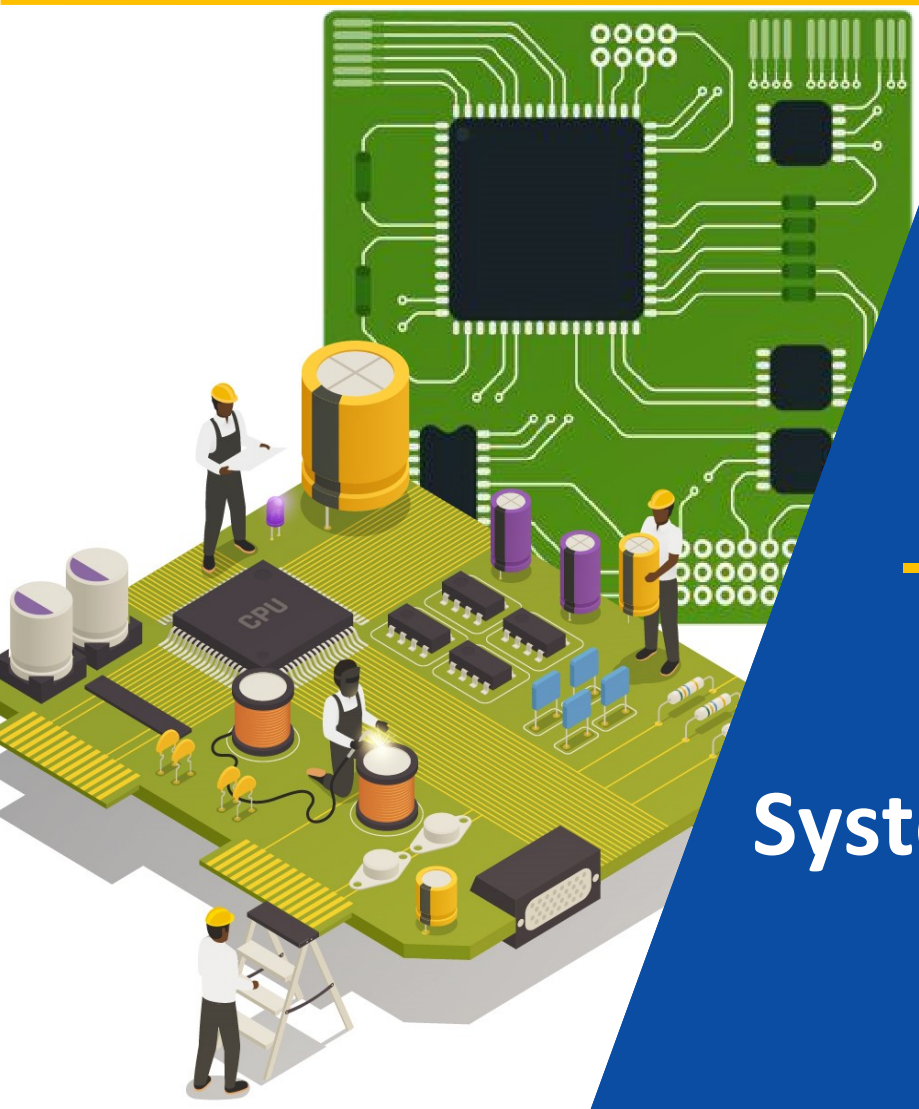
RQF LEVEL 4



CSAES401

COMPUTER SYSTEM AND ARCHITECTURE

Embedded System Hardware Development

TRAINEE'S MANUAL

October, 2024

EMBEDDED SYSTEM HARDWARE DEVELOPMENT

KOICA Korea International Cooperation Agency

TQUM TVET Quality Management Project

2024

AUTHOR'S NOTE PAGE (COPYRIGHT)

Copies available from:

- *HQs: Rwanda TVET Board-RTB*
- *Web: www.rtb.gov.rw*
- **KIGALI-RWANDA**

Original published version: October 2024

ACKNOWLEDGEMENTS

The publisher would like to thank the following for their assistance in the elaboration of this training manual:

Rwanda TVET Board (RTB) extends its appreciation to all parties who contributed to the development of the trainer's and trainee's manuals for the TVET Certificate IV in Computer System and Architecture specifically for the module " **CSAES401**: **Embedded System Hardware Development**".

We extend our gratitude to KOICA Rwanda for its contribution to the development of these training manuals and for its ongoing support of the TVET system in Rwanda.

We extend our gratitude to the TQUM Project for its financial and technical support in the development of these training manuals.

We would also like to acknowledge the valuable contributions of all TVET trainers and industry practitioners in the development of this training manual.

The management of Rwanda TVET Board extends its appreciation to both its staff and the staff of the TQUM Project for their efforts in coordinating these activities.

This training manual was developed:

Under Rwanda TVET Board (RTB) guiding policies and directives

Under Financial and Technical support of

COORDINATION TEAM

RWAMASIRABO Aimable
MARIA Bernadette M. Ramos
MUTIJIMA Asher Emmanuel

Production Team

Authoring and Review

UWIKUNDA Idesbald

IMENA MAKUZA Patrick

Validation

NIYOYITA Amani

TUYISENGE Jean Claude

NSENGIYUMVA Alfred

Conception, Adaptation and Editorial works

HATEGEKIMANA Olivier

GANZA Jean Francois Regis

HARELIMANA Wilson

NZABIRINDA Aimable

DUKUZIMANA Therese

NIYONKURU Sylvestre

BIZIMANA Eric

Formatting, Graphics, Illustrations, and infographics

YEONWOO Choe
SUA Lim
SAEM Lee
SOYEON Kim
WONYEONG Jeong
HABIMANA Emmanuel

Financial and Technical support

KOICA through TQUM Project

TABLE OF CONTENT

ACRONYMS

AC: Alternating Current
ADC: Analog-to-Digital Converter:
CAD: Computer-Aided Design:
CNC: Computer Numerical Control
DC: Direct Current
EDA: Electronic Design Automation
ESD: Electrostatic Discharge:
GND: Ground:
IC: Integrated Circuit
KOICA: Korean International Cooperation Agency
LED: Light Emitting Diode
PCB: Printed Circuit Board
PWM: Pulse Width Modulation
RTB: Rwanda TVET Board
SMD: Surface-Mount Device:
SPICE: Simulation Program with Integrated Circuit Emphasis
TQUM Project: TVET Quality management project
VLSI: Very Large Scale Integration

INTRODUCTION

This trainee's manual includes all the knowledge and skills required in computer system and architecture specifically for the module of **"Embedded System Hardware Development".** Students enrolled in this module will engage in practical activities designed to develop and enhance their competencies. The development of this training manual followed the Competency-Based Training and Assessment (CBT/A) approach, offering ample practical opportunities that mirror real-life situations.

The trainee's manual is organized into Learning Outcomes, which is broken down into indicative content that includes both theoretical and practical activities. It provides detailed information on the key competencies required for each learning outcome, along with the objectives to be achieved.

As a trainee, you will start by addressing questions related to the activities, which are designed to foster critical thinking and guide you towards practical applications in the labour market. The manual also provides essential information, including learning hours, required materials, and key tasks to complete throughout the learning process.

All activities included in this training manual are designed to facilitate both individual and group work. After completing the activities, you will conduct a formative assessment, referred to as the end learning outcome assessment. Ensure that you thoroughly review the key readings and the 'Points to Remember' section.

MODULE CODE AND TITLE: CSAES401 Embedded System Hardware development

Learning Outcome 1: Setup Workplace
Learning Outcome 2: Build Embedded System Hardware
Learning Outcome 3: Integrate Embedded System Hardware

Indicative Contents
1.1 Identification of Embedded System Requirements **1.2 Identification of Tools, Materials and Equipment** **1.3 Organising the Workplace**

Key Competencies for Learning Outcome 1: Set up Workplace

Knowledge	Skills	Attitudes
● Description of embedded system ● Description of microcontrollers and microprocessors ● Description of embedded system memory. ● Description of embedded system inputs and outputs. ● Description of peripheral requirements of embedded system. ● Description of real time operating system ● Identification of embedded system application scope ● Identification of tools, materials and equipment of embedded system.	● Applying safety measure ● Arranging tools, materials and equipment ● Determining the application scope	● Having Precision ● Being Attentive ● Having self-confident ● Having accountability ● Respecting time ● Being patient ● Having self-motivation ● Being organized ● Being passionate ● Having creativity

Duration: 30 hrs

Learning outcome 1 objectives:

By the end of the learning outcome, the trainees will be able to:

1. Describe Cleary embedded system according to the work to be done.
2. Describe correctly the peripheral requirements for embedded system based on work to be done
3. Describe properly real time operating system based on to the work to be done
4. Organize correctly the workplace based on standard operation procedures (SOP) document.
5. Arrange properly tools, materials and equipment according to arrangement techniques
6. Determine properly application scope for embedded system based on customer needs.

Resources

Equipment	Tools	Materials
• Microcontroller • Development Kits (with debugger and programmer) • Multimeter • Soldering Station • Power Supplies	• Soldering Iron • Pliers • DE soldering Pump	• PCB Copper Clad • Active components • Passive Components • Sensors • soldering tin • Wires

Indicative content 1.1: Identification of embedded system requirements

Duration: 24 hrs

Theoretical Activity 1.1.1: Description of Embedded System

Tasks:

1: Answer the following questions:

i. What is an embedded system?
ii. What are the main components of embedded system?
iii. What are types of embedded system?
iv. What are characteristics of embedded system
v. Where embedded system can be applied?

2: Provide the answers for the asked questions and write them on flipchart/papers.

3: Present the findings/answers to the whole class.

4: For more clarification, read the key readings 1.1.1.

5: In addition, ask questions where necessary.

Key readings 1.1.1.: Description of embedded system

1. Definition

Embedded System is an integrated system that is formed as a combination of computer hardware and software for a specific function.

Three main components of embedded systems are:

- Hardware: are physical part of the embedded system. It includes:

- **Microcontroller/Microprocessor**: The core processing unit that controls the operations of the system.
- **Memory:** Used to store data and instructions (e.g., RAM, ROM, Flash memory).
- **Peripherals:** Devices connected to the microcontroller, such as sensors, actuators, communication modules, and input/output (I/O) interfaces.
- **Power Supply:** Provides the necessary energy to run the embedded system.

- Software: are the high-level programs that control the operations of the embedded system. This includes:

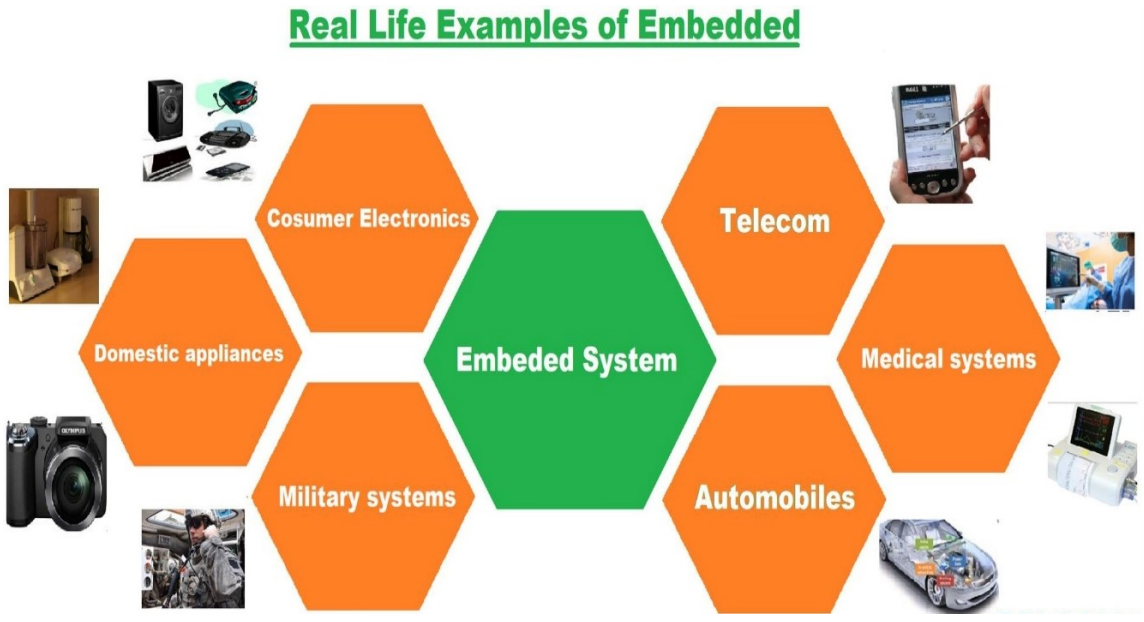
- **Application software:** Specific programs that carry out the tasks for which the embedded system is designed (e.g., controlling a washing machine or operating a car's engine).

- **Operating system (if present):** Some embedded systems use Real-Time Operating Systems (RTOS) to manage time-sensitive tasks and processes efficiently.
- **Drivers:** Code that helps interface the hardware components (e.g., sensors) with the software.

Firmware: Firmware is a specialized form of software that is embedded directly into the hardware. It acts as a bridge between the hardware and the application software. It is typically stored in non-volatile memory (e.g., Flash memory or ROM) and provides low-level control of the hardware.

Some examples of embedded systems are:

- Digital watches
- Washing Machine
- Toys
- Televisions
- Digital phones
- Laser Printer
- Cameras
- Industrial machines
- Electronic Calculators
- Automobiles
- Medical Equipment



2. types of embedded system

We can classify embedded systems based on performance and functional requirements and based on the performance of the microcontroller.

2.1. Classification of embedded systems based on performance and functional requirements

2.1.1. Stand-alone Embedded Systems

This type of embedded systems does not require a host system like a computer or a processor as it works by itself and displays data on the connected device or make necessary changes on the device. Input data is taken from the ports as analog or digital signals, and processing is done in the port itself. The result after proper calculation and conversion is displayed through a connected device.

2.1.2. Real-time Embedded Systems

When an output is required at a particular time, real-time embedded systems can be used. An external environment is controlled with the help of computer systems and connected through sensors or any other output/input interfaces. We can schedule the output either through a static or dynamic manner. There are two types under this category. They are soft and hard real-time embedded systems.

2.1.3. Network Embedded Systems

When a program is running inside another device, a network is formed. This is called network embedded systems, where a microprocessor or a controller controls the running program. A network is related to this system, and they can be either LAN or WAN. It is not necessary that the connection should be wired or wireless.

2.1.4. Mobile Embedded Systems

All the devices that are portable and working with an embedded system is a mobile embedded system. The best example that we can connect easily is mobile phones, laptops, and calculators.

2.2. Classification of embedded systems based on the performance of the microcontroller

2.2.1. Small Scale Embedded Systems

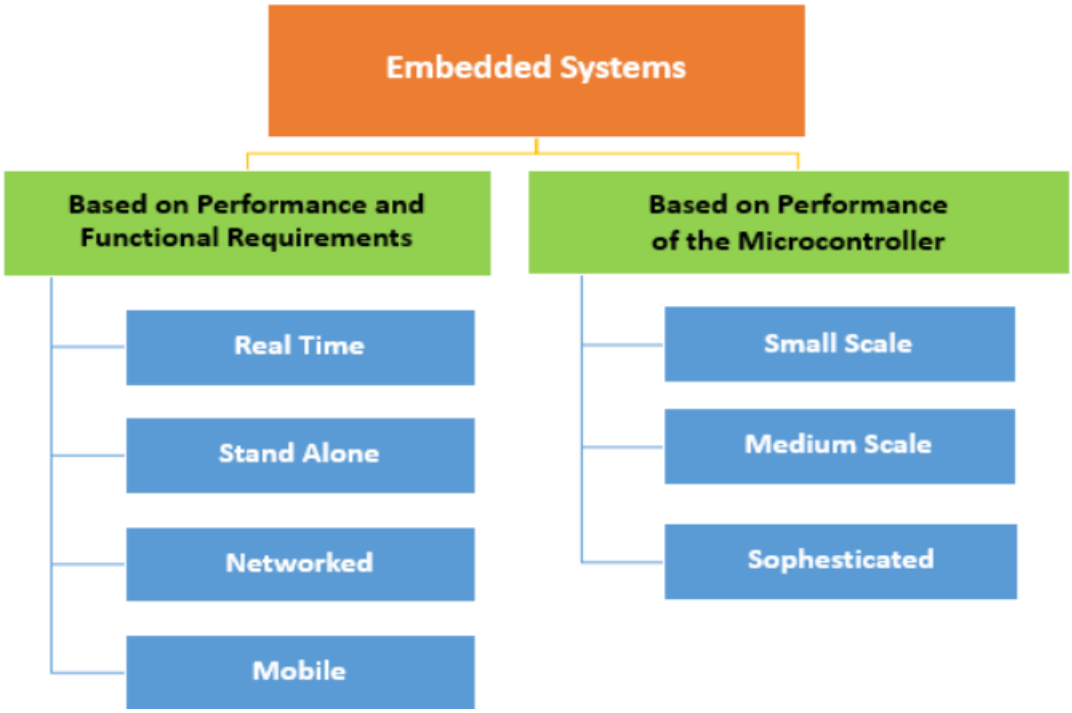
8 bit or 16-bit microcontrollers are used to design these and work with the help of batter in the system. Several programming tools are used to develop small scale embedded systems. The hardware is very small, and the processor is slow. The memory is also less. The codes for developing these embedded systems can be written with the help of any IDE.

2.2.2. Medium Scale Embedded Systems

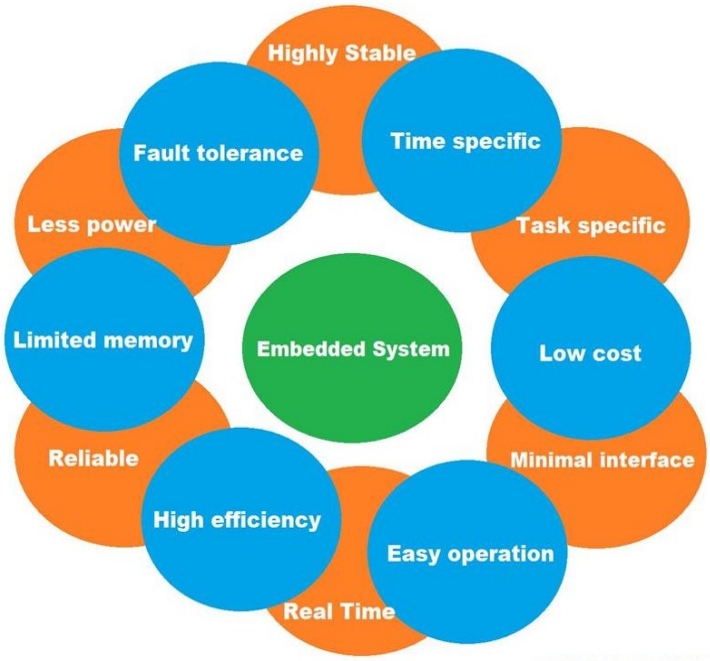
16 bit or 32-bit microcontrollers are used to develop medium systems. In addition, these can be developed with DSPs or RISCs. Hardware and software functionalities are complex, and several coding languages can be used as programming tools. As small-scale systems, an IDE is required for medium scale systems also. We can use medium-scale systems in high-end applications with large memory and processing data.

2.2.3. Sophisticated Embedded Systems

The most complex embedded system with all the difficult complexities of hardware and software that makes the system useful for all is called sophisticated embedded systems. These systems require registers of huge memory, scalable processors, and IPs to work well in any environment. They are used in systems with graphical screens, touchpads, and cutting-edge options where software and hardware are equally needed for performance.

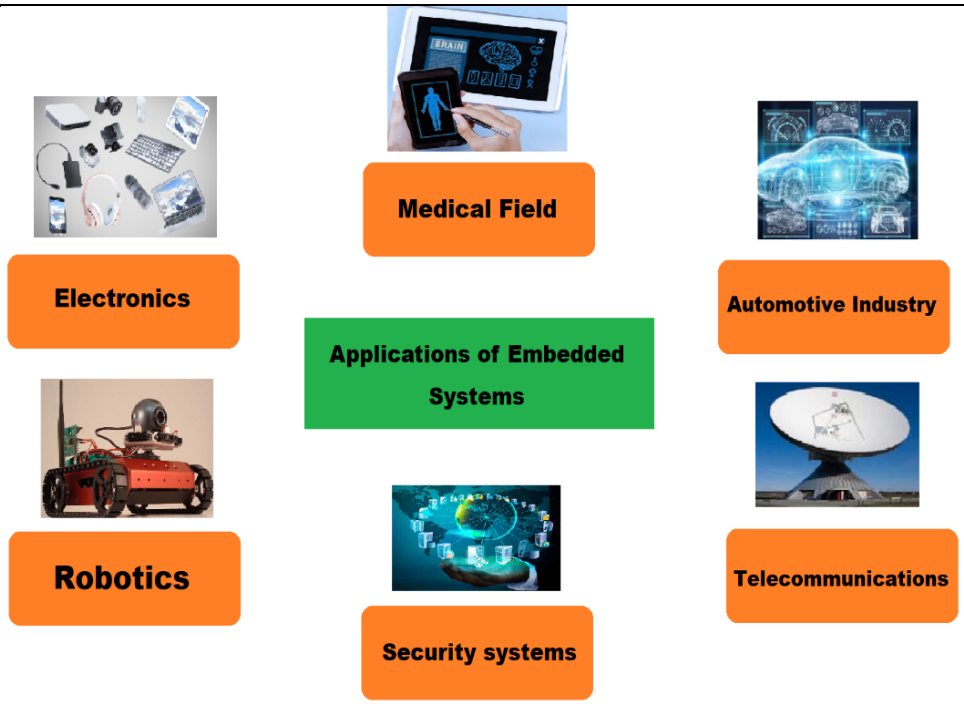


3. characteristics of embedded system



- **Performs specific task:** Embedded systems perform some specific function or tasks.
- **Low Cost:** The price of an embedded system is not so expensive.
- **Time Specific:** It performs the tasks within a certain period.
- **Low Power:** Embedded Systems do not require much power to operate.
- **High Efficiency:** The efficiency level of embedded systems is so high.
- **Minimal User interface:** These systems require less user interface and are easy to use.
- **Less Human intervention:** Embedded systems require no human intervention or very less human intervention.
- **Highly Stable:** Embedded systems do not change frequently mostly fixed maintaining stability.
- **High Reliability:** Embedded systems are reliable they perform tasks consistently well.
- **Use microprocessors or microcontrollers:** Embedded systems use microprocessors or microcontrollers to design and use limited memory.
- **Manufacturable:** The majority of embedded systems are compact and affordable to manufacture. They are based on the size and low complexity of the hardware.
- **Integration:** Hardware and software components in embedded systems are tightly integrated to achieve optimal performance and reliability. Custom hardware may be designed to meet specific requirements.
- Long Lifecycle: Embedded systems often have longer lifecycles compared to consumer electronics.

4. Applications of embedded system



4.1. Consumer Electronics

- Smartphones and tablets.
- Digital cameras and camcorders.
- Smart TVs and set-top boxes.
- Gaming consoles and handheld gaming devices.

4.2. Industrial Automation

- Programmable Logic Controllers (PLCs) for factory automation.
- Supervisory Control and Data Acquisition (SCADA) systems.
- Robotics and automated manufacturing systems.
- Process control systems for chemical and petrochemical plants.

4.3. Medical Devices

- Infusion pumps for drug delivery.
- X-ray machines and medical imaging equipment.
- Monitoring and diagnostic devices.

4.4. Telecommunications

- Network routers and switches.
- Base stations for cellular networks.
- Voice over IP (VoIP) phones.
- Modems and network equipment.

4.5. Aerospace and Defence

- Avionics systems for aircraft.
- Guidance and navigation systems for spacecraft.
- Radar and missile systems.
- Unmanned aerial vehicles (UAVs) and drones.

4.6. Home Automation

- Home security and surveillance systems.
- Smart lighting and appliances.

4.7. IoT (Internet of Things)

- Connected devices like smart meters and smart locks.
- Environmental monitoring and control systems.
- Industrial IoT solutions for data collection and analysis.

4.8. Energy Management

- Smart grid systems for efficient energy distribution.
- Energy-efficient appliances and lighting.
- Solar inverters and wind turbine control systems.

4.9. Transportation

- Automatic fare collection systems in public transportation.
- Traffic signal control and management.
- Railway signalling and control systems.

4.10. Entertainment and Multimedia

- Audio and video processing systems.
- Home theater systems.
- Portable media players and e-readers.

4.11. Security and Access Control

- Biometric access control systems.
- Surveillance cameras and video analytics.

Theoretical Activity 1.1.2: Description of Microcontroller and Microprocessor

Tasks:

1: Answer the following questions:

i. Define the following terms
   a. Microprocessor.
   b. Microcontroller.
ii. Differentiate microcontroller from microprocessor.
iii. List at least 5 applications of microprocessor and microcontroller.

2: Provide the answers for the asked questions and write them on flipchart/papers.

3: Present the findings/answers to the whole class.

4: For more clarification, read the **key readings 1.1.2**.

5: In addition, ask questions where necessary.

Key readings 1.1.2.: Description microcontroller and microprocessor

1. Description of microcontroller

1.1. Definition

A **microcontroller** is an electronic device belonging to the microcomputer family. These are fabricated using the VLSI technology on a single chip. There are **microcontrollers** available in the present market with different word length starting from 4 bit, 8 bit, 64 bit to 128 bit.

In simple term, a microcontroller is a small computer that is capable for performing specific task.

1.2. Classification of microcontrollers

Microcontrollers are classified:

- based on family (manufacturer)
- Based on bits
- Based on memory
- Instruction set

1.2.1. Classification based on family

The microcontrollers can be classified according to their family. Family is usually a company or a manufacturer in which controller is fabricated. Each family have its own architecture and instruction set. Some though **8051** is famous but **PIC** and **ARM** are going more popular than 8051.

1.2.2. Classification based on bits

The bits in microcontroller are 8-bits, 16-bits and 32-bits microcontroller.

In **8-bit** microcontroller, the point when the internal bus is 8-bit then the ALU is performs the arithmetic and logic operations. The examples of 8-bit microcontrollers are Intel 8031/8051, PIC1x and Motorola MC68HC11 families.

The **16-bit** microcontroller performs greater precision and performance as compared to 8-bit. For example, 8 -bit microcontrollers can only use 8 bits, resulting in a final range of 0×00 – 0xFF (0-255) for every cycle. In contrast, 16 bit microcontrollers with its 16-bit data width has a range of 0×0000 – 0xFFFF (0-65535) for every cycle. A longer timer extreme worth can likely prove to be useful in certain applications and circuits. It can automatically operate on two 16-bit numbers. Some examples of 16-bit microcontroller are 16-bit MCUs are

extended 8051XA, PIC2x, Intel 8096 and Motorola MC68HC12 families.

The **32-bit** microcontroller uses the 32-bit instructions to perform the arithmetic and logic operations. These are used in automatically controlled devices including implantable medical devices, engine control systems, office machines, appliances and other types of embedded systems. Some examples are Intel/Atmel 251 family, PIC3x.

1.2.2.1. Classification based on memory

a. Classification according to memory Devices

According memory devices microcontroller are divided into two types:

- Embedded memory microcontroller
- External memory microcontroller

Embedded memory microcontroller: When an embedded system has a microcontroller, unit that has all the functional blocks available on a chip is called an embedded microcontroller. For example, 8051 having program & data memory, I/O ports, serial communication, counters and timers and interrupts on the chip is an embedded microcontroller.

External Memory Microcontroller: When an embedded system has a microcontroller unit that has not all the functional blocks available on a chip is called an external memory microcontroller. For example, 8031 has no program memory on the chip is an external memory microcontroller.

b. Classification According to Memory Architecture

According to memory architecture microcontroller are classified into two types:

- Harvard memory architecture microcontroller
- Princeton memory architecture microcontroller

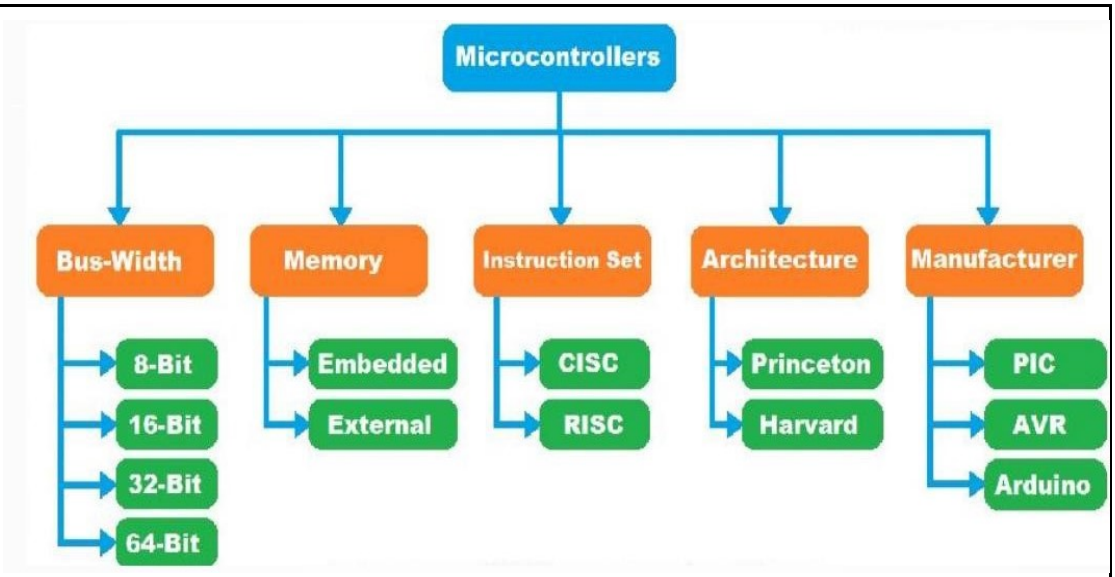
Harvard Memory Architecture Microcontroller: The point when a microcontroller unit has a dissimilar memory address space for the program and data memory, the microcontroller has Harvard memory architecture in the processor.

