IESTI05 - Edge AI Engineering

Part 2 Final Project: GenAI at the Edge

Project Overview

This project serves as the culmination of Part 2 of the course, where you will apply all concepts learned to create a practical GenAI system at the edge. Working in groups of 3 students, you will design, implement, and deploy a complete Small Language Model (SLM) solution on the Raspberry Pi 5, integrated with sensors and actuators to create an intelligent IoT system.

Building on the foundation from Part 1, this project introduces the exciting frontier of generative AI at the edge. You will transform a Raspberry Pi 5 into an AI hub capable of natural language understanding, environmental monitoring, and intelligent decision-making—all running locally without cloud dependencies.

This project represents a significant advancement in your Edge AI journey, combining language models, physical computing, and IoT integration. Success requires innovative thinking, careful system design, and effective integration of multiple technologies.

> Remember: The goal is not perfection, but demonstrating understanding of GenAI principles at the edge and creating a functional, innovative prototype!

Project Requirements

Core Technical Requirements

1. **Hardware Platform**: Raspberry Pi 5 with appropriate cooling
2. **AI Model**: Must use a Small Language Model (SLM) with ≤5B parameters, quantized to 4-bit
3. **Physical Computing**: Minimum of 2 sensors and two actuators. You could choose the available kit sensors/LEDs or request alternative and/or additional ones with Prof. José Alberto or José Anderson at IESTI (If available).
4. **Real-time Operation**: System must respond to inputs and generate outputs in real-time
5. **Local Processing**: All AI inference must run on-device (no cloud dependencies)

Software Requirements

- Python-based implementation
- Ollama for SLM deployment OR alternative frameworks (llama.cpp, Transformers)
- Appropriate sensor/actuator libraries (GPIOZero, Adafruit, etc.)
- Function calling and/or RAG implementation (at least one optimization technique)
- Clean, well-commented, modular code
- Proper error handling and logging

Project Categories (Suggestions only)

Category A: Smart Environment Monitoring & Control

Intelligent Climate Control

- Monitor temperature, humidity, and air quality
- Use SLM to analyze conditions and make control decisions
- Actuate fans, heaters, or ventilation based on AI analysis (LEDs can simulate atuation)
- Provide natural language status reports and recommendations
- *Extension*: Implement predictive control based on historical patterns (optional)

Smart Garden Assistant

- Monitor soil moisture, light levels, and temperature
- SLM provides care recommendations based on plant needs
- Control watering system and grow lights automatically
- Answer natural language queries: "Should I water the plants today?"
- *Extension*: Multi-plant monitoring with species-specific care

Indoor Air Quality Manager

- Integrate multiple environmental sensors (CO2, VOC, temperature, humidity)
- SLM analyzes air quality and provides health recommendations
- Control ventilation and air purifiers
- Generate daily air quality reports in natural language
- *Extension*: Implement RAG with air quality standards database

Category B: Interactive Assistive Systems

Voice-Controlled Home Assistant

- Integrate microphone for voice input
- Use SLM for natural language command understanding
- Control multiple home devices (lights, appliances, etc.)
- Provide spoken responses about system status
- *Extension*: Multi-room control with context awareness

Smart Accessibility Aid

- Monitor user environment and activity
- Provide proactive assistance based on sensor data
- Respond to natural language queries about surroundings
- Control assistive devices based on detected needs

- *Extension*: Implement emergency detection and alert system

Educational Lab Assistant

- Monitor laboratory conditions (temperature, light, safety sensors)
- Answer questions about current conditions and procedures
- Provide step-by-step guidance for experiments
- Alert users to unsafe conditions
- *Extension*: Log data and generate experiment reports

Category C: Industrial & Safety Applications

Smart Safety Monitor

- Integrate multiple safety sensors (smoke, gas, motion, etc.)
- SLM analyzes multi-sensor data for anomaly detection
- Control warning lights and alarms based on threat level
- Generate safety incident reports in natural language
- *Extension*: Implement predictive maintenance alerts

