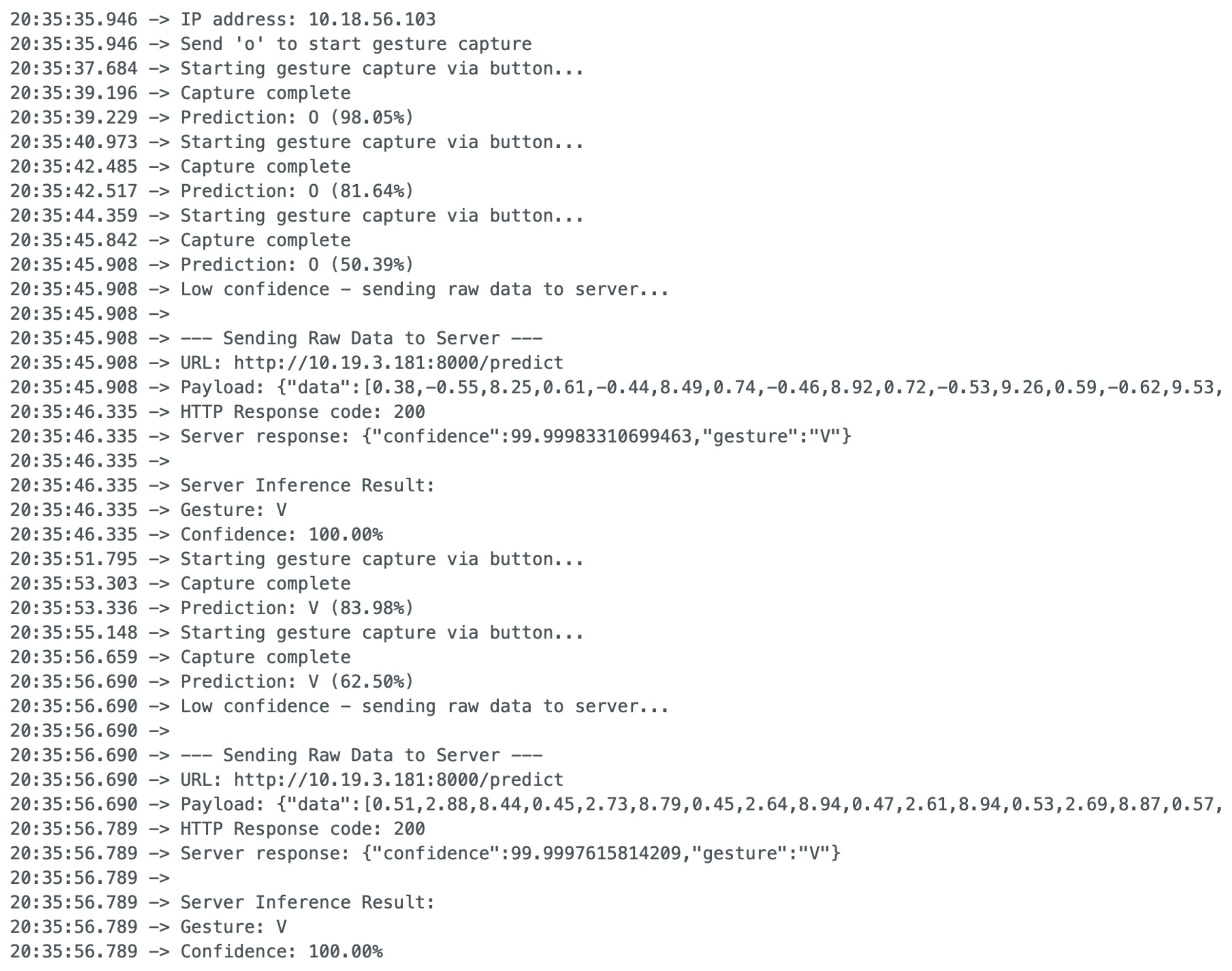
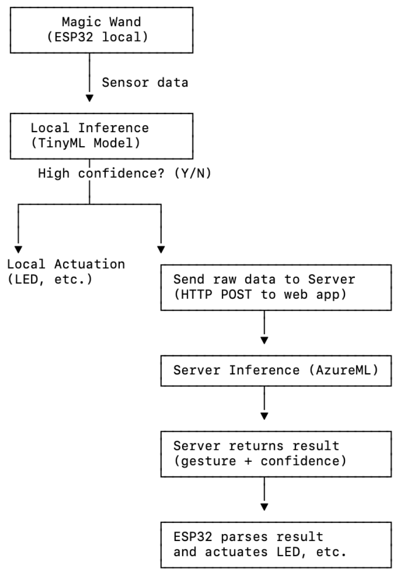


```
20:35:35.946 -> IP address: 10.18.56.103
20:35:35.946 -> Send 'o' to start gesture capture
20:35:37.684 -> Starting gesture capture via button...
20:35:39.196 -> Capture complete
20:35:39.229 -> Prediction: O (98.05%)
20:35:40.973 -> Starting gesture capture via button...
20:35:42.485 -> Capture complete
20:35:42.517 -> Prediction: O (81.64%)
20:35:44.359 -> Starting gesture capture via button...
20:35:45.842 -> Capture complete
20:35:45.908 -> Prediction: O (50.39%)
20:35:45.908 -> Low confidence - sending raw data to server...
20:35:45.908 ->
20:35:45.908 -> --- Sending Raw Data to Server ---
20:35:45.908 -> URL: http://10.19.3.181:8000/predict
20:35:45.908 -> Payload: {"data":[0.38,-0.55,8.25,0.61,-0.44,8.49,0.74,-0.46,8.92,0.72,-0.53,9.26,0.59,-0.62,9.53,
20:35:46.335 -> HTTP Response code: 200
20:35:46.335 -> Server response: {"confidence":99.99983310699463,"gesture":"V"}
20:35:46.335 ->
20:35:46.335 -> Server Inference Result:
20:35:46.335 -> Gesture: V
20:35:46.335 -> Confidence: 100.00%
20:35:51.795 -> Starting gesture capture via button...
20:35:53.303 -> Capture complete
20:35:53.336 -> Prediction: V (83.98%)
20:35:55.148 -> Starting gesture capture via button...
20:35:56.659 -> Capture complete
20:35:56.690 -> Prediction: V (62.50%)
20:35:56.690 -> Low confidence - sending raw data to server...
20:35:56.690 ->
20:35:56.690 -> --- Sending Raw Data to Server ---
20:35:56.690 -> URL: http://10.19.3.181:8000/predict
20:35:56.690 -> Payload: {"data":[0.51,2.88,8.44,0.45,2.73,8.79,0.45,2.64,8.94,0.47,2.61,8.94,0.53,2.69,8.87,0.57,
20:35:56.789 -> HTTP Response code: 200
20:35:56.789 -> Server response: {"confidence":99.9997615814209,"gesture":"V"}
20:35:56.789 ->
20:35:56.789 -> Server Inference Result:
20:35:56.789 -> Gesture: V
20:35:56.789 -> Confidence: 100.00%
```

