Smart Tanning Device

status active

Smart Tanning Device

Table of Contents

About

This repo contains

- Backend
- Firmware
- Client auto-Installer script
- Detailed instructions

for Smart Tanning Device.

Getting Started

These instructions will get you a copy of the project up and running on your system.

Prerequisites

Things you need to install the FW.

```
- Arduino IDE
```

Installing

A step by step series that tell you how to get the Firmware and Backend running

ESP32 Configuration

You should have Arduino IDE Installed

1. Add ESP32 Board to your Arduino IDE
2. In your Arduino IDE, go to File> Preferences Installing ESP32 Add-on in Arduino IDE Windows, Mac OS X, Linux open preferences
3. Enter `https://dl.espressif.com/dl/package_esp32_index.json` into the “Additional Board Manager URLs” field then, click the “OK” button: Note: if you already have the ESP32 boards URL, you can separate the URLs with a comma(each board will go to neaw line) as follows: `https://dl.espressif.com/dl/package_esp32_index.json,\n http://arduino.esp8266.com/stable/package_esp8266com_index.json`
4. Open the Boards Manager. Go to Tools > Board > Boards Manager…
5. Search for ESP32 and press install button for the ESP32 by Espressif Systems“:
6. That’s it. It should be installed after a few seconds.
7. In your Arduino sketchbook directory, create tools directory if it doesn't exist yet.
8. Unpack the tool into tools directory(present in libs/ESP32FS-1.0.zip) (the path will look like <home_dir>/Arduino/tools/ESP32FS/tool/esp32fs.jar).
9. Close and re-open the Arduino IDE.
10. Now copy the contents of the libs folder to the libraries directory of your Arduino
    1. If you are using windows, the libraries directory will be Documents/Arduino/libraries

