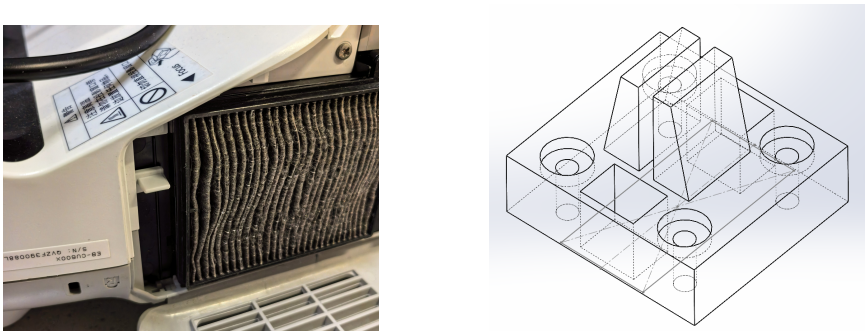


Figure 4: Focus Lever on EPSON EB-CU600X Projector and Connector Design

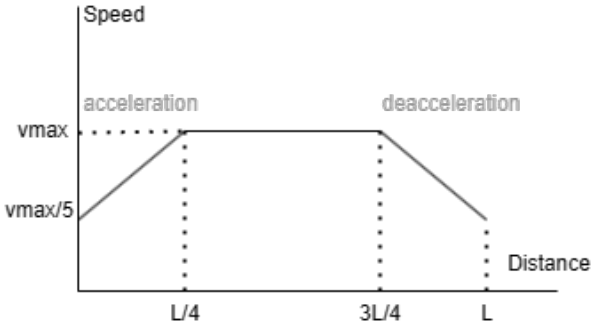


Figure 5: Stepper Motor Speed vs. Distance

step sequence, which is a sequence of 8 steps that the motor goes through to complete one full activation loop. Compared to a full-step sequence that only has 4 steps per activation loop, the half-step sequence provides higher precision and less vibration at low-speed operation [9]. With a $1.8°$ step angle and a sliding table with $1\,\mathrm{mm}$ lead, a half-step sequence could achieve $0.0025\,\mathrm{mm}$ accuracy in theory, which is enough for focusing lever control. By controlling the delay time between each step, the stepper motor can be controlled to move at different speeds. The motor function pseudocode is shown in Appendix D.

The drive board connects with the stepping motor through a 4-pin connector. L298N chip drives the stepping motor, which is a dual H-bridge motor driver integrated circuit. The L298N can drive up to 2A per channel and has a peak current of 3A per channel. The L298N has a thermal shutdown feature to prevent overheating and a short-circuit protection feature to prevent damage to the stepping motor. It is simple to use and provides low noise and high stability, which is suitable to control the focus lever.

The relationship between the position of the focusing lever and the distance between the projector and the focused surface is determined after acquiring data points. As the standard of being in focus is just determined by visual judgment, an approximated linear relationship is used for its simplicity and effectiveness. The distance is obtained from the sensor subsystem.

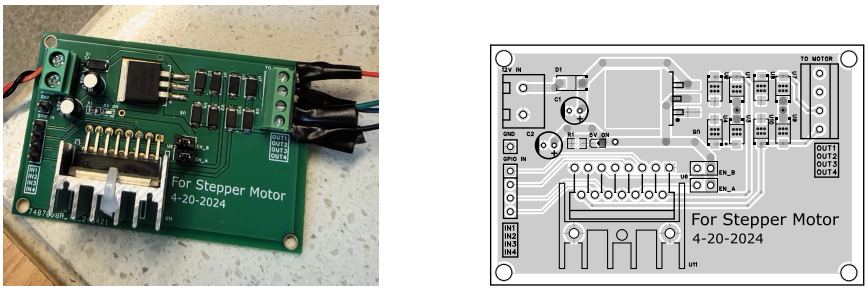


Figure 6: PCB Design of Stepper Motor Drive Board

2.3.3 Projector-Camera System Calibration

Projector-camera system calibration is a crucial step in the AR sandbox project. Compared with existing methods such as the KinectProjectorToolkit [10] that uses a printed checkerboard pattern, this AR sandbox uses a different method to calibrate the off-axis projector-camera system with just the projector. Considering that this AR sandbox aims for educational use and the printed checkerboard pattern may be lost or damaged, this would be a much better solution compared to the existing ones.

As in most AR applications, the relative position between the projector and the Kinect camera is fixed. To speed up the calibration process when applying a transformation to colored contours, this AR sandbox first uses a simple homography transformation to warp the colored contours to the projector's perspective. The equation used is defined as:

$$\begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} = H_{3\times3} \begin{bmatrix} x_c \\ y_c \\ 1 \end{bmatrix} \tag{1}$$

Here, $H_{3\times3}$ is the homography matrix that maps the camera coordinate system to the projector coordinate system. x_p and y_p are the pixel coordinates in the projector coordinate system, and x_c and y_c are the pixel coordinates in the camera coordinate system. The benefit of this method is that the processing speed is very fast and makes the AR sandbox more responsive as it does not consider the depth information of the sand surface. The drawback of this method is that it is not accurate and the final topography map offsets more than $5\,\mathrm{cm}$ with the sand surface when the sand surface has a more than $5\,\mathrm{cm}$ height difference. One homography applies when points in the image are on the same plane and multiple planes in the image require multiple homography, one for each plane [11]. Considering that the sandbox this project uses contains sand with a maximum height difference of around $10\,\mathrm{cm}$, this method is not suitable for this project. Therefore, the AR sandbox uses a more accurate method to calibrate the projector-camera system.

Our AR sandbox takes the depth information into account, which requires the intrinsic, extrinsic, and distortion parameters of both the camera and the projector for accurate pro-

jection. There are two main steps in the calibration process: camera calibration and projector calibration. The camera calibration considers the intrinsic parameters of the camera, such as focal length, principal point, and distortion coefficients. The projector calibration considers the intrinsic and extrinsic parameters of the projector, which is modeled as a camera. The detailed calibration process and an overview of the projector-camera system are shown in Figure 7. Some simplifications and assumptions made in the calibration process will be discussed in the following sections.

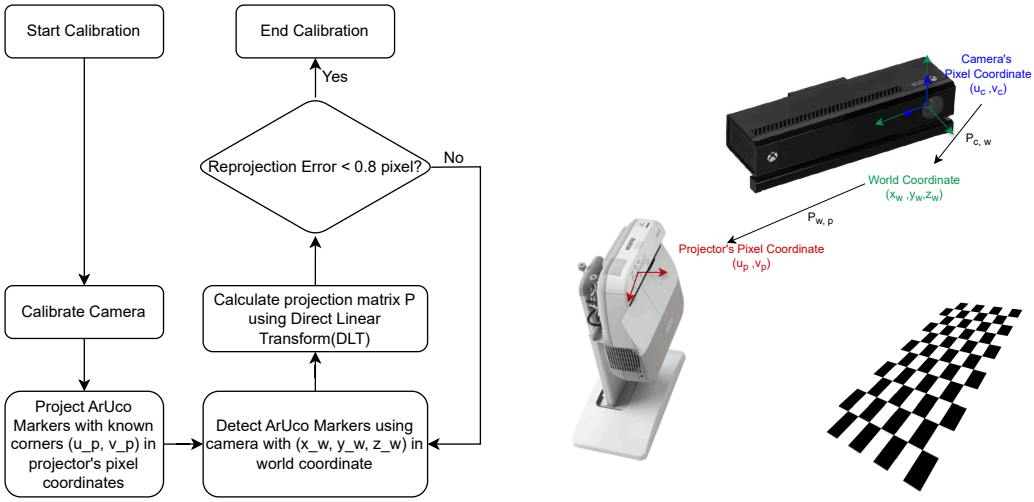


Figure 7: Flowchart of Projector-Camera System Calibration and Overview of Projector-Camera System

Camera Calibration Camera calibration is the process of estimating the intrinsic parameters of the camera, such as focal length, principal point, and distortion coefficients. These parameters are crucial for converting the pixel coordinates of the camera to real-world coordinates. The relation between 2D data points in the camera coordinate system and 3D data points in the world coordinate system is given by the following equation according to the pinhole camera model proposed in Zhang's paper [12]:

$$Z_c \begin{bmatrix} x_c \\ y_c \\ 1 \end{bmatrix} = K_c \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{2}$$

$$K_c = \begin{bmatrix} \frac{1}{dx} & 0 & C_x \\ 0 & \frac{1}{dy} & C_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_x & 0 & 0 & 0 \\ 0 & f_y & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{3}$$

By defining the world coordinate system's origin to be the same as the camera's pixel coordinate system and aligning the pixel coordinate's u-v plane with the x-y plane of the world coordinate, the rotation matrix R is the identity matrix and the translation vector t is a zero vector. For the distortion coefficients, the camera model uses a standard Brown-Conrady model, which is a radial-tangential distortion model. The equation describing distortion is in Equation 4 with tangential and distortion terms [13]. The intrinsic parameters and distortion parameters of the camera are provided in a paper [14] and source code of a library for Kinect [15] as shown below:

cx	cy	fx	fy	k1	k2	k3	p1	p2
254.878	205.395	365.456	365.456	0.0905474	-0.26819	0.0950862	0.0	0.0

Table 1: Camera Parameters

$$\begin{bmatrix} x_{\text{distorted}} \\ y_{\text{distorted}} \end{bmatrix} = \begin{bmatrix} x_{\text{undistorted}} \\ y_{\text{undistorted}} \end{bmatrix} \times (1+k_1 r^2+k_2 r^4+k_3 r^6) + \begin{bmatrix} 2p_1 x_{\text{undistorted}} y_{\text{undistorted}} + p_2(r^2 + 2x^2_{\text{undistorted}}) \\ p_1(r^2 + 2y^2_{\text{undistorted}}) + 2p_2 x_{\text{undistorted}} y_{\text{undistorted}} \end{bmatrix} \tag{4}$$

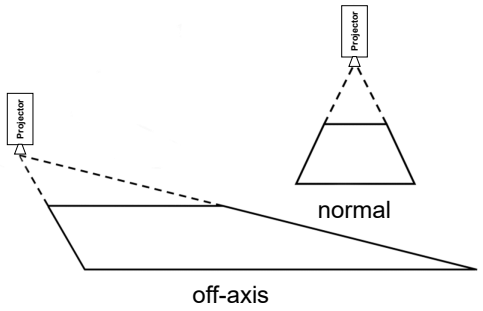


Figure 8: Comparison between On-axis and Off-axis Projection

Projector Calibration Projector calibration is the process of estimating the intrinsic and extrinsic parameters of the projector, which is modeled as a camera. Considering that this projector exhibits strong off-axis projection as shown in Figure 8, the intrinsic matrix K_p is different from the camera's intrinsic matrix K_c. The relation between 2D data points in the projector coordinate system and 3D data points in the world coordinate system is given by the following equation:

$$Z_p \begin{bmatrix} u_p \\ v_p \\ 1 \end{bmatrix} = K_p \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{5}$$

where K_p is the intrinsic matrix of the projector, R is the rotation matrix, and t is the translation vector. Considering that the origin of the world coordinate is fixed on the camera, there is no need to calculate the intrinsic, rotation matrix, and translation vector separately. Thus, the calibration process chooses to directly estimate the projection matrix using the Direct Linear Transformation (DLT) algorithm.

