

GIRLS WITH ENERGY HACKATHON

GreenBot

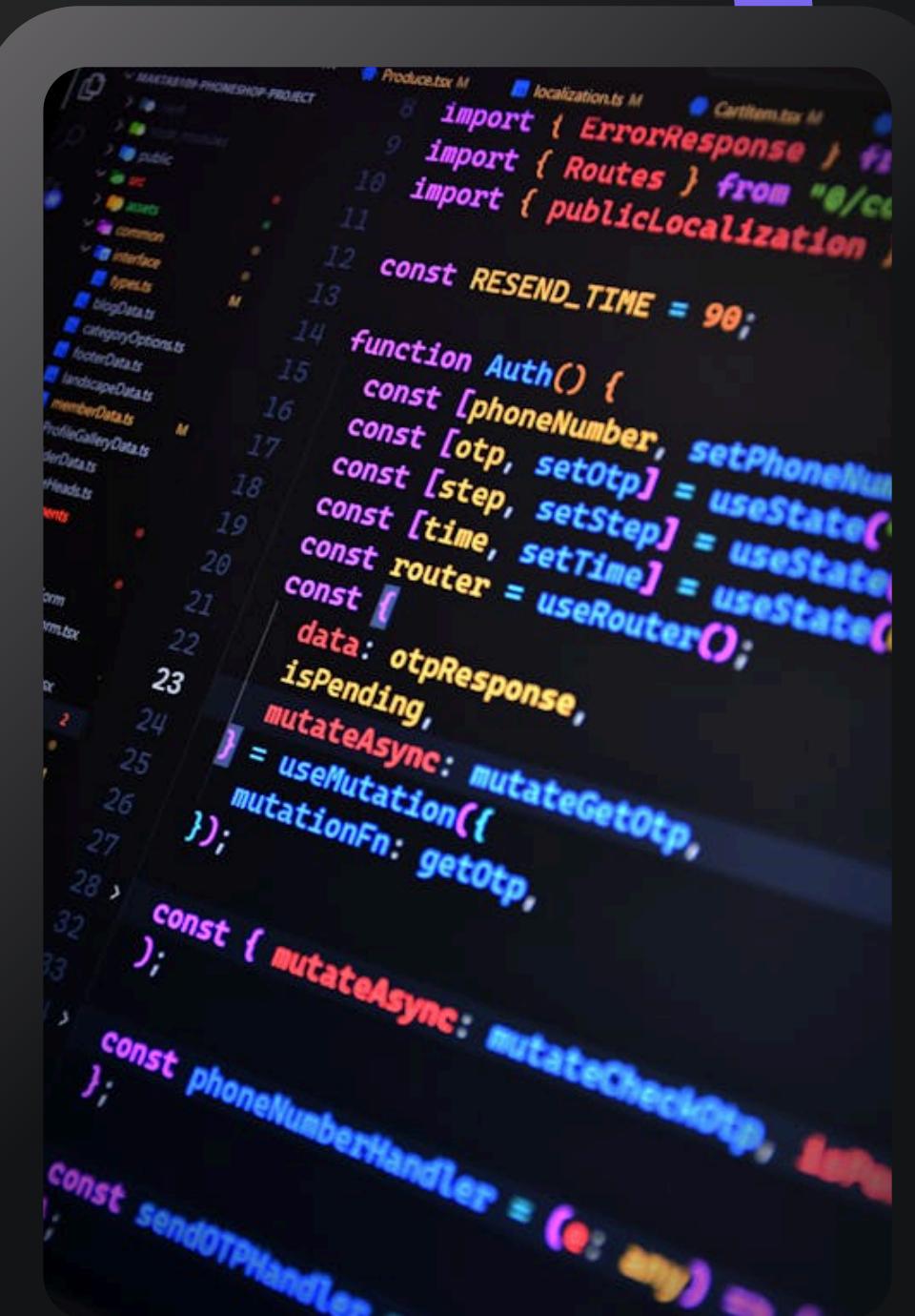
IMPROVING AIR QUALITY, ONE ROBOT AT A TIME.

JIZHEN HU, ZIQI TANG, ZI XIN ZHOU

GWE04

About Us

- Jizhen, Ziqi, Zi Xin
- 10th graders in Montreal, Canada
- Interested in computer science since a young age



```
localization.ts M
  8 import { ErrorResponse } from "src/error";
  9 import { Routes } from "src/routes";
 10 import { publicLocalization } from "src/localization";
 11
 12 const RESEND_TIME = 90;
 13
 14 function Auth() {
 15   const [phoneNumber, setPhoneNumber] = useState("");
 16   const [otp, setOtp] = useState("");
 17   const [step, setStep] = useState(1);
 18   const [time, setTime] = useState(new Date());
 19   const router = useRouter();
 20
 21   const [
 22     data: otpResponse,
 23     isPending,
 24   ] = useMutation({
 25     mutationFn: getOtp,
 26     mutationAsync: mutateGetOtp,
 27   });
 28
 29   const [
 30     mutateAsync: mutateCheckOtp,
 31     data: otp,
 32   ] = useMutation({
 33     mutationFn: checkOtp,
 34     mutationAsync: mutateCheckOtp,
 35   });
 36
 37   const phoneNumberHandler = (e: any) => {
 38     setPhoneNumber(e.target.value);
 39   };
 40
 41   const sendOTPHandler = () => {
 42     if (!phoneNumber) return;
 43
 44     if (isPending) return;
 45
 46     if (otp === "") {
 47       toast.error("Please enter a valid OTP");
 48       return;
 49     }
 50
 51     if (otp !== otpResponse) {
 52       toast.error("Incorrect OTP");
 53       return;
 54     }
 55
 56     if (step === 1) {
 57       setStep(2);
 58       setTime(new Date());
 59     } else if (step === 2) {
 60       setStep(1);
 61       setTime(new Date());
 62     }
 63   };
 64
 65   const resendOTPHandler = () => {
 66     if (isPending) return;
 67
 68     if (time.getTime() - time.getTime() < RESEND_TIME) {
 69       toast.error("Resend OTP in 90 seconds");
 70       return;
 71     }
 72
 73     setOtp("");
 74     setStep(1);
 75     setTime(new Date());
 76   };
 77
 78   const handleStepChange = (step: number) => {
 79     if (step === 1) {
 80       setOtp("");
 81     }
 82   };
 83
 84   const handleTimeChange = (time: Date) => {
 85     setTime(time);
 86   };
 87
 88   const handleRouterChange = (url: string) => {
 89     router.push(url);
 90   };
 91
 92   const handleLogout = () => {
 93     localStorage.removeItem("token");
 94     router.push("/");
 95   };
 96 }
```

Our Team



Jizhen Hu
Team Leader



Ziqi Tang
Team Member



Zi Xin Zhou
Team Member

Air Pollution: A Global Health Crisis



WHERE?