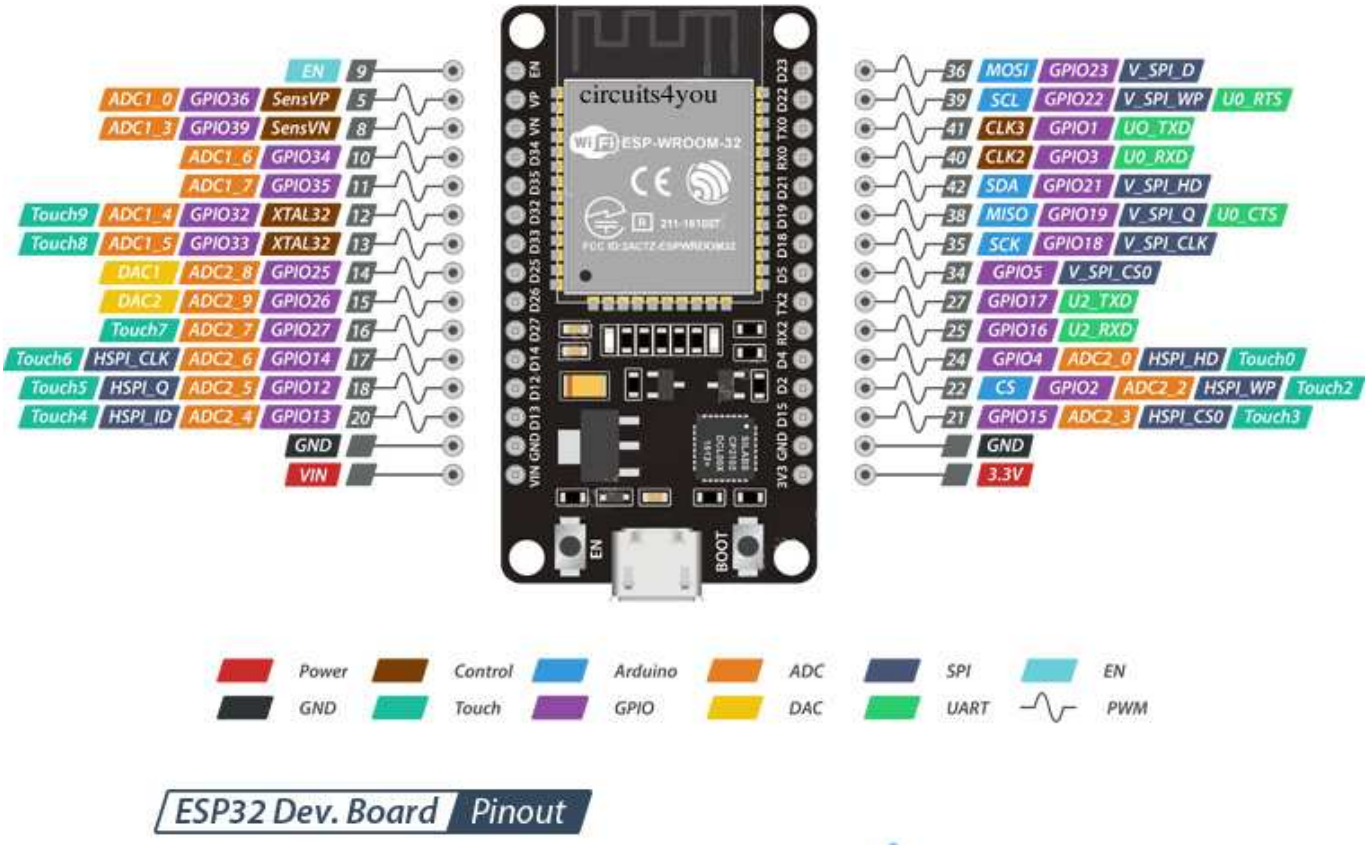
ESP32 Node FW Uploading

1. Select ESP32 Dev Module from Tools->Board->ESP32
2. Select the correct port from Tools->Port
3. Then open Firmware.ino file,
4. Select Tools > ESP32 Sketch Data Upload menu item. This should start uploading the files into ESP32 flash file system.
5. Now Upload the Code to your ESP32 Dev Module.
6. Your ESP32 is now ready to be used.

