REVOLUTIONIZING PET CARE

THE IOT SMART DOG KENNEL PROJECT

Enhancing Comfort, Safety, and Interaction for Man's Best Friend

Dvir Zaguri

8.3.2024

Dog kennels way-up

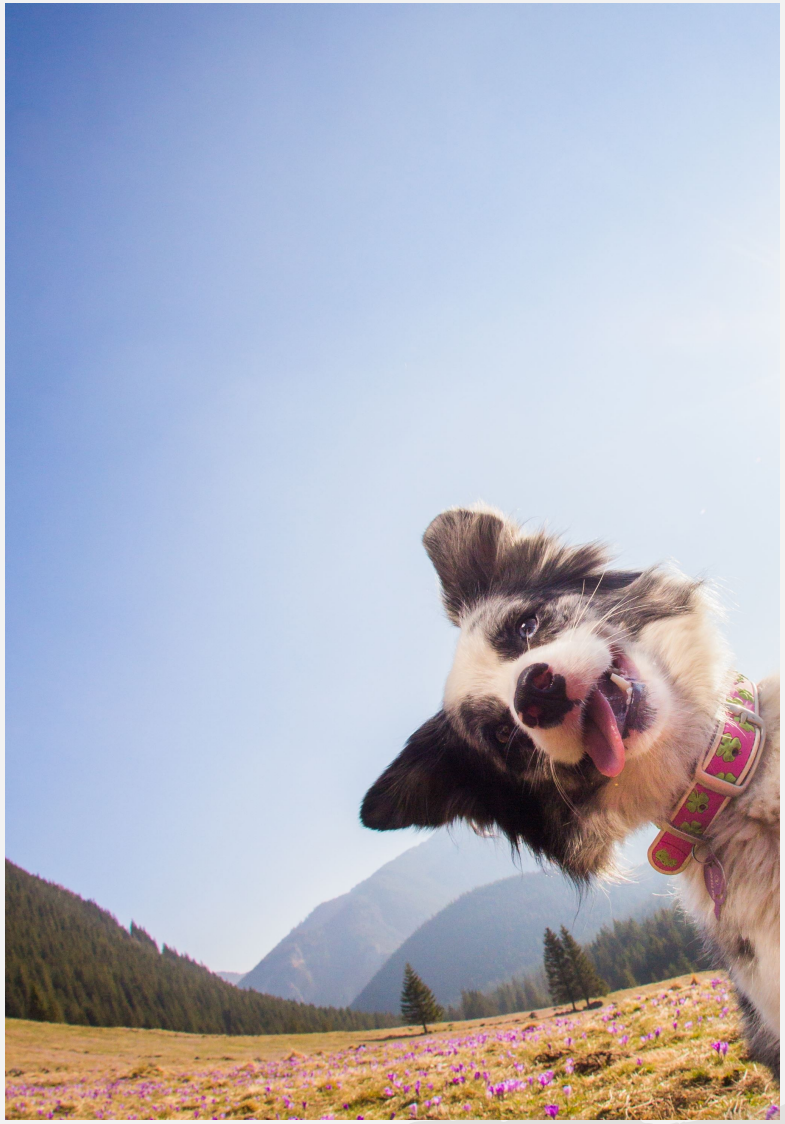
As pet owners, we understand the importance of providing the best care and comfort for our furry companions.

However, traditional dog kennels often lack the features needed to ensure our dogs' well-being when we're away.

This is where the necessity of a smart dog kennel becomes evident.

What are our dog's needs ?

- Ideal Conditions
- Staying Hydrated and
- Steady Nutrition
- Attention
- Good Sleep habits



Ideal Conditions

Real-time temperature and humidity data are prominently displayed on an easy-to-read screen within the kennel.

Pet owners can conveniently access temperature and humidity data from anywhere via a user-friendly web panel and can take action to adjust it!

Hydration and Nutrition

Our smart pet feeding system allows pet owners to remotely schedule and control the filling of their dog's food and water bowls, ensuring consistent nutrition and hydration even when they're away from home.

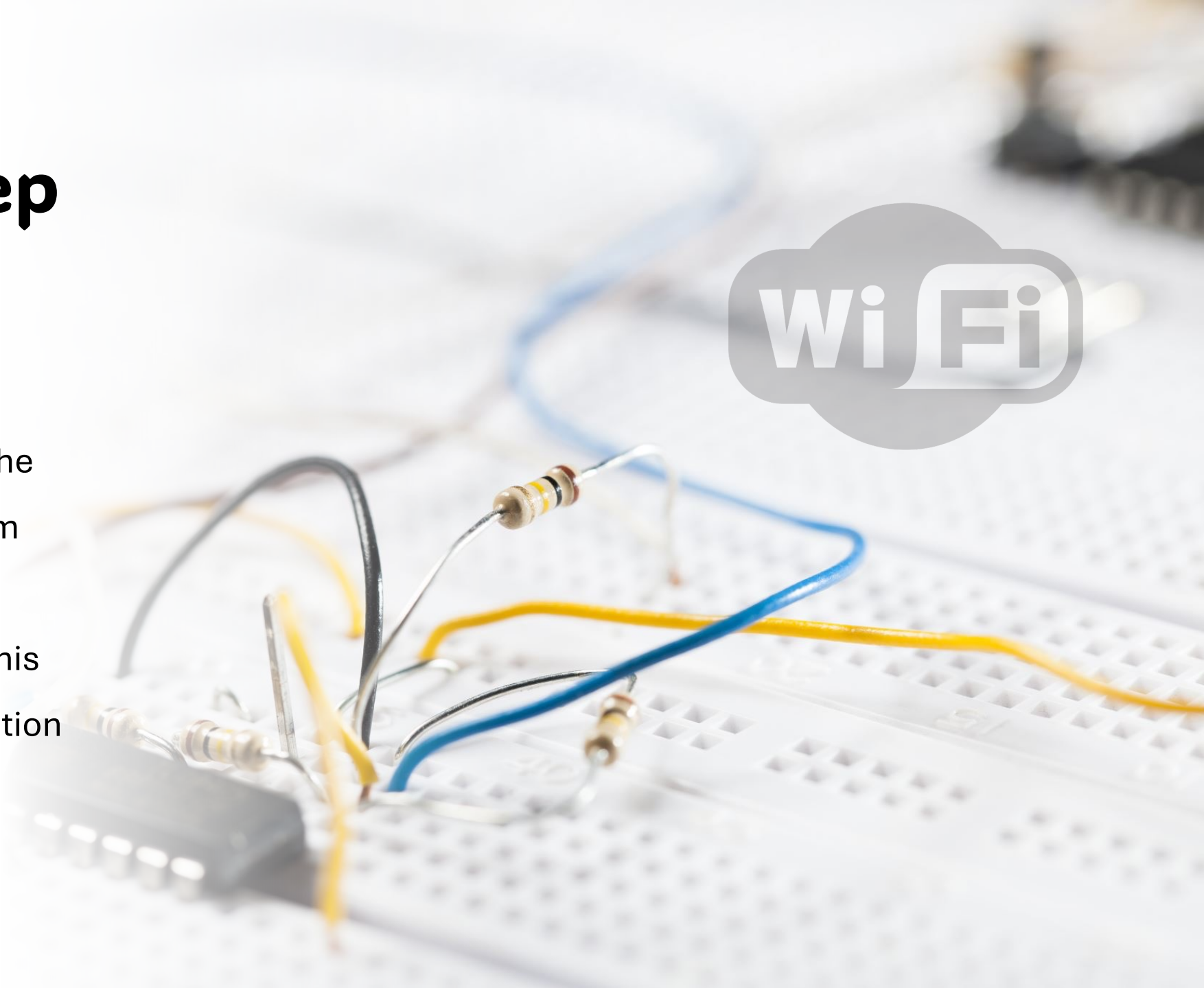
Attention and Sleep habits

The integrated sensor in our smart kennel provides real-time data on the amount of time they spend inside, as well as when and for how long. This helps pet owners maintain a healthy balance of activity and rest for their furry friends.

