

Interacting with Tesla vehicles locally over BLE using ESPHome

Yasir Ekinici

<https://github.com/yoziyu/esphome-tesla-ble>

<https://github.com/yoziyu/tesla-ble>

yoziyu/**esphome-tesla-ble**



Interact with Tesla vehicles over BLE using ESPHome and Home Assistant

5
Contributors

26
Issues

151
Stars

27
Forks



Dynamic charging with Home Assistant



Solar excess charging

Battery boost

Load balancing

Overload protection

Off peak charging

Capacity tariff limits

Update every 10 seconds

= up to 8.640 vehicle commands per day

The screenshot shows the Home Assistant interface for an EV charging station. At the top, it displays 'EV' and a battery level of 81%. Below this are two adjustable settings: 'Solar Start' set to 60 and 'Charge Limit' set to 80. A central row shows 'Battery' at 81%, 'Actual' current at 0 A, and 'Desired' current at 0 A. At the bottom, three status cards are visible: 'Load balance' is On, 'Off peak' is Off, and 'Charge 100%' is Off (Last week).

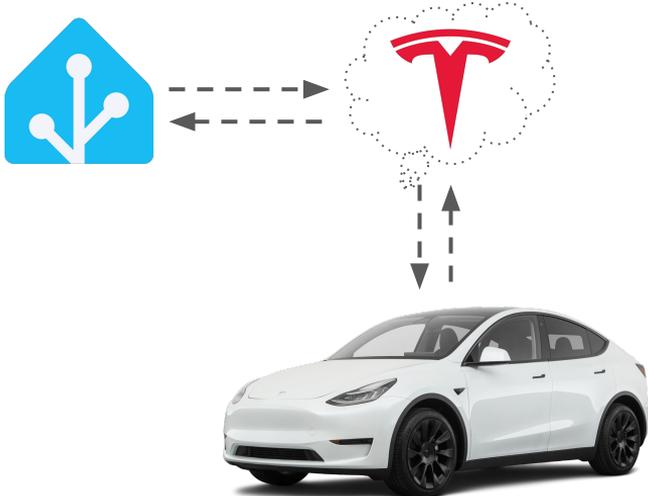
Setting	Value
Solar Start	60
Charge Limit	80
Battery	81 %
Actual	0 A
Desired	0 A
Load balance	On
Off peak	Off
Charge 100%	Off · Last week

Relying on cloud = bad

Tesla API rate limits introduced

8.640 > 50 requests / day

Scrambled for solutions



<  r/homeassistant · 7 mo. ago
ezfrag2016

Tesla Fleet API request limits - how are you coping?

I recently had to switch to using the Fleet API since the Tesla integration stopped sending commands. Unfortunately I'm struggling with the API request limits.

I was using HA to vary the charge amps every minute during daytime charging to use excess solar energy but since we only have 50 commands per day this is now screwed.

How have you guys changed your Tesla automations to cope?

19:04

TESLA Developer Menu

Subscription	plan	Discovery
Device data limits	200 requests / vehicle / day	
Commands limits	50 requests / vehicle / day	
Wake limits	15 requests / vehicle / day	
Charging commands	5 request / vehicle / day	
Energy Device data limits	50k API requests / day And 5 API requests / sec And 1 API request / energy site / 5 min.	
Energy Settings limits	50k API requests / day And 5 API requests / sec And 5 API requests / energy site / day And 1 tariff change / energy site / month.	
Capabilities	API access Login with Tesla Tesla for Business access Enterprise charging API Invoices Limited Fleet Telemetry streaming	

developer.tesla.com

Could it be done locally? No internet, no rate limits

Why ESP32?