For a 3-by-4 projection matrix $P_{w,p}$, 12 unknown parameters need to be solved, which needs at least 6 corresponding points between the world coordinate system and the projector coordinate system. These points are obtained by projecting an ArUco [16] marker board onto a surface and detecting the corners of the markers in the camera coordinate system. The ArUco marker board is a planar board with a set of markers with known dimensions and IDs, which does not require the camera to capture the whole board. The world coordinates of detected corners of the markers are then calculated with the previously calibrated depth camera. The following equation is used to calculate the projection matrix $P_{w,p}$:

$$Z_p \begin{bmatrix} u_p \\ v_p \\ 1 \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{6}$$

To solve for the parameters p_{ij} in $P_{w,p}$, the Direct Linear Transformation (DLT) [17] transforms the equation into a linear system $L \cdot p = 0$ as shown in the following equation:

$$L = \begin{bmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -u_1X_1 & -u_1Y_1 & -u_1Z_1 & -u_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -v_1X_1 & -v_1Y_1 & -v_1Z_1 & -v_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & -u_nX_n & -u_nY_n & -u_nZ_n & -u_n \\ 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -v_nX_n & -v_nY_n & -v_nZ_n & -v_n \end{bmatrix} \tag{7}$$

$$p = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} & p_{21} & p_{22} & p_{23} & p_{24} & p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix}^T \tag{8}$$

The projection matrix $P_{w,p}$ is then estimated by solving the linear system $L \cdot p = 0$ using the singular value decomposition (SVD) method. The following equation calculates the reproduction error of the projection matrix:

$$\text{Reproduction Error} = \sqrt{\frac{\sum_{i=1}^{n} (u_p - \hat{u}_p)^2 + (v_p - \hat{v}_p)^2}{n}} \tag{9}$$

where u_i and v_i are the pixel coordinates of the ith point in the projector coordinate system, and $\hat{u}_i$ and $\hat{v}_i$ are the pixel coordinates of the ith point in the projector coordinate system calculated by the projection matrix $P_{w,p}$. The projection matrix $P_{w,p}$ is then used to project the colored contours to the projector coordinate system. The reproduction error of the projection matrix is required to be less than $0.8\,\text{pixel}$ for the calibration to be considered successful.

Considering that the projector has a built-in distortion correction feature that cannot be turned off and performs correction well, the calibration process ignores the distortion coefficients of the projector.

Applying Projector-Camera System Calibration There are two steps to apply the projector-camera system calibration. First, the sensor subsystem provides the depth information of the sand surface. The depth information is then converted to real-world coordinates based on the calibrated camera. Second, the projector's projection matrix projects the real-world coordinates to the projector coordinate system and produces the final topography map.

When converting from the camera coordinate system to the projector coordinate system, a vectorization method is proposed. The open source library called "libfreenect2" [5] provides a function `getPointXYZ` that can only perform the conversion for one point at a time. To speed up the process, the function is vectorized to convert multiple points at a time. The code is shown in Appendix F.

When projecting the real-world coordinates to the projector coordinate system, the projection matrix is used. The code is also vectorized to speed up the process. The vectorized code is shown in Appendix G. Vectorizing the function shortens the processing speed by 20 times. The final topography map is then projected onto the sand surface. The final topography map is shown in Figure 9.

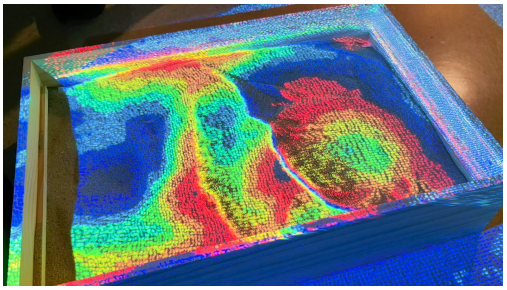


Figure 9: Topography Map Projected onto Sand Surface

2.4 Powering Subsystem

The powering subsystem is responsible for providing power to the sensor, processing, and display subsystems. The powering subsystem should be able to provide power to the sensor, processing, and display subsystems with sufficient power and voltage. The

powering subsystem should be able to provide power to the sensor, processing, and display subsystems with high reliability and high safety.

The Kinect sensor and projector take $220\,\mathrm{V}$ AC power, while the processing unit takes $5\,\mathrm{V}$ DC power and the drive board of the stepping motor takes $12\,\mathrm{V}$ DC power. One power converter safely converts $220\,\mathrm{V}$ AC power to $5\,\mathrm{V}$ DC power and provides $20\,\mathrm{W}$ at maximum for the processing unit. The Jetson Nano takes $5\,\mathrm{V}$ DC power and $2\,\mathrm{A}$ current at maximum. The stepping motor takes $12\,\mathrm{V}$ DC power and $2\,\mathrm{A}$ current at maximum. One power converter safely converts $220\,\mathrm{V}$ AC power to $12\,\mathrm{V}$ DC power for the drive board of the stepping motor. The power consumption of these two parts is within the power supply's capacity. A power strip with high current and temperature protection connects the Kinect sensor, projector, stepping motor, and processing unit to one plug and provides at most $2500\,\mathrm{W}$ power to the whole system. It is safe and reliable to use.

In the auto-focusing subsystem, a UZ1085 LDO (low-voltage dropout) voltage regulator is used to convert the $12\,\mathrm{V}$ DC power to $5\,\mathrm{V}$ DC power for chips on the PCB board and the stepper motor. The LDO voltage regulator can provide a stable $5\,\mathrm{V}$ DC power and up to $2\,\mathrm{A}$ current, which satisfies the need of the stepper motor. The LDO voltage regulator has a thermal shutdown feature to prevent overheating, short-circuit protection, and an overcurrent protection feature to prevent damage to the processing unit. It is simple to use and provides low noise and high stability voltage output.

2.5 Database and Wireless Display Subsystem

2.5.1 Selection

Database The database serves as a centralized repository for storing frame images that are captured as requested by the user from the depth camera. Images captured by the depth camera contain important depth information, making them necessary for calibration, segmentation, and contour painting. By structuring these images within a database, retrieval, and management of the captured frames are ensured, and seamless access for analysis and processing on the back end of the project becomes available for developers. It is not only useful for debugging back-end programs but also important to the overall user experience for the project. As a well-designed database, scalability is ensured so that the system can accommodate increasing data loads without experiencing degradation in responsiveness or reliability.

SQLite [18] was chosen as the database solution for this project due to several distinct advantages it offers over alternatives such as PostgreSQL [19] and MySQL [20]. First of all, SQLite is a simpler self-contained database that is lightweight and does not require additional separate server processes to be running. Secondly, SQLite can smoothly integrate with the Python environment in which most back-end programs of this project run. Developers can perform CRUD (Create, Read, Update, Delete) operations with minimal boilerplate code by using Python with SQLite. As the database in this project is designed to perform only read, write, and delete, SQLite provides the perfect solution.

Wireless Display The integration of wireless transmission capabilities within the web page enables real-time streaming of the camera feed to end-users who are using either mobile devices or PCs. Wireless transmission eliminates the need for physical connections, offering flexibility in accessing the camera feed from diverse locations and devices. Through the web page, users can initiate live streams, navigate through archived frames, and enjoy the view of colored contours. As the overseer of the sandbox, parents can monitor their children's behavior even if they are not near the sandbox, providing extra safety precautions. To integrate into different devices, the web page layout is designed to be responsive as every element can scale according to the size of the window to provide a comfortable user experience everywhere. [21]

Flask [22] paired with pure HTML and CSS offers a lightweight, flexible, and efficient approach to web page development that aligns well with the requirements and constraints of the project. This choice stands out compared to the other alternatives that are available for this project. Being a lightweight micro-framework, Flask provides the essentials for building web applications without imposing unnecessary complexity. By leveraging pure HTML and CSS alongside Flask, it is easy to maintain a simple and straightforward development workflow, and the debugging process can be a lot easier. SCSS [23] as the more advanced version of CSS is discarded because the Linux system can not handle SCSS well and some web page features may be inconsistent on different user devices. Javascript is widely used to define button movement on web page development. Additionally, pure HTML and CSS offer complete flexibility and make it possible to develop a simple but fully functional responsive web page layout that can adapt to any window size well. On the Javascript part, an alternative is to use the Babel [24] version of Javascript so that many interesting visual effects can be realized. This approach is also discarded as Babel requires additional installation on the Linux system that may not support full customization to simplify the structure of this project.

2.5.2 Design Details

Database and wireless display are deeply connected with each other in this project, and the design details of both shall be discussed together. As shown in Figure 10, overall 10 Python Functions and 3 Javascripts for buttons have been implemented to realize both database setup and web page layout.

