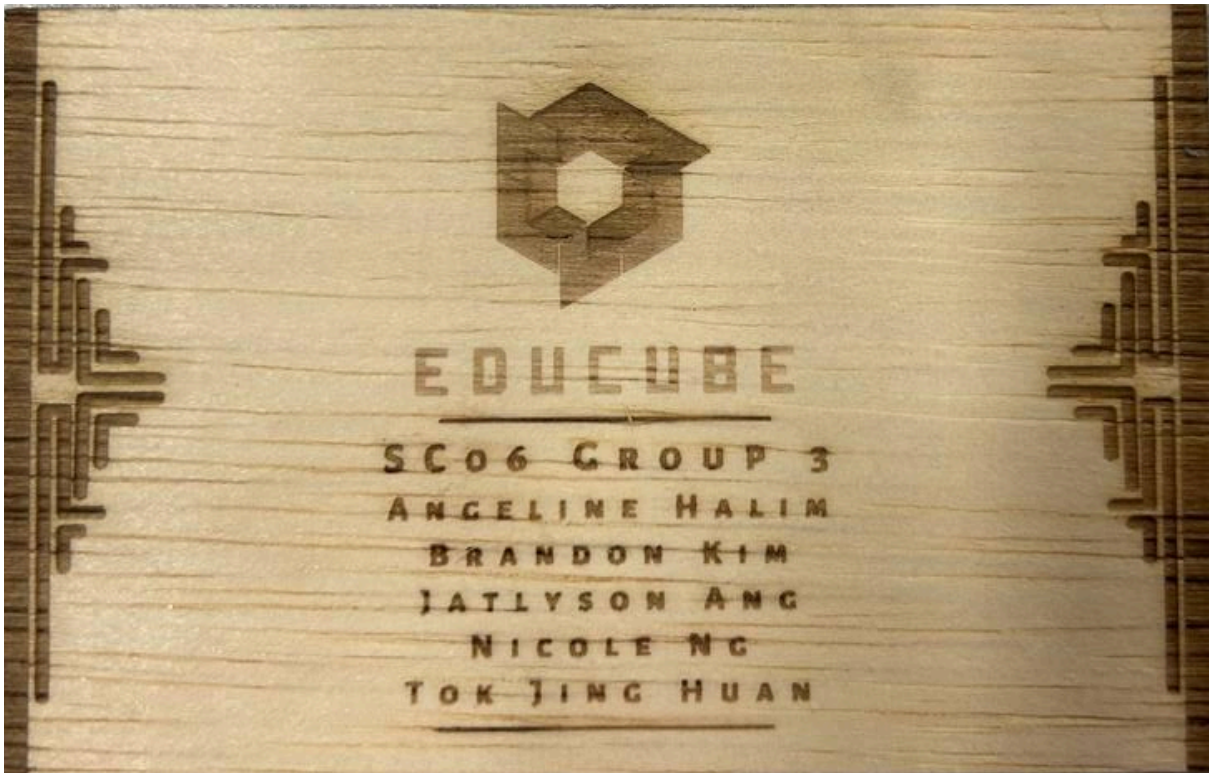
3.007 Design Thinking & Innovation

Project Part 4



CC06 | Group 3 - Nigogo Power Rangers

Introduction to Our Chosen Site

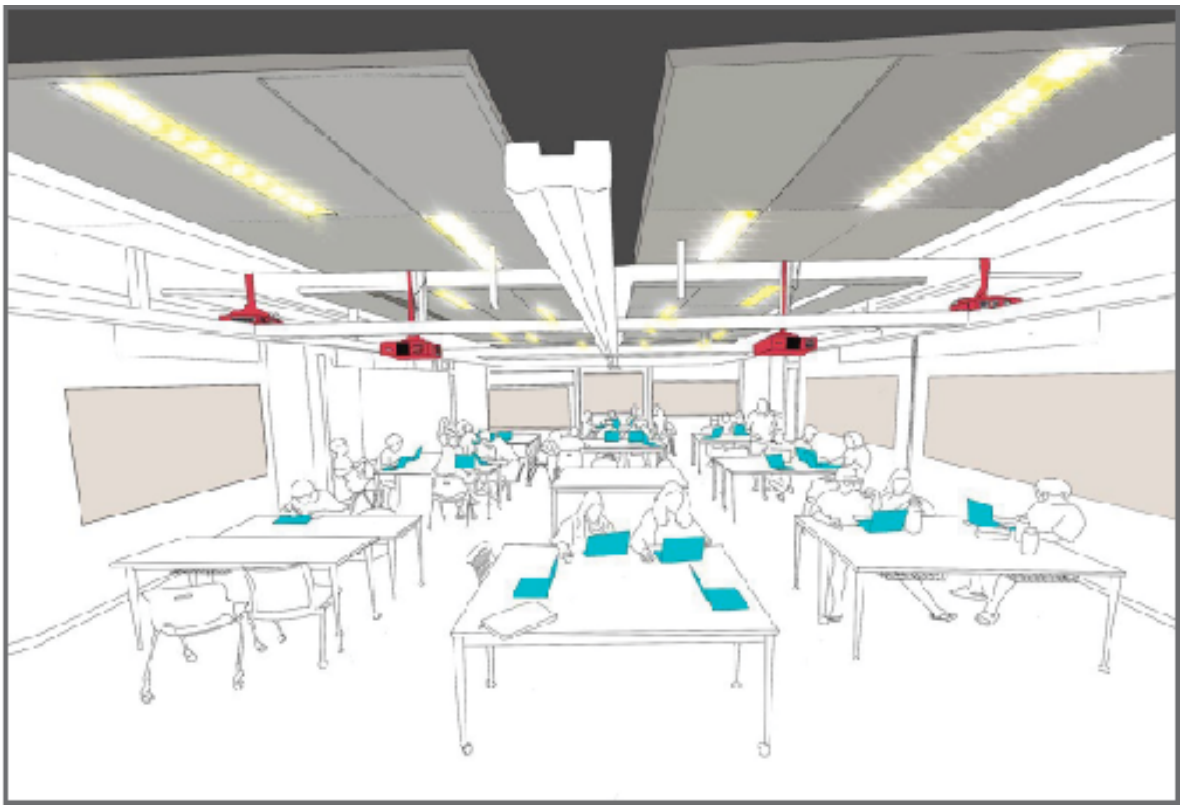
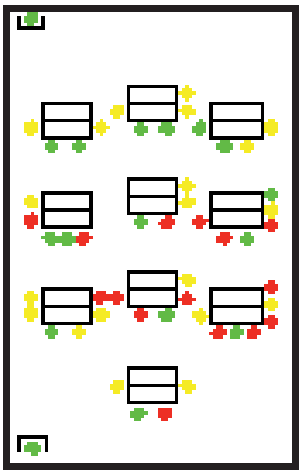
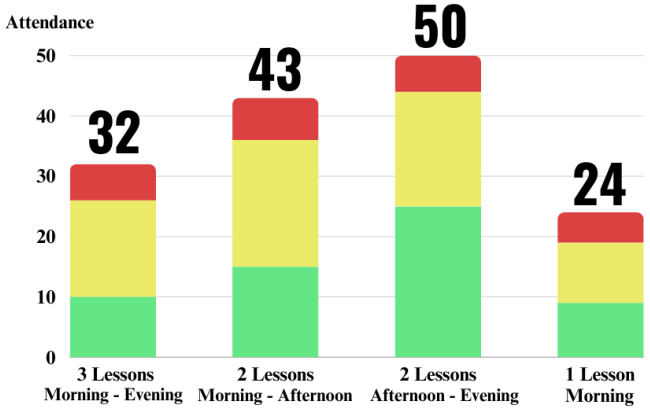


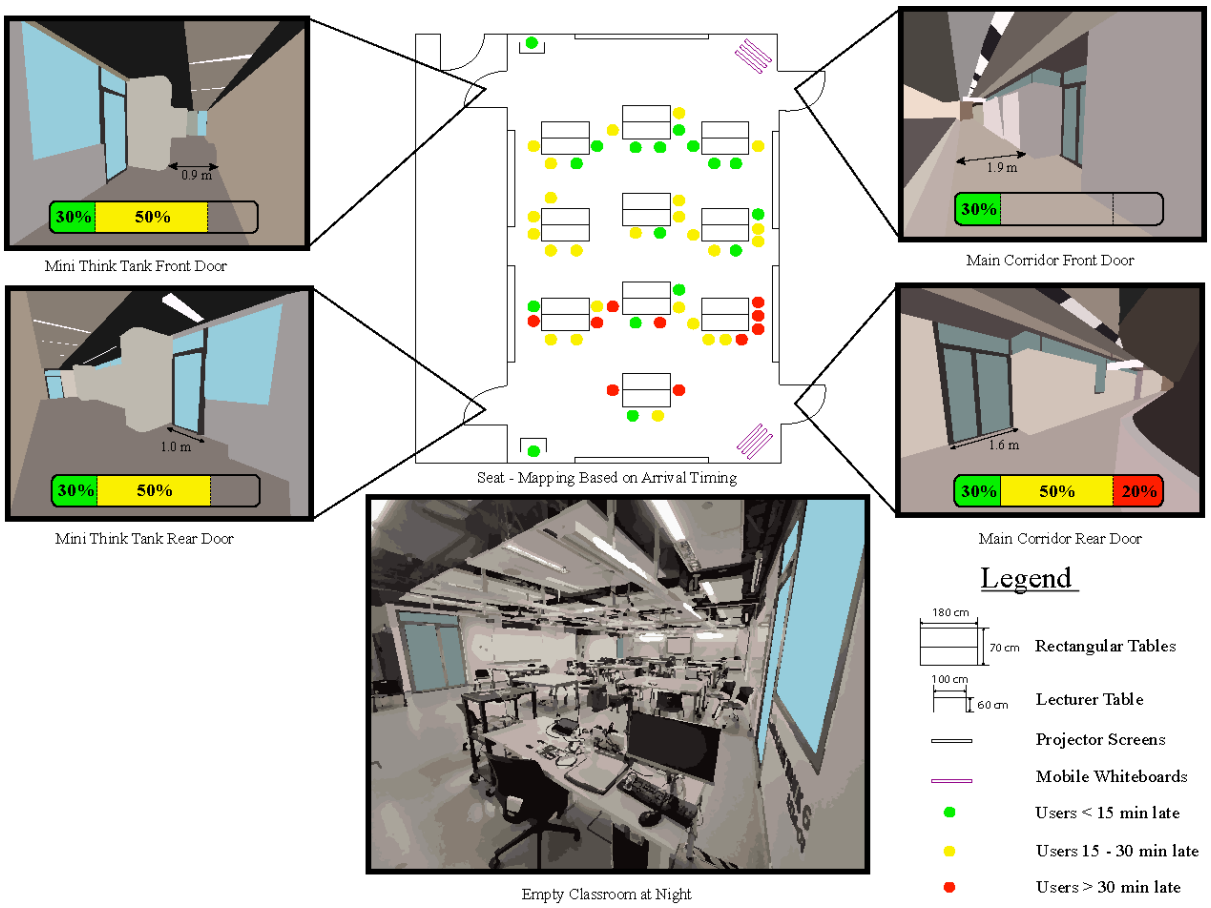
Figure 1: Sketch of the Classroom.

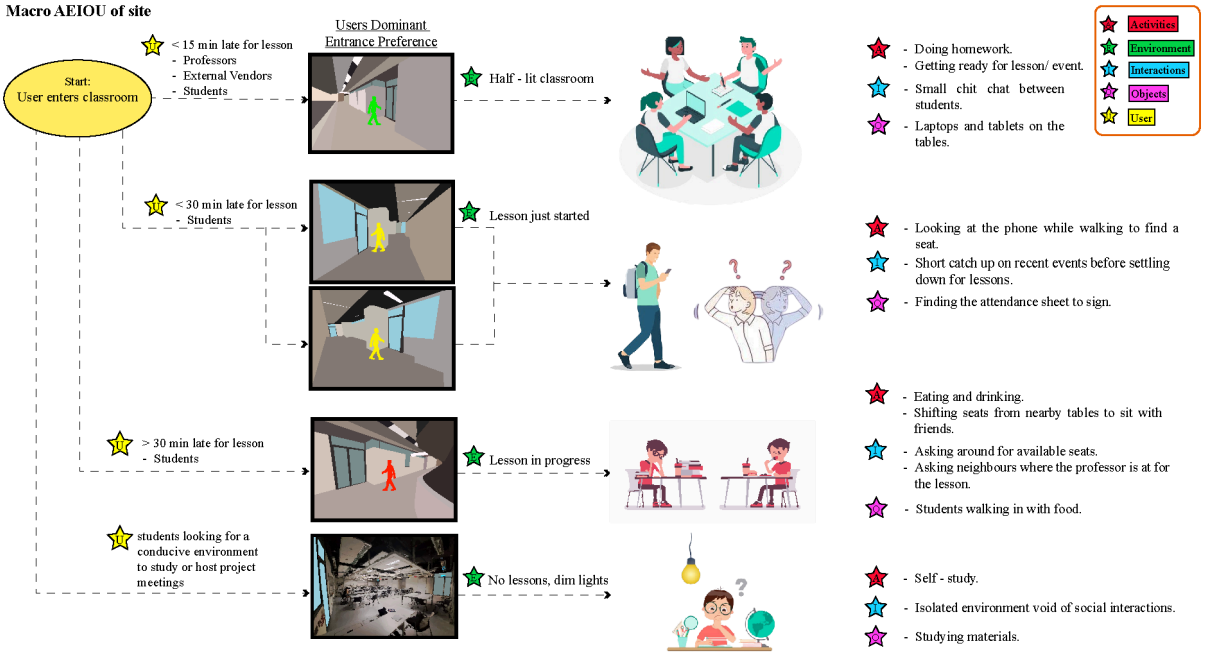
Our project focuses on optimising the cohort classroom environment to improve interactions between students, speakers, and teaching assistants (TAs) to enhance students' learning experiences. We analysed the student-student, student-speaker, and student-TA interactions, considering different types of students (late-comers, early birds, disruptive students) and factors affecting attention span (visibility of board, distractions, fear of asking questions). Your goal is to address issues affecting students paying attention in class, thus enhancing students' learning experience in class.

In our analysis, we explored potential problems and pain points that reduce the effectiveness of the class. One of them includes analysing the behaviour of late students.









There are a couple of reasons why we chose Classroom as our primary site:

a) Timeless (classroom will always be there)
b) Applicable to anywhere (every country has a classroom)

Therefore, our chosen site is globally applicable.

