Republic of Rwanda
Ministry of Education

RTB | RWANDA TVET BOARD

SPEES402

Embedded Systems Software Integration

Integrate Embedded Systems Software

Competence

RQF Level: 4

Learning Hours: 80

Credits: 8

Sector: ICT and Multimedia

Trade: Software Programming and Embedded Systems

Module Type: Specific Module

Curriculum: ICTSES4002 TVET Certificate IV in Software Programming and Embedded Systems

1200

Issue Date: August 2023

Purpose statement	The purpose of this module is to equip learners with comprehensive knowledge and practical skills in embedded systems, spanning from firmware development to deployment. Through a holistic approach, this module aims to cultivate a deep understanding of embedded systems architecture, sensor integration, actuator control, and their seamless integration into the Internet of Things ecosystem. Learners will gain hands-on experience in designing, programming, and optimizing embedded systems firmware, while also exploring strategies for efficient deployment in real-world applications. By the end of this module, participants will be proficient in developing, interfacing, and deploying embedded systems, positioning them to tackle contemporary challenges and innovations in the field of IoT and embedded technology.

Learning assumed to be in place	- Embedded Systems Hardware Design - Networking Fundamentals - Web APIs - Applied physics I (LU4-Digital electronics) - Fundamentals of programming with c/c++

Delivery modality	**Training delivery**		**100%**	**Assessment**		**Total 100%**
	Theoretical content		**30%**	**Formative assessment**	**30%**	**50%**
	Practical work:		**70%**		**70%**	
	Group project and presentation	**30%**				
	Individual project /Work	**40%**				
				Summative Assessment		**50%**

Elements of Competence and Performance Criteria

Elements of competence	Performance criteria
1. Develop Embedded Systems Firmware	1.1. Firmware architecture diagrams are properly drawn based on embedded system architecture
	1.2. Development environment is correctly prepared based on embedded platform specifications.
	1.3. Resources are effectively optimized according to the performance requirement.
	1.4. Real-time requirements are perfectly handled according to the performance requirement.
	1.5. Firmware Image is correctly exported in line with the deployment requirement.
	1.6. Firmware is properly documented according to the work done
2. Integrate Peripherals with the Microcontroller	2.1. Hardware is properly selected based on the embedded system requirement.
	2.2. Data Flow Diagram is properly established based on the embedded system requirement.
	2.3. Hardware is properly connected to the microcontroller based on the data flow diagram.
	2.4. Communication code is properly written based on the embedded system requirement.
3. Communicate data over the network	3.1. Socket Programming is properly applied inline based on the required data exchange.
	3.2. IoT Networking is properly applied inline according to the system requirement.
	3.3. Errors occurring during data transmission are properly handled following debugging procedures.
	3.4. Web APIs are properly written to enable seamless communication and interaction between IoT devices based on the embedded system requirement.

Intended Knowledge, Skills and Attitude

Knowledge	Skills	Attitude
✓ Identify hardware components ✓ Describe the embedded system structure ✓ Identify Design principals and techniques ✓ Identify test types ✓ Identify debug technics ✓ Identify and prepare the environment	✓ Create data flow diadram ✓ Develop firmware (communication code) ✓ Install software ✓ Connect hardware ✓ Apply socket programming ✓ Apply IoT networking ✓ Perform error handling ✓ Create design layout ✓ Write APIs ✓ Prepare design project presentation	✓ Able to plan for the project ✓ Use creativity and innovation throughout the design works ✓ Pay attention projects details ✓ Demonstrate punctuality during the implementation ✓ Be resourcefulness in the new design trends ✓ Patience

Course content

Learning outcomes	At the end of the module the learner will be able to: 1. **Develop Embedded Systems Firmware** 2. **Integrate Peripherals with the Microcontroller** 3. **Communicate data over the network**

Learning outcome 1: **Develop Embedded Systems Firmware**	Learning hours: **24**

Indicative content

- **Firmware Architecture Diagram Making:**
  Following elements are shown on the diagram:
  - ✓ Microcontroller
  - ✓ Peripherals
    - Sensors
    - Actuators
    - Communication modules
  - ✓ Firmware Layers
    - Kernel
    - Drivers
    - Middleware
    - Application layers
  - ✓ Interactions shown as arrowed lines representing data and control flow.
  - ✓ External Interfaces
    - Ports
    - Connectors
    - Communication protocols