2.6 Structure Subsystem

2.6.1 The Main Part of Sandbox

The main part of the sandbox is dedicated to the containment of sand and constitutes the central and largest portion that interconnects with all other segments. This integral section comprises the main body of the sandbox and a removable lid as shown in Figure 11. With the lid in place, the sandbox assumes external dimensions of 600x400x150 millimeters, while the internal cavity measures 578x378x128 millimeters. The lid is a composite structure made of two boards that are pivotally affixed using hinges, enabling a full 180-degree swivel. The interface between the lid and the main body is established through

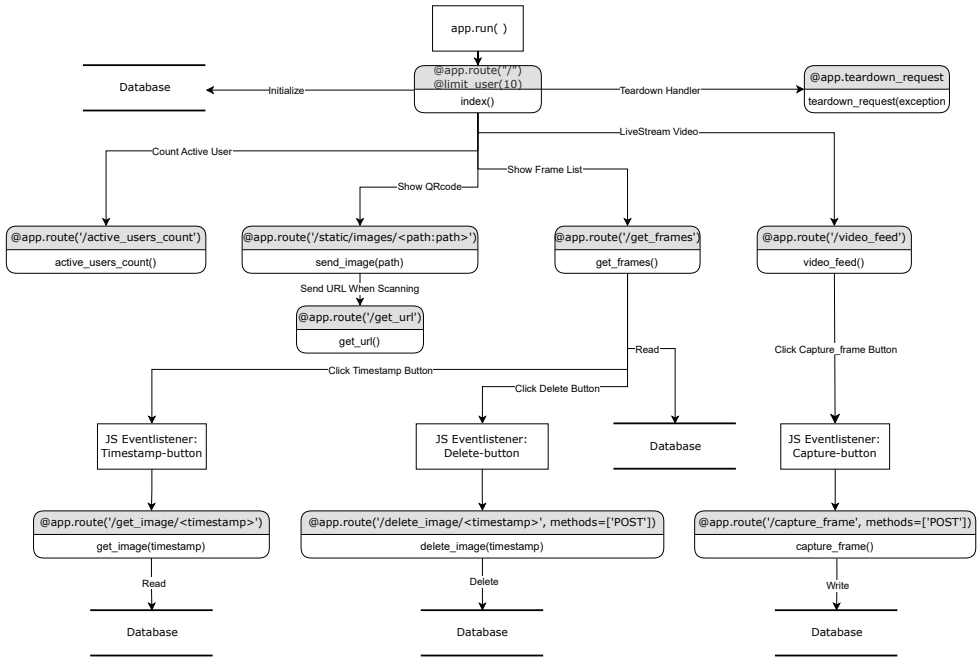


Figure 10: Flowchart on database and wireless transmission subsystem

a pliable silicone sheet, which guarantees an unfettered 270-degree movement. The design rationale for this feature is to allow the lid to be manipulated with ease, enabling it to be stored flush against the side of the sandbox during utilization. This configuration ensures unobstructed access and convenience for the user around the entire perimeter of the sandbox.

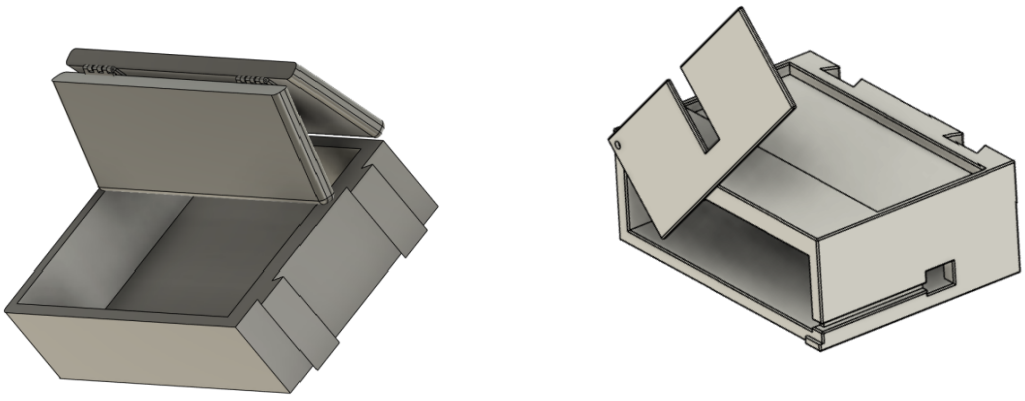


Figure 11: The CAD model of the main part and separable of sandbox

The material used for this section is pine wood. The box body is constructed by join-

ing pine boards with a thickness of $12\,\mathrm{mm}$. The selection of materials for this part was greatly influenced by budgetary constraints. The ideal material for fabrication would be PLA (Poly Lactic Acid) plastic, which offers higher strength properties. However, calculations and simulation validations have confirmed that the structural integrity of pine wood is sufficient to support the weight of the sand, while also making the sandbox more portable.

2.6.2 The Separable Part

The overall structure of the separable part is depicted in the diagram. Its function is to house the projector, connect sensors, and accommodate all electronic devices. The top of this part features a recess measuring 160x380x10 mm, slightly larger than the projector's base, allowing the projector to be securely inserted. The sensors will be mounted on a pan-tilt mechanism and secured by a rod to the upper edge of the separable part. The pan-tilt mechanism enables free rotation, facilitating the convenient use of the sensors. The rod is made of aluminum alloy and is adjustable in length. The hollow center of the separable part is designed for storing the processing system and power supply system, ensuring that the entire system can be powered by a single wire extending from the box. The open side will be covered with a rotating plate to maintain a sleek, integrated appearance. The plate features a rectangular opening to aid in positioning the clamping point of the rod connecting to the sensors. The connection between the main body of the sandbox and the separable part is achieved using a dovetail joint structure. This design is based on the reason that when the sandbox is placed horizontally, the joint will not be subjected to significant force, but it must ensure stability. The simple structure also allows for easier assembly and disassembly. The significance of this section lies in the ability to use our electronic devices independently. This means that our electronic devices are not dependent on the original sandbox but can be adapted to various scenarios and different sand tables, increasing the flexibility of product use.

The material for the separable part is PLA, produced by 3D printing. In actual production, more cost-effective methods such as injection molding can be considered. The strength of the PLA-made separable part fully meets safety requirements. Cheaper materials with slightly lower strength could also be suitable alternatives.

2.6.3 The Recovery Subsystem

The primary function of this system is to facilitate the rapid and convenient re-leveling of used sand tables. To avoid complicating the use and maintenance of the sandbox with electric equipment, a special shovel with a unique structure was designed to achieve this purpose. The structure of the shovel is depicted in the diagram. Its mechanism operates by pushing excess sand into the shovel when thrust forward, where it is stored in the hollow segment in the center, and upon returning, the stored sand fills the gaps through the bottom slit. By placing the shovel on the guide rail as shown, the lower end of the shovel is set at the original height of the sand surface in the sandbox. This design allows the user to restore the sand surface simply by pushing the shovel. The edges of the shovel have

been blunted after optimization to prevent accidental injury to the user. Furthermore, the manual operation of the recovery system eliminates the risk of user harm, making it decidedly more convenient and safe.

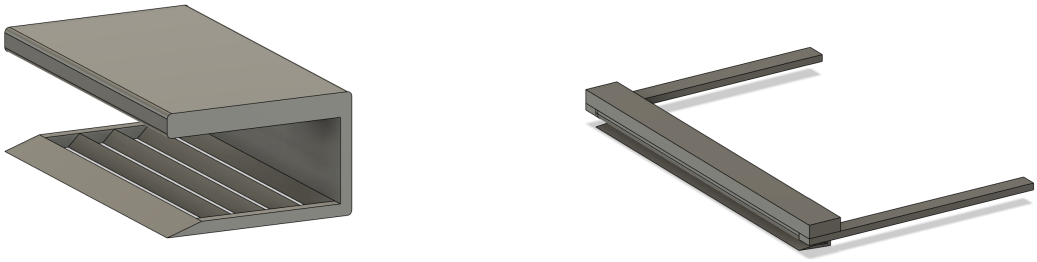


Figure 12: The CAD model of the shovel and the recovery system

The shovel is fabricated using PLA material via 3D printing, and in actual production, more cost-effective methods such as injection molding can be considered. The guide rail maintains the same pine wood material as the main body of the sandbox, which is rougher than anticipated in the design phase. The friction can be reduced by applying polytetrafluoroethylene (PTFE) tape to the surface of the rail.

2.6.4 The Mobility System

The purpose of this section is to make the entire AR sandbox portable. The weight of all the structures of the AR sandbox, including the electronic equipment, approaches $30\,\mathrm{kg}$. Manual handling is impractical and also increases the risk of damage to the device. The mobility system consists of two parts: a metal frame and wheels. The metal frame is welded from angle steel with a thickness of $3\,\mathrm{mm}$ and a width of $50\,\mathrm{mm}$, and it is reinforced with one longitudinal and three transverse cross braces at the bottom to enhance strength and stability. The dimensions of the metal frame are 1500x400 millimeters, taking into account the need to move the projector for focusing. The material used for this part is 304 steel, which has a very high hardness. There are a total of six wheels: the four outer wheels are swivel casters with brakes, and the two in the middle are ordinary swivel casters. The wheels are made of nylon and are 2 inches in size. This ensures that the system can be fixed in place during use and will not move easily. Testing has shown that the mobility system can support a weight of $200\,\mathrm{kg}$ without deformation, which fully meets the usage requirements.

3 Verification

A detailed requirement and verification table is in Appendix B.

3.1 Sensor Subsystem

By logging the frame rate and latency when fetching RGB, IR, and depth frames from the Kinect camera, the performance of the sensor system can be verified. The frame rate of the camera is higher than $30\,\mathrm{fps}$ with latency less than $20\,\mathrm{ms}$. The log is shown in Figure 13.

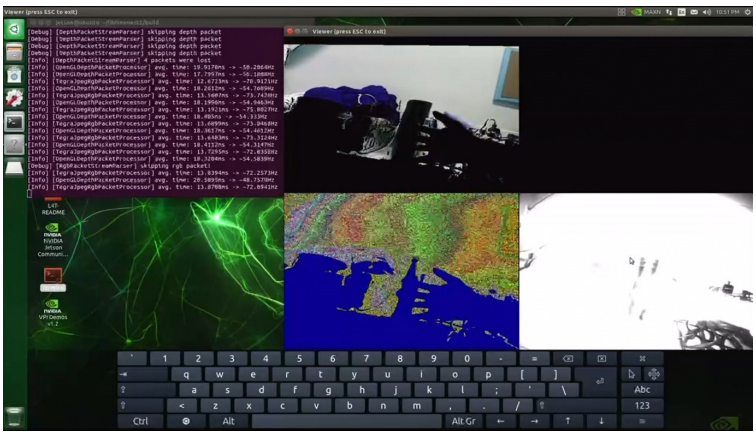


Figure 13: Log of Sensor System

3.2 Verification of Processing System - Colored Contour Display

The display system is designed to show 10 different colors, each representing a $5\,\mathrm{cm}$ height change. To verify this feature, a tilted whiteboard was placed on the ground and observed under the colored contour display. The verification results demonstrated that the system could reliably differentiate the $5\,\mathrm{cm}$ height changes with an error margin of $\pm 0.3\,\mathrm{cm}$. Furthermore, when shaping the sand in the sandbox, the colored contours accurately showed the ridges and valleys, confirming the system's ability to display different topographical features.