Design Solution

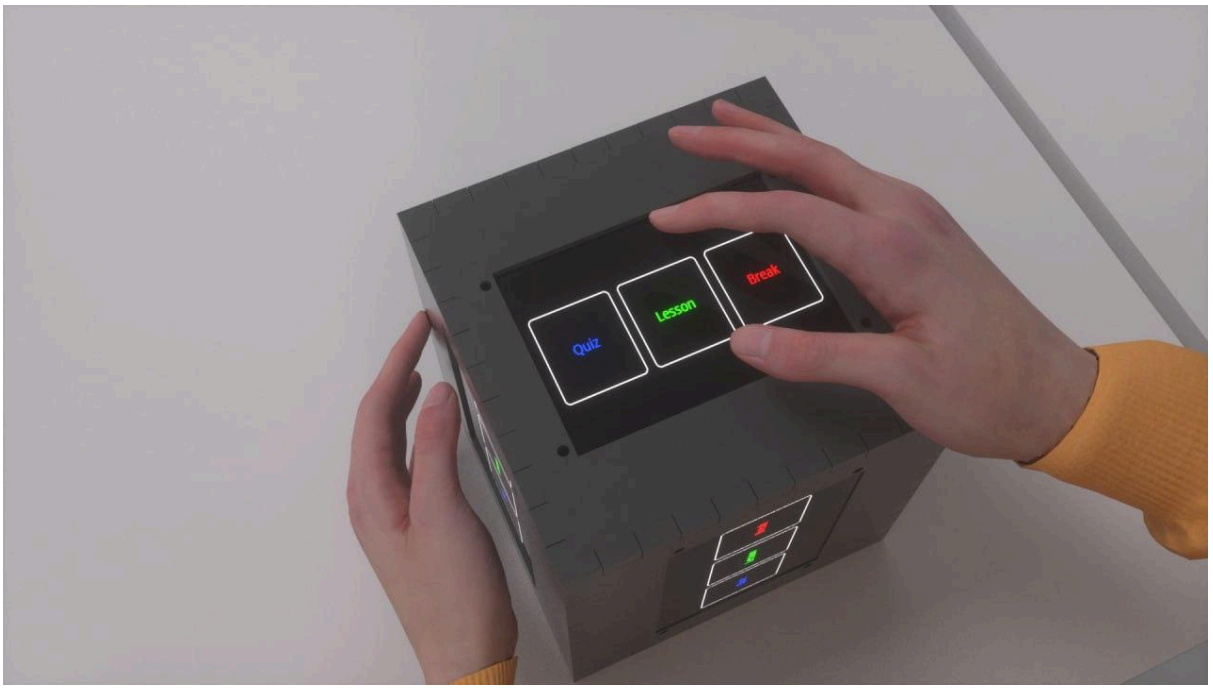


Figure 2: EduCUBE has 3 modes: Quiz Mode, Lesson Mode and Break Mode.

a) Quiz Mode: Allow students to participate in the quizzes launched by the professor.
b) Lesson Mode: Allow students to interact with the professor or teaching assistant (TA).
c) Break Mode: Have a timer indicating how long the given break is.

For our prototype, EduCUBE has 3 main features:

Feature 1: EduCUBE's quiz mode: Students can answer by rotating and pressing on to the cube.

Figure 3: Nic Ho figuring out which answer to select

Figure 3.1: Nic Ho chooses C as her answer

Feature 2: EduCUBE timer mode allows students to keep track of their break time. This is used when the professor gives a break in the middle of the lesson.

Figure 4: EduCUBE showing 5 minutes timer

Feature 3: EduCUBE lesson mode allows students to call for a TA, request for the professor to slow down, ask a question or repeat his point.

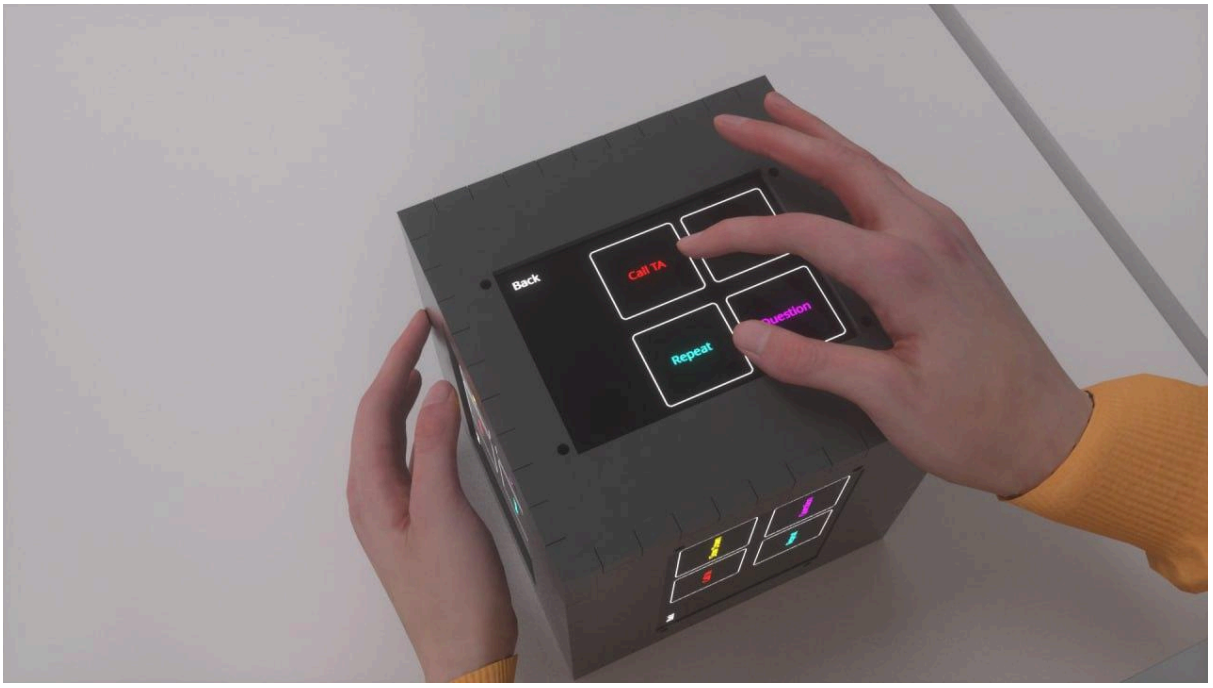


Figure 5.1: Options to call TA, ask the professor to repeat what he is saying, ask a question, or slow down

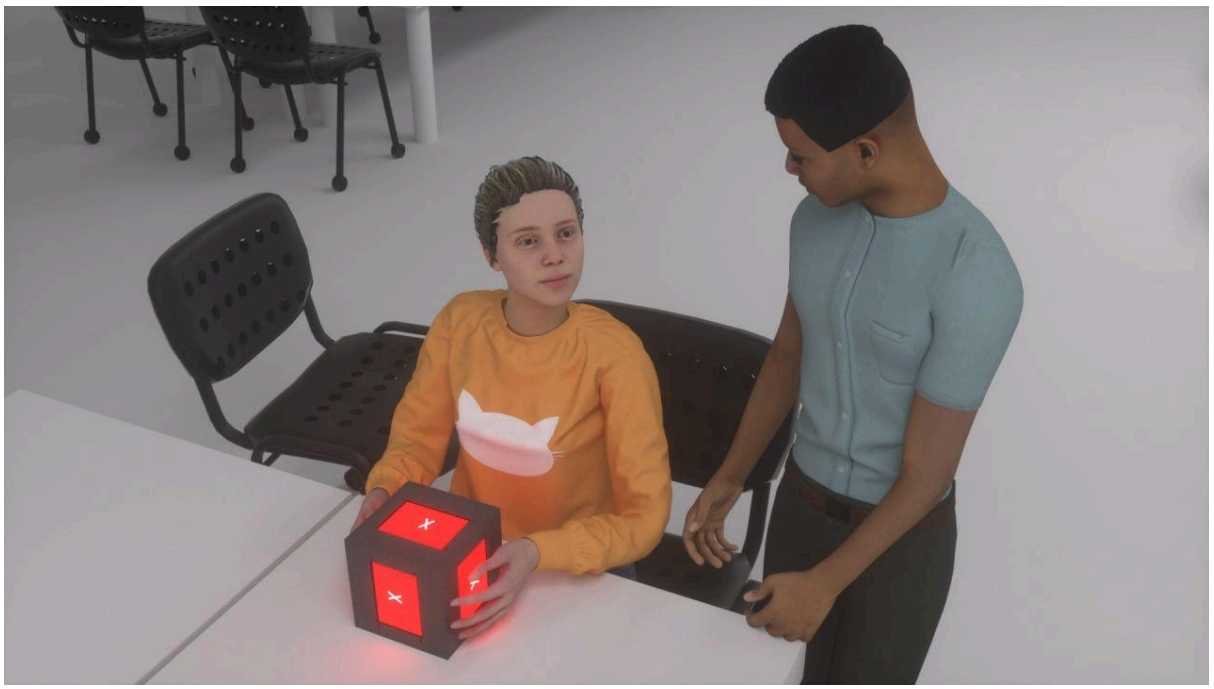


Figure 5.2: Nic Ho calls TA over to help her

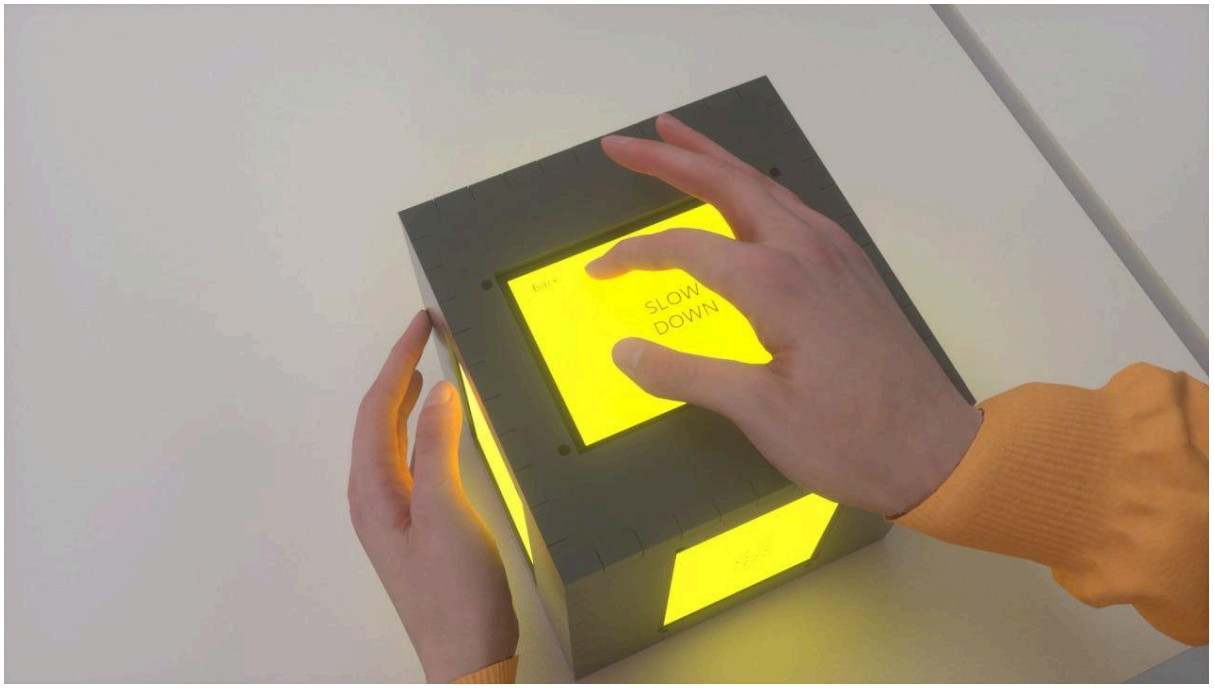
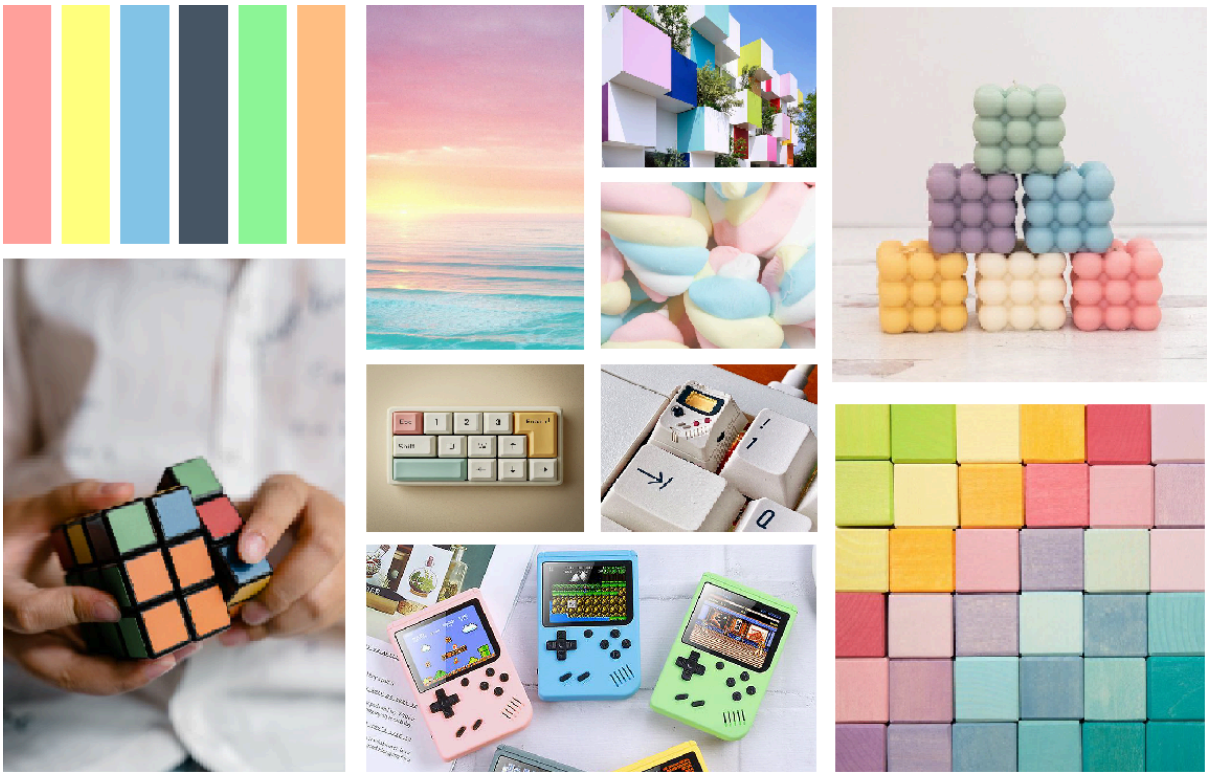
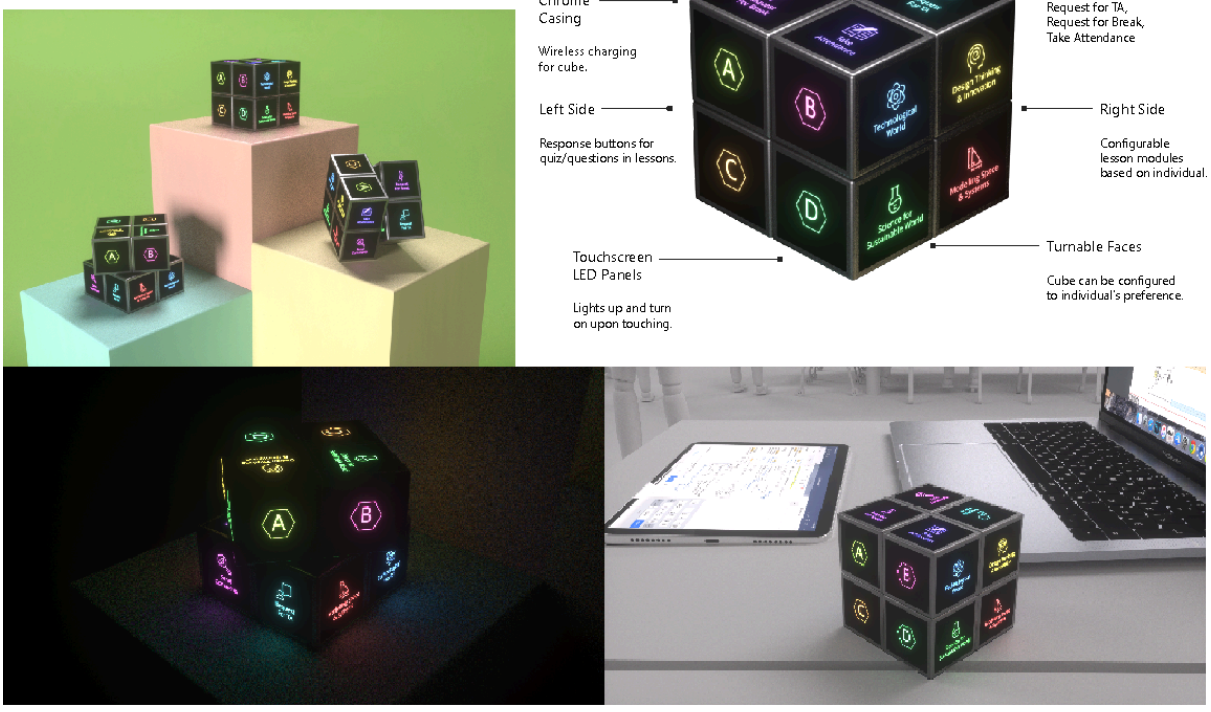