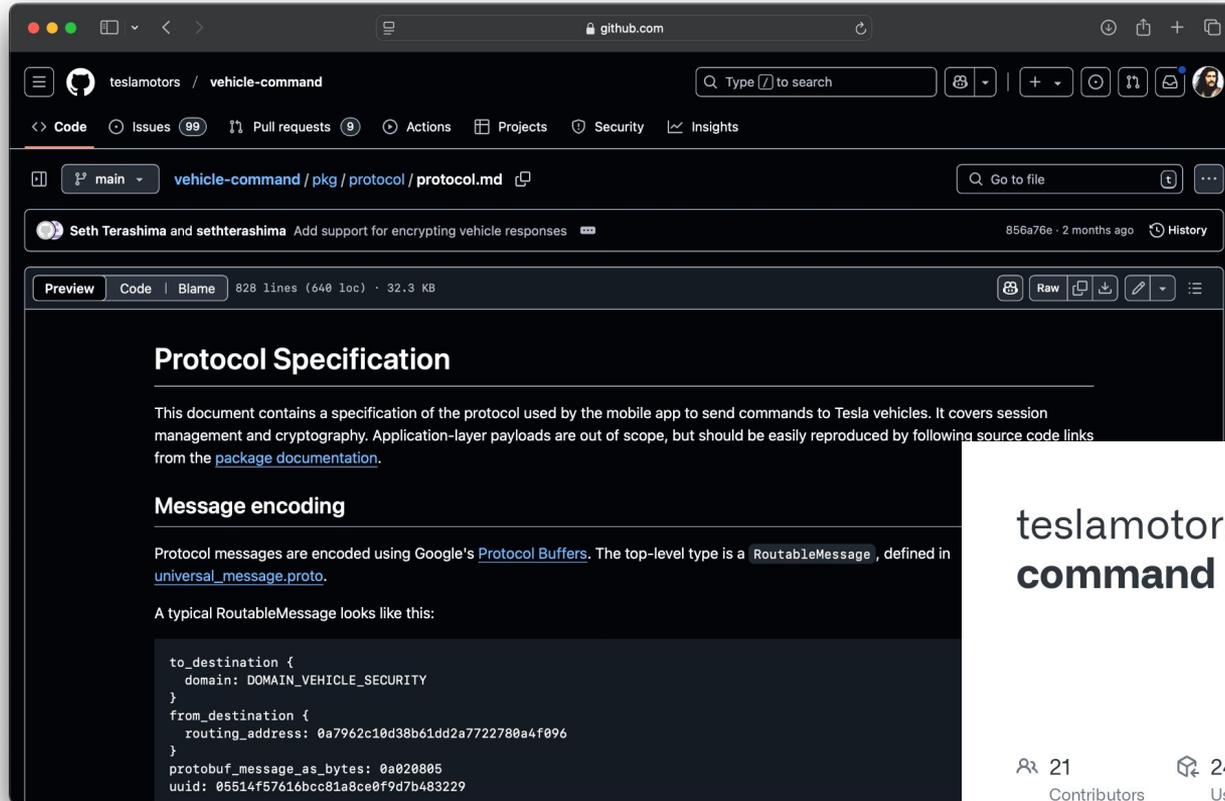
- Low cost, low power
- Small = easy to place close to car
- BLE built-in

Challenge:

- Limited 2MB flash, 512kB SRAM
- Low power: single core 160 Mhz
- ESPHome with custom C++



Tesla OSS: vehicle-command (Golang)



The screenshot shows the GitHub repository page for `teslamotors/vehicle-command`. The repository is on the `main` branch, and the file `pkg/protocol/protocol.md` is selected. The document is titled "Protocol Specification" and contains the following text:

This document contains a specification of the protocol used by the mobile app to send commands to Tesla vehicles. It covers session management and cryptography. Application-layer payloads are out of scope, but should be easily reproduced by following source code links from the [package documentation](#).

Message encoding

Protocol messages are encoded using Google's [Protocol Buffers](#). The top-level type is a `RoutableMessage`, defined in [universal_message.proto](#).

A typical `RoutableMessage` looks like this:

```
to_destination {
  domain: DOMAIN_VEHICLE_SECURITY
}
from_destination {
  routing_address: 0a7962c10d38b61dd2a7722780a4f096
}
protobuf_message_as_bytes: 0a020805
uuid: 05514f57616bcc81a8ce0f9d7b483229
```

teslamotors/**vehicle-**
command



21
Contributors

24
Used by

478
Stars

115
Forks



Tesla Vehicle Protocol: 2 subsystems

VCSEC (Vehicle Security)

- Controls locks, remote start, trunk
- Request simple data about vehicle state: lock state, sleep status, user presence
- Always listening over BLE

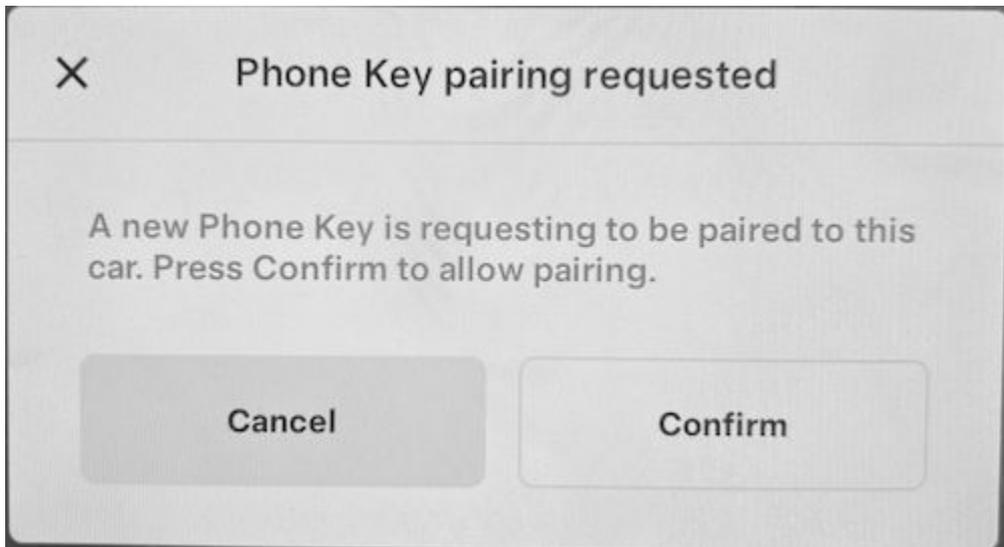
INFOTAINMENT

- All other commands: charging, heating, ..
- Request more data about vehicle (added in recent update 3 months ago!)
- Goes to sleep when not active
 - Consumes significantly more power!

Each domain has different public keys

→ separate handshakes + session state!

ESPHome pairing



Controls

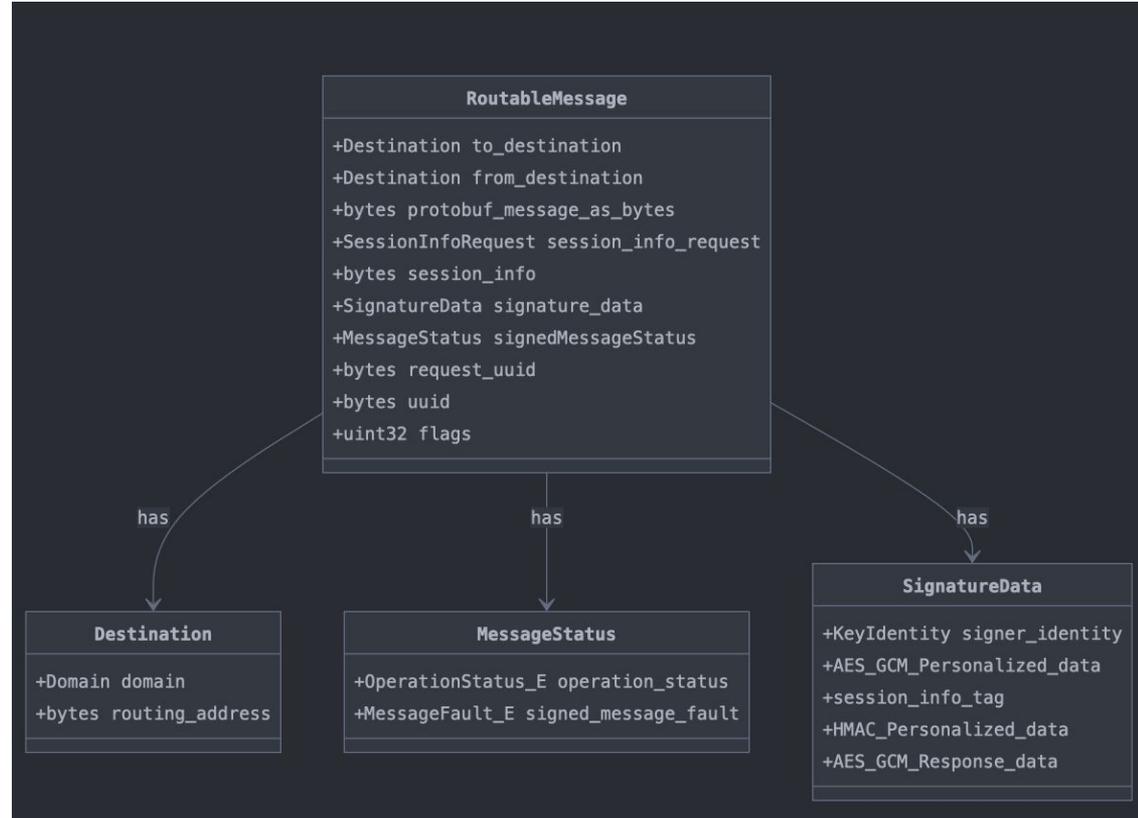
-  Charger switch  
-  Charging amps  14 A
-  Charging limit  98%
-  Wake up PRESS

Sensors

-  Asleep Off
-  Charge flap Open
-  Doors Locked
-  User presence Clear

Tesla Vehicle Protocol: UniversalMessage

320208033a121210934f10691deda
826a7982e92c4fce83f52069685b9
b0a2886a80010a430a41045310b89
94bb511e99e44d245be0537447a6d
c73c51d6409cf9f548c31360e0fc5
ad80cc30c06d98e1e0c478f5cd850
4e2d5573625527b75177e471561a3
697b22a390a104c463f9cc0d3d269
06e982ed224adde6120c9b4d39f6f
7e8a105d3feeda5180325640a0000
2a10179ab82400df1623a6fec4435
3b728dd9a0310d5f3d9e45bfa6cc3
51e220ae0ce10698





UniversalMessage

```
to_destination = 32020803 (DOMAIN_INFOTAINMENT)
```

```
from_destination = 3a121210934f10691deda826a7982e92c4fce83f
```

```
payload.protobuf_message_as_bytes = 52069685b9b0a288
```

```
SignatureData: KeyIdentity + =
```

```
6a80010a430a41045310b8994bb511e99e44d245be0537447a6dc73c51d6409cf9f548c31360  
e0fc5ad80cc30c06d98e1e0c478f5cd8504e2d5573625527b75177e471561a3697b22a390a10  
4c463f9cc0d3d26906e982ed224adde6120c9b4d39f6f7e8a105d3feeda5180325640a00002a  
10179ab82400df1623a6fec44353b728dd
```

```
request_uuid = 9a0310d5f3d9e45bfa6cc351e220ae0ce10698
```



ESP BLE: `gattc_event_handler`

`ESP_GATTC_OPEN_EVT` / `ESP_GATTC_CLOSE_EVT`

Connection opened / closed event

`ESP_GATTC_REG_FOR_NOTIFY_EVT`

Register for receiving notifications from vehicle

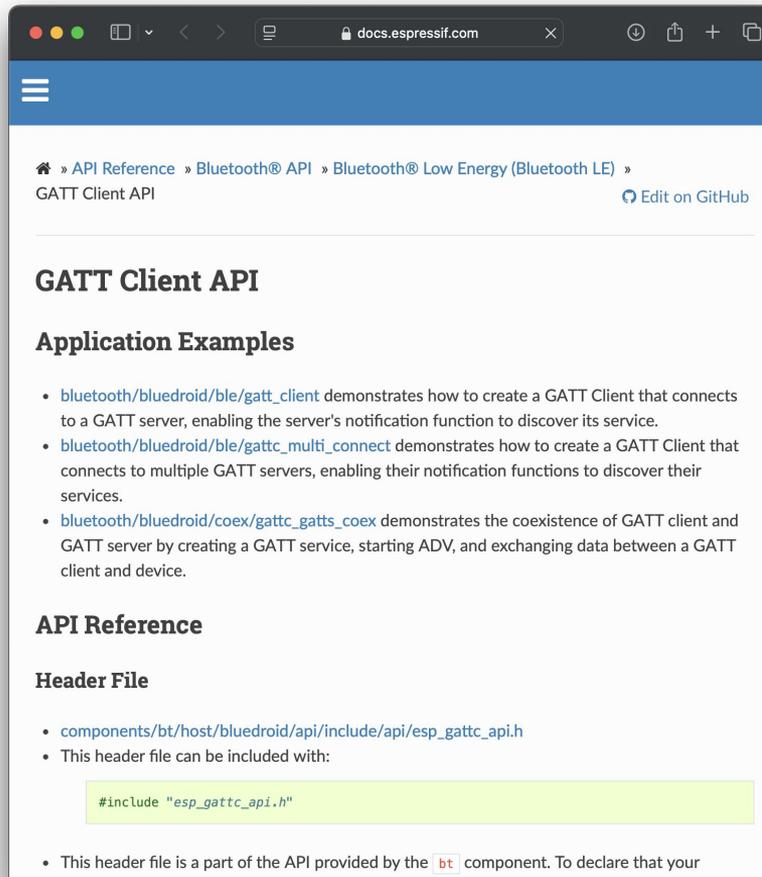
`ESP_GATTC_NOTIFY_EVT`

Handle incoming data, can come in chunks!

vehicle_command: chunks of 20 bytes

`esp_ble_gattc_write_char`

Send data



The screenshot shows a web browser window displaying the ESP8266 API Reference page for the GATT Client API. The page is titled "GATT Client API" and is part of the "Bluetooth® Low Energy (Bluetooth LE)" section. It includes a navigation breadcrumb: "API Reference » Bluetooth® API » Bluetooth® Low Energy (Bluetooth LE) » GATT Client API". There is a link to "Edit on GitHub". The page content is organized into sections: "GATT Client API", "Application Examples", and "API Reference". Under "Application Examples", there are three bullet points describing how to create a GATT Client that connects to a GATT server, how to connect to multiple GATT servers, and how to demonstrate the coexistence of GATT client and GATT server. Under "API Reference", there is a "Header File" section with a link to "components/bt/host/bluedroid/api/include/api/esp_gattc_api.h" and a note that this header file can be included with the code snippet:

```
#include "esp_gattc_api.h"
```

Queueing messages

```
void TeslaBLEVehicle::loop()
{
  if (this->node_state != espbt::ClientState::ESTABLISHED)
  {
    if (!command_queue_.empty())
    {
      // clear command queue if not connected or on first boot (prevent restore value triggering commands)
      command_queue_.pop();
    }
    return;
  }

  process_ble_read_queue();
  process_response_queue();
  process_command_queue();
  process_ble_write_queue();
}
```

Command state machine

Change charging amps → Charging = INFOTAINMENT domain

Can only talk to INFOTAINMENT when car is awake

1. First need to wake up car
 - a. If session info are outdated (epoch, key, counter) → Request new session info
 - b. Wait for new session info
 - c. Update session info key information
2. Wait for confirmation that car is awake
3. Send command
4. Wait for command confirmation

Built-in (up to 5) retries for robustness and comms failures

Want to learn more?

<https://github.com/yoziro/esphome-tesla-ble>

<https://github.com/yoziro/tesla-ble>