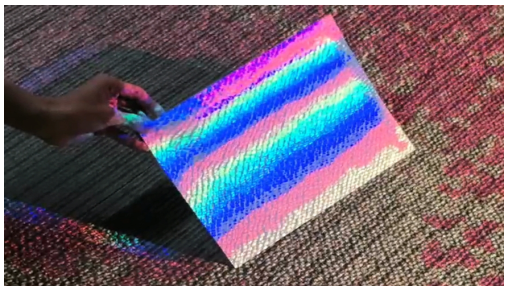


Figure 14: Color Display for Different Height Levels

Figure 15: Colored Contours Verification on Sand With and Without IR Markers

3.3 Verification of Processing System - Sandbox Edge Detection

To test the sandbox locating functionality, several IR reflective markers were placed on the sand surface, and the locate button was pressed on the web interface. The display region was then outlined by a polygon formed by these markers with minimal deviation. During testing, the average error distance from each corner of the polygon to the center of the IR reflective marker was found to be $1.2\,\mathrm{cm}$. When changing the positions of the markers, the display region is updated correctly to match the new positions, maintaining the same level of accuracy.

3.4 Projection Subsystem

3.4.1 Auto Focusing

Although the current driving capability of the GPIO pins on Jetson Nano is only around $0.5\,\mathrm{mA}$, it is sufficient to drive the H-bridge chip and control the stepper motor. Figure 16 is the output of one of the GPIO pins on the Jetson development board, which shows acceleration, steady speed, and deacceleration periods.

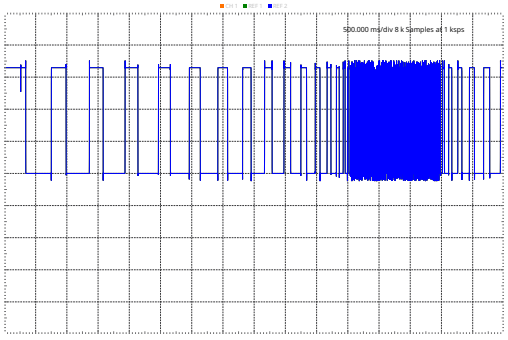


Figure 16: PWM Output of Jetson Nano

The stepper motor can control the position of the sliding table correctly and accurately. The sliding table can move to the desired position and stop at the correct position. The motor can also move in the opposite direction. This is verified by measuring the location of the slide table with a ruler. The error of the movement is about $0.5\,\mathrm{mm}$.

3.4.2 Projector-Camera System Calibration

The verification of the projection system is first achieved by checking the reproduction error of the projected image. In Figure 17, 19 corner points are detected as valid data points and used for projection matrix estimation. Below is the projection matrix estimated with a projection error of 0.7941067124646294 using these points:

$$\begin{bmatrix} -2.86547238e-01 & 2.77225866e-01 & -5.01825389e-01 & 3.49394704e+02 \\ -1.98248169e-02 & 6.11044295e-01 & -5.56395603e-01 & 4.34260722e+02 \\ -3.97751643e-05 & 7.87623720e-04 & -1.39620048e-03 & 1.00000000e+00 \end{bmatrix}$$

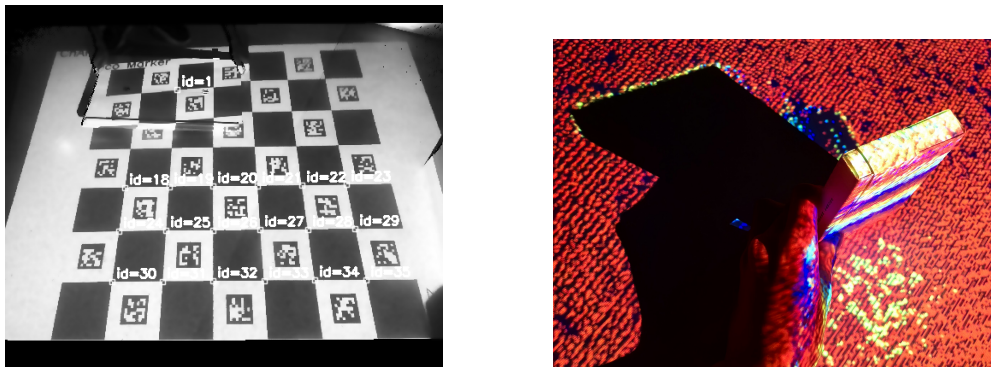


Figure 17: Calibration of Projection System and Results

The effectiveness of the calibration process is also verified by checking the alignment of an elevated surface with the topography projected on it, considering that this system deals with the alignment of projected images with objects at different heights. Here a box with a height of $11\,\mathrm{cm}$ is placed in front of the projector. The error of the alignment is about $0.6\,\mathrm{cm}$ for both the top and bottom of the box. The result is shown in Figure 17.

The speed of the calibration process is verified by timing the calibration process. The calibration process takes on average $0.17\,\mathrm{s}$ to complete over 50 consecutive runs. The time taken processing one depth frame is $202.45\,\mathrm{ms}$ in total, which is within the requirement of less than $1\,\mathrm{s}$ latency.

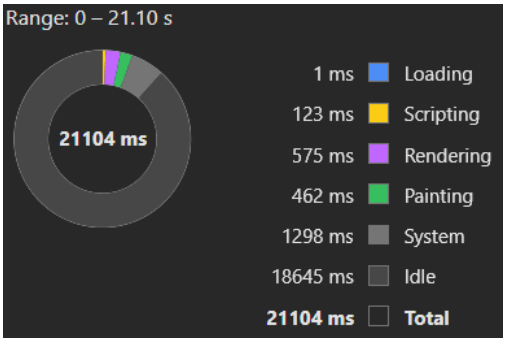
3.5 Database and Wireless Display Subsystem

For the database as well as the web page, testing is rather intuitive.

To test database functionalities, which specifically are read, write, and delete, another `DBreader.py` (See Appendix K) is written so that directly viewing contents of the database is available. All contents will be outputted into a folder named output, and by visually checking the number of frames captured as well as the contents of each figure, it is easy to determine whether Writing into the database is correct or not. Reading functionality is checked by clicking the timestamp button on the web page after capturing frames.

If the frame correctly shows up on the desired area, then reading functionality is good. There are two ways to check whether delete is implemented correctly or not. First of all, by clicking the delete button beside the timestamp button, the corresponding line shall be erased from the page. Secondly, `DBreader.py` is used before and after the delete button is clicked, and if the number of frames decreases, then the delete functionality is checked. There is a maximum amount of frames that can be stored in the database. After reaching the hard cap, the newest frame will replace the most outdated frame. Using `DBreader.py` twice to access the contents of the database can decide whether this functionality is implemented correctly or not.

The web page is designed to be responsive so that every element can comfortably scale up and down based on the display devices of users. If the user opens the web page on mobile devices, functional buttons including `Capture frame`, `timestamp` and `delete` shall work just like on PC. The positions of every element shall move according to the window size to provide a comfortable and stable user experience. The top navigation bar shall be fixed on top and when scrolling down the page it is always visible. On the menu list in the navigation bar, users can be able to click the titles and jump immediately to corresponding areas. For example, by clicking the check button in the menu list, the user shall be able to jump to the area where the frame list and the stored image display area are. The QR code shall have consistency so that whether the user scans from the screen of a PC or a phone the web page can jump to this web page correctly. The livestream area shall display the contours correctly and constantly. The `Back to Top` button on the bottom of the page shall correctly scroll the page back to the top after being clicked.



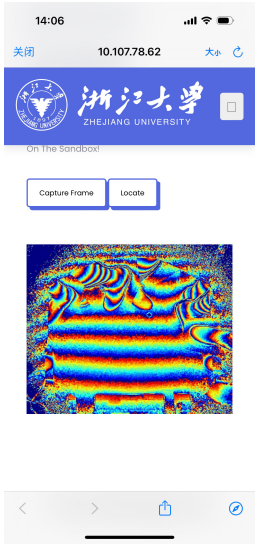


Figure 18: Performance Summary and completed web page testing on iPhone, when using the web page. Rendering, scripting, and painting processes each take less than $600\,\mathrm{ms}$ (Rendering being the slowest), which is fast enough.

3.6 Structure Subsystem

When conducting a tolerance analysis of sandbox mechanical structures, it is considered necessary to investigate whether structural errors and the influence of different materials on product properties are acceptable. Models using ABS plastic [25] are fully capable of meeting the functional needs of AR sandboxes, which has been confirmed. The samples used were produced by 3D printing and can reproduce CAD models almost perfectly. However, in actual production, it is difficult to achieve such precise production. Whether it is the production method of 3D printing or the choice of ABS materials, it will increase the cost of the product. Therefore, the tolerance analysis in this study focuses on whether structural errors and the influence of different materials on product performance are acceptable.

First of all, in actual production, the high-probability production method is injection molding. This is suitable for rapid, large-scale production and can effectively reduce costs. However, the biggest drawback of the injection molding process is that the accuracy is not as good as 3D printing. According to the data, the typical tolerance for injection molding is usually $\pm 0.1\,\mathrm{mm}$, while the very strict tolerance is $\pm 0.025\,\mathrm{mm}$. To ensure that the results are convincing, the $0.1\,\mathrm{mm}$ tolerance was enlarged tenfold in the analysis. This means that the thickness of each part of the model will be reduced by $2\,\mathrm{mm}$.

Second, although acrylonitrile-butadiene-styrene (ABS) is a good choice, the significantly lower price of polystyrene (PS) [26] and polypropylene (PP) [27] for the same weight can undoubtedly reduce production costs. It is worth noting that the injection molding process is also suitable for common materials such as polymethyl methacrylate (PMMA) [28], nylon polyamide (PA), polycarbonate (PC) [29], polyethylene (PE) [30], thermoplastic elastomer (TPE) [31] and thermoplastic polyurethane (TPU) [32]. Some of these materials are too expensive (for example, PE is almost 1.5 times more expensive than ABS), while others have obvious flaws in structural strength (for example, TPE is too soft). Therefore, we do not consider these materials.

For PP and PS, the process of filling glass fibers is ignored to increase their strength. Although this method is often used in everyday production, it reduces the generality of our analysis. In summary, this study will simulate models of thinner PP and PS materials to confirm that the design has very good tolerances. The selected analysis part is the separable part with the most cavities, the highest pressure, and the most susceptible to external interference. If this part is performing well, it shows that other parts are also working effectively.

Starting with the control group, the representative was considered the perfect model, and this study added the equivalent of $5\,\mathrm{kg}$ of force into the groove to simulate the weight of the projector used. The force applied to the edge consists of two parts. First, a camera will be placed in this position; Secondly, after analysis, it is determined that this position is the most easy to deform, so it is the most suitable point for simulating external forces. As shown in Figure 25 and 26, a downward force of $500\,\mathrm{N}$ is applied to this position to simulate the worst-case scenario. The study found that in this case, the pressure and deformation of the separable part are still negligible.