Figure 5.3: Nic Ho chooses the slow-down option to request for the professor to slow down

Inspiration



Contextual Drawings



Hardware Development Process

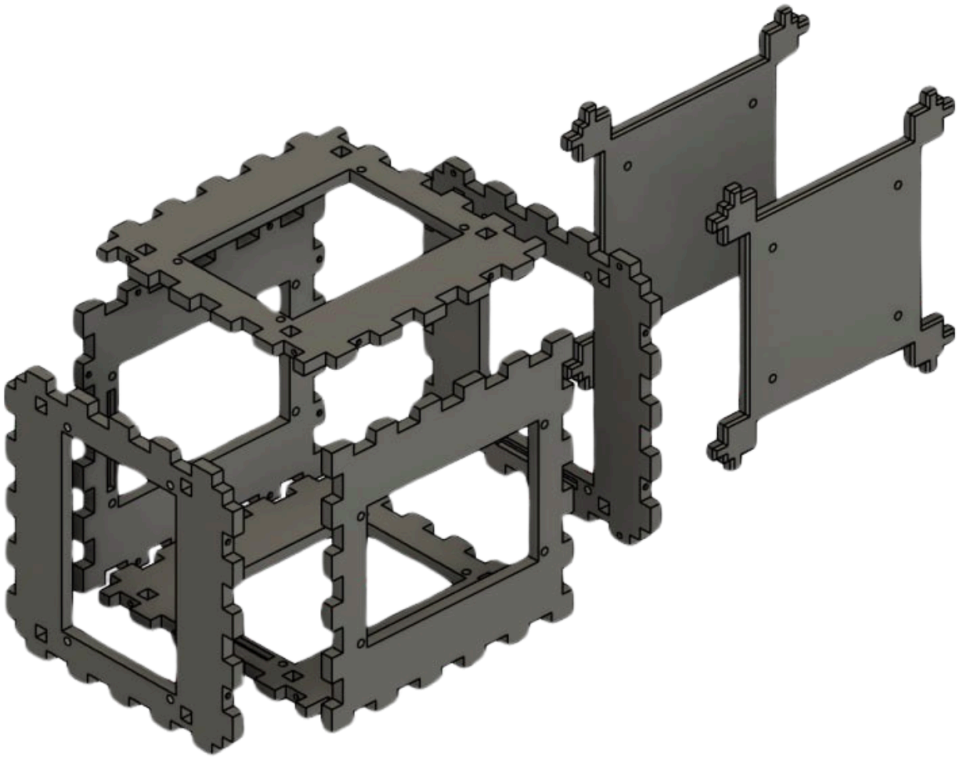


Figure 6.1: 1st CAD Desgin

Inspired by the elegance and precision of traditional Japanese woodworking techniques, we incorporated Sashimono[1] into the design of EduCUBE. This approach led us to create a design where each face of the cube has sides with five to six grooves, which fit perfectly when put together.

To begin, we carefully studied the principles of Sashimono, understanding how each joint and connection could be applied. We sought to blend the timeless beauty of Sashimono with the advanced functionality of a smart cube, creating a product that not only looked stunning but also performed flawlessly.

The process involved meticulous planning and attention to detail. We sketched numerous designs, exploring different ways to integrate sensors, lights, and interactive elements into the cube while maintaining the integrity of Sashimono craftsmanship. With each iteration, We refined the design, ensuring that every joint was not only structurally sound but also aesthetically pleasing.

[1] Sashimono – Japan Woodcraft Association

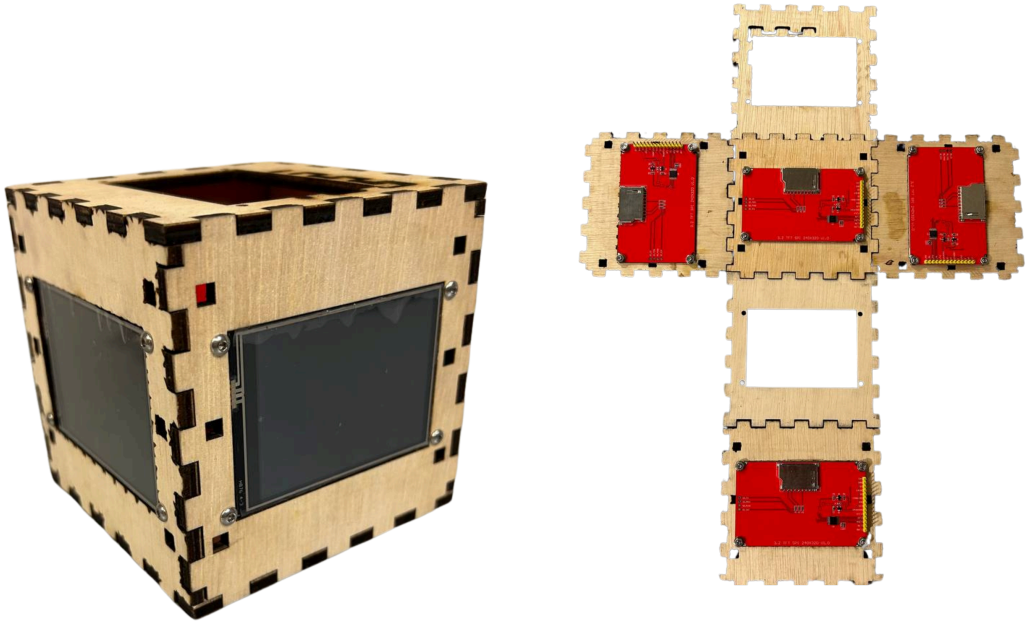


Figure 6.2: 1st iteration of our prototype using laser-cut plywood

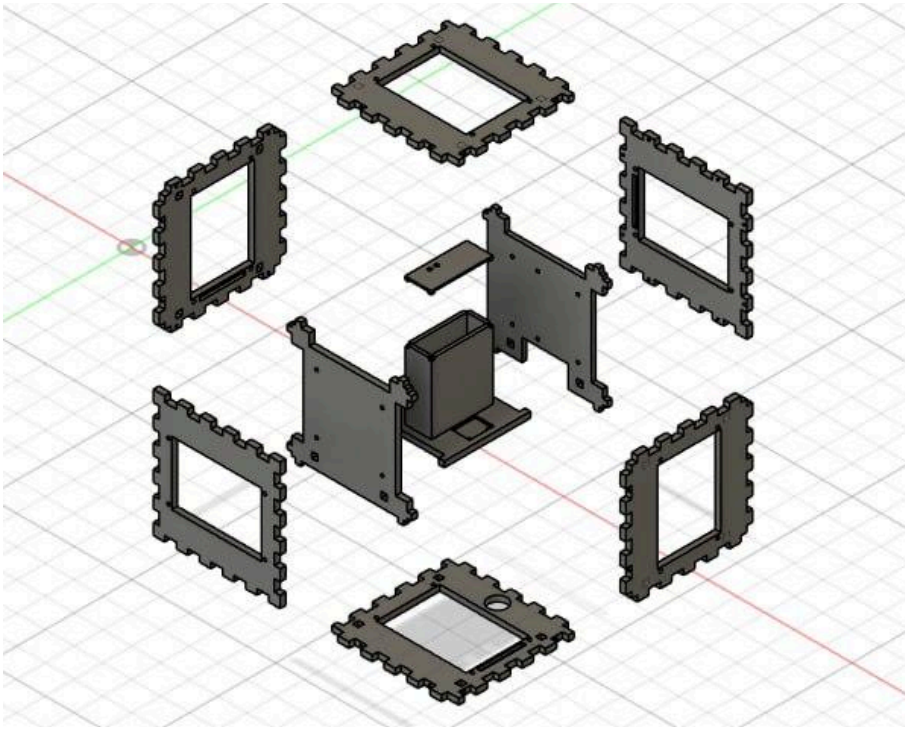


Figure 6.3: Refined CAD Design

This is the CAD view of the frame of the cube. The middle compartment is for the battery to power the cube and hold the sensor in place. The stripboards are screwed onto both sides of the inner walls and the ESPs are soldered onto the stripboards.

To attach the stripboards to the inner walls, we accessed the Fablab drill and we used the 3mm drill bit to fit the size of the screws used.

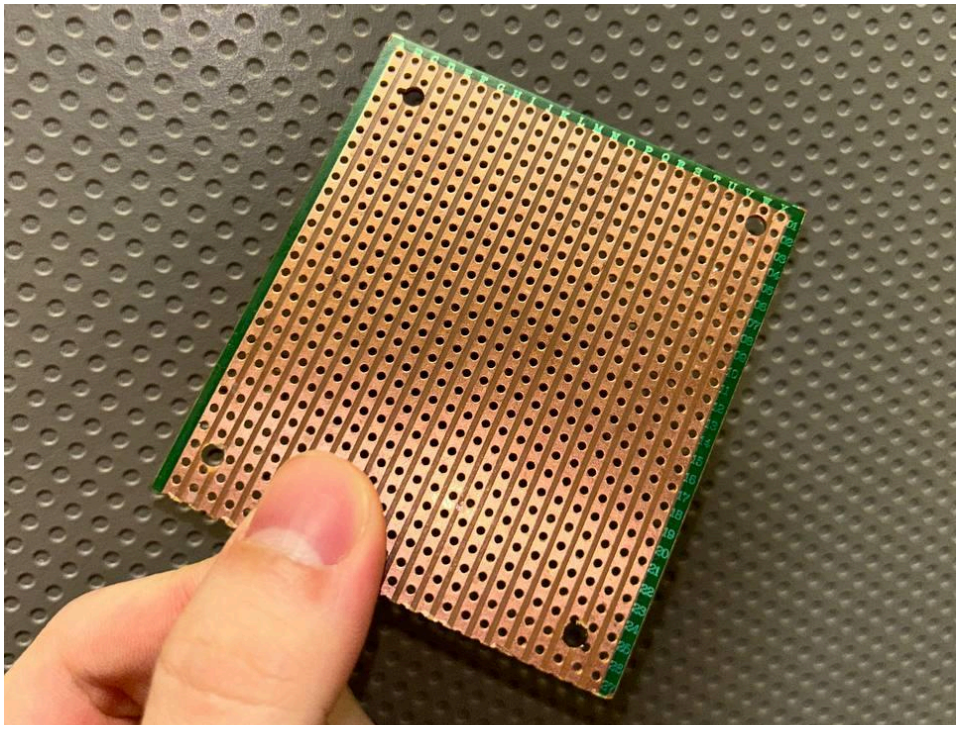


Figure 7: Picture of a stripboard after drilling

To make the wiring and placement of the electronics tight fitting, we desoldered the header pins of the sensor(accelerator and gyroscope) as well as one of the ESPs.

Figure 8: Wirings of ESP to the displays in the cube

Wiring the cube is a complex task due to its compact design and the multitude of wires required for connecting the LCD TFT screens to the ESPs. The confined space within the cube makes it challenging to neatly organise and manage the numerous wires, adding to the intricacy of the wiring process. Each LCD TFT screen needs to be meticulously connected to the ESP microcontroller, requiring careful attention to detail to ensure all connections are secure and functioning properly. During the construction process, we measured and stripped the wires to reduce bulk, have better connections, easier routing and neater appearance as seen below.

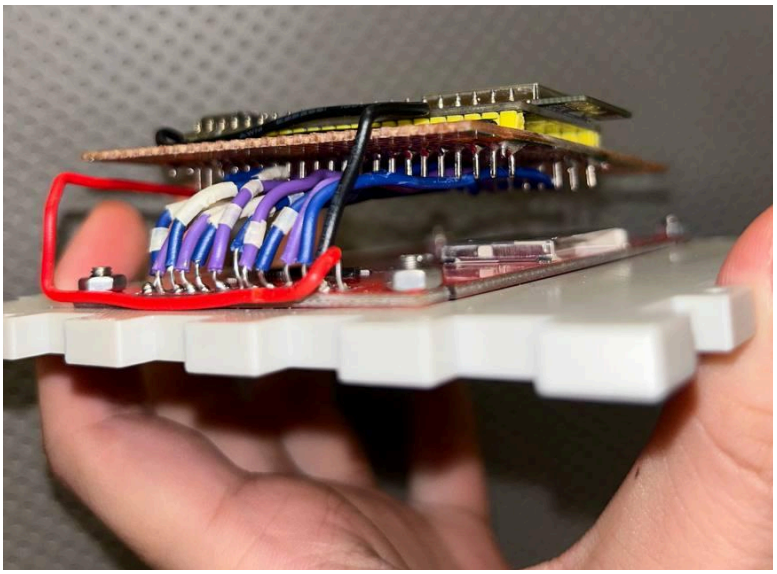


Figure 9: Connection of ESP to 1 display screen

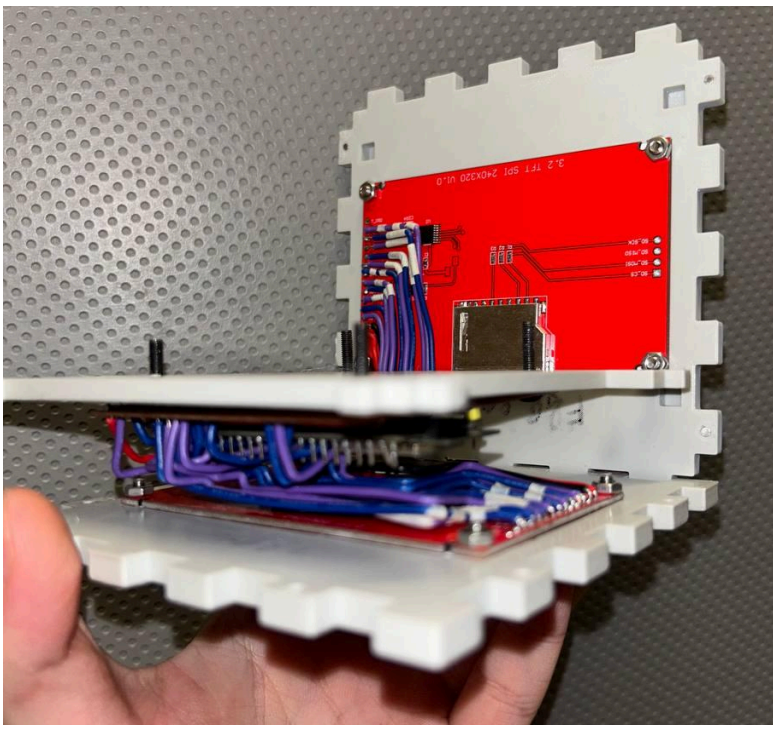


Figure 9.1: Connection of ESP to 2 display screens

Soldering has to be done precisely to ensure that there is no short circuit. We used a multimeter to test for continuity between points to ensure no short circuit in all the stripboards.

Referring to Figure 10, the multicore wire (bottom, brown wire) is flexible while the single core wire (top, purple wire) is rigid. To ensure that the wire doesn't break, we soldered both together at the ends and used electrical insulating tape to wrap around it as seen below. These wires are for the connection for the sensor and the face that is detachable for the replacement of the battery. The rest of the faces of the cubes are fixed with magnets. (However, we did not account for the pressure that the bent wire would exert on the faces, thus, the magnets were not strong enough to hold them and we needed to use super glue on top of the magnets.)