1.
Yes, the server's confidence was consistently higher than the wand's confidence during my tests. A hypothetical reason for this observation is that the server is using a more powerful model hosted on Azure with better data availability and computational resources. In contrast, the ESP32 device uses a lightweight, compressed version of the model, which naturally leads to lower confidence due to the limitations in memory and processing power. The cloud model might also have access to more comprehensive training data, further improving its accuracy and reliability.

2.


Magic Wand (ESP32 local)
Sensor data
Local Inference (TinyML Model)
High confidence? (Y/N)
Local Actuation (LED, etc.)
Send raw data to Server (HTTP POST to web app)
Server Inference (AzureML)
Server returns result (gesture + confidence)
ESP32 parses result and actuates LED, etc.

3.
Reliance on Connectivity
- Pros: Only relies on internet connectivity when local inference confidence is low, minimizing overall data transmission.
- Cons: If connectivity is poor, the fallback to server might fail or introduce delays.

Latency
- Pros: Local inference provides fast response for high-confidence predictions, reducing perceived latency.
- Cons: For low-confidence cases, sending data to the cloud introduces additional latency.

Prediction Consistency
- Pros: The server’s model can be more consistent due to more powerful models and larger datasets.
- Cons: The switch between local and cloud models may lead to slightly different prediction outputs, introducing inconsistency.

Data Privacy
- Pros: Local inference preserves privacy for most inferences since data never leaves the device when confidence is high.
- Cons: In low-confidence scenarios, sensitive sensor data is transmitted to the cloud, which could pose privacy risks.

4.
A good strategy to mitigate latency and connectivity reliance is to implement a caching mechanism or a hybrid on-device prefetching approach. For instance, the ESP32 can store or locally update frequently used inference results or partial models for common gestures, reducing the need to constantly send data to the cloud and enhancing system resilience even with intermittent connectivity.