Princeton Memory Architecture Microcontroller: The point when a microcontroller has a common memory address for the program memory and data memory, the microcontroller has Princeton memory architecture in the processor.

1.2.2.2. Classification according to Instruction Set

CISC: CISC is a Complex Instruction Set Computer. It allows the programmer to use one instruction in place of many simpler instructions.

RISC: The RISC is stands for Reduced Instruction set Computer; this type of instruction sets reduces the design of microprocessor for industry standards. It allows each instruction to operate on any register or use any addressing mode and simultaneous access of program and data.



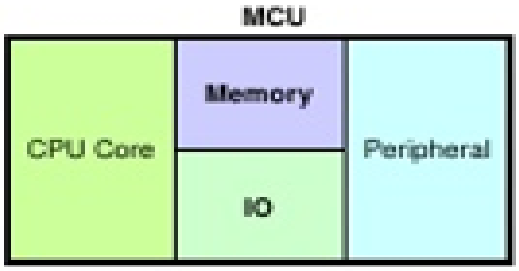
1.3. Microcontroller parts

A Microcontroller consists of the following components.

- **Central Processing Unit (CPU):** is the brain of the Microcontroller. It consists of an Arithmetic Logic Unit (ALU) and a Control Unit (CU). A CPU reads, decodes and executes instructions to perform Arithmetic, Logic and Data Transfer operations.
- **Program Memory (ROM – Read Only Memory):** is a program memory that is used for program and fixed data storage.
- **Data Memory (RAM – Random Access Memory):** is a data memory that is used for data storage.
- **Timers and Counters:** a timer is a type of clock that is used to measure time intervals. A counter is a device that records the number of times a specific event or process occurred about a clock signal.
- **I/O Ports (I/O – Input/Output):** are basic interface to control output and monitor input external events.
- **Serial Communication Interface:** The serial interface can be used to download the program and for general communication with the development PC. Serial interfaces can also communicate with external peripheral devices. Most controllers include a variety of interfaces such as SPI, SCI, PCI, USB, and Ethernet.
- **Interrupt Mechanism:** it's Interrupt Handling Mechanism. Interrupts can be external, internal, hardware related or software related.

1.4. Block diagrams of microcontroller

A block diagram of a microcontroller provides a visual representation of the major components and their interconnections within the device. In microcontroller All the components (CPU core, memory, interfaces, I/O) are embedded on the single chip.



2. Microprocessor

2.1. Definition

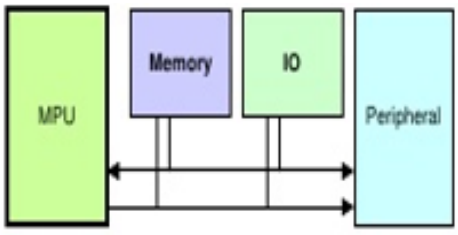
A **microprocessor** is a controlling unit of a microcomputer wrapped inside a small chip. It performs Arithmetic Logical Unit (ALU) operations and communicates with the other devices connected with it.

2.2. Microprocessor parts

- **Arithmetic and Logical Unit:** this unit is responsible for performing arithmetic tasks like addition, subtraction, multiplication, division moreover, it also makes logical decisions like greater than less than, etc. Hence the name, the 'brain' of the computer.
- **Control Unit:** this unit is responsible for looking after all the processing. It organizes and manages the execution of tasks of the CPU.
- **Registers:** The CPU directly uses these memory areas for processing. So, it's function is to store data from input or store data between calculations. Besides, it also stores the output results. Moreover, accessing registers is much faster than accessing the RAM.

2.3. Block diagrams

Microprocessor contains only the **main processor (CPU core)** on the chip and additional integrated peripherals (such as memory or I/O controllers) are connected internally.



3. Difference between Microprocessor and Microcontroller

Difference between Microprocessor and Microcontroller

Microprocessor	Microcontroller
Microprocessor is the heart of Computer system.	Micro Controller is the heart of an embedded system.
It is only a processor, so memory and I/O components need to be connected externally	Micro Controller has a processor along with internal memory and I/O components.
Memory and I/O has to be connected externally, so the circuit becomes large.	Memory and I/O are already present, and the internal circuit is small.
You can't use it in compact systems	You can use it in compact systems.
Cost of the entire system is high	Cost of the entire system is low
Due to external components, the total power consumption is high. Therefore, it is not ideal for the devices running on stored power like batteries.	As external components are low, total power consumption is less. So it can be used with devices running on stored power like batteries.
Most of the microprocessors do not have power saving features.	Most of the microcontrollers offer power-saving mode.
It is mainly used in personal computers.	It is used mainly in a washing machine, MP3 players, and embedded systems.
Microprocessor has a smaller number of registers, so more operations are memory-based.	Microcontroller has more register. Hence, the programs are easier to write.
Microprocessors are based on Von Neumann model	Micro controllers are based on Harvard architecture
It is a central processing unit on a single silicon-based integrated chip.	It is a byproduct of the development of microprocessors with a CPU along with other peripherals.

It has no RAM, ROM, Input-Output units, timers, and other peripherals on the chip.	It has a CPU along with RAM, ROM, and other peripherals embedded on a single chip.
It uses an external bus to interface to RAM, ROM, and other peripherals.	It uses an internal controlling bus.
Microprocessor-based systems can run at a very high speed because of the technology involved.	Microcontroller based systems run up to 200MHz or more depending on the architecture.
It is used for general purpose applications that allow you to handle loads of data.	It is used for application-specific systems.
It is complex and expensive, with a large number of instructions to process.	It is simple and inexpensive with less number of instructions to process.

4. Application of microcontroller and microprocessor

4.1. Application Microcontrollers

4.1.1. Consumer Electronics

- **Home Appliances:** Microcontrollers are used in washing machines, microwave ovens, and refrigerators for managing operations and controls.
- **Remote Controls:** They handle the communication between the remote and the device it controls.

4.1.2. Automotive Systems

- **Engine Control Units (ECUs):** Microcontrollers manage engine functions such as fuel injection and ignition timing.
- **Airbag Systems:** They monitor and deploy airbags in the event of a collision.

4.1.3. Industrial Automation

- **Programmable Logic Controllers (PLCs):** Used for automation of machinery and processes in factories.
- **Sensor Monitoring:** Microcontrollers process data from sensors to control machinery or other devices.

4.1.4. Medical Devices:

- **Patient Monitoring Systems:** Microcontrollers manage data collection and communication for devices like heart rate monitors.
- **Infusion Pumps:** They control the delivery of medication in a controlled manner.

4.1.5. Consumer Devices:

- **Smart Home Devices:** Microcontrollers are used in thermostats, smart locks,

and lighting systems.

- **Wearable Technology:** Fitness trackers and smartwatches use microcontrollers for data processing and communication.

4.2. Application Microprocessors

4.2.1. Computers

- **Personal Computers (PCs):** Microprocessors serve as the heart of desktop and laptop computers, managing all computing tasks.
- **Servers:** They handle large-scale data processing and run applications for businesses and services.

4.2.2. Embedded Systems

- **Automotive Infotainment Systems:** Modern vehicles use microprocessors for complex multimedia and navigation systems.
- **Networking Equipment:** Routers and switches use microprocessors to handle data traffic and network management.

4.2.3. Consumer Electronics

- **Smartphones and Tablets:** They use microprocessors for running applications, managing communication, and processing multimedia.
- **Gaming Consoles:** High-performance microprocessors handle graphics, gameplay, and system management.

4.2.4. Industrial Computers

- **Control Systems:** Used in advanced manufacturing systems for real-time control and monitoring.
- **Data Acquisition Systems:** Microprocessors manage data collection and processing from various sources.

4.2.5. Networking

- **Routers and Switches:** Microprocessors manage and route data between devices in a network.

Theoretical Activity 1.1.3: Description of embedded system memory

Tasks:

1: Answer the following questions:

i. Define the term “memory”.
ii. What are 2 types of memory used in embedded system?
iii. Differentiate 2 main types of memory used in embedded system.

2: Provide the answers for the asked questions and write them on flipchart/papers.

3: Present the findings/answers to the whole class.

4: For more clarification, read the **key readings 1.1.3.**

5: In addition, ask questions where necessary.

Key readings 1.1.3: Description of embedded system memory

1. Definition

Memory, refers to the electronic components or devices used to store and retrieve data, instructions, and information that a computer or electronic system needs to operate or process.

2. Types of memory

There are two main types of memory used in embedded system:

RAM (Random Access Memory)

ROM (Read-Only Memory)

2.1. RAM (Random Access Memory)

RAM (Random Access Memory): RAM is the primary or main memory of a computer. It is volatile memory, meaning its contents are lost when the power is turned off. RAM is used to temporarily store data and program instructions that the CPU (Central Processing Unit) needs while it is actively processing tasks. It provides fast access to data but is relatively limited in capacity compared to other storage types.

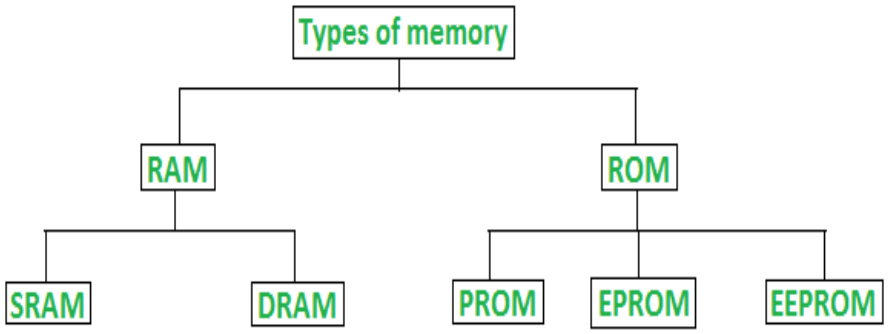
RAM is further classified into two types- SRAM (Static Random Access Memory) and DRAM (Dynamic Random Access Memory).

2.2. ROM (Read-Only Memory)

ROM (Read-Only Memory): ROM is non-volatile memory that stores firmware or software instructions that are permanently written during manufacturing. It contains the computer's initial boot-up instructions (BIOS/UEFI) and firmware for embedded systems.

ROM is further classified into PROM, EPROM, and EEPROM.

- **PROM (Programmable read-only memory)**: it can be programmed by user. Once programmed, the data and instructions in it cannot be changed.
- **EPROM (Erasable Programmable read only memory)**: It can be reprogrammed. To erase data from it, expose it to ultra violet light. To reprogram it, erase all the previous data.
- **EEPROM (Electrically erasable programmable read only memory)**: The data can be erased by applying electric field, no need of ultra violet light. We can erase only portions of the chip.



3. **Difference between two main types of memory used in embedded systems (RAM and ROM).**

RAM	**ROM**
Temporary storage	Permanent storage
Store data in MBs	Store data in GBs
Volatile	Non-volatile
Used in normal operations	Used for start-up process
Writing data is faster	Writing data is slower

Theoretical Activity 1.1.4: Description of input and output interfaces of embedded System

Tasks:

1: Answer the following questions:

i. What does the input and output interfaces mean in embedded system?
ii. List at least 4 types of input/output interfaces used in embedded system
iii. Differentiate 4 input/output interfaces used in embedded system

2: Provide the answers for the asked questions and write them on flipchart/papers.

3: Present the findings/answers to the whole class.

4: For more clarification, read the **key readings 1.1.4**.

5: In addition, ask questions where necessary.

Key readings 1.1.4: Description of Input/output (I/O) Interfaces

1. Definition

Input/output (I/O) interfaces are important components in computing systems that facilitate communication between the computer and external devices or peripherals.

These interfaces enable data transfer, control signals, and communication protocols, allowing the computer to interact with the outside world.

2. Types of input/output interfaces used in embedded system

2.1. Digital Input/output

Digital input/output (I/O) refers to the process of sending or receiving digital signals between a computing system (such as a microcontroller, microprocessor, or computer) and external devices or peripherals. Digital signals in this context are binary in nature, typically represented as either a high voltage level (often denoted as "1" or "high") or a low voltage level (often denoted as "0" or "low").

2.1.1. Digital Input (DI)

Purpose: Digital input is used for receiving binary signals or data from external devices. These signals can represent various states or conditions, such as switches being on or off, sensors detecting an event, or data transmitted from another device.

Voltage Levels: In digital input, a high voltage level typically represents logic "1," indicating an active or "on" state, while a low voltage level represents logic "0," indicating an inactive or "off" state.

Sensors and Switches: Digital input is often used to interface with sensors like motion detectors, temperature sensors, and pressure sensors, as well as with simple switches and buttons.

Signal Processing: The received digital signals can be processed by the computing system to trigger specific actions, monitor events, or make decisions based on the state of the input signals.

2.1.2. Digital Output (DO)

Purpose: Digital output is used for sending binary signals from the computing system to external devices or actuators to control their operation or state.

Voltage Levels: Digital output signals can drive external devices by toggling voltage levels between high and low. A high voltage level might activate a device, while a low voltage level might deactivate it.

Actuators and Displays: Digital output can control various devices, including LEDs, relays, motors, and displays. For example, it can turn on an LED, activate a motor to open a valve, or display binary information on an LED matrix.

Feedback and Control: Digital output can provide feedback to the computing system

about the state of controlled devices. This feedback enables closed-loop control systems, where the system adjusts its output based on the sensed state.

2.1.3. GPIO (General-Purpose Input/Output)

GPIO Pins: Many microcontrollers and microprocessors feature GPIO pins that can be configured as digital inputs or outputs. These pins provide flexibility for connecting to various external devices and sensors.

Programming: Digital input and output are often controlled and configured through programming. Software can set the state of digital output pins, read the state of digital input pins, and respond accordingly.

Protocols: Digital I/O can also involve communication protocols such as I2C, SPI, and UART, where digital signals are used for more complex data exchange between devices

2.2. Analog Input/output

Analog input and output (Analog I/O) refer to the process of sending or receiving continuous, varying voltage or current signals between a computing system (such as a microcontroller, microprocessor, or computer) and external devices or peripherals.

2.2.1. Analog Input (AI)

Purpose: Analog input is used to measure and capture continuously varying signals from the external world. These signals can represent real-world physical phenomena, such as temperature, pressure, light intensity, or voltage levels.

Voltage Levels: In analog input, the measured signals are typically represented as varying voltage levels. An analog-to-digital converter (ADC) is used to convert these analog signals into digital values that the computing system can process.

Sensors and Transducers: Analog input is commonly used to interface with sensors and transducers, such as temperature sensors (thermistors), photodetectors (photo resistors), strain gauges, and microphones that produce analog voltage signals proportional to the measured quantity.

Resolution: The quality of analog-to-digital conversion is often described in terms of resolution, which defines how finely the continuous input signal is divided into discrete digital values. Higher resolution ADCs provide more accurate representation of the analog signal.

2.2.2. Analog Output (AO)

Purpose: Analog output is used to generate continuous, varying voltage or current signals from the computing system to control external devices. These signals can be used for tasks like setting motor speeds, controlling variable voltage power supplies, or producing audio.

Voltage Levels: Analog output signals are typically specified as voltages within a specified range, often from 0 to a maximum value (e.g., 0 to 5 volts). A digital-to-analog converter (DAC) is used to convert digital values into analog voltage levels.

Actuators and Control: Analog output can be used to control various actuators,

including motors, valves, and variable voltage power supplies. It can also be used to produce audio signals for speakers or headphones.

Signal Precision: The precision and accuracy of the analog output signal depend on the quality of the DAC and its resolution. Higher-resolution DACs can generate more accurate and fine-grained analog signals.

2.3. Serial Communication Interfaces

Serial communication interfaces are interfaces that use methods for transmitting and receiving data sequentially, one bit at a time, over a single communication channel. common serial communication interfaces are:

2.3.1. UART (Universal Asynchronous Receiver/Transmitter)

Type: Asynchronous

Data Format: Start bit, data bits (typically 8 bits), optional parity bit, stop bits.

Use: Commonly used for general-purpose serial communication between microcontrollers, sensors, and peripheral devices.

Applications: Serial communication in embedded systems, serial ports on PCs, GPS modules, Bluetooth modules.

2.3.2. SPI (Serial Peripheral Interface)

Type: Synchronous

Data Format: Full-duplex communication with a master device and one or more slave devices. Typically, it uses separate data in and data out lines, along with a clock and chip-select lines.

Use: Designed for high-speed, short-distance communication between integrated circuits, often on the same circuit board.

Applications: Communication between microcontrollers and sensors, flash memory, display controllers, and other peripheral devices.

2.3.3. I2C (Inter-Integrated Circuit)

Type: Synchronous (with bidirectional data line)

Data Format: Uses a master-slave architecture with a serial data (SDA) line and a serial clock (SCL) line. Addresses are used to select specific slave devices.

Use: Suited for low-speed, short-distance communication between multiple devices on the same bus.

Applications: Sensors, EEPROMs, real-time clocks, and many other integrated circuits in embedded systems.

2.3.4. CAN (Controller Area Network)

Type: Asynchronous

Data Format: Data frames with an identifier (ID), data payload (typically 8 bytes), and error-checking bits.

Use: Designed for robust and reliable communication in noisy environments, often used in automotive and industrial applications.

Applications: In-vehicle networks, industrial automation, and control systems.

2.3.5. RS-232 (Recommended Standard 232)

Type: Asynchronous

Data Format: A widely used older serial communication standard with a voltage level range for "high" and "low" states, often using DB-9 or DB-25 connectors.

Use: Historically used for serial communication between computers and peripheral devices, though it has become less common due to newer standards.

Applications: Legacy serial communication in older computers, modems, and some industrial equipment.

2.3.6. RS-485 (Recommended Standard 485)

Type: Asynchronous or synchronous

Data Format: A balanced differential signal standard designed for long-distance communication and multiple devices on a single bus.

Use: Suitable for industrial and long-distance communication with multiple nodes on a network.

Applications: Building automation systems, industrial control networks, and remote monitoring.

2.4. Networking Interfaces

Network interfaces, also known as network adapters or network cards, are hardware components or devices that enable computers, servers, and other devices to connect to computer networks. They facilitate the exchange of data between devices on the network, allowing for communication, data sharing, and access to network resources.

Here are some common types of network interfaces:

2.4.1. Ethernet Network Interface Card (NIC)

Purpose: Ethernet NICs are used for wired Ethernet connections, enabling devices to connect to local area networks (LANs) and the internet using Ethernet cables.

Variants: Gigabit Ethernet NICs, 10 Gigabit Ethernet NICs, and more, each with varying data transfer speeds.

Applications: Commonly used in desktop computers, laptops, servers, switches, and routers.

2.4.2. Wireless Network Interface Card (Wi-Fi NIC)

Purpose: Wi-Fi NICs enable wireless connections to Wi-Fi networks, allowing devices to connect to wireless LANs and access the internet without physical cables.

Variants: Wi-Fi NICs support various Wi-Fi standards (e.g., 802.11ac, 802.11ax) with different data transfer rates and frequency bands.

Applications: Found in laptops, smartphones, tablets, wireless access points, and IoT devices.

2.4.3. Bluetooth Adapter

Purpose: Bluetooth adapters enable short-range wireless communication between devices, such as smartphones, headsets, keyboards, and mice.

Applications: Used in laptops, desktops, and various consumer electronics for connecting Bluetooth peripherals.

2.4.4. Fiber Optic Network Interface Card (Fiber NIC):

Purpose: Fiber NICs are used for high-speed, long-distance data transmission over optical fiber networks.

Variants: Single-mode and multi-mode fiber NICs, supporting different types of optical connectors.

Applications: Commonly used in data centers, enterprise networks, and telecommunications infrastructure.

2.4.5. Modem (Modulator-Demodulator)

Purpose: Modems are used to convert digital data from a computer into analog signals for transmission over telephone lines (DSL: digital subscriber line modem) or to convert digital data into cable signals for cable internet (cable modem).

Applications: Used in homes and businesses for broadband internet access.

2.4.6. Cellular Modem (Cellular Network Interface)

Purpose: Cellular modems connect to cellular networks (e.g., 3G, 4G, and 5G) and provide mobile internet access to devices.

Applications: Used in laptops, tablets, and IoT devices for mobile connectivity.

2.4.7. Network-on-Chip (NoC) Interface

Purpose: Network-on-Chip interfaces are used in System-on-Chip (SoC) designs to enable communication between different processing cores and subsystems on a single chip.

Applications: Found in complex SoCs used in smartphones, IoT devices, and embedded systems.

2.4.8. Virtual Network Interfaces (Virtual NICs)

Purpose: Virtual NICs are software-based network interfaces used in virtualization environments to connect virtual machines (VMs) to the physical network.

Applications: Common in data centers and cloud computing environments where virtualization is prevalent.

Theoretical Activity 1.1.5: Description of power supply and peripheral requirements of embedded System

Tasks:

1: Answer the following questions:

i. Explain the following terms used in embedded system.
   a. power supply
   b. peripherals

ii. List at least 4 peripheral required in embedded system.

iii. By Referring to question (2), Explain the functions of those peripherals required in embedded system

2: Provide the answers for the asked questions and write them on flipchart/papers.

3: Present the findings/answers to the whole class.

4: For more clarification, read the **key readings 1.1.5**.

5: In addition, ask questions where necessary.

Key readings 1.1.5.: Description of power supply and peripheral requirements of embedded system

Description of power supply and peripheral requirements of embedded system

1. **Power supply**

   A power supply is the component or system that provides the necessary electrical energy to power the embedded device.

   Power supply is a part responsible for providing the necessary electrical power to the embedded system. These components include voltage regulators, batteries, or external power sources. Different types of voltage regulators, such as LDO (Low Drop-Out), Buck converters, Boost converters, and Buck-Boost converters are used based on the power supply requirements of the embedded hardware.

2. **peripheral requirements of embedded system**

   Peripherals in an embedded system refer to external devices or components that are connected to the main processing unit to extend its capabilities. these could include:

   2.1.1. Timers

   Timers are essential hardware or software components in computing systems that are used to measure time intervals, generate precise timing signals, or trigger specific actions at predefined time intervals.

2.1.2. Types of Timers

a. Countdown Timers

Countdown Timers are timers that start with a set initial value and decrement it until reaching zero. They are often used to implement time delays or countdowns.

b. Interval Timers

Interval timers are timers that trigger actions at predefined intervals. They are commonly used for tasks like periodic data sampling, system maintenance, or updating displays.

PWM (Pulse Width Modulation) Timers: PWM timers generate pulse-width modulated signals, which are used for tasks like controlling motor speed, adjusting brightness in displays, and generating analog-like signals

c. Real time clock

Real time clock is a hardware component or integrated circuit used in computing and electronics to keep track of the current time and date, independently of the main system's central processing unit (CPU). RTCs are designed to provide accurate and continuous timekeeping, even when the device is powered off or in a low-power state.

2.2. Counters

Counters are digital devices or circuits that are used to count events or occurrences of specific signals, pulses, or transitions.

2.2.1. Basic Characteristics of counters

a. Event counting

Counters are designed to increment or decrement a digital value in response to specific events. These events could be clock pulses, input signal edges, or other triggering conditions.

b. Counting modes

Counters can operate in various counting modes, including up-counting (incrementing), down-counting (decrementing), and bidirectional (counting in both directions).

c. Count width

The number of bits in a counter determines its count range. For example, an 8-bit counter can count from 0 to 255 in binary.

d. Clock inputs

Most counters have clock inputs that control the rate at which they count. Counters can be synchronous, where they respond to an external clock signal, or asynchronous, where they respond to individual input pulses or transitions.

2.3. PWM controllers

PWM controllers, or Pulse Width Modulation controllers, are electronic devices or circuits used to generate Pulse Width Modulation signals.

2.3.1. Basic operation of PWM controllers

- **Pulse width modulation (PWM):** PWM is a modulation technique that varies the duty cycle (the ratio of the time the signal is high to the total period) of a square wave to control the average voltage or current supplied to a load.
- **Duty cycle:** The duty cycle is expressed as a percentage and represents the fraction of time during which the PWM signal is "on" (high). A higher duty cycle corresponds to a higher average voltage or current.
- **Frequency:** PWM controllers operate at a specific frequency, which determines how quickly the PWM signal repeats its cycle. The choice of frequency depends on the application and the requirements of the load.

2.3.2. Applications PWM controllers

- **Motor speed control**

PWM controllers are commonly used to control the speed of DC motors and fans by varying the voltage applied to them. Higher duty cycles result in higher speeds.

- **LED brightness control**

PWM is used to adjust the brightness of LEDs in displays, lighting, and indicators. Higher duty cycles make the LEDs appear brighter.

- **Voltage regulation**

Some power supplies use PWM control to regulate the output voltage, ensuring a stable voltage level for electronic devices.

- **Temperature control**

PWM controllers can be used to control heaters and thermoelectric coolers by varying the power supplied to maintain a desired temperature.

- **Audio amplifiers**

Class D audio amplifiers often use PWM techniques to efficiently amplify audio signals while minimizing power dissipation.

2.4. DMA Controllers

DMA (Direct Memory Access) controllers are specialized hardware components or subsystems in a computer or microcontroller system that facilitate efficient data transfers between peripheral devices and memory (typically RAM) without involving the CPU (Central Processing Unit).

2.4.1. Basic Operation of DMA controller

- **Data transfer:** DMA controllers enable data transfers between various peripheral devices (e.g., I/O ports, storage devices, network interfaces) and memory without CPU intervention. Either the CPU or the peripherals can initiate these transfers.
- **Memory access:** DMA controllers access the system memory directly to read or write data. This access is typically performed using the system bus or memory-mapped I/O.
- **Address generation:** DMA controllers generate memory addresses for data transfer operations. They can increment or decrement memory addresses

automatically, allowing for sequential or block data transfers.

2.4.2. Advantages of DMA Controllers

- **Reduced CPU overhead**

DMA controllers significantly reduce the CPU's involvement in data transfer operations. This results in lower CPU utilization and frees up processing power for other tasks, improving overall system efficiency.

- **Faster data transfer**

DMA transfers data at high speeds since the CPU is not involved in the process. This is particularly beneficial for real-time and high-throughput applications.

- **Improved system responsiveness**

By offloading data transfer tasks to the DMA controller, the CPU can focus on more critical tasks, leading to improved system responsiveness.

2.4.3. Applications

- **Storage devices**

DMA controllers are commonly used in storage interfaces like SATA (Serial Advanced Technology Attachment) and IDE (Integrated Drive Electronics) to transfer data between hard drives or solid-state drives and system memory.

- **Networking**

Network interface cards (NICs) often incorporate DMA controllers to handle data packets efficiently, reducing CPU overhead during data transmission and reception.

- **Audio and video processing**

In multimedia systems, DMA controllers assist in streaming and processing audio and video data, ensuring smooth playback and recording.

- **Graphics processing**

In graphics cards, DMA is used to accelerate data transfers between video memory and system memory, improving graphics performance.

- **Data acquisition**

DMA controllers play a vital role in data acquisition systems by efficiently moving data from sensors and analog-to-digital converters to memory.

2.4.4. DMA Controller Configurations:

- **Single-channel DMA**

A single-channel DMA controller can handle one data transfer at a time. It is suitable for systems with lower data transfer requirements.

- **Multi-channel DMA**

Multi-channel DMA controllers can manage multiple data transfers simultaneously using separate DMA channels. This configuration is useful in systems with diverse data transfer needs.

- **Bus master DMA**

Bus master DMA controllers can initiate data transfers independently without

CPU intervention. They have direct access to the system bus and memory.

2.5. Watchdog timers

A Watchdog Timer (WDT) is a hardware component or feature found in many microcontrollers, embedded systems, and computer systems. Its primary purpose is to monitor the proper functioning of a system and take corrective action in case of a malfunction or system lockup.

2.5.1. Basic operation

- **Timeout mechanism:** A watchdog timer is essentially a countdown timer with a predefined timeout period. The timer is continually reset or "fed" by the system's software at regular intervals, preventing it from reaching zero.
- **Reset trigger:** If the watchdog timer ever reaches zero (i.e., the system fails to reset it within the specified timeout period), it triggers a reset or other predefined action. This action effectively restarts the system, returning it to a known, safe state.

2.5.2. Applications

- **System recovery**

Watchdog timers are used to recover a system from various faults or hangs, such as software errors, memory corruption, or hardware malfunctions. When the system becomes unresponsive, the watchdog timer initiates a reset, allowing the system to restart cleanly.