Circuit

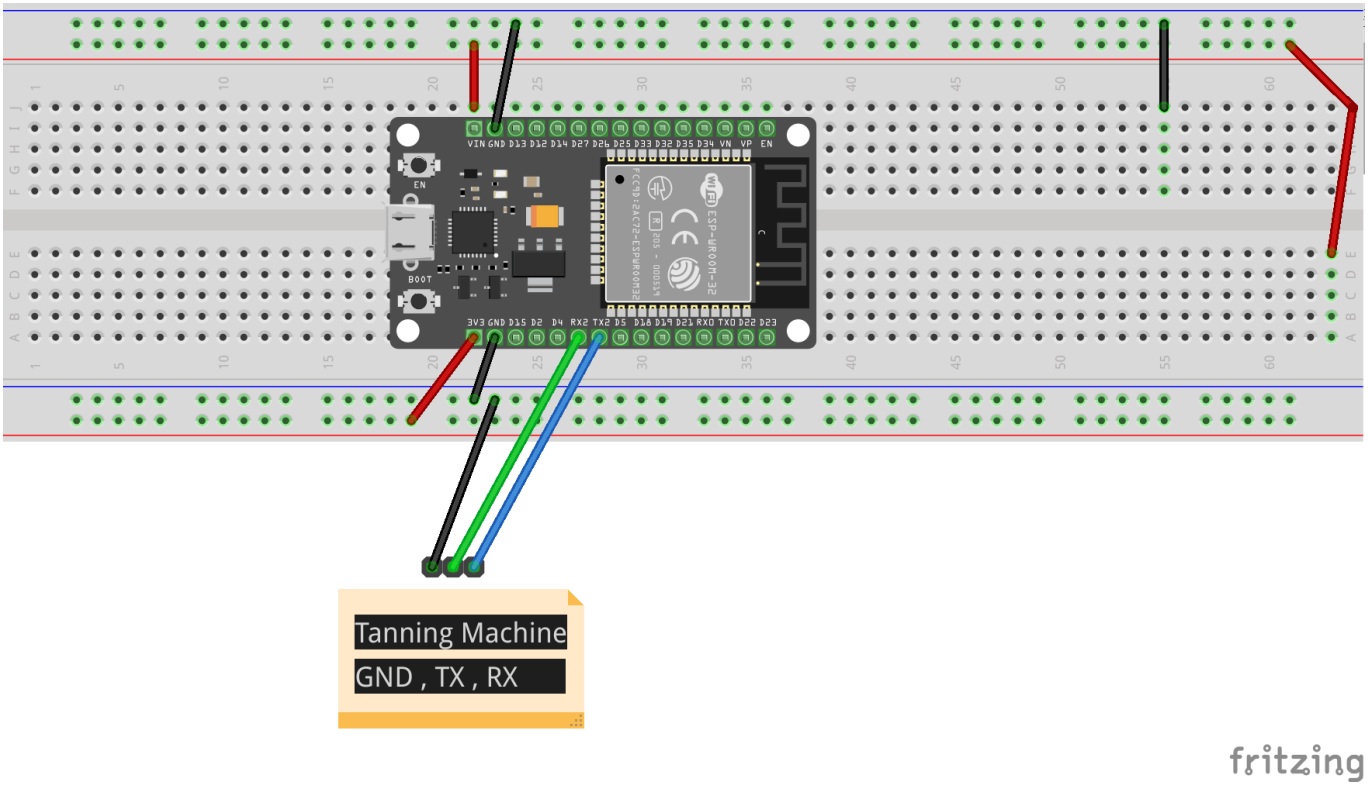
ESP32 Dev Module Pinout

Follow the pinout diagram given below to connect different components to your TTGO LORA32 board.



Complete Circuit Diagram

Here's the complete circuit diagram of the system.



Server Details

Monitoring

- NetData Monitro
- https://captain.app.cloudsolarium.com/net-data-monitor

List of Packages installed on server

- Mosquitto Broker
- NodeJS, NPM, Node
- Docker
- ufw
- CapRover

Version Details

- Node v10.19.0
- NPM v6.14.4

Reference Links

- Layout: https://docs.google.com/spreadsheets/d/1pLSYBU_jmj0MHLfFhjrONCmabcauOlFwBAStOadKreI/edit

Server Links

- MQTT Broker Link: 34.214.65.82:1883
- CapRover: https://captain.app.cloudsolarium.com/
- Backend Link: https://smart-tanning-device-backend.app.cloudsolarium.com
- Frontend Link: https://smart-tanning-device-frontend.app.cloudsolarium.com/

Backend

- Backend is based on NodeJS and it is being run as a Docker Container and managed by CapRover. It starts automatically on server start.

Database

MongoDB is used as a database and the data format is shown belo

```
macAddress: {
      type: String,
      required: true,
    },
    Alive: {
      type: String,
      unique: true,
    },
    TotalRunningTime: {
      type: String,
      unique: true,
```

```
    },
    TotalSessionCount: {
      type: String,
      unique: true,
    },
    TotalSessionCorrectlyEnded: {
      type: String,
      unique: true,
    },
    TotalSessionEndedBeforeTime: {
      type: String,
      unique: true,
    },
    TotalSessionNotEndedCorrectly: {
      type: String,
      unique: true,
    },
    StartSession: {
      type: String,
      unique: true,
    },
    EndSession: {
      type: String,
      unique: true,
    },
    EndSessionType: {
      type: String,
      unique: true,
    },
    Temperature: {
      type: String,
      unique: true,
    },
    AnemometerSensor: {
      type: String,
      unique: true,
    },
    PresencePhases: {
      type: String,
      unique: true,
    },
    SensorFilters: {
      type: String,
      unique: true,
    },
    LampMaintenance: {
      type: String,
      unique: true,
    },
    AnnualMaintenance: {
      type: String,
      unique: true,
    },
    ActualLastTemp: {
```

```
      type: String,
      unique: true,
    },
    HighestTemp: {
      type: String,
      unique: true,
    },
    PowerFactorCorrection: {
      type: String,
      unique: true,
    },
    PFDeviationFromOptimalLevel: {
      type: String,
      unique: true,
    },
    LastFanSpeed: {
      type: String,
      unique: true,
    },
    InputVoltage: {
      type: String,
      unique: true,
    }
```

MQTT Topic Details

Topics List

From ESP32 Perspective

- smartdata/# (WRITE-ONLY) The data should be send as a JSON with keys mentioned above in the `Database` Section
- {macAddress}/poll (READ-ONLY) ESP32 with the macAddress will receive the `poll` string on this topic when `Poll` button is pressed on the dashboard.
- {macAddres}/fieldData (READ-ONLY) ESP32 with the selected macAddress witll receive the editable fields data when `Transmit` button is pressed on the Dashboard.

Fimrware

API Details

Admin Login

```
POST https://smart-tanning-device-backend.app.cloudsolarium.com/api/users/login
```

Parameter	Type	Description
`Email`	`string`	**Required**. *Email address of the admin*

Parameter	Type	Description
`Password`	`string`	**Required**. *Password of the admin*

Responses

Many API endpoints return the JSON representation of the resources created or edited. However, if an invalid request is submitted, or some other error occurs, Gophish returns a JSON response in the following format:

```
{
  "status"  : int,
  "message" : string
}
```

The `message` attribute contains a message commonly used to indicate errors or to return the logged status/

The `status` attribute describes if the transaction was successful or not.

Status Codes

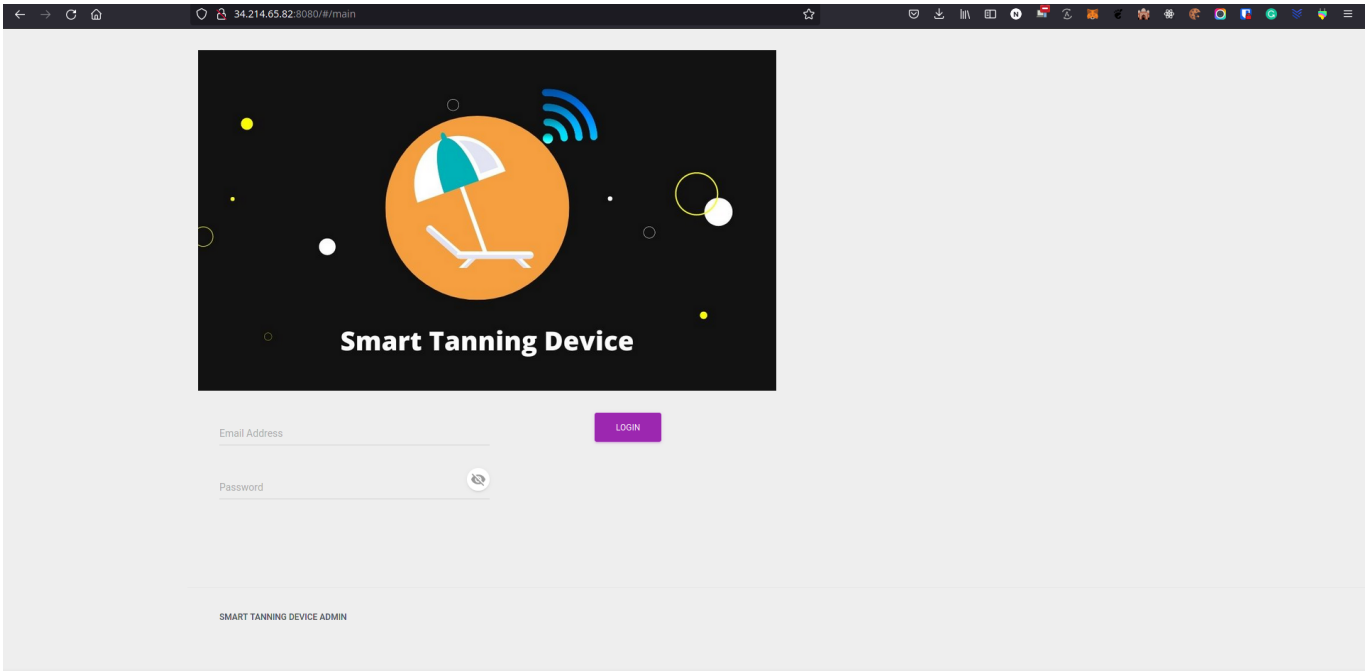
IoTManagementSystem Backend returns the following status codes in its API:

Status Code	Description
200	`OK`
201	`CREATED`
400	`BAD REQUEST`
404	`NOT FOUND`
500	`INTERNAL SERVER ERROR`