Figure 10.1: Top purple wire (single core) and bottom brown wire (multicore)

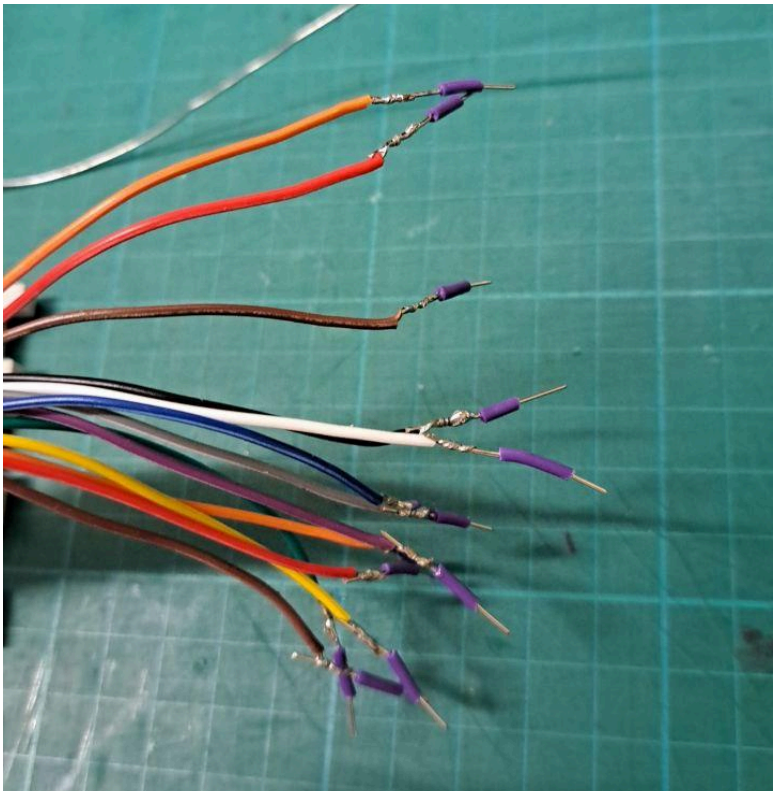


Figure 10.2: After soldering the wires together.

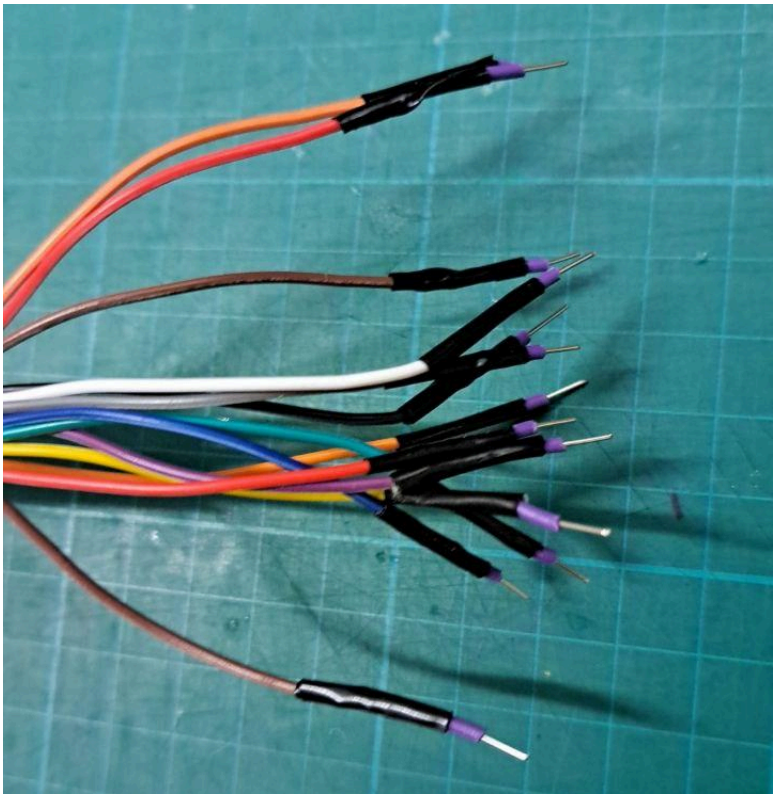


Figure 10.3: After wrapping with electrical insulating tape.

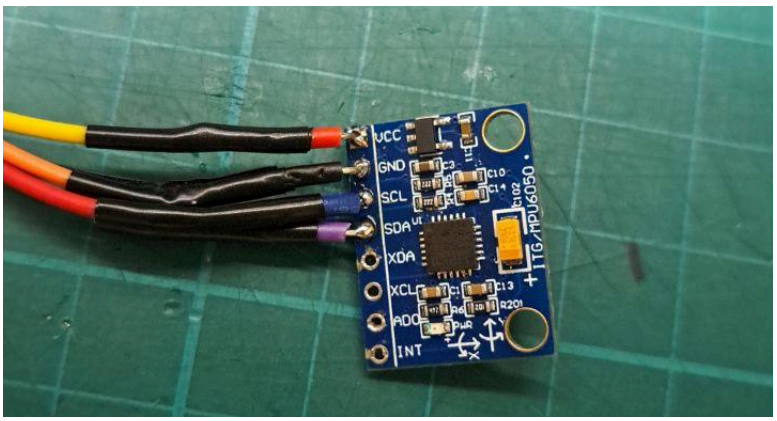


Figure 10.4: The wires when connected to the sensor

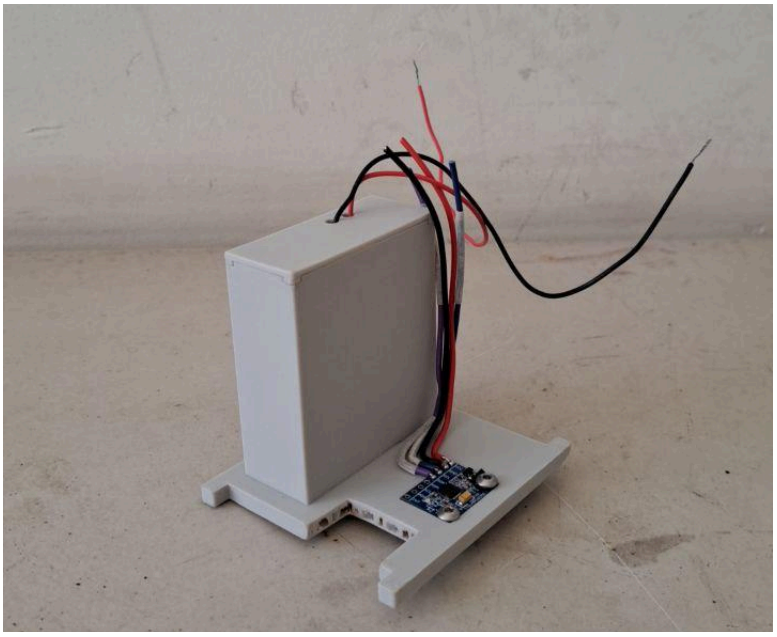
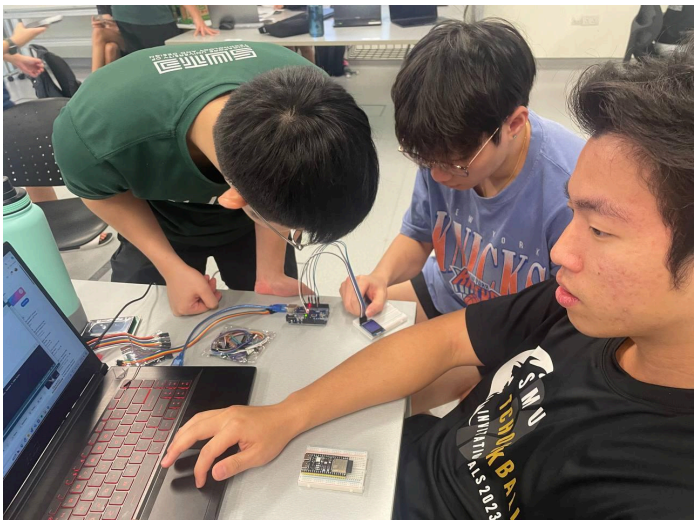


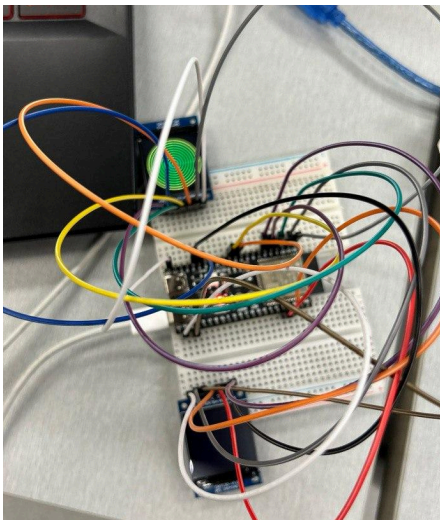
Figure 11: 3D printed battery casing and sensor screwed onto it

Above is the battery compartment and the sensor is screwed on to it as we want the sensor to be in the middle of the cube as much as possible.

Setting up the display screen (connecting the screen to an ESP):

Trying it out on a small display screen first:





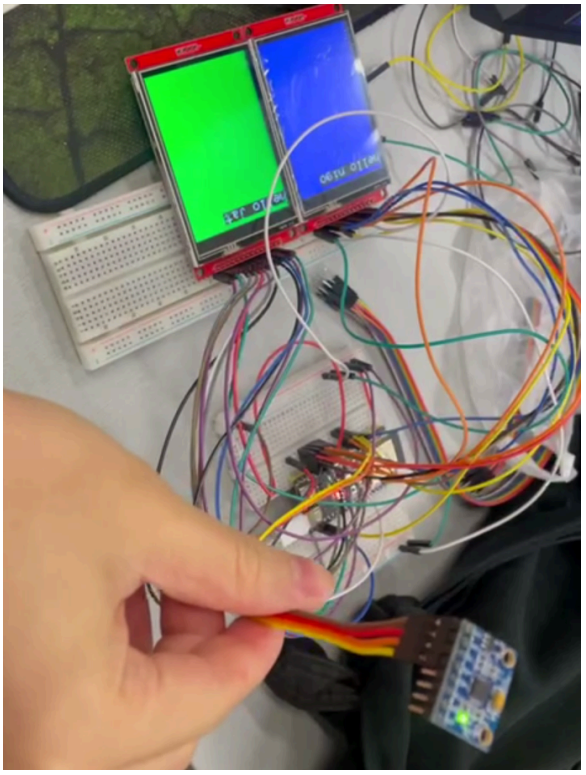


Figure 12.3: Testing the sensor(gyroscope and accelerometer) with 2 screens

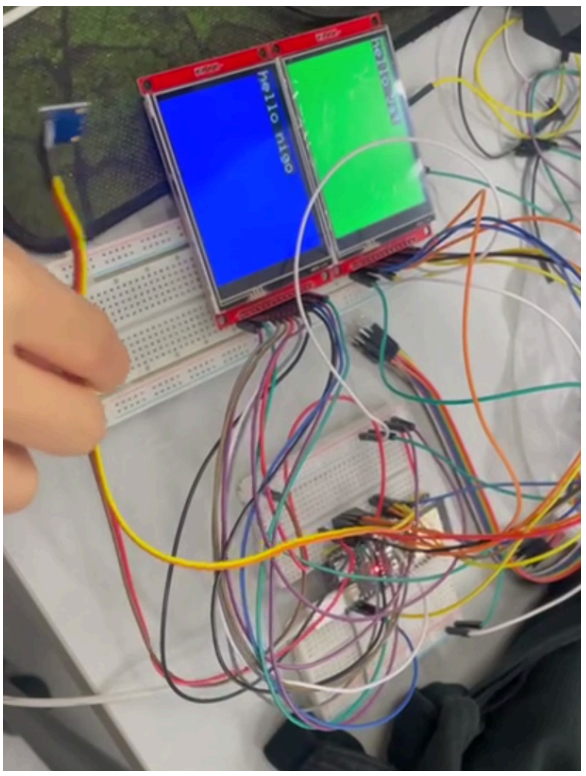


Figure 12.4: Testing the sensor(gyroscope and accelerometer) with 2 screens after moving the sensor. When the gyroscope is lifted, the screens' display swap and the orientation changes.

Software Development Process

Libraries:

```
// Importing Libraries
#include "FS.h"
#include <SPI.h>
#include <TFT_eSPI.h>
#include <TFT_eWidget.h>           // Widget library
// MPU6050 Libraries
#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>
#include <Wire.h>
```

Setting up double screens and gyroscope:

```
// Defining CS Pins for Both Screens
#define firstScreenCS 9
#define secondScreenCS 10

// Defining MPU6050 Pins
int sda_pin = 17; // GPIO8 as I2C SDA
int scl_pin = 18; // GPIO9 as I2C SCL
```

Break Mode:

```
// Check screen is on Break Mode
if (mode == "break") {
  digitalWrite(firstScreenCS, LOW);
  // Checks whether breakRun has been ran.
  if (breakRun == 0) {
    breakMode();
    breakRun = 1;
    btn4.drawSmoothButton(false, 2, TFT_BLACK); // 3 is outline width, TFT_BLACK is the surrounding background colour for anti-aliasing
    tft.setTextSize(5);
  }
  timerCount();
}
```

Lesson mode & go back function:

```
  // If mode is on lesson, check lesson
  if (mode == "lesson" and pressed == false)
  {
    for (uint8_t b = 0; b < lsnCount; b++) {
      if (lsn[b]->contains(t_x, t_y)) {
        lsn[b]->press(true);
        lsn[b]->pressAction();
      }

      else {
        lsn[b]->press(false);
        lsn[b]->releaseAction();
      }
    }
  }

  if (goBack) {
      if (btn4.contains(t_x, t_y)) {
        btn4.press(true);
        btn4.pressAction();
      }
    }

  else {
    btn4.press(false);
    btn4.releaseAction();
  }

  for (uint8_t b = 0; b < buttonCount; b++) {
    if (pressed) {
      if (btn[b]->contains(t_x, t_y)) {
        btn[b]->press(true);
        btn[b]->pressAction();
      }
    }

    else {
      btn[b]->press(false);
      btn[b]->releaseAction();
    }
  }
}
```

During Quiz Mode (Function to answer the quiz by rotation):

```
  // During Quiz Mode (Rotation and display different options when cube is rotated)
if (mode == "quiz") {
  if (quizRun == 0)
  {
    // Draw Buttons
    btn4.drawSmoothButton(false, 2, TFT_BLACK);
    quizRun = 1;
    dispOptHor(horstate);
  }
  if (g.gyro.z >= tolerance) {
    if (horstate - 1 < 0) {
      horstate = 3;
    }
    else{
      horstate = horstate - 1;
    }
    dispOptHor(horstate);
    btn4.drawSmoothButton(false, 2, TFT_BLACK);
  }
  else if (g.gyro.z <= -tolerance) {
    if (horstate + 1 > 3) {
      horstate = 0;
    }
    else{
      horstate = horstate + 1;
    }
    dispOptHor(horstate);
    btn4.drawSmoothButton(false, 2, TFT_BLACK);
  }
  if (g.gyro.y >= tolerance) {
    if (verstate - 1 < 0) {
      verstate = 3;
    }
    else{
      verstate = verstate - 1;
    }
    dispOptVer(verstate);
    btn4.drawSmoothButton(false, 2, TFT_BLACK);
  }
  else if (g.gyro.y <= -tolerance) {
    if (verstate + 1 > 3) {
      verstate = 0;
    }
    else{
      verstate = verstate + 1;
    }
    dispOptVer(verstate);
    btn4.drawSmoothButton(false, 2, TFT_BLACK);
  }
```

To check the acceleration of the cube (If the cube is lifted/moved, the screens will turn on):

```
if (hasRun == false) {
  checkDisp(a.acceleration.y);
}

// Display goes to sleep after count of 10000
if (timer > sleepTime and a.acceleration.y >= -0.5) {
  timer = 0;
  disp = false;
  hasRun = false;
  tft.sleep(disp);
}
```