- **Real-time systems**

In real-time systems, where timely responses to external events are critical (e.g., automotive control systems, medical devices, industrial automation), watchdog timers ensure that the system remains operational.

- **Embedded systems**

Microcontroller-based embedded systems often employ watchdog timers to maintain reliability in battery-powered or remote devices. They help prevent the system from being stuck in unexpected states.

- **Critical infrastructure**

Watchdog timers are used in infrastructure systems like routers, switches, and network equipment to ensure continuous operation and minimize downtime.

2.6. Drivers

In the context of embedded systems, drivers are software components that allow the operating system or application software to interact with and control hardware peripherals.

- **Characteristics:** Drivers provide an abstraction layer that enables higher-level software to communicate with specific hardware without needing to understand the underlying hardware details.
- **Applications:** Drivers are essential for enabling the operating system to manage and control the various hardware components and peripherals in an

embedded system.

Theoretical Activity 1.1.6: Description of real time operating system

Tasks:

1: Answer the following questions:

i. What does real - time operating system (RTOS) means in embedded system hardware development?
ii. What are the components of the RTOS?
iii. What are the types of RTOS?
iv. List at least 3 features of RTOS.

2: Provide the answers for the asked questions and write them on flipchart/papers.

3: Present the findings/answers to the whole class.

4: For more clarification, read the key readings 1.1.6.

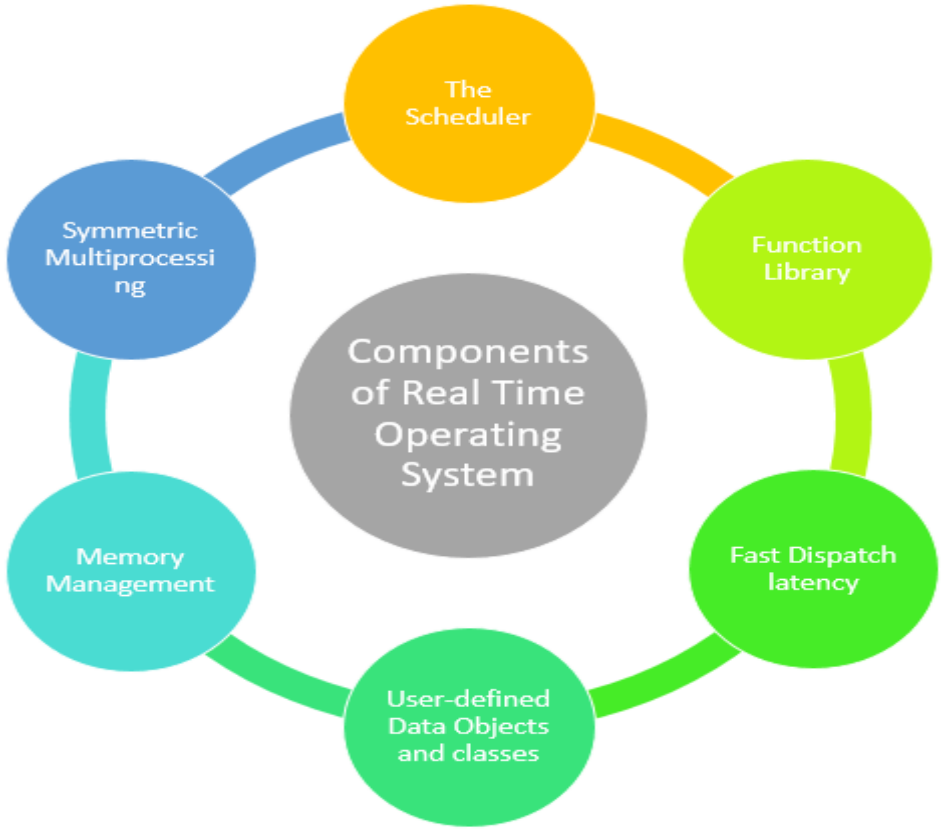
5: In addition, ask questions where necessary.

Key readings 1.1.6.: Description of real time operating system

1. Definition

A real time operating system is a software system that is designed to manage real time applications. Real time applications require immediate responses to inputs and events, and the real time OS is responsible for ensuring that these responses occur in a timely and deterministic manner.

2. Components of an real time operating system



2.1. The Scheduler

This component of RTOS tells that in which order, the tasks can be executed which is generally based on the priority.

2.2. Symmetric Multiprocessing (SMP)

It is a number of multiple different tasks that can be handled by the RTOS so that parallel processing can be done.

2.3. Function Library

Is an important element of RTOS acts as an interface that helps you to connect kernel and application code. This application allows you to send the requests to the Kernel using a function library so that the application can give the desired results.

2.4. Memory Management

This element is needed in the system to allocate memory to every program, which is the most important element of the RTOS.

2.5. Fast dispatch latency

It is an interval between the termination of the task that can be identified by the OS and the actual time taken by the thread, which is in the ready queue that has started processing.

2.6. User-defined data objects and classes

RTOS system makes use of programming languages like C or C++, which should be organized according to their operation.

3. Types of Real Time Operating Systems

There are three types of real time OS:

3.1. Hard real time OS

A hard real time operating system is a type of real-time system that guarantees that all tasks will be completed within a certain deadline, without exception. These systems are designed to provide deterministic behaviour, ensuring that critical tasks are completed on time, every time. Hard real-time systems are commonly used in applications where missed deadlines can have serious consequences, such as in aerospace and defense, medical devices, and certain types of industrial automation.

3.2. Soft real time OS

A soft real time operating system is a type of real-time system that does not guarantee that all tasks will be completed within a certain deadline. Instead, it provides the best-effort service, attempting to complete tasks as quickly as possible, but without making any guarantees about response time or deadline completion. Soft real-time systems are commonly used in applications where occasionally missed deadlines are tolerable, but overall system performance is important. Some examples of soft real-time applications include multimedia streaming, interactive gaming, and certain types of data processing.

3.3. Firm real time OS

A firm real time operating system (RTOS) is a type of real time system that guarantees that tasks will be completed within a certain deadline but with a degree of flexibility. Unlike hard real time systems that have to meet hard deadlines without exception, firm real time systems can tolerate occasional deadline misses, but they should be infrequent and not affect the overall system operation. Firm real time systems are commonly used in applications that require real-time performance but can tolerate occasional deadline misses, such as multimedia systems, gaming systems, and certain types of automation systems.

4. Features of RTOS in embedded system

4.1. Interrupt handling

In embedded systems, interrupts are signals generated by hardware or software events that require immediate attention from the CPU.

Interrupt handling is the process of temporarily stopping the current task to handle the event and then returning to the interrupted task.

4.1.1. Characteristics of Interrupt handling

Priority: Interrupts have different priority levels. Higher priority interrupts can preempt lower priority ones.

Response Time: Handling interrupts quickly is crucial, especially in real-time

systems, to ensure timely response to critical events.

Context Switching: When an interrupt occurs, the CPU saves the current context (state) of the interrupted task, processes the interrupt, and then restores the context to continue the original task.

4.1.2. Applications

Interrupt handling is used for responding to time-sensitive events such as sensor input, communication events, and hardware faults.

4.2. Error Handling

Error handling is the process of detecting, reporting, and recovering from errors that may occur during the execution of a program.

4.2.1. Characteristics of Error Handling

Error Detection: Methods for identifying errors, which may include hardware faults, software bugs, or exceptional conditions.

Error Reporting: Mechanisms to communicate the occurrence of an error to the appropriate software component or to an external system for logging or further action.

Error Recovery: Strategies to handle or mitigate the effects of errors, which may involve correcting the error, restarting the affected component, or taking other appropriate actions.

4.2.2. Applications

Error handling is essential in safety-critical systems, where failures can have serious consequences. It is also important in systems that require fault tolerance or self-healing capabilities.

4.3. Scheduling

Scheduling is the process of determining which task or process the CPU should execute at a given time.

4.3.1. Characteristics of Scheduling

Priority-Based: Tasks with higher priority levels are given precedence in execution.

Preemptive vs Non-preemptive: In preemptive scheduling, higher-priority tasks can interrupt lower-priority ones. Non-preemptive scheduling lets a task run until it voluntarily gives up the CPU.

Round-Robin Scheduling: Tasks are assigned a fixed time slice to run, ensuring fair execution.

Task Prioritization: The scheduler determines the order in which tasks are executed based on factors like priority, deadlines, and resource availability.

4.3.2. Applications

Scheduling is crucial in multi-tasking systems where multiple tasks need to be managed concurrently. It is used in a wide range of applications including real-time systems, multimedia processing, and multitasking operating systems.

- **Task Management**

RTOS handles task creation, scheduling, and priority assignment, ensuring critical tasks meet deadlines while optimizing resource usage.

- **Resource Management**

RTOS efficiently manages system resources such as CPU time, memory, and peripherals, preventing conflicts and enhancing system performance through resource sharing.

- **Memory Management**

RTOS provides memory services for dynamic allocation, preventing fragmentation and ensuring efficient use of memory resources.

- **Power Management**

Some RTOS implementations optimize energy consumption in embedded systems through power-saving modes, dynamic scaling, and sleep modes, extending battery life.

5. Examples of RTOS

Here are some RTOS examples:

- **FreeRTOS:** FreeRTOS is a popular open-source Real time OS. It is designed for microcontrollers and small embedded systems.
- **VxWorks:** VxWorks is a real time operating system developed by Wind River Systems. It is widely used in the aerospace, defense, and industrial automation industries.
- **QNX:** QNX is a commercial real time operating system developed by BlackBerry. It is used in mission-critical applications such as automotive, medical devices, and nuclear power plants.
- **ThreadX:** ThreadX is a real time operating system developed by Express Logic. It is widely used in consumer electronics, medical devices, and automotive applications.
- **Nucleus RTOS:** Nucleus RTOS is a real time operating system developed by Mentor Graphics. It is used in a wide range of applications, including consumer electronics, medical devices, and automotive systems.

Theoretical Activity 1.1.7: Description of application scope for embedded system

Tasks:

1: Answer the following questions:

i. What do you understand about the term "Determination of the Application Scope"?

ii. What are the criteria should be considered when determining the application scope of an embedded system?

2: Provide the answers for the asked questions and write them on flipchart/papers.

3: Present the findings/answers to the whole class.

4: For more clarification, read the key readings 1.1.7.

5: In addition, ask questions where necessary.

Key readings 1.1.7.: Description of application scope for embedded system

1. Introduction

Determination of the Application Scope in an embedded system means defining the system's goals, functions, target users, and operational environment. It also involves setting performance requirements and understanding constraints like power consumption, cost, and integration with other systems to ensure the system meets its purpose effectively.

2. Criteria to consider when determining the application scope of an embedded system

2.1. Gathering Information about the Application

Involves collecting detailed information about the environment, tasks, and requirements of the embedded system. This can help us for:

- Understanding user needs and expectations.
- Analysing the context in which the system will operate.
- Identifying stakeholders and their roles.

2.2. Functional Requirements

Functional requirements specify what the embedded system must do in terms of specific tasks, operations, and functionalities. This may include

- **Define Core Functions:** What are the essential operations the embedded system must perform?
- **User Needs:** What features or capabilities are required by the end-users?
- **System Inputs/Outputs:** What sensors, actuators, and interfaces are needed?

2.3. Performance

Performance requirements define the expected behaviour of the system in terms of speed, responsiveness, and efficiency. It include:

- **Processing speed:** What level of CPU or GPU performance is required?
- **Memory:** How much RAM and storage are necessary?
- **Speed and Responsiveness:** What are the acceptable latency and throughput requirements?

2.4. Environmental Constraints

Environmental constraints include factors related to the physical surroundings in which the embedded system will operate. It include:

- **Temperature Range:** What are the operating and storage temperature conditions?
- **Humidity and Dust:** Will the system be exposed to high humidity or dust?
- **Vibration and Shock:** Will the system be subject to mechanical stress or vibration?

2.5. Power Requirements

Power requirements involve specifying the energy sources, consumption levels, and power management strategies of the embedded system. It includes:

- **Power Source:** What type of power supply is used (battery, AC, etc.)?
- **Power Consumption:** What is the expected power draw, and are there power-saving modes needed?
- **Battery Life:** For battery-operated systems, what is the desired battery life?

2.6. Communication Requirements

Communication requirements define how the embedded system will interact with other devices or systems, including protocols, data formats, and bandwidth requirements. It include:

- **Interfaces:** What types of communication (e.g., serial, USB, Ethernet, Wi-Fi) are needed?
- **Protocols:** What communication protocols (e.g., MQTT, HTTP, and Bluetooth) are required?
- **Data Rates:** What is the required bandwidth for data transmission?

2.7. Safety and Regulatory standards

Safety and regulatory standards include legal and industry specific requirements that the embedded system must adhere to. It include:

- **Compliance:** Which industry standards (e.g., ISO, IEC, CE, and UL) must be adhered to?
- **Safety Features:** What safety mechanisms are needed to protect users and the system

2.8. User Interface Requirements

User interface requirements define how users interact with the embedded system, including elements like displays, buttons, touchscreens, and input methods. It include:

- **Input Methods:** What input methods (e.g., buttons, touchscreens) are needed?
- **Output Display:** What type of display or output mechanism is required (e.g., LED, LCD)?
- **Usability:** How should the interface be designed for ease of use?

2.9. Scalability and Future Expansion Constraints

Scalability requirements consider whether the system can handle expansion in terms of data volume, users, or features, and the constraints that may limit scalability. It include:

- **Modular Design:** Can the system be easily upgraded or expanded?
- **Compatibility:** How will future updates or expansions integrate with the existing system?

2.10. Cost and Budget Constraints

Cost and budget constraints involve considerations related to the financial aspects of developing and producing the embedded system. This include:

- **Development Budget:** What is the allocated budget for development and production?
- **Cost of Components:** What are the cost constraints for individual components?
- **Manufacturing Costs:** What are the expected costs for manufacturing and assembly?

Practical Activity 1.1.8: Determining the application scope for embedded system

Task:

1: Read key reading 1.1.8

2: Referring to key reading 1.1.8, you are sked to determine application scope for embedded system.

3: Present your work to the trainer and whole class.

4: Ask clarification where necessary.

Key readings 1.1.8: Determining the application scope for embedded system

1. Steps of determining the application scope for embedded system

1.1. Determine information about the Application.

- **Understand the Purpose:** Identify what the system is designed to accomplish. Is it for controlling a specific hardware component, processing data, or monitoring a process?
- **Identify Stakeholders:** Understand whom the end-users, operators, or stakeholders are, and gather their expectations.
- **Research Similar Systems:** Study existing systems or solutions to understand industry standards, common practices, and available technologies.

1.2. Determine information about Functional Requirements

- **Core Functions:** List the essential tasks the embedded system must perform. This could include tasks like monitoring sensors, controlling motors, managing inputs/outputs, and performing specific calculations.
- **User-Driven Features:** Define additional features that enhance user experience, such as data logging, remote control, or feedback mechanisms.

1.3. Determine information about Performance

- **Throughput:** Determine how much data or how many operations the system needs to process in a given time frame.
- **Real-Time Processing:** Identify any real-time constraints, such as tasks that must be completed within specific time limits.
- **Latency:** Specify acceptable response times for the system, particularly in time-critical applications.
- **Reliability and Uptime:** Establish the expected reliability, including uptime and error tolerance for critical operations.

1.4. Determine information about Environmental Constraints

- **Operating Temperature:** Define the temperature range within which the system must operate reliably.
- **Humidity and Moisture:** Specify whether the system must operate in environments with high humidity, condensation, or exposure to water.
- **Vibration and Shock:** Assess whether the system will be subject to mechanical stress, such as vibrations or impacts, and design for durability accordingly.
- **EMI (Electromagnetic Interference):** Ensure the system is protected from or designed to minimize interference from electromagnetic fields, especially for systems in industrial or medical environments.

1.5. Determine information about Power Requirements

- **Power Source:** Identify the system's power source, such as battery, solar, or mains power, and assess the availability of the power supply.

- **Energy Efficiency:** Set goals for minimizing power consumption, especially for battery-powered or portable systems.
- **Low-Power Modes:** Plan for power-saving modes (e.g., sleep or idle) to conserve energy when full operation is not required.

1.6. Determine information about Communication Requirements

- **Communication Protocols:** Identify the protocols the system will use to communicate with other devices (e.g., SPI, I2C, UART, Ethernet, Wi-Fi, Bluetooth).
- **Data Transfer Rate:** Determine the required speed for data transmission and any constraints on bandwidth.
- **Wired vs. Wireless:** Decide whether the system will use wired or wireless communication, depending on the application environment and mobility needs.
- **Inter-System Communication:** Define how the embedded system will interact with other systems, such as cloud servers, external sensors, or control systems.

1.7. Determine information about Safety and Regulatory Standards

- **Safety Standards:** Identify any industry-specific safety standards the system must comply with, such as ISO 26262 for automotive or IEC 61508 for industrial safety.
- **Fail-Safe Mechanisms:** Design fail-safe features to ensure the system behaves safely in the event of a failure (e.g., emergency shutdowns, watchdog timers).
- **Certifications and Compliance:** Plan for obtaining necessary certifications (e.g., CE, FCC, UL) to ensure regulatory compliance for the target market or environment.

1.8. Determine information about User Interface Requirements

- **Input/output Mechanisms:** Define how users will interact with the system, such as through buttons, touchscreens, keypads, or remote control interfaces.
- **Display Requirements:** Specify whether the system will need a display (e.g., LCD, OLED) to provide feedback or control, and the complexity of the interface (e.g., simple readout vs. graphical user interface).
- **Accessibility:** Consider accessibility features such as visual indicators, audio feedback, or customizable controls for different user groups.

1.9. Determine information about Scalability and Future Expansion Constraints

- **Modular Design:** Design the system to be modular, allowing for future upgrades or the addition of new features without major redesign.
- **Hardware Upgrades**: Ensure the hardware platform can support future needs (e.g., additional sensors, communication interfaces, or processing power).
- **Software Extensibility:** Plan for software updates and enhancements, ensuring

the system is flexible enough to accommodate new protocols, algorithms, or functionalities.

- **Storage and Processing Capabilities:** Ensure there is sufficient memory and processing power to handle potential future tasks or data loads.

1.10. Determine information about Cost and Budget Constraints

- **Development Costs:** Estimate the budget for designing and developing the embedded system, including hardware, software, and labor.
- **Component Costs:** Assess the cost of key components (e.g., processors, sensors, communication modules) to ensure they fit within the project's budget.
- **Production Costs:** Consider the cost of mass production, including materials, assembly, and testing.
- **Maintenance Costs:** Plan for the cost of maintaining the system over its lifecycle, including software updates, hardware replacements, and technical support.
- **Return on investment (ROI):** Evaluate the system's ROI, ensuring that the cost is justified by the value the system will bring to its users or stakeholders.

2. Conduct risk analysis

2.1. Definition

Conducting a risk analysis is a systematic process of identifying, assessing, and mitigating potential risks associated with a project or system.

In the context of embedded systems, this involves evaluating potential hazards and vulnerabilities that may arise during the development, deployment, and operation of the system.

2.2. Importance of conducting risk analysis

- It helps in recognizing possible issues or dangers that could affect the embedded system's performance, reliability, or safety.
- Risk analysis provides a roadmap for implementing measures to reduce or eliminate identified risks.
- Enhances system, it helps ensure safety and reliability: By addressing potential issues proactively the overall sturdiness and dependability of the embedded system.
- Identify potential risks, which could include hardware or software failures, environmental factors, regulatory compliance issues, and more.
- Implement measures to reduce or eliminate identified risks, which may involve design changes, additional safety features, or process adjustments.

2.3. Steps of conducting risk analysing for embedded system hardware development

2.3.1. Identify risks

- **Systematic brainstorming**: Engage stakeholders (engineers, developers, and

end-users) to identify potential risks at each stage of hardware development.

- **Categorize risks**: Focus on risks related to hardware components, design processes, environmental factors, regulatory compliance, and manufacturing.
    - **Component risks**: Unreliable or obsolete components, supply chain issues, or component failure.
    - **Design risks**: Design errors, inadequate testing, or integration issues with embedded systems.
    - **Environmental risks**: Temperature, humidity, electromagnetic interference (EMI), and mechanical stress.
    - **Manufacturing risks**: Production defects, quality control failures, or delays.

2.3.2. Assess risk possibility and impact

- **Risk possibility**: Estimate the probability of each risk occurring (e.g., low, medium, high).
- **Risk impact**: Determine the severity of each risk on the project's success (e.g., minor, moderate, major).
- **Risk matrix**: Create a risk matrix to prioritize risks based on likelihood and impact. High-priority risks should be addressed first.

2.3.3. Analyze risk mitigation strategies

- **Preventive actions**: Identify ways to prevent risks from occurring, such as:
    - **Redundancy**: Add backup components or systems to reduce the impact of component failure.
    - **Design verification**: Conduct thorough design reviews and validation to catch potential errors early.
    - **Supplier reliability**: Work with reputable suppliers for quality components.
- **Corrective actions**: Define steps to take if a risk materializes:
    - **Contingency planning**: Develop backup plans for critical risks, such as delays in component supply or hardware failures.
    - **Component Substitution**: Plan for alternate components if specific ones are unavailable.

2.3.4. Develop and implement risk control measures

- **Design testing protocols**: Implement extensive testing, including:
    - **Prototyping and simulation**: Test the hardware design through simulations before production.
    - **Hardware in the loop (HIL)**: Test the interaction between software and hardware.
    - **Environmental testing**: Ensure the system can operate under the required environmental conditions (e.g., temperature, vibration).
- **Safety and regulatory compliance**: Verify the hardware design complies with

safety standards and certifications (e.g., CE marking, UL certification).

2.3.5. Monitor and review risks

- **Ongoing monitoring**: Continuously track identified risks throughout the project life cycle. Ensure any changes in the development process are assessed for new risks.
- **Regular reviews**: Hold risk review meetings periodically to assess if any new risks have emerged or if previously identified risks require updates.
- **Risk documentation**: Maintain a risk log to document all identified risks, mitigation strategies, and any incidents.

2.3.6. Communicate and report

- **Stakeholder communication**: Regularly update stakeholders on the risk status, including mitigation progress and any changes in the risk profile.
- **Reporting**: Provide reports on risk analysis findings and the effectiveness of mitigation efforts, which may be required for compliance or quality assurance.

Theoretical Activity 1.1.9: Description of enclosure and physical interface for embedded system

Tasks:

1: Answer the following questions:

i. Explain what does enclosure means in embedded system.
ii. What do you mean by the term “physical interface” in embedded system?
iii. Explain the importance of having enclosure and physical interface when developing hardware embedded system?

2: Provide the answers for the asked questions and write them on flipchart/papers.

3: Present the findings/answers to the whole class.

4: For more clarification, read the key readings 1.1.9.

5: In addition, ask questions where necessary.

Key readings 1.1.9: Description of enclosure and physical interface for embedded system

1. **Description Enclosure and Physical Interface**

2.1. Definition

- In the context of embedded systems, the enclosure refers to the physical casing or housing that contains the electronic components.
- The physical interface refers to the points of interaction between the embedded system and the external environment.

2.2. Importance of enclosure

- **Protection:** It shields the internal components from environmental factors like dust, moisture, and physical damage.
- **Safety:** Prevents user contact with potentially hazardous components (e.g., high-voltage circuits).
- **Aesthetics:** Provides a presentable and professional appearance for the system.

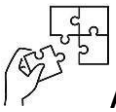
2.3. Importance of physical interface

- **Connectivity:** Defines how the embedded system interacts with external components, including sensors, actuators, and communication modules.
- **User Interaction:** Includes interfaces like buttons, touchscreens, or ports for user input and output

Points to Remember

- Embedded System is defined as an integrated system that is formed as a combination of computer hardware and software for a specific function.
- There are 3 main components of embedded system, which are Hardware, Software and Firmware.
- They are some different applications of embedded system such as Consumer Electronics, Industrial Automation, Medical Devices, Telecommunications, and Home Automation.
- A microprocessor is defined as a controlling unit of a microcomputer wrapped inside a small chip while a microcontroller is defined as an electronic device belonging to the microcomputer family.
- Microcontrollers and microprocessors are differentiated according to their physical images, block diagram, applications and power consumption.
- Microcontroller are mainly used in embedded Systems while microprocessors are mainly used in personal computers.
- Memory refers to the electronic components or devices used to store and retrieve data, instructions, and information that a computer or electronic system needs to operate or process.
- There are two main types of memory used in embedded system which are: RAM (Random Access Memory) and ROM (Read-Only Memory)
- In embedded system, Input/output (I/O) interfaces are defined as components in that facilitate communication between the embedded system and external devices or peripherals.
- There are different types of I/O interfaces used in embedded systems including digital input and output interface, analog input/output interface, serial Communication Interfaces and networking interfaces

- A power supply is a part or system that provides the necessary electrical energy to power the embedded device. It converts the input voltage (which could be from a battery, mains electricity, or another source) to the correct voltage and current needed by the embedded system components to function properly.
- A power supply is the component or system that provides the necessary electrical energy to power the embedded device.
- Peripherals in an embedded system refer to external devices or components that are connected to the main processing unit to extend its capabilities.
- The common types of peripherals required in embedded system are timers, Counters, PWM controllers, DMA Controllers and drivers.
- A Real-Time Operating System (RTOS) is defined as a specialized operating system designed for applications that require deterministic and predictable response times to events.
- RTOS has different features which are interrupt handling, error handling and scheduling
- When Determining application scope for embedded system, follow the following steps:
    - Determine information about the application
    - Determine information about functional Requirements
    - Determine information about performance
    - Determine information about environmental constraints
    - Determine information about power requirements
    - Determine information about communication requirements
    - Determine information about safety and regulatory standards
    - Determine information about user interface requirements
    - Determine information about scalability and future expansion constraints
    - Determine information about cost and budget constraints
- In the context of embedded systems, the enclosure refers to the physical casing or housing that contains the electronic components.
- The physical interface refers to the points of interaction between the embedded system and the external environment.



Application of learning 1.1.

RTY Solutions is a company that provides different electronic services like to automate various home functions, including lighting, security, climate control, and appliance management. You have been assigned to determine the application scope for an embedded system that will power an automatic smart home control system.

Indicative content 1.2: Identification of tools, materials and equipment

Duration: 3 hrs

Theoretical Activity 1.2.1: Description of tools, materials and equipment for embedded System

Tasks:

1: Answer the following questions:

i. Define the following terms used in embedded system hardware development:
   a. Tool
   b. Material
   c. Equipment

ii. List at 3 least examples of tools. Materials and equipment used when developing embedded system hardware.

iii. By refer to the question (ii), Give the uses of those examples.

2: Provide the answers for the asked questions and write them on flipchart/papers.

3: Present the findings/answers to the whole class.

4: For more clarification, read the key readings 1.2.1.

5: In addition, ask questions where necessary.

Key readings 1.2.1.: Description of tools, materials and equipment for embedded system

1. Definitions

1.1. Tool

A tool is a device that is designed to perform a specific task or function, typically by making physical operation.

1.2. Material

Material refers to a substance or matter that is used to create or construct physical objects

1.3. Equipment

Equipment refer to necessary set of tools required to do a particular function.

2. Examples of tools, Materials and equipment used when developing embedded system hardware.

2.1. Tools

- **Soldering iron:** is a tool used for melting solder to join or repair electrical components, wires, or circuits.

- **Wire cutter:** is a hand tool designed for cutting wires or cables.

- **Wire stripper:** is hand tools used to remove the insulation or outer covering from electrical wires

- **Tweezers:** are small, handheld tools with pointed tips that are used for picking up and manipulating small objects, particularly in tasks that require precision.

- **Screwdrivers:** are a hand tool designed for turning screws by applying torque to their head, allowing them to be driven into or removed from a material.

- **Pliers:** hand tools used for twisting, cutting, and manipulating various materials, such as wires, cables, and small objects.

- **Scissors:** are hand-operated cutting tools with two sharp blades that are used for cutting various materials, such as paper, fabric, plastic, and more.

- **Crimping tool:** is a hand tool used to create a permanent electrical or mechanical connection between two pieces of metal, typically wires, by deforming or compressing them together.

- **ESD-safe mats:** are specially designed mats used to prevent electrostatic discharge in environments where sensitive electronic components and devices are handled or assembled.

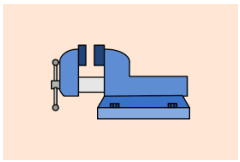
- **Wrist straps:** are wearable devices used to protect sensitive electronic components and equipment from electrostatic discharge**.**

- **ESD-safe brushes:** safe brushes are specialized brushes designed to prevent electrostatic discharge and damage to sensitive electronic components during cleaning, maintenance, and assembly processes.

- **Logic probes:** is a handheld test tool used to determine the logic state (high or low) of digital signals in electronic circuits.

- **Test leads:** are flexible, insulated wires with connectors at one end. These leads are used to connect test and measurement instruments (e.g., multimeters, oscilloscopes) to various points in a circuit for voltage, current, or resistance measurements.

- **Bench vises:** is a mechanical device that is typically mounted on a workbench or a sturdy surface. It is used to securely hold and immobilize workpieces while performing tasks like cutting, drilling, filing, or soldering.



