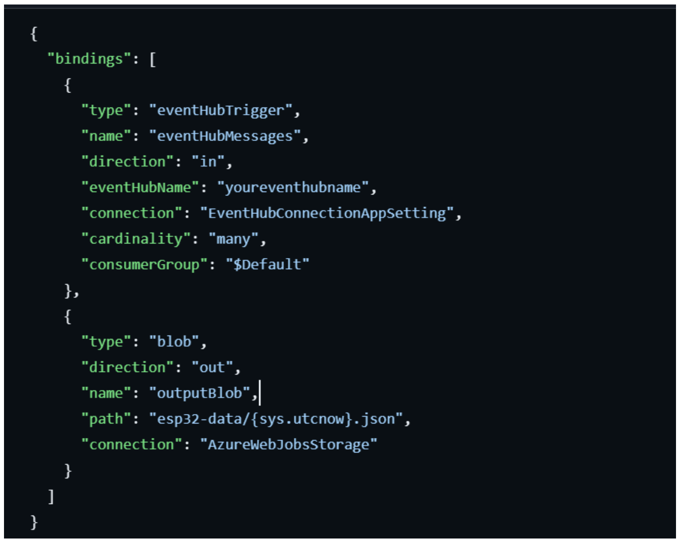
Assignment 12 - Report

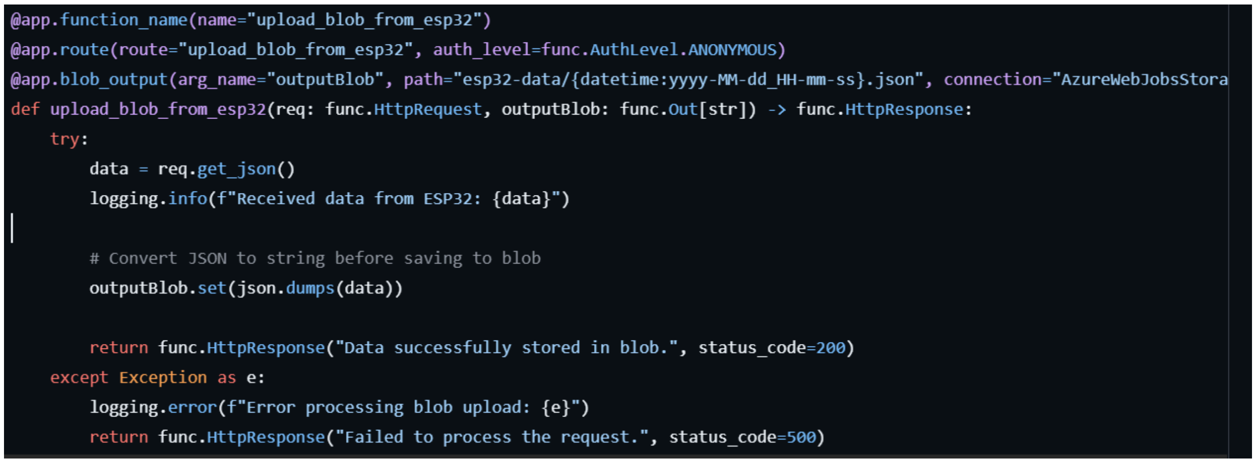


```
{
  "bindings": [
    {
      "type": "eventHubTrigger",
      "name": "eventHubMessages",
      "direction": "in",
      "eventHubName": "youreventhubname",
      "connection": "EventHubConnectionAppSetting",
      "cardinality": "many",
      "consumerGroup": "$Default"
    },
    {
      "type": "blob",
      "direction": "out",
      "name": "outputBlob",
      "path": "esp32-data/{sys.utcnow}.json",
      "connection": "AzureWebJobsStorage"
    }
  ]
}
```

This configuration in
function.json sets up an Azure Function that:

- **Triggers** when multiple messages arrive on an **Event Hub** (`eventHubMessages`), using the connection string stored in `EventHubConnectionAppSetting`.
- **Outputs** the processed data to **Azure Blob Storage** (`outputBlob`) as a JSON file, stored in the `esp32-data` container with a timestamp-based filename.