4 Cost and Schedule

The labor cost is calculated based on the assumed salary of $20 per hour per person. The time spent on the project for each person is 25 hours per week. The total duration for this project is 10 weeks. For components, non-standard parts like the structure part is produced in the school laboratory for free. The PCB boards are printed in JLC for free. The cost of consumables would not be included in the cost estimate as it is inconsistent.

name	manufacturer	part #	quantity	cost (CNY)
JETSON NANO B01 with $5\,\mathrm{V}, 4\,\mathrm{A}$ power supply	Yahboom	B01	1	1199
Wooden Box	Fanguo	$60\,\mathrm{cm}$ * $40\,\mathrm{cm}$ * $15\,\mathrm{cm}$	1	70
Acrylic Hinge	Acrylic Accessory Store	$25\,\mathrm{mm}$ * $35\,\mathrm{mm}$	4	12
EPSON CU600X Ultra Short Throw HD Projector	EPSON	CU600X	1	584.5
Microsoft Kinect V2 camera	Microsoft	V2	1	770
IR Reflective Tape	3M	$1\,\mathrm{m}$ * $2\,\mathrm{cm}$	1	12.79
RTL 8822CE Wireless Network Card	REALTEK	8822CE	1	25
Brushless Fan	Delta Electronics	AFB0412VHA	1	5
Superfine River Sand	TIANSHISHUIZU	unknown	3*$5\,\mathrm{kg}$	53.4
Screw Type Terminal Block	KEFA	C474881	4	2.5
L298N	STMicroelectronics	L298N	1	3.2
Desktop retractable stand	Chenxin Digital Accessories	1/4 inch - external teeth	1	23.4
Linear Sliding Table with Stepper Motor	Ouli Transmission	28 T6*1-50 mm	1	85
Wiring Board	Hanhu Electronics	$2.8\,\mathrm{meter}$ cord	1	30.8

Table 2: Cost of the Components

Total Labor cost = $20 \times 25 \times 10 \times 4 = \$20000$

Total Component cost = 2876.59 in CNY

A detailed table of schedule is shown in Appendix A.

5 Conclusion

5.1 Accomplishments

This redesigned AR sandbox has a faster response, easier installation, and safer design than existing solutions. It realizes a less than 1-second latency of projecting topography, $0.6\,\mathrm{cm}$ topography projection offset, accurate sandbox locating with IR markers, multi-threading software backend, sturdy structure with detachable design, sand surface recovery design, reliable database and convenient wireless control. It is an easy-to-use AR sandbox that has a responsive topography projection, safe and convenient installation, and the ability to adapt to different shapes of sandboxes.

5.2 Uncertainties

Here are some uncertainties that may affect the performance of the AR sandbox and discussion of the reliability of the system:

- The accuracy of the calibration process is affected by the noise in the camera image and the quality of the projected calibration board captured by the camera. The calibration process may fail if certain points on the board are not detected correctly or certain parts of the image are overexposed or underexposed. However, using ChArUco board with at most 36 corners being detected in the image, the calibration process has a high success rate and the reprojection error is within an acceptable range. To further improve the accuracy of the calibration process, more complex models of lens and nonlinear iterative optimizations can be used to improve the calibration process.
- The calibration process is affected by the working distance of the camera. The calibration process may fail if the camera is less than $0.5\,\mathrm{m}$ away from the calibration board. However, the design of the sandbox ensures that the camera is always at least $0.5\,\mathrm{m}$ away from the calibration board.
- The accuracy of the depth information from the camera is affected by sunlight and other light sources according to modeling of Kinect camera's noise [33]. The accuracy of depth information would be largely affected if the light source is too strong. However, the depth information is processed with a median filter to filter out some of the noise and ensure the depth information is stable under normal indoor lighting conditions.
- The projected image is not stable even when the sand is not moving. This image flickering may affect the user experience. The main reason for this is the inaccurate calibration of the projector-camera system and the noise from depth information. This is improved by applying a filter to the depth information and the projected image, but the flickering is still noticeable.

5.3 Future Work

It can participate in education and make kids interested in geology in an interactive way. This AR sandbox would be useful in other fields such as urban planning, water management, and environmental protection, and become a useful tool for education and research in the future.

In the future, this AR sandbox can be further improved in the following aspects:

- The AR module of the sandbox could be improved to be more compact and easy to move.
- The calibration process could be automated to make it easier for users to set up the sandbox.
- The calibration process could be improved to be more accurate by considering more complex lens distortion models.
- The software could be improved to support more features such as water simulation, landslide simulation, interaction with human hands, and more.

5.4 Ethical Issues

High-luminance projector would be energy-consuming. The user may forget to turn off the AR sandbox after use. This would bring potential environmental issues. As stated in the ACM Code of Ethics and Professional Conduct, "human well-being requires a safe natural environment. Therefore, computing professionals should promote environmental sustainability both locally and globally"[34]. It is important to lower the power consumption during use and automatically detect the leave of users.

Also, because people with different skin colors may have different reflection rates, the depth camera may not work well for people with dark skin. This would be a potential ethical issue and we need to consider it seriously, as mentioned in the IEEE Code of Ethics that "to treat all persons fairly and with respect, and to not engage in discrimination based on characteristics such as race, religion, gender, disability, age, national origin, sexual orientation, gender identity, or gender expression"[35]. Functionality testing with people with different skin colors needs to be implemented.

References

[1] O. Kreylos, *Oliver Kreylos' research and development homepage - Augmented Reality Sandbox*. [Online]. Available: https://web.cs.ucdavis.edu/~okreylos/ResDev/SARndbox/.

[2] F. Wellmann, S. Virgo, D. Escallon, *et al.*, "Open AR-Sandbox: A haptic interface for geoscience education and outreach," *Geosphere*, vol. 18, no. 2, pp. 732–749, Feb. 2022, ISSN: 1553-040X. DOI: 10.1130/GES02455.1. eprint: https://pubs.geoscienceworld.org/gsa/geosphere/article-pdf/18/2/732/5576191/732.pdf. [Online]. Available: https://doi.org/10.1130/GES02455.1.

[3] A. Corti, S. Giancola, G. Mainetti, and R. Sala, "A metrological characterization of the kinect v2 time-of-flight camera," *Robotics and Autonomous Systems*, vol. 75, pp. 584–594, 2016, ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2015.09.024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889015002195.

[4] L. Yang, L. Zhang, H. Dong, A. Alelaiwi, and A. E. Saddik, "Evaluating and improving the depth accuracy of kinect for windows v2," *IEEE Sensors Journal*, vol. 15, no. 8, pp. 4275–4285, 2015. DOI: 10.1109/JSEN.2015.2416651.

[5] L. Xiang, F. Echtler, C. Kerl, *et al.*, *Libfreenect2: Release 0.2*, version v0.2, Apr. 2016. DOI: 10.5281/zenodo.50641. [Online]. Available: https://doi.org/10.5281/zenodo.50641.

[6] N. Corporation, *Nvidia jetson nano*. [Online]. Available: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-nano/product-development/.

[7] S. Mallick, *Blob detection using opencv (python, c++)*. [Online]. Available: https://learnopencv.com/blob-detection-using-opencv-python-c/.

[8] Itseez, *Colormaps in opencv*, 2024. [Online]. Available: https://docs.opencv.org/4.x/d3/d50/group__imgproc__colormap.html.

[9] I. Virgala, M. Kelemen, A. Gmiterko, and T. Lipták, "Control of stepper motor by microcontroller," *Journal of Automation and Control*, vol. 3, no. 3, pp. 131–134, 2015.

[10] G. Kogan, *Kinect projector toolkit for image mapping and calibration.[online, github].(july 2014)*, 2014.

[11] S. Mallick, *Homography examples using opencv (python / c ++)* —, Jan. 2024. [Online]. Available: https://learnopencv.com/homography-examples-using-opencv-python-c/.

[12] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000. DOI: 10.1109/34.888718.

[13] D. Brown, *Close-range camera calibration photogeom*, 1971.

[14] Zhang, R. Huang, and Z. Zhao, "A new model of rgb-d camera calibration based on 3d control field," *Sensors*, vol. 19, p. 5082, Nov. 2019. DOI: 10.3390/s19235082.

[15] Shiffman, *Openkinect-for-processing at master · shiffman/openkinect-for-processing*. [Online]. Available: %5Curl%7Bhttps://github.com/shiffman/OpenKinect-for-Processing/blob/master/OpenKinect-Processing/examples/Kinect%5C_v2/CameraPointCloud2/CameraParams.pde%7D.

[16] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, 2014.
[17] K.-Y. Shin and J. H. Mun, "A multi-camera calibration method using a 3-axis frame and wand," *International Journal of Precision Engineering and Manufacturing*, vol. 13, pp. 283–289, 2012.
[18] [Online]. Available: https://www.sqlite.org/.
[19] P. G. D. Group, May 2024. [Online]. Available: https://www.postgresql.org/.
[20] [Online]. Available: https://www.mysql.com/.
[21] P. LePage and R. Andrew, *Responsive web design basics : Articles : Web.dev*, Feb. 2019. [Online]. Available: https://web.dev/articles/responsive-web-design-basics.
[22] [Online]. Available: https://flask.palletsprojects.com/en/3.0.x/.
[23] [Online]. Available: https://sass-lang.com/.
[24] [Online]. Available: https://babeljs.io/.
[25] Wikipedia contributors, *Acrylonitrile butadiene styrene — Wikipedia, the free encyclopedia*, [Online; accessed 15-May-2024], 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Acrylonitrile_butadiene_styrene&oldid=1217604073.
[26] Wikipedia contributors, *Polystyrene — Wikipedia, the free encyclopedia*, [Online; accessed 15-May-2024], 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Polystyrene&oldid=1222708934.
[27] Wikipedia contributors, *Polypropylene — Wikipedia, the free encyclopedia*, [Online; accessed 15-May-2024], 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Polypropylene&oldid=1223546299.
[28] Wikipedia contributors, *Poly(methyl methacrylate) — Wikipedia, the free encyclopedia*, [Online; accessed 15-May-2024], 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Poly(methyl_methacrylate)&oldid=1222876167.
[29] Wikipedia contributors, *Polycarbonate — Wikipedia, the free encyclopedia*, [Online; accessed 15-May-2024], 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Polycarbonate&oldid=1221912888.
[30] Wikipedia contributors, *Polyethylene — Wikipedia, the free encyclopedia*, [Online; accessed 15-May-2024], 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Polyethylene&oldid=1223607411.
[31] Wikipedia contributors, *Thermoplastic elastomer — Wikipedia, the free encyclopedia*, [Online; accessed 15-May-2024], 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Thermoplastic_elastomer&oldid=1223923451.
[32] Wikipedia contributors, *Thermoplastic polyurethane — Wikipedia, the free encyclopedia*, [Online; accessed 15-May-2024], 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Thermoplastic_polyurethane&oldid=1221142823.
[33] C. V. Nguyen, S. Izadi, and D. Lovell, "Modeling kinect sensor noise for improved 3d reconstruction and tracking," in *2012 second international conference on 3D imaging, modeling, processing, visualization & transmission*, IEEE, 2012, pp. 524–530.
[34] D. Gotterbarn, B. Brinkman, C. Flick, *et al.*, "Acm code of ethics and professional conduct," 2018.
[35] *IEEE Code of Ethics*. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html.