- **PCB holder:** is a specialized tool used in electronics and PCB assembly. Its primary purpose is to securely hold and position PCBs during soldering, desoldering, assembly, and inspection processes.

- **Third hand tool:** is a hand device used in electronics and soldering work to hold and position small components, wires, or circuit boards.

- **Magnifying glass or loupe** is a simple optical instrument used to enlarge and enhance the visual details of small objects or fine print.

- **Desoldering pump or solder wick** is a hand-held tool used for removing solder from electronic circuits or connections.

- **Breadboards or prototyping boards:** is a tool used in electronics and electrical engineering for building and testing electronic circuits without the need for soldering

- **Wire wrapping tool:** is a specialized hand tool used in electronics and telecommunications for creating reliable and secure electrical connections between wires and electronic components**.**

- **PCB design software:** is software used to create and layout electronic circuit schematics and design the physical PCB for manufacturing.
- **CAD software for mechanical design:** is a type of software used by engineers and designers to create, modify, and analyze detailed 2D and 3D models of mechanical parts and assemblies.
- **Flying probe testers:** are specialized automated testing devices used in the electronics industry to perform testing and inspections on printed circuit boards (PCBs) and electronic assemblies.
- **Vibration testers:** is a piece of equipment used to assess and evaluate the vibration and mechanical behavior of products, components, and structures.

- **Ultrasonic cleaner:** is a device that uses high-frequency sound waves, typically beyond the range of human hearing, to clean various items, including jewellery, dental instruments, electronic components, and small mechanical parts.

2.2. Materials

- **Electronic components:** are basic electronic parts used in the construction of embedded systems.

Examples: Resistors, capacitors, diode, integrated circuits, transistors.

- **PCB copper clad** Boards on which electronic components are mounted and with conductive traces.

- **Enclosures and Casings:** Physical housings that protect the internal components of the system.

Examples: Plastic or metal enclosures, waterproof casings.

- **Power Sources:** Components or devices that provide electrical power to the system.

Examples: Batteries, power adapters, voltage regulators.

- **Connectors and Cables:** are materials that facilitate the interconnection of various components within the embedded system.

Examples: Headers, connectors, wires, ribbon cables.

- **Semiconductor Materials**: Materials used in the fabrication of electronic components and integrated circuits.

Examples: Silicon wafers, gallium arsenide

- **Sensors:** are devices or instruments that detect and measure physical properties or changes in the environment and convert this information into an electrical or digital signal.

- **Actuators:** are devices or components that are used to convert various forms of energy into mechanical motion or physical movement.

- **Insulating Materials:** are substances that are used to impede the flow of electricity.

- **Thermal Interface Materials:** are substances or compounds used to enhance the thermal conductivity between two surfaces in contact with each other, typically to improve heat transfer between electronic components and heat sinks or other cooling systems.

- **Adhesives and Sealants:** Adhesives and sealants are substances used to bond or seal various materials together in a wide range of applications.

- **Heat Sink Materials:** are passive cooling components used to dissipate heat generated by electronic devices, such as CPUs, GPUs, power transistors, and other integrated circuits.

- **Display Materials** are substances or technologies used to create visual information on screens, panels, or surfaces.
- **Wires:** are conductive materials, typically made of metal, that are used to transmit electrical signals or carry electrical current from one point to another.

- **Glue:** is a type of adhesive substance used to bond or join materials together.

- **Soldering tin:** is a fusible metal alloy used to join or bond other metals together by melting the solder and creating a strong, electrical conductive connection as it cools and solidifies.

- **Soldering paste:** is a material used in electronics assembly and soldering processes to facilitate the attachment of electronic components to printed circuit boards (PCBs).

2.3. Equipment

- **Oscilloscopes:** is measuring instruments used to visualize and analyse electronic signals.
- **Logic analysers:** is instrument used for capturing, displaying, and analysing digital signals in embedded systems.
- **Spectrum analysers:** is instruments used to measure and analyse the frequency spectrum of signals.
- **Signal generators:** is an instrument that produce electrical waveforms for testing and calibration.
- **Programmers and debuggers:** things used for programming microcontrollers and debugging embedded software.
- **Environmental testing equipment:** Instruments used to simulate and test the

effects of different environmental conditions on embedded systems.

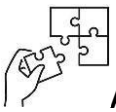
- **Microcontroller development kits (with debugger and programmer):** are hardware platforms designed to help engineers, students, and hobbyists in the development, prototyping, and testing of embedded systems based on microcontrollers or microprocessors.
- **3D printer:** A 3D printer is a type of additive manufacturing technology that creates three-dimensional objects by adding material layer by layer.
- **CNC machine**: A CNC (Computer Numerical Control) machine is a computer-controlled manufacturing tool that operates through programmed sequences of instructions to produce precision parts and components from various materials.
- **Pick-and-place machine:** is a type of automated equipment used in manufacturing, particularly in the electronics industry, for the high-speed and precise placement of electronic components onto printed circuit boards (PCBs).
- **Reflow oven:** is a specialized piece of equipment used in electronics manufacturing to solder surface-mounted components onto printed circuit boards (PCBs) or substrates.
- **Wave soldering machine:** is a specialized piece of equipment used in electronics manufacturing to solder through-hole components onto printed circuit boards (PCBs) or substrates.
- **PCB milling machine:** is a computer-controlled device designed for the precise removal of unwanted copper from a printed circuit board (PCB) to create custom PCBs.
- **ESD-safe workstations and benches:** are specially designed work surfaces and furniture used in environments where electrostatic discharge can be harmful to sensitive electronic components and devices.
- **Environmental chambers (temperature and humidity testing):** are specialized enclosures or chambers designed to simulate various environmental conditions and climates for the testing, calibration, and evaluation of products, materials, and components.
- **EMI test equipment**: Electromagnetic Interference (EMI) test equipment, also known as EMC (Electromagnetic Compatibility) test equipment, is used to assess and verify the electromagnetic compatibility of electronic and electrical devices and systems.
- **Thermal imaging camera:** is a device that captures images of objects and scenes by detecting the heat (infrared radiation) they emit.
- **Automated test equipment (ATE):** refers to computer-controlled equipment and systems used in the electronics and manufacturing industries for the automated testing and verification of electronic devices, components, and

systems.

- **In-circuit test (ICT) systems:** are specialized automated test equipment used in the electronics manufacturing industry to perform comprehensive electrical testing of populated printed circuit boards (PCBs) and electronic assemblies.
- **IC programmers (EEPROM Programmers):** is an automated or semi-automated system designed to inspect and verify the quality and integrity of PCBs used in electronics manufacturing.
- **Multimeters:** is an essential handheld electronic measurement tool used to measure various electrical and electronic parameters in electrical circuits.
- **Soldering station:** soldering station, also known as a soldering iron station or soldering rework station, is a specialized tool used for soldering and desoldering electronic components, wires, and circuits.
- **Hot Air Rework Station:** A hot air rework station, often referred to as a hot air gun or reflow station, is a specialized tool used in electronics manufacturing and repair for soldering and desoldering surface-mounted electronic components, including integrated circuits (ICs), connectors, and passive components.
- **Network analysers:** is a specialized electronic test equipment used in the field of electronics, telecommunications, and RF (radio frequency) engineering to analyze and characterize the electrical behavior of electrical networks, components, and devices at various frequencies.
- **Power analyser:** are essential tools for electrical engineers, energy auditors, maintenance professionals, and anyone responsible for managing and optimizing electrical power systems and energy consumption. They play a vital role in ensuring the quality, efficiency, and reliability of electrical power in various applications.
- **Power supplies:** A power supply, often simply referred to as a PSU (Power Supply Unit) or power source, is an electronic device that provides electrical energy to other devices or systems by converting an input power source into the appropriate voltage, current, and frequency required for their operation.

Points to Remember

- Tools are devices designed for specific tasks, like soldering irons or wire cutters.
- Materials are substances used to create objects, such as electronic components or PCB copper clads.
- Equipment includes essential tools for functions, like oscilloscopes or logic analyzers in embedded system hardware development.



Application of learning 1.2.

Makuans Solutions is a company that provide different electronic services, the engineering team is preparing to build an embedded system for a smart home device. As the lab technician, your first task is to identify and organize all necessary tools, materials, and equipment.

Indicative content 1.3: Organising the work place

Duration: 3 hrs

Theoretical Activity 1.3.1: Description of safety measures at workplace

Tasks:

1: Answer the following questions:

i. What do you understand by the term safety measures?

ii. List at least 5 general safety measures to consider when organising the workplace

iii. What are 5 arrangement techniques to consider when arranging tools, materials and equipment in a workplace?

2: Provide the answers for the asked questions and write them on flipchart/papers.

3: Present the findings/answers to the whole class.

4: For more clarification, read the key readings 1.3.1.

5: In addition, ask questions where necessary.

Key readings 1.3.1.: Description of safety measures at workplace

1. **Definition of Safety measures**

Safety measures are precautions and practices put in place to prevent accidents, injuries, or harm at workplace.

2. **General safety measures to consider:**

- **Risk Assessment:** Identify potential risks and hazards in your environment or activities.
- **Safety Equipment**: Use appropriate personal protective equipment (PPE), such as helmets, gloves, goggles depending on the situation.
- **Emergency Planning:** Develop and communicate emergency plans.
- **Fire Safety:** Install smoke detectors, fire extinguishers, and have a fire escape plan in place.
- **First Aid:** Learn basic first aid and have a well-equipped first aid kit.
- **Safe Handling of Chemicals:** Follow proper procedures for handling, storing, and disposing of chemicals, including reading labels and using appropriate protective gear.
- **Electrical Safety:** Be cautious around electrical equipment and follow safety guidelines for using and maintaining electrical devices and wiring.
- **Machine and Tool Safety**: Use machines and tools only as intended, and

always follow safety instructions.

- **Regular Inspections**: Periodically inspect and maintain equipment, structures, and safety systems to ensure they are in proper working condition.

3. Arrange tools, materials and equipment

3.1. Definition

Arrange tools, materials and equipment refer to how these items are organized and stored in workplace for efficient use, easy access, and safety.

3.2. Arrangement techniques

3.2.1. Arrangement by Type

Description: Grouping items based on their category or type, such as components, tools, or documentation.

Examples:

- **Components:** Separate area for resistors, capacitors, ICs (Integrated Circuits), connectors, and other electronic components.
- **Tools:** Organize soldering equipment, oscilloscopes, multimeters, and other testing instruments in dedicated sections.

3.2.2. Arrangement by Shape

Description: Organizing items according to their physical shape or form, which can be particularly useful for physical components and tools.

Examples:

- **Storage Boxes:** Use boxes or drawers of different shapes to match the shapes of components, like small bins for ICs and larger bins for modules.
- **Component Trays:** Arrange components in trays or organizers based on their shape, such as cylindrical components in one tray and flat components in another.

3.2.3. Arrangement by Use

Description: Organizing items based on their function or how they are used in the development process.

Examples:

- **Workstations:** Set up workstations with specific tools and components for different stages of development, such as prototyping, soldering, or

debugging.

- **Process Flow:** Arrange tools and components in the order they are used in the development workflow, from initial assembly to final testing.

3.2.4. Arrangement by Size

Description: Organizing items based on their dimensions, which helps in efficient storage and accessibility.

Examples:

- **Shelving:** Larger components and tools are stored on lower shelves or in larger bins, while smaller components are kept in drawers or on higher shelves.
- **Drawer Organizers:** Use dividers in drawers to separate items by size, such as small resistors versus larger modules.

3.2.5. Arrangement by Manufacturer Instruction

Description: Organizing items according to the guidelines or recommendations provided by the manufacturer, which is crucial for sensitive components and tools.
Examples:

- **Storage Conditions:** Store sensitive components like semiconductors and memory chips in anti-static bags or containers as per manufacturer instructions.
- **Handling and Use:** Follow specific handling instructions for delicate components or tools to ensure proper functioning and longevity.

Practical Activity 1.3.2: Organizing the Workplace

Task:

1: Read key reading 1.3.2
2: Referring to key reading 1.3.2, you are sked to organise the workplace.
3: Present your work to the trainer and whole class
4: Ask clarification where necessary.

Key readings 1.3.2.: Organizing the Workplace

Steps to organize the workplace before building embedded system hardware

1. **Workplace Preparation**

- Remove any unnecessary items, tools, or clutter from the work area.
- Ensure that only essential tools and components are present.

2. **Organize Tools and Components**

- Arrange tools and components in an orderly manner, grouping similar items together.
- Use labelled containers or trays for small components like resistors, capacitors, and ICs.

3. **Establish a Workbench Layout**

- Designate specific areas for soldering, testing, and assembling components.
- Ensure that frequently used tools are easily accessible.

4. **Apply General Safety Measures**

- Verify that all electrical outlets and power supplies are properly grounded.
- Use surge protectors and ensure that power cords are in good condition.
- Keep liquids away from electrical components and tools.
- Wear safety goggles to protect your eyes from solder splashes and flying debris.
- Use anti-static wristbands to prevent electrostatic discharge (ESD) damage to sensitive components.
- Ensure proper ventilation in the workspace, especially when soldering, to avoid inhaling fumes.
- Use fume extractors or work in a well-ventilated area.

5. **Arrange the Workstation**

- Adjust the height of your workbench to a comfortable level to reduce strain.
- Use a comfortable chair with proper back support.
- Ensure the workspace is well lit, with focused lighting on the workbench.
- Use adjustable lamps for precision tasks like soldering and inspection.
- Organize and secure cables to prevent tripping hazards.
- Label cables for easy identification and avoid tangling.

6. **Check and Maintain Tools**

- Inspect all tools before use to ensure they are in good working condition.
- Replace or repair any damaged tools.
- Calibrate testing instruments like multimeters and oscilloscopes to ensure accurate measurements.

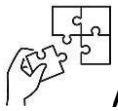
- Regularly clean the workbench and tools to remove dust and debris.
- Dispose of waste materials, such as solder scraps and packaging, in designated bins.

7. Safety Documentation and Signage

- Place safety signs around the workspace to remind everyone of potential hazards.
- Post safety procedures in a visible location, including emergency contact information and first aid instructions.
- Ensure that all personnel are trained in safety protocols and the proper use of tools and equipment.

Points to Remember

- Safety measures are precautions to prevent accidents, injuries, or harm in the workplace, including emergency planning, fire, and electrical safety.
- Proper arrangement techniques include organizing items by type, shape, use, size, and following user manual instructions.
- When organizing the workplace before building embedded system hardware, follow those steps:
    - Workplace Preparation
    - Apply General Safety Measures
    - Arrange the Workstation
    - Check and Maintain Tools
    - Safety Documentation and Signage



Application of learning 1.3.

ABC Solutions is a company that provide different electronic services, the engineering team is preparing to develop advanced embedded system hardware for a smart home device. As a technician trained in workplace organization, your responsibility is to prepare the electronics lab before the hardware development phase. This includes cleaning the workspace, organizing tools and components, and ensuring all safety measures are properly in place.

Learning outcome 1 end assessment

Theoretical assessment

1. Choose the correct answer
    a. Safety measures are defined as precautions and practices to put in place to prevent accidents, injuries, or harm at workplace.
    b. Safety measures are defined as precautions and practices to put in place to measure accidents, injuries, or harm at workplace.
    c. Safety measures are defined as precautions and practices to put in place to destroy the workplace.
2. List at least four general safety measures to consider when organising the workplace.
3. Choose the correct answer:

Here's arrangement techniques to follow when you need organize the workplace EXCEPT

a) Arrangement by their type.
b) Arrangement by their shape.
c) Arrangement by their customer need
d) Arrangement by their size.
e) Arrangement by the formula

4. What are Steps required to organize the work place?
5. Answer the following question by TRUE or FALSE
    i. Memory refers to the electronic components or devices used to store and retrieve data, instructions, and information that a computer or electronic system does not need to operate or process.
    ii. Embedded System is an integrated system that is formed as a combination of computer hardware and software for a specific function
6. Differentiate microcontroller from microprocessor according to their block diagram.
7. Match the column **A** of tools and equipment to its corresponding functions to the column **B**.

A (Tools and equipment)	**B (Functions)**	
1. Soldering Iron	**a.** is measuring instruments used to visualize and analyse electronic signals.	

2. Tweezers	b. are small, handheld tools with pointed tips that are used for picking up and manipulating small objects, particularly in tasks that require precision	
3. Oscilloscopes	c. is tool used for melting solder to join or repair electrical components, wires, or circuits.	
4. Power Analyzer	d. is an electronic device that provides electrical energy to other devices or systems by converting an input power source into the appropriate voltage, current, and frequency required for their operation.	
5. Power Supplies		

Practical assessment

At ABC Solutions is a company located in Kigali city that provide different electronic services, the engineering team is gearing up to develop an advanced embedded system hardware for a smart home device. As technician, your responsibility is to organize the electronics lab before the hardware development phase begins. This involves cleaning the workspace, organizing tools and components, and ensuring safety measures are in place.

END

References

All About Circuits – PCB Design. (n.d.). Retrieved from AllAboutCircuits.com: https://www.allaboutcircuits.com

Assembly, P. D. (n.d.). EDN Network. Retrieved from https://www.edn.com/pcb-design-tips-for-assembly/

Camenzind, H. (2005). Designing Analog Chips. In H. Camenzind.

Circuit Cellar – Embedded Design. (n.d.). Retrieved from CircuitCellar.com: https://circuitcellar.com

Givargis, V. a. (October 2001). Embedded System Design. In V. a. Givargis.

How to Assemble a PCB. (n.d.). Retrieved from Newbury Electronics: https://www.newburyelectronics.co.uk/blog/how-to-assemble-a-pcb

Introduction, A. U. (n.d.). Wikipedia – Embedded System. Retrieved from Wikipedia.org: https://en.wikipedia.org/wiki/Embedded_system

Jones, D. L. (2011). PCB Design Tutorial. In D. L. Jones.

Lab, G. –S. (n.d.). Retrieved from GeeksforGeeks.org: https://www.geeksforgeeks.org

Marwedel, P. (2021). Embedded Systems Foundations of Cyber-Physical Systems, and the Internet of Things".

Microcontroller Fundamentals for Engineers and Scientists. (2021). In S. F. Pack, Microcontroller Fundamentals for Engineers and Scientists.

Microprocessor Tutorial. (n.d.). Retrieved from Javatpoint.com: https://www.javatpoint.com/microprocessor

Microprocessors, M. v. (n.d.). Retrieved from IBM.com: https://www.ibm.com

N. Senthil Kumar, M. S. (2016). Microprocessors and Microcontrollers.

Organizing Electronics Labs for Development. (n.d.). Retrieved from ResearchGate.net: https://www.researchgate.net

PCB, H. t. (n.d.). Retrieved from www.Instructables.com: https://www.instructables.com/How-to-Create-a-PCB/

Scheible, J. L. (2024). Fundamentals of Layout Design for Electronic Circuits. In J. L. Scheible.

Tinkercad – Circuits. (n.d.). Retrieved from Tinkercad.com: https://www.tinkercad.com/circuits

Wilmshurst, T. (2019). Designing Embedded Systems with PIC Microcontrollers.

www.Wikipedia.org/microcontroller. (n.d.). Retrieved from www.Wikipedia.org: https://en.wikipedia.org/wiki/Microcontroller

Indicative Contents
2.1 Selection of PCB Electronic Design Software (CAD Software) **2.2 Designing Circuits Schematic Diagrams** **2.3 Circuit Simulation and Optimization** **2.4 Systematic Designing of PCB Layout** **2.5 Print and Assemble PCB**

Key Competencies for Learning Outcome 2: Build Embedded System Hardware

Knowledge	Skills	Attitudes
• Description of CAD software • Description of designing complexity • Description of vendor support and updates	• Selecting PCB electronic design software • Using CAD software • Designing PCB of embedded system • Designing circuit schematic diagram • Printing pcb • Assembling PCB	• Having Precision • Being Attentive • Having self-confident • Having accountability • Respecting time • Being patient • Having self-motivation • Being organized • Being passionate • Having creativity

Duration: 60 hrs

Learning outcome 2 objectives:

By the end of the learning outcome, the trainees will be able to:

1. Describe properly CAD software used for designing PCB layout.
2. Select appropriately PCB Electronic Design software (CAD software) in accordance with design requirements.
3. Describe properly designing complexity in accordance with design requirements.
4. Design accurately Circuits schematic diagrams based on Hardware requirements.
5. Execute properly Circuit simulation and optimization according to desired system functionality.
6. Design systematically PCB Layout with respect to the Circuit schematic diagram.
7. Print PCB correctly according to circuit design.
8. Assemble PCB correctly according to circuit design.

Resources

Equipment	Tools	Materials
• Computer • UPS	• Software Tools (Autodesk EAGLE, Ki Cad, Mentor Graphics PADS, easy EDA, Proteus PCB Design, NI multisim, LT pspice) • CAD Software for Mechanical Design (free CAD, Solid works)	• Printed Circuit Board (PCB) Materials • Semiconductor • Sensors and actuators • Electricity

Indicative content 2.1: Selection of PCB Electronic Design software (CAD software)

Duration: 10 hrs

Theoretical Activity 2.1.1: Description of CAD software

Tasks:

1: You are asked to answer the following questions:

i. Define the term CAD software.
ii. Explain 3 broad types of Computer-aided design (CAD).
iii. Explain features of CAD software.
iv. Discuss on applications of CAD software.

2: Provide the answers for the asked questions and write them on flipchart/papers.

3: Present the findings/answers to the whole class.

4: For more clarification, read the key readings 2.1.1.

5: In addition, ask questions where necessary.

Key readings 2.1.1.: Description of CAD software

1. **Definition**

   PCB design software, often referred to as CAD (Computer-Aided Design) software for PCBs, is a specialized tool used to create, design, and layout printed circuit boards.

2. **Key features (tools) of CAD software**

- **Schematic design:** PCB design software allows users to create electronic schematics.
- **Component libraries:** These software tools come with extensive component libraries that include standard components like resistors, capacitors, ICs, connectors, and more.
- **PCB layout:** PCB design software provides a canvas where you can arrange components in a physical layout. You can place components, route connections, and define the board's dimensions.
- **Auto routing:** Many PCB design tools offer auto-routing capabilities, which can automatically route connections between components while adhering to design rules and constraints.

- **Design rule checking (DRC):** DRC checks ensure that your design adheres to specified rules, such as clearance, trace width, and other design constraints.
- **File generation:** PCB design software can generate Gerber files, which are the industry-standard file format for PCB manufacturing. These files contain all the information needed to fabricate the PCB.
- **3D visualization:** Some advanced PCB design tools offer 3D visualization, allowing you to view your PCB in a three-dimensional model. This can help in identifying and resolving interference issues.
- **BOM generation:** and Bill of Materials (BOM) can be automatically generated from your PCB design, listing all components their specifications, which is essential for procurement and assembly.
- **Simulations:** Some PCB design software includes simulation features to analyze the behavior of your circuit, check for signal integrity, and identify potential issues before physical prototyping.
- **Design collaboration:** Collaboration features enable multiple users to work on the same design, making it easier for teams to collaborate on complex projects.
- **Manufacturability Analysis:** Advanced tools offer manufacturability checks to ensure that your design is feasible for production, reducing the risk of costly errors.
- **Import/Export Formats:** PCB design software typically supports various file formats for importing schematics and exporting designs.
- **Community and Support:** Many PCB design software tools have active user communities and offer customer support to help users troubleshoot issues and provide guidance.

3. Types of PCB design software (CAD software)

3.1. Professional/Commercial PCB Design Software

These are comprehensive and feature-rich PCB design tools used by professionals and companies for complex projects. They often offer a wide range of advanced features, 3D visualization, and simulation capabilities.

Examples: Altium designer, Cadence Allegro, Mentor Graphics PADS, OrCAD, Zuken CR-8000, KiCad Professional.

3.2. Freemium/Open-Source PCB Design Software

These tools are either free or open-source, making them accessible to hobbyists, students, and small businesses. While they may not have all the advanced features of commercial software, they are capable of designing functional PCBs.

Examples: KiCad, Eagle (now part of Autodesk Fusion 360), EasyEDA, Fritzing, PCBWeb, gEDA

3.3. Electronics Design Suites

Some software packages offer a complete electronics design solution that includes schematic design, PCB layout, simulation, and even mechanical design

capabilities. These are often used for multi-disciplinary design projects.

Examples: Autodesk Fusion 360, Siemens Solid Edge, and Altium Designer (which also offers a complete electronics design suite), Mentor Graphics Xpedition, Cadence Sigrity (integrated with other Cadence tools)

3.4. Online PCB Design Tools

These are web-based PCB design tools that allow users to design PCBs in a browser without the need for software installation. They are often user-friendly and suitable for quick prototyping.

Examples: Upverter, EasyEDA, and CircuitMaker (by Altium), PCBWay Online Gerber Viewer, Nano Dimension's Design for Additive Manufacturing (DFAM).

3.5. Educational PCB Design Software

These tools are designed for educational purposes and are often simplified for beginners. They can be used in classrooms or for learning PCB design.

Examples: Autodesk Tinkercad, Circuit Wizard, and Proteus Design Suite (Proteus Labcenter), 123D Circuits (by Autodesk)

3.6. Specialized PCB Design

Some PCB design software is specialized for specific industries or applications, such as high-frequency design, RF PCBs, or microwave circuits. These tools cater to unique requirements.

Examples: Keysight ADS (Advanced Design System), AWR Microwave Office, and ANSYS HFSS.

3.7. Entry-Level and Basic PCB Design Software

These tools are designed for users who need to create simple PCBs without the complexity of professional-grade software. They often focus on ease of use and affordability.

Examples: DesignSpark PCB, ExpressPCB, and FreePCB, TinyCAD Target 3001**.**

4. Application of PCB design software (CAD software)

- **Electronics prototyping and development:** PCB design software is widely used for developing and prototyping electronic circuits.
- **Consumer electronics:** Consumer electronics manufacturers use PCB design software to create layouts for devices like smartphones, tablets, laptops, gaming consoles, and smart home products.
- **Industrial control systems:** PCB design software is essential in designing control systems for industrial applications, including automation, robotics, and manufacturing equipment.
- **Aerospace and defence:** PCB design software is used in the aerospace and defense industries to create reliable and high-performance circuit boards for applications such as avionics, radar systems, and military equipment.
- **Automotive electronics:** PCB design software is applied to design the electronic systems in vehicles, including engine control units, and safety features.

- **Medical devices:** The medical industry relies on PCB design software for creating circuit boards for medical devices like patient monitors, MRI machines, and diagnostic equipment.
- **Telecommunications:** Telecommunications companies use PCB design software for developing network equipment, such as routers, switches, and communication devices.
- **Energy and power electronics:** PCB design software is employed to create circuit boards for power electronics, renewable energy systems, and energy storage devices.
- **Research and development:** Researchers use PCB design software in various scientific and engineering fields to develop custom electronic prototypes and measurement systems.
- **Education and training:** PCB design software is used for educational purposes in universities and technical schools to teach electronics design and PCB layout principles.
- **IoT (Internet of Things):** PCB design software is applied in designing IoT devices and sensor nodes that are a part of the growing IoT ecosystem.
- **Custom electronics manufacturing:** Small and medium-sized electronics manufacturers use PCB design software to design and produce custom circuit boards for various applications.
- **Prototyping Services:** PCB design software is used by prototyping and PCB fabrication services to translate customer designs into physical boards.
- **Renewable energy:** The renewable energy sector uses PCB design software to create control systems and power electronics for solar inverters, wind turbines, and energy storage systems.

Practical Activity 2.1.2: Selecting PCB electronic design software

Task:

1: Read **key reading 2.1.2**
2: Referring to key reading 2.1.2, you are sked to select PCB electronic design software.
3: Present your work to the trainer and whole class
4: Ask clarification where necessary.

Key readings 2.1.2.: Selecting PCB electronic design software

Steps to select PCB electronic design software for designing PCB of embedded system hardware

1. **Define Project Requirements**

- **Complexity**: Determine the complexity of the PCB design (e.g., single-layer, multi-layer, high-speed, mixed-signal).
- **Team Collaboration**: Identify whether multiple team members need to collaborate on the design.
- **Budget**: Set a budget for the software, considering free or low-cost options versus premium software with advanced features.
- **PCB Size and Layers**: Check the software's ability to handle the desired number of layers and board dimensions.

2. **Research Available Software Options**

- **Industry Standards**: Research commonly used software like Altium Designer, KiCad, Eagle, OrCAD, and EasyEDA.
- **User Reviews**: Read user feedback and reviews to understand the strengths and weaknesses of each software.

3. **Assess Key Features**

- **Schematic Capture and Design Tools**: Ensure the software provides intuitive tools for creating and editing schematics and layout designs.
- **Component Library**: Look for a rich and up-to-date component library that fits your project.
- **Simulation and Analysis**: Check if it supports design validation, simulation, and signal integrity analysis (important for high-speed designs).
- **Design Rule Checking (DRC)**: Ensure that the software has DRC to avoid common design errors.
- **3D Visualization**: Some software offers 3D PCB visualization, which can help in reviewing component placements and layouts.

