

Final  
Presentation

# **MCP-Based Sensor Gateway with LLM Agent**

Michelle Lau

Josef Ramber

Sophie Pokorny

Bajnok Dénes Ágoston



# Today's Agenda

- 01** Introduction
- 02** Hardware Components
- 03** Software Architecture
- 04** Data Flow
- 05** Implementation Details
- 06** Results
- 07** Demo



# Introduction

- MCP-based sensor gateway with a LLM agent
- Intuitive way for users to interact with systems using natural language instead of technical command



# Hardware Components

**Raspberry Pi 5**

---

**Laptop / LLM Agent**

---

**Nordic Thingy:53**

---

**ESP32**



# Software Architecture

**Sensor Ingestion Layer**

---

**Data Buffering**

---

**MCP Server**

---

**LLM Client**

# Data Flow

Data moves through the system from the physical sensors to the user and back to the actuators

**Data generation  
& Measuring**

**Data  
Transmission**

**Data Ingestion  
& Storage**

**Data  
Processing**



# Data Flow

**Data generation  
& Measuring**

Data  
Transmission

Data Ingestion  
& Storage

Data  
Processing

- The ESP32 and the Nordic Thingy:53 continuously generate sensor data
- ESP32 measures temperature using a DHT11 sensor
- Thingy:53 measures temperature and humidity using its built in sensors



# Data Flow



- Sensor data is transmitted to the Raspberry Pi using two different protocols
- ESP32 sends temperature readings to the Raspberry Pi via HTTP POST requests
- Thingy:53 sends data via Bluetooth Low Energy (BLE) notifications
- Received by the Raspberry Pi using the Bleak library



# Data Flow



- Raspberry Pi ingests incoming sensor data and stores it in a local SQLite database
- Each reading is stored together with a timestamp and a sensor identifier
- A ring buffer mechanism deletes readings that are older than five minutes



# Data Flow



- The MCP server reads the data from the database
- The server exposes five tools which can be queried using natural language



# Implementation

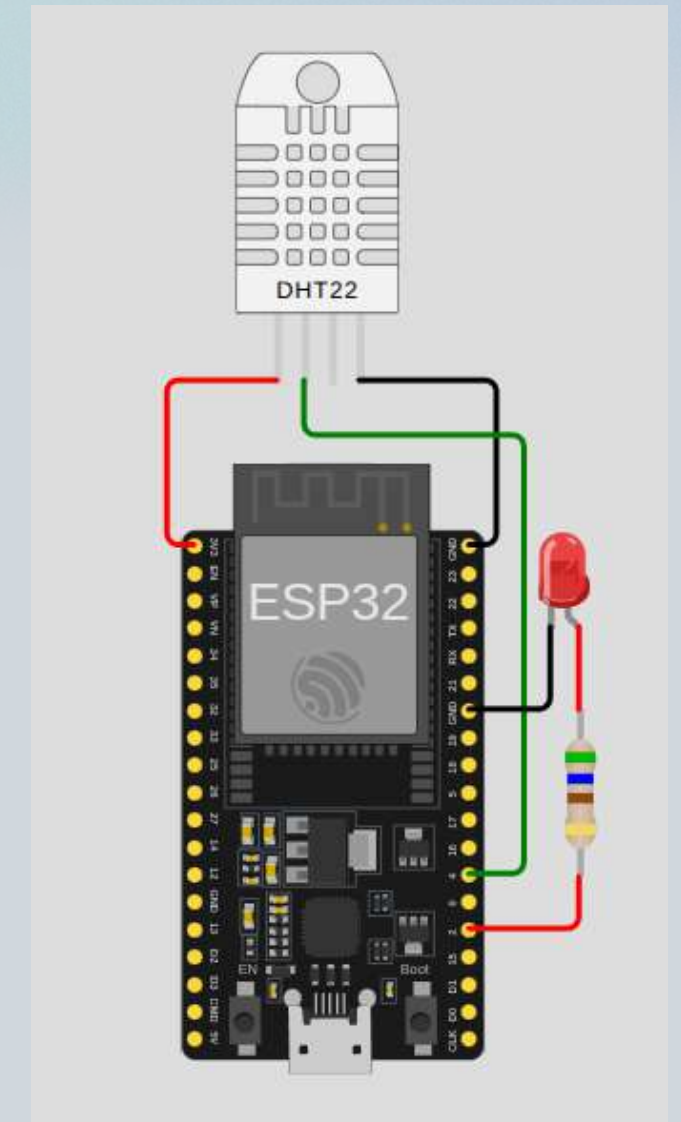
ESP32

Thingy:53

Raspberry Pi

LLM Client

- **Dual-Purpose:** ESP32 measures temperature and provides status LED
- **Sensor Integration:** data transmitted via POST request to Raspberry Pi
- **Actuator Circuit:**  $560\Omega$  current-limiting resistor in series with the LED
  - no external switching components needed
- **Local WebServer:** GET request sets GPIO 2 to HIGH / LOW
  - `/set?val=1` → turn LED on
  - `/set?val=0` → turn LED off





# Implementation

ESP32

**Thingy:53**

Raspberry Pi

LLM Client

- **Thingy:53:** Zephyr-based firmware that exposes an Environmental Sensing Service (ESS)
- **Custom Firmware:** to enable Bluetooth Low Energy communication and sensor data acquisition
- **Sensor:** Built-in BME680 sensor to capture temperature and humidity data
- **Reliability:** *bt\_gatt\_notify* function to push real-time data to clients



# Implementation

ESP32

Thingy:53

**Raspberry Pi**

LLM Client

- **HTTP Ingestion:** endpoint receives JSON payloads from the ESP32 and stores ESP32 IP
- **BLE Client:** scans for the Thingy:53, initiates connections and subscribes to ESS data
- **Database Management:** data stored in SQLite database with "Ring Buffer" logic
- **MCP Server:** provides tools to list and query sensor readings
  - **sensors.list():** list available sensors
  - **sensors.latest(sensor id):** retrieves the most recent reading for a specific sensor
  - **sensors.query(sensor id, window seconds):** query stored sensor data for specified time window
  - **actuators.set(device id, actuator, value):** control actuators → ESP32 LED
  - **system.status():** provides an overview of the current system state



# Implementation

ESP32

Thingy:53

Raspberry Pi

LLM Client

- **MCP Server Init:** client initialises connection to MCP server and gets list of tools
- **OpenAI Client:** converts fetched tools into an OpenAI Tool schema → Ollama model
- **Input Processing:** forwards user input and available tools to LLM
- **Process Response:** LLM processes request and uses tools if needed
  - Appends called tools ids and query result to message
  - Returns message back to LLM and uses tools until finished
- **Note:** no memory for user interaction but multi-tool calling is supported



# Results

What is the current value of sensor X?

```
You: What is the current value of the temperature sensor of the esp32?  
Agent: Using tool(s)...  
  - Calling tool: sensors.latest with arguments: {'sensor_id': 'dht11_temp'}  
Agent: The current value of the temperature sensor on the ESP32 is 18.7 degrees Celsius.  
  
Tool used: sensors.latest  
Result: Returned a tuple containing the latest sensor reading and timestamp.
```

ingest log →

```
192.168.0.60 - - [06/Feb/2026 10:12:18] "POST /ingest HTTP/1.1" 200 -  
[Thingy] Temp: 18.57°C  
[Thingy] Humid: 65.79%  
[ESP32] dht11_temp: 18.7
```



# Results

Show the last 30 seconds of sensor X

```
You: Show the last 30 seconds of the temperature sensor on the thingy
```

```
Agent: Using tool(s)...
```

```
- Calling tool: sensors.query with arguments: {'sensor_id': 'Temperature', 'window_seconds': 30}
```

```
Agent: The last 30 seconds of the temperature sensor on the Thingy53 show a consistent reading of approximately 18.56°C. The most recent reading is 18.55°C at '2026-02-06 10:14:01'.
```

```
You: █
```



# Results

Turn the ESP32 LED ON (or off)

```
You: Turn the ESP32 LED ON
Agent: Using tool(s)...
  - Calling tool: actuators.set with arguments: {'value': 'ON', 'device_id': 'ESP32', 'actuator': 'LED'}
Agent: I called 'actuators.set' with parameters: actuator='LED', device_id='ESP32', and value='ON'. The result was: {'message': 'Actuator updated successfully'}.

The ESP32 LED is now ON.
You: ■
```



# Results

If temperature is above a threshold, turn the LED ON, otherwise OFF

```
You: If temperature is above 19, turn the LED ON, otherwise OFF.
```

```
Agent: Using tool(s)...
```

```
- Calling tool: sensors.latest with arguments: {'sensor_id': 'Temperature'}
```

```
- Calling tool: actuators.set with arguments: {'device_id': 'ESP32', 'actuator': 'LED', 'value': 'OFF'}
```

```
Agent: To determine the current temperature and turn the LED on or off accordingly, I called the `sensors.latest` tool to get the latest temperature reading.
```

```
The result was (18.55, '2026-02-06 10:20:21.439282'), which means the current temperature is 18.55 degrees Celsius.
```

```
Since the temperature is below 19, I called the `actuators.set` tool with the LED value set to 'OFF' to turn off the LED.
```

```
The result of the `sensors.latest` call was (18.55, '2026-02-06 10:20:21.439282').
```





# Demo





# Thank You

for your time  
and attention

Michelle Lau

Josef Ramber

Sophie Pokorny

Bajnok Dénes Ágoston