Appendix A Schedule

starting date of each week	Haoze Gao	Haowen Zheng	Qiran Pan	Yiheng Zhang
25-Mar	choose stepmotor and material for building physical structure	configure the processor and try to run a basic demo.	finalize structure design on the container and the supporter	try out some contour painting methods using Python. Study digital signal processing methods for processing depth data.
1-Apr	Collect data for training segmentation algorithm of body parts.	connect depth info with color and do a simple projection	combine projector and the sensor into a unified structure.	learn OpenCV in C++ about denoising and dehazing
8-Apr	do segmentation training	realize projecting contours onto sand	do physical testing on the models and build prototypes	write program that does ISP and do some testing
15-Apr	design PCB for motor control and write program on turning RGB color into auto-focusing control signal	build database for temporary contour storage	build the sandbox and the container	help build the container and the sandbox
22-Apr	solder PCB and route with stepmotor	learn about wireless transmission protocol	build the supporting pillar and connect all structure together	configure power system and test on the active power

29-Apr	help organize parts in the container	write program on wireless transmission	organize parts in the container	write program on automatic recognition of projection range so that sandbox boundaries can be defined
6-May	Mock demo	Mock demo	Mock demo	Mock demo
13-May	Final demo	Final demo	Final demo	Final demo
20-May	final individual design report	final individual design report	final individual design report	final individual design report

Table 3: Schedule of the Project

Appendix B Requirements and Verification Table

Requirement	**Verification**
The RGB, IR and depth frames should transmit to the processing unit at a frame rate of at least $30\,\mathrm{fps}$.	Run a program with basic frames retrieving and displaying function. Log the frame rate and the log shows a stable transmission of frame data of $42\,\mathrm{fps}$ on average over 10 minutes.
The depth sensor should provide depth information with a resolution of 512×424 pixels.	The dimension readings of depth data from the code is 512×424 pixels.
The depth sensor should run for a long time without overheating or performance issues.	Run the sensor for an one hour and log the frame rate. The frame rate keeps above our desired value of $30\,\mathrm{fps}$.
The topography rendering module should be able to construct a topography map from depth information and project the topography map onto the sand surface.	Build sand models in different shapes and check if the contour line and color is correctly assigned.

The system must ensure that the wireless connection, facilitated by the REALTEK 8822CE network card, is stable and can support multiple users. Also, the database shoud be updated correctly when multiple users are saving models or loading models at the same time.	Use more than 5 devices to access the website at the same time and each do operations like reloading homepage, saving or loading a model. Then check the connection stability and see if the save and load operations are correct. Each user connected wirelessly should not experience operation delay longer than 3 seconds.
The calibration algorithm should be able to track the projected image alignment with the sandbox and the alignment error should be less than $1\,\mathrm{cm}$ when installed with our sandbox.	Use the calibration algorithm to track the projected image alignment with the sandbox. Put boxes with known dimensions on the sandbox and project the image. Check if the offset between the projected image and the boxes is within the acceptable range.
The auto focusing algorithm should be able to adjust the focus of the projector according to the height of the sand surface.	Use the auto focusing algorithm to adjust the focus of the projector according to the height of the sand surface. Adjust the height of sand table and project image with thin lines with one pixel wide onto it. Check if the projected lines are clear.
The powering subsystem should be able to provide power to the sensor, processing, and display subsystems with sufficient power and voltage.	Connect the sensor, processing, and display subsystems to the powering subsystem. Check if the sensor, processing, and display subsystems are powered on. Put all subsystems at work for more than an hour, check if sufficient power is provided and no components failed.
Design a sturdy and stable support structure capable of safely holding the weight of the AR sandbox ($25\,\mathrm{kg}$) and the projector set($5\,\mathrm{kg}$) without risk of collapse or instability during use. The weight of the sandbox and projector should be distributed evenly across the support structure.	Conduct load testing to verify that the support structure can safely withstand the combined weight of the AR sandbox, sand and the projector set. Apply enough sand to simulate real-world usage scenarios. Check for any signs of stress concentration or uneven loading that could compromise stability.
Ensure that the design minimizes potential hazards for children interacting with the AR sandbox. Eliminate sharp edges, protruding components, or other features that could cause injury.	Perform a safety inspection of the support structure to identify any potential hazards or safety concerns. All edges should be rounded, sharp corners be eliminated, and there are no small parts that could pose choking hazards to children.

The projector set on top should be separable, and the separable part should be able to be used separately and can adapt to different use environments and scenarios.	Conduct a practical test where the projector is detached from the support structure. Verify that the separable part functions independently and can be easily moved to different locations or setups as needed. Then assess the reattachment process to confirm that it can be securely and quickly reconnected to the support structure without compromising stability or safety.
The restoration system should achieve the re-tiling of the sand surface in a simple way.	Conduct a practical test where the sand is deliberately disturbed or unevenly distributed. Use the restoration system and observe its ability to re-tilt the sand surface to a uniform level. Then evaluate the consistency and smoothness of the restored sand surface to by projecting topography onto the sand.

Appendix C PCB Design of Stepper Motor Drive Board

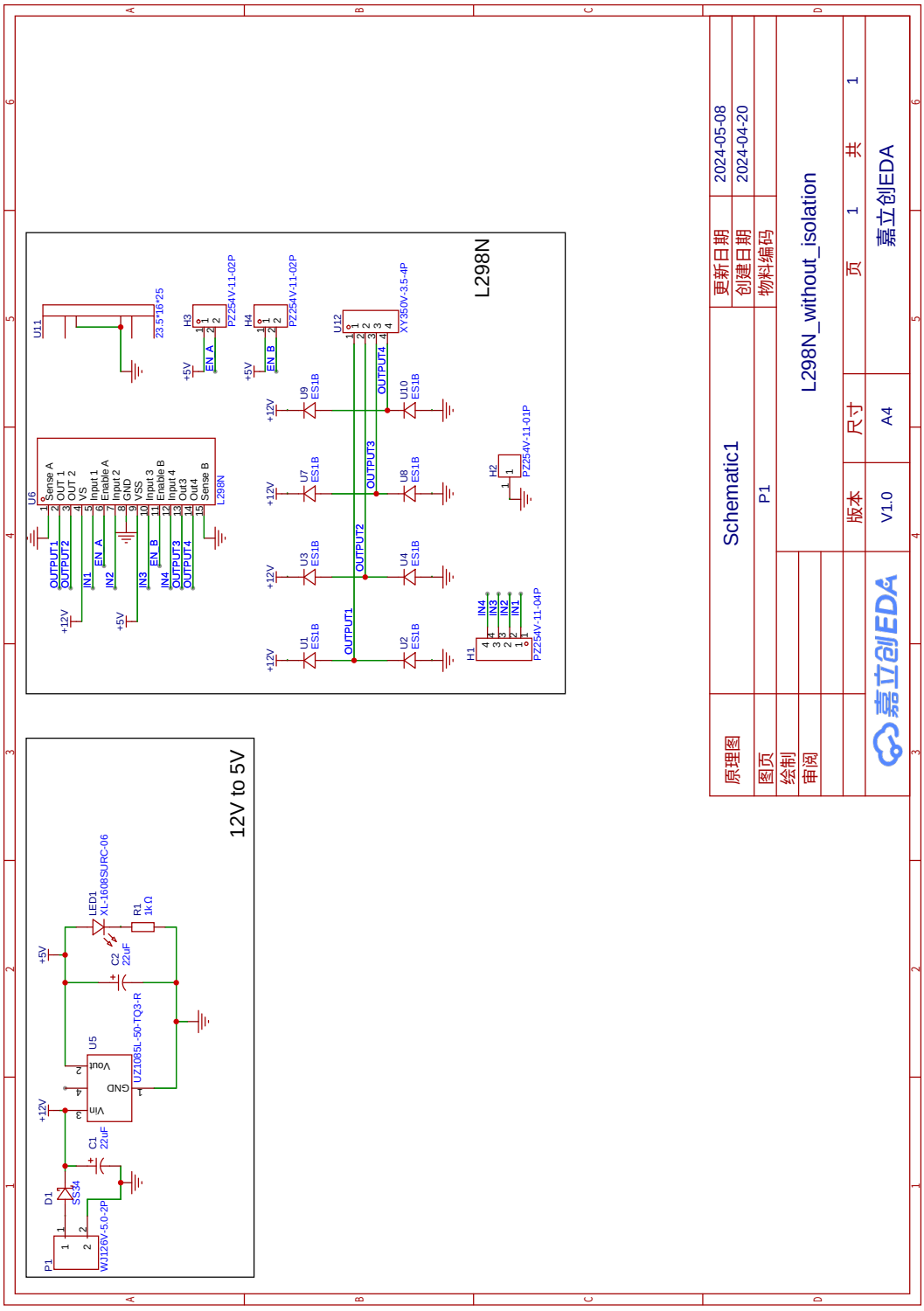


Figure 19: Circuit Design of Stepper Motor Drive Board

Appendix D Pseudocode for Motor Control

Algorithm 1 Motor Function Pseudocode

```
1:  procedure MOTOR(distance, direction)
2:      Import necessary libraries (GPIO and time)
3:      if distance is not valid or direction is not valid then
4:          Exit procedure
5:      end if
6:      Set GPIO mode to BOARD
7:      Define control_pins and set them as GPIO output
8:      Define halfstep sequence for the motor
9:      if direction is -1 then
10:         Reverse the halfstep sequence
11:     end if
12:     Calculate total steps required
13:     Define minimum and maximum delay times for motor control
14:     Define acceleration, deceleration and steady time
15:     Initialize counter i to 0
16:     while i is less than total steps do
17:         Get the current halfstep based on i
18:         Calculate the delay time based on the current step number
19:         Output the corresponding value in halfstep
20:         Pause for delay time
21:         Increment i by 1
22:     end while
23:     Cleanup GPIO
24: end procedure
```