4. **Evaluate User Interface and Ease of Use**

- **Intuitiveness**: Choose software with a user-friendly interface that aligns with your expertise and reduces the learning curve.
- **Documentation and Support**: Ensure the software has comprehensive documentation, tutorials, and customer support.

5. **Check Integration with Other Tools**

- **File Compatibility**: Make sure the software supports popular file formats (e.g., Gerber, DXF, IDF) for fabrication and collaboration.
- **Simulation Integration**: Verify if it integrates with simulation tools like SPICE or other analysis tools for testing your design.

6. **Test with a Free Trial or Demo**

- **Hands-On Testing**: If possible, use free trials or demos to evaluate the functionality and see how it fits with your workflow.

Theoretical Activity 2.1.3: Description of designing complexity

Tasks:

1: You are asked to answer the following questions:

i. What do you mean by the term assessment of designing complexity?
ii. Explain the elements to consider when assessing the designing of complexity of CAD software PCB.

2: Provide the answers for the asked questions and write them on flipchart/papers.
3: Present the findings/answers to the whole class.
4: For more clarification, read the key readings 2.1.3.
5: In addition, ask questions where necessary.

Key readings 2.1.3.: Description of designing complexity

1. **Definition**

Assessment of designing complexity refers to evaluating the level of difficulty and specific challenges of a PCB project to ensure the selected software can effectively handle them.

2. **Elements to consider when assessing the designing of complexity of CAD software PCB.**

2.1. Consider Design Type (Analog, Digital, Mixed-Signal, etc.)

In embedded systems hardware development, the design type can be categorized into three main areas based on the components and functions involved: digital design, analog design, and mixed-signal design.

- **Digital Design**

Digital design focuses on creating circuits and systems that process and manipulate digital signals. Assessment involves evaluating factors such as clock distribution, signal propagation delays, and power consumption.

Applications: microcontrollers, digital signal processors (DSPs), microprocessors, Field Programmable Gate Arrays (FPGAs), and digital communication systems.

Components: logic gates, flip-flops, registers, multiplexers, and microcontrollers.

- **Analog Design**

Analog design involves creating circuits that work with continuous signals, typically voltage or current. Assessing the complexity involves understanding factors such as noise, distortion, linearity, and sensitivity to component variations.

Applications: Analog design is essential for applications like audio amplifiers, sensor interfaces, power management, and analog-to-digital conversion (ADC).

Components: operational amplifiers (op-amps), transistors, resistors, capacitors, and inductors.

- **Mixed-Signal Design**

Mixed-signal design combines both digital and analog components within a single system. It deals with the interfacing and integration of digital and analog domains. Assessing complexity includes evaluating the interaction between analog and digital subsystems, as well as potential issues such as clock jitter and data corruption. Mixed-signal designs may require specialized simulation tools capable of handling both analog and digital aspects concurrently.

Applications: Mixed-signal design is common in applications that require the conversion between analog and digital signals, such as data acquisition systems, analog-to-digital converters (ADCs), digital-to-analog converters (DACs), and communication interfaces.

Components: It involves a combination of both digital and analog components, often with integrated circuits that include ADCs, DACs, and microcontrollers.

2.2. Available Libraries of Components and Symbols

- Assess the completeness and quality of libraries available for different design types.
- Evaluate the ease of access and compatibility of libraries with the chosen design tools.
- Consider the availability of models and symbols for commonly used components, as well as specialized components required for specific applications.

Common Sources of libraries for components and symbols are:

- **PCB designsoftware libraries**
  Most PCB design software tools come with built-in libraries that include a wide range of electronic components, footprints, and symbols.
- **Manufacturer-specific libraries**
  Component manufacturers often provide libraries and models for their components.
- **Online component libraries**
  Many PCB design software tools provide access to online component libraries, allowing users to search for and download component symbols and footprints from a cloud-based repository.
- **Open-source Libraries**
  some open-source projects and communities create and maintain libraries of components and symbols for popular PCB design software. These libraries are often free to use and may include a wide variety of components.
- **User-created libraries**
  Designers often create their own libraries of components and symbols based on their specific needs or the components they commonly use. These user libraries can be shared with others or kept for personal use.
- **Component distributor libraries**
  some distributors of electronic components offer libraries for popular PCB design software.
- **Industry-specific.libraries**
  certain industries may have specialized libraries containing components and symbols tailored to the unique requirements of those industries.
- **User.communities.and.forums**
  Online user communities and forums related to PCB design and embedded systems often share user-generated libraries of components and symbols.
- **Specialized.libraries**
  Some libraries are focused on specific types of components, such as connectors, sensors, or RF components.

2.3. Simulation and analysis capabilities

In embedded systems hardware development, simulation and analysis capabilities are essential for designing and validating the performance and reliability of electronic circuits. Various tools and techniques are used for different types of simulations and analyses.

Common simulation and analysis capabilities in embedded systems hardware development are:

- **SPICE (Simulation Program with Integrated Circuit Emphasis)**
  SPICE is a widely used tool for simulating and analyzing analog and mixed-signal

electronic circuits. It allows designers to analyze circuit behavior, including voltage, current, and component characteristics.

- **Signal Integrity (SI) analysis**
  SI analysis is critical for high-speed digital and mixed-signal designs. It assesses the quality of signal transmission to ensure that signals are not distorted, experience reflections, or encounter other issues that can affect system performance.
- **Thermal analysis**
  Thermal analysis tools are used to predict and analyze the temperature distribution within an embedded system. This is important for preventing overheating and ensuring proper thermal management.
- **Electromagnetic interference (EMI) analysis**
  EMI analysis focuses on identifying and mitigating electromagnetic interference issues that can disrupt the operation of electronic systems or lead to non-compliance with regulatory standards.
- **Digital logic simulation**
  Digital logic simulators are used to model and test the behavior of digital circuits, such as logic gates, flip-flops, and microcontroller-based systems.
- **Mixed-signal simulation**
  Mixed-signal simulators allow designers to combine analog and digital simulations to analyze interactions between digital and analog components in a system.
- **Power integrity analysis**
  Power integrity analysis ensures that the power distribution network of a system can deliver clean, stable power to all components, preventing voltage drops or power noise.

2.4. PCB design and layout features

PCB (Printed Circuit Board) design and layout software typically offer a wide range of features to assist designers in creating and optimizing PCBs for electronic devices. These features help ensure the functionality, manufacturability, and reliability of the PCB.

Common PCB design and layout features

- **Schematic capture:** allows designers to create electronic schematics and define the logical connections between components.
- **Component libraries:** provides a collection of pre-defined electronic components, footprints, and symbols to speed up the design process.
- **PCB layout editor**: enables designers to arrange components and route traces on the PCB, taking into consideration placement, spacing, and connections.
- **Auto-placement:** offers automated component placement algorithms that suggest or automatically place components on the board to optimize layout.

- **Auto routing:** automatically routes connections between components using algorithms that adhere to design rules and constraints.
- **Design Rule Checking (DRC):** performs checks to ensure that the design meets specified rules, including clearances, trace width, and other constraints.
- **Gerber file export:** generates Gerber files, which are industry-standard files used by PCB manufacturers to fabricate the PCB.
- **3D visualization**: provides a 3D view of the PCB, including components and their placement, helping designers identify interference and clearance issues.
- **BOM Generation:** automatically generates a Bill of Materials (BOM) listing all components and their specifications for procurement and assembly.
- **Simulation and analysis integration**: Integrates with simulation tools to analyze signal integrity, thermal performance, and other factors.
- **Component rotation and alignment:** allows for precise positioning and rotation of components for optimal layout.
- **User-defined footprints:** supports the creation of custom footprints for unique or non-standard components.
- **File import/export formats:** supports various file formats for importing schematics, exporting designs, and collaborating with other design tools.

2.5. Ease of use and learning curve

Consider the availability of tutorials, user guides, and community support to facilitate the learning process and minimize design iterations due to user errors.

General factors that influence the ease of use and learning curve for PCB design software are:

- **User interface (UI) design:** well-organized user interface with clear menus, toolbars, and shortcuts can significantly improve the software's ease of use.
- **Guided tutorials and documentation**: quality tutorials, user manuals, and online documentation can help new users get started and learn how to use the software effectively.
- **Schematic and PCB editor integration:** a seamless transition between the schematic capture and PCB layout editors can make the design process more straightforward.
- **Component libraries**: a wide-ranging library of pre-defined components and symbols can save users time and effort when creating schematics and layouts.
- **Automation and smart tools:** automated features, such as auto-routing and auto-placement, can simplify the design process, reducing the manual workload.
- **User community and support**: active user communities and responsive customer support can provide assistance and guidance for users facing challenges.
- **Compatibility and integration**: the ability to import and export data in various

file formats and integrate with other tools can enhance the software's usability.

- **Customization options**: software that allows users to customize shortcuts, toolbars, and settings to match their workflow preferences can enhance ease of use.
- **Error handling and feedback:** clear error messages and feedback during design, especially during design rule checks, can help users identify and rectify issues.
- **Trial versions:** offering a free trial version of the software allows users to explore its features and determine if it aligns with their needs and preferences.
- **Learning curve support:** some software providers offer training and certification programs to help users become proficient with their tools.

2.6. Integration with other tools and formats

Key areas of integration and formats to consider during Assessment of Designing Complexity

- **Schematic capture integration**
  PCB design software should integrate smoothly with schematic capture tools to ensure that the schematic and PCB layout are in sync.
- **Simulation and analysis tools**
  Integration with simulation and analysis software (e.g., SPICE, SI, and thermal analysis tools) allows designers to validate and optimize their designs.
- **Mechanical design and CAD software**
  Collaboration with mechanical design software is crucial for ensuring proper fit and clearance within the physical enclosure. File formats like STEP and IGES are commonly used for mechanical integration.
- **Bill of materials (BOM) management**
  Integration with BOM management tools can reorganise the procurement and assembly process, allowing for easy export of component data.
- **File format compatibility**
  Support for various file formats, such as DXF, DWG, PDF, and CSV, is essential for importing and exporting design data, collaborating with external partners, and sharing design documentation.
- **Library and component exchange**
  Compatibility with industry-standard library file formats like IPC-2581 (also known as DPMX) and component exchange formats like EDIF (Electronic Design Interchange Format) allows for efficient component sharing.
- **3Dvisualization and MCAD integration**
  Integration with MCAD (Mechanical Computer-Aided Design) software allows designers to visualize PCBs in a 3D context, ensuring proper fit and clearances within the enclosure.
- **Importing and exporting ECAD/MCAD data**
  Smooth exchange of electronic and mechanical design data between ECAD

(Electronic Computer-Aided Design) and MCAD tools is fundamental for collaborative product development.

- **Embedded software development tools**
  Integration with embedded software development tools allows for concurrent hardware and software development, making it easier to design and test embedded systems.

Practical Activity 2.1.4: Assessing designing complexity

Task:

1: Read **key reading 2.1.4**
2: Referring to key reading 2.1.4, you are sked to select PCB electronic design software.
3: Present your work to the trainer and whole class
4: Ask clarification where necessary.

Key readings 2.1.4.: Assessing designing complexity

Steps assessing designing complexity for embedded system hardware

1. **Understand project requirements**

   The first step is to clearly define the project's goals and technical specifications. This includes understanding the overall system functionality, performance criteria, power requirements, and any constraints such as size, cost, or environmental conditions. This clarity helps guide design decisions and ensures the final product meets the intended purpose.

2. **Define component complexity**

   After understanding the project requirements, it's important to assess the complexity of individual components, including the number and types of components (e.g., microcontrollers, sensors, power supplies). More complex components often require advanced design considerations, more intricate circuit layouts, and thorough integration. Evaluating this early on helps anticipate potential challenges.

3. **Evaluate design features**

In this step, you assess the key features and functionalities of the design. This includes determining how the components will interact, the design's scalability, and whether advanced features like communication protocols, power management, or user interfaces need to be integrated. Evaluating these features early helps in selecting appropriate design tools and frameworks.

4. **Check simulation and verification tools**

   Simulation and verification tools are crucial for testing and validating the design before manufacturing. It's important to assess whether the available tools support your design and are capable of running accurate simulations for signal integrity, power analysis, and thermal performance. A strong simulation setup minimizes errors during the actual manufacturing process.

5. **Assess manufacturing requirements**

   Once the design is evaluated, you need to ensure that the design can be manufactured efficiently. This includes evaluating PCB size, layer count, material selection, and assembly techniques. Complex designs may require specialized manufacturing processes, which should be identified early to avoid issues during production.

6. **Assess user support and learning curve**

   Finally, consider the level of user support and the learning curve associated with the design tools and software you are using. Some tools may have extensive community and vendor support, while others may require more training and experience to use effectively. Choosing tools with good support and manageable learning curves helps reduce development time and improves design quality.

Theoretical Activity 2.1.5: Description of vendor support and updates

Tasks:

1: You are asked to answer the following questions:

i. What do you mean by the term vendor support and updates when selecting PCB design software for embedded systems hardware development?

ii. Explain the vendor support and updates that should be considered before selecting PCB design software for embedded systems hardware development.

2: Provide the answers for the asked questions and write them on flipchart/papers.

3: Present the findings/answers to the whole class.

4: For more clarification, read the key readings 2.1.5.

5: In addition, ask questions where necessary.

Key readings 2.1.5: Description of vendor support and updates

1. Definition

Vendor support refers to the technical assistance and customer service provided by the software company (the vendor) to help users solve problems, troubleshoot issues, and optimize their use of the software.

Updates refer to periodic improvements, bug fixes, and new features added to the software by the vendor. These updates may include enhancements in performance, security, compatibility, and functionality.

2. Vendor support and updates to be considered before selecting PCB

2.1. Frequency of software updates

The frequency of software updates for PCB design software can vary widely depending on the vendor and the specific product. Regular updates ensure access to new features, bug fixes, and compatibility improvements, enhancing the overall user experience and productivity.

➢ **General considerations regarding the frequency of software updates**

- **Regular Updates**: Vendors provide quarterly or semi-annual updates to address bug fixes, improve performance, and introduce new features.
- **Critical Updates**: Released immediately when major security issues or bugs are

identified to ensure software stability.

- **Feature Updates**: New features and enhancements, driven by user requests and industry trends, are typically released once or twice a year.
- **Compatibility Updates**: Ensuring compatibility with new operating systems, hardware, and industry standards when significant changes occur.
- **Maintenance Updates**: Minor issues and performance improvements are addressed in monthly or bimonthly updates.
- **Long-Term Support Versions**: LTS versions offer extended support and updates over several years, ideal for long-term projects.
- **User Feedback and Bug Reports**: Updates are often influenced by user feedback, with vendors prioritizing needs and issues.

2.2. Cost and licensing considerations for PCB design software

When selecting PCB design software for embedded systems hardware development, the cost structure and licensing model play a critical role. Different software vendors offer a wide range of pricing and licensing options, which can significantly affect both short-term and long-term project budgets.

➢ **key factors to consider**

- **Upfront license cost:** Some PCB design software packages require a one-time purchase or license fee, which can involve higher initial costs but no recurring fees.
- **Subscription-based licensing:** Many providers offer subscription models, requiring users to pay periodically (monthly, quarterly, or annually) for software access and updates.
- **Free and open-source software (FOSS):** Tools such as KiCad and Fritzing are freely available and open-source, making them attractive options for cost-conscious projects or those that value community-driven development.
- **Educational and student pricing:** Some vendors provide discounted or free licensing for educational institutions and students, making these tools more accessible for academic and training purposes.
- **Free trial versions:** Many PCB design software packages offer trial versions, allowing users to explore the features and evaluate the software before committing to a purchase.
- **Cloud-based PCB design:** Cloud-based platforms often adopt a pay-as-you-go pricing model, where users are billed based on usage and storage requirements, offering flexibility and scalability.
- **Feature tiering:** Some software solutions offer tiered pricing based on available features, enabling users to select a package that aligns with their specific needs and budget constraints.

- **Maintenance and support fees:** In addition to the initial license or subscription cost, some vendors charge ongoing fees for software updates, technical support, and access to additional resources.
- **Multi-year agreements:** Some vendors offer discounts for multi-year licensing commitments, providing cost savings for projects with longer timelines.
- **Pay-as-you-grow models:** Flexible licensing options allow users to start with basic features and scale up as project requirements evolve, offering a cost-effective path for growing teams.

2.3. Community and User Feedback

Community and user feedback are invaluable resources for assessing and selecting PCB design software for embedded systems hardware development. Online forums, communities, and social media channels can provide a wealth of insights into user experiences, common issues, and workarounds.

➢ **Importance of community and user feedback**

- **Real-World Experiences:** User feedback offers first-hand accounts of how the software performs in real-world applications, helping you understand its strengths, weaknesses, and potential challenges.
- **Best Practices and Tips:** Community forums and user feedback often contain valuable tips and best practices for using the software efficiently, saving you time and effort.
- **Problem Solving:** If you encounter issues while using the software, community forums and user feedback can be excellent sources for troubleshooting tips and potential solutions.
- **Feature Requests and Improvements:** User feedback can highlight desired features and improvements. Software vendors may consider these requests when planning updates, potentially influencing the future development of the tool.
- **Comparative Insights:** Users may compare different PCB design software options, providing insights into their strengths, weaknesses, and suitability for various project types.
- **New User Advice:** Community members often provide advice and recommendations for newcomers, helping them get started and learn the basics more effectively.
- **Feedback on Updates and Bug Fixes:** User feedback can help you assess how well the software vendor addresses bug fixes and implements updates, indicating their responsiveness to user needs.
- **Discussion of Limitations:** Users may openly discuss limitations of the software, helping you determine if it is suitable for your specific project requirements.

- **User Community Support:** Engaging with a user community can provide a support network for answering questions, solving problems, and staying informed about software developments.
- **Evaluating Suitability:** By reviewing user feedback, you can assess whether the software aligns with your project's complexity, goals, and constraints.

2.4. Industry Standards Compliance

Industry standards compliance is a critical consideration in embedded systems hardware development, particularly in the design of PCBs (Printed Circuit Boards).

➢ **Key industry standards and compliance**

- **IPC standards:** The Institute for Printed Circuits (IPC) sets numerous standards related to PCB design, manufacturing, and assembly.
- **EMC/EMI Standards:** Compliance with electromagnetic compatibility (EMC) and electromagnetic interference (EMI) standards is crucial to prevent interference between electronic devices and ensure that your hardware operates without causing or suffering interference.
- **Safety standards (e.g., UL, IEC):** Depending on your application, you may need to comply with safety standards such as UL (Underwriters Laboratories) or IEC (International Electrotechnical Commission). These standards address electrical and fire safety, among other aspects.
- **RoHS (Restriction of Hazardous Substances):** RoHS compliance restricts the use of hazardous materials in electrical and electronic equipment.
- **FCC (Federal Communications Commission) Compliance:** Compliance with FCC regulations is necessary for products that emit radio frequency (RF) energy, such as wireless communication devices.
- **ISO Standards:** Depending on your industry and application, ISO standards may apply. For example, ISO 13485 is relevant to medical devices, while ISO 26262 is for automotive functional safety.
- **Aerospace and Defence Standards:** Aerospace and defence applications have specific standards, such as DO-254 for airborne electronic hardware and MIL-STD-810 for environmental testing.
- **Automotive Standards:** The automotive industry has standards like ISO 16750 for electrical and electronic systems and ISO 26262 for functional safety.
- **Medical Device Standards:** Medical devices must adhere to standards such as IEC 60601 for safety and performance.
- **Industry-Specific Standards:** Different industries, such as telecommunications, industrial automation, and consumer electronics, have their own standards and requirements that may need to be met.

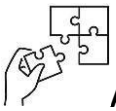
2.5. Trial and Evaluation

Trial and evaluation of PCB design software is a crucial step before committing to a specific tool for your embedded systems hardware development. A trial period allows potential users to evaluate suitability, performance, and ease of integration within their workflow before making a purchasing decision.

Points to Remember

- PCB design software is a type of CAD tool used for creating and laying out printed circuit boards. It includes features like schematic design, component libraries, auto routing, and design rule checking.
- Common applications of PCB design software include industrial control systems, telecommunications, and electronics prototyping.
- When select PCB electronic design software for designing PCB of embedded system hardware, follow those steps:
    - Define project requirements
    - Research available software options
    - Evaluate user interface and ease of use
    - Check integration with other tools
    - Test with a free trial or demo
- When select PCB electronic design software for designing PCB of embedded system hardware, follow those steps:
    - Define project requirements
    - Research available software options
    - Evaluate user interface and ease of use
    - Check integration with other tools
    - Test with a free trial or demo
- When selecting PCB CAD software, design complexity is assessed by evaluating the project's difficulty and challenges to ensure the software meets requirements.
- Key elements to consider when assessing the designing of complexity of CAD software PCB include the design type, available component libraries, simulation capabilities, ease of use, and integration with other tools.
- When assessing designing complexity for embedded system hardware, follow those steps:
    - Understand Project Requirements
    - Define Component Complexity
    - Evaluate Design Features
    - Check Simulation and Verification Tools
    - Assess Manufacturing Requirements
    - Assess User Support and Learning Curve

- There are vendor support and updates to consider when choosing PCB design software for embedded system hardware development, such as frequency of update, cost and licensing, community feedback, and adherence to industry standards.
- The difference between updates and vendor support is that updates enhance performance and security, while vendor support assists with troubleshooting.



Application of learning 2.1.

RTY Solutions is a company that provide different electronic services, the engineering team is engaged in the development of an embedded system for a smart home control system. As part of this initiative, your primary responsibility is to select the appropriate PCB (Printed Circuit Board) Electronic Design Software (CAD software) for designing the hardware. The selected software will facilitate the creation of the circuit layout for the embedded system, ensuring that the design is efficient, cost-effective, and compliant with the system's specifications.

Indicative content 2.2: Designing circuits schematic diagrams

Duration: 15 hrs

Theoretical Activity 2.2.1: Description of circuit schematic diagrams

Tasks:

1: You are asked to answer the following questions:

i. What do you mean by the following term?
   a) Circuit schematic diagrams.
   b) Schematic diagram block diagram.
   c) Connectivity and schematic software.
   d) Design guide lines
   e) Annotate components

ii. Explain the elements required to consider when designing circuit schematic diagram.

2: Provide the answers for the asked questions and write them on flipchart/papers.

3: Present the findings/answers to the whole class.

4: For more clarification, read the key readings 2.2.1.

5: In addition, ask questions where necessary.

Key readings 2.2.1.: Designing circuits schematic diagrams

1. Definition

1.1. Circuit schematic diagrams

A circuit schematic diagram is a graphical representation of an electrical or electronic circuit, showing the connections between components using standardized symbols. It illustrates how different elements such as resistors, capacitors, and transistors are connected to each other and the flow of current through the circuit.

1.2. Schematic diagram block diagram

A block diagram is a simplified version of a schematic diagram that represents the system or circuit using blocks. Each block typically represents a larger subsystem or functional component, and the connections between the blocks show the flow of information or signals. Block diagrams are useful for illustrating overall system architecture without delving into detailed connections.

1.3. Connectivity and schematic software

Connectivity and schematic software refers to software tools used to

create, edit, and analyse circuit schematic diagrams. These tools ensure that all the electrical connections are correctly made and may include features for checking connectivity between components, generating netlists (lists of connections), and providing visualization of circuit paths.

1.4. Design guidelines

Design guidelines are a set of best practices and recommendations that guide engineers in creating effective, reliable, and optimized circuit designs. These guidelines can cover a range of topics, including layout rules, component selection, power distribution, signal integrity, and compliance with industry standards.

- **Component selection**

- Choose components with appropriate ratings for power, voltage, and temperature.
- Consider component reliability, availability, and cost.
- Prioritize components with known performance characteristics and datasheets.
- Use reputable suppliers to ensure component quality.

- **Layout rules**

- Adhere to specific layout rules for your PCB fabrication process to minimize manufacturing defects.
- Maintain appropriate clearances between traces and components to prevent shorts and crosstalk.
- Use ground planes or copper fills to reduce noise and improve signal integrity.
- Optimize trace lengths and routing to minimize impedance mismatches and reflections.

- **Power distribution:**

- Design a robust power distribution network to deliver clean, stable power to all components.
- Use decoupling capacitors to filter out high-frequency noise.
- Consider the impact of current flow on trace impedance and voltage drops.
- Implement bypass capacitors near high-current components to reduce voltage spikes.

- **Signal integrity:**

- Minimize signal path lengths to reduce propagation delays and crosstalk.
- Use controlled impedance traces to ensure proper signal transmission.
- Consider the effects of termination resistors on signal reflections.
- Employ differential signaling for high-speed data transmission to improve noise immunity.

- **Thermal management:**

- Ensure adequate heat dissipation to prevent component overheating.
- Use heat sinks or forced air cooling if necessary.
- Consider the thermal conductivity of PCB materials and component mounting methods.
- Monitor component temperatures during operation to identify potential thermal issues.

- **Compliance with standards:**

- Adhere to relevant industry standards, such as IEC, UL, and FCC.
- Ensure compliance with safety regulations and electromagnetic compatibility (EMC) requirements.
- Consider certification requirements for specific applications or markets.

- **Design verification and testing:**

- Conduct thorough simulations and analysis to verify circuit performance.
- Perform functional testing on prototypes to identify and address any issues.
- Implement a robust testing plan to ensure product quality and reliability.

1.5. Annotate components

Annotating components refers to the process of labeling and assigning reference designators (such as R1 for resistors, C1 for capacitors) to each component in a schematic diagram. Proper annotation ensures that each part is uniquely identifiable, which is essential for clear documentation, communication, and manufacturing of the circuit.

2. Elements to consider when designing circuit schematic diagrams

When designing a circuit schematic diagram, several key elements must be taken into account to ensure the circuit functions correctly and is easy to understand. These include:

2.1. Component selection

Choose appropriate components based on the circuit's requirements, such as voltage, current ratings, and functionality. Ensure the components are available and suitable for the operating conditions.

2.2. Correct symbol usage

Use standardized symbols for each electronic component (resistors, capacitors, transistors, etc.) to ensure clarity and adherence to industry norms. This makes the schematic understandable for anyone reading it.

2.3. Proper connections and layout

Ensure that all components are connected correctly as per the circuit requirements. The layout should be organized to minimize confusion, with signals flowing logically from left to right or top to bottom.

2.4. Power supply and grounding

Clearly show how power is supplied to the circuit and how components are grounded. Proper labeling of voltage levels and ground points is essential

for understanding the circuit's operation.

2.5. Signal paths and flow

Maintain a logical flow of signals and connections, making it easy to follow the signal paths through the circuit. Use clear labels for input and output signals to clarify functionality.

2.6. Annotation and labelling

Annotate all components with unique reference designators and label important nodes or signals. This helps in identifying components and understanding the circuit during troubleshooting or manufacturing.

2.7. Electrical rules checking (ERC)

Use schematic design software's built-in tools to check for errors like missing connections, conflicting nets, or incorrect wiring that could lead to malfunctions in the circuit.

2.8. Component ratings

Verify that all components can handle the expected voltage, current, and power in the circuit. Proper ratings are crucial for preventing failure or damage.

2.9. Design for manufacturability (DFM)

Consider the practical aspects of manufacturing the circuit. Ensure that the design follows the necessary guidelines for assembly, testing, and production, including spacing between components and ease of routing.

2.10. Cross-referencing with PCB layout:

Ensure that the schematic diagram aligns with the PCB layout and that any changes in one are reflected in the other. This includes the placement of components and routing of signals in the final design.

Practical Activity 2.2.2: Designing circuit schematic diagram

Task:

1: Read key reading 2.2.2

2: Referring to key reading 2.2.2, you are requested to design circuit schematic diagram of Automatic Smart Home appliance control system**.** This system is designed to automate various home functions, including lighting, security, climate control, and appliance management.

3: Present your work to the trainer and whole class.

4: Ask clarification where necessary.

Key readings 2.2.2: Designing circuit schematic diagram

Steps of designing circuit schematic diagram

1. **Creation of block diagrams**

   Begin by creating a high-level block diagram outlining the system architecture and functional blocks.

   This simple structure shows the relationship between the power supply, microcontroller, sensors, actuators, and communication modules.

2. **Selection of components and footprint assignment**

   Choose appropriate components based on functional requirements and specifications. Assign footprints ensuring compatibility with the chosen PCB design tools and manufacturing processes.

   When selecting components and assigning footprints for an embedded system, it's important to follow a systematic approach to ensure compatibility, performance, and manufacturability. Here are the key steps involved in this process:

3. **Determine the connectivity and schematic software**

   Establish connections between components to reflect their intended functionality. Utilize schematic capture software such as Altium Designer, KiCad, or Eagle to create the schematic diagram.

   Determining connectivity and selecting schematic software are crucial steps in the design process of an embedded system. Here's how you can approach these tasks:

   1.Determine Connectivity

   A. Define Communication Needs

   Identify the components that need to communicate with each other (e.g., microcontroller, sensors, actuators, and communication modules).

   Decide on the type of communication required, such as:

   Digital Communication: SPI, I2C, UART, CAN, etc.

   Analog Communication: ADC/DAC interfaces.

   Wireless Communication: Wi-Fi, Bluetooth, ZigBee, etc.

   B. Create a Connectivity Plan

   Map out how each component will connect to the microcontroller or processor.