- **Preparation of Firmware Development Environment:**
  - ✓ **IDE Selection depending on the microcontroller**
    - Eclipse or
    - Keil or
    - IAR Embedded Workbench or
    - PlatformIO
  - ✓ **Toolchain Installation**
    - Install the toolchain specific to the microcontroller
    - Make sure to include compilers
    - Make sure to include assemblers
    - Make sure to include debuggers
  - ✓ **Hardware Setup**
    - Development board
    - Programmer/debugger
    - Any other necessary peripherals
  - ✓ **Version Control (e.g., Git) to manage the codebase**
  - ✓ **Project Configuration**
    - Build settings
    - Memory layout

- **Memory Usage Optimization in Firmware Development:**
  - ✓ Static Analysis
  - ✓ Code Profiling
  - ✓ Memory Pooling
  - ✓ Data Compression
  - ✓ Optimize Data Types

- **Power Consumption Optimization in Firmware Development:**
  - ✓ Low-Power Modes
  - ✓ Peripheral Management
  - ✓ Dynamic Voltage and Frequency Scaling

- ✓ Efficient Algorithms
- ✓ Interrupt-Based Design

- **Real-Time Requirements Handling in Firmware Development:**
  - ✓ **RTOS Selection for task scheduling**
    - FreeRTOS
    - Micrium (now part of Silicon Labs)
    - NuttX
    - ChibiOS/RT
    - CMSIS-RTOS (e.g., Keil RTX)
    - TI-RTOS (formerly SYS/BIOS)
    - embOS
    - Zephyr
    - RIOT
    - ThreadX
  - ✓ **Priority Management**
  - ✓ **Deadlock Prevention**
  - ✓ **Timing Analysis**
  - ✓ **Firmware Image Exportation**:
    - Programming Tools
    - Bootloaders
    - Image Signing

- **Firmware Documentation**
  - ✓ Code Comments
  - ✓ API Documentation
  - ✓ User Manuals
  - ✓ Version History
  - ✓ Schematics and Diagrams

Resources required for the learning outcome

Equipment	• Microcontroller Development Board • Programmer/Debugger Hardware • Development Computer • Peripherals (Sensors, actuators, communication modules) • External Interfaces like USB for programming/debugging • Hardware for hardware schematics (e.g., circuit design tools)
Materials	• IDE (Integrated Development Environment) Software (e.g., Eclipse, Keil, IAR Embedded Workbench, PlatformIO) • Toolchain (compilers, assemblers, debuggers) • Project-specific Configurations • Static Analysis Tools (e.g., Lint) • Code Profiling Tools • Custom Memory Management Code

	• Data Compression Techniques (e.g., run-length encoding) • Real-Time Operating System (RTOS) Software (e.g., FreeRTOS, Micrium) • Priority Assignment Guidelines • Synchronization Mechanisms Design Guidelines • User Manual Documentation • Version History/Changelog • Documentation Templates
Tools	• Memory Usage Optimization Tools (e.g., memory analysis tools) • Power Consumption Optimization Tools • Interrupt-Based Design Guidelines • Timing Analysis Tools • Bootloaders (for firmware updates) • Image Signing Tools (for security) • Code Commenting Guidelines • API Documentation Guidelines • Effective Documentation Practices • Version Control System (e.g., Git) • Debugging Tools provided by the microcontroller manufacturer
Facilitation techniques	➢ Demonstration and simulation ➢ Practical exercise ➢ Individual work ➢ Trainer-guided ➢ Group discussion
Formative assessment methods /(CAT)	✓ Oral assessment ✓ Written assessment ✓ Practical assessment

Learning outcome 2: **Integrate Peripherals with the Microcontroller**	Learning hours: **32**
Indicative content	
• **Hardware Selection for an Embedded System** ✓ **Microcontroller/Processor** + Power consumption + Cost + Ecosystem support ✓ **Power Supply:** + Voltage levels + Current capacity + Energy efficiency	

- ✓ **Sensors and Actuators**
  - Identify sensors (e.g., temperature sensors, accelerometers, ...)
  - Identify Actuators (e.g., motors, relays, ...)
- ✓ **Communication Interfaces**
  - UART
  - SPI
  - I2C
  - Ethernet
  - Any other suitable communication interface

- ✓ **Additional Peripherals**
  - Displays
  - Input devices
  - WiFi Communication modules (e.g., Wi-Fi, Bluetooth)
- ✓ **Environmental Considerations**
  - HumidityShock resistance
  - Cost and Availability
  - Scalability

