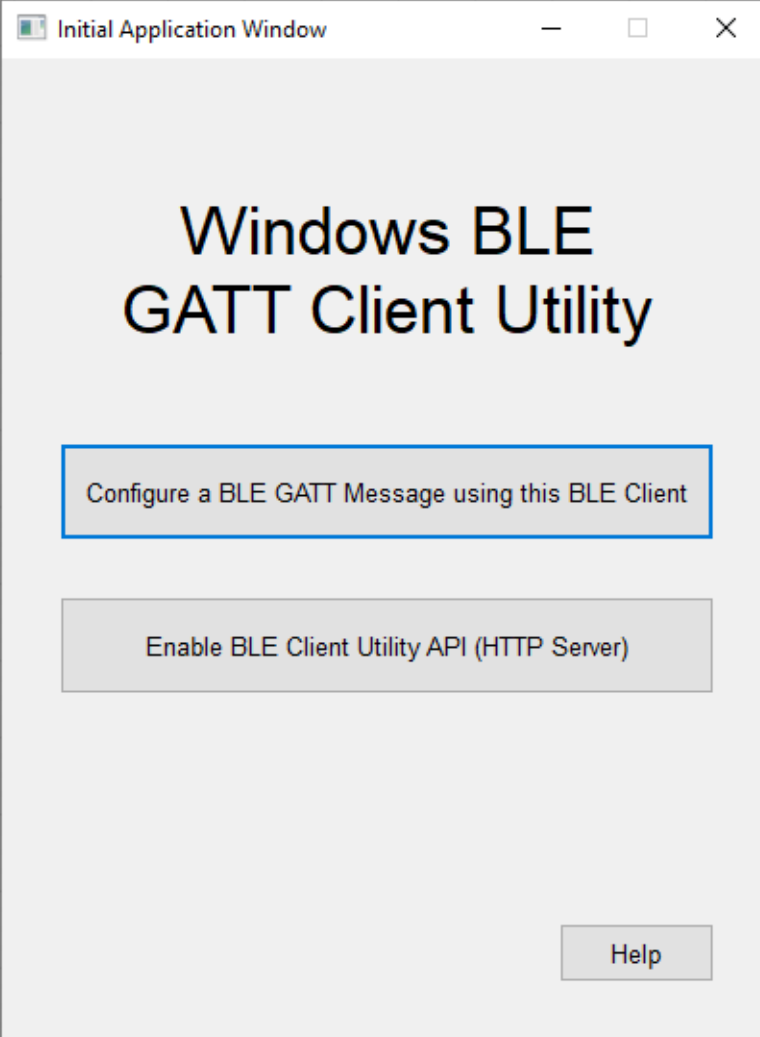## Initial App Screen

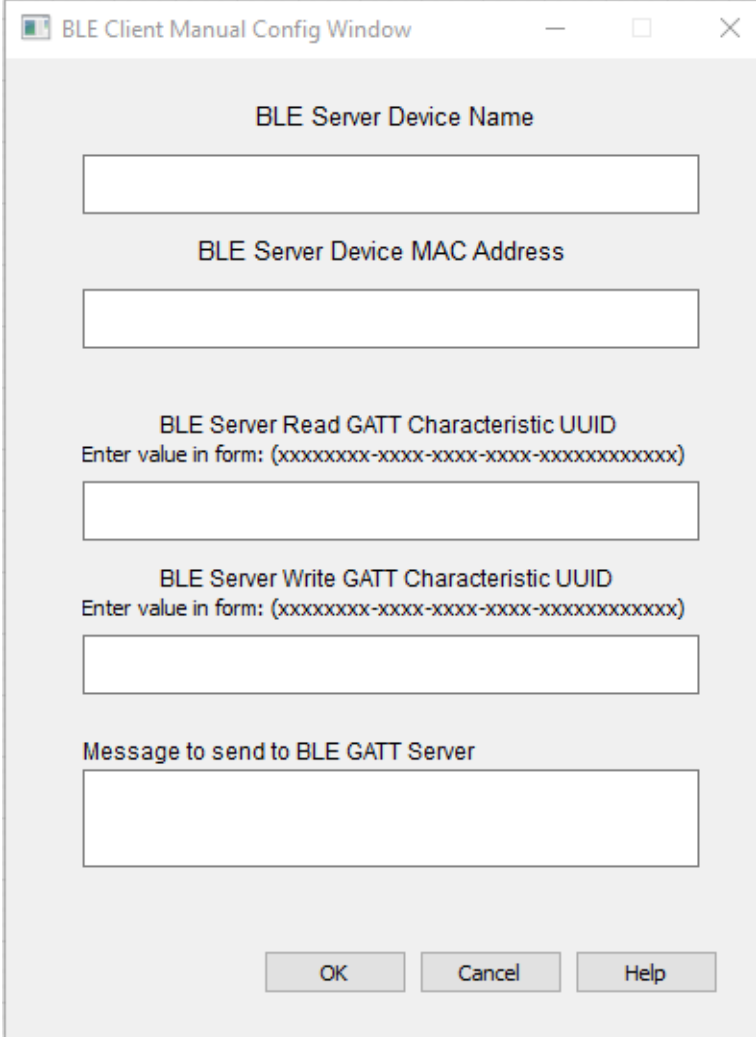**Initial Application Window** — □ ✕

Windows BLE
GATT Client Utility

Configure a BLE GATT Message using this BLE Client

Enable BLE Client Utility API (HTTP Server)

Help

Screen for Manually Configuring a BLE Message

BLE Client Manual Config Window — □ ✕