To check the rotation of the cube (gyroscope):

```
void setMPU6050() {
  Wire.setPins(sda_pin, scl_pin); // Set the I2C pins before begin
  Wire.begin(); // join i2c bus (address optional for master)

  while (!Serial)
    delay(10); // will pause Zero, Leonardo, etc until serial console opens

  Serial.println("Adafruit MPU6050 test!");

  // Try to initialize!
  if (!mpu.begin()) {
    Serial.println("Failed to find MPU6050 chip");
    while (1) {
      delay(10);
    }
  }
  Serial.println("MPU6050 Found!");
```

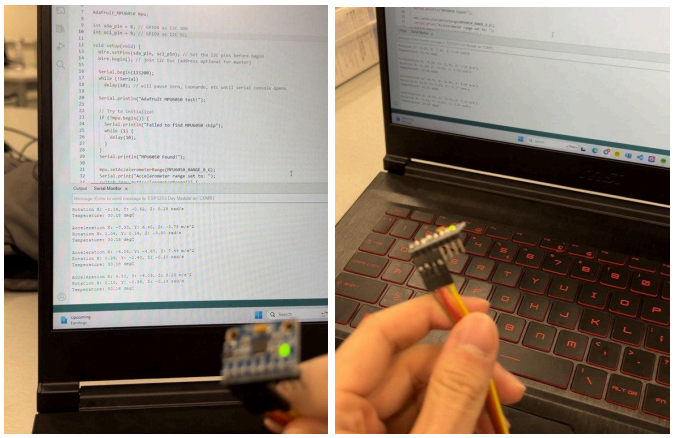


Figure 13.1: Moving the sensor and having the code detect the changes in acceleration and angular acceleration(rotation)

Calibrating the touchscreen function:

```
// Calibrating Touch Function
void touch_calibrate()
{
  uint16_t calData[5];
  uint8_t calDataOK = 0;

  // check file system exists
  if (!LittleFS.begin()) {
    Serial.println("Formating file system");
    LittleFS.format();
    LittleFS.begin();
  }

  // check if calibration file exists and size is correct
  if (LittleFS.exists(CALIBRATION_FILE)) {
    if (REPEAT_CAL)
    {
      // Delete if we want to re-calibrate
      LittleFS.remove(CALIBRATION_FILE);
    }
    else
    {
      File f = LittleFS.open(CALIBRATION_FILE, "r");
      if (f) {
        if (f.readBytes((char *)calData, 14) == 14)
          calDataOK = 1;
        f.close();
      }
    }
  }

  if (calDataOK && !REPEAT_CAL) {
    // calibration data valid
    tft.setTouch(calData);
  } else {
    // data not valid so recalibrate
    tft.fillScreen(TFT_BLACK);
    tft.setCursor(20, 0);
    tft.setTextFont(2);
    tft.setTextSize(1);
    tft.setTextColor(TFT_WHITE, TFT_BLACK);

    tft.println("Touch corners as indicated");

    tft.setTextFont(1);
    tft.println();

    if (REPEAT_CAL) {
      tft.setTextColor(TFT_RED, TFT_BLACK);
      tft.println("Set REPEAT_CAL to false to stop this running again!");
    }

    tft.calibrateTouch(calData, TFT_MAGENTA, TFT_BLACK, 15);

    tft.setTextColor(TFT_GREEN, TFT_BLACK);
    tft.println("Calibration complete!");

    // store data
    File f = LittleFS.open(CALIBRATION_FILE, "w");
    if (f) {
      f.write((const unsigned char *)calData, 14);
      f.close();
    }
  }
}
```

Timer function:

```
// Function to start the break timer
void timerCount() {
  unsigned long currentMillis = millis();

    // Check if it's time to update the countdown
  if (currentMillis - previousMillis >= interval) {
    previousMillis = currentMillis;

    static int remainingMinutes = countdownMinutes;
    static int remainingSeconds = 0;

    // Update remaining time
    if (remainingSeconds == 0) {
      if (remainingMinutes > 0) {
        remainingMinutes--;
        remainingSeconds = 59;
      }
      else {
        // Display "Time's up!" when countdown is finished
        tft.fillScreen(TFT_RED); // Clear the screen
        tft.setTextColor(TFT_WHITE); // Set text color to red
        tft.setTextSize(5);
        int xPos = (screenWidth - tft.textWidth("Time's up!")) / 2 + 50;
        int yPos = (screenHeight - tft.fontHeight()) / 2 - 35;
        tft.setCursor(xPos, yPos);
        tft.print("Time's up!");
        delay(2000);
        homeScreen();
        breakRun = 0;
        previousMillis = 0;
        remainingMinutes = countdownMinutes;
      }
    }
    else {
      remainingSeconds--;
    }
    if (mode != "menu") {
      // Update the display with the new countdown time
      int xPos = (screenWidth - tft.textWidth(formatTime(remainingMinutes, remainingSeconds))) / 2 + 50;
      int yPos = (screenHeight - tft.fontHeight()) / 2 - 35;
      tft.fillRect(xPos, yPos, tft.textWidth("00:00"), tft.fontHeight(), TFT_BLACK); // Clear previous text
      tft.setCursor(xPos, yPos);
      tft.print(formatTime(remainingMinutes, remainingSeconds));
    }
  }
}
```

Figure 14.1: The touch screen on one display transmits the output to another display

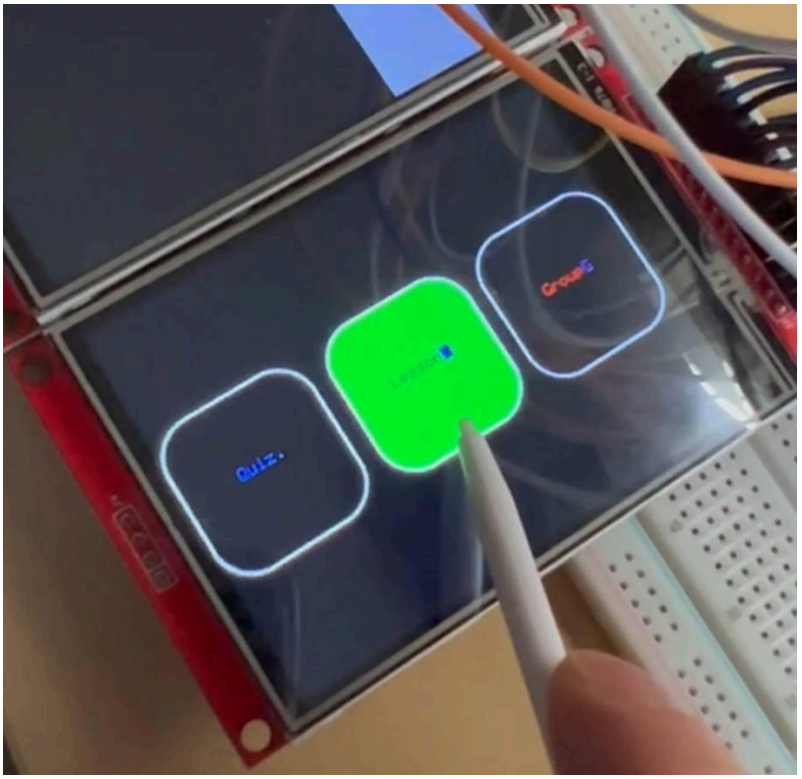


Figure 14.2: Touch screen on the buttons created

Physical Prototype

3D printed PLA vs Laser cut wood

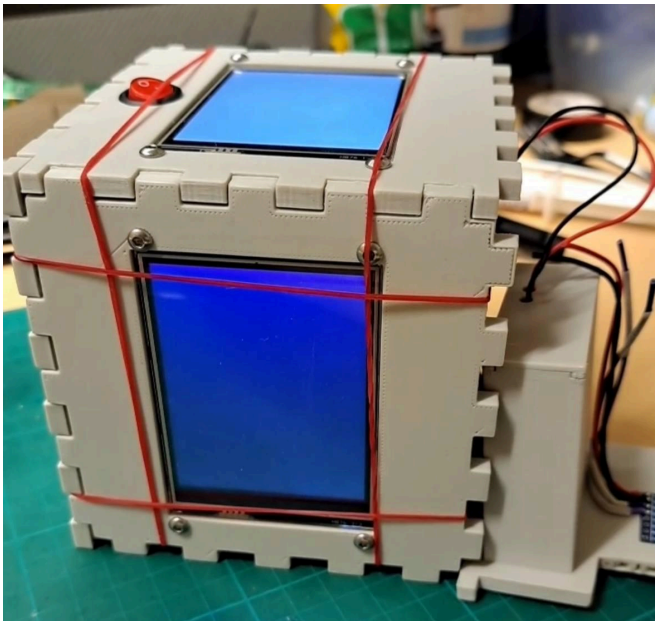


Figure 15.1: 3D printed parts put together to form cube

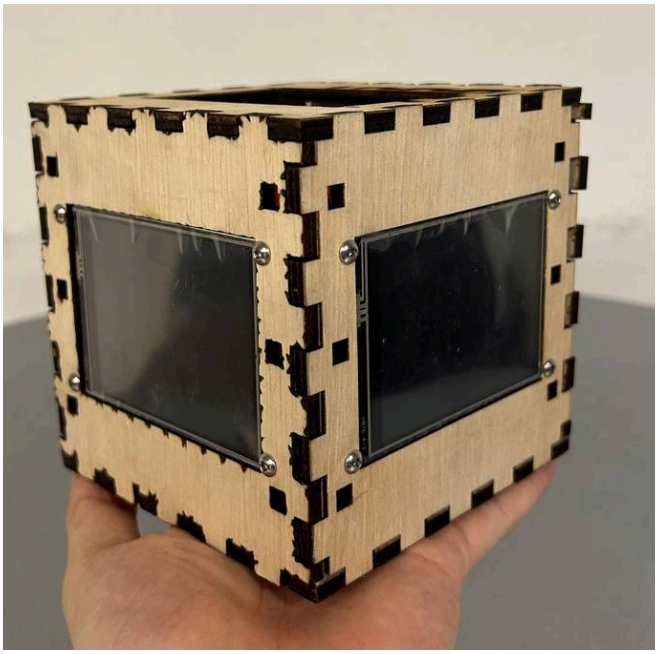


Figure 15.2: Laser cut wood parts put together to form cube

We decided to go with the 3D printed PLA as we can fit the magnets in it as compared to wood where we have to use a rubber band.

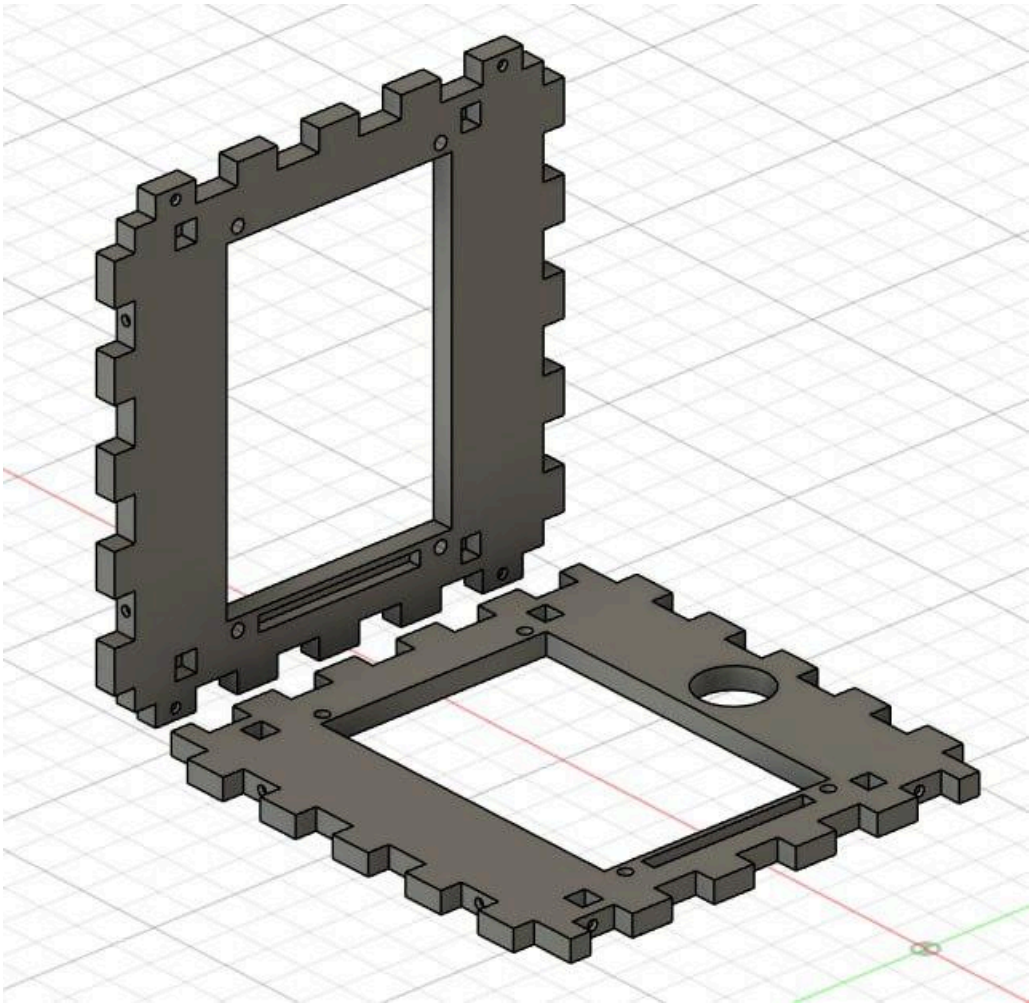


Figure 15.3: CAD design with holes for magnet placement

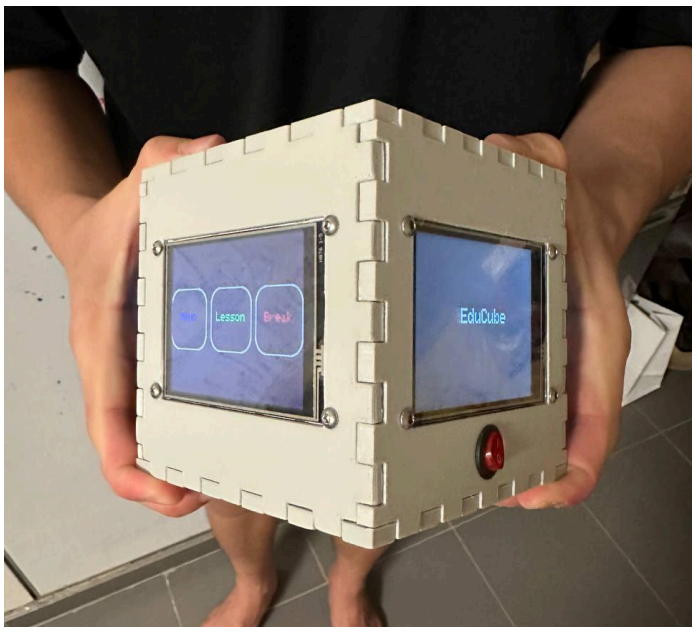


Figure 16.1: Physical EduCUBE prototype with screens working (side view)

Figure 16.2: Physical EduCUBE prototype with screens working (bottom view)

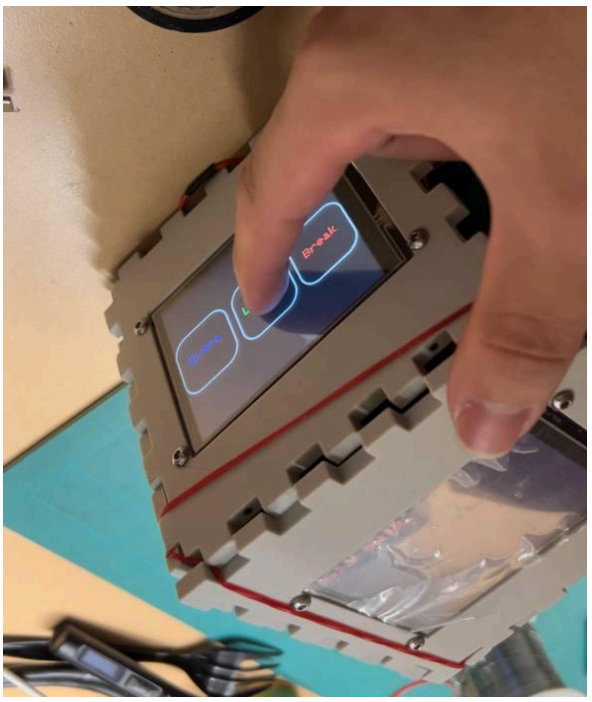


Figure 16.3: Pressing onto the Lesson Mode function

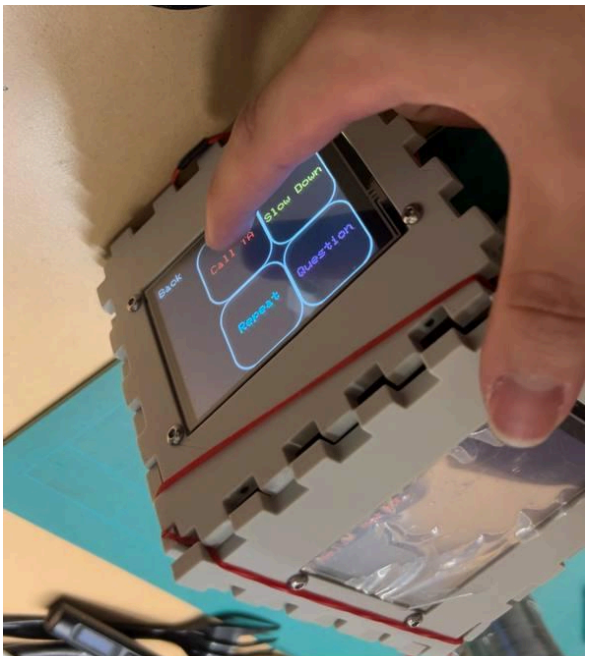


Figure 16.4: Lesson Mode shows 4 features (Call TA, Slow Down, Repeat and Question)

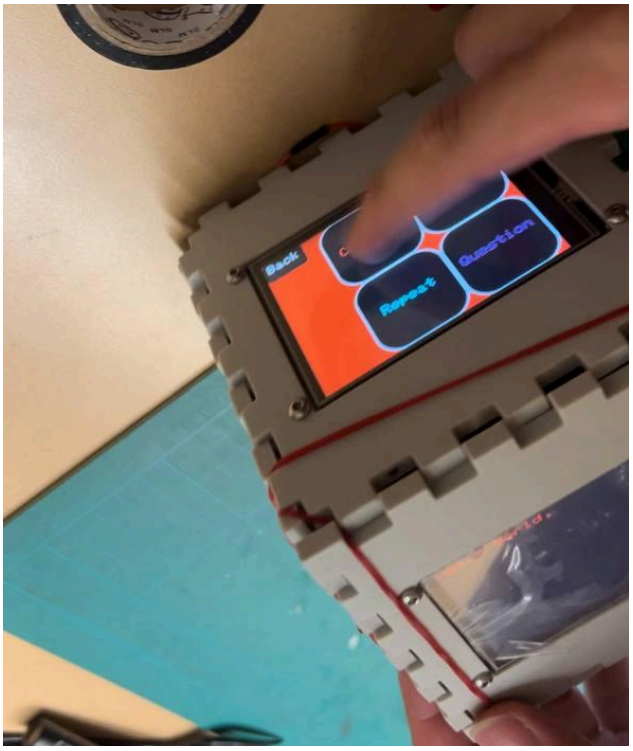


Figure 16.5: Selecting the Call TA feature

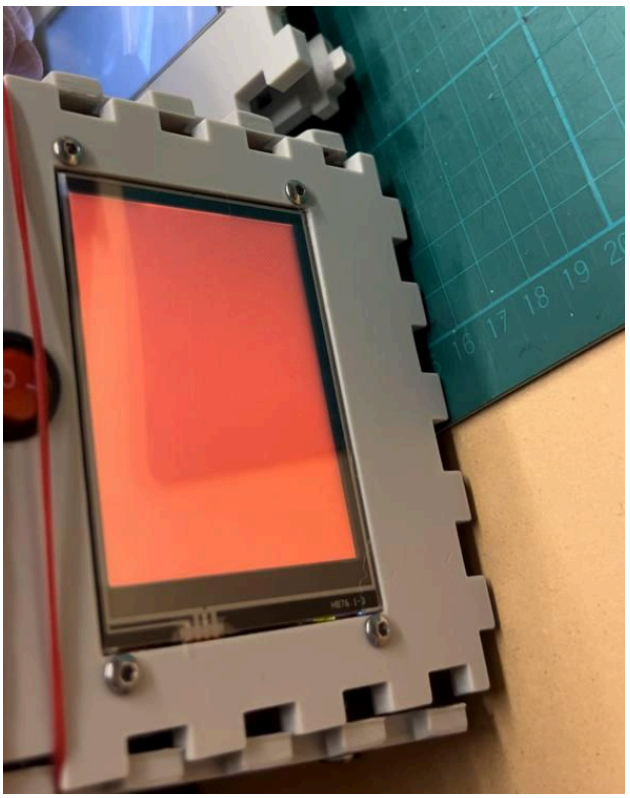


Figure 16.6: Call TA feature lights up EduCUBE red

Figure 16.7: Repeat feature lights up EduCUBE light blue

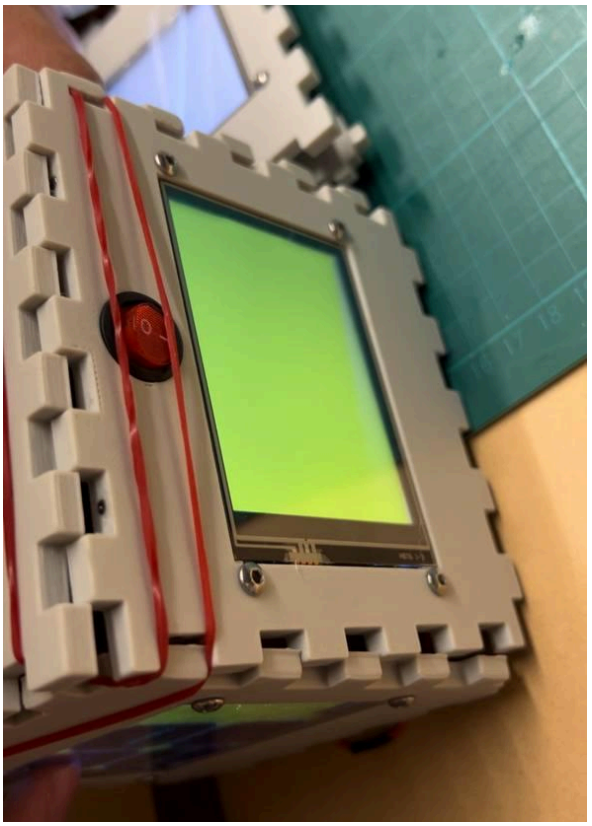


Figure 16.8: Slow Down feature lights up EduCUBE yellow

Figure 16.9: Question feature lights up EduCUBE purple

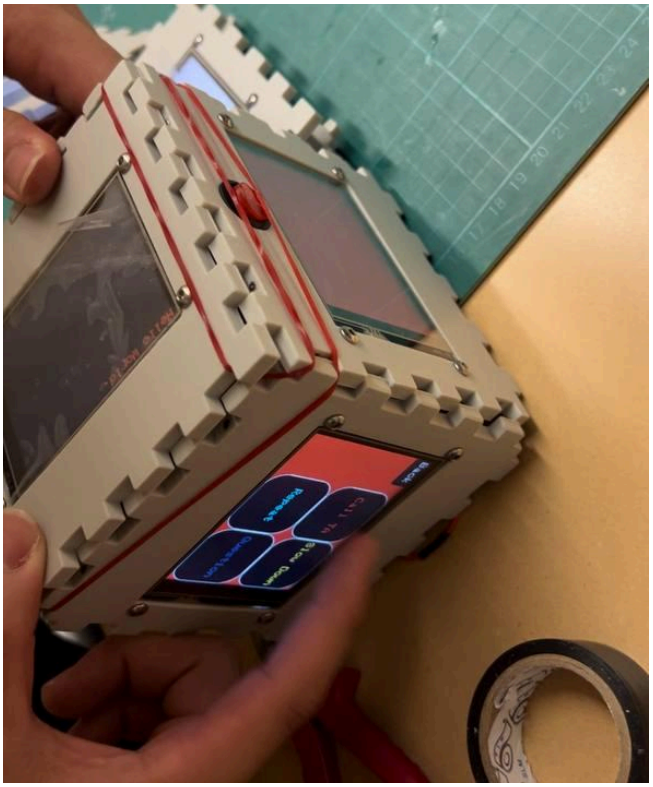


Figure 16.10: Showing all sides of the cube lighting up

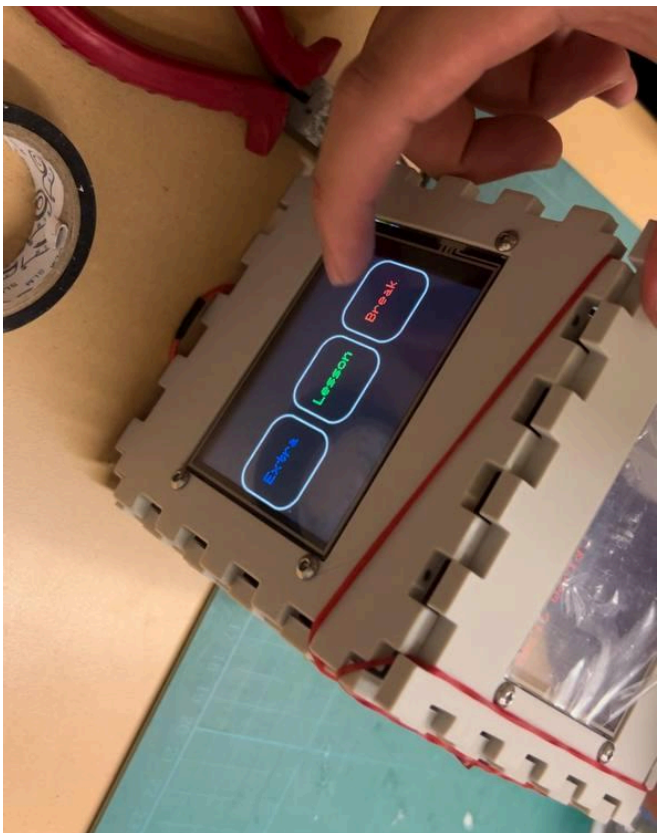


Figure 16.11: Selecting the break mode

Figure 16.12: Display shows a timer countdown

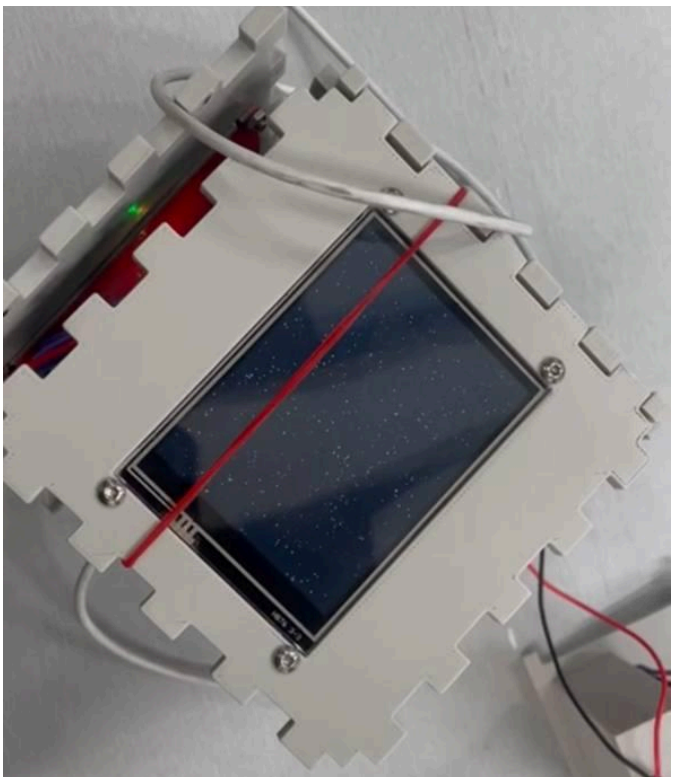


Figure 16.13: Display in sleep mode (cosmos background)

Site Model

Figure 17.1: Side view of site model

Figure 17.2: Top view of site model

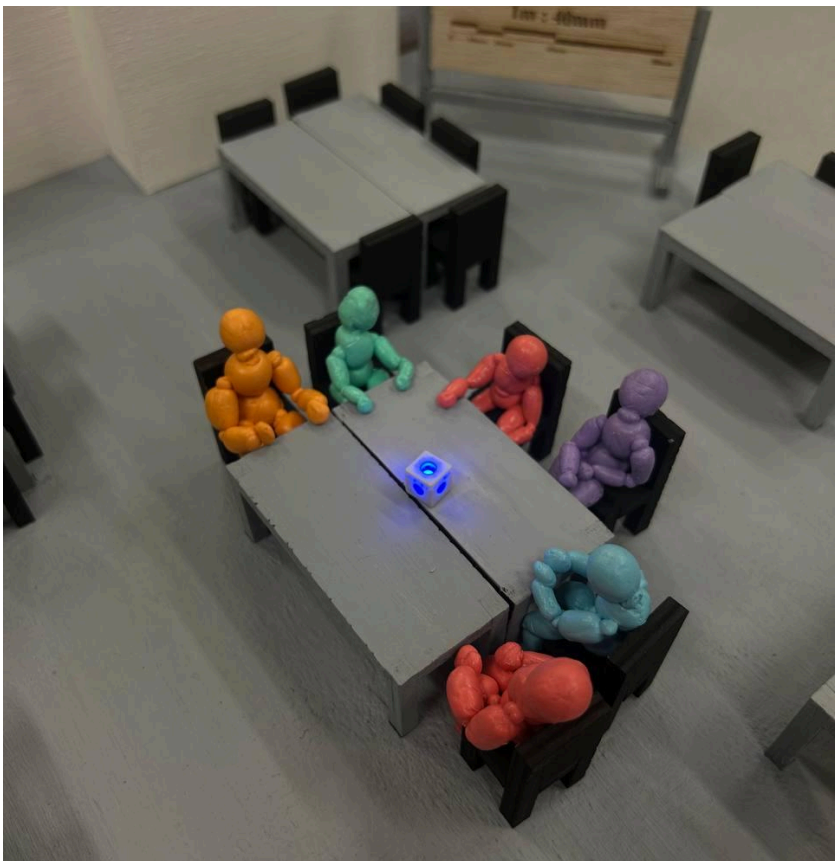


Figure 17.3: Table of 6 students made of clay

Figure 17.4: Professor Breadley at the front of the class

Figure 17.5: TA at the back of the class with an EduCUBE

Figure 17.6: Scale of our site model

Figure 17.7: Making professor Breadley stand with wires

When making the clay models, we encountered various problems such as struggling to make our model stand and making them stick together and stay in shape. We overcome these problems by improvising, such as using metal wires to poke through the clay to create an internal skeleton structure so that it can stand and using a blow dryer to dry the clay so that they can melt a little and stick together better.

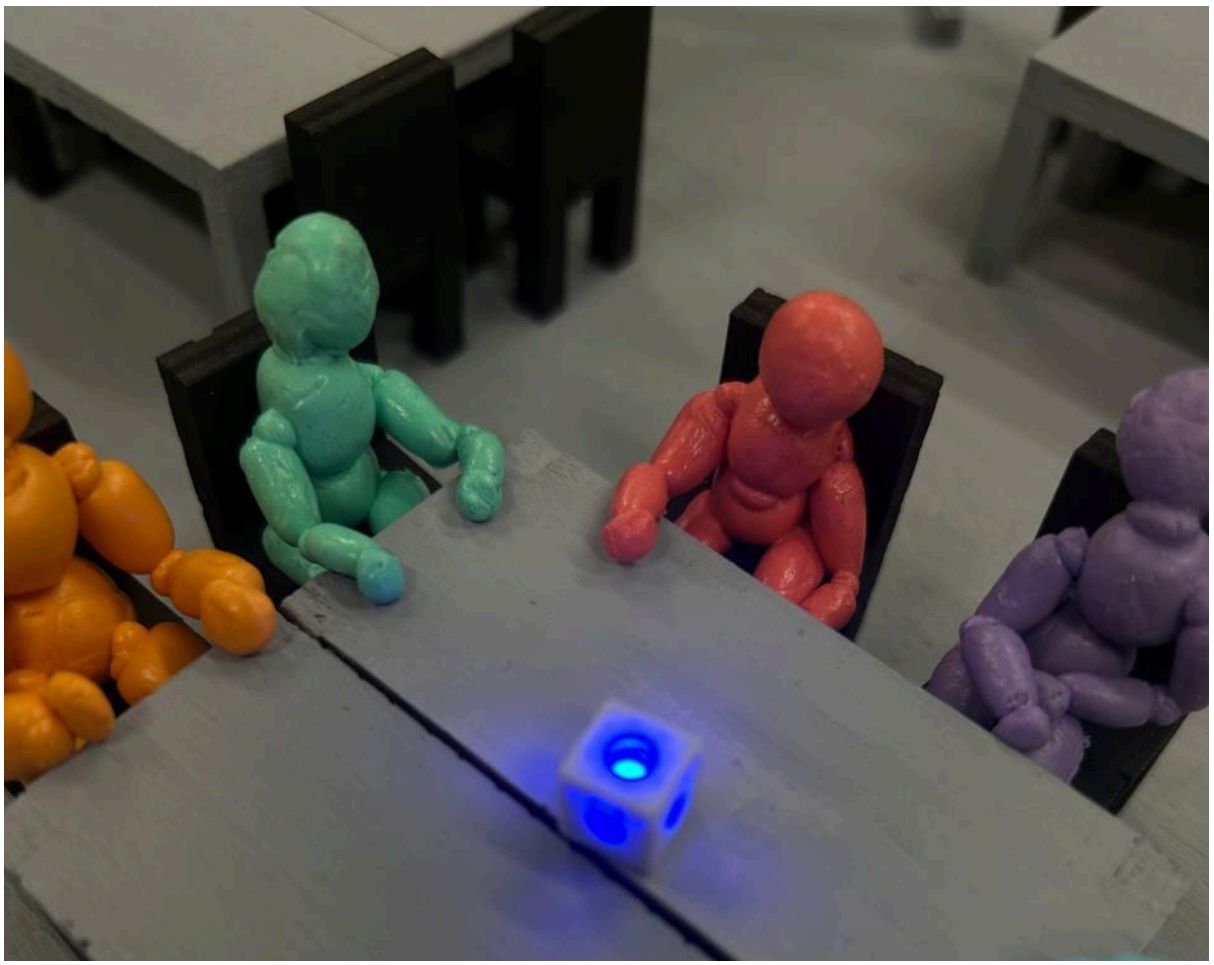


Figure 17.8: Showing the LED light in the EduCUBE model

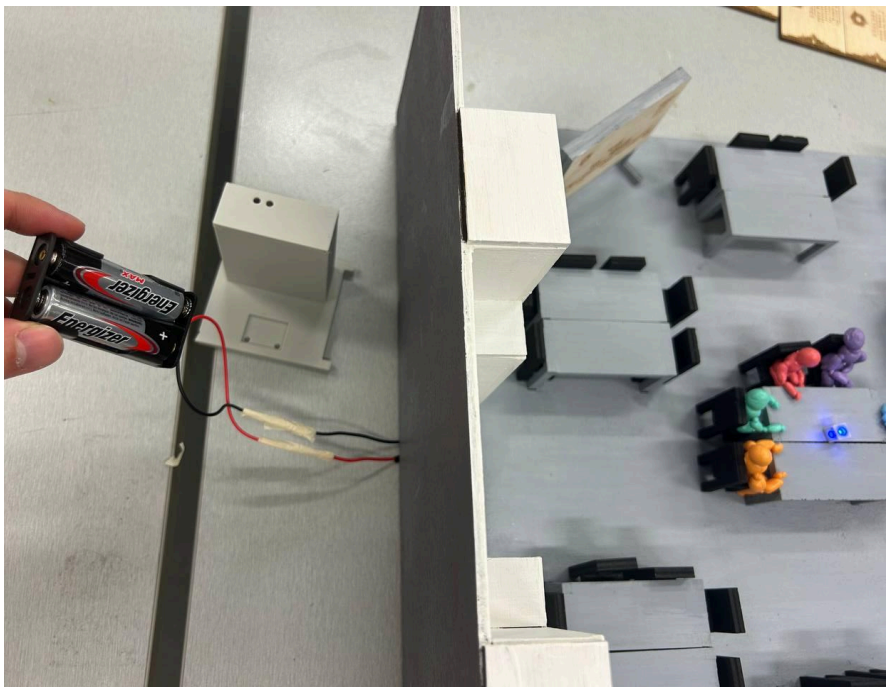


Figure 17.9: Connection and wiring of the LED

User Interaction and Feedback

After promoting our ideas to potential users of the cube, below are some of the comments and feedback we've received. In general, the cube could further be expanded by enabling features to interact within cubes. This can be through professor and student interaction, or student-to-student (cube-to-cube) interactions.

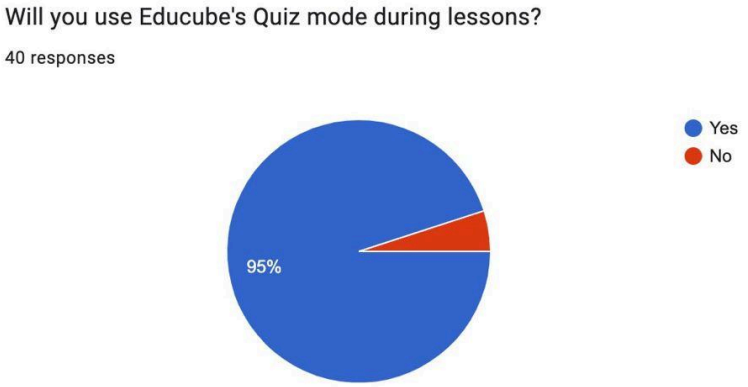


Figure 17.1: Pie chart showing survey responses for quiz mode

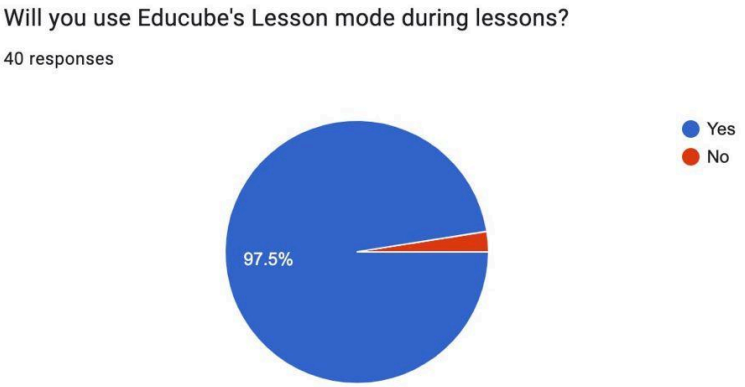


Figure 17.2: Pie chart showing survey responses for lesson mode

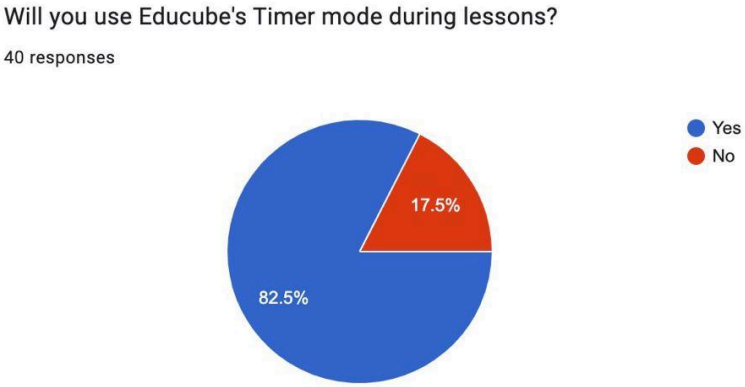


Figure 17.3: Pie chart showing survey responses for timer mode

Overall, from the 40 responses we got from students, 91.7% of them would want to use EduCUBE.
Here are the responses for any feedback/further improvements for our product.

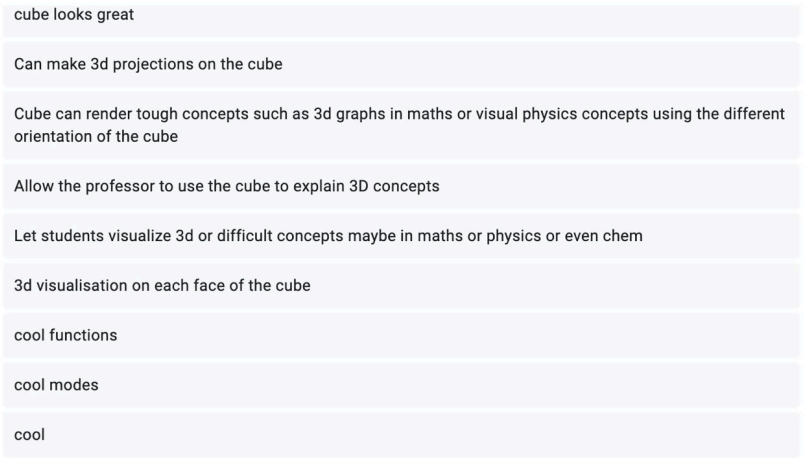


Figure 17.4: Feedback on improvements for EduCUBE

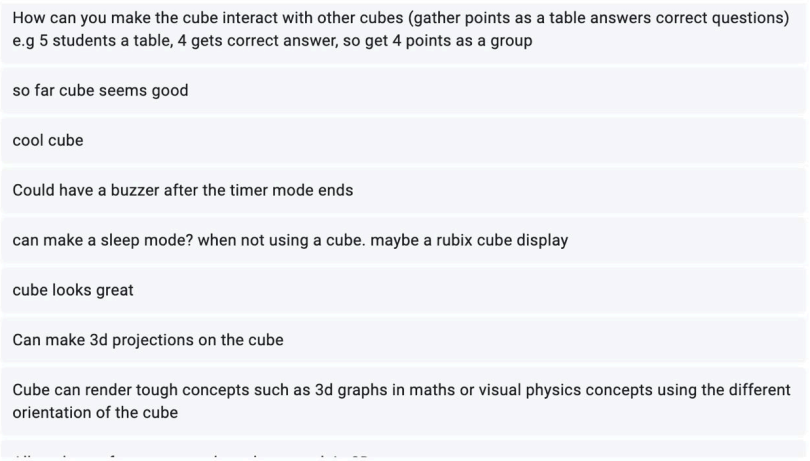


Figure 17.5: Feedback on improvements for EduCUBE

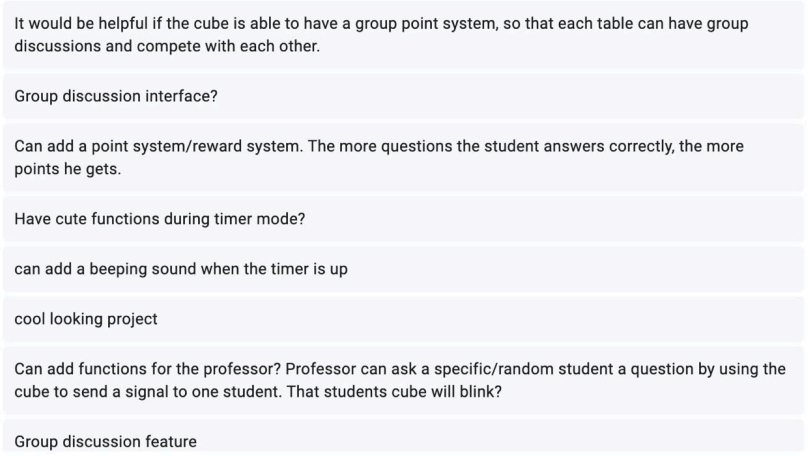


Figure 17.6: Feedback on improvements for EduCUBE

Further Ideation/ Improvements

EduCUBE can be further integrated with so many other different functions, making it a highly effective innovation. Some of the future integrations would include:

Figure 18.1: EduCUBE lights up red when student selects the wrong answer

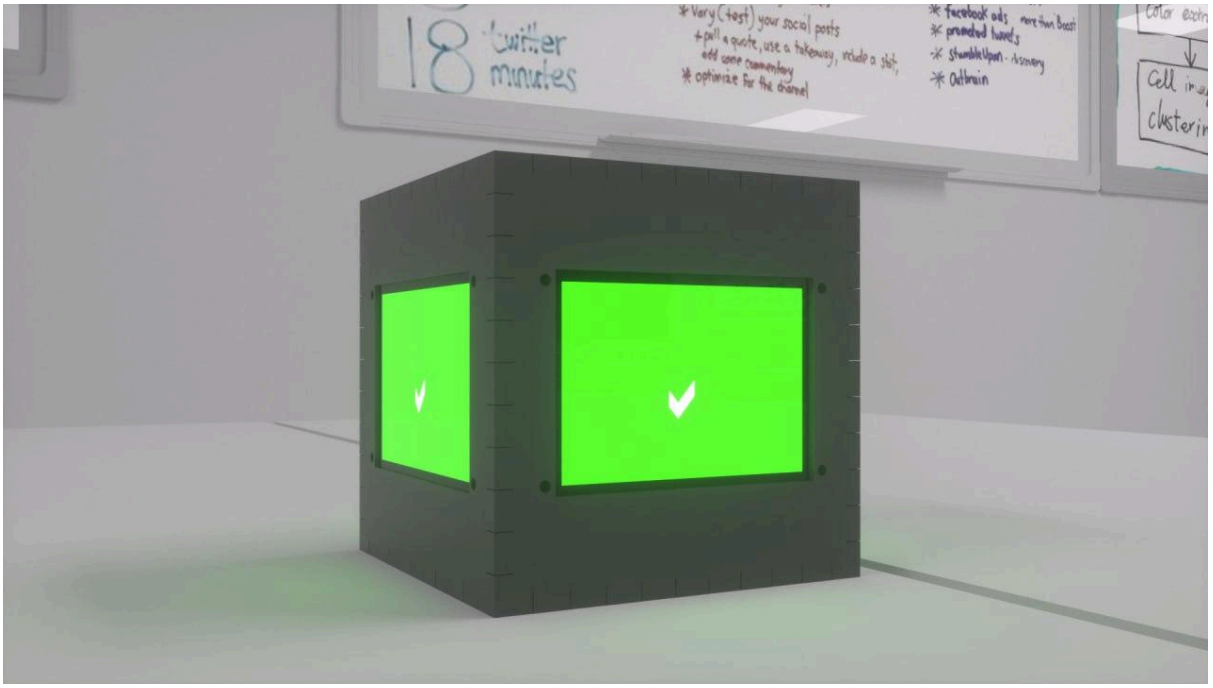


Figure 18.2: EduCUBE lights up green when student selects the correct answer

1) Lights up when a student gets a wrong answer and a correct answer

2) Wifi connectivity to collect answers for quiz time (similar to classpoint, but better! Since students will not need to use their laptop, it is less likely for them to get distracted)

3) Anonymous question and answer: collect student's questions anonymously (to help those who are reluctant and shy to ask questions so that they can still express their concern without the feeling of embarrassment! EduCUBE provides better alternative compared to raising hands)

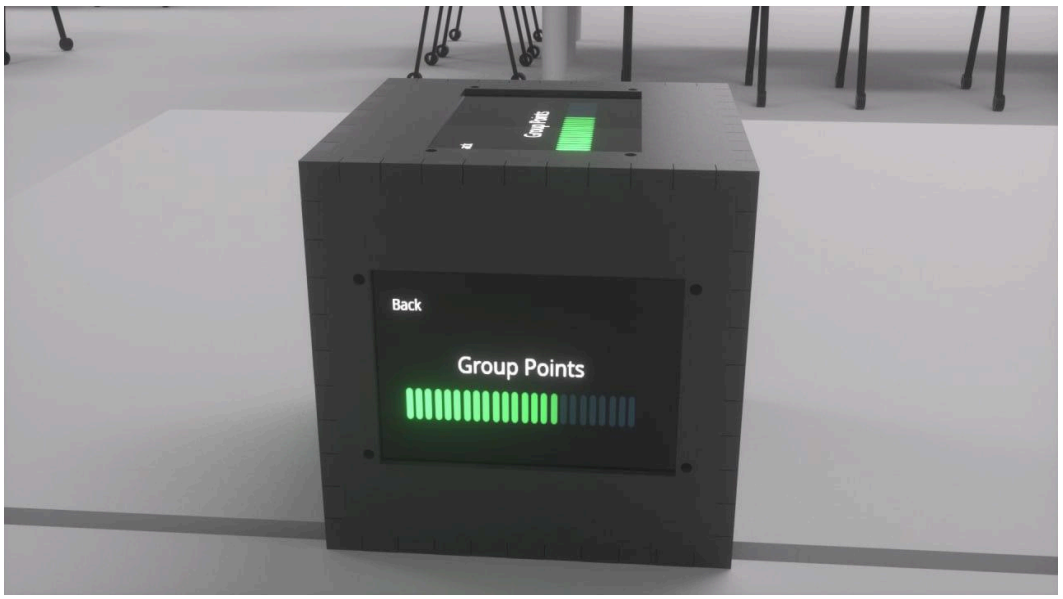


Figure 18.3: EduCUBE showing points system

Figure 18.4: Low energy level bar

4) Reward system/energy level bar (based on volume of discussions)

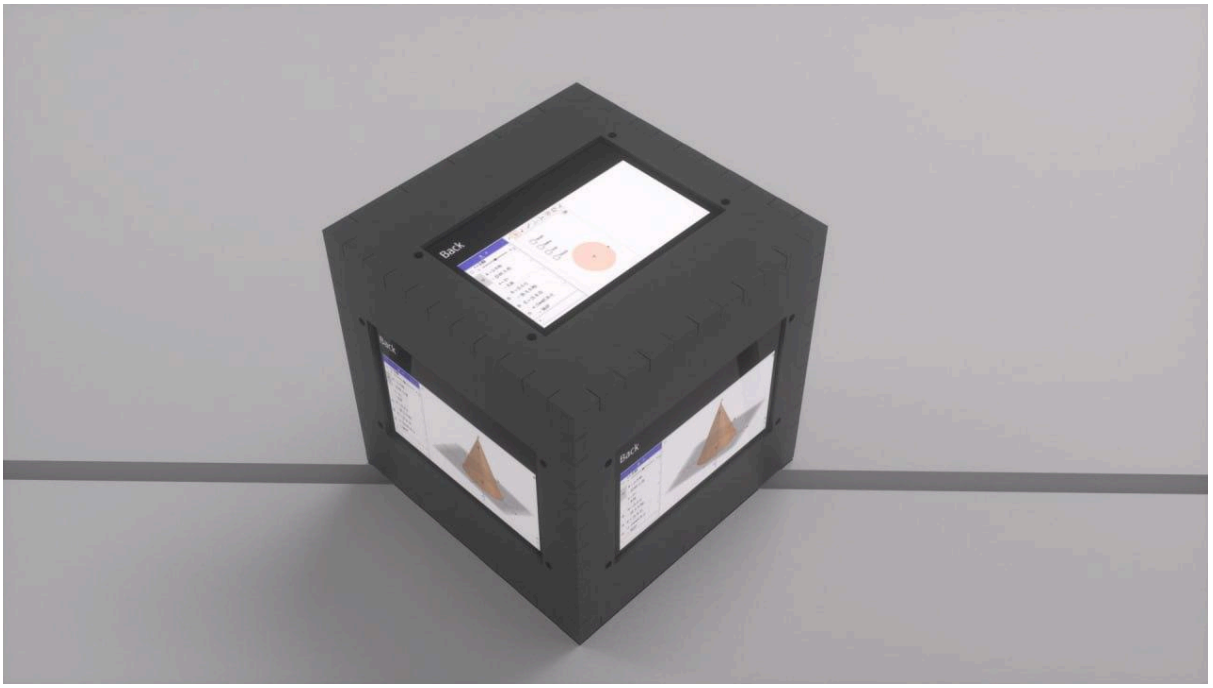


Figure 18.5: 3D projections of difficult concepts

Figure 18.6: Students having a group discussion, while energy level bar determines their “energy” through their discussion

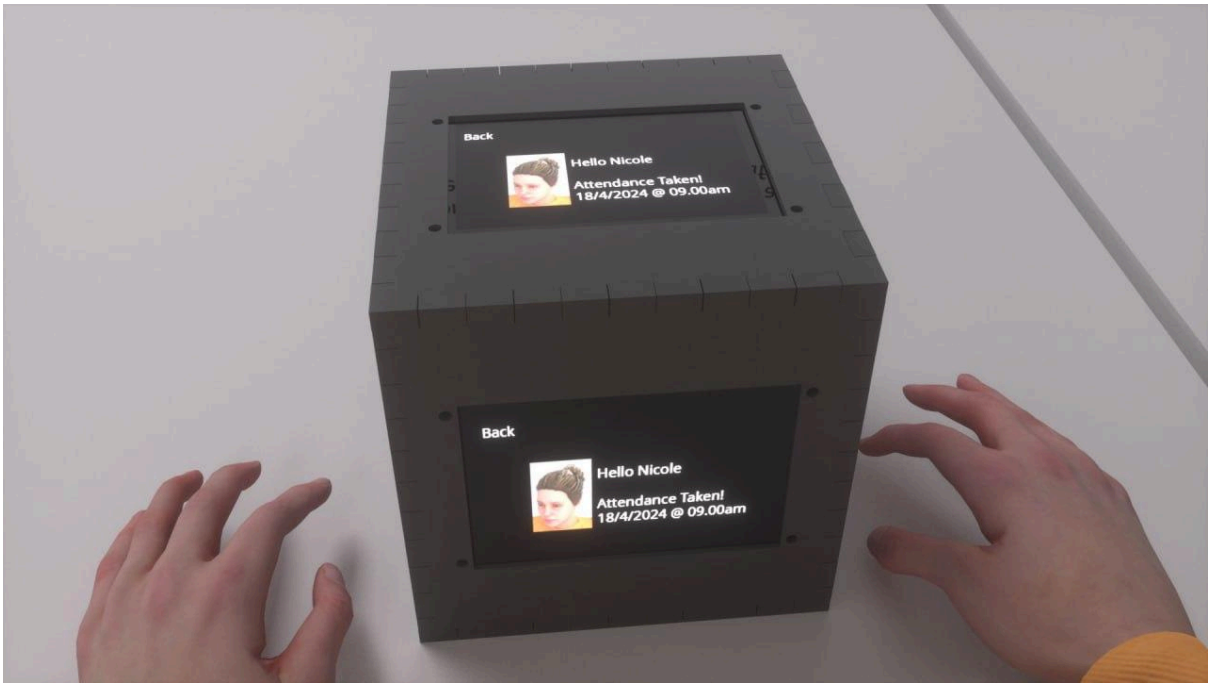


Figure 18.7: Attendance function

3) Attendance taking (integrating facial recognition to prevent students from faking attendance)

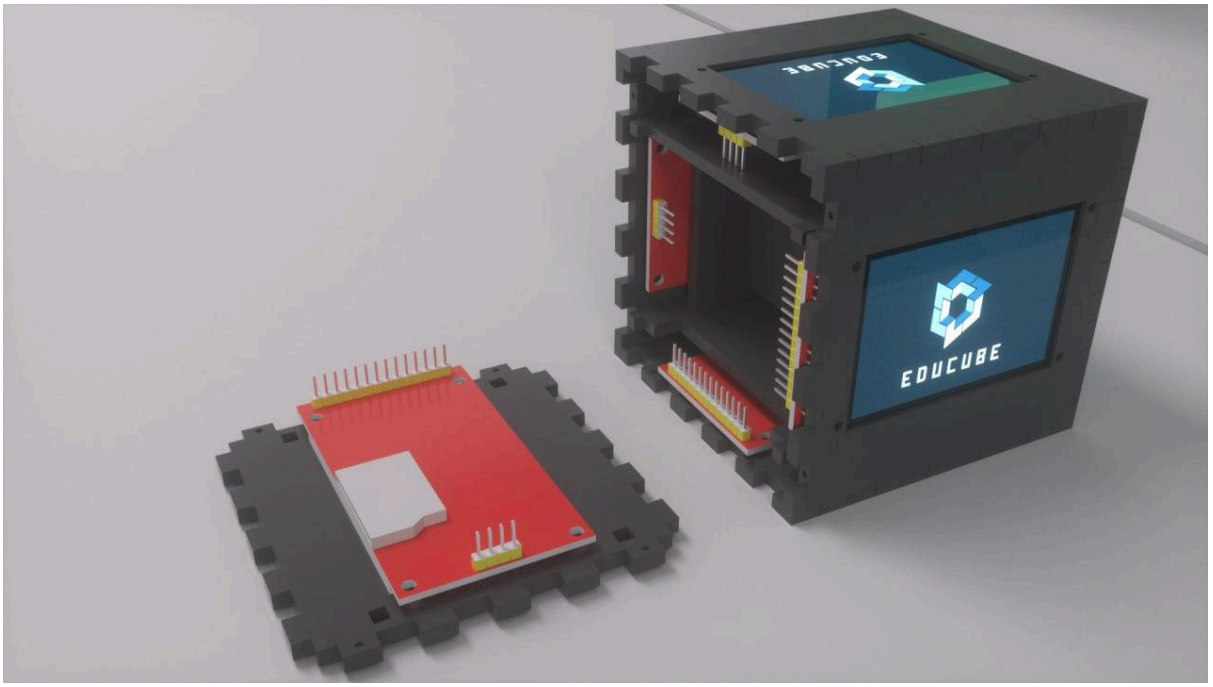


Figure 18.8: Disassembly of EduCUBE

5) Disassembly of the cube (for individual uses). So each group would have 6 display screens and can answer individually!

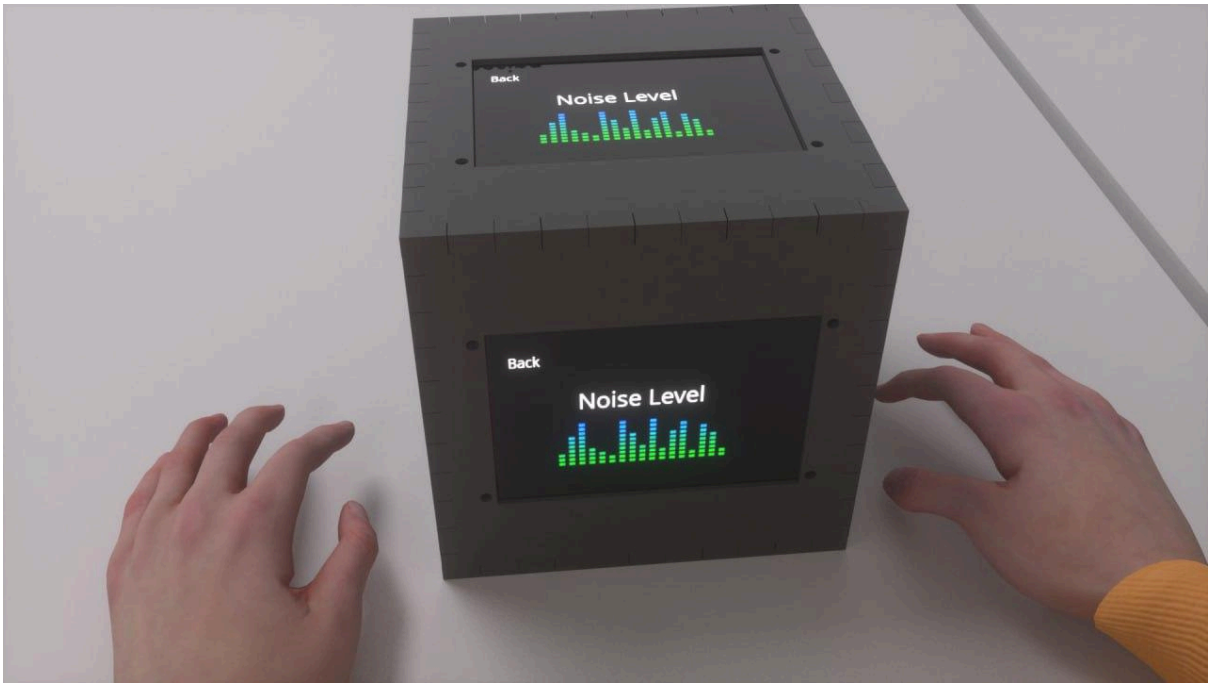


Figure 18.9: Noise control system showing the level of noise

6) Noise control through the cube (monitor class's behaviour): can send a pop-up warning when the group seated at the table is too noisy! Improve the overall conducive environment!

Summary

- The project aims to enhance the cohort classroom environment to foster better interactions among students, speakers, and teaching assistants (TAs). The focus is on improving students' attention and learning experiences by addressing different behavioural categories and environmental factors that affect classroom dynamics.
- The classroom was chosen due to its universal presence across educational institutions globally, making the findings and solutions widely applicable.

Design Solution: EduCUBE

- EduCUBE is an innovative educational tool designed to enhance learning interactions through three operational modes:
    1. Quiz Mode: Allows students to participate in quizzes by interacting with the cube.
    2. Lesson Mode: Facilitates interaction with professors or TAs which enables students to request attention from TAs, ask the professor to slow down, repeat points, or ask questions discreetly.
    3. Break Mode: Features a timer to track break durations.

Inspiration and Design Process

- Inspired by Japanese Sashimono woodworking, the EduCUBE integrates traditional craftsmanship with modern technology. The design process involved extensive research into woodworking techniques, meticulous planning, and integration of technological components like sensors and screens, ensuring both functionality and aesthetic appeal.

Hardware and Software Development

- The EduCUBE features intricate internal components like stripboards, ESP32 microcontrollers, and sensors (gyroscope and accelerometer), necessitating precise wiring and assembly to ensure functionality within a compact space. Challenges included managing wire integrity and ensuring secure, aesthetically pleasing joints.
- The software for EduCUBE includes functionalities to manage double screens, gyroscopes for orientation change, and touchscreen calibration. Additional functionalities involve quiz participation through cube rotation and acceleration sensing.

Prototype Testing and Further Ideation

- Initial testing has led to ideas for further enhancements such as:
    1. Integration of lighting cues for quiz responses.
    2. Facial recognition for attendance verification.
    3. Anonymous question features to encourage participation.
    4. Disassembly features for individual or group use.

5. Noise control and behaviour monitoring functionalities.

Conclusion

- EduCUBE represents a significant innovation in educational technology, with the potential for broad application and scalability in educational settings globally. Future improvements could further increase its utility and effectiveness in enhancing classroom learning environments.