- **Data Flow Diagram**

  - ✓ Processes
    - Data Flow
    - Data Stores
  - ✓ **Entities**
    - Internal
    - External
  - ✓ **Data Flow Labels**
    - Identification of data type
    - Data format
    - Directions
  - ✓ **Context Diagram**
  - ✓ **Leveling**

- **Connecting Peripherals to the Microcontroller**
  - ✓ Pin Selection
    - Wiring
    - Power Supply
    - Pull-up/Pull-down Resistors
  - ✓ Clock Configuration
  - ✓ Initialization

- **Code to Make Communication Between Microcontroller and Peripherals**
  - ✓ Select the Communication Protocol
    - Identify the Microcontroller Pins
    - Initialize the Communication Interface
    - Implement Communication Functions
  - ✓ Main Program
    - Censors
    - Configure Peripheral
  - ✓ Testing and Debugging
    - Error Handling
    - Optimization
  - ✓ Safety and Reliability

✓ Documentation	
Equipment	• Microcontroller/Processor • Power Supply • Sensors and Actuators • Communication Interfaces • Peripherals • Environmental Testing Equipment • Budget and Supplier Information • Whiteboard or Paper • Marker or Pen • Diagram Drawing Software • Wires and Cables • Multimeter • Resistors • Computer with Development Environment • USB Cable for Microcontroller Programming • Debugging Tools (e.g., oscilloscope, logic analyzer) • Documentation Tools
Materials	• Lecture Slides and Presentations • Textbooks and Reference Materials • Code Examples • Online Resources (e.g., tutorials, documentation, forums) • Hands-on Lab Exercises and Projects (with instructions) • Safety Guidelines and Documentation • Data Sheets and Datasheets for Microcontrollers and Components • Microcontroller and Peripheral Hardware Documentation (e.g., user manuals) • Example Code Snippets (for communication protocols) • Sample Hardware Selection Case Studies • Assessment Tools (e.g., quizzes, assignments)
Tools	• Diagram Drawing Software (for data flow diagrams) • Documentation Templates • Simulation Software (if applicable)
Facilitation techniques	➢ Demonstration and simulation ➢ Practical exercise ➢ Individual work ➢ Trainer-guided

	➢ Group discussion
Formative assessment methods /(CAT)	✓ Oral assessment ✓ Written assessment ✓ Practical assessment

Learning outcome 3: **Communicate data over the network**	**Learning hours: 24hrs**

Indicative content

- **Network Socket Programming basics:**

✓ Network Socket Creation
  - Data Transmission: TCP, UDP
  - WebSocket Basics
  - WebSocket Protocol

✓ Binding and Addressing
  - Server-Client Model
  - Host machine
  - Specific IP address and port number
  - Error Handling
  - Network Socket Closing
  - Real-Time Communication

✓ Security and Best Practices

- **IoT Networking:**

✓ Low-Power Protocols
  - IoT devices
  - Communication protocols

✓ Wireless Technologies
  - Types of wireless
  - Advantage and disadvantages

✓ IPv6 Adoption

✓ Mesh Networking
  - Advantages
  - Connectivity between devices

✓ Edge Computing

✓ Security
  - Security measures
  - Security procedures

- **Techniques to Handle Errors During Data Transmission:**

✓ Error Detection and Correction
  - Error identification
  - Error detection methods
  - Implementation of error detection
  - Retry Strategies

✓ Acknowledgement and Retransmission
  - Automatic repeat request (ARQ)
  - Identify network protocol
  - Timeouts
  - Flow Control

- **Web APIs for IoT Communication:**

✓ RESTful APIs
  - REST methods
  - Data Formats (JSON and XML)
  - Endpoints and Resources

✓ Security measures
  - Authentication
  - Authorization
  - Attacks
  - Error Handling
  - Rate Limiting

✓ Documentation
  - Purpose and structure
  - Tools

Resources required for the Learning outcome

Equipment	• Networking Equipment (e.g., computers or Raspberry Pi devices for hands-on socket programming, computers or servers to host and test APIs, Wi-Fi routers, switches, IoT development boards) • IoT Devices and Sensors (e.g., Raspberry Pi, Arduino, IoT sensors) • Simulation Hardware (if physical IoT network simulation is used)
Materials	• Lecture Slides • Textbooks on Socket Programming (e.g., "Python Network Programming" by Dr. M. O. Faruque Sarker) • Code Examples (Python socket programming code samples) • Security Tools (OpenSSL for encryption) • Assessment Tools (Socket programming coding exercises) • Simulation Software (e.g., IoT network simulation software like Contiki) • Online Resources (Links to socket programming tutorials) • Assessment Tools (IoT project assignments) • Simulation Software (e.g., Network simulators like NS-3) • Code Examples (Error handling code samples in Python) • Assessment Tools (Error handling coding challenges)
Tools	• Programming Environments (e.g., IDEs like Visual Studio Code, PyCharm) • Online Resources (e.g., tutorials, documentation, forums) • Assessment Tools (e.g., quizzes, assignments) • WebSocket Server Software (e.g., Node.js WebSocket library) • Development Tools (e.g., Postman, Swagger) • API Documentation Tools