Appendix E Pseudocode for Locating Sandbox Edge

Algorithm 2 Locate Sandbox Edge Pseudocode

function LOCATESANDBOXEDGE(img)
    Apply histogram equalization to the input image
    $img\_hist \leftarrow$ CV2.EQUALIZEHIST(img)
    Apply a binary threshold to the image
    $binary \leftarrow$ CV2.THRESHOLD$(img_hist, 250, 255, cv2.THRESH_BINARY)$
    Initialize parameters for blob detection
    $params \leftarrow$ CV2.SIMPLEBLOBDETECTOR_PARAMS
    $params.blobColor \leftarrow 255$
    $params.filterByArea \leftarrow True$
    $params.filterByCircularity \leftarrow True$
    Create a blob detector with specified parameters
    $detector \leftarrow$ CV2.SIMPLEBLOBDETECTOR_CREATE$(params)$
    Detect blobs in the binary image
    $keypoints \leftarrow$ DETECTOR.DETECT$(binary)$
    if LEN(points) ¡ 3 **then**
        Print "No sandbox detected"
        return $im\_with\_keypoints, points, binary$
    end if
    Compute the convex hull of points
    $hull \leftarrow$ CV2.CONVEXHULL$(points)$
    Draw the convex hull as the contour on the image
    CV2.DRAWCONTOURS(im_with_keypoints, [hull], 0, (0, 255, 0), 3)
    return $im\_with\_keypoints, points$
end function

Appendix F Code for Converting Camera's Pixel Coordinates to World Coordinates

```
def depthMatrixToPointCloudPos(z, scale=100):
    C, R = np.indices(z.shape).astype(np.float64)

    np.subtract(R, CameraParams['cx'], out=R)
    np.multiply(R, z, out=R)
    np.divide(R, CameraParams['fx'] * scale, out=R)

    np.subtract(C, CameraParams['cy'], out=C)
    np.multiply(C, z, out=C)
    np.divide(C, CameraParams['fy'] * scale, out=C)

    return np.column_stack((R.ravel(), -C.ravel(), z.ravel() /
        scale))
```

Appendix G Code for Converting World Coordinates to Projector's Pixel Coordinates

Here is the code for converting world coordinates to pixel coordinates in the projector. This is optimized for speed by using vectorized operations in NumPy and changing the data types to float32.

```
def cameraToProjector_f32(color, depth, P_projector):
    if P_projector is None:
        return color

    points_in_world = depthMatrixToPointCloudPos(depth.astype(np.
        float32))

    points_in_world = points_in_world.reshape(-1, 3).astype(np.
        float32)
    world_points_h = np.hstack((points_in_world, np.ones((
        points_in_world.shape[0], 1), dtype=np.float32)))

    points_projector = np.dot(P_projector.astype(np.float32),
        world_points_h.T).T
    points_projector = points_projector[:, :2] / points_projector
        [:, 2:]

    color_projector = np.zeros((DISPLAY_HEIGHT, DISPLAY_WIDTH, 3)
        , np.uint8)

    valid_indices = ~np.isnan(points_projector).any(axis=1)

    valid_points = points_projector[valid_indices].astype(int)

    valid_indices = (valid_points[:, 0] >= 0) & (valid_points[:,
        0] < DISPLAY_WIDTH) & \
                    (valid_points[:, 1] >= 0) & (valid_points[:,
                        1] < DISPLAY_HEIGHT)

    valid_points = valid_points[valid_indices]
    valid_indices = valid_indices.nonzero()[0]

    color_values = color.reshape(-1, 3)[valid_indices]

    color_projector[valid_points[:, 1], valid_points[:, 0]] =
        color_values

    return color_projector
```

Appendix H Simulation Results of the Structure Subsystem

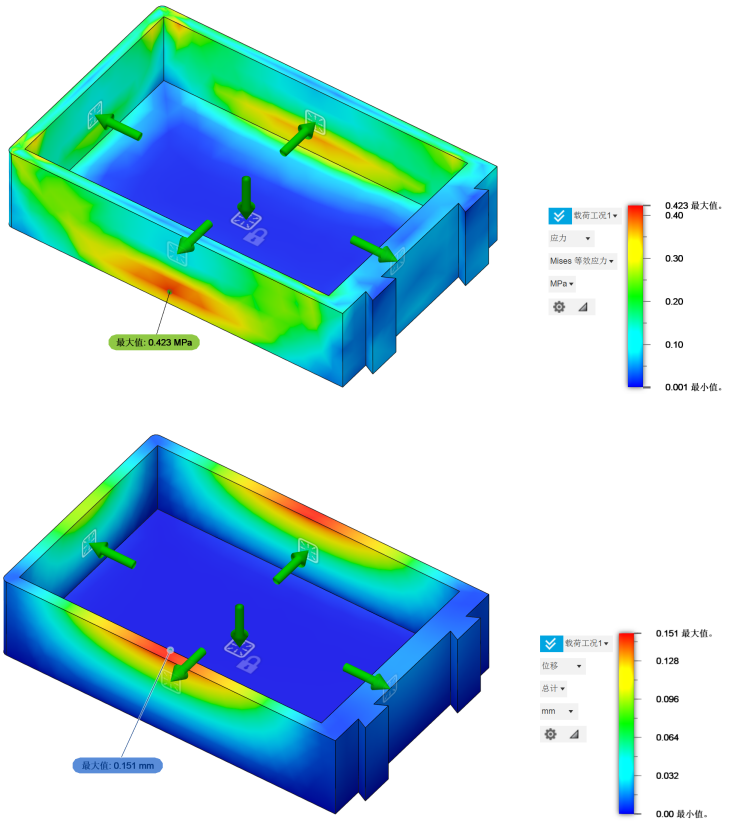


Figure 20: Bottom and Side Pressure Simulation of the Sandbox with Sand in the Box