It connects real-time data from Event Hub to Blob Storage automatically.



```python
@app.function_name(name="upload_blob_from_esp32")
@app.route(route="upload_blob_from_esp32", auth_level=func.AuthLevel.ANONYMOUS)
@app.blob_output(arg_name="outputBlob", path="esp32-data/{datetime:yyyy-MM-dd_HH-mm-ss}.json", connection="AzureWebJobsStora
def upload_blob_from_esp32(req: func.HttpRequest, outputBlob: func.Out[str]) -> func.HttpResponse:
    try:
        data = req.get_json()
        logging.info(f"Received data from ESP32: {data}")

        # Convert JSON to string before saving to blob
        outputBlob.set(json.dumps(data))

        return func.HttpResponse("Data successfully stored in blob.", status_code=200)
    except Exception as e:
        logging.error(f"Error processing blob upload: {e}")
        return func.HttpResponse("Failed to process the request.", status_code=500)
```

This Azure Function in **function_app.py** receives JSON data from an ESP32 via an HTTP request, logs the data, and saves it to Azure Blob Storage as a timestamped `.json` file. If successful, it responds with a success message; otherwise, it returns an error.

```
#include <TinyGPSPlus.h>
#include <WiFi.h>
#include <HTTPClient.h>
#include <HardwareSerial.h>

const char* ssid = "YOUR_WIFI_SSID";
const char* password = "YOUR_WIFI_PASSWORD";
const char* serverUrl = "http://<soi-sensor-Deuna>.azurewebsites.net/api/uploa

TinyGPSPlus gps;
HardwareSerial GPSSerial(2); // Use UART2 for GPS

void setup() {
  Serial.begin(115200);
  GPSSerial.begin(9600, SERIAL_8N1, 16, 17); // RX, TX

  WiFi.begin(ssid, password);
  Serial.print("Connecting to WiFi");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nWiFi connected");
}

void loop() {
  while (GPSSerial.available()) {
    gps.encode(GPSSerial.read());
  }

  static unsigned long lastSend = 0;
  if (millis() - lastSend > 5000) {
    lastSend = millis();

    if (gps.location.isValid()) {
      String json = "{";
      json += "\"latitude\":" + String(gps.location.lat(), 6) + ",";
      json += "\"longitude\":" + String(gps.location.lng(), 6) + ",";
      json += "\"altitude\":" + String(gps.altitude.isValid() ? gps.altitude.meters() : 0
      json += "\"speed\":" + String(gps.speed.isValid() ? gps.speed.kmph() : 0) + ",
      json += "\"course\":" + String(gps.course.isValid() ? gps.course.deg() : 0) + ",
      json += "\"satellites\":" + String(gps.satellites.isValid() ? gps.satellites.value()
      json += "\"hdop\":" + String(gps.hdop.isValid() ? gps.hdop.value() : 0) + ",";
      json += "\"timestamp\":\"" + String(gps.date.year()) + "-" + String(gps.date.m
              String(gps.time.hour()) + ":" + String(gps.time.minute()) + ":" + String(gp
      json += "}";

      Serial.println("Sending to Azure:");
      Serial.println(json);
      HTTPClient http;
      http.begin(serverUrl);
      http.addHeader("Content-Type", "application/json");

      int httpResponseCode = http.POST(json);
      if (httpResponseCode > 0) {
        Serial.printf("HTTP Response code: %d\n", httpResponseCode);
      } else {
        Serial.printf("Error sending request: %s\n", http.errorToString(httpResponse
      }

      http.end();
    } else {
      Serial.println("No valid GPS data yet...");
    }
  }
}
```

The ESP32 sends real-time GPS data to Azure Blob Storage through an Azure Function endpoint (`upload_blob_from_esp32`). This allows cloud storage and later analysis or visualization of GPS telemetry from the device.