	● Error Handling Code Examples ● IoT Development Tools (e.g., Arduino IDE, Raspberry Pi tools) ● IoT Simulation Hardware (if physical IoT network simulation is used)
Facilitation techniques	➢ Demonstration and simulation ➢ Practical exercise ➢ Individual work ➢ Trainer-guided ➢ Group discussion
Formative assessment methods /(CAT)	✓ Oral assessment ✓ Written assessment ✓ Practical assessment

Integrated/Summative assessment (For specific module)

Integrated situation

Step into the dynamic world of ABC Electronics, a rising startup committed to crafting innovative home automation products. At the heart of this venture lies an exciting project—an intelligent environmental monitoring and security system. While the circuit design stands firm, the bridge between hardware and functionality is yet to be built. As the appointed firmware developer, the mission is to orchestrate the symphony of code that harmonizes seamlessly with the system hardware. the task is to craft a software solution that embraces indoor temperature and humidity monitoring while capturing images upon detecting motion. The code will not only enhance user experiences but also create a haven of optimal living conditions. The grand stage awaits the mastery in firmware development.
Develop firmware for real-time indoor temperature and humidity monitoring, displaying on LCD. Enable remote server access for data tracking. Create motion-triggered image capture firmware with remote server access for storage and user alerts.Craft robust code for sensor anomalies and communication, integrating remote server access for error reporting and diagnostics.

Resources

Tools	● Wire Strippers and Cutters ● Crimping Tool ● Hand Tools (e.g., screwdrivers, pliers) ● Safety Equipment (e.g., safety glasses, gloves)
Equipment	● Power Supply Unit (PSU) ● Voltage Regulators ● Storage Medium (e.g., SD Card) ● Enclosure ● Mounting Hardware ● Hot Air Rework Station ● 3D Printer ● Oscilloscope Probes ● Electrostatic Discharge (ESD) Protection ● Bench Power Supply
Materials/ Consumables	● Heat Shrink Tubing and Electrical Tape ● Breadboard Jumpers ● PCB Design Software and Services ● Wire Labels and Cable Management

Assessable outcomes	Assessment criteria (Based on performance criteria)	Indicator	Observation		Marks allocation
			Yes	No	
Learning outcome 1: Design Firmware Architecture (20%)	Analysis of Firmware requirements Preparing of Development environment	C or C++ Programming Language is selected			1
		IDE is selected			1
		Documenting tool is selected			1
		Diagramming Tools			1
		IDE is configured			2
	Selecting of Tools, materials and equipment	Diagramming Tools			1
	Drawing of Firmware architecture diagrams	Functional Flow Diagram(flowchart)			2
		Architecture Diagram.			2
		Data Flow and Communication (Sequence Diagram)			2
	Documentation of firmware architecture	Use case			1
		components of firmware architecture			1
		User guide			1
Learning outcome 2: Implement firmware system design. (64%)	Preparation of Development environment Exporting Firmware Image	LDC ports are configured			4
		Flashing tool is installed			4
	Integration of Communication Protocols	I2C			2
		Spi Communication modulus			2
		wifi			2
		Integration of communication protocols			2
	Implementation of firmware modules	Initialization			2
		Data Acquisition			2
		Data Processing (image capture)			2
		Control (motion detection)			2
		Communication (protocol motion)			2

		Memory Management (optimization)			2
		RTOS			2
		Interrupt Handler Module			2
		Power Management (sleep mode)			2
	Performance of memory management in Embedded C.	EEPROM management			2
		Designing the own API			2
		Mapping Memory with Pointers			2
	Writing Firmware source codes	Temperature is displayed in °C			1
		Humidity is displayed in %			
		High Temperature alert is configured			2
		motion-triggered			1
		image capture			2
		Low Temperature alert is configured			3
Learning Outcome 3: Deploy Firmware (15%)	Preparation of Deployment environment	Flashing tool is installed			2
	Exporting Firmware Image	.hex file for the firmware is created			4
		The firmware is flashed			4
	Documentation of Firmware	Documentation tool is installed			2
		Documentation tool is configured			2
Total marks					79
Percentage Weightage					100%
Minimum Passing line % (Aggregate): 70%					