BLE Server Device Name

BLE Server Device MAC Address

BLE Server Read GATT Characteristic UUID
Enter value in form: (xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx)
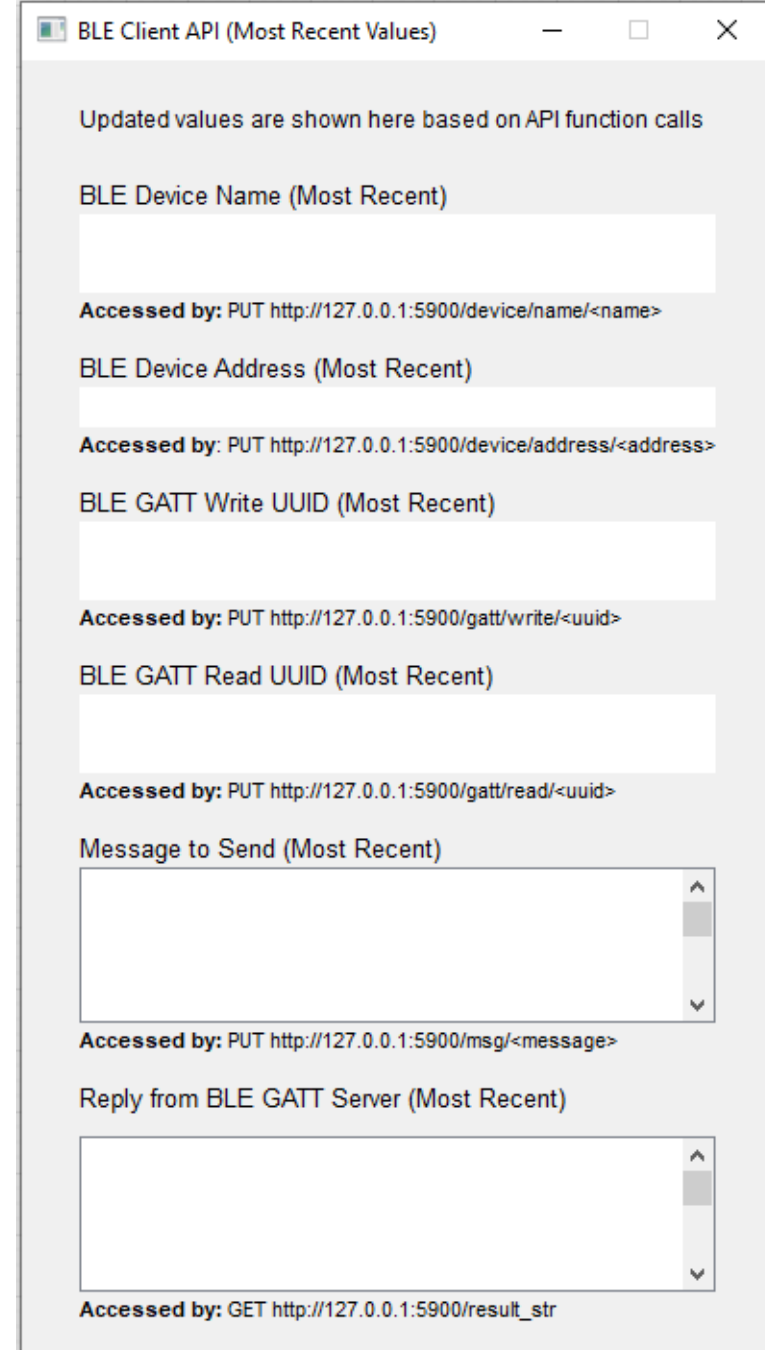
BLE Server Write GATT Characteristic UUID
Enter value in form: (xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx)