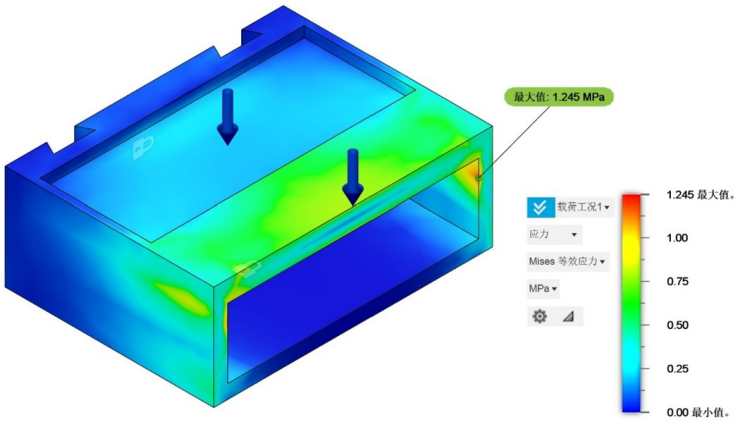


Figure 21: Pressure Analysis on PS Material Defects

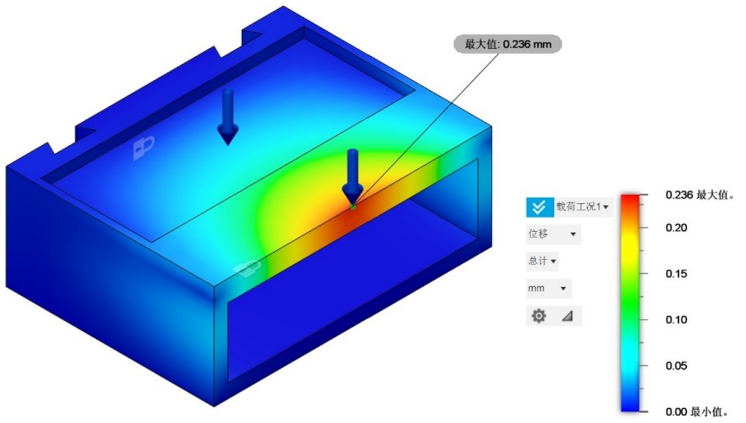


Figure 22: Deformation Analysis on PS Material Defects

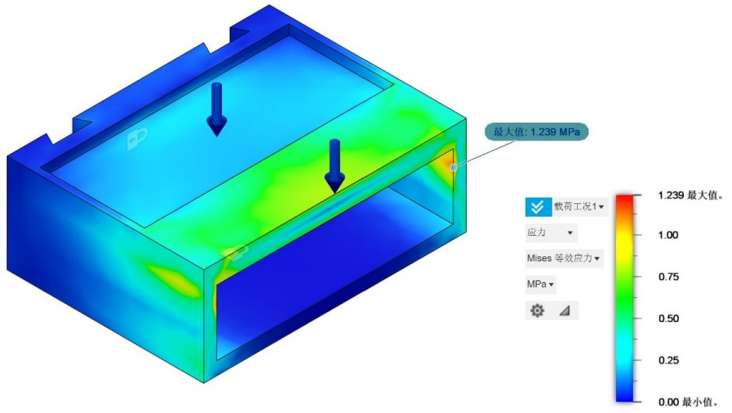


Figure 23: Pressure Analysis on PP Material Defects

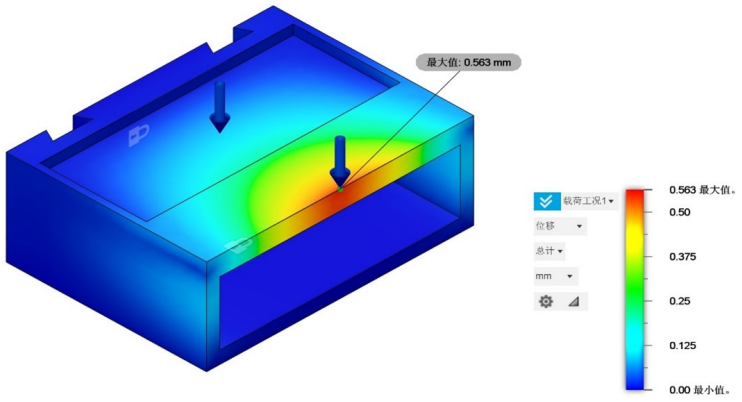


Figure 24: Deformation Analysis on PP Material Defects

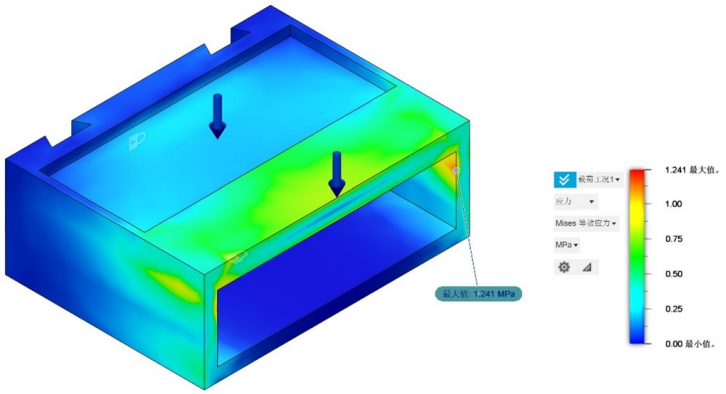


Figure 25: Pressure Analysis on ABS Material Standard Part

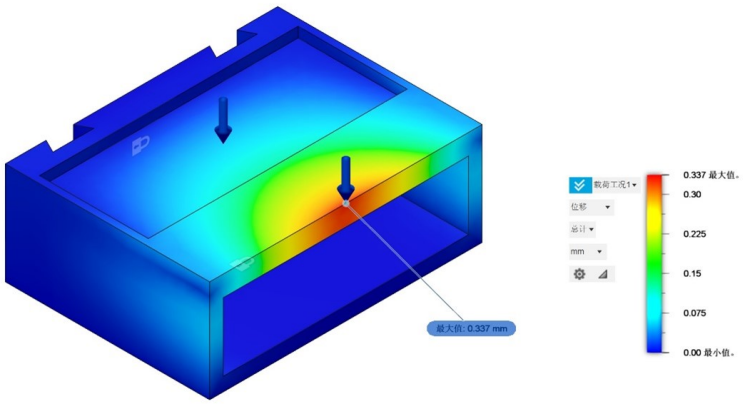


Figure 26: Deformation Analysis on ABS Material Standard Part

Appendix I PCB Design of Stepper Motor Drive Board

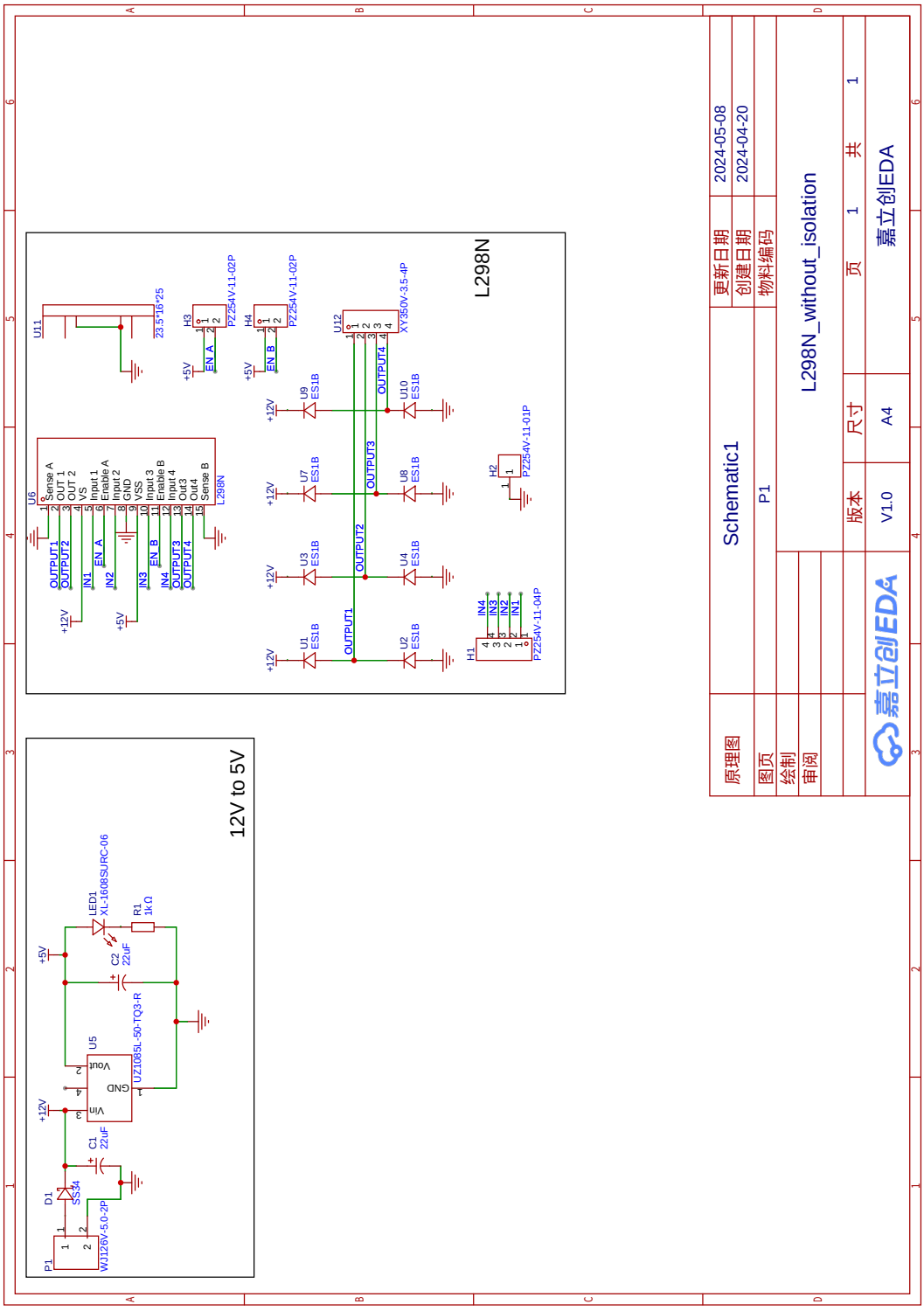


Figure 27: Circuit Design of Stepper Motor Drive Board

Appendix J Database and wireless transmission block diagram dig-in

Decorator: `@app.route('/')`

Decorator: `@limit_users(10)`

Function: `index`

Description: This function serves as a route handler for the root URL ('/'). It limits access to a maximum of 10 users concurrently using the `limit_users` decorator. The function increments the count of active users when a new user accesses the page and deletes the existing 'frames.db' database file, which is the database that saves and loads frame profiles. It then fetches the list of captured frames from the database using the `get_frame_list` function. After generating a QRcode for the current URL, it saves the QRcode image and renders the 'index.html' template, passing the frames and QRcode image path as context variables.

Decorator: `@app.route('/static/images/<path:path>')`

Function: `send_image`

Description: This function serves static images located in the 'static/images' directory. It takes a path parameter representing the image file path relative to the 'static/images' directory. It uses Flask's `send_from_directory` function to locate and send the requested image file, which is the QRcode to the client.

Decorator: `@app.route('/video_feed')`

Function: `video_feed`

Description: This function serves as a route handler for `video_feed`. It streams video frames using the `generate_frame` function, which continuously takes in frames from RGB portal of the depth camera. It returns a Flask `Response` object with the generator function `generate_frame()` as its content.

Decorator: `@app.route('/get_frames')`

Function: `get_frames`

Description: This function serves as a route handler for `get_frames`. It fetches the list of captured frame named after their timestamp from the database using the `get_frame_list` function. It then renders the `frame_list.html` template, passing the fetched frames as context variables.

Decorator: `@app.route('/capture_frame', methods=['POST'])`

Function: `capture_frame`

Description: This function serves as a route handler for capturing a frame via POST request. It starts by checking if the current frame is available, returning a status code of

500 if it's not. It then converts the current frame to a JPEG image, retrieves the current timestamp, and saves the JPEG image to a BytesIO object. After connecting to the SQLite3 database, it creates the 'frames' table if it doesn't exist and checks the count of frames currently stored in the database. If the count exceeds the maximum frame limit (MAX_FRAMES), it overwrites the oldest frame with the newest. Finally, it inserts the current frame into the database with its timestamp. The function returns an empty response with a status code of 204 (No Content) to indicate that the frame has been successfully captured and saved to the database.

Decorator: `@app.route('/get_image/<timestamp>')`

Function: `get_image`

Description: This function serves as a route handler for fetching an image corresponding to a given timestamp from the database. It takes the timestamp as a parameter in the URL. The function connects to the SQLite3 database, executes a query to select the image data for the given timestamp, and fetches the row. If the row exists, it retrieves the image data and returns it as a response using Flask's `send_file` function with the mimetype set to 'image/jpeg'. If the image is not found for the given timestamp, it aborts with a status code of 404 (Not Found). In case of any exception during the process, it prints an error message and aborts with a status code of 500 (Internal Server Error).

Decorator: `@app.route('/delete_image/<timestamp>', methods=['POST'])`

Function: `delete_image`

Description: This function serves as a route handler for deleting an image corresponding to a given timestamp from the database via a POST request. It takes the timestamp as a parameter in the URL.The function connects to the SQLite3 database, executes a delete query to remove the image data for the given timestamp, and commits the changes. If the deletion is successful, it prints a success message and returns an empty response with a status code of 204 (No Content). In case of any exception during the deletion process, it prints an error message and aborts with a status code of 500 (Internal Server Error).

Decorator: `@app.route('/active_users_count')`

Function: `active_users_count`

Description: This function serves as a route handler for fetching the count of active users. It returns a JSON object containing the count of active users.

Decorator: `@app.teardown_request`

Function: `teardown_request`

Description: This function is registered as a teardown handler by Flask using the `@app.teardown_req` decorator. It is called after each request. Within the function, it attempts to remove the session ID of the current user from the `active_users` set, effectively decrementing the count of active users when a user leaves the page. If an error occurs during this process, it is caught and printed for debugging purposes.

Appendix K dbReader.py

Algorithm 3 Export Images from Database Pseudocode

```
1: function EXPORT_IMAGES_FROM_DB(database_file, output_folder)
2:     Connect to the SQLite database using database_file
3:     Create a cursor object
4:     if output_folder does not exist then
5:         Create output_folder
6:     end if
7:     Execute SQL query to fetch names of all tables in the database
8:     Store result in tables
9:     for each table in tables do
10:        Extract table_name from table
11:        Execute SQL query to select 'image' and 'timestamp' columns from table_name
12:        Store result in images
13:        for each (image_data, timestamp) in images with index do
14:            Construct image_filename using table_name, timestamp, and index
15:            Open a new file in output_folder with name image_filename in write binary mode
16:            Write image_data to the file
17:            Close the file
18:        end for
19:    end for
20:    Close the database connection
21: end function
```