References:

1. Spiceworks. (2022, October 10). What is Firmware? Architecture and Best Practices. https://www.spiceworks.com/tech/devops/articles/what-is-firmware/

2. Berg, H. K., Rao, P., & Shriver, B. D. (1982). Firmware quality assurance. *National Computer Conference.* https://doi.org/10.1145/1500774.1500776

3. IEEE Conference Publication | IEEE Xplore. (2016, March 1). Verifying information flow properties of firmware using symbolic execution. https://ieeexplore.ieee.org/abstract/document/7459333/

4. Al-Hammouri, A. T. (2012). A comprehensive co-simulation platform for cyber-physical systems. *Computer Communications, 36*(1), 8–19. https://doi.org/10.1016/j.comcom.2012.01.003

5. Beghi, A., Marcuzzi, F., & Rampazzo, M. (2016). A virtual laboratory for the prototyping of Cyber-Physical systems. *IFAC-PapersOnLine.* https://doi.org/10.1016/j.ifacol.2016.07.154

6. Baheti, R., & Gil, H. (2011). Cyber-physical Systems. *Google Scholar.* https://scholar.google.com/scholar_lookup?title=Cyber-physical%20Systems&publication_year=2011&author=R.%20Baheti&author=H.%20Gil

7. Di Matematica "Tullio Levi-Civita" - Dm, D. (2014). Computing from LaTeX: automated numerical computing from LaTeX expressions. http://paduaresearch.cab.unipd.it/6930/

8. Beghi, A., Marcuzzi, F., & Rampazzo, M. (2016). A virtual laboratory for the prototyping of Cyber-Physical systems. *IFAC-PapersOnLine.* https://doi.org/10.1016/j.ifacol.2016.07.154

9. Qt Company. (n.d.). Embedded Software Programming Languages: pros, cons, and comparisons of popular languages. https://www.qt.io/embedded-development-talk/embedded-software-programming-languages-pros-cons-and-comparisons-of-popular-languages

10. Udemy. (2023, January). An Introduction to the Fundamentals of Firmware Engineering for Embedded Systems.

https://www.udemy.com/course/firmware-engineering/?amp=&aff_code=Ewh3Y1lWQH8FQR93MkBPbG1RGXFfW1h8B14ZeU5TQ3YBRxFzWj5XMRM%3D&pmtag=CAREERS24LEARN15&utm_source=adwords&utm_medium=udemyads&utm_campaign=DSA_Catchall_la.EN_cc.ROW&utm_content=deal4584&utm_term=_._ag_88010211481_._ad_535397282064_._kw__._de_c_._dm__._pl__._ti_dsa-52949608673_._li_1012087_._pd__._&matchtype=&gclid=Cj0KCQjw2qKmBhCfARIsAFy8buIG982c3sLuhWj48XsDI0owJY42DJNT7YhDzSwck8tEbQSaN7Ig5eUaAvBqEALw_wcB

11. 5V Media. (2022, September 28). Blog - 4 Communication protocols embedded engineers should know. https://www.weare5vmedia.com/media/communication-protocols-for-an-embedded-engineer-to-know

12. Neil's Log Book. (n.d.). Lab Guide: Embedded Memory Management. https://nrqm.ca/nrqm.ca/mechatronics-lab-guide/lab-guide-embedded-memory-management/index.html

13. Cyrille, C., Dross, C., Gilcher, F., & Moy, Y. (n.d.). Dynamic Memory Management in Critical Embedded Software.

14. Beningo, J. (2013, January 10). Building reusable device drivers for microcontrollers. *Embedded.com*. https://www.embedded.com/building-reusable-device-drivers-for-microcontrollers/

15. Tutorialspoint. (n.d.). C - Memory management. https://www.tutorialspoint.com/cprogramming/c_memory_management.htm

16. GeeksforGeeks. (2022). Memory layout of C programs. https://www.geeksforgeeks.org/memory-layout-of-c-program/

17. Gupta, E. (2022, February 28). Memory Layout in C - Scaler topics. *Scaler Topics*. https://www.scaler.com/topics/c/memory-layout-in-c/