   Specify pin assignments for each connection, ensuring that the correct protocols are used for communication.

   Consider any necessary pull-up/down resistors, capacitors, or other components needed for signal integrity.

   C. Consider Power Distribution

Ensure that power supply connections are properly designed for each component, accounting for voltage levels and current requirements.

2.Select Schematic Software

Choosing the right schematic capture software is essential for creating accurate and efficient designs. Here are some popular options:

A. Free and Open-Source Software

KiCad: A powerful, open-source EDA tool that supports schematic capture and PCB layout. It has a user-friendly interface and a large community for support.

Fritzing: Good for beginners, especially for projects involving Arduino and other prototyping platforms. It allows for easy visualization of circuits.

B. Commercial Software

Altium Designer: A professional-grade tool with extensive features for schematic capture, PCB layout, and design management. It is widely used in the industry.

Eagle: A popular choice among hobbyists and professionals, known for its ease of use and integration with PCB design.

OrCAD: A robust tool for schematic capture and PCB design with strong simulation capabilities, often used in larger projects.

LTspice: While primarily a simulation tool, it can also be used for schematic capture and analysis of analog circuits.

C. Considerations for Selection

Ease of Use: Choose software that matches your experience level and comfort with EDA tools.

Features: Look for features such as library management, simulation capabilities, and integration with PCB layout tools.

Cost: Consider your budget, as some tools have free versions while others require licenses.

Community and Support: Evaluate the availability of tutorials, forums, and documentation to help you learn and troubleshoot.

3. Create the Schematic

Start by placing components in the schematic software based on your connectivity plan.

Connect the components using wires, ensuring that all connections are correctly represented.

Label the connections and components clearly for easy reference.

4. Review and Validate

Double-check the schematic for errors, such as incorrect connections or missing components.

Use the software's validation tools to identify potential issues before moving on to PCB design.

4. **Follow design guide lines**

   Adhere to design guidelines and standards relevant to the industry and project requirements. Consider factors like signal integrity, power distribution, and EMI/EMC considerations.

5. **Label components and signals**

   Clearly label components and signals to ensure clarity and ease of understanding for stakeholders and manufacturing personnel.

6. **Annotate Components**

   Annotate components with unique reference designators to facilitate identification during assembly and debugging processes.

   Labeling components and signals in your schematic and PCB layout is crucial for clarity, ease of understanding, and effective communication among team members. Here are some best practices for labeling components and signals:

   1. Component Labeling

   A. Use Standard Designations

   Follow standard conventions for component designations (e.g., R for resistors, C for capacitors, U for integrated circuits, Q for transistors).

   For example:

   R1, R2, R3 for resistors

   C1, C2 for capacitors

   U1, U2 for integrated circuits

   Q1, Q2 for transistors

   B. Unique Identifiers

   Ensure that each component has a unique identifier within the schematic to avoid confusion.

   If multiple identical components are used, label them sequentially (e.g., R1, R2, R3) to indicate their positions.

   C. Include Values and Ratings

   Next to each component label, include its value or specification where applicable (e.g., R1: 10kΩ, C1: 100nF).

   For ICs, include part numbers or descriptions to clarify their function (e.g., U1: ATmega328P).

   2. Signal Labeling

   A. Clear Signal Names

   Use descriptive names for signals that clearly indicate their function (e.g., `Vcc`, `GND`, `TX`, `RX`, `SDA`, `SCL`).

   Avoid using generic names like `Signal1` or `WireA`, as they do not convey meaningful information.

   B. Consistent Naming Conventions

Establish a naming convention and stick to it throughout the design. For **example**:

Use prefixes to indicate signal types, such as:

I_` for input signals (e.g., `I_Temperature`)

`O_` for output signals (e.g., `O_LED`)

`S_` for sensor signals (e.g., `S_Temperature`)

`PWM_` for pulse-width modulation signals (e.g., `PWM_Motor`)

C. Use Hierarchical Labels

In complex designs, consider using hierarchical labeling to group related signals (e.g., `Sensor1/Temperature`).

This approach can help organize signals logically and improve readability.

3. Ground and Power Labels

A. Common Ground and Power Labels

Use standard labels for power and ground connections (e.g., `GND`, `Vcc`, `Vdd`).

- If multiple voltage levels are used, clearly label them (e.g., `V1.8`, `V3.3`, `V5`).

B. Power Distribution Network

Label power distribution traces clearly to indicate their purpose and voltage levels.

4. Documentation and Notes

A. Include a Legend or Key

If your design contains many signals or components, consider including a legend or key that explains the labeling conventions used.

This can be especially helpful for new team members or reviewers.

B. Annotations and Comments

Use annotations or comments in your schematic to provide additional context or explanations for specific components or signals.

5.Review and Consistency

A. Review Labels

Before finalizing your design, review all labels for consistency, accuracy, and clarity.

Ensure that all components and signals are labeled according to the established conventions.

7. Validate and review

Validate the schematic design against functional requirements and specifications. Conduct thorough reviews to identify errors, inconsistencies, and areas for improvement.

Validating and reviewing your design is a critical step in the development process of an embedded system. This ensures that your schematic and PCB

layout meet the required specifications and function as intended. Here's a structured approach to effectively validate and review your design:

1.Schematic Validation

A. Electrical Rule Check (ERC)

Use the built-in ERC tools in your schematic software to identify errors such as unconnected pins, shorts, and incorrect connections.

Review warnings and errors generated by the ERC and address them accordingly.

B. Functional Review

Verify that the schematic accurately represents the intended functionality of the circuit.

Check that all components are correctly connected according to the design specifications.

C. Component Values and Ratings

Ensure that all component values (e.g., resistances, capacitances) and ratings (e.g., voltage, current) are correct and suitable for the application.

Cross-check the part numbers and specifications against the manufacturer's datasheets.

D. Signal Integrity Analysis

If applicable, perform signal integrity analysis to identify potential issues with high-speed signals, such as reflections or crosstalk.

2. PCB Layout Validation

A. Design Rule Check (DRC)

Run the DRC in your PCB design software to check for layout issues such as trace width violations, clearance violations, and incorrect pad sizes.

Address any violations reported by the DRC.

B.Layer Stack-Up Review

Verify that the layer stack-up is appropriate for the design, particularly in multi-layer boards.

Ensure that power and ground planes are correctly implemented.

C. Trace Routing and Width

Review the routing of all traces to ensure they are as short and direct as possible.

Confirm that trace widths are adequate for the expected current loads.

D. Component Placement

Check that components are placed logically, with related components grouped together.

Ensure that polarized components (e.g., capacitors, diodes) are oriented correctly.

4.Prototype Testing

A. Build a Prototype

If feasible, create a prototype of your design to test its functionality in real-world conditions.

Use a breadboard or a development board for initial testing if a full PCB is not yet available.

B. Functional Testing

Test the prototype to verify that it performs as intended and meets the design specifications.

Check all inputs and outputs, and validate communication between components.

C. Debugging

If issues arise during testing, use debugging tools (e.g., oscilloscopes, multimeters) to identify and resolve problems.

Review the schematic and layout to locate potential design flaws.

5. Peer Review

A. Collaborate with Team Members

Share your design with colleagues or team members for a peer review. Fresh eyes can catch errors you may have overlooked.

Encourage constructive feedback and suggestions for improvement.

B. Review Checklist

Create a checklist of design criteria to guide the review process. This can include aspects like component selection, layout considerations, and compliance with design guidelines.

6. Final Review and Approval

A. Sign-Off Process

Establish a sign-off process where key stakeholders review and approve the final design before moving to production.

Ensure that all necessary revisions are made based on feedback from the review process.

B. Archive Design Files

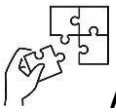
Once approved, archive all design files and documentation for future reference and potential updates.

8. **Update iteratively**

Iterate on the design based on feedback from validation and reviews. Update the schematic diagram iteratively to incorporate changes and optimizations, ensuring continuous improvement and alignment with project objectives.

Points to Remember

- A circuit schematic diagram visually represents electrical circuits, detailing component connections with standardized symbols. In contrast, a block diagram simplifies this by grouping components into functional blocks, showing system architecture.
- Connectivity and schematic software aid in creating and validating these diagrams, while design guidelines ensure effective and reliable designs.
- Key design elements include component selection, correct symbol usage, proper connections, power supply labeling, signal flow clarity, and adherence to electrical rules and manufacturability standards.
- To design a circuit schematic diagram for embedded system, follow these steps:
    - ✓ Create block diagrams
    - ✓ Select components and footprint
    - ✓ Determine the connectivity and schematic software
    - ✓ Follow design guide lines
    - ✓ Label components and signals
    - ✓ Annotate Components
    - ✓ Validate and review
    - ✓ Update iteratively



Application of learning 2.2

XYB Solutions is a company located in Kigali city that provide different electronic services, the engineering team is designing the circuit schematic diagram for an Automatic Smart Home Appliance Control System. This system aims to automate functions like lighting, security, climate control, and appliance management based on environmental conditions and user preferences. Key components include sensors, smart relays, and communication modules to enable seamless operation. As the assigned technician for this project, your responsibilities involve reviewing system requirements, selecting appropriate components, creating the block diagram, and developing the detailed circuit schematic for the system.

Indicative content 2.3: circuit simulation and optimization

Duration: 15hrs

Theoretical Activity 2.3.1: Description of circuit simulation and optimization

Tasks:

1: You are asked to answer the following questions:

i. Define the following term:
   a) Circuit simulation
   b) Circuit optimization
ii. What are the purpose of circuit simulation?
iii. What are the purpose of Circuit optimization?

2: Provide the answers for the asked questions and write them on flipchart/papers.

3: Present the findings/answers to the whole class.

4: For more clarification, read the key readings 2.3.1.

5: In addition, ask questions where necessary.

Key readings 2.3.1: Description of circuit simulation and optimization

1. Definition

1.1. Circuit simulation

Circuit simulation is a process in which a model of an electronic circuit is created and analysed using various software.

Popular Circuit Simulation Software:

- **SPICE** (Simulation Program with Integrated Circuit Emphasis): Widely used for analog circuit simulation.
- **LTspice**: Free software for simulating electrical circuits, commonly used for power electronics.
- **Proteus**: Offers both circuit simulation and PCB design.
- **Multisim**: Great for both educational and industrial circuit design and simulation.
- **KiCad**: Open-source software for electronic design automation (EDA), including simulation.

Simulation Tools

There are many circuit simulation tools available, each with its own strengths and

weaknesses. Some popular options include:

- **SPICE:** A widely used open-source simulator that can handle a variety of circuit types, including analog, digital, and mixed-signal.
- **LTspice:** A free simulator from Analog Devices that is easy to use and integrates well with their component libraries.
- **OrCAD:** A commercial tool that offers a comprehensive suite of design tools, including schematic capture, PCB layout, and simulation.
- **MATLAB/Simulink:** A powerful platform for modelling and simulating complex systems, including electronic circuits.
- **Cadence Virtuoso:** A high-end design environment used for advanced IC design and simulation

1.2. Circuit optimization

Circuit optimization is a process of reducing circuit complexity/cost and to improve performance.

Optimization Techniques:

- **Component Tuning**: Adjust component values (e.g., resistors, capacitors) based on simulation feedback to optimize performance.
- **Layout Optimization**: Refine the physical layout to reduce noise, signal interference, and parasitic in the circuit.
- **Thermal Management**: Ensure proper heat dissipation by optimizing the placement of heat-generating components.
- **Power Optimization**: Adjust the circuit design to reduce power consumption without sacrificing performance.

There are some of the most popular circuit optimization tools:

Commercial Tools

- **Cadence Virtuoso:** A high-end design environment used for advanced IC design and simulation, offering powerful optimization capabilities.
- **Synopsys Primetime:** A timing analysis and optimization tool that helps ensure circuit performance meets timing constraints.
- **Ansys Sherlock:** A thermal simulation and optimization tool that can help prevent overheating and improve circuit reliability.
- **Mentor Graphics QuestaSim:** A functional verification tool that includes optimization features for digital circuits.

Open-Source Tools

- **SPICE:** A widely used open-source circuit simulator that can be combined with optimization scripts or libraries.
- **Pspice:** A Python-based interface to SPICE, providing a flexible environment for circuit simulation and optimization.
- **SciPy:** A scientific computing library that includes optimization algorithms that can be applied to circuit design problems.

2. Purpose of circuit simulation

- **Validate design**: Ensure the circuit functions as intended without needing to build physical prototypes.
- **Test performance**: Simulate the circuit under various operating conditions (e.g., temperature, voltage levels).
- **Identify issues**: Detect potential issues like signal noise, component failures, and power consumption early in the design phase.
- **Speed up development**: Avoid costly errors in hardware and speed up development by running simulations instead of building multiple prototypes.

3. Purpose of circuit optimization

- **Improve Efficiency**: Minimize power consumption, signal loss, and heat dissipation.
- **Optimize Cost**: Use fewer or more cost-effective components while maintaining performance.
- **Maximize Performance**: Enhance speed, accuracy, or responsiveness of the circuit.
- **Reduce Size**: Optimize the layout to reduce the physical size of the PCB and components.

Practical Activity 2.3.2: Simulating and Optimizing circuit

Task:

1: Read key reading 2.3.2

2: Referring to the key readings 2.3.2, you are requested to Simulate and Optimize circuit of Automatic Smart Home Appliance control system. This system is designed to automate various home functions, including lighting, security, climate control, and appliance management.

3: Present your work to the trainer and whole class.

4: Ask clarification where necessary.

Key readings 2.3.2: Simulating and Optimizing circuit

1. Steps of simulating a circuit

1.1. Select a Simulation Tool

Choose a suitable circuit simulation tool based on the complexity of the circuit and the desired analysis capabilities. Common tools include SPICE simulators

like LTspice, Pspice, or commercial solutions like Cadence Spectre.

1.2. Specify Simulation Settings

Define simulation settings such as the type of analysis (DC, AC, transient), simulation time, tolerances, and model parameters. Ensure all relevant circuit components and models are properly defined.

1.3. Run the Simulation

Execute the simulation within the chosen software environment. Verify that the simulation converges without errors and progresses according to the specified settings.

1.4. Analyze Results

Review simulation results to gain insights into circuit behavior. Analyze key parameters such as voltages, currents, power dissipation, and frequency response to assess circuit performance and identify any anomalies or areas for improvement.

2. Steps of optimizing a circuit

1.1. Identify Goals and Constraints

Define specific optimization goals such as maximizing efficiency, minimizing power consumption, or achieving certain performance metrics. Identify any constraints related to component values, cost, or physical size.

1.2. Choose an Optimization Tool

Select an appropriate optimization tool or algorithm suited to the complexity of the circuit and the optimization objectives. This could include built-in optimization features within circuit simulation software or external optimization packages.

1.3. Specify Optimization Parameters

Define the parameters to be optimized, such as component values, topology configurations, or operating conditions. Set constraints to ensure feasibility and practicality of the optimized solution.

1.4. Run the Optimization

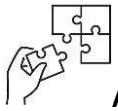
Initiate the optimization process within the chosen tool or software environment. Allow the optimization algorithm to iteratively adjust parameter values to improve performance while adhering to specified constraints.

1.5. Evaluate Results

Assess the optimized circuit design based on the defined goals and constraints. Evaluate key metrics and performance indicators to determine if the optimization objectives have been achieved satisfactorily. Iterate on the optimization process as needed to fine-tune the design and improve results further.

Points to Remember

- Circuit simulation is a process in which a model of an electronic circuit is created and analysed using various software.
- The purpose of circuit simulation include validating the design and testing performance under varying conditions.
- Circuit optimization is a process of reducing circuit complexity/cost and to improve performance.
- The purpose of circuit optimization includes improving efficiency, lowering costs, maximizing speed and accuracy, and minimizing the circuit's physical size.
- To simulate embedded system hardware circuit, follow these steps:
    - Select a Simulation Tool
    - Specify Simulation Settings
    - Run the Simulation
    - Analyse Results
- To optimize embedded system hardware circuit, follow these steps:
    - Identify Goals and Constraints
    - Choose an Optimization Tool
    - Specify Optimization Parameters
    - Run the Optimization
    - Evaluate Results
    - Refine the Circuit Design



Application of learning 2.3.

You are a technician at ABC Innovations, a company focused on creating smart home solutions. The company has been tasked with developing an Automatic Smart Home Appliance Control System to automate home functions like lighting, security, climate control, and appliance management. Your role is to simulate and optimize the circuit design for this system before it is physically built and deployed.

Indicative content 2.4: Systematic designing of PCB layout

Duration: 10hrs

Practical Activity 2.4.1: Designing PCB layout of embedded system hardware

Task:

1: Read key reading 2.4.1

2: Referring to the key readings 2.4.1 , you are requested to design PCB layout of embedded system hardware of Automatic Smart Home Appliance control system. This system is designed to automate various home functions, including lighting, security, climate control, and appliance management.

3: Present your work to the trainer and whole class.

4: Ask clarification where necessary.

Key readings 2.4.1.: Designing PCB layout of embedded system hardware

1. Steps for designing the PCB Layout for an embedded system hardware

1.1. Creating a Schematic Diagram of the Circuit

The first step in designing a PCB is to create a schematic diagram, which represents the electronic components and how they are interconnected. For the smart home control system, the schematic will include components like:

Microcontroller (e.g., ESP32 or Arduino)

Relays to control appliances

Sensors (e.g., temperature, humidity, motion)

Communication modules (Wi-Fi, Bluetooth)

Power supply circuits

Connectors and switches

The schematic defines the logical connections and serves as a blueprint for the PCB layout.

1.2. Interpret the Schematic Diagram

After creating or reviewing the schematic, it's important to interpret the diagram

carefully:

Check component connections: Ensure that each component is correctly connected (e.g., power pins, signal lines, ground connections).

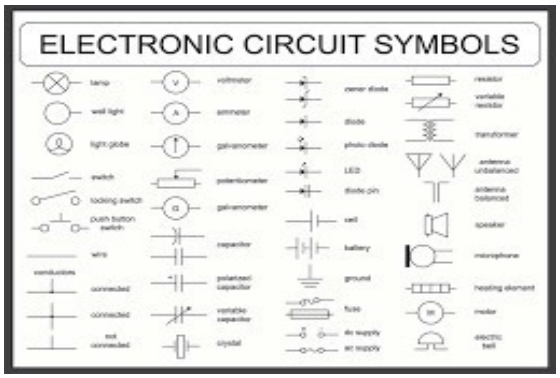
Understand circuit flow: Identify how data signals, control lines, and power will flow through the circuit.

Look for errors or omissions: Verify that no components are missing and that every connection is accurately represented.

This step ensures that the schematic correctly represents the intended functionality of the system before moving to the PCB layout.

Understand the connections, components, and signal paths depicted in the schematic diagram.

Ensure clarity on the intended functionality and design requirements before proceeding to layout.



1.3. Choosing a PCB design software

Once the schematic is ready, you need to choose the right PCB design software that meets your requirements and skill level. Some options include:

Altium Designer: Suitable for advanced users needing complex features.

Ki CAD: Free, open-source software with extensive features for beginners and professionals.

Eagle: Widely used for creating PCB layouts with integrated schematic capture.

Proteus: Great for embedded systems, as it allows both simulation and PCB design.

Select software that offers the tools you need to handle your components, layers, and complexity, and that you are comfortable using.

1.4. Importing the components from the schematic into the PCB design Software

Most PCB design tools allow you to import the schematic directly into the PCB layout environment:

Component footprints: The software will automatically assign physical footprints to each component based on the schematic.

Netlist: The connections between the components (netlist) will be imported, helping you lay out the physical connections in the PCB layout.

Library management: If some components are not in the default library, you may need to import or create custom footprints.

Ensure that all components and their interconnections from the schematic are properly represented in the PCB design software.

1.5. PCB Layer Stackup Design

Designing the layer stackup defines how the PCB will be built in terms of its layers:

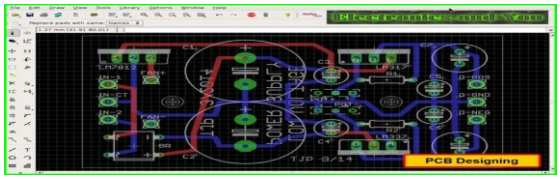
Single-layer or multi-layer design: For simple designs, a single-layer PCB may suffice, but for more complex circuits (e.g., with power and signal traces), you may need a multi-layer PCB (e.g., two-layer, four-layer).

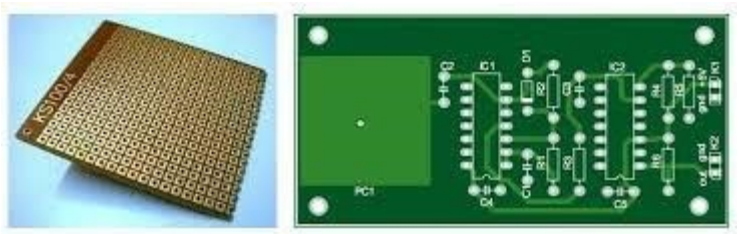
Power and ground planes: Add dedicated layers for power and ground to ensure stable power distribution and minimize noise.

Signal layers: Separate high frequency signals from noisy components, and use different layers for sensitive signal lines (e.g., microcontroller data and sensor lines).

Optimize the PCB's electrical performance, reduce noise, and improve heat dissipation by designing a proper layer stackup.

Determine the layer stack up configuration based on factors such as signal integrity, power distribution, and manufacturability. Define the number of layers, signal routing, and placement of power and ground planes to optimize performance and reduce electromagnetic interference.





Dotted PCB Layout PCB

1.6. Connecting the components using traces (Copper Lines) on the PCB

Now that the components are placed on the board, the next step is to connect them using copper traces:

Trace width and thickness: Choose appropriate trace widths to handle the current levels flowing through the circuit. High-current traces (e.g., power lines) should be thicker than signal lines.

Routing: Route the traces between components. For sensitive signals, keep the traces as short as possible and avoid routing near high-power traces to minimize noise.

Via placement: Use vias to connect different layers of the PCB where necessary, especially for multi-layer designs.

Grounding: Use a solid ground plane for improved signal integrity and noise reduction.

Ensure that all components are correctly connected with traces, optimizing for performance, power distribution, and noise reduction.

1.7. PCB Prototype

Once the PCB layout is finalized, you can create a prototype to test its functionality:

Fabrication: Send the PCB design files to a manufacturer to create a physical prototype.

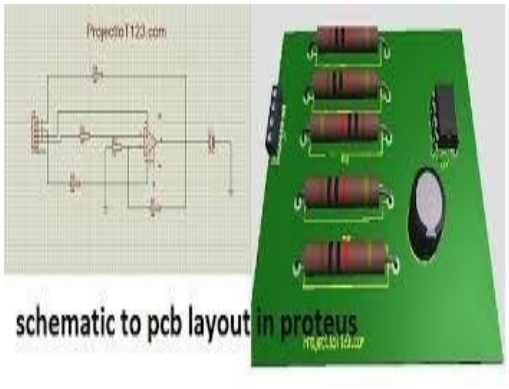
Assembly: Assemble the components onto the fabricated PCB, either manually or through automated assembly.

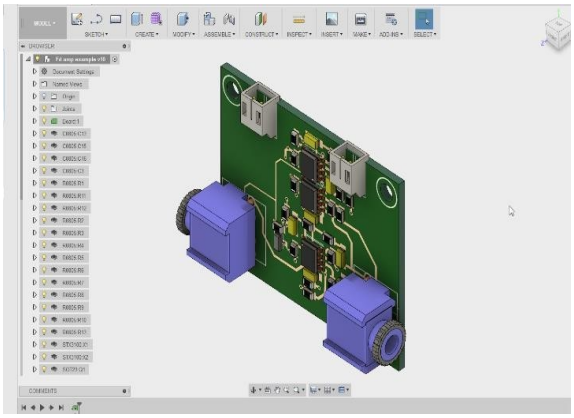
Testing Test the prototype to ensure it functions as expected, verifying signal integrity, power distribution, and overall performance.

Debugging: If any issues arise, make necessary adjustments in the design (e.g., trace routing or component values) and re-fabricate the PCB if needed.

Build and test a working prototype to identify any design flaws before mass production.

Fabricate a prototype PCB to validate the design and functionality before mass production. Use rapid prototyping techniques or PCB manufacturing services to produce a small batch of boards for testing and verification.





1.8. Export File

After the design is validated, you'll export the necessary files for manufacturing:

Gerber files: These are the standard files that contain the layout information for each layer of the PCB (copper layers, solder mask, silkscreen).

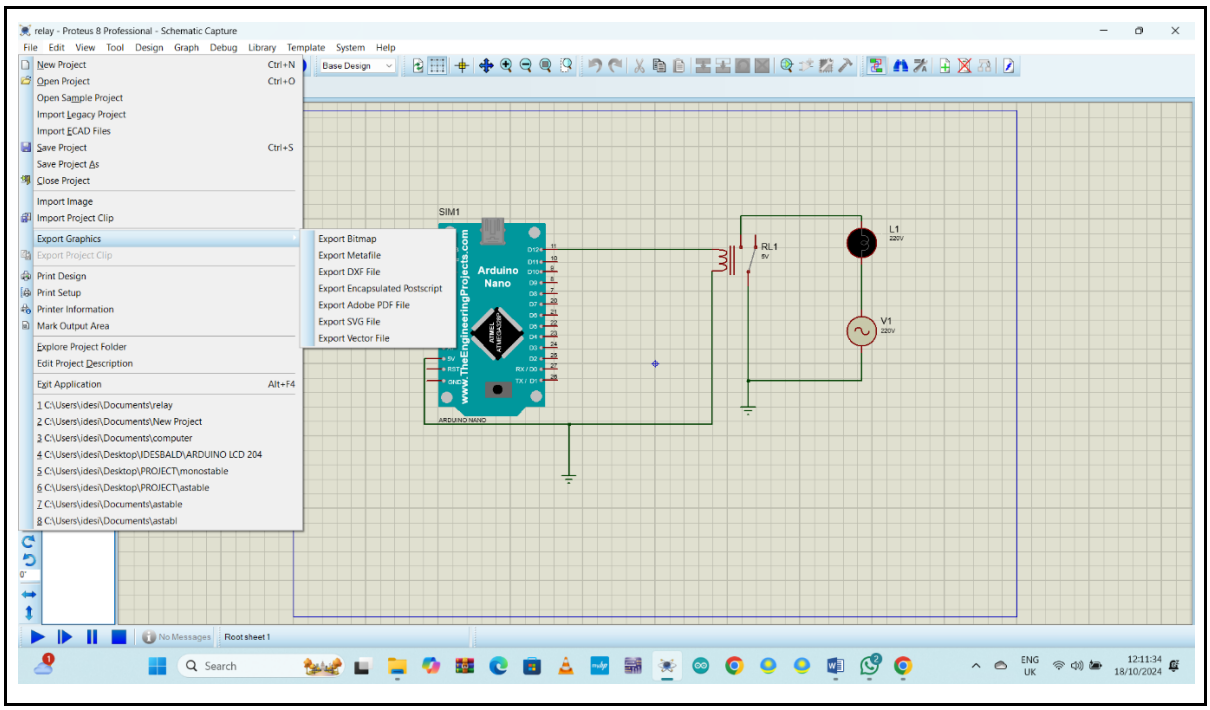
Bill of Materials (BOM): A list of all components needed to assemble the PCB, including part numbers and quantities.

Drill files: These specify the locations of holes and vias on the board.

Pick and place file: If automated assembly is used, this file guides the machines on where to place components.

Provide the manufacturer with all necessary files to fabricate and assemble the PCB accurately.

Export the PCB layout design files in the appropriate format for manufacturing. Common file formats include Gerber files for PCB fabrication, Excellon files for drill data, and Bill of Materials (BOM) for component procurement. Ensure compliance with manufacturer specifications and requirements.



Points to Remember

- To design PCB layout for embedded system hardware, follow these steps:
  - Create a schematic diagram of the circuit.
  - Interpret the schematic diagram
  - Choose a PCB design software that suits your needs and level of expertise.
  - Import the components from the schematic into the PCB design software.
  - PCB Layer stackup design
  - Connect the components using traces (copper lines) on the PCB.
  - PCB Prototype and after Export file

Application of learning 2.4.

You are a technician at ABC Innovations, a company that specializes in smart home solutions. Your current task is to design the PCB layout for an automatic smart home appliance control System. This system automates various household functions, such as lighting, climate control, security, and appliance management. Your responsibility is to ensure that the design is efficient, reliable, and ready for real-world application.

Indicative content 2.5: print and assemble PCB

Duration: 15hrs

Practical Activity 2.5.1: Printing and assembling embedded System Hardware PCB

Task:

1: Read key reading 2.5.1

2: Referring to the key readings 2.5.1, you are requested to Print and assemble embedded System Hardware PCB of automatic smart Home Appliance control system. This system is designed to automate various home functions, including lighting, security, climate control, and appliance management.

3: Present your work to the trainer and whole class.

4: Ask clarification where necessary.

