Smart Vase (Smart Greenhouse) Project

Automated plant care with sensors, pumps, grow light, and a Bluetooth mobile app

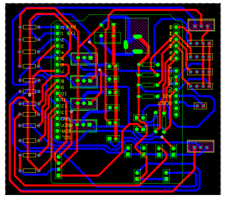
Shayan Doroudiani • Mositto Robotics Academy
Teachers: Mr. Kalehr • Ms. Alizadeh

Agenda

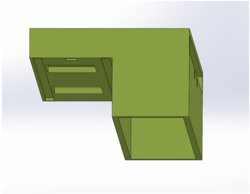
What we built and what you'll see

1. Introduction: What is a smart greenhouse?
2. Project goals & design constraints
3. Hardware: sensors, pumps, fan, and grow light
4. Circuit + PCB design (Altium)
5. Arduino firmware (auto/manual modes)
6. Mobile app (MIT App Inventor) + Bluetooth
7. Enclosure design (SolidWorks) + assembly
8. Results, challenges, and future upgrades

Soil moisture sensing (YL-69)



Custom PCB routing



SolidWorks enclosure model

Introduction

What is a smart greenhouse?

A smart greenhouse uses sensors + automation to ke

In this project, we built a compact “smart vase” system that monitors light

Main parts of the project:

- Enclosure (box) + water tank
- Circuit + custom PCB
- Arduino programming (auto/manual)
- Mobile app (Bluetooth control)

Project Goals

Why we built it

Save water

Water only when soil moisture is low (prevents ov

Healthier growth

Maintain better light + temperature conditions fo

Prevent drying

Automated reminders and control reduce th

Tools & Materials

Hardware + software used in the build

Hardware

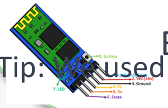

4× 12V mini pumps


12V grow LED strip


12V fan


Soil moisture sensors (YL-69)


LDR (light sensor) + thermistor

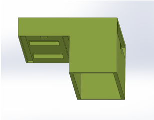

Bluetooth module (HC-05 style)

Software


Arduino IDE (firmware)


MIT App Inventor (mobile app)

Altium Designer (PCB design)


SolidWorks (enclosure design)

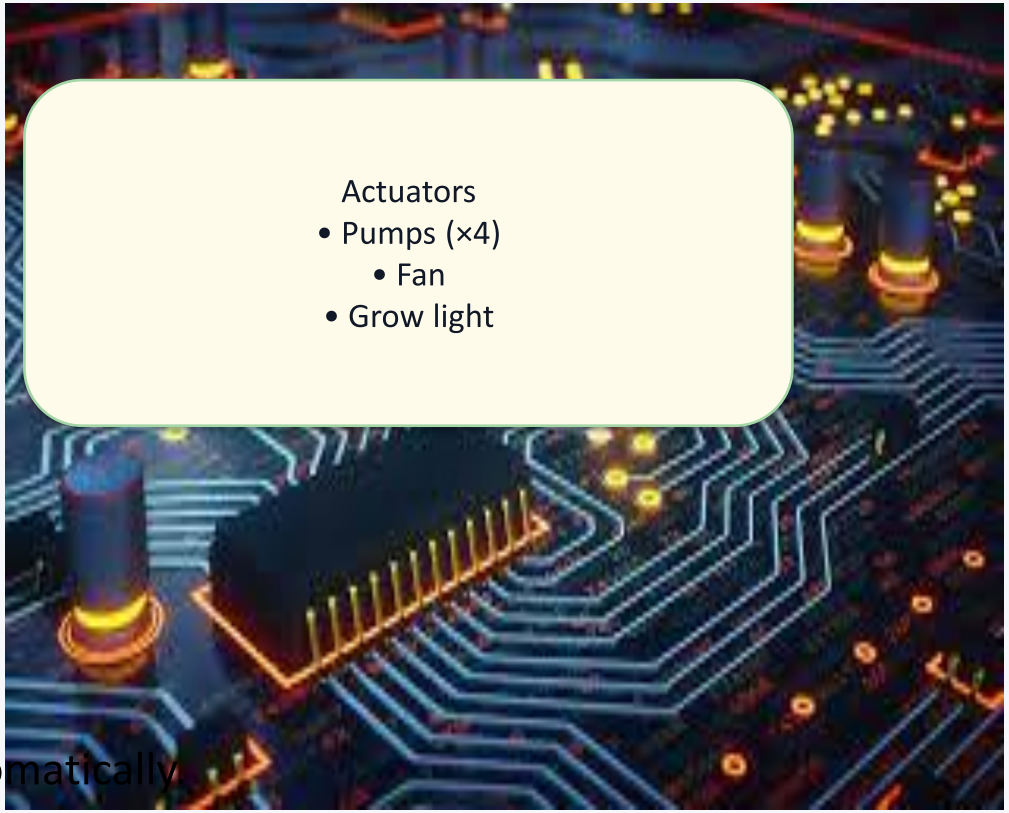
Tip: used a 12V power rail for actuators and regulated logic signals through the Arduino + driver transistors.

System Architecture

How everything connects (high-level)

Sensors
• Soil moisture (×4)
• Light (LDR)
• Temperature (thermistor)

Arduino UNO (Decision + Control)



Actuators
• Pumps (×4)
• Fan
• Grow light

Mobile App (Bluetooth)

Two operating modes:

- Auto: Arduino reads sensors and controls pumps/fan/light automatically.
- Manual: App sends commands over Bluetooth.

Sensors

What we measure to make decisions

Soil Moisture (YL-69)

Detects when soil is dry → triggers watering.

Example thresholds (tuned during testir

• Moisture < 300 = dry

• Light < 300 = dark

Light (LDR)

Measures brightness → controls grow light when needed.

Example thresholds (tuned during testir

• Moisture < 300 = dry

• Light < 300 = dark

Temperature (Thermistor)

Measures heat → can trigger fan for airflow.

Example thresholds (tuned durir

• Moisture < 300 = dry

• Light < 300 = dark

Actuators

What the system controls

Water Pumps (×4)

Each pot can be watered independently.

Driver circuit: IRF740 MOSFET + resisto

Grow Light (12V)

Adds light when the environment is too dark.

Driver circuit: IRF740 MOSFET + resisto

Fan (12V)

Improves airflow and helps regulate temperature.

Driver circuit: IRF740 MOSFET + re

We used a separate 12V rail for motors/LEDs and shared a common ground with the Arduino to avoid unstable readings.

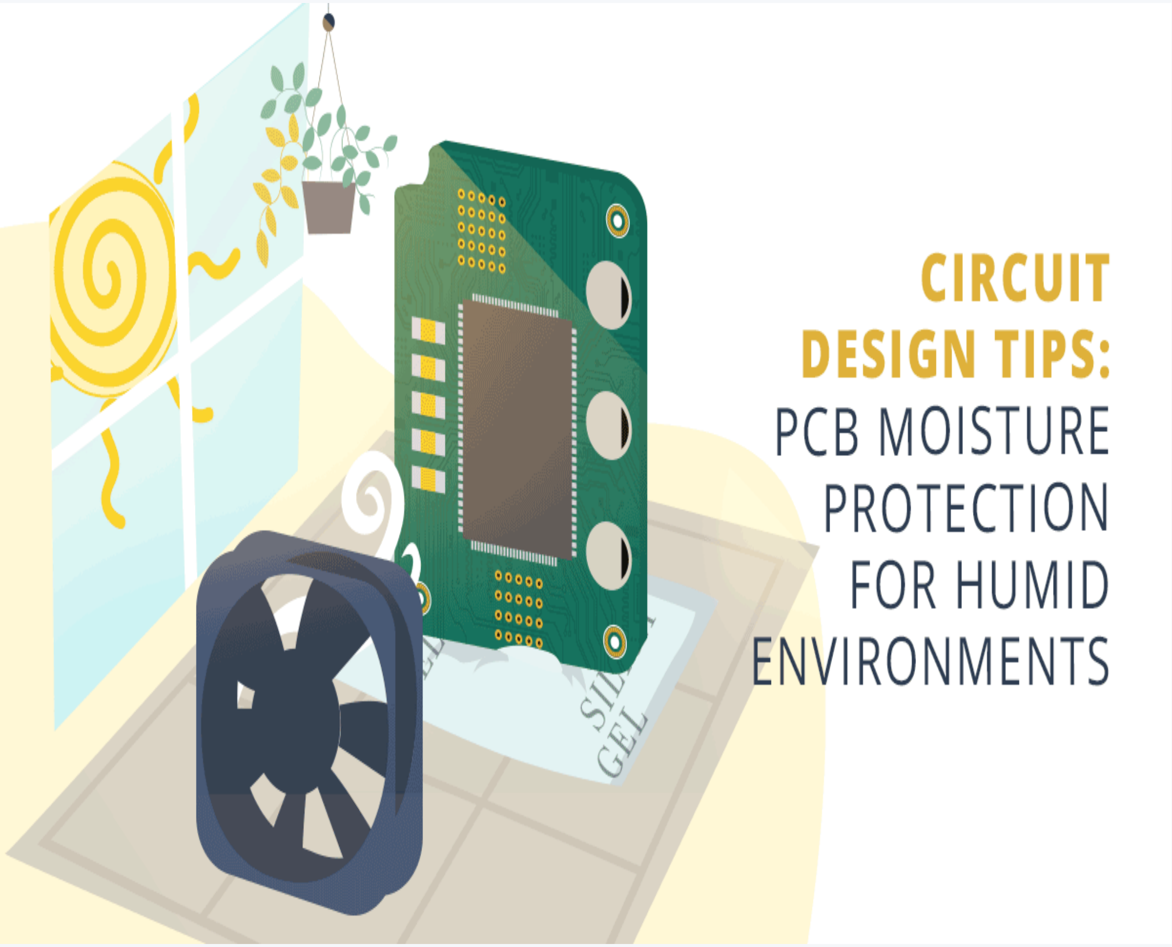
Circuit + PCB Design

Designed in Altium Designer

ALTIUM
DESIGNER

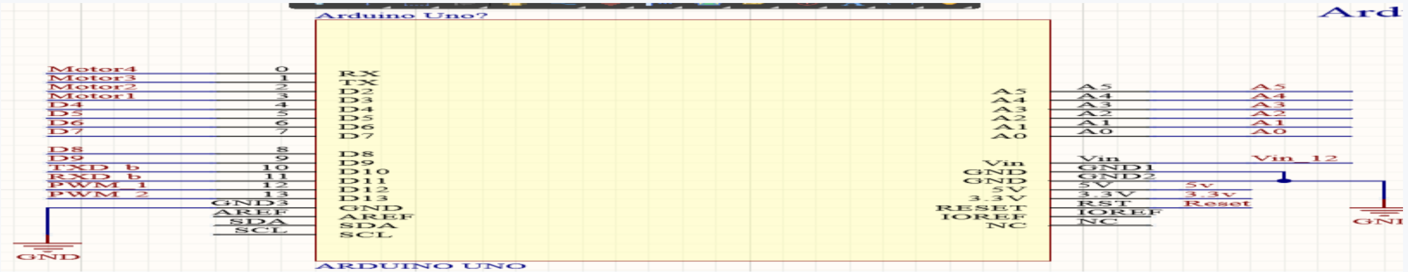
Why a custom PCB?

- Cleaner wiring and easier debugging
- More reliable connections for multiple sensors
- Dedicated MOSFET drivers for 12V pumps/fan/light
- Compact form factor that fits inside the enclosure

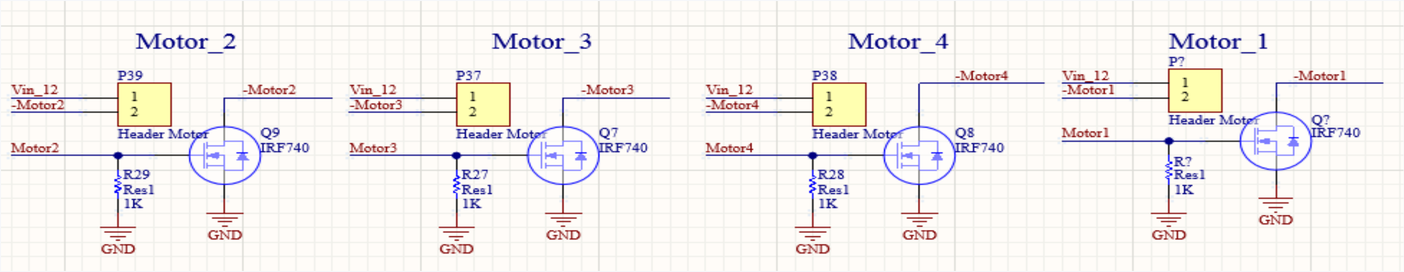


Schematic Highlights

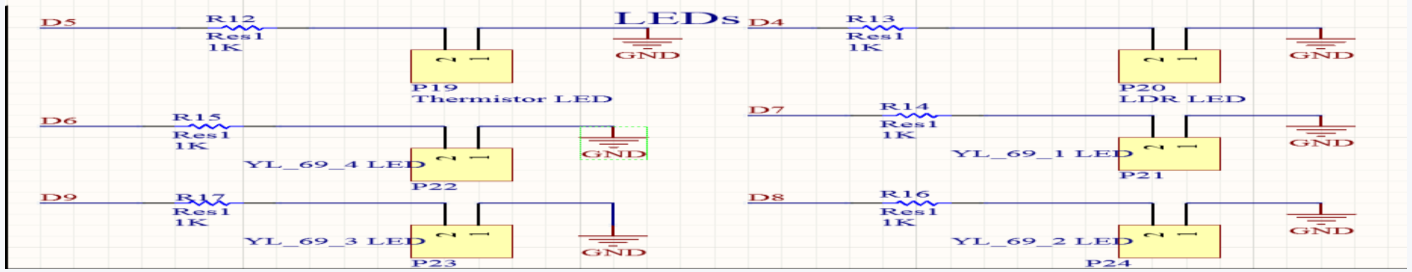
Key sections of the circuit



Arduino pin mapping (digital + analog)



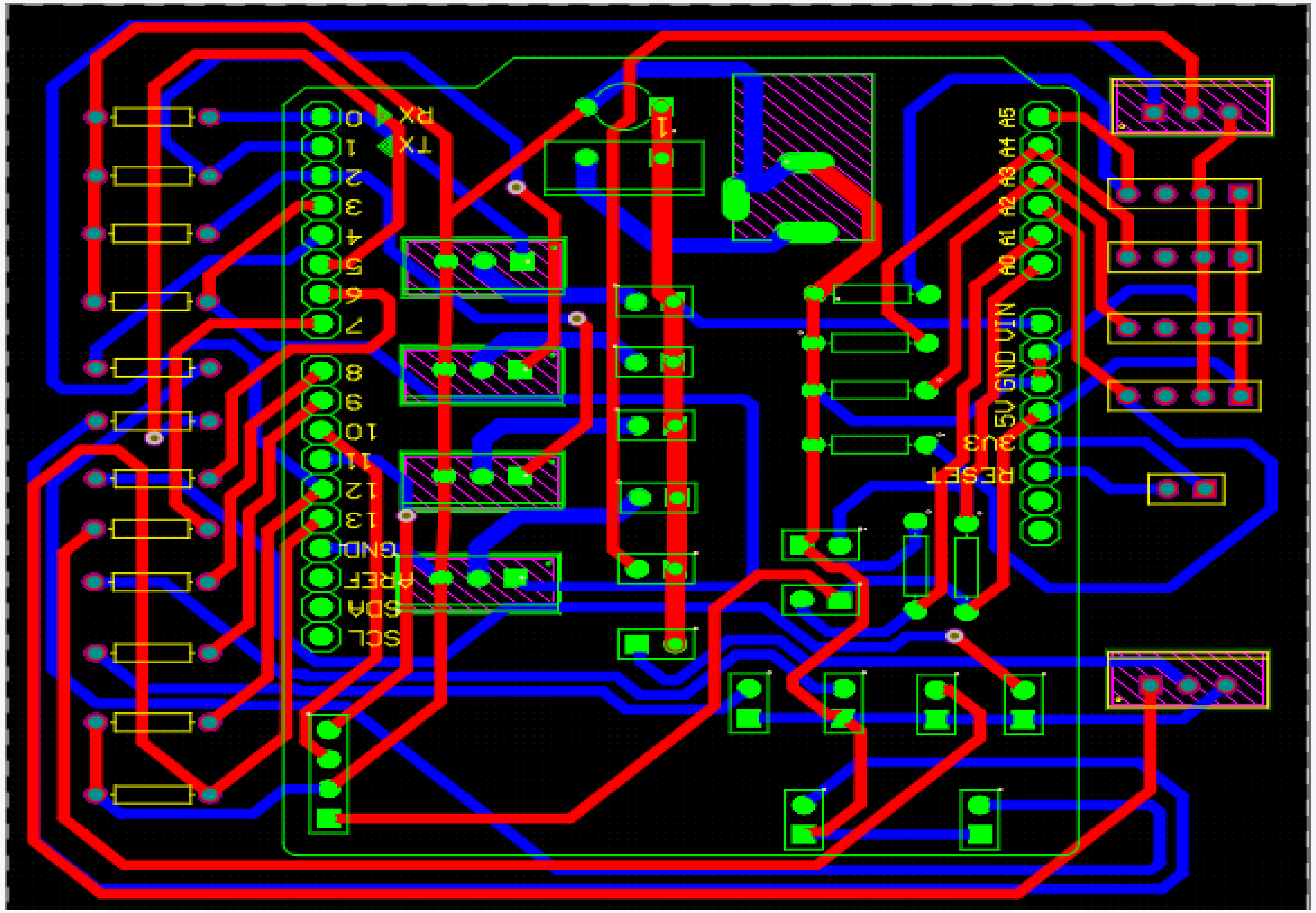
Motor drivers (MOSFET switches)

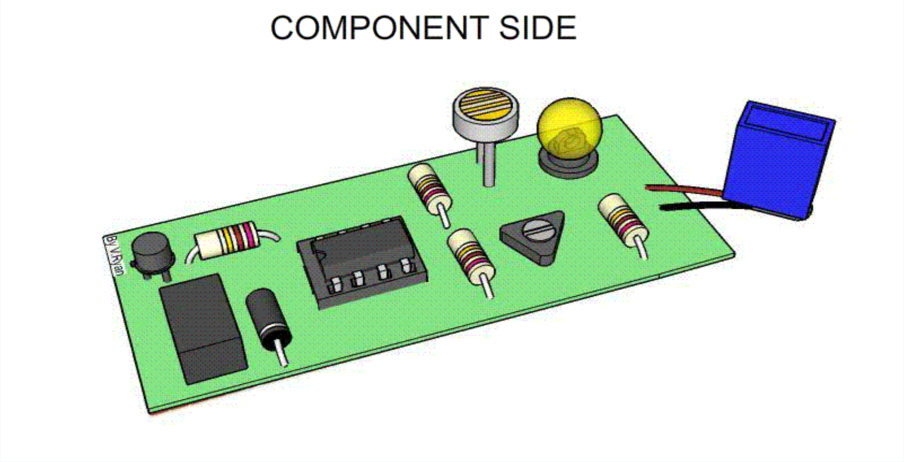


Indicator LEDs for sensors

PCB Layout

Routing + 3D view





Moisture protection

Since this project works near water and soil, we:

- Kept high-current traces wider
- Added spacing between 12V and signal traces
- Used an enclosed box to reduce splashes

Arduino Firmware

Auto / manual logic (highlights)

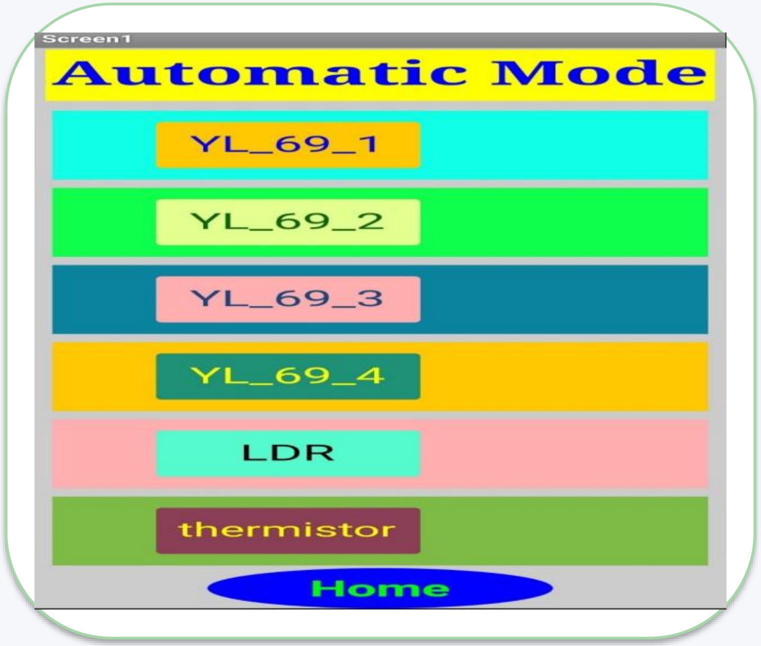
Reads 4 moisture sensors + LDR + thermistor
Auto mode: compares readings to thresholds → turns pumps/fan/light on/off
Manual mode: app can directly toggle outputs via Bluetooth commands
Safety: pumps are time-limited to avoid flooding

```
// Example: read moisture and drive one pump in Auto mode
int moisture = analogRead(A0);
if (autoMode) {
  if (moisture < DRY_THRESHOLD) {
    digitalWrite(PUMP_1, HIGH);
    delay(1200);          // short burst watering
    digitalWrite(PUMP_1, LOW);
  }
}
```

Mobile App (Bluetooth)

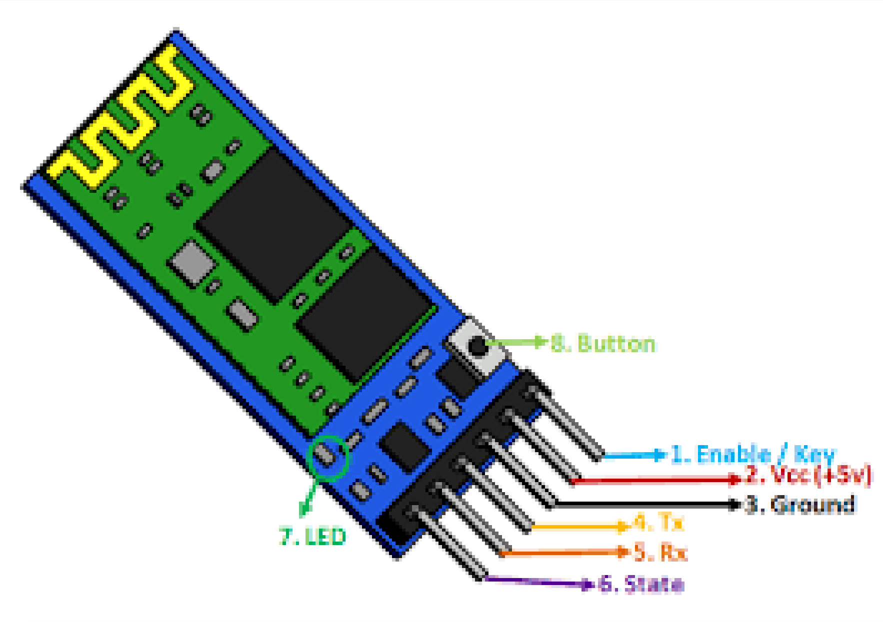
Built with MIT App Inventor

Connects to the Bluetooth module and sends single-character commands.
Home screen: connect + choose Auto or Manual.
Manual mode: directly toggle Pump1–Pump4, Fan, and Lamp.
Auto mode: displays live readings and system status (on/off).



Bluetooth Communication

Simple command protocol



Module setup

• VCC → 5V, GND → GND
• TX/RX connected to Arduino serial pins
• Tested pairing and stable baud rate before final assembly

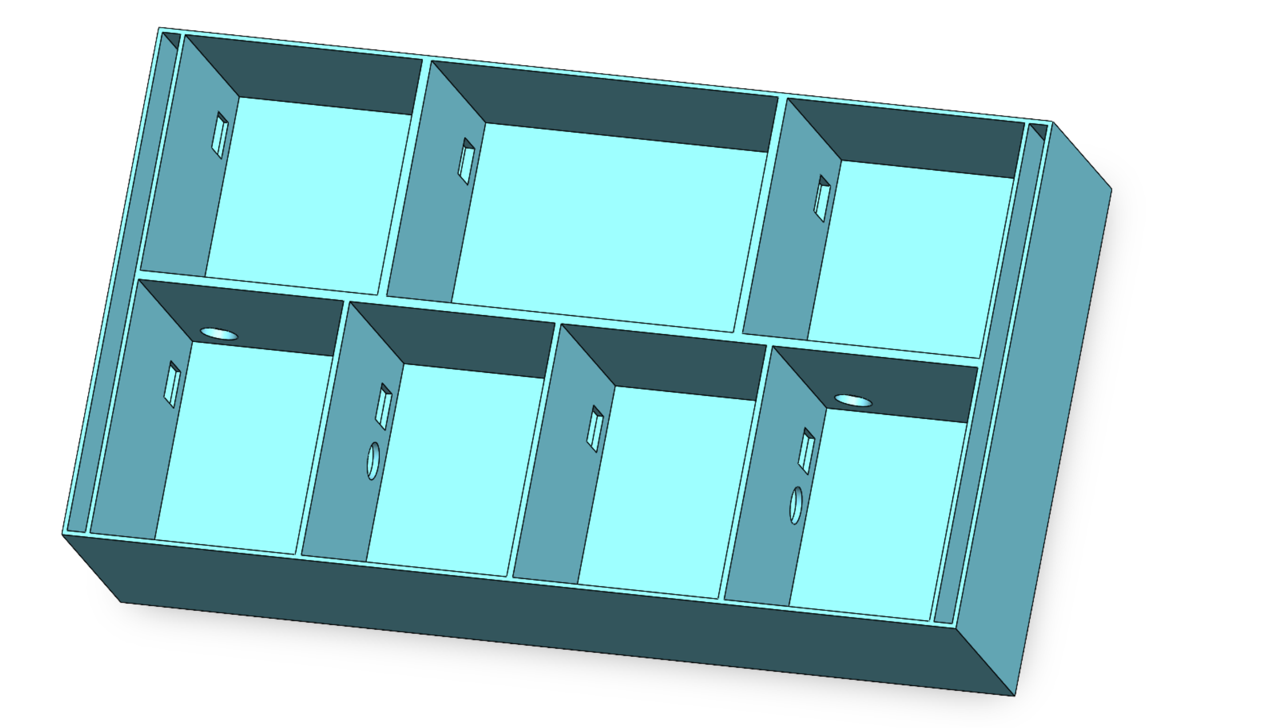
Command examples

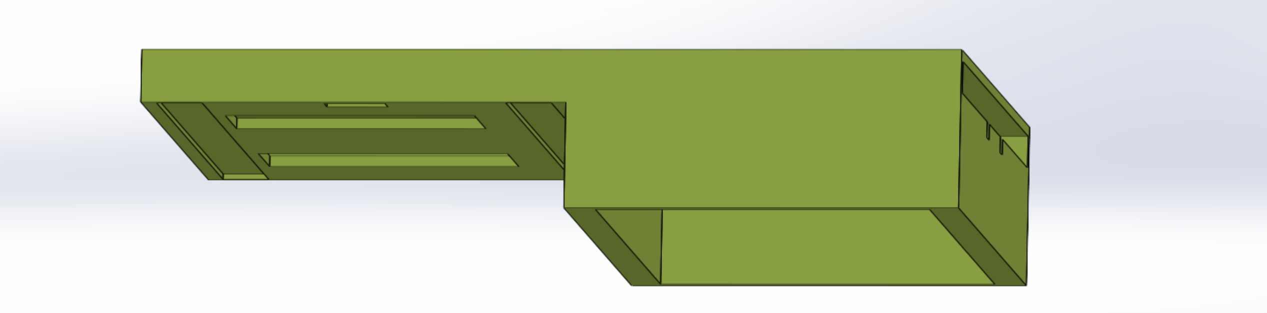
```
P1_ON, P1_OFF, ...
FAN_ON / FAN_OFF
LAMP_ON / LAMP_OFF
AUTO / MANUAL
```

Design choice: a lightweight command list kept the app simple and reduced bugs during testing.

Enclosure Design

SolidWorks model + build requirements





Design requirements

- Separate electronics chamber (protected from leaks)
- Dedicated slots for pumps + tubing
- Airflow openings for the fan
- Accessible lid for maintenance + refilling the water tank
- Sensor placement aligned with pots for accurate readings
- Cable routing holes to keep the interior organized

Final Prototype

Build + demo photos

What worked well

- Stable sensor readings after using a common ground and cleaner wiring
- Auto watering responded correctly to dry-soil thresholds
- App successfully controlled each device in Manual mode
- Fan + light improved plant environment during testing

Demo visuals

Challenges & Fixes

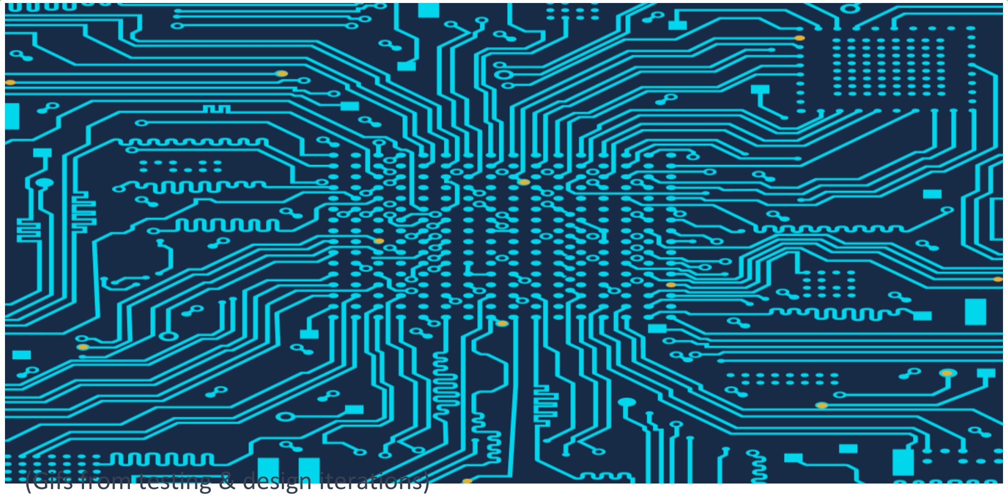
What we learned during debugging

Noisy sensor readings
Fixed by better wiring, common ground, and moving power lines away from analog inputs.

Pump backflow / leaks
Used tighter tubing + checked seals; kept electronics in a separate chamber.

Bluetooth disconnects
Kept commands simple; tested baud rate and pairing before final mounting.

Space constraints
Repositioned components and used a custom PCB layout to reduce clutter.

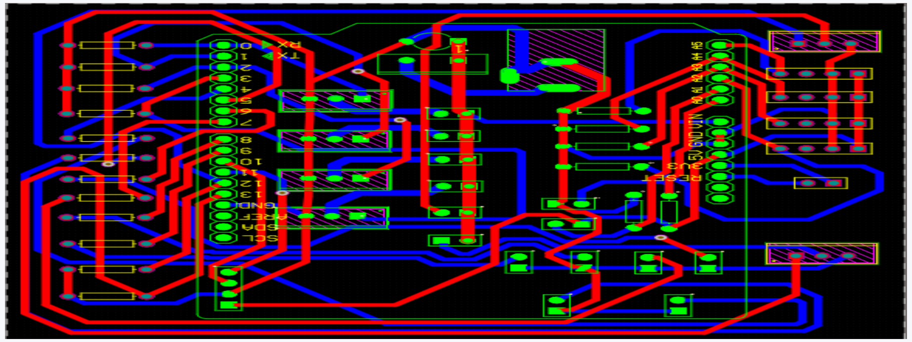


(Gifs from testing & design iterations)

Future Improvements

How we would make it even better

- Add a real-time display (OLED) to show moisture/temp/light on-device
- Replace Bluetooth with Wi-Fi (ESP32) for remote monitoring
- Add a water-level sensor to avoid running pumps dry
- Improve moisture sensors (capacitive type) for longer lifetime
- Log data to graphs (weekly trends) inside the app
- 3D-print enclosure parts for stronger waterproofing and cleaner finish



Conclusion

Key takeaway

By combining sensors, control logic, and a simple mobile i

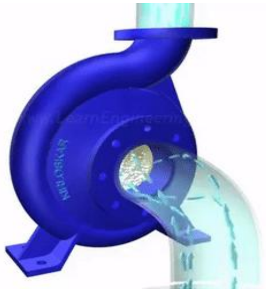
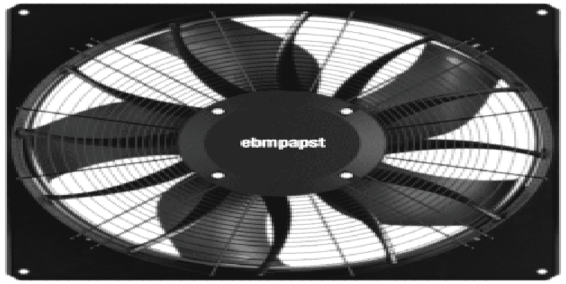
This project strengthened our skills in:
• Electronics + power management
• PCB design (schematic → layout)
• Embedded programming (Arduino)
• App development + Bluetooth communication
• Mechanical design + prototyping

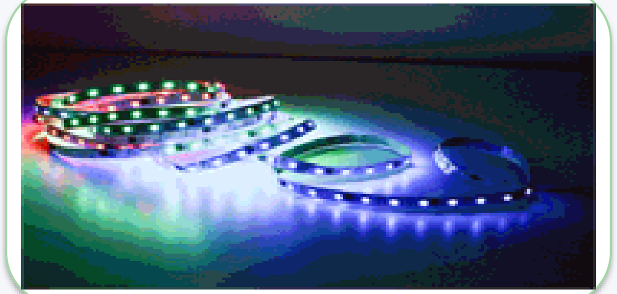
Project files / download (from original deck):

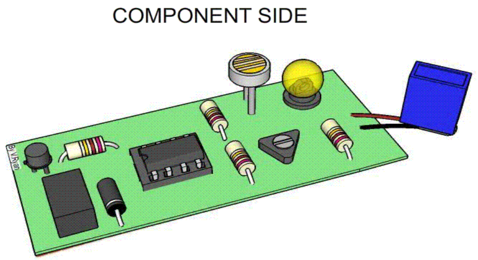
picofile.com/vhEN/Smart_Vase.rar.html

Visual Appendix

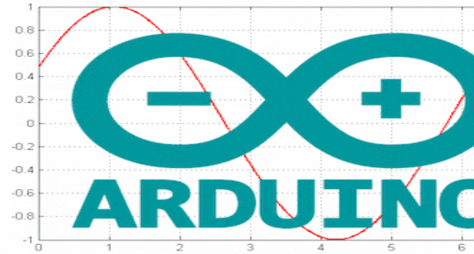
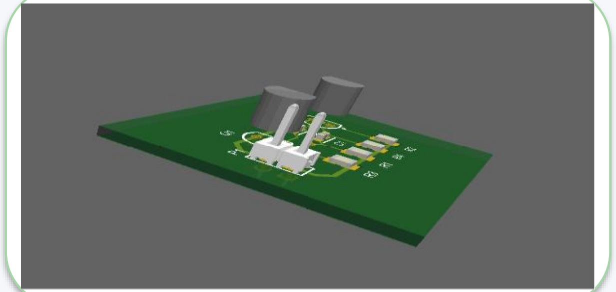
All images & GIFs from the original presentation (page 1)

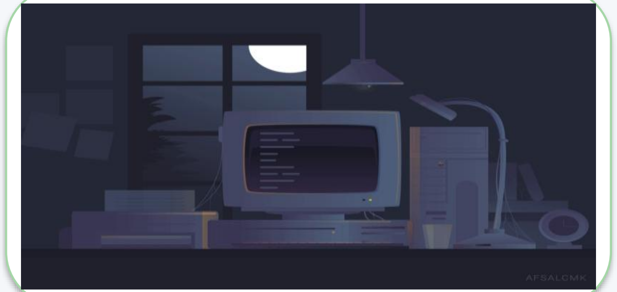
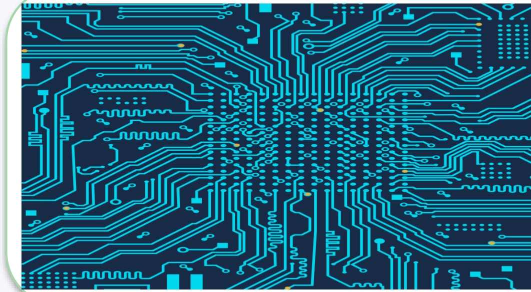







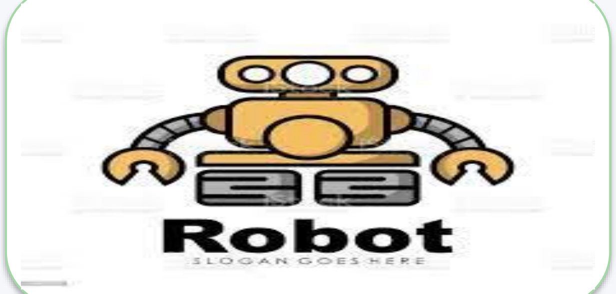


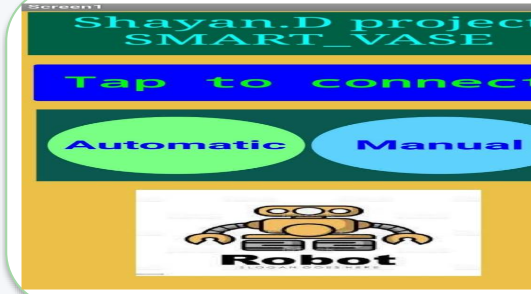



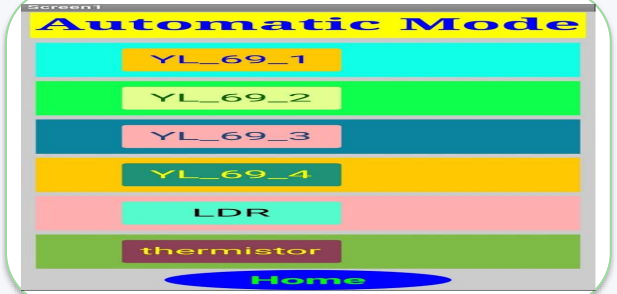



Visual Appendix

All images & GIFs from the original presentation (page 2)

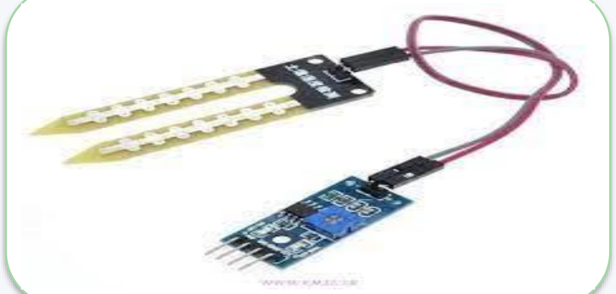


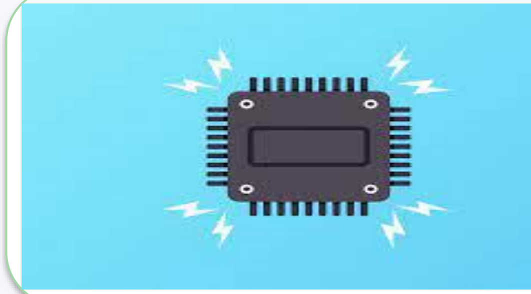




Visual Appendix

All images & GIFs from the original presentation (page 3)



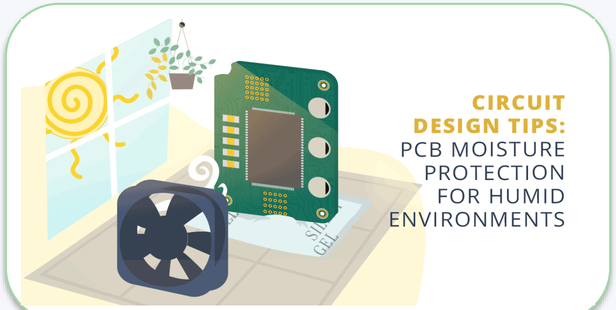




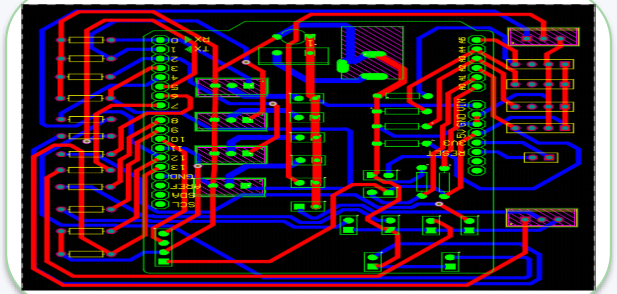


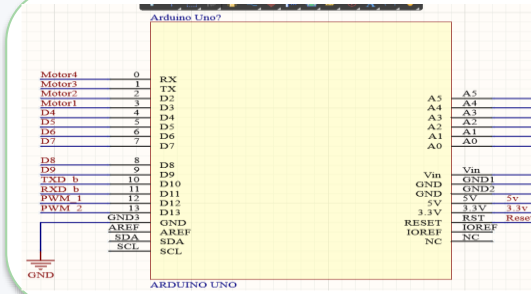
Visual Appendix

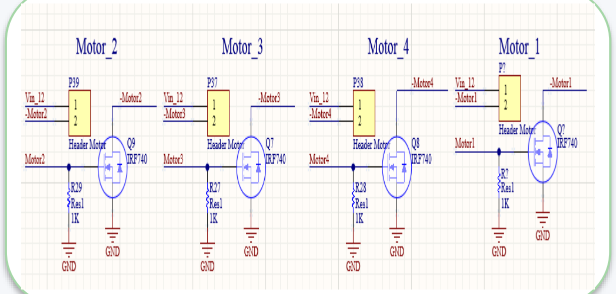
All images & GIFs from the original presentation (page 4)