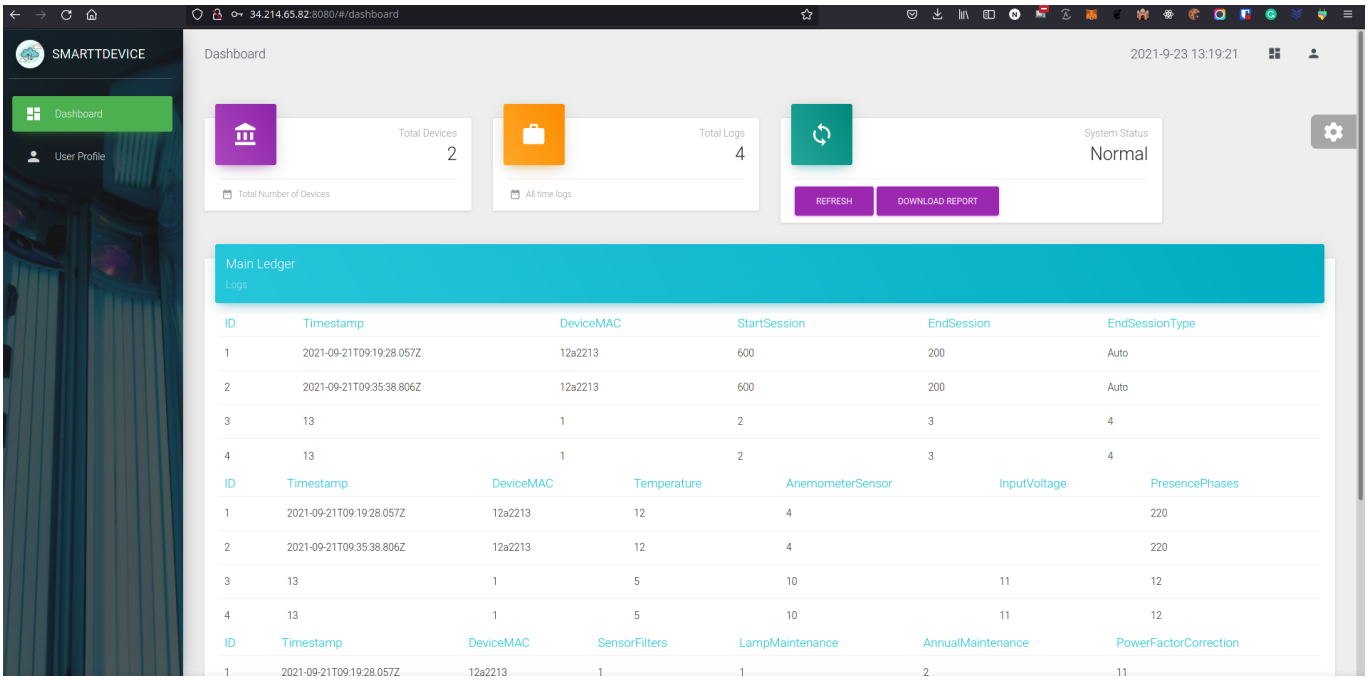
Usage

1. Upload the code to your ESP32.
2. Connect the ESP32 with your Tanning Machine.
3. Open the dashboard to monitor the parameters.
    1. Dashboard Default credentials
        1. Email Address: **admin@tanningdevice.com**
        2. Password: **admin**

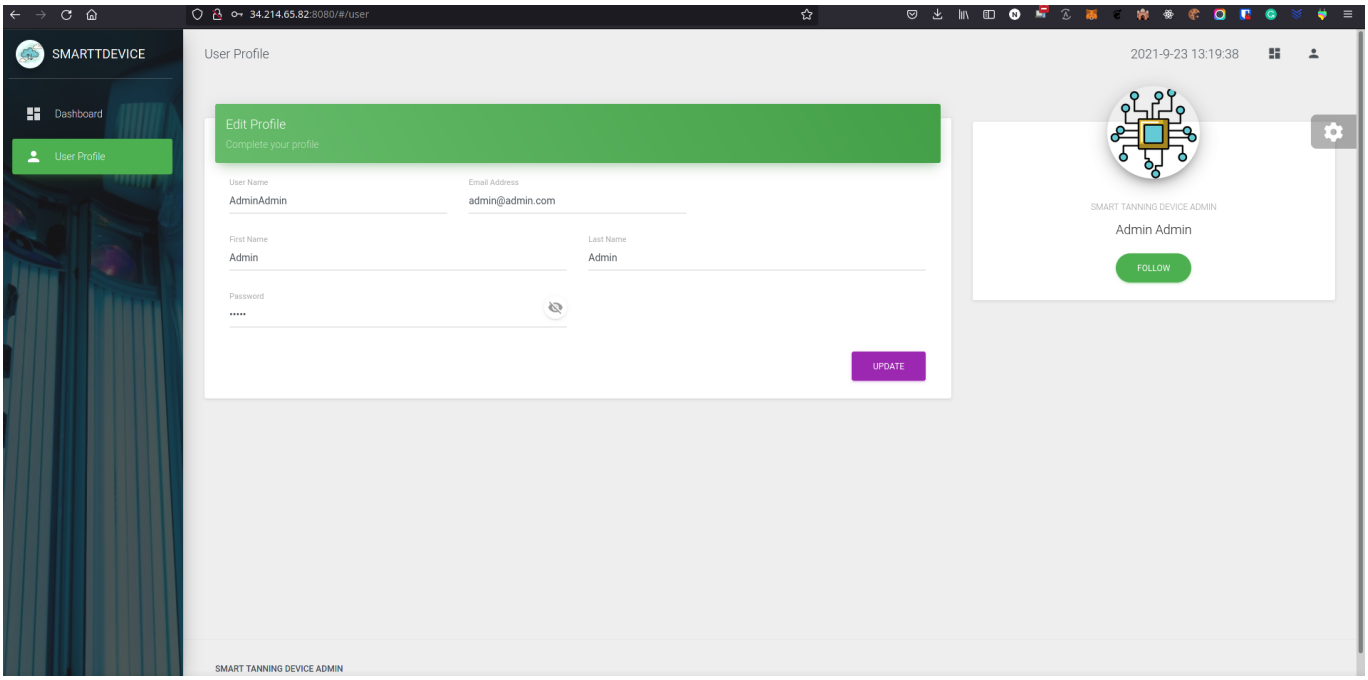
Dashboard Login Page



Dashboard Home Page



Dashboard Profile Page



4. You can also download the logs in CSV format from the dashboard home page. 5. Power on your ESP32, it will present you with an AP named `TanningD-abc` (while `TanningD` can be changed in the portal and `abc` is a unique id for each esp32) 6. Default captive portal password `123456789AP` which can be changed in captive portal. 7. Connect to the ESP32 access point and open the web-browser and navigate to the link `http://esp32.local/_ac`. This link will work on most of the operating systems but if your operating system is not allowing to open it, you may want to check the captive portal IP Address from the serial monitor and can use that IP address inplace of the above mentioned URL. 8. The default access IP Address is `http://192.168.4.1/_ac` 9. You will be presented with a main dashboard as shown below(based on your device)

5. Once connected to a WiFi network, you can again access the captive portal using same URL or the IP Address from the Serial monitor.
6. The data is published to the MQTT Topic `TanningD/{hostname}` while the hostname is the one which you can define in Settings page of the captive portal.
7. You can open settings page with following default credentials
    1. User: **AP Name (TanningD)**
    2. Password: **admin**

List of Components

Following components are used to make this project

1. ESP32 Dev Kit Module https://www.amazon.com/HiLetgo-ESP-WROOM-32-Development-Microcontroller-Integrated/dp/B0718T232Z/ref=sr_1_3?crid=5EOAXOANUSCU&dchild=1&keywords=esp32+nodemcu&qid=1629587138&sprefix=esp32+node%2Caps%2C201&sr=8-3

🔨 Built Using

- NodeJS - JS Framework for Backend Programming
- Eclipse Paho MQTT - MQTT Client for Backend and RPiClient Software
- Arduino - Embedded Framework and IDE - For Sensor Node Design
- VueJS - For Dashboard Design

✍️ Authors

- @Nauman3S - Development and Deployment