Key readings 2.5.1.: Printing and Assembling Embedded System Hardware PCB

1. **Definition**

1.1. Printing PCB

Printing PCB refers to the process of fabricating or manufacturing the actual physical printed circuit board (PCB) based on the design files. It doesn't involve literal printing on paper, but rather the creation of the board using various materials and techniques

1.2. Assembling PCB

Assembling PCB is the process of attaching electronic components to the fabricated PCB. The assembly transforms the bare PCB into a working circuit by placing and soldering components like resistors, capacitors, ICs, and connectors onto the board.

1.3. CNC machine

A **CNC machine** (Computer Numerical Control machine) used for printing or fabricating PCBs is a specialized device that automates the process of creating printed circuit boards by precisely removing material (usually copper) from a substrate to form the necessary circuit traces, holes, and patterns.

2. Main methods used for printing embedded system hardware PCB

There are 2 main methods used when printing embedded system hardware PCB

- Printing using CNC
- Printing using Chemicals

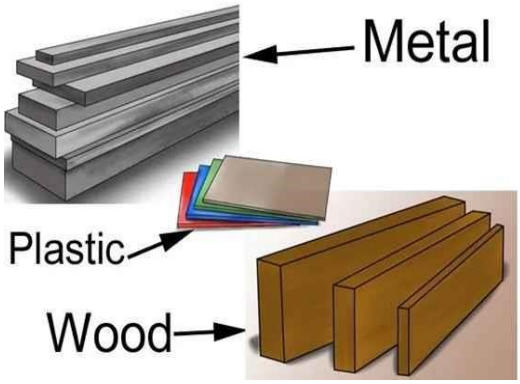
2.1. Printing using CNC Machine

Steps to follow when printing PCB using CNC Machine

- ✓ Prepare materials and tools
- ✓ CNC Machine Setup
- ✓ Upload Gerber file
- ✓ CNC Milling and Drilling
- ✓ PCB Inspection and Cleaning
- ✓ Silkscreen Placement
- ✓ Component placement
- ✓ Test the circuit

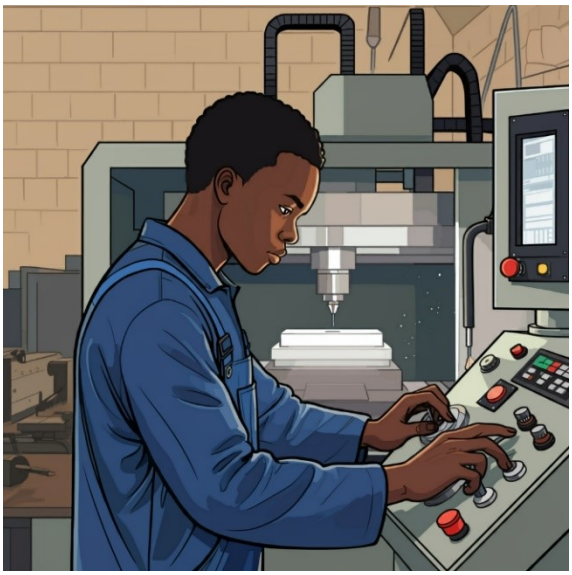
2.1.1. Prepare materials and tools

Gather PCB substrate, copper-clad board, cutting tools, and safety equipment. Ensure tools are appropriate for CNC machining and materials are clean and free from defects.



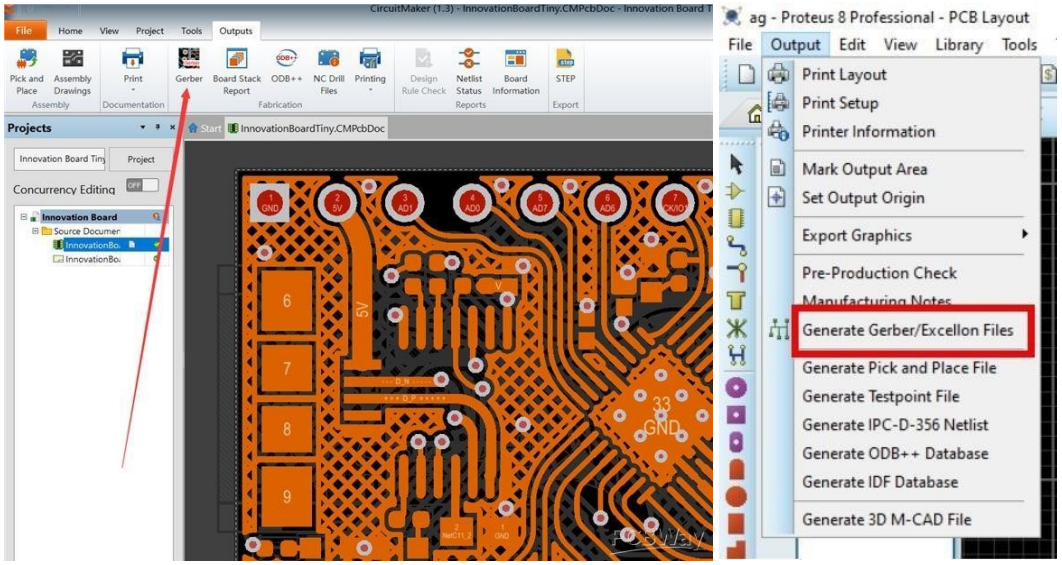
2.1.2. CNC machine setup

Calibrate the CNC machine, adjusting spindle speed, feed rate, and tool offsets. Secure the PCB substrate to the machine bed and align it accurately for precise machining.



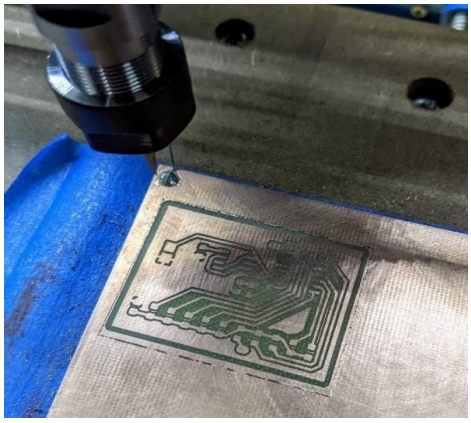
2.1.3. Upload Gerber file

Transfer the Gerber file from PCB design software to the CNC machine's control software. Verify file compatibility and check layer alignment before proceeding.



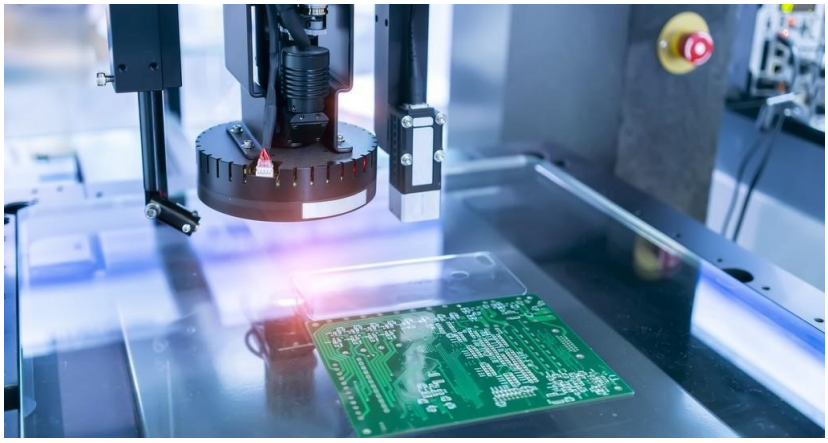
2.1.4. CNC Milling and Drilling

Execute CNC milling and drilling operations as per the Gerber file specifications. Mill PCB traces and drill holes for component placement, ensuring accuracy and adherence to design requirements.



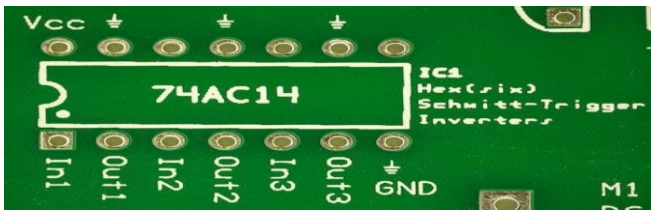
2.1.5. PCB inspection and cleaning

Inspect the milled PCB for quality, checking for accuracy, completeness, and any defects. Clean the PCB thoroughly to remove debris and residues from machining.



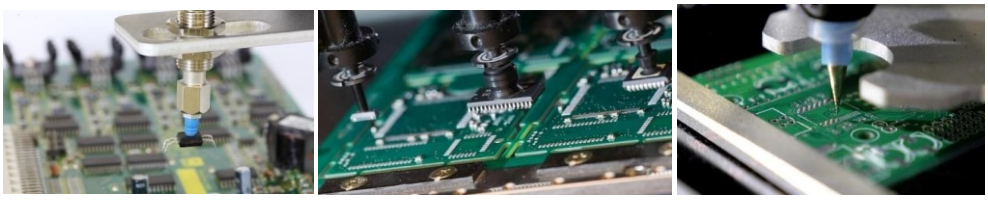
2.1.6. Silkscreen Placement

Apply silkscreen markings onto the PCB surface using appropriate printing techniques. Ensure alignment with the circuit layout and legibility for component identification.



2.1.7. Component placement

Place electronic components onto the PCB according to the schematic diagram. Ensure correct orientation and alignment for proper soldering and functionality.



2.1.8. Test the circuit

Perform functional testing of the assembled PCB to verify circuit operation. Use testing equipment like multimeters or oscilloscopes to check voltages, currents, and signal integrity. Troubleshoot any issues and make necessary adjustments for optimal performance.

2.2. Printing using Chemicals

2.2.1. Preparing the Copper Board

Clean the copper-clad board thoroughly to remove any dirt, grease, or oxidation. Ensure a smooth and uniform surface for proper adhesion of the PCB design.

2.2.2. Transferring the PCB Design

Print the PCB design onto transfer paper using a laser printer, ensuring accurate scaling and alignment. Place the printed design onto the copper board and apply heat and pressure to transfer the toner onto the board.

2.2.3. Removing the Paper

Carefully peel off the transfer paper from the copper board, ensuring that the toner adheres securely to the board's surface. Inspect for any areas where the toner may not have transferred properly and touch up if necessary.

2.2.4. Etching Process

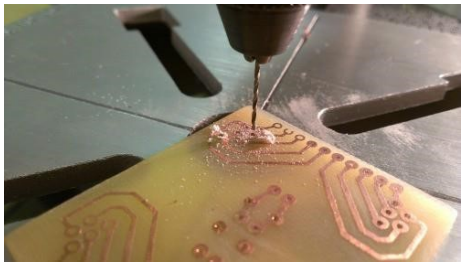
Immerse the copper board in an etching solution, such as ferric chloride or ammonium persulfate, to remove the exposed copper and reveal the circuit traces. Agitate the solution periodically to ensure even etching and monitor the process closely to prevent over-etching.

2.2.5. Cleaning and Finishing

Rinse the etched PCB thoroughly with water to remove any etchant residue. Neutralize the board in a solution drop them into a basin with cold water to stop the etching process. Dry the board completely before proceeding to the next steps

2.2.6. Drilling and Assembling

Drill holes in the PCB for component mounting using a suitable drill bit size. Ensure precise hole placement and alignment with the PCB design. Assemble electronic components onto the PCB according to the schematic diagram and solder them in place.



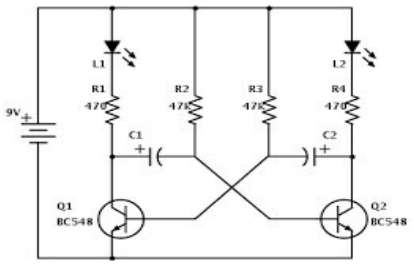
2.2.7. Circuit testing

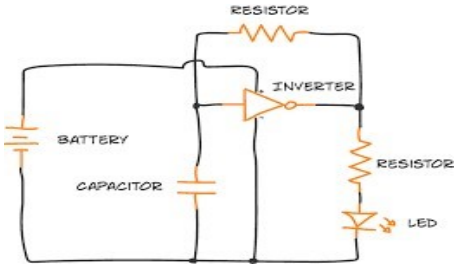
Perform functional testing of the assembled PCB to verify circuit operation. Apply power to the circuit and use testing equipment such as multimeters or oscilloscopes to check for proper voltages, currents, and signal integrity. Troubleshoot any issues and make necessary adjustments for optimal performance.

2.3. Documentation

2.3.1. Schematic Drawings

Create comprehensive schematic drawings illustrating the circuitry of the electronic design. Include symbols representing components, connections, and annotations detailing component values, reference designators, and signal names.





2.3.2. Bill of Material

Compile a detailed Bill of Materials listing all components required for the electronic assembly. Include information such as part numbers, descriptions, quantities, manufacturers, suppliers, and unit costs to facilitate procurement and assembly processes.

Single pane Window 500mm x 500mm						
BoM level	Part #	Description	Qty	Units	Unit Cost	Cost
1	756	Window framing	1	4	$3.00	$12.00
1	95	Brackets	1	4	$0.75	$3.00
1	PRS045	Rubber seal	2	metre	$0.50	$1.00
2	342	Glass pane	1	1	$9.50	$9.50
2	LB8579	Safety label	1	1	$0.10	$0.10
3	GH098	Hinges	2	1	$2.25	$4.50
3	GS664	Screws	8	10	$4.95	$3.96
3	587	Latch	1	1	$2.20	$2.20
3	588	Latch hook	1	1	$0.88	$0.88
4	GS660	Screws for latch and hook	6	10	$4.95	$2.97
5	812	Protective wrap	1.5	metre	$0.65	$0.98
6	XYZ123	Cardboard box 600mm x 600mm	1	1	$1.00	$1.00
6	LB7487	Box label barcode	1	1	$0.10	$0.10
		Total number parts	27.5		Total costs	$42.19

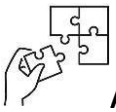
2.3.3 PCB Specification information

Document PCB specifications detailing key parameters such as board dimensions, layer stackup configuration, copper thickness, substrate material, and surface finish. Include additional information regarding tolerances, impedance requirements, and design constraints to ensure accurate fabrication and assembly.

PCB Specification	
PCB Name	BokTech PCB v0.1
PCB Material	(Rigid)FR—4
PCB Dimension (mm)	100*50
PCB Layers	2
PCB Thickness	1.6mm
Copper Weight	1oz
PCB Colour	Green
Silkscreen	White
Surface Finish	Hasl-lead free
Via solder mask	Solder mask cover
Comment:	common specification is OK

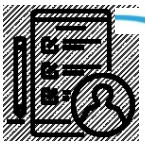
Points to Remember

- To print and assemble PCB for embedded system hardware using CNC, follow these steps:
  - Prepare materials and tools
  - CNC machine setup
  - Upload Gerber file
  - CNC milling and drilling
  - PCB inspection and cleaning
  - Silkscreen placement
  - Component placement
  - Test the circuit
- To print and assemble PCB for embedded system hardware using chemicals, follow these steps:
  - Preparing the Copper Board
  - Transferring the PCB Design
  - Removing the Paper
  - Etching
  - Cleaning and Finishing
  - Drilling and Assembling
  - Circuit testing



Application of learning 2.5.

You are a technician at ABC Innovations, and your team has designed an Automatic Smart Home Appliance Control System. This system will automate home functions such as lighting, security, climate control, and appliance management. After completing the PCB design for the embedded system hardware, it is now your responsibility to oversee the printing and assembly process to bring the design into a functional circuit board.



Learning outcome 2 end assessment

Theoretical assessment

1. **Complete the following using: schematics, layouts, KiCad, simulation, optimization**.

a. PCB design software is used to create …….. and …… for printed circuit boards.

b. A key feature of CAD software for PCB design is the ability to perform …….. and …… of circuits.

c. …….. is a commonly used PCB CAD software that is open-source and supports a variety of platforms.

2. Answer by true or false

   a. Circuit simulation requires selecting a simulation tool.

   b. Running the simulation is done before specifying simulation settings.

   c. Analyzing results is the final step in circuit simulation.

   d. The evaluation of results comes after running the optimization process.

   e. Optimization parameters are specified after running the optimization process.

3. Choose the correct answer

   What is the primary purpose of interpreting a schematic diagram in PCB design?

   A. To determine the size of the PCB

   B. To identify the electronic components and their connections

   C. To select the color of the PCB

   D. To arrange the components in 3D modeling

4. What is the main objective of creating a PCB prototype?

a. To test the electrical performance and functionality before mass production

b. To finalize the color scheme of the PCB

c. To check how well the PCB fits in the packaging

d. To minimize the number of vias in the design

5. Arrange correctly the following steps of printing PCB using chemicals.

   A. Preparing the Copper Board

   B. Removing the Paper

   C. Cleaning and Finishing

   D. Transferring the PCB Design

   E. Drilling and Assembling Circuit testing

   F. Etching

6. Identify six (6) steps of printing PCB using Printing using CNC.

7. Match each circuit schematic diagram design task with its corresponding description:

Task	**Description**
1. Creation of Block Diagrams	A. Ensure that the schematic follows standards and best practices
2. Selection of Components and Footprint	B. Name the components and show the

Assignment	connection paths in the schematic
3. Follow Design Guidelines	C. Review and check the schematic for errors or inconsistencies
4. Label Components and Signals	D. Develop a high-level diagram showing how the main components interact
5. Validate and Review	E. Select the right components and their physical representations on the PCB
6.Determining the connectivity and schematic software	

Practical assessment

ZGB Company located in Muhanga district, is developing the embedded system hardware for a smart home appliance control device, your role as an embedded system developer involves designing and developing that embedded system. This includes selecting appropriate PCB Electronic Design Software (CAD software), designing detailed circuit schematic diagrams, running circuit simulations, optimizing the design for efficiency, creating a PCB layout, printing the PCB, and assembling the final hardware for the device.

END

References

All About Circuits – PCB Design. (n.d.). Retrieved from AllAboutCircuits.com: https://www.allaboutcircuits.com

Assembly, P. D. (n.d.). EDN Network. Retrieved from https://www.edn.com/pcb-design-tips-for-assembly/

Camenzind, H. (2005). Designing Analog Chips. In H. Camenzind.

Circuit Cellar – Embedded Design. (n.d.). Retrieved from CircuitCellar.com: https://circuitcellar.com

Givargis, V. a. (October 2001). Embedded System Design. In V. a. Givargis.

How to Assemble a PCB. (n.d.). Retrieved from Newbury Electronics: https://www.newburyelectronics.co.uk/blog/how-to-assemble-a-pcb

Introduction, A. U. (n.d.). Wikipedia – Embedded System. Retrieved from Wikipedia.org: https://en.wikipedia.org/wiki/Embedded_system

Jones, D. L. (2011). PCB Design Tutorial. In D. L. Jones.

Lab, G. –S. (n.d.). Retrieved from GeeksforGeeks.org: https://www.geeksforgeeks.org

Marwedel, P. (2021). Embedded Systems Foundations of Cyber-Physical Systems, and the Internet of Things".

Microcontroller Fundamentals for Engineers and Scientists. (2021). In S. F. Pack, Microcontroller Fundamentals for Engineers and Scientists.

Microprocessor Tutorial. (n.d.). Retrieved from Javatpoint.com: https://www.javatpoint.com/microprocessor

Microprocessors, M. v. (n.d.). Retrieved from IBM.com: https://www.ibm.com

N. Senthil Kumar, M. S. (2016). Microprocessors and Microcontrollers.

Organizing Electronics Labs for Development. (n.d.). Retrieved from ResearchGate.net: https://www.researchgate.net

PCB, H. t. (n.d.). Retrieved from www.Instructables.com: https://www.instructables.com/How-to-Create-a-PCB/

Scheible, J. L. (2024). Fundamentals of Layout Design for Electronic Circuits. In J. L. Scheible.

Tinkercad – Circuits. (n.d.). Retrieved from Tinkercad.com: https://www.tinkercad.com/circuits

Wilmshurst, T. (2019). Designing Embedded Systems with PIC Microcontrollers.

www.Wikipedia.org/microcontroller. (n.d.). Retrieved from www.Wikipedia.org: https://en.wikipedia.org/wiki/Microcontroller

Indicative Contents
3.1 Assessing Hardware Parts and Peripherals Specifications **3.2 Interconnection of Hardware System Parts** **3.3 Installation of Required Peripherals** **3.4 Testing of Embedded System Hardware** **3.5 Documentation of Embedded System Hardware**

Key Competencies for Learning Outcome 3: Integrate Embedded System Hardware

Knowledge	Skills	Attitudes
• Description of hardware parts and peripherals specifications • Description of testing techniques	• Assessing hardware parts and peripherals • Interconnecting embedded system hardware parts • Testing embedded system hardware • Installing required peripherals for embedded system hardware • Documenting embedded system hardware	• Having Precision • Being Attentive • Having self-confident • Having accountability • Respecting time • Being patient • Having self-motivation • Being organized • Being passionate • Having creativity

Duration: 30 hrs

Learning outcome 3 objectives:

By the end of the learning outcome, the trainees will be able to:

1. Describe correctly hardware parts and peripherals specifications for embedded system hardware
2. Assess clearly hardware parts and peripherals specifications for embedded system hardware
3. Interconnect correctly embedded system hardware parts according to the system design
4. Install correctly required peripherals for embedded system hardware
5. Describe correctly testing techniques for embedded system hardware
6. Test correctly embedded system hardware based on testing techniques
7. Document effectively embedded system hardware based on the work done.

Resources

Equipment	Tools	Materials
• Computer • Multimeter	• Arduino IDE • PCB simulation tools • PCB design tools • Testing tools	• Microcontroller or Microprocessor • Sensors and Actuators • Power Supply • Breadboard • Printed Circuit Board (PCB) • Connecting Wires • Electronic components

Indicative content 3.1: Assessing hardware parts and peripherals specifications

Duration: 5 hrs

Theoretical Activity 3.1.1: Description of hardware parts and peripherals specifications for embedded system hardware.

Tasks:

1: You are asked to answer the following questions:

What are specifications of the following hardware parts and peripherals?

a. Microcontroller or Microprocessor specifications?
b. External storage.
c. Connectivity options.
d. Form factor and mechanical considerations.
e. Display and user interface.
f. Power Supply.
g. Sensors and Actuators.
h. Security Features.

2: Provide the answers for the asked questions and write them on flipchart/papers.

3: Present the findings/answers to the whole class.

4: For more clarification, read the key readings 3.1.1**.**

5: In addition, ask questions where necessary.

Key readings 3.1.1: Description of hardware parts and peripherals specifications for embedded system hardware.

1. Microcontroller or Microprocessor specifications

- **Performance:** Clock speed (MHz/GHz), number of cores, and architecture (e.g., ARM, x86).
- **Memory:** On board RAM and Flash memory.
- Power Consumption: Operating voltage and current draw, especially in active and sleep modes.
- **Peripherals:** Built-in features like ADC/DAC, PWM, timers, UART, SPI, I2C, and GPIO pins.
- **Development Ecosystem:** Availability of development boards, software tools, libraries, and community support.

2. External Storage Interfaces specifications

- **Types:** Support for SD cards, USB storage, or external flash.
- **Speed:** Data transfer rates for read and write operations.
- **Capacity:** Maximum supported storage size.

- **File System Support:** Compatibility with file systems like FAT32, exFAT, or NTFS.
- **Durability:** Consider industrial-grade storage for harsh environments.

3. Connectivity Options

- **Wired:** Ethernet, USB, CAN bus, RS232/RS485.
- **Wireless:** Wi-Fi, Bluetooth, Zigbee, LoRa, NB-IoT, LTE/5G.
- **Antennas:** Built-in vs. external antennas and their impact on range and performance.
- **Protocols:** Support for standard protocols like TCP/IP, MQTT, HTTP, etc.

4. Form Factor and Mechanical Considerations

- **Size:** Dimensions of the PCB or module.
- **Mounting:** Options for mounting (e.g., surface mount, through-hole).
- **Connectors:** Type and placement of connectors (e.g., headers, sockets, edge connectors).
- **Durability:** Materials used ruggedness, and compliance with standards (e.g., IP ratings for water/dust resistance).

5. Display and User Interface specifications

- **Type:** LCD, OLED, e-ink, touch screens.
- **Size and Resolution:** Diagonal size in inches and resolution (e.g., 800x480 pixels).
- **Interface:** Connection method (e.g., SPI, I2C, parallel).
- **Touch Capability:** Capacitive vs. resistive touch, multi-touch support.
- **UI Libraries:** Availability of graphical libraries and support for UI development.

6. Power Supply

- **Voltage Range:** Input voltage range and tolerance.
- **Power Consumption:** Active and standby power requirements.
- **Power Sources:** Compatibility with batteries, AC adapters, PoE, solar panels.
- **Regulation:** Built-in voltage regulators and their efficiency.
- **Protection:** Over-voltage, under-voltage, and short-circuit protection features.

7. Sensors and Actuators

- **Types of Sensors:** Temperature, humidity, pressure, light, motion, proximity, etc.
- **Accuracy and Precision:** Measurement accuracy, precision, and response time.
- **Interface:** Analog vs. digital (e.g., I2C, SPI, UART).
- **Range:** Operating range and limits of the sensor.
- **Actuators:** Motors (DC, stepper, servo), relays, solenoids, etc., and their control requirements.

8. Security Features

- **Encryption:** Hardware support for encryption standards (e.g., AES, RSA).
- **Authentication:** Secure boot, hardware-based key storage, and secure ID.
- **Tamper Detection:** Mechanisms for detecting physical tampering.

- **Firmware Updates:** Secure firmware update mechanisms (e.g., OTA updates).
- **Compliance:** Adherence to security standards and certifications (e.g., FIPS, GDPR).

Practical Activity 3.1.2: Assessing hardware parts and peripherals specifications for embedded system hardware

Task:

1: Read key reading 3.1.2

2: Referring to key reading 3.1.2, you are sked to assess hardware parts and peripherals specifications for embedded system hardware.

3: Present your work to the trainer and whole class.

4: Ask clarification where necessary.

Key readings 3.1.2: Assessing hardware parts and peripherals specifications for embedded system hardware

1. **Steps of assessing hardware parts and peripherals specifications for embedded system hardware**

1.1. Define System Requirements

Clearly outline the specific functionalities and performance requirements of your embedded system. This includes factors such as processing power, memory usage, power consumption, communication capabilities, and environmental conditions.

1.2. Evaluate Component Specifications

Carefully examine the technical specifications of each component, including:

Functionality: Does the component meet the required functionalities?

Performance: Consider factors like processing speed, power consumption, and data transfer rates, and memory usage.

Compatibility: Ensure compatibility with other system components and interfaces.

Reliability and Durability: Evaluate the component's longevity and resistance to environmental factors.

Cost and Maintenance: Assess the initial cost and ongoing maintenance expenses.

Security: Consider the component's security features and compliance with relevant standards.

1.3. Test and Validate

If possible, obtain samples of components and test them in your system to verify their performance and compatibility.

Conduct thorough testing under various operating conditions to ensure the components meet your system's requirements.

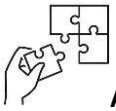
1.4. Document Your Findings

Create a detailed record of the selected components, including their specifications and reasons for choosing them.

This documentation will be helpful for future reference, troubleshooting, and maintaining the system.

Points to Remember

- There are many hardware parts and peripherals specifications, but the common specifications are the following: performance, size, and power requirements.
- To assess hardware parts and peripherals specifications for embedded system, follow these steps:
    - ✓ Define system requirements
    - ✓ Evaluate component specifications
    - ✓ Test and Validate
    - ✓ Document Your Findings



Application of learning 3.1

YXSS Solutions, located in Kigali city, Kicukiro District, is developing an advanced embedded system for smart home security system. As technician, you are requested to assess the specifications of various hardware components and peripherals to ensure they meet the system's requirements.

Indicative content 3.2: Interconnection of hardware system parts

Duration: 10 hrs

Practical Activity 3.2.1: Interconnecting hardware system parts for embedded system.

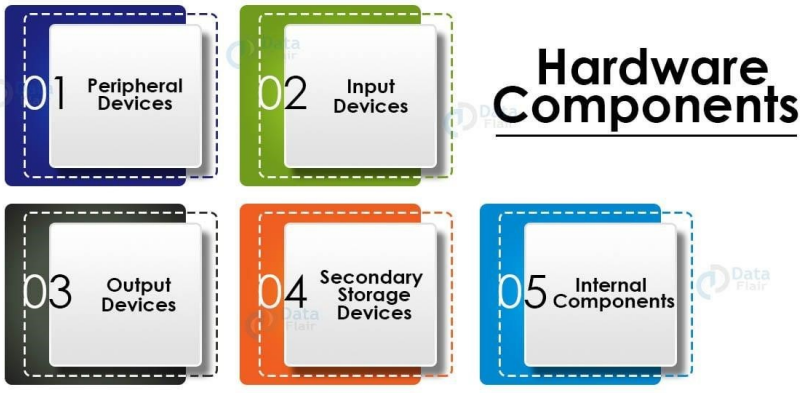
Task:

1: Read key reading 3.2.1
2: Referring to key reading 3.2.1, you are sked to interconnect hardware system parts for embedded system.
3: Present your work to the trainer and whole class.
4: Ask clarification where necessary.

Key readings 3.2.1.: Interconnecting hardware system parts for embedded system.

1. Prepare the Hardware Components

Ensure all hardware components are ready for integration, including microcontrollers, sensors, actuators, displays, and other peripherals. Verify that components are functioning properly and meet design specifications.

