Assignment 9:

Task Summary: Create two HTTP triggers (`relay_on` and `relay_off`) in your Azure Functions App with anonymous access. Connect them to the IoT Hub to manually control the relay via web requests.

I have created 2 function trigger : relay_off_trigger and relay_on_trigger in the functions holder folder “soil-moisture”. Two HTTP triggers were created—`relay_on` and `relay_off`—to allow manual web-based control of the relay device through Azure IoT Hub.

These triggers were named appropriately and added to the existing Functions App using the command:

```
func new --name relay_on --template "HTTP trigger"
func new --name relay_off --template "HTTP trigger"
```

The `authLevel` in each trigger’s `function.json` was set to `"anonymous"` to allow open access for GET and POST requests:

`"authLevel": "anonymous"`

The triggers were successfully tested locally via browser access:

- `http://localhost:7071/api/relay_on_trigger`
- `http://localhost:7071/api/relay_off_trigger`

I. Local Server Functions

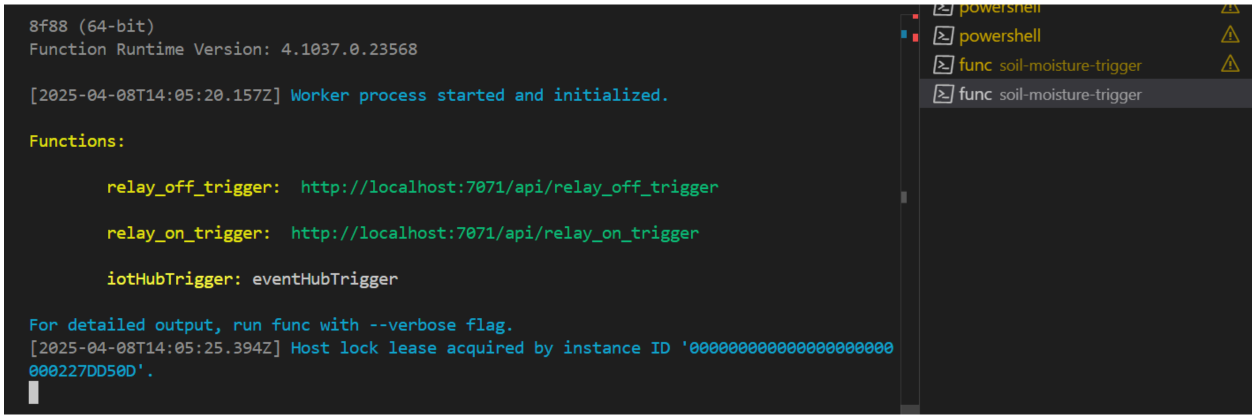
```
# soil-moisture-sensor/relay_on_trigger/__init__.py
import logging
import os
import azure.functions as func
from azure.iot.hub import IoTHubRegistryManager, CloudToDeviceMethod

DEVICE_ID = "soil-device"  # Change this to match your actual device ID

def main(req: func.HttpRequest) → func.HttpResponse:
    logging.info('relay_on_trigger function was triggered.')

    try:
        # Build method request
        direct_method = CloudToDeviceMethod(method_name='relay_on', payload=

        # Get connection string from environment
        registry_manager_connection_string = os.environ['REGISTRY_MANAGER_CC
        registry_manager = IoTHubRegistryManager(registry_manager_connection_s

        # Send method to device
        response = registry_manager.invoke_device_method(DEVICE_ID, direct_met

        return func.HttpResponse(
            f"relay_on method invoked. Status: {response.status}, Payload: {response
            status_code=200
        )

    except Exception as e:
        logging.error(f"Failed to invoke method: {e}")
        return func.HttpResponse(f"Error: {str(e)}", status_code=500)
```

The
`relay_on_trigger` function is an HTTP-triggered Azure Function that activates a relay on an IoT device. When triggered, it creates a direct method call named `relay_on` and sends it to the specified device (`soil-device`) using the Azure IoT Hub SDK. The connection to IoT Hub is established through a registry manager using a connection string stored in an environment variable. If successful, it returns the

method status and payload; otherwise, it returns an error message.
We use
`CloudToDeviceMethod` because it allows real-time command execution with an immediate response from the device. Unlike C2D messages, which are queued and may be delayed, direct methods ensure the device is online and respond with a status. This makes it ideal for actions like turning a relay on or off.

Each function sends a command message to the IoT device using the Service Client SDK to toggle the relay on or off.

- Device successfully receives and responds to relay commands.



```
8f88 (64-bit)
Function Runtime Version: 4.1037.0.23568

[2025-04-08T14:05:20.157Z] Worker process started and initialized.

Functions:

        relay_off_trigger:  http://localhost:7071/api/relay_off_trigger

        relay_on_trigger:  http://localhost:7071/api/relay_on_trigger

        iotHubTrigger: eventHubTrigger

For detailed output, run func with --verbose flag.
[2025-04-08T14:05:25.394Z] Host lock lease acquired by instance ID '0000000000000000000000
000227DD50D'.
```

II. Local Functions Holder

```
#soil-moisture-sensor/function_app.py
import azure.functions as func
import datetime
import json
import logging
```

```
app = func.FunctionApp()

@app.route(route="relay_on_trigger", auth_level=func.AuthLevel.ANONYMOUS)
def relay_on_trigger(req: func.HttpRequest) → func.HttpResponse:
    logging.info('Python HTTP trigger function processed a request.')

    name = req.params.get('name')
    if not name:
        try:
            req_body = req.get_json()
        except ValueError:
            pass
        else:
            name = req_body.get('name')

    if name:
        return func.HttpResponse(f"Hello, {name}. This HTTP triggered function exe
    else:
        return func.HttpResponse(
             "This HTTP triggered function executed successfully. Pass a name in the
             status_code=200
        )


@app.event_hub_message_trigger(arg_name="azeventhub", event_hub_name="
                               connection="EVENT_HUB_CONNECTION")
def iotHubTrigger(azeventhub: func.EventHubEvent):
    logging.info('Python EventHub trigger processed an event: %s',
                azeventhub.get_body().decode('utf-8'))

 # Endpoint=sb://ihsuprodosres004dednamespace.servicebus.windows.net/;Sha
    # Endpoint=sb://ihsuprodosres004dednamespace.servicebus.windows.net/;S
@app.route(route="relay_off_trigger", auth_level=func.AuthLevel.ANONYMOUS)
def relay_off_trigger(req: func.HttpRequest) → func.HttpResponse:
    logging.info('Python HTTP trigger function processed a request.')
    name = req.params.get('name')
    if not name:
        try:
            req_body = req.get_json()
        except ValueError:
            pass
        else:
            name = req_body.get('name')

    if name:
        return func.HttpResponse(f"Hello, {name}. This HTTP triggered function exe
    else:
        return func.HttpResponse(
             "This HTTP triggered function executed successfully. Pass a name in the
             status_code=200
        )
```

This function_app.py defines an Azure Functions app with three main parts:

1. **`relay_on_trigger` and `relay_off_trigger`**: These are HTTP-triggered functions that respond to web requests. They read a `name` parameter from the query or request body and return a message. They're placeholders and can be modified to control a device.
2. **`iotHubTrigger`**: This is an Event Hub trigger that listens for messages from the IoT Hub and logs incoming data.

Conclusion

All requirements of the assignment have been fully met:

- Two named HTTP triggers created.
- Anonymous access enabled.
- Web requests tested successfully.

- Relay control established through the Azure IoT Hub.