- Pollution: Chad, Bangladesh, Pakistan, India, etc.
- Cleanest: Australia, New Zealand, Finland, etc.

WHO'S
AFFECTED?

- Citizens in countries with poor air quality
- Especially: Children, the elderly, people with existing health conditions (heart, lungs)

IMPACTS?

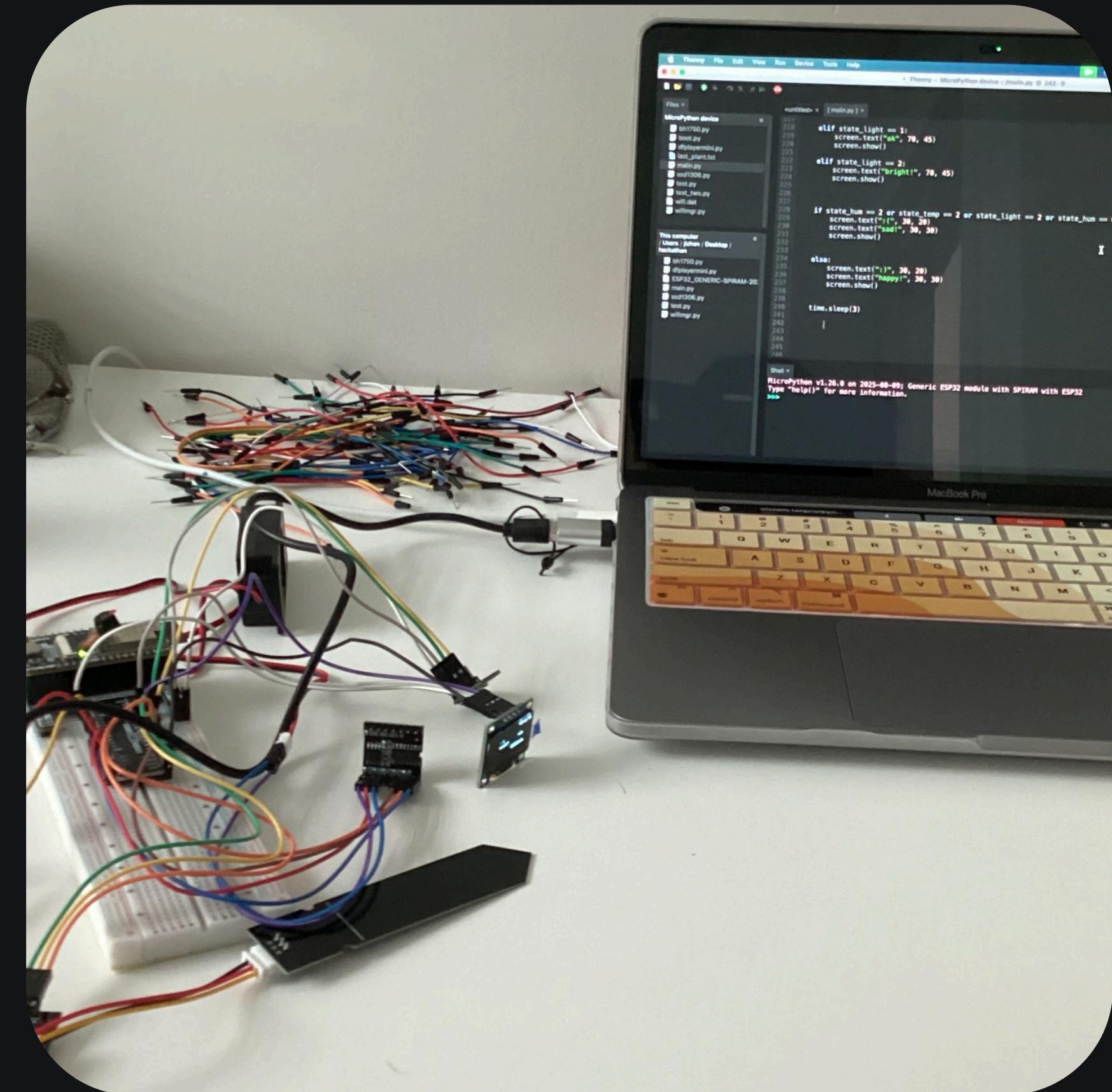
- Health: Chronic respiratory diseases, lung cancer, heart diseases, shorter life span
- Environmental: Acid rain formation, climate change acceleration, soil & water contamination

SOLUTION?

- Our robot!

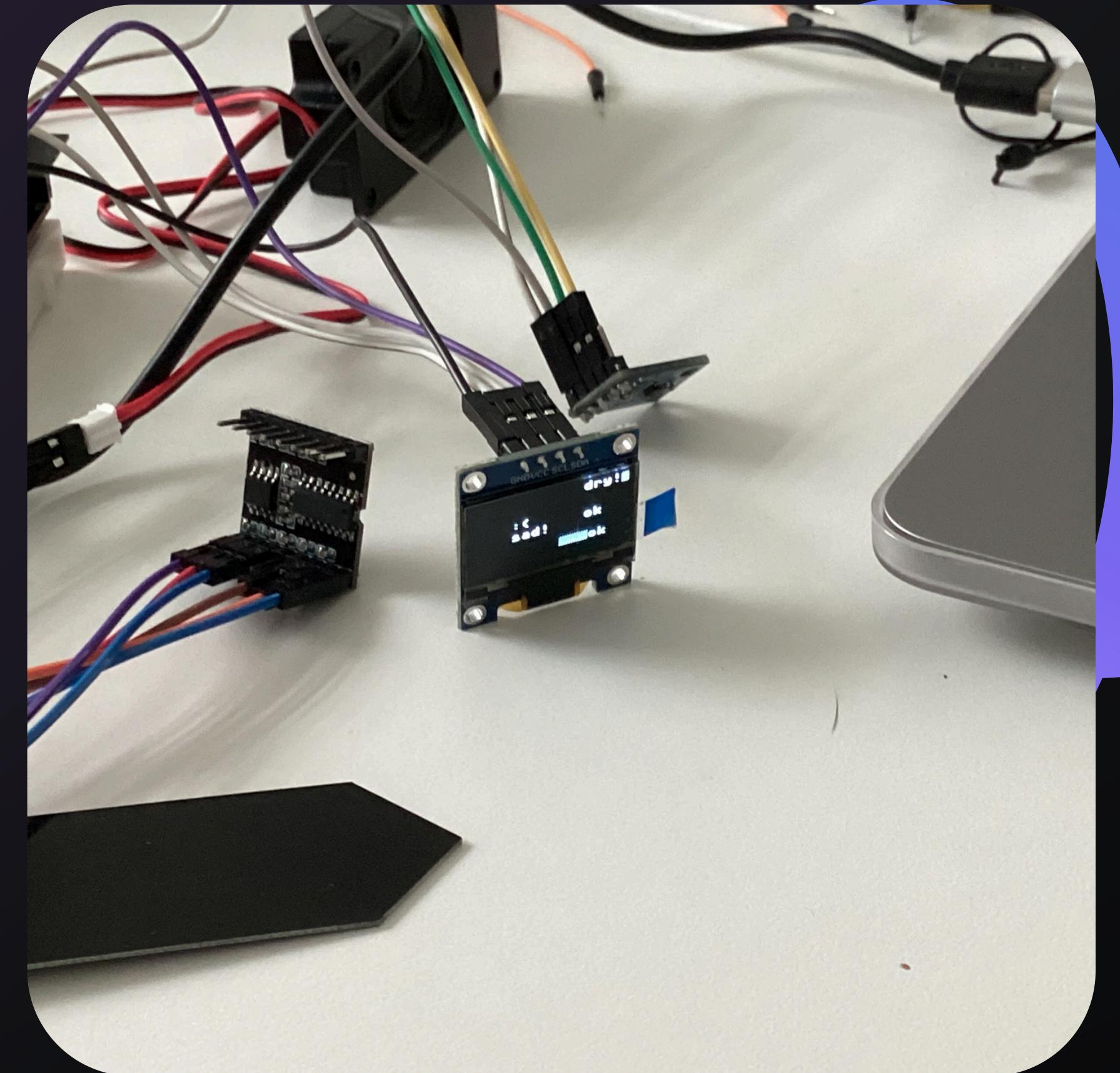


A Solution for Tomorrow's Problems



What Does It Do?

- Teaches the user how to take care of any plant
- Displays any plant's needs with an emoji
- Notifies the user when the plant needs water, or care (text and speaker)
 - User doesn't need to remember about watering plants (user friendly)
- Plants stay healthy because of our bot
- User can access the web for more specifications

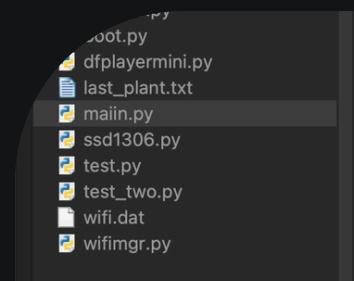


How It Works

- Detects a plant's condition by running code stored internally
- Can connect to the Internet
 - It sends the plant's data to the web by flask
 - The screen will display an emoji corresponding to the state (needs water, too much water, something else)
 - Pre-recorded message plays (DFPlayer)
 - The web will also display an emoji corresponding to the state (needs water, too much water, something else)
 - Plant state + corresponding emoji sent to the web, then generating a user-personalized message with ChatGPT API
 - Can analyze plant image to detect problems
- If it doesn't, bot processes the data locally (wifi or web don't answer)
 - Pre-recorded message plays (DFPlayer)
 - The screen will display an emoji corresponding to the state (needs water, too much water, something else)

PYTHON

```
1  if value <= lum["low"]:
2      return 0
3  elif value >= lum["high"]:
4      return 2
5  elif lum["low"] < value < lum["high"]:
6      return 1
7  else:
8      return 4
9
10 i2c = SoftI2C(scl=Pin(22), sda=Pin(21))
11 screen = ssd1306.SSD1306_I2C(128, 64, i2c)
12
13 state_hum = get_hum_soil()
14 state_temp = get_temp()
15 state_light = get_light()
16
17 if state_hum == 0:
18     screen.text("dry!!", 90, 0)
19     screen.show()
20
21 elif state_hum == 1:
22     screen.text("ok", 90, 0)
23     screen.show()
24
25 elif state_hum == 2:
26     screen.text("wet!!", 90, 0)
27     screen.show()
```

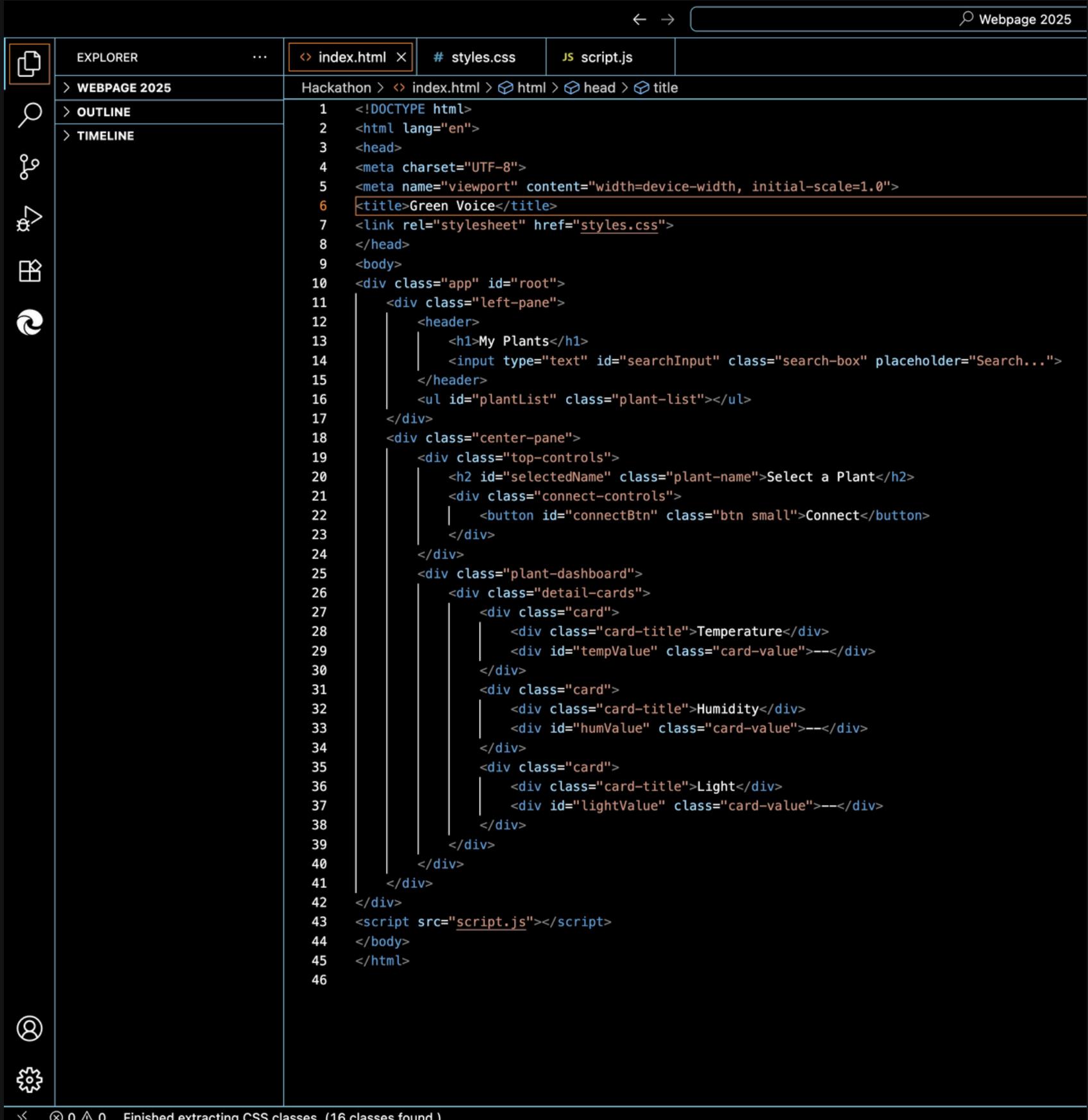


```
1 from ssd1306 import SSD1306_I2C
2 import ssd1306
3 import dht
4 from bh1750 import BH1750
5 import dfplayermini
6 import urequests
7 import wifimgr
8
9 plants_name = {
10     "desert_cactus": {
11         "soil": {"low": 3481, "high": 2457},
12         "temp": {"low": 8, "high": 38},
13         "light": {"low": 800, "high": 3000}
14     },
15     "succulent": {
16         "soil": {"low": 3276, "high": 2048},
17         "temp": {"low": 10, "high": 35},
18         "light": {"low": 600, "high": 2500}
19     },
20     "tropical_fern": {
21         "soil": {"low": 2048, "high": 410},
22         "temp": {"low": 16, "high": 28},
23         "light": {"low": 50, "high": 400}
24     },
25     "tropical_foliage": {
26         "soil": {"low": 2457, "high": 819},
27         "temp": {"low": 15, "high": 30},
28         "light": {"low": 100, "high": 1500}
29     },
30     "mediterranean_herb": {
31         "soil": {"low": 3071, "high": 1638},
32         "temp": {"low": 12, "high": 32},
33         "light": {"low": 800, "high": 2500}
34     },
35     "soft_herb": {
```

Impact

- Air quality will get better bit by bit
- Clean air = better health, less hospital visits, overall healthier population
- Better environment

HTML

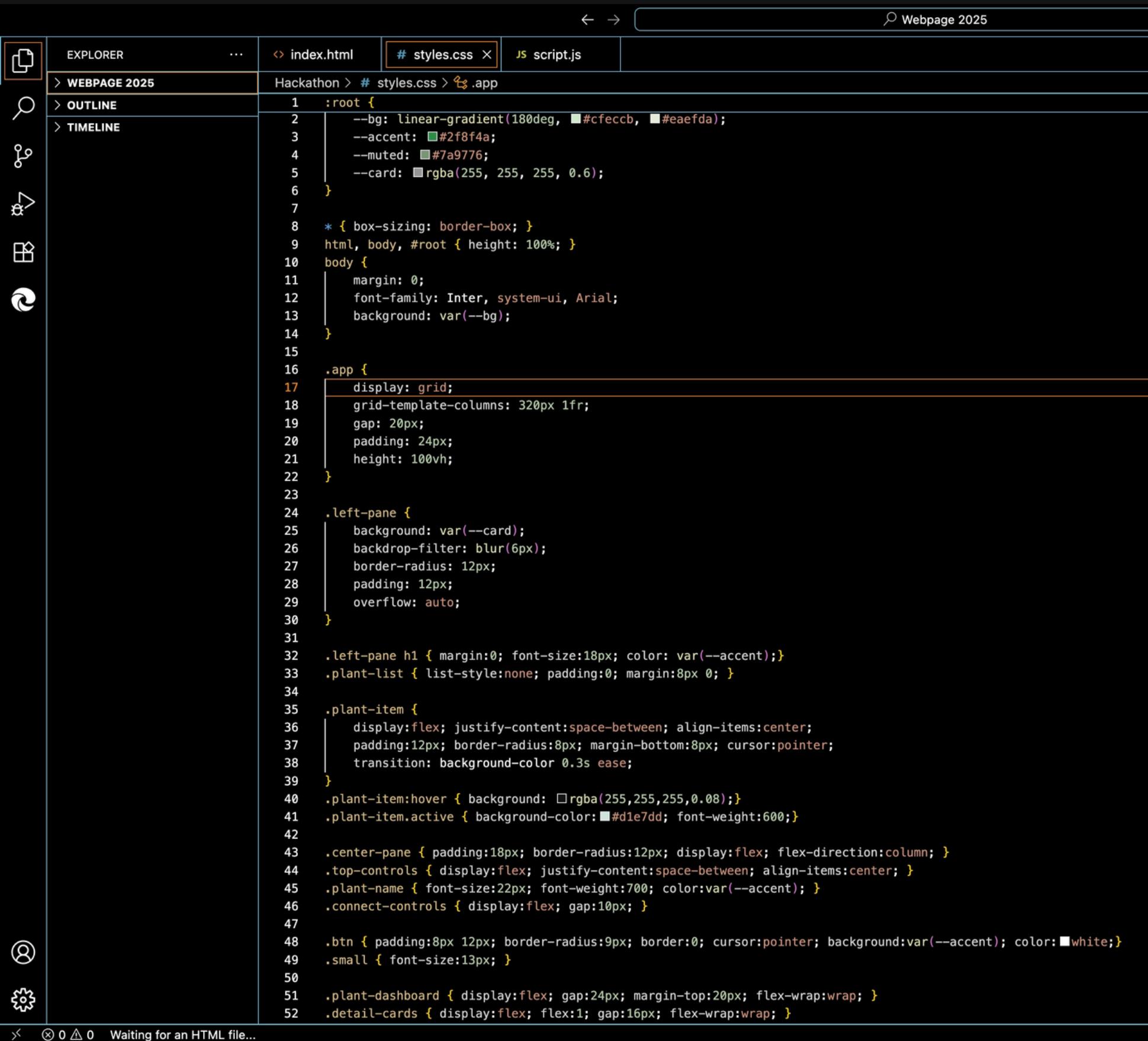


The screenshot shows the VS Code interface with the title bar "Webpage 2025". The left sidebar includes icons for Explorer, Search, Find, Timeline, and Settings. The main area displays the HTML code for "index.html". The code structure is as follows:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Green Voice</title>
<link rel="stylesheet" href="styles.css">
</head>
<body>
<div class="app" id="root">
  <div class="left-pane">
    <header>
      <h1>My Plants</h1>
      <input type="text" id="searchInput" class="search-box" placeholder="Search...">
    </header>
    <ul id="plantList" class="plant-list"></ul>
  </div>
  <div class="center-pane">
    <div class="top-controls">
      <h2 id="selectedName" class="plant-name">Select a Plant</h2>
      <div class="connect-controls">
        <button id="connectBtn" class="btn small">Connect</button>
      </div>
    </div>
    <div class="plant-dashboard">
      <div class="detail-cards">
        <div class="card">
          <div class="card-title">Temperature</div>
          <div id="tempValue" class="card-value">--</div>
        </div>
        <div class="card">
          <div class="card-title">Humidity</div>
          <div id="humValue" class="card-value">--</div>
        </div>
        <div class="card">
          <div class="card-title">Light</div>
          <div id="lightValue" class="card-value">--</div>
        </div>
      </div>
    </div>
  </div>
  <script src="script.js"></script>
</body>
</html>
```

At the bottom, a status bar indicates "Finished extracting CSS classes. (16 classes found.)".

CSS



The screenshot shows the VS Code interface with the title bar "Webpage 2025". The left sidebar includes icons for Explorer, Search, Find, Timeline, and Settings. The main area displays the CSS code for "styles.css". The code defines several CSS variables and components:

```
:root {
  --bg: linear-gradient(180deg, #cfeccb, #eaeefda);
  --accent: #2f8f4a;
  --muted: #7a9776;
  --card: rgba(255, 255, 255, 0.6);
}

* { box-sizing: border-box; }
html, body, #root { height: 100%; }
body {
  margin: 0;
  font-family: Inter, system-ui, Arial;
  background: var(--bg);
}

.app {
  display: grid;
  grid-template-columns: 320px 1fr;
  gap: 20px;
  padding: 24px;
  height: 100vh;
}

.left-pane {
  background: var(--card);
  backdrop-filter: blur(6px);
  border-radius: 12px;
  padding: 12px;
  overflow: auto;
}

.left-pane h1 { margin: 0; font-size: 18px; color: var(--accent); }
.plant-list { list-style: none; padding: 0; margin: 8px 0; }

.plant-item {
  display: flex; justify-content: space-between; align-items: center;
  padding: 12px; border-radius: 8px; margin-bottom: 8px; cursor: pointer;
  transition: background-color 0.3s ease;
}
.plant-item:hover { background: rgba(255, 255, 255, 0.08); }
.plant-item.active { background-color: #d1e7dd; font-weight: 600; }

.center-pane { padding: 18px; border-radius: 12px; display: flex; flex-direction: column; }
.top-controls { display: flex; justify-content: space-between; align-items: center; }
.plant-name { font-size: 22px; font-weight: 700; color: var(--accent); }
.connect-controls { display: flex; gap: 10px; }

.btn { padding: 8px 12px; border-radius: 9px; border: 0; cursor: pointer; background: var(--accent); color: white; }
.small { font-size: 13px; }

.plant-dashboard { display: flex; gap: 24px; margin-top: 20px; flex-wrap: wrap; }
.detail-cards { display: flex; flex: 1; gap: 16px; flex-wrap: wrap; }
```

At the bottom, a status bar indicates "Waiting for an HTML file...".

JAVASCRIPT

The screenshot shows a dark-themed code editor interface with the following details:

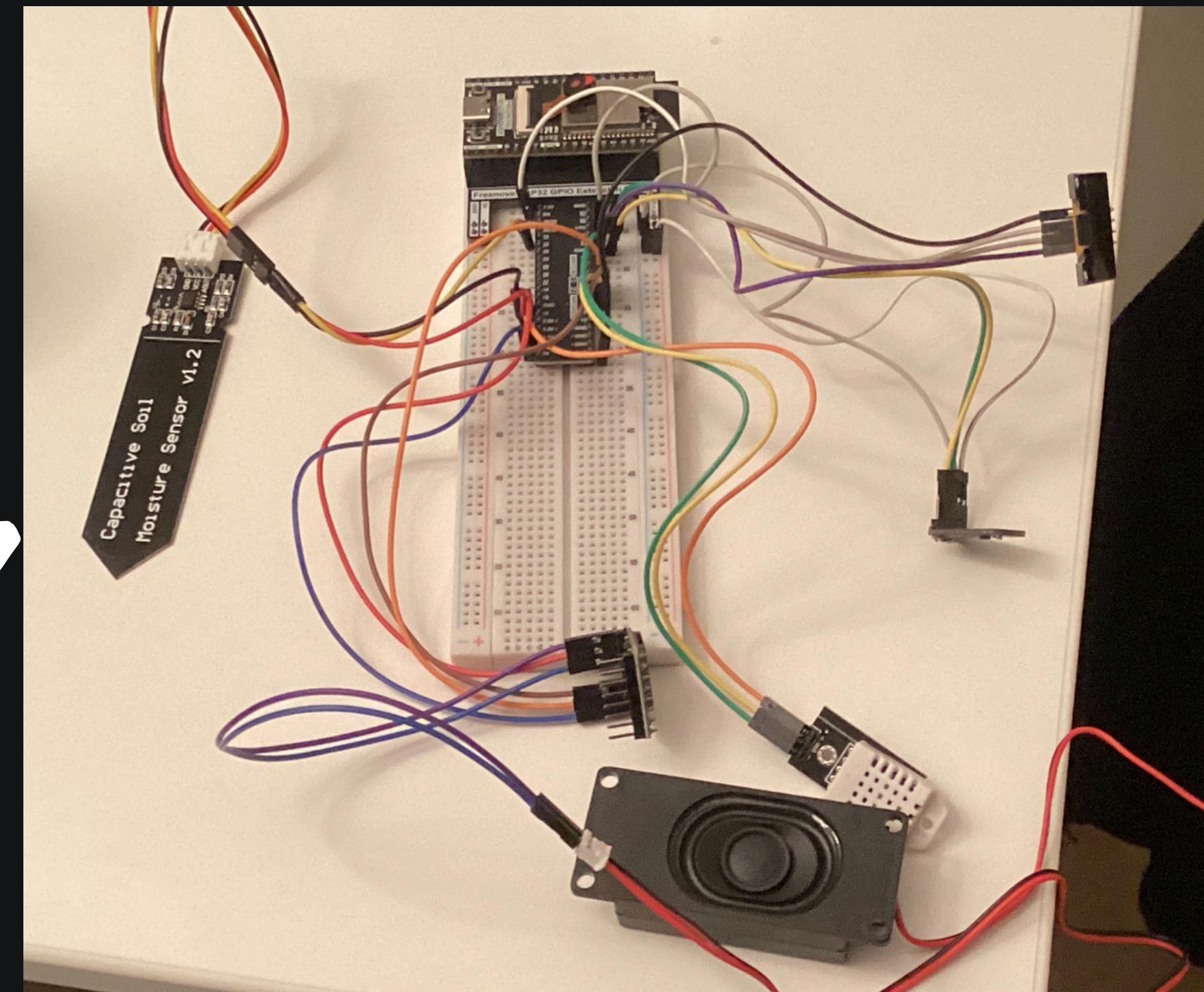
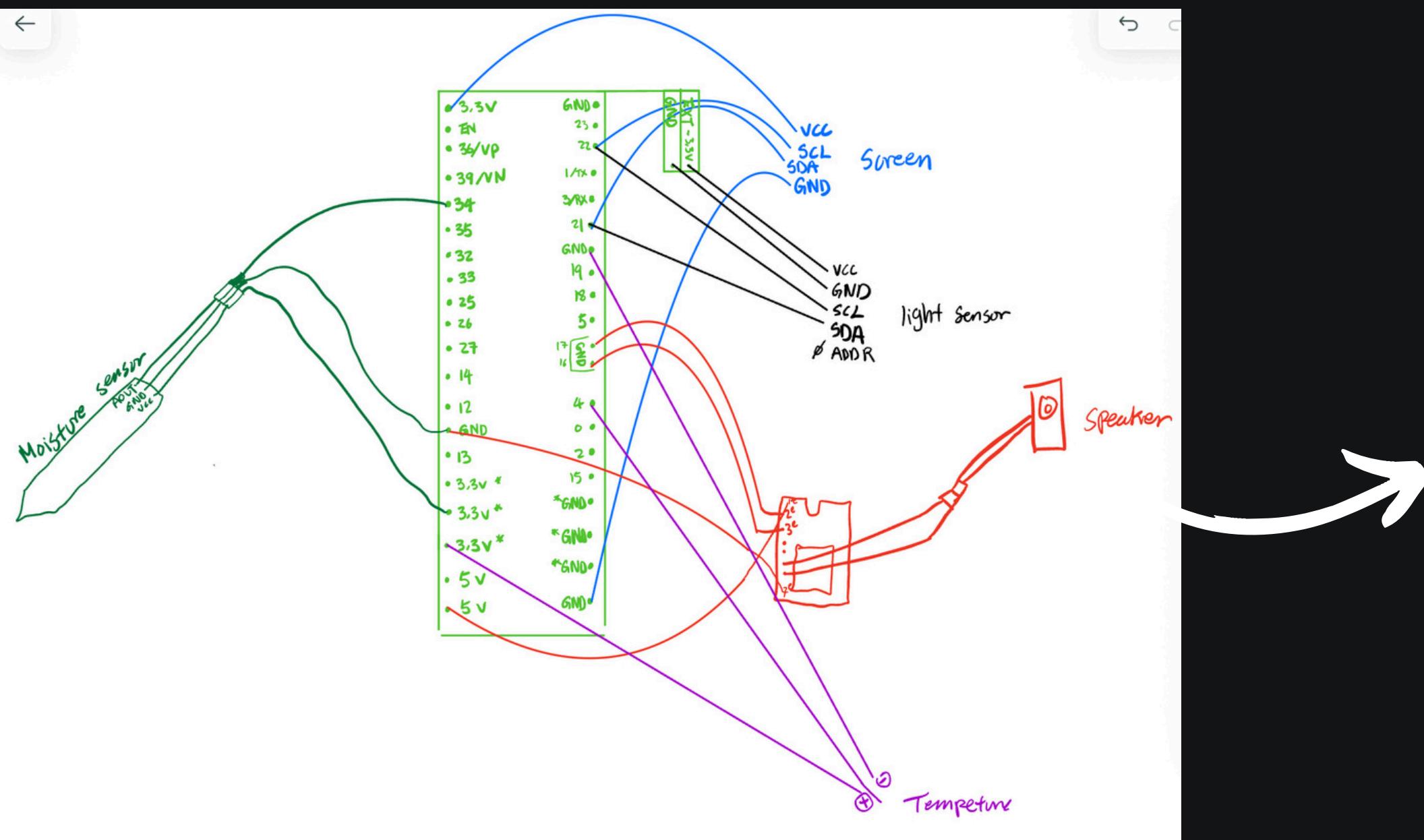
- File Explorer:** Shows files: index.html, styles.css, and script.js (selected).
- Search Bar:** Displays "Webpage 2025".
- Code Editor:** Contains the following JavaScript code:

```
1 const PLANT_CATEGORIES = [
2   "desert_cactus", "succulent", "tropical_fern", "tropical_foliage",
3   "mediterranean_herb", "soft_herb", "orchid", "shade_loving",
4   "bright_flowering", "temperate_tree", "fruit_veg", "bonsai"
5 ];
6
7 const plantListEl = document.getElementById("plantList");
8 const selectedName = document.getElementById("selectedName");
9 const tempValue = document.getElementById("tempValue");
10 const humValue = document.getElementById("humValue");
11 const lightValue = document.getElementById("lightValue");
12 const connectBtn = document.getElementById("connectBtn");
13 const searchInput = document.getElementById("searchInput");
14
15 let currentPlant = null;
16 let connectedPlant = null;
17
18 // Render list
19 function renderPlantList(filter=""){
20   plantListEl.innerHTML = "";
21   PLANT_CATEGORIES
22     .filter(c => c.toLowerCase().includes(filter.toLowerCase()))
23     .forEach(cat=>{
24       const li = document.createElement("li");
25       li.className="plant-item";
26       if(currentPlant==cat) li.classList.add("active");
27       li.textContent = cat.replace(/_/g," ").replace(/\b\w/g, l=>l.toUpperCase());
28
29       const status = document.createElement("span");
30       status.textContent = (connectedPlant==cat) ? "🟢" : "🔴";
31       li.appendChild(status);
32
33       li.addEventListener("click", ()=>{
34         currentPlant = cat;
35         updateDashboard();
36         renderPlantList(searchInput.value);
37       });
38     });
39   plantListEl.appendChild(li);
40 }
41
42 // Update Dashboard
43 function updateDashboard(){
44   selectedName.textContent = currentPlant ? currentPlant.replace(/_/g," ").replace(/\b\w/g,l=>l.toUpperCase()) : "Select a Plant";
45   if(!currentPlant) return;
46
47   // Fetch UI state from backend
48   fetch("http://192.168.0.104:5001/ui_state")
49     .then(res=>res.json())
50     .then(data=>{
51       tempValue.textContent = data.temp?.label || "?";
52       humValue.textContent = data.soil?.label || "?";
53     });
54 }
```

The code defines a `script.js` file for a webpage. It includes a list of plant categories, a function to render a list of plants based on a filter, and a function to update the dashboard with UI state from a backend. The dashboard includes temperature and humidity values.

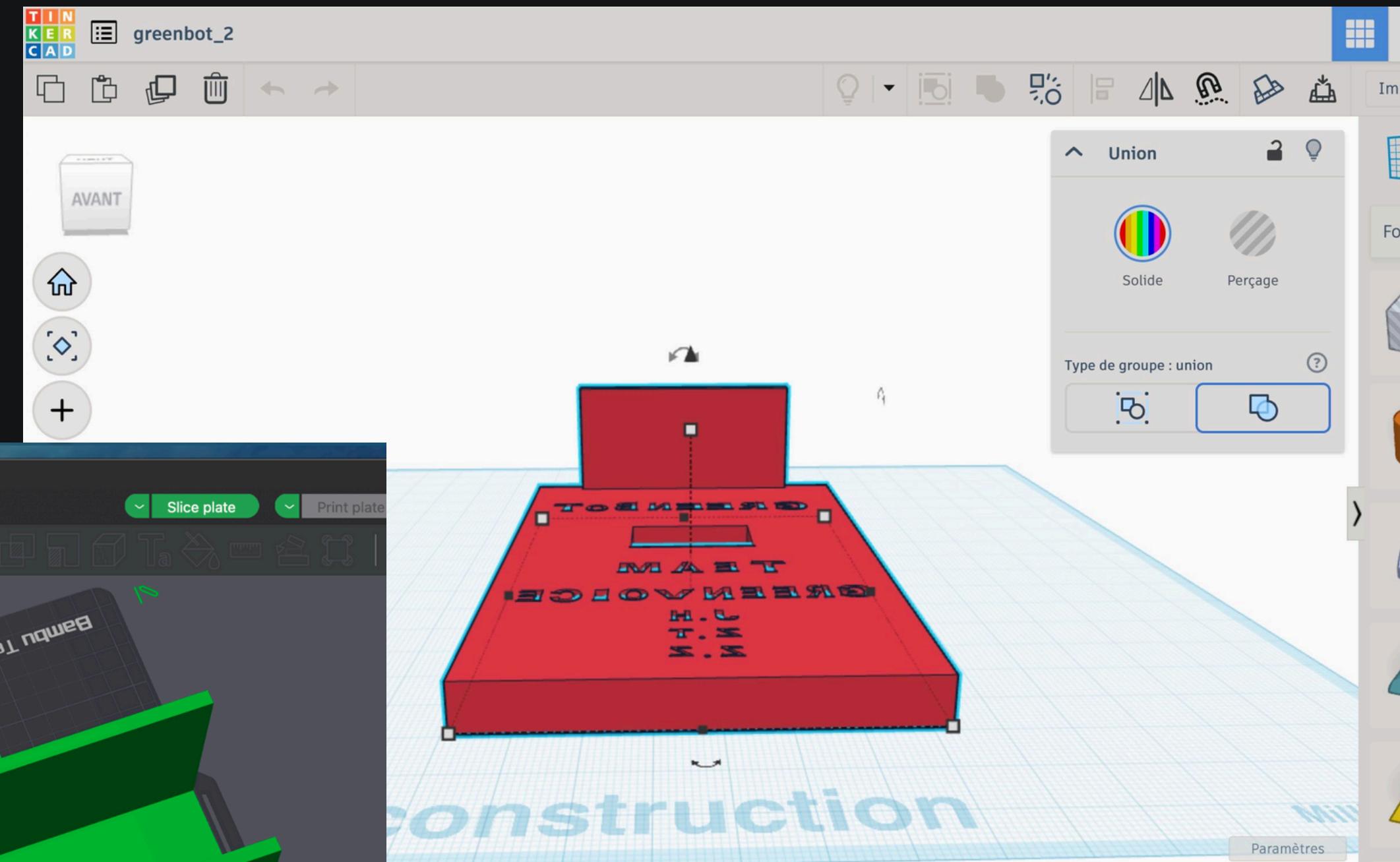
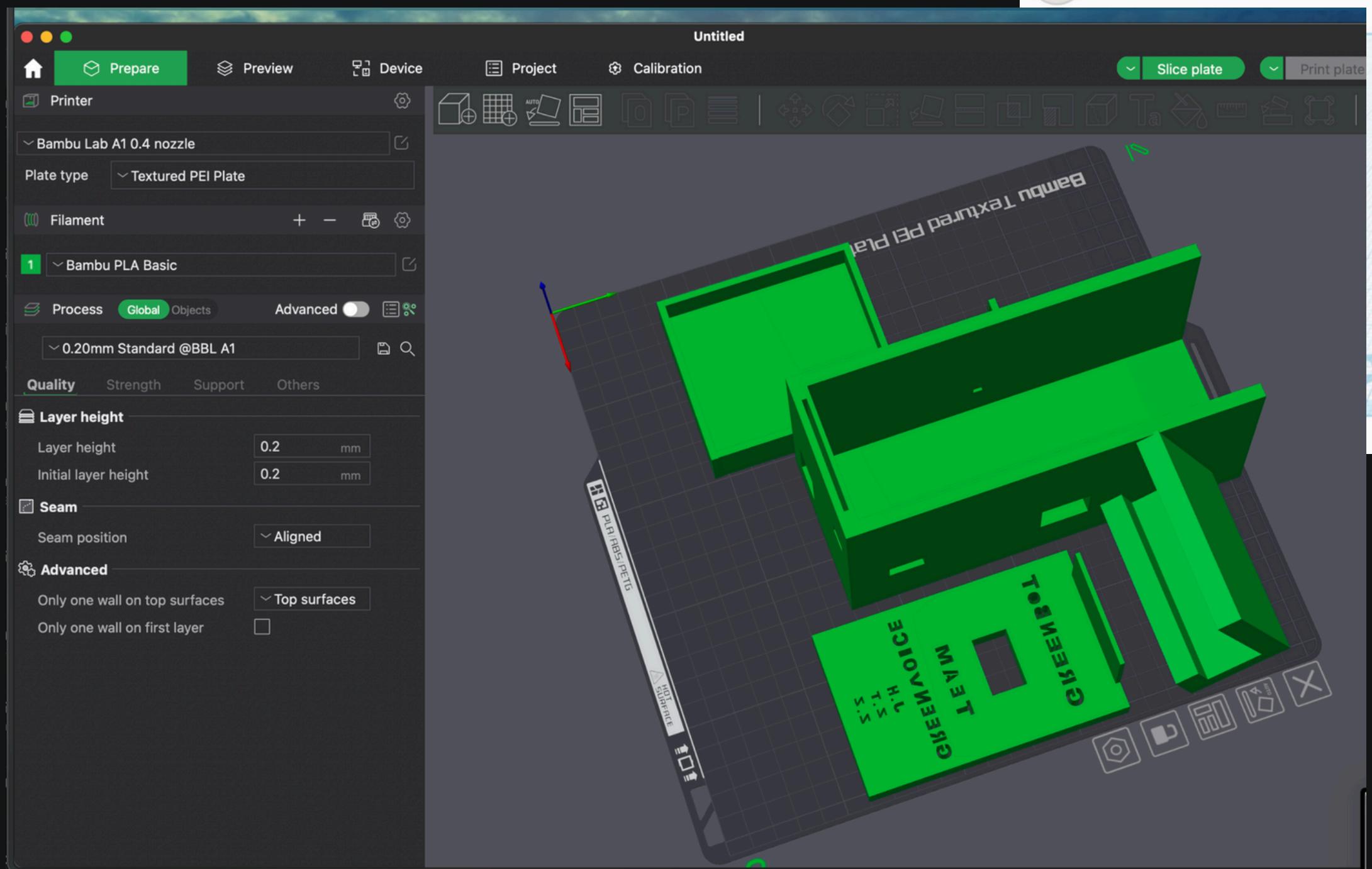
Breadboard circuit:

A microcontroller(ESP32), a soil moisture sensor, a temperature, humidity sensor and a small speaker by dfplayer



3D PRINTING

-Tinkercad





Thank You