Quality Control Station

- Monitor production parameters with multiple sensors
- Use SLM to analyze if conditions meet specifications
- Control indicators for pass/fail status
- Generate detailed quality reports
- *Extension*: Implement RAG with quality standards documentation

Equipment Condition Monitor

- Track temperature, vibration, sound levels of equipment
- SLM analyzes sensor trends to predict maintenance needs
- Control status indicators (LEDs, displays)
- Provide natural language equipment status updates
- *Extension*: Implement data logging with trend visualization

Category D: Agriculture & Environmental

Smart Greenhouse Controller

- Monitor multiple growing conditions
- SLM optimizes growing parameters for specific crops
- Control irrigation, lighting, and ventilation
- Answer queries: "Is this good weather for tomatoes?"

- *Extension*: Implement multi-crop management with different zones

Weather Station with AI Analysis

- Collect atmospheric data (temperature, pressure, humidity, wind)
- Use SLM to generate weather forecasts and insights
- Control outdoor lighting/systems based on conditions
- Provide natural language weather reports
- *Extension*: Implement RAG with local weather patterns database

Pest Detection & Management

- Integrate camera and environmental sensors
- Use SLM to analyze conditions conducive to pests
- Control preventive systems (lighting, ultrasonic deterrents)
- Generate pest risk reports
- *Extension*: Combine with image detection for pest identification

Category E: Personal Wellness & Health

Smart Medication Reminder

- Monitor time, environmental conditions, and user input
- SLM provides personalized medication reminders
- Control visual/audio alerts appropriately
- Answer queries: "When should I take my medicine?"
- *Extension*: Track adherence and generate health reports

Workspace Wellness Monitor

- Track ambient light, noise levels, air quality, and posture sensors
- SLM provides ergonomic and wellness recommendations
- Control desk lighting and reminder systems
- Generate daily wellness reports
- *Extension*: Implement break reminders based on activity patterns

Sleep Quality Analyzer

- Monitor bedroom temperature, humidity, light, and noise
- SLM analyzes conditions and provides sleep optimization tips
- Control bedroom environment (lights, temperature)
- Generate sleep quality reports
- *Extension*: Implement long-term sleep pattern analysis

Technical Specifications

Hardware Requirements

Minimum Sensor Suite:

- At least two different types of sensors
- Examples: Camera, DHT22 (temp/humidity), BMP280 (pressure), buttons, PIR sensors, light sensors, sound sensors, moisture sensors

Minimum Actuator Suite:

- At least two different types of actuators (LEDS could simulate actuators)
- Examples: LEDs, relays, motors, servos, buzzers, displays, solenoids

Recommended Additional Components:

- Push buttons for manual input
- Status indicators (multi-color LEDs)
- Breadboard and jumper wires
- Appropriate resistors for circuits

SLM Requirements

Model Selection:

- Choose from: Llama 3.2 (1B/3B), Gemma 3 (1B/4B), Phi-3.5 (3.8B), or similar
- Model must be quantized to 4-bit (Q4_0 or Q4_K_M)
- Maximum model size: ~2.5GB
- Must demonstrate: text understanding, reasoning, and structured output generation

Optimization Techniques (implement at least ONE):

- **Function Calling**: Structured tool use for sensor reading or actuator control
- **RAG (Retrieval-Augmented Generation)**: Integration with knowledge base for enhanced accuracy
- **Prompt Engineering**: Advanced prompting strategies with JSON schema
- **Pydantic Integration**: Structured output validation

System Architecture Requirements

Modular Design:

- Separate hardware interface module (`hardware.py`)
- SLM inference module (`inference.py`)
- Main application logic (`main.py`)
- Configuration file for parameters

Data Management:

- Implement sensor data logging (CSV or JSON)
- Store command history
- Generate system performance metrics

User Interface (choose ONE or MORE):

- Terminal-based interactive interface
- Web dashboard (Flask/FastAPI)
- Physical interface (buttons + LEDs)
- Voice interface (bonus)

Deliverables

1. Technical Documentation

Project Report (PDF):

a) **Introduction & Motivation**

- Problem statement
- Real-world application context
- Project objectives and success criteria

b) **Literature Review**

- Related work in edge AI and IoT
- SLM capabilities and limitations
- Relevant optimization techniques

c) **System Design**

- Complete architecture diagram
- Hardware components and connections
- Software architecture and data flow
- SLM model selection and justification

d) **Implementation**

- Hardware setup and circuit diagrams
- Software implementation details
- SLM integration approach
- Optimization techniques applied
- Prompt engineering strategy

e) **Testing & Evaluation**

- Performance metrics (latency, accuracy, resource usage)
- Test scenarios and results
- Failure cases and limitations
- Reliability analysis

f) **Results & Discussion**

- Key achievements
- Challenges faced and solutions
- System limitations
- Lessons learned

g) **Future Work & Conclusions**

- Potential improvements
- Scalability considerations
- Alternative approaches

Technical Specifications Document:

- Complete parts list with sources
- Pin assignment table
- Installation and setup instructions
- API/function documentation
- Configuration parameters

2. Source Code and Implementation (Create a GitHub)

Code Quality Requirements:

- Modular Python code with clear separation of concerns
- Comprehensive comments and docstrings
- Type hints where appropriate
- Error handling for all sensor/actuator operations
- Configuration file for easy parameter adjustment
- README with setup instructions
- Requirements.txt for dependencies

Repository Structure: (sugestion)

```
project_name/
├── README.md
├── requirements.txt
├── config.yaml
├── hardware.py          # Hardware interface module
├── inference.py         # SLM inference module
├── main.py             # Main application
├── utils.py            # Helper functions
├── data/               # Data logs
├── models/             # Model files (if not using Ollama)
├── docs/               # Additional documentation
└── tests/              # Test scripts
```

3. Video Demonstration (Timing is only a suggestion)

Project Demonstration Video (10-15 minutes):

Must include:

a) **Team Introduction (1 minute)**

- Team members and roles
- Project overview

b) **System Overview (2-3 minutes)**

- Hardware components walkthrough
- System architecture explanation
- Key features demonstration

c) **Live Demonstration (5-7 minutes)**

- Show actual hardware setup
- Demonstrate real-time sensor readings
- Show SLM processing and responses
- Demonstrate actuator control
- Show multiple use case scenarios
- Display performance metrics

d) **Technical Deep Dive (2-3 minutes)**

- Show code structure
- Explain optimization techniques
- Demonstrate how SLM generates responses
- Show a prompt engineering approach

e) **Results & Conclusions (1-2 minutes)**

- Summarize achievements

- Discuss challenges and solutions
- Future improvements

Video Quality Requirements:

- Clear audio (use good microphone)
- Stable camera work (use tripod if possible)
- Show actual running system (no simulations)
- Include close-ups of hardware
- Display terminal/code when relevant
- Show real-time metrics

Submission Guidelines

File Organization

Submit a single ZIP file named: `teamX_project_name_part2.zip`

Required contents:

```
teamX_project_name_part2/
├── Report/
│   └── Team_X_Final_Report.pdf
│   └── Technical_Specifications.pdf
├── Code/
│   └── [complete source code repository]
├── Video/
│   └── Team_X_Demo_Video.mp4
└── README.md
```

README.md Suggestion:

- Team information (names, roles)
- Project title and description
- Hardware requirements
- Software dependencies
- Installation instructions
- Usage instructions
- Known issues
- Credits and acknowledgments

Submission Platform

- Upload to course platform (Moodle)

- Ensure all files are included and accessible
- Test the ZIP file before submission
- One submission per group

Submission Deadline

December 7th, 2024 (23:59)

Late submissions will not be accepted.

Important Notes

SLM Limitations

- Be aware that SLMs can produce inconsistent outputs
- Implement validation for critical decisions
- Document failure cases in your report
- Consider fallback mechanisms for safety-critical applications

Hardware Safety

- Follow proper electrical safety practices
- Use appropriate resistors with LEDs
- Verify voltage levels (3.3V for GPIO)
- Implement emergency stop mechanisms if needed

Performance Expectations

- SLM inference on Raspberry Pi 5 will have latency (30-90 seconds typical)
- Design your system to accommodate this latency
- Consider user experience during processing time
- Implement progress indicators

Testing & Debugging

- Test individual components before integration
- Use logging extensively for debugging
- Test edge cases and failure scenarios
- Verify system stability over extended periods

Resources

Recommended Tools

- **Ollama**: Easiest way to run SLMs locally
- **GPIO Zero**: Modern GPIO library for Raspberry Pi
- **Jupyter Notebooks**: For development and testing
- **CircuitPython**: For sensor integration
- **VSCode**: IDE with remote development capabilities

Documentation References

- Course textbook (https://mjrovai.github.io/EdgeML_Made_Ease_ebook/)
- Ollama documentation: https://ollama.com
- GPIO Zero documentation: https://gpiozero.readthedocs.io
- Raspberry Pi documentation: https://www.raspberrypi.org/documentation/

Community Support

- Raspberry Pi forums
- Ollama Discord channel
- Stack Overflow

Tips for Success

1. **Start Early**: SLM integration takes time to get right
2. **Test Hardware First**: Verify all sensors/actuators work before adding AI
3. **Iterate on Prompts**: Prompt engineering is crucial for reliable responses
4. **Log Everything**: Comprehensive logging helps debugging and analysis
5. **Handle Errors Gracefully**: SLMs can fail; design for resilience
6. **Consider User Experience**: Think about how users will interact with your system
7. **Document As You Go**: Don't leave documentation to the last minute
8. **Test Edge Cases**: What happens when sensors fail or give unexpected values?
9. **Monitor Resources**: Keep an eye on memory and CPU usage
10. **Have Fun**: This is an opportunity to create something innovative!

Inspiration & Example Ideas

- Smart plant that "tells" you what it needs
- An AI sous chef that monitors cooking conditions
- Intelligent pet feeder that adjusts portions based on activity
- Smart bike that monitors conditions and provides safety alerts
- AI-powered weather station with natural language forecasts
- An intelligent lab safety monitor that explains hazards

- Smart workspace that optimizes for productivity
- AI garden assistant that identifies plant problems

FAQ

Q: Can we use a pre-trained model without modification?
A: Yes, but you must implement at least one optimization technique (function calling, RAG, or advanced prompting).

Q: What if our SLM gives inconsistent responses?
A: Document this in your report and implement validation or retry mechanisms.

Q: Can we use cloud services?
A: No, all AI inference must run locally on the Raspberry Pi 5.

Q: How much code do we need to write?
A: Quality over quantity. A well-structured 500-line codebase is better than 2000 lines of messy code.

Q: Can we combine multiple project categories?
A: Absolutely! The categories are just suggestions.

Q: What if we can't get specific hardware components?
A: Contact instructors early. Alternative components may be acceptable. Use LEDS and/or buttons as simulations

Q: How important is the video quality?
A: Content is more important than production value, but clear audio and stable video are essential.

Q: Can we use models larger than 5B parameters?
A: Only if you can demonstrate acceptable inference time (<60s). Larger models typically won't perform well on Raspberry Pi 5.

Academic Integrity

- All code must be original or properly attributed (Including LLMs like ChatGPT or Claude)
- Cite all external resources and references

Conclusion

This project is your opportunity to demonstrate a good understanding of edge AI concepts and create an innovative system. By combining Small Language Models with physical computing, you're working at the cutting edge of IoT and AI integration.

Remember that the field of edge AI is rapidly evolving, and perfect solutions don't exist yet. Your project should demonstrate:

- Understanding of SLM capabilities and limitations
- Practical implementation skills
- Creative problem-solving
- Professional documentation

- Critical thinking about real-world applications

We look forward to seeing your innovative solutions!

Good luck, and may your tokens flow swiftly! 🤖

For questions or clarifications, please email me. Prof. Rovai