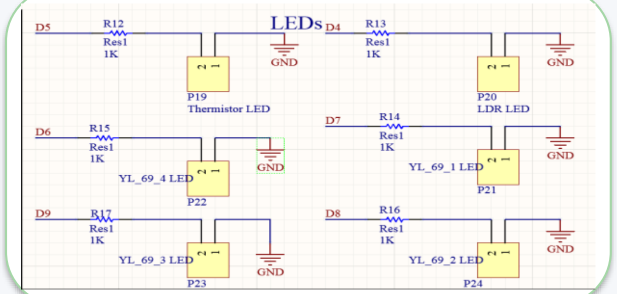


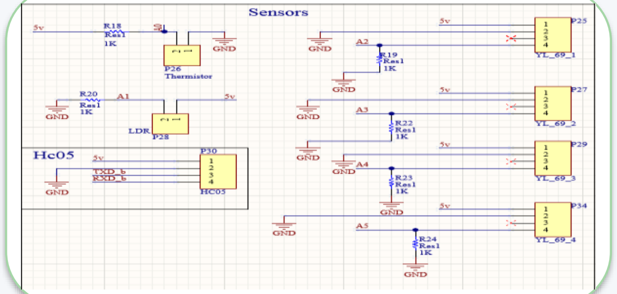
ALTIUM DESIGNER

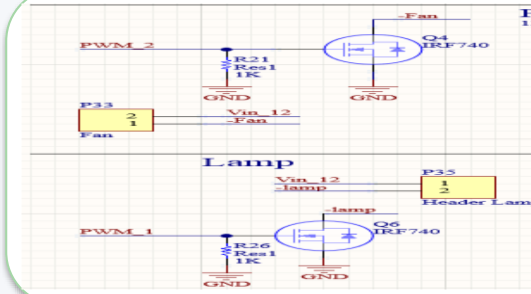






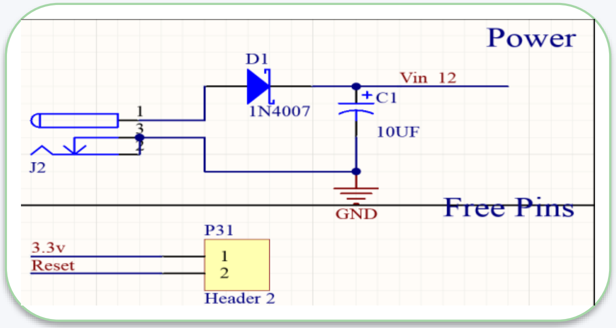


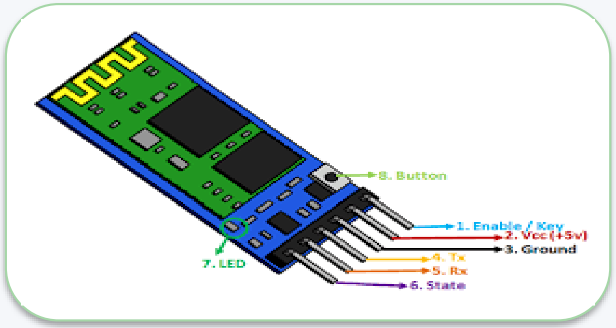


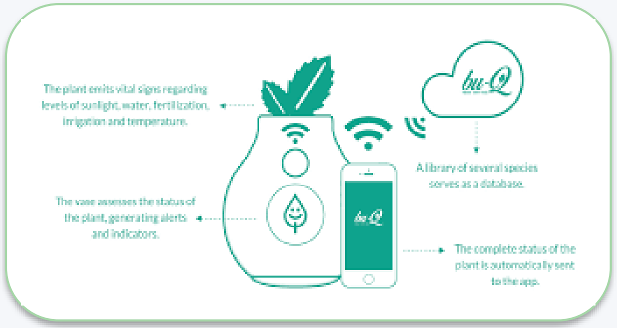


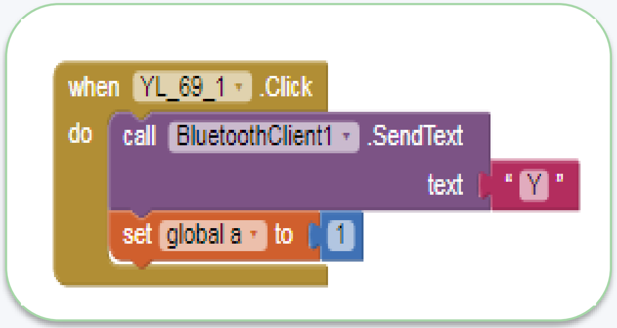
Visual Appendix

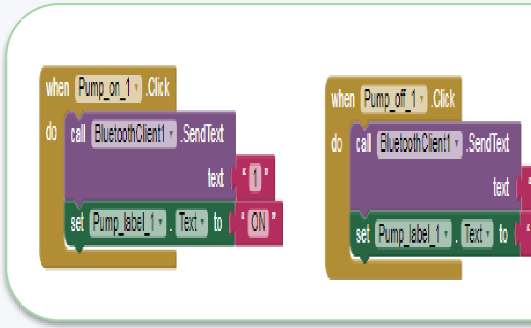
All images & GIFs from the original presentation (page 5)

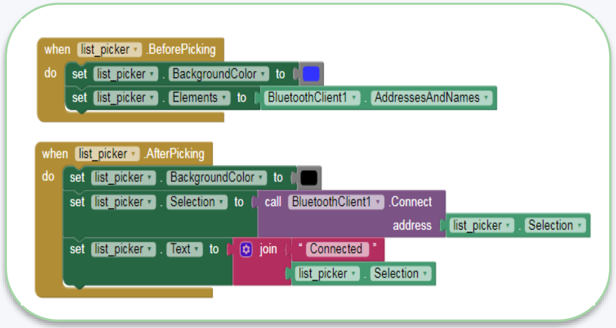


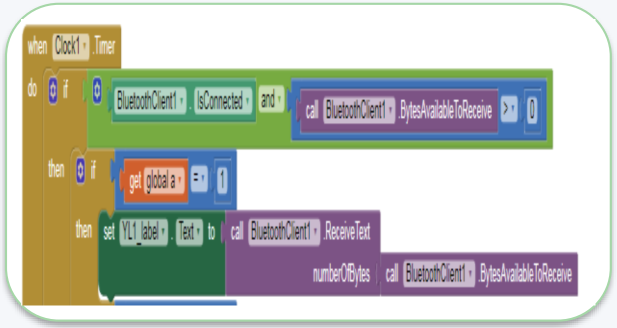


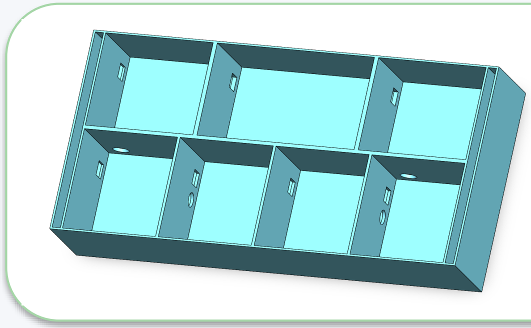












Visual Appendix

All images & GIFs from the original presentation (page 6)