Message to send to BLE GATT Server

OK   Cancel   Help

Note: This app assumes that the BLE-capable server has a BLE GATT Characteristic which is dedicated to sending replies back (like an ACK signal or a reply string). That is the role of the BLE Server Write Characteristic UUID (because it will write a reply back to us).

Screen when the User Chooses to Use this app in API Mode

BLE Client API (Most Recent Values) — □ ✕

Updated values are shown here based on API function calls

BLE Device Name (Most Recent)

Accessed by: PUT http://127.0.0.1:5900/device/name/<name>

BLE Device Address (Most Recent)

Accessed by: PUT http://127.0.0.1:5900/device/address/<address>

BLE GATT Write UUID (Most Recent)

Accessed by: PUT http://127.0.0.1:5900/gatt/write/<uuid>

BLE GATT Read UUID (Most Recent)

Accessed by: PUT http://127.0.0.1:5900/gatt/read/<uuid>

Message to Send (Most Recent)

Accessed by: PUT http://127.0.0.1:5900/msg/<message>
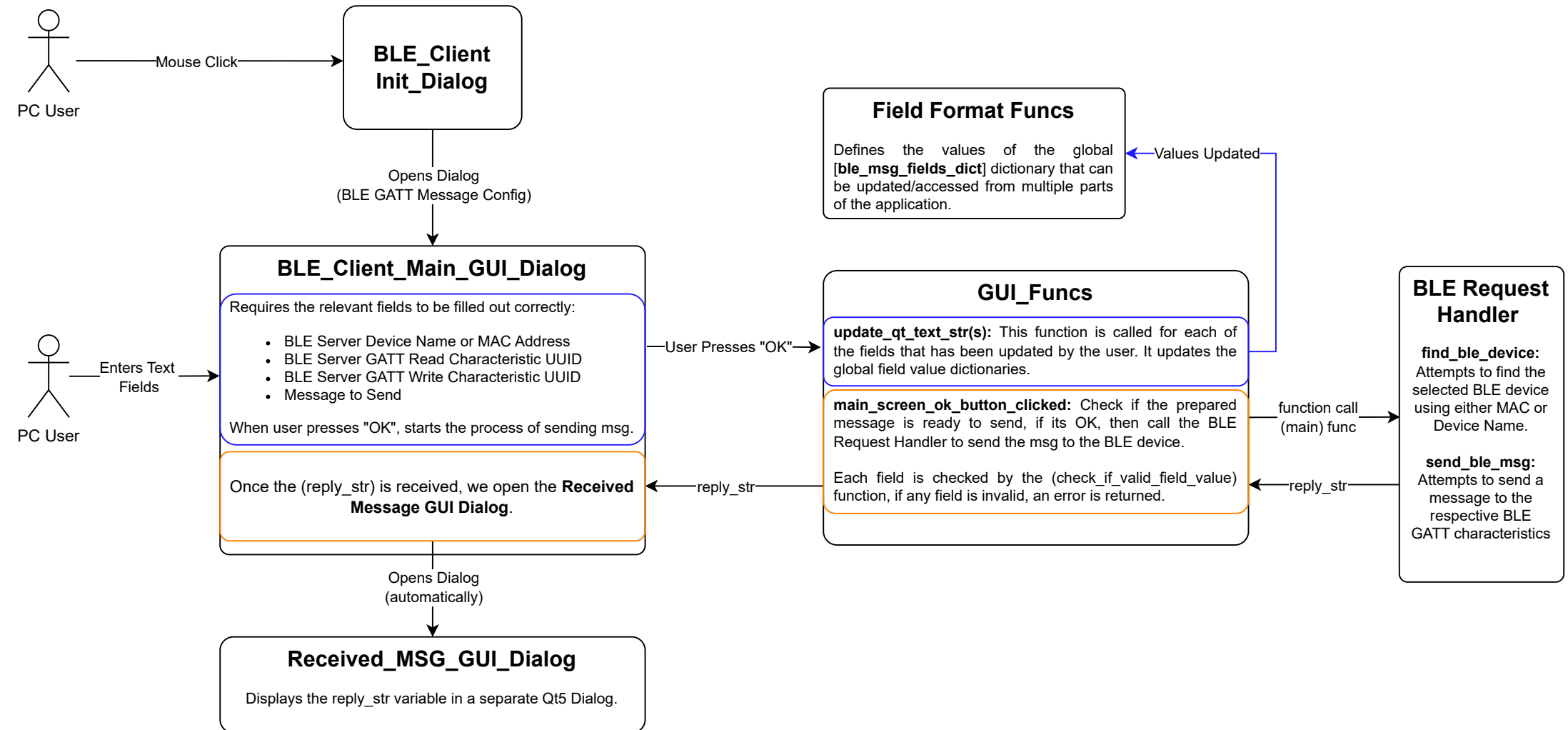
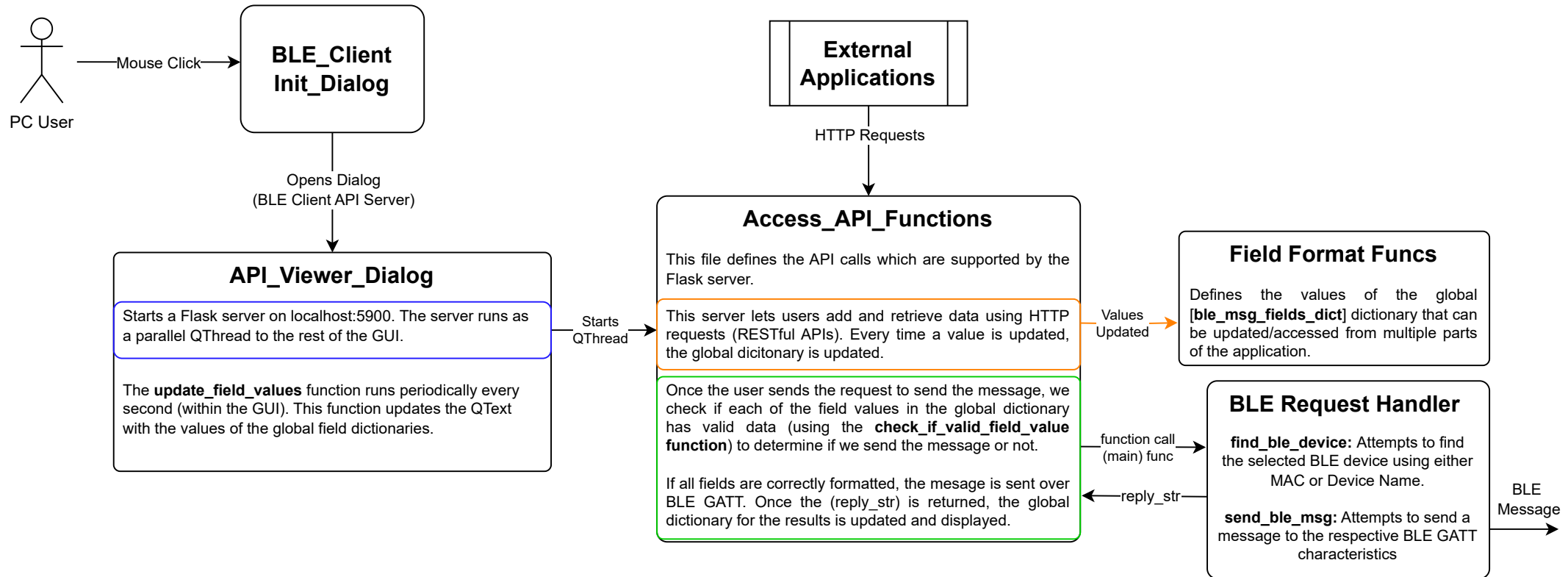Reply from BLE GATT Server (Most Recent)

Accessed by: GET http://127.0.0.1:5900/result_str

Note: This app assumes that the BLE-capable server has a BLE GATT Characteristic which is dedicated to sending replies back (like an ACK signal or a reply string). That is the role of the BLE Server Write Characteristic UUID (because it will write a reply back to us).

Manually Configuring a BLE GATT message

PC User

—Mouse Click→

BLE_Client Init_Dialog

Opens Dialog
(BLE GATT Message Config)

BLE_Client_Main_GUI_Dialog

Requires the relevant fields to be filled out correctly:

- BLE Server Device Name or MAC Address
- BLE Server GATT Read Characteristic UUID
- BLE Server GATT Write Characteristic UUID
- Message to Send

When user presses "OK", starts the process of sending msg.

Once the (reply_str) is received, we open the **Received Message GUI Dialog**.

PC User

—Enters Text Fields→

—User Presses "OK"—→

—reply_str—

Opens Dialog
(automatically)

Received_MSG_GUI_Dialog

Displays the reply_str variable in a separate Qt5 Dialog.

Field Format Funcs

Defines the values of the global [**ble_msg_fields_dict**] dictionary that can be updated/accessed from multiple parts of the application.

←Values Updated—

GUI_Funcs

update_qt_text_str(s): This function is called for each of the fields that has been updated by the user. It updates the global field value dictionaries.

main_screen_ok_button_clicked: Check if the prepared message is ready to send, if its OK, then call the BLE Request Handler to send the msg to the BLE device.

Each field is checked by the (check_if_valid_field_value) function, if any field is invalid, an error is returned.

function call
(main) func →

—reply_str—

BLE Request Handler

find_ble_device: Attempts to find the selected BLE device using either MAC or Device Name.

send_ble_msg: Attempts to send a message to the respective BLE GATT characteristics

Starting RESTful server to configure BLE GATT message through API calls

PC User

Mouse Click →

BLE_Client Init_Dialog

Opens Dialog
(BLE Client API Server)
↓

API_Viewer_Dialog

Starts a Flask server on localhost:5900. The server runs as a parallel QThread to the rest of the GUI.

The **update_field_values** function runs periodically every second (within the GUI). This function updates the QText with the values of the global field dictionaries.

Starts QThread →

External Applications

HTTP Requests
↓

Access_API_Functions

This file defines the API calls which are supported by the Flask server.

This server lets users add and retrieve data using HTTP requests (RESTful APIs). Every time a value is updated, the global dicitonary is updated.

Once the user sends the request to send the message, we check if each of the field values in the global dictionary has valid data (using the **check_if_valid_field_value function**) to determine if we send the message or not.

If all fields are correctly formatted, the mesage is sent over BLE GATT. Once the (reply_str) is returned, the global dictionary for the results is updated and displayed.

Values Updated →

Field Format Funcs

Defines the values of the global [**ble_msg_fields_dict**] dictionary that can be updated/accessed from multiple parts of the application.

function call (main) func →
← reply_str

BLE Request Handler

find_ble_device: Attempts to find the selected BLE device using either MAC or Device Name.

send_ble_msg: Attempts to send a message to the respective BLE GATT characteristics

BLE Message →

RESTful API Requests

The following HTTP requests show examples on how to configure the necessary parameters to communicate with a BLE Server using BLE GATT.

Setting up the GATT Read/Write Characteristic UUIDs to use for this message:

- http://127.0.0.1:5900/gatt/write/0000XXXX-0000-1000-8000-00805f9b34fb
- http://127.0.0.1:5900/gatt/read/0000XXXX-0000-1000-8000-00805f9b34fb

Setting up the BLE Device by using the Device Name or MAC Address:

- http://127.0.0.1:5900/device/name/XXXXXXXXXXXXXXX
- http://127.0.0.1:5900/device/address/aa:bb:cc:dd:ee:ff

To configure the actual message (string) to send to the BLE Server, use the following request:

- http://127.0.0.1:5900/msg/XXXXXXXXXXXXXXXXXXXXXXXXXX

Once the user has defined all the necessary fields to send a message over BLE, we can send the message using the following request:

- http://127.0.0.1:5900/send_msg

String Encoding:

This application checks the formatting of each of the following fields (and returns an error message instead of updating the values if certain conditions aren't met):

- GATT Read and Write Characteristic UUIDs
- Device Name
- MAC Address

When updating the actual message field, ensure that you take into consideration factors like URL encoded versions of strings, as this app will not automatically convert strings into URL-compatible strings.