18. Mathew, H. (2022, October 12). The Complete Guide To Embedded Firmware Development. *Live Positively*. https://technocore360.livepositively.com/the-complete-guide-to-embedded-firmware-development/

19. Xukyo. (2021). Generating and uploading HEX files to an Arduino. *AranaCorp*. https://www.aranacorp.com/en/generating-and-uploading-hex-files-to-an-arduino/

20. Arduino Forum. (2017, August 8). Creating a flashable *.hex-file from a C-Source using the arduino tools. https://forum.arduino.cc/t/creating-a-flashable-hex-file-from-a-c-source-using-the-arduino-tools/473620/11

Glossary

1. **Purpose Statement**: A statement that defines the objectives and goals of a module or course.
2. **Embedded Systems**: Systems that consist of hardware and software designed for specific functions within a larger system.
3. **Firmware Development**: The process of creating software that is permanently programmed into a hardware device, typically an embedded system.
4. **Deployment**: The process of making software or hardware available for use in a real-world environment.
5. **IoT (Internet of Things)**: A network of interconnected physical devices or objects that can communicate and exchange data over the internet.
6. **Learning Assumed to be in Place**: The foundational knowledge and skills that learners are expected to have before taking the module.
7. **Delivery Modality**: The method or approach used to deliver training, such as in-person training, online courses, etc.
8. **Assessment**: The process of evaluating a learner's performance or understanding of the material.
9. **Theoretical Content**: The portion of the module that focuses on theoretical knowledge.
10. **Formative Assessment**: Ongoing assessments used to provide feedback to learners and improve their understanding.
11. **Practical Work**: Hands-on exercises or projects that allow learners to apply their knowledge.
12. **Group Project and Presentation**: Collaborative projects that involve multiple learners working together.
13. **Individual Project/Work**: Independent work or projects undertaken by individual learners.
14. **Summative Assessment**: A final assessment used to evaluate a learner's overall understanding and performance.
15. **Elements of Competence**: Specific skills or abilities that learners are expected to develop.
16. **Performance Criteria**: Criteria used to measure a learner's achievement of competence in a particular area.
17. **Firmware Architecture**: The structure and organization of software code for embedded systems.
18. **IDE (Integrated Development Environment)**: A software application that provides comprehensive facilities for software development.
19. **Toolchain**: A set of programming tools used for building software for a specific target platform.
20. **RTOS (Real-Time Operating System)**: An operating system designed for real-time applications that require precise timing and control.
21. **Socket Programming**: The use of software sockets to enable communication between computers or devices over a network.

22. **Web APIs (Application Programming Interfaces)**: Interfaces that allow different software applications to communicate with each other over the internet.
23. **Data Flow Diagram**: A visual representation of how data flows within a system or process.
24. **Microcontroller**: A small computer on a single integrated circuit used in embedded systems.
25. **Communication Protocols**: Standard rules and conventions for data communication between devices.
26. **HTTP (Hypertext Transfer Protocol)**: The protocol used for transferring data on the World Wide Web.
27. **WebSockets**: A communication protocol that enables bidirectional, real-time communication between clients and servers.
28. **IoT Networking**: Networking technologies and protocols used in the Internet of Things.
29. **MQTT (Message Queuing Telemetry Transport)**: A lightweight, publish-subscribe messaging protocol designed for low-bandwidth, high-latency, or unreliable networks.
30. **CoAP (Constrained Application Protocol)**: A protocol designed for resource-constrained devices and networks in IoT applications.
31. **LoRaWAN (Long Range Wide Area Network)**: A low-power, wide-area networking protocol designed for long-range communication in IoT devices.
32. **IPv6**: The latest version of the Internet Protocol, designed to accommodate the growing number of devices connected to the internet.
33. **Mesh Networking**: A network topology in which each node relays data for the network.
34. **Error Handling**: The process of detecting, reporting, and managing errors in software or hardware.
35. **API Documentation**: Documentation that explains how to use an API, including endpoints, data formats, and authentication.
36. **Demonstration and Simulation**: Teaching techniques that involve showing or simulating how something works.
37. **Practical Exercise**: Hands-on activities that reinforce theoretical learning.
38. **Individual Work**: Independent work or assignments completed by individual learners.
39. **Trainer-Guided**: Learning activities facilitated by an instructor or trainer.
40. **Formative Assessment Methods (CAT)**: Continuous assessment methods used to monitor learner progress.
41. **Integrated/Summative Assessment**: A final assessment that evaluates overall learning outcomes.
42. **Aggregate**: The total or combined result or percentage.