That’s How it WORKS!

The First Step

The ESP32 serves as the foundational platform for the system, gathering data from sensors via a secured WiFi network. It then forwards this data to the subsequent station for further processing.



biggest_mult_sens | Arduino IDE 2.3.2

ESP32-WROOM-DA M...

biggest_mult_sens.ino | certs.h | utils.h

```
  doc["temperature"] = dht_sensor.readTemperature();
  doc["humidity"] = dht_sensor.readHumidity();
  doc["traffic_lights_status"] = tl;
  doc["timestamp"] = millis();

// Get current time from system clock in milliseconds since epoch
  // Serialize the JSON object to a buffer
  char jsonBuffer[256];
  serializeJson(doc, jsonBuffer);

  // Publish the JSON string to the MQTT topic
  client.publish(AWS_IOT_PUBLISH_TOPIC, jsonBuffer);

  delay(2500); // Publish every 2 seconds
}
```

Output | Serial Monitor

Message (Enter to send message to 'ESP32-WROOM-DA Module' on '/dev/cu.usbserial-0001') | No Line Ending | 115200 baud

```
16:09:49.997 -> load:0x3fff0030,len:1344
16:09:49.997 -> load:0x40078000,len:13964
16:09:49.997 -> load:0x40080400,len:3600
16:09:49.997 -> entry 0x400805f0
16:09:54.949 -> Connecting to Wi-Fi
16:09:55.447 -> ...Connecting to AWS IOT
16:09:58.591 -> AWS IoT Connected!
```

Ln 63, Col 1 ESP32-WROOM-DA Module on /dev/cu.usbserial-0001 2

One Scope Outside

After connecting to the secured WiFi the ESP32 connects to a MQTT broker of AWS

Starts publishing its raw sensors data to a topic named ‘esp32/pub’

subscribe to another topic ‘esp32/sub’ there he receives the commands to control the I/O.

AWS IoT

Monitor

Connect
- Connect one device
- Connect many devices

Test
- Device Advisor
- MQTT test client
- Device Location New

Manage
- All devices
- Greengrass devices
- LPWAN devices
- Software packages New
- Remote actions
- Message routing

Subscriptions

esp32/pub

esp32/analyze

esp32/pub

Pause Clear Expo

Message payload

```
{
  "message": "Hello from AWS IoT console"
}
```

Additional configuration

Publish

esp32/pub February 24, 2024, 16:10:5

```
{
  "temperature": 23,
  "humidity": 73,
  "traffic_lights_status": 0,
  "timestamp": 68462
}
```

Properties

© 2024, Amazon Web Services, Inc. or its affiliates. Privac

AWS IoT

Monitor

Connect
- Connect one device
- Connect many devices

Test
- Device Advisor
- MQTT test client
- Device Location New

Manage
- All devices
- Greengrass devices
- LPWAN devices
- Software packages New
- Remote actions
- Message routing

Subscriptions

esp32/pub

esp32/analyze

esp32/analyze

Pause Clear Expo

Message payload

```
{
  "message": "Hello from AWS IoT console"
}
```

Additional configuration

Publish

esp32/analyze February 24, 2024, 16:11:0

```
{
  "temp": 23,
  "hum": 73,
  "TL": 0,
  "timestamp": "1970-01-01T21:02:46Z"
}
```

Properties

© 2024, Amazon Web Services, Inc. or its affiliates. Privac

The Bigger Picture

The MQTT Broker connects to AWS, where raw data is processed and reformatted for AWS Analytics. This data is then used to generate datasets, which are visualized using AWS Quicksight for actionable insights.

AWS IoT Analytics

Introducing the new IoT Analytics console experience

Result preview

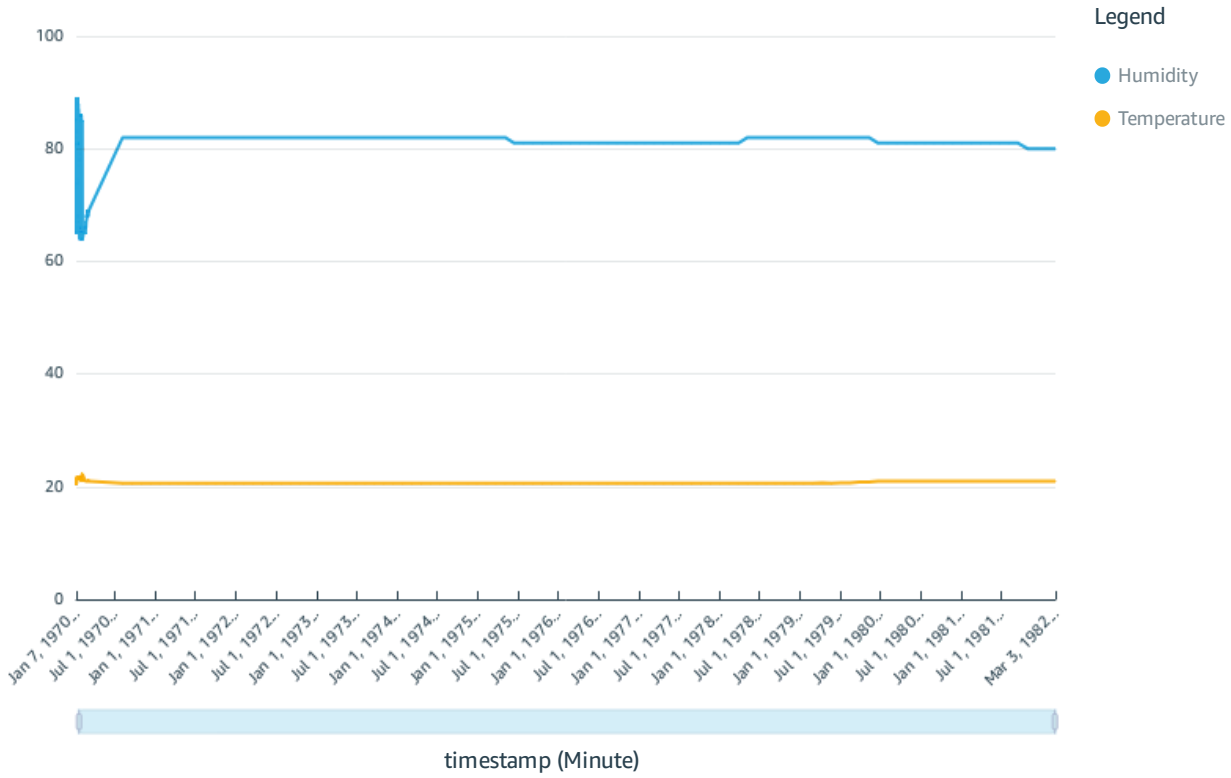
c7354afd-8f62-4a9d-af29-f692c623cada.csv

temp	hum	tl	timestamp	time	__dt
21.20000076	89	0	1970-01-08T10:20:55Z		2024-02-18 00:00:00.000
21.20000076	89	0	1970-01-08T11:21:51Z		2024-02-18 00:00:00.000
21.20000076	89	0	1970-01-08T12:22:44Z		2024-02-18 00:00:00.000
21.29999924	89	0	1970-01-08T13:23:38Z		2024-02-18 00:00:00.000
21.29999924	89	0	1970-01-08T14:24:32Z		2024-02-18 00:00:00.000
21.39999962	89	0	1970-01-08T15:25:25Z		2024-02-18 00:00:00.000
21.39999962	89	0	1970-01-08T16:26:20Z		2024-02-18 00:00:00.000
21.39999962	89	0	1970-01-08T17:27:15Z		2024-02-18 00:00:00.000
21.0	66	0	1970-02-12T19:56:51Z		2024-02-18 00:00:00.000

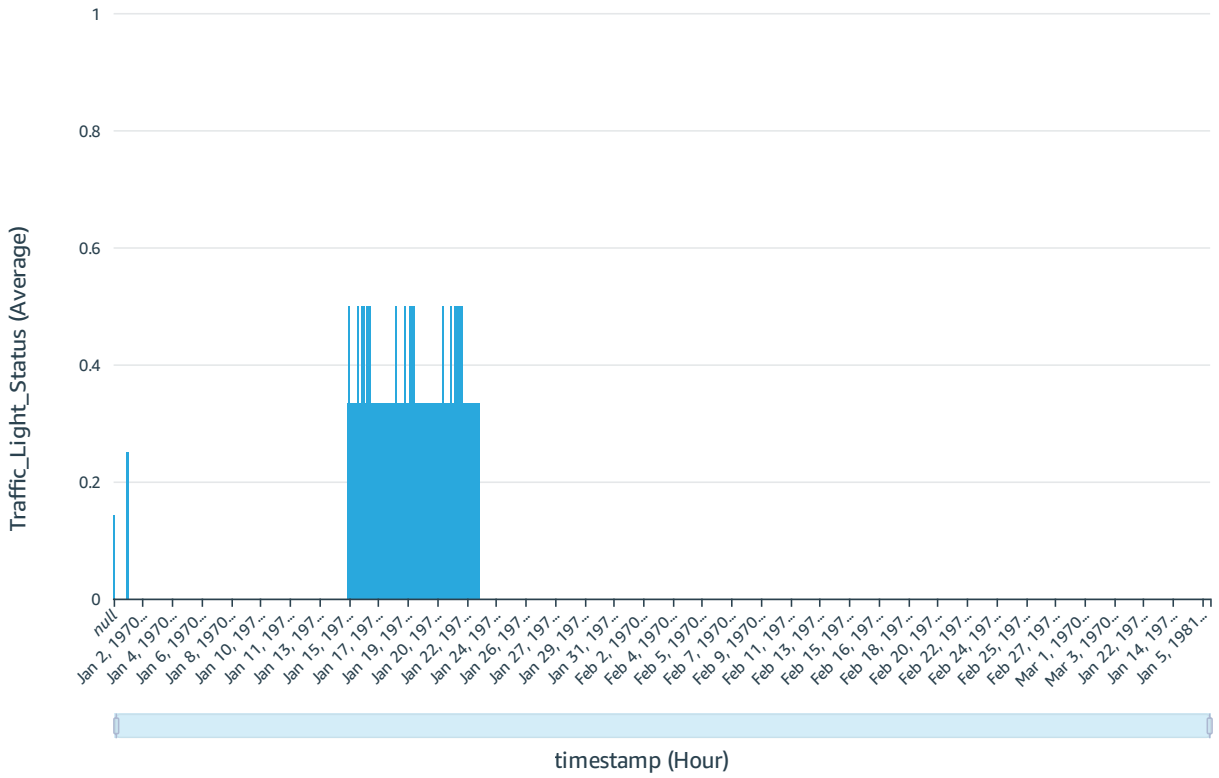
© 2024, Amazon Web Services, Inc. or its affiliates.

Average of Temperature and Average of Humidity by Timestamp

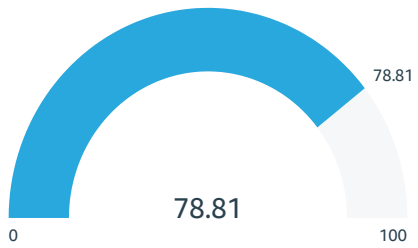
SHOWING TOP 2500 IN TIMESTAMP



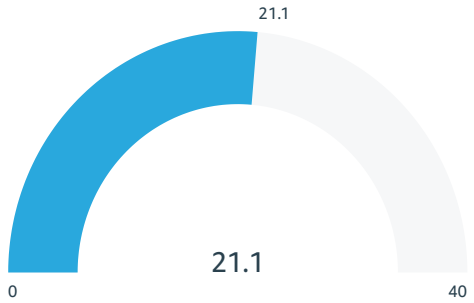
Average of Traffic_light_status by Timestamp



Average of Humidity



Average of Temperature



This is my esp32 AWS QuickSight visualization of data transmitted my the esp itself according to its automatic sensor rountines.

copyrights belogs to Dvir Zaguri