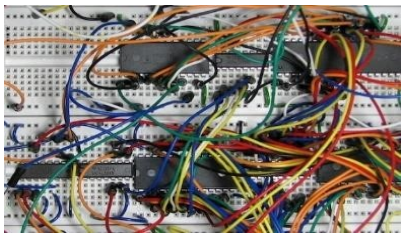


2. Perform Connector and Header Placement

Place connectors and headers on the PCB layout or hardware platform according to the interconnection requirements. Ensure proper alignment, spacing, and orientation to facilitate easy connection and disconnection of cables and peripherals.

3. Perform Wiring and Cabling (if applicable)

Wire and cable the hardware components together based on the interconnection diagram or schematic. Use appropriate cables, wires, and connectors to establish reliable electrical connections while minimizing signal interference and cross-talk.

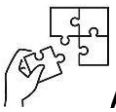


4. Perform Enclosure and Mounting

Enclose the hardware system in a suitable enclosure or chassis to protect components from environmental factors and provide mechanical support. Ensure proper mounting of PCBs, displays, connectors, and other components within the enclosure, considering accessibility for maintenance and serviceability.

Points to Remember

- To interconnect hardware system parts for embedded system, follow these steps:
  - ✓ Prepare the Hardware Components
  - ✓ Performing Connector and Header Placement
  - ✓ Perform Wiring and Cabling
  - ✓ Performing enclosure and Mounting



Application of learning 3.2

ABC Solutions is a company located in Kigali City, Kicukiro district. Engineering team is tasked with developing an advanced embedded system for automatic domestic lighting. As embedded system developer, you are tasked to interconnect the various hardware components that will make up this system.

Indicative content 3.3: Installation of required peripherals.

Duration: 5 hrs

Practical Activity 3.3.1: Installing required peripherals for embedded system.

Task:

1: Read key reading 3.3.2

2: Referring to key reading 3.3.2, you are sked to interconnect hardware system parts for embedded system.

3: Present your work to the trainer and whole class.

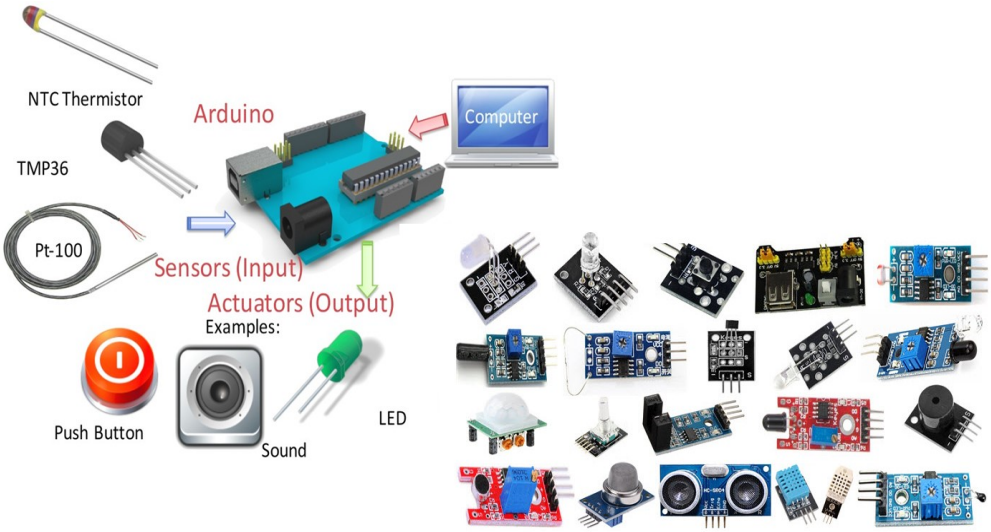
4: Ask clarification where necessary.

Key readings 3.3.2.: Installing required peripherals for embedded system.

Steps to install required peripherals

1. **Identify Required Peripherals**

- Identify the necessary peripherals (e.g., sensors, actuators, communication modules).



- Create a comprehensive list of required peripherals along with their specifications and functions.

2. **Verify Compatibility**

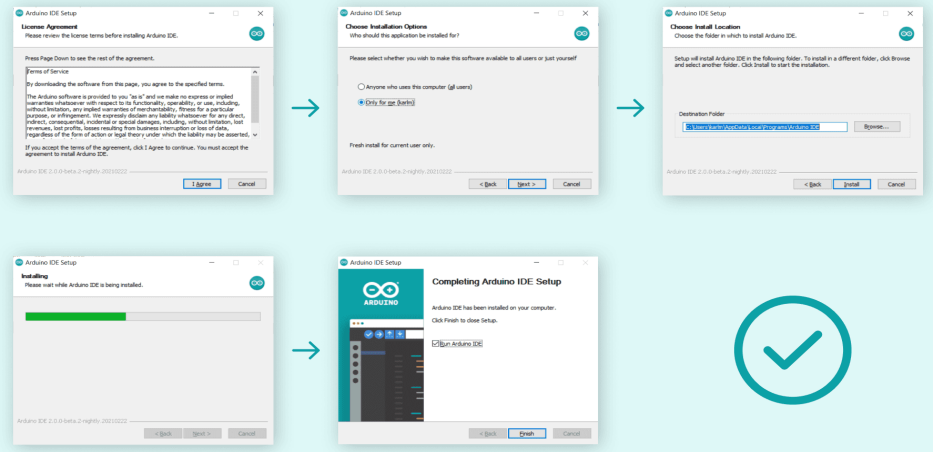
- Ensure that the peripherals are compatible with the microcontroller's interfaces (e.g., I2C, SPI, UART).
- Check that the voltage and power requirements of the peripherals match the

microcontroller's capabilities.

- Confirm that the peripherals fit within the physical space available and can be mounted securely.
- Verify that software drivers are available and compatible with the microcontroller's development environment.

2. Flashing the .hex File to Microcontroller

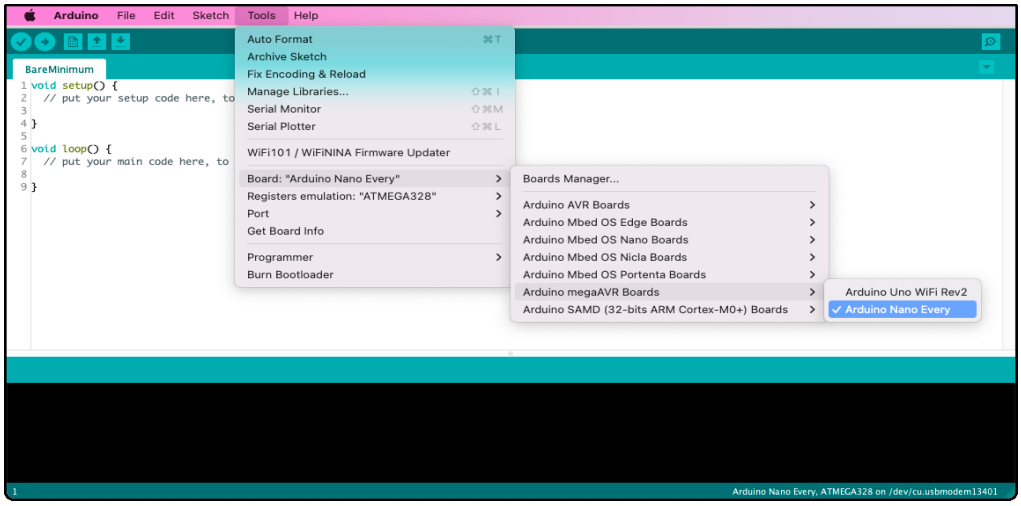
- **Install Programming Software**: Install the necessary programming software on your computer to facilitate the flashing process.



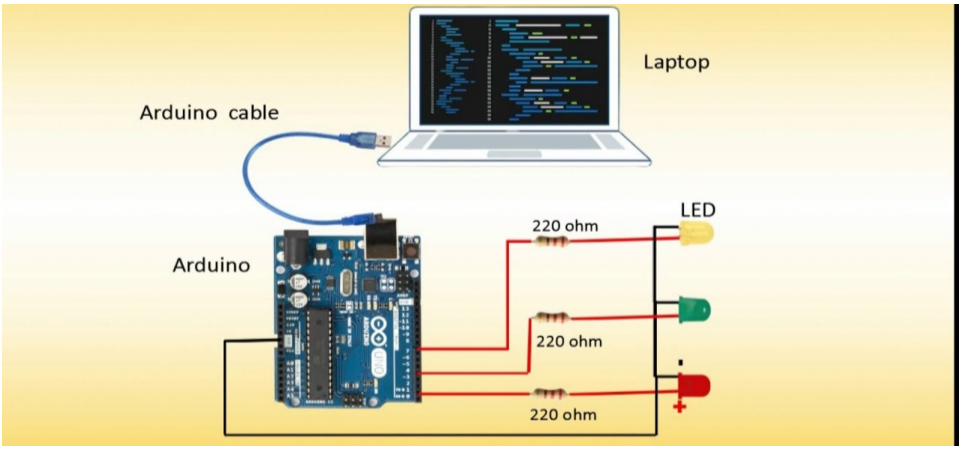
- **Prepare the .hex File**: Ensure the .hex file is correctly generated from the development environment, with the necessary code and configurations.

```
ColourPicker
// Michael Klements
// RGB Colour Picker
// 3 January 2019
// www.the-diy-life.com

#include <LiquidCrystal.h>  //Include the required libraries
#include <Wire.h>
#include "Adafruit_TCS34725.h"

byte gammatable[256];      //Table to convert measured RGB values in more realistic visulisation colour on the RGB LED

Adafruit_TCS34725 tcs = Adafruit_TCS34725(TCS34725_INTEGRATIONTIME_50MS, TCS34725_GAIN_4X);  //Setup the colour sensor through Adafruit library

LiquidCrystal lcd(15, 14, 16, 4, 5, 8, 7);  //Assign LCD pins

int pinLED = A0;      //Assign pins for the colour picker LED, push button and RGB LED
int pinButton = A1;
int redLED = 6;
int greenLED = 9;
int blueLED = 10;

#define commonAnode false   //Set up RGB LED common cathode or anode for gammatable - thanks PhilB for this gamma table

void setup()
{
  pinMode (pinLED, OUTPUT);     //Assign output pins
  pinMode (pinButton, INPUT);
  pinMode (redLED, OUTPUT);
  pinMode (greenLED, OUTPUT);
  pinMode (blueLED, OUTPUT);
  lcd.begin(16,2);              //Defines the number of characters and rows on our LCD
  lcd.clear();                  //Clear the screen
  lcd.setCursor(0,0);           //Set the cursor to first character, first row
  lcd.print("Colour Picker");   //Display this text
  analogWrite(redLED, 255);     //Routine to quickly fade change the LED from red to green to blue, just a visual effect
  for (int i=0; i<=255 ; i++)
  {
```

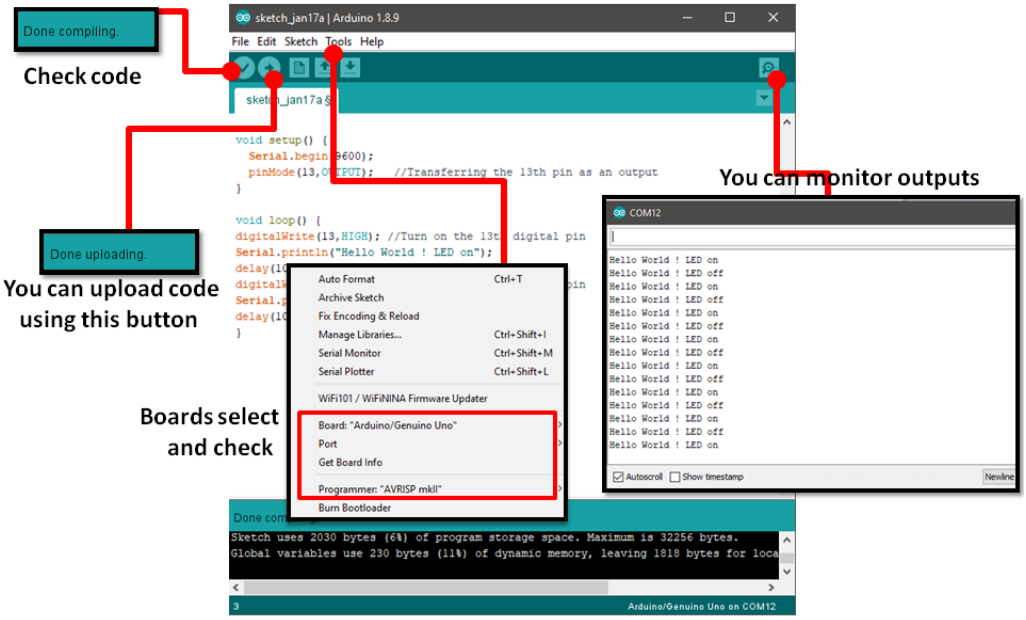
- **Select Programming Tool**: Choose an appropriate programming tool (e.g., USB programmer, in-system programmer) compatible with the microcontroller.



- **Connect Programmer to Microcontroller**: Connect the programming tool to the microcontroller using the correct interface (e.g., ISP header, JTAG).

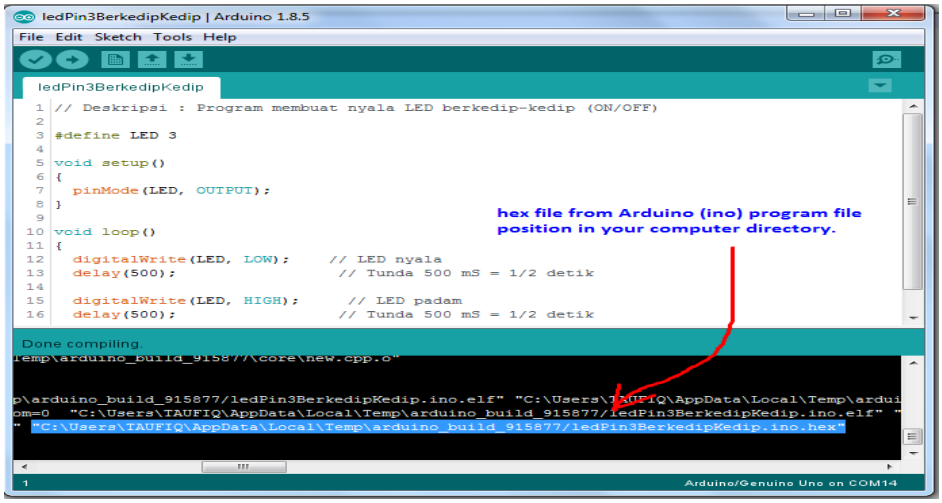


- **Load .hex File**: Load the .hex file into the programming software.



- **Flash the .hex File**: Execute the flashing process to transfer the .hex file from

the computer to the microcontroller.



- **Verification**: Verify that the .hex file has been successfully flashed by running diagnostic tests or checking the microcontroller's functionality.

- **Testing Peripherals**: After flashing, test each peripheral to ensure it interacts correctly with the microcontroller and performs as expected.

Points to Remember

- To installing required peripherals for embedded system, follow these steps:
  - ✓ Identify Required Peripherals
  - ✓ Verify Compatibility
  - ✓ Flashing the .hex file to microcontroller

Application of learning 3.3

KZT Solutions, located in Kigali city, Kicukiro District, is developing an advanced embedded system for automatic object detection. The system is designed to detect objects using various sensors and provide corresponding feedback. The engineering team is focused on developing the hardware. As a technician, your task is to install and connect the peripherals required for the automatic object detection system.

Indicative content 3.4: Testing of embedded system hardware

Duration: 5 hrs

Theoretical Activity 3.4.1: Description of embedded system hardware testing techniques

Tasks:

1: You are asked to answer the following questions:

i. List four embedded system hardware testing techniques.
ii. Explain those four testing technique on the question above?

2: Provide the answers for the asked questions and write them on flipchart/papers.

3: Present the findings/answers to the whole class.

4: For more clarification, read the key readings 3.4.1.

5: In addition, ask questions where necessary.

Key readings 3.4.1.: Description of embedded system hardware testing techniques

Embedded system hardware testing techniques

1. Functional Test

1.1. Purpose

Functional testis performed to verify that all hardware components and the system as a whole operate correctly according to design specifications.

1.2. Methods

Unit testing: Testing individual components or modules of the hardware.

Integration testing: Testing how multiple components interact with each other.

System testing: Testing the entire system as a whole.

1.3. Examples

- For a microcontroller, testing its ability to execute instructions, perform calculations, and communicate with peripherals.
- For a sensor, testing its accuracy, sensitivity, and response time.

2. Performance Test

2.1. Purpose

Performance test is performed to evaluate the hardware's speed, throughput, and responsiveness under various conditions.

2.2. Methods

Benchmarking: Comparing the hardware's performance to known

standards or benchmarks.

Load testing: Simulating heavy workloads to assess the hardware's performance under stress.

Stress testing: Pushing the hardware to its limits to identify potential bottlenecks or failures.

2.3. Examples

- For a microcontroller, measuring its clock speed, instruction execution time, and memory access latency.
- For a network interface, testing its data transfer rate, packet loss, and latency.

3. Power Consumption and Efficiency Test

3.1. Purpose

Power consumption and efficiency test is performed to measure the system's power usage and ensure it operates efficiently under various states (idle, active, or standby).

3.2. Methods

Power measurement: Using specialized equipment to measure the power consumed by the hardware.

Efficiency calculation: Calculating the ratio of output power to input power.

Thermal analysis: Evaluating the hardware's temperature and cooling requirements.

3.3. Examples

- For a battery-powered device, measuring its battery life and standby power consumption.
- For a high-performance processor, testing its power consumption under various workloads.

4. Security Test

4.1. Purpose

Security test is used to identify and address potential security vulnerabilities in the hardware.

4.2. Methods

Vulnerability scanning: Using automated tools to detect known vulnerabilities.

Penetration testing: Simulating attacks to assess the hardware's security defences.

Secure coding practices: Ensuring that the hardware's firmware and software are developed securely.

4.3. Examples

- For a wireless device, testing its resistance to unauthorized access and data interception.

- For a microcontroller, verifying that its security features, such as encryption and authentication, are implemented correctly.

Practical Activity 3.4.2: Testing embedded system hardware

Task:

1: Read key reading 3.4.2.

2: Referring to key reading 3.4.2, you are sked to test embedded system hardware.

3: Present your work to the trainer and whole class.

4: Ask clarification where necessary.

Key readings 3.4.2.: Testing embedded system hardware

Steps of testing embedded system hardware

1. **Define Testing Requirements**

- **Identify Objectives:** Clearly outline what you want to achieve with the testing. This might include verifying functionality, performance, power consumption, security, or compatibility.
- **Determine Scope:** Define the boundaries of the testing. Will you be testing the entire system or specific components?
- **Prioritize Tests:** Based on the criticality and risk associated with different aspects, prioritize the tests to ensure efficient allocation of resources.

2. **Create a Test Plan**

- **Develop Test Cases:** Write detailed test cases that cover various scenarios and inputs to ensure thorough testing.
- **Define Test Data:** Prepare the necessary data and inputs for each test case.
- **Assign Responsibilities:** Assign roles and responsibilities to individuals involved in the testing process.
- **Schedule Testing:** Create a timeline for conducting the tests, considering dependencies and resource availability.

3. **Set Up the Test Environment**

- **Hardware and Software:** Assemble the required hardware components and install necessary software tools.
- **Test Fixtures:** If needed, create or procure test fixtures to simulate real-world conditions or provide controlled environments.

- **Instrumentation:** Set up instruments like oscilloscopes, logic analyzers, and power meters to measure and analyze system behavior.

4. **Conduct Functional Testing**

- **Verify Functionality:** Test each feature and component of the system to ensure it operates as expected.
- **Input Validation:** Validate the system's response to valid and invalid inputs.
- **Boundary Value Testing:** Test the system's behavior at the boundaries of input ranges.
- **Error Handling:** Verify how the system handles errors and exceptions.

5. **Perform Performance Testing**

- **Measure Response Time:** Evaluate how quickly the system responds to different inputs and workloads.
- **Throughput:** Determine the maximum amount of data the system can process within a given time.
- **Resource Utilization:** Monitor CPU usage, memory consumption, and other resource metrics to identify bottlenecks.

6. **Conduct Power Consumption Testing**

- **Measure Power Draw:** Measure the power consumed by the system under various operating conditions.
- **Identify Power-Hungry Components:** Determine which components contribute significantly to power consumption.
- **Optimize Power Usage:** Explore techniques to reduce power consumption without compromising performance.

7. **Security Testing**

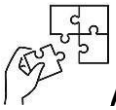
- **Vulnerability Assessment:** Identify potential security vulnerabilities in the system, such as weaknesses in software, hardware, or communication protocols.
- **Penetration Testing:** Simulate attacks to assess the system's resilience to malicious activities.
- **Secure Coding Practices:** Ensure that the system's software adheres to secure coding guidelines.

8. **Environmental Testing**

- **Temperature and Humidity:** Test the system's operation in extreme temperature and humidity conditions.
- **Vibration and Shock:** Evaluate the system's resistance to mechanical stress.
- **Electromagnetic Interference (EMI):** Assess the system's susceptibility to EMI and its potential to emit EMI.

Points to Remember

- Embedded system hardware testing techniques includes functional, performance, power, and security testing. These tests ensure correct operations, speed under stress, energy efficiency, and resistance to vulnerabilities.
- When performing embedded system hardware testing, follow those steps:
    - ✓ Define Testing Requirements
    - ✓ Create a Test Plan
    - ✓ Set Up the Test Environment
    - ✓ Conduct Functional Testing
    - ✓ Perform Performance Testing
    - ✓ Conduct Power Consumption Testing
    - ✓ Security Testing
    - ✓ Environmental Testing



Application of learning 3.4

XYZ Solutions, located in Muhanga District, is working on the development of an advanced embedded system for home automation. This system is designed to control various household devices, such as lighting, temperature, security cameras, and door locks, remotely via a central control unit. The engineering team has completed the hardware design. As a technician, you have been assigned the responsibility to test the hardware system to ensure its proper functionality.

Indicative content 3.5: Documentation of embedded system hardware

Duration: 5 hrs

Theoretical Activity 3.5.1: Description of embedded system hardware documentation

Tasks:

1: You are asked to answer the following questions:

i. What are the main section of embedded system hardware documentation?

ii. What can be included in those sections?

2: Provide the answers for the asked questions and write them on flipchart/papers.

3: Present the findings/answers to the whole class.

4: For more clarification, read the key readings 3.5.1.

5: In addition, ask questions where necessary.

Key readings 3.5.1: Description of embedded system hardware documentation

1. Introduction

Documentation of Embedded System Hardware is a comprehensive written description and record of all aspects of the hardware components of an embedded system. It serves as a reference guide for developers, engineers, testers, and anyone involved in the lifecycle of the embedded system. The documentation is crucial for understanding how the hardware is designed, how components interact, and how to troubleshoot, maintain, or upgrade the system.

2. Main sections of embedded system hardware documentation

2.1. Hardware Specifications

- **Processor:** Type, speed, architecture, and instruction set.
- **Memory:** RAM size, type, and ROM size.
- **I/O:** Number and types of input/output ports.
- **Peripherals:** Details of any additional components such as sensors, actuators, or displays.
- **Power Supply:** Voltage, current, and regulation requirements.

2.2. System Architecture and Design

- **Block Diagram:** A visual representation of the system's components and their interconnections.
- **Schematic Diagram:** A detailed drawing showing the electrical connections

between components.

- **PCB Layout:** The physical arrangement of components on a printed circuit board**.**

2.3. Components and Materials

- **Component List:** A detailed list of all components used in the system, including part numbers, manufacturers, and specifications.
- **Materials List:** A list of materials used in the system, such as PCB material, connectors, and enclosures.

2.4. Power and Communication

- **Power Supply:** Details of the power supply circuit, including voltage regulators, filters, and protection components.
- **Communication Interfaces:** Information about communication protocols used (e.g., UART, I2C, SPI) and their connections.

2.5. Connectivity and Pinout

- **Pinout Diagram:** A diagram showing the pin assignments for each component, including power, ground, and signal lines.
- **Connector Specifications**: Details of connectors used, including type, pin count, and mating specifications.

2.6. Operation Instruction

- **Functional Description:** A detailed explanation of the system's functionality and how it operates.
- **User Interface:** Information about how the system interacts with the user, including input and output methods.
- **Operating Procedures:** Step-by-step instructions on how to operate the system.

2.7. Troubleshooting Instruction

- **Common Issues:** A list of common problems that may occur and their potential causes.
- **Troubleshooting Steps:** Detailed instructions on how to diagnose and resolve issues.
- **Error Codes:** A list of any error codes that may be generated by the system and their meanings.

2.8. Assembly Instructions

- **Component Placement:** Instructions on how to place components on the PCB or other assembly structures.
- **Soldering Guidelines:** Guidelines for soldering components to the PCB, including soldering techniques and safety precautions.
- **Assembly Sequence:** The recommended order for assembling the system, including any specific assembly procedures.

Practical Activity 3.5.2: Documenting embedded system hardware

Task:

1: Read key reading 3.5.2.
2: Referring to key reading 3.5.2, you are sked to document embedded system hardware.
3: Present your work to the trainer and whole class.
4: Ask clarification where necessary.

Key readings 3.5.2.: Documenting embedded system hardware

Steps of documenting embedded system hardware

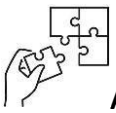
1. **Define the structure of documentation**
   Begin by outlining the structure of the documentation. Define key sections and organize the content logically for ease of reference.
2. **Record hardware specifications**
   Provide a comprehensive description of the hardware specifications, including technical details like processor type, memory, storage, and peripheral components.
3. **Create system architecture and design**
   Document the overall system architecture, showing how different hardware components interconnect. Include diagrams to illustrate the design of the system.
4. **Create a detailed Bill of Materials (BOM)**
   List all components and materials used in the embedded system in a Bill of Materials (BOM), including part numbers, manufacturers, and quantity.
5. **Record power and communication details**
   Explain the power requirements and the communication protocols supported by the system. Specify voltage levels, power supplies, and connectivity interfaces.
6. **Create connectivity and pinout diagrams**
   Provide clear diagrams showing the connectivity between various hardware components and the pinout configurations for any ports or connectors.
7. **Prepare operation instructions**
   Include detailed instructions on how to operate the embedded system, from initial setup to routine usage, ensuring that the instructions are user-friendly.
8. **Record Troubleshooting Guidelines**
   Provide a section dedicated to troubleshooting common hardware issues,

offering step-by-step guidance on identifying and resolving problems.

9. **Develop Assembly Instructions**
   Include detailed assembly instructions that guide the reader through the process of assembling the hardware, including diagrams and parts placement.
10. **Develop a comprehensive safety precautions**
    Clearly define safety guidelines and precautions to ensure the safe handling and operation of the embedded system, emphasizing any potential hazards

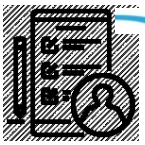
Points to Remember

- The documentation for embedded system hardware typically includes details on hardware specifications, system design, components, power, connectivity, operation, troubleshooting, and assembly.
- When performing embedded system hardware documentation, follow those steps:
  - ✓ Define the structure of documentation
  - ✓ Record hardware specifications.
  - ✓ Create system architecture and design
  - ✓ Create a detailed Bill of Materials (BOM)
  - ✓ Record power and communication
  - ✓ Create connectivity and pinout diagrams
  - ✓ Prepare operation instructions
  - ✓ Record troubleshooting guidelines
  - ✓ Develop assembly Instructions
  - ✓ Develop a comprehensive safety precautions



Application of learning 3.5.

KKZ Solutions, a company located in Kigali City, Gasabo district. The company has developed a computerized door access control system that uses fingerprint recognition for authentication. As an embedded system hardware technician, you are requested to document that embedded system hardware.



Learning outcome 3 end assessment

Theoretical assessment

1. Answer by true or false
    a. A battery is the only way to power an embedded system.
    b. A microcontroller and a microprocessor are essentially the same thing.
    c. Touchscreens are the only type of user interface suitable for embedded systems.
2. What are some common input devices used in embedded systems?
3. How input / output devices are typically connected to embedded systems?
4. What considerations should be taken when selecting and installing input/output devices?
5. Match the following sections of documentation in Column B with their purpose in Column C and fill in Column A.

Answer	Sections	purpose
1. …….	1. Hardware Specifications	a. Offer instructions on how to fix common problems, including component replacement and reconfiguration steps.
2. …….	2. Troubleshooting Instruction	b. Provide detailed pinout diagrams for all connectors and headers, including pin functions and signal names.
3. …….	3. Connectivity and Pinout	c. Provide a detailed list of all components and materials used in the system.
4. …….	4. Bill of Materials (BOM)	d. Provide detailed information about the individual hardware components and their interconnections.

Practical assessment

XYZ Company Ltd, located in the Western Province, Ngororero District, specializes in producing various embedded system equipment, As an Embedded System Hardware technician at XYZ Company Ltd, you have been assigned the task of designing and integrating an embedded system that controls the lighting in a home based on motion detection. