Solar Sage AI – IoT System

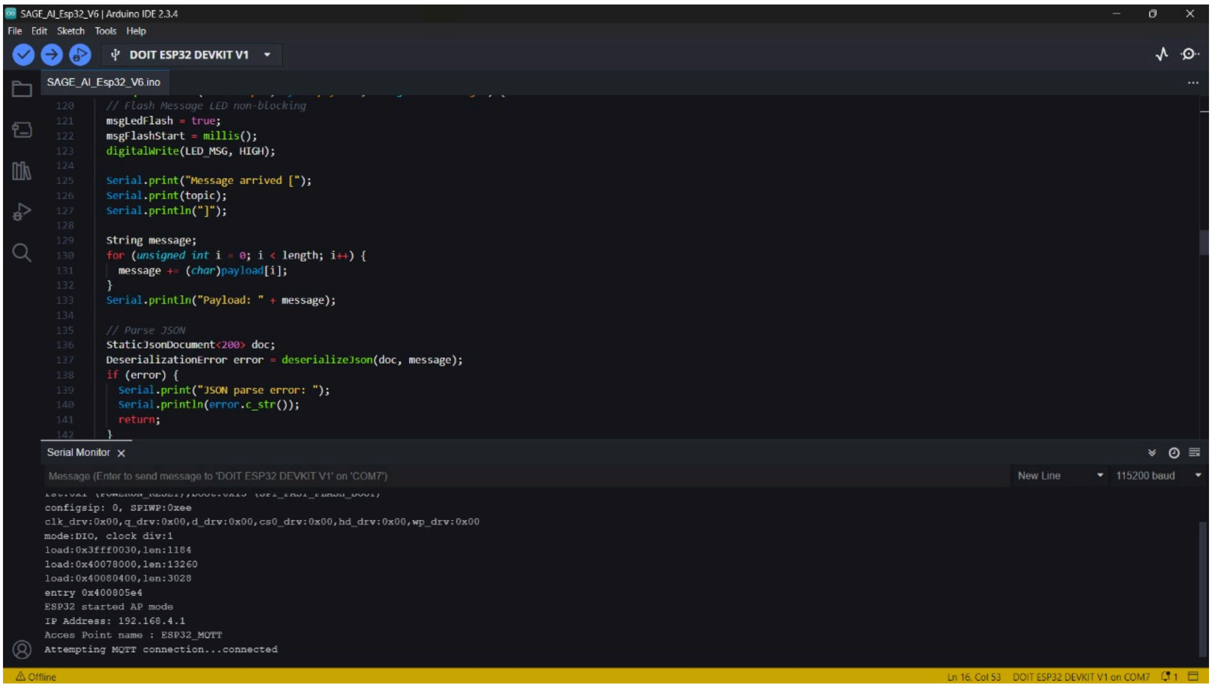
🧩Section 1: Overview of IoT Subsystem

- **Objective:**

  A robust IoT subsystem for **solar panel heatmap visualization** and **automated nozzle spraying** based on efficiency metrics and data send AI via agents.

- **ESP32 Responsibilities:**
  - Establish Wi-Fi AP(Local Soft Access Point : Offline Mode) and MQTT broker (MQTT protocol : Secure and Low Latency)
  - Subscribe to topics:
    - `spray/control` – Control spray start/stop (for basic prototype : can further increase functionalities depending upon usecase)
    - `spray/heatmap_data` – Receive simulated heatmap JSON data
  - Drive NeoPixel 5×5 LED matrix for **panel efficiency visualization (Heatmap)**
  - Control **servo/solenoid nozzles** for spray (Shown via controlling 5 neoPixel LEDs representing nozzles present at each Pannel)

IOT Code + Console Log (Starting MQTT Connection)



```
  // Flash Message LED non-blocking
  msgLedFlash = true;
  msgFlashStart = millis();
  digitalWrite(LED_MSG, HIGH);

  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.println("]");

  String message;
  for (unsigned int i = 0; i < length; i++) {
    message += (char)payload[i];
  }
  Serial.println("Payload: " + message);

  // Parse JSON
  StaticJsonDocument<200> doc;
  DeserializationError error = deserializeJson(doc, message);
  if (error) {
    Serial.print("JSON parse error: ");
    Serial.println(error.c_str());
    return;
  }
```

Message (Enter to send message to 'DOIT ESP32 DEVKIT V1' on 'COM7')

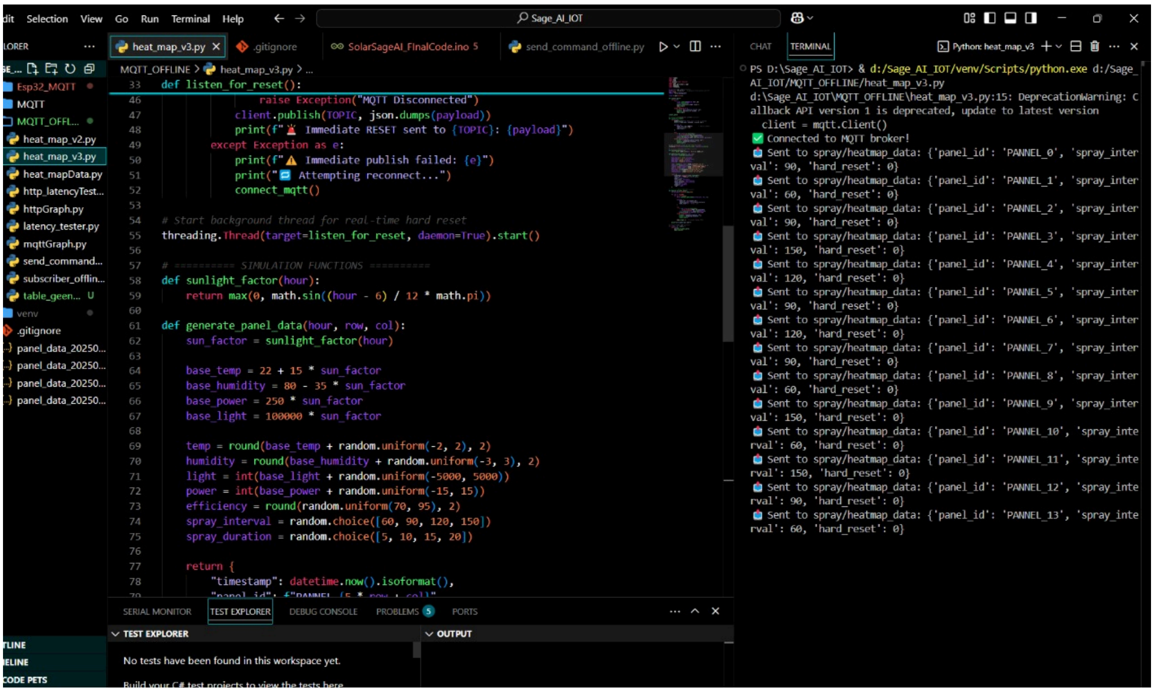
```
rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0030,len:1184
load:0x40078000,len:13260
load:0x40080400,len:3028
entry 0x400805e4
ESP32 started AP mode
IP Address: 192.168.4.1
Acces Point name : ESP32_MQTT
Attempting MQTT connection...connected
```

🧩 Section 2: Heatmap Visualization

- **Panel Grid:** 5×5 (25 Panels)
- **Display:** NeoPixel LED Matrix
- **Data Format:**

```
json
    return {
        "timestamp": datetime.now().isoformat(),
        "panel_id": f"PANNEL_{5 * row + col}",
        "power": power,
        "efficiency": efficiency,
        "spray_interval": spray_interval,
        "spray_duration": spray_duration,
        "temperature": temp,
        "humidity": humidity,
        "light": light
    }
```

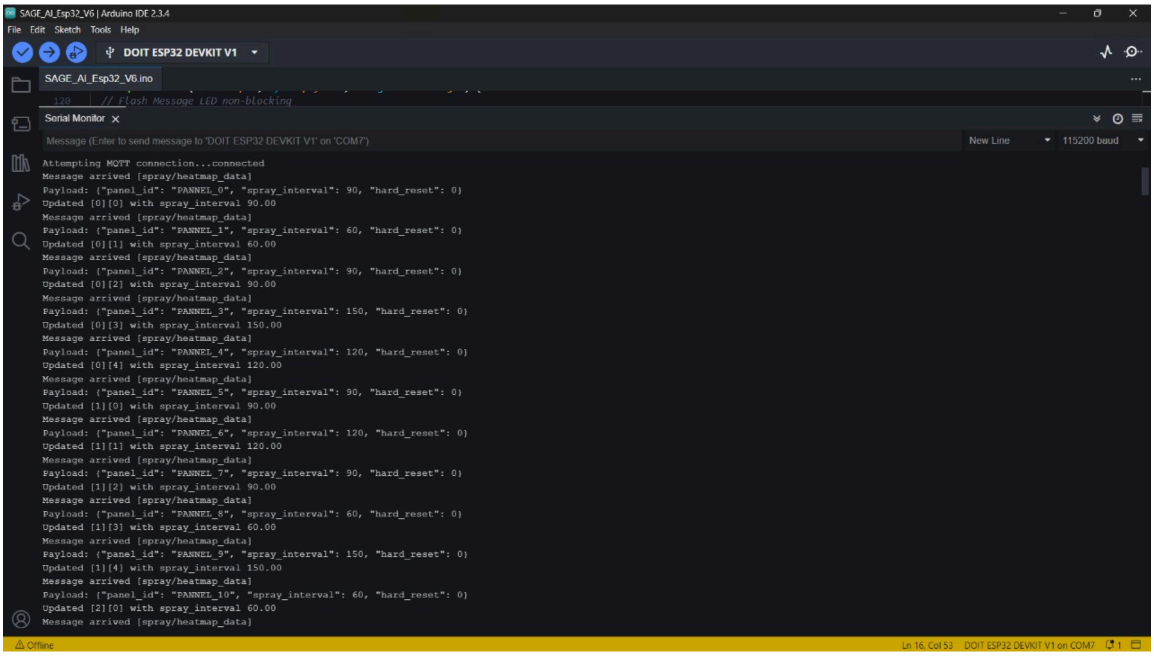
Heatmap Simulated Data Send via MQTT protocol



```python
def listen_for_reset():
                raise Exception("MQTT Disconnected")
            client.publish(TOPIC, json.dumps(payload))
            print(f"🚨 Immediate RESET sent to {TOPIC}: {payload}")
        except Exception as e:
            print(f"⚠ Immediate publish failed: {e}")
            print("🔄 Attempting reconnect...")
            connect_mqtt()

# Start background thread for real-time hard reset
threading.Thread(target=listen_for_reset, daemon=True).start()

# ========== SIMULATION FUNCTIONS ==========
def sunlight_factor(hour):
    return max(0, math.sin((hour - 6) / 12 * math.pi))

def generate_panel_data(hour, row, col):
    sun_factor = sunlight_factor(hour)

    base_temp = 22 + 15 * sun_factor
    base_humidity = 80 - 35 * sun_factor
    base_power = 250 * sun_factor
    base_light = 100000 * sun_factor

    temp = round(base_temp + random.uniform(-2, 2), 2)
    humidity = round(base_humidity + random.uniform(-3, 3), 2)
    light = int(base_light + random.uniform(-5000, 5000))
    power = int(base_power + random.uniform(-15, 15))
    efficiency = round(random.uniform(70, 95), 2)
    spray_interval = random.choice([60, 90, 120, 150])
    spray_duration = random.choice([5, 10, 15, 20])

    return {
        "timestamp": datetime.now().isoformat(),
```

No tests have been found in this workspace yet.

```
PS D:\Sage_AI_IOT> & d:/Sage_AI_IOT/venv/Scripts/python.exe d:/Sage_
AI_IOT/MQTT_OFFLINE/heat_map_v3.py
d:\Sage_AI_IOT\MQTT_OFFLINE\heat_map_v3.py:15: DeprecationWarning: C
allback API version 1 is deprecated, update to latest version
  client = mqtt.Client()
✅ Connected to MQTT broker!
📤 Sent to spray/heatmap_data: {'panel_id': 'PANNEL_0', 'spray_inter
val': 90, 'hard_reset': 0}
📤 Sent to spray/heatmap_data: {'panel_id': 'PANNEL_1', 'spray_inter
val': 60, 'hard_reset': 0}
📤 Sent to spray/heatmap_data: {'panel_id': 'PANNEL_2', 'spray_inter
val': 90, 'hard_reset': 0}
📤 Sent to spray/heatmap_data: {'panel_id': 'PANNEL_3', 'spray_inter
val': 150, 'hard_reset': 0}
📤 Sent to spray/heatmap_data: {'panel_id': 'PANNEL_4', 'spray_inter
val': 120, 'hard_reset': 0}
📤 Sent to spray/heatmap_data: {'panel_id': 'PANNEL_5', 'spray_inter
val': 90, 'hard_reset': 0}
📤 Sent to spray/heatmap_data: {'panel_id': 'PANNEL_6', 'spray_inter
val': 120, 'hard_reset': 0}
📤 Sent to spray/heatmap_data: {'panel_id': 'PANNEL_7', 'spray_inter
val': 90, 'hard_reset': 0}
📤 Sent to spray/heatmap_data: {'panel_id': 'PANNEL_8', 'spray_inter
val': 60, 'hard_reset': 0}
📤 Sent to spray/heatmap_data: {'panel_id': 'PANNEL_9', 'spray_inter
val': 150, 'hard_reset': 0}
📤 Sent to spray/heatmap_data: {'panel_id': 'PANNEL_10', 'spray_inte
rval': 60, 'hard_reset': 0}
📤 Sent to spray/heatmap_data: {'panel_id': 'PANNEL_11', 'spray_inte
rval': 150, 'hard_reset': 0}
📤 Sent to spray/heatmap_data: {'panel_id': 'PANNEL_12', 'spray_inte
rval': 90, 'hard_reset': 0}
📤 Sent to spray/heatmap_data: {'panel_id': 'PANNEL_13', 'spray_inte
rval': 60, 'hard_reset': 0}
```



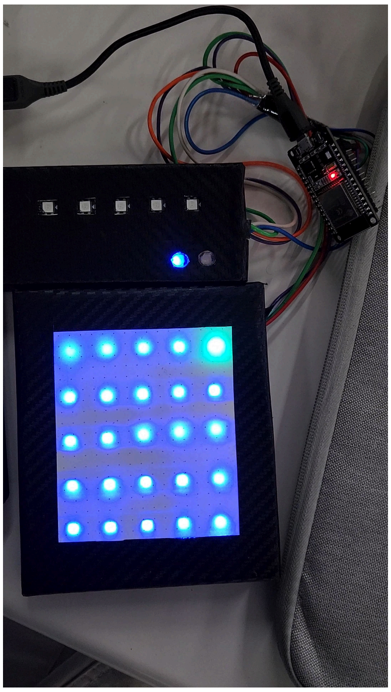
```
// Flash Message LED non-blocking
```

Message (Enter to send message to 'DOIT ESP32 DEVKIT V1' on 'COM7')

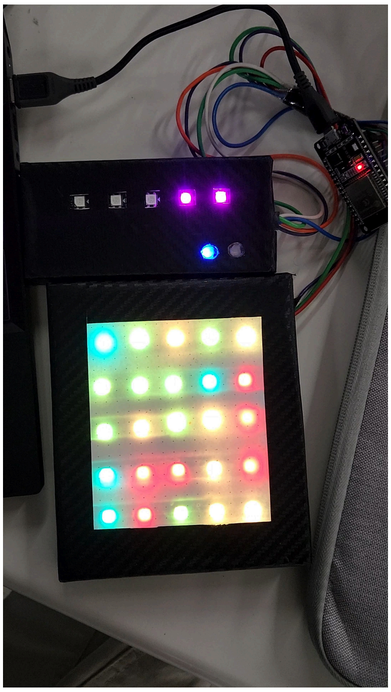
```
Attempting MQTT connection...connected
Message arrived [spray/heatmap_data]
Payload: {"panel_id": "PANNEL_0", "spray_interval": 90, "hard_reset": 0}
Updated [0][0] with spray_interval 90.00
Message arrived [spray/heatmap_data]
Payload: {"panel_id": "PANNEL_1", "spray_interval": 60, "hard_reset": 0}
Updated [0][1] with spray_interval 60.00
Message arrived [spray/heatmap_data]
Payload: {"panel_id": "PANNEL_2", "spray_interval": 90, "hard_reset": 0}
Updated [0][2] with spray_interval 90.00
Message arrived [spray/heatmap_data]
Payload: {"panel_id": "PANNEL_3", "spray_interval": 150, "hard_reset": 0}
Updated [0][3] with spray_interval 150.00
Message arrived [spray/heatmap_data]
Payload: {"panel_id": "PANNEL_4", "spray_interval": 120, "hard_reset": 0}
Updated [0][4] with spray_interval 120.00
Message arrived [spray/heatmap_data]
Payload: {"panel_id": "PANNEL_5", "spray_interval": 90, "hard_reset": 0}
Updated [1][0] with spray_interval 90.00
Message arrived [spray/heatmap_data]
Payload: {"panel_id": "PANNEL_6", "spray_interval": 120, "hard_reset": 0}
Updated [1][1] with spray_interval 120.00
Message arrived [spray/heatmap_data]
Payload: {"panel_id": "PANNEL_7", "spray_interval": 90, "hard_reset": 0}
Updated [1][2] with spray_interval 90.00
Message arrived [spray/heatmap_data]
Payload: {"panel_id": "PANNEL_8", "spray_interval": 60, "hard_reset": 0}
Updated [1][3] with spray_interval 60.00
Message arrived [spray/heatmap_data]
Payload: {"panel_id": "PANNEL_9", "spray_interval": 150, "hard_reset": 0}
Updated [1][4] with spray_interval 150.00
Message arrived [spray/heatmap_data]
Payload: {"panel_id": "PANNEL_10", "spray_interval": 60, "hard_reset": 0}
Updated [2][0] with spray_interval 60.00
Message arrived [spray/heatmap_data]
```

Simulation on Hardware:

1. Initial Start



2. Received simulated Data (PIXEL = 1 solar pannel)

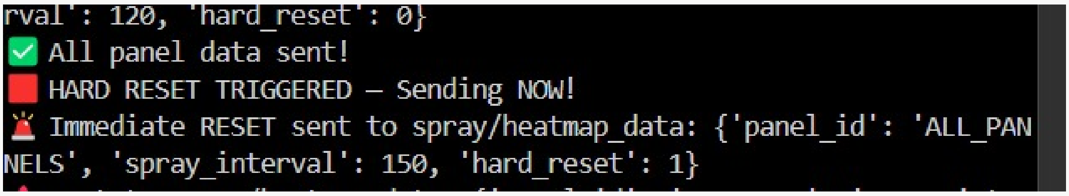


- **Color Logic:**
  - Spray Interval mapped to color spectrum

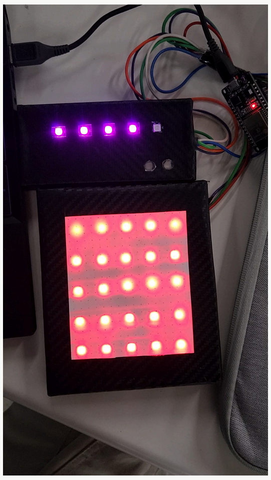
- Short = Blue → Cyan
- Short Medium = Cyan → Green ,
- High Medium = Green → Yellow,
- Large = Yellow → Red,

```
uint32_t getHeatColor(float interval) {
  float norm = constrain(interval / 150.0, 0.0, 1.0);
  byte r, g, b;

  if (norm < 0.25) {        // Blue → Cyan
    r = 0;
    g = norm * 4 ∗ 255;
    b = 255;
  } else if (norm < 0.5) {  // Cyan → Green
    r = 0;
    g = 255;
    b = (1 - (norm - 0.25) * 4) * 255;
  } else if (norm < 0.75) { // Green → Yellow
    r = (norm - 0.5) * 4 ∗ 255;
    g = 255;
    b = 0;
  } else {                  // Yellow → Red
    r = 255;
    g = (1 - (norm - 0.75) * 4) * 255;
    b = 0;
  }

  return heatMap.Color((int)r, (int)g, (int)b);
}
```

💡 Esp32 Logic Supports HARD RESET in case of emergency situations where we need large spr5ay intervals. Hard reset logic is handeled via MQTT DATA itself



```
rval': 120, 'hard_reset': 0}
✅ All panel data sent!
🟥 HARD RESET TRIGGERED — Sending NOW!
🚨 Immediate RESET sent to spray/heatmap_data: {'panel_id': 'ALL_PAN
NELS', 'spray_interval': 150, 'hard_reset': 1}
```



- **Result:**

  Real-time feedback on which panels are using more water ⇒ i.e. a large SPRAY Interval

  Therefore helps visually identify defects / issues

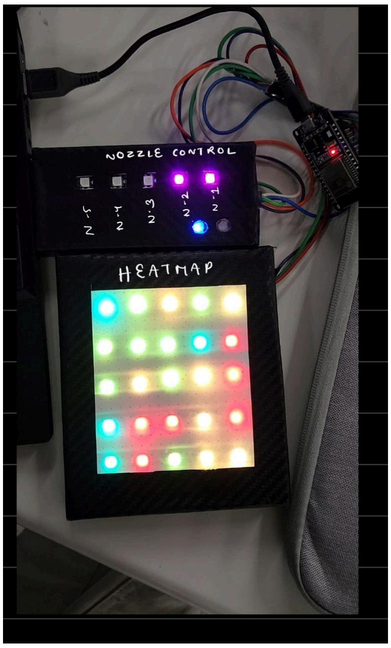
🧩Section 3: Nozzle Control via MQTT

- **Command Topic:** `spray/control`
- **Message Format:**

```
json
   Spray_command = {
   "nozzle_id" : f"NOZZLE_{i}",
   "amount_ml" : 20*i,
   "spray_timeout" : 10*i
   }
   send_command(Spray_command , topic) #just a simulated conceptual data
```

- **ESP32 Action:**
  - Trigger spray routine for selected panel ID
  - Run servo/solenoid for given duration (in our case NEOPixel LED)
- **Hard Reset Support:**
  - Full reset or refresh via `reset_all` command

Nozzle Controlling Via MQTT Protocol and Esp32



Z-5
Z-4
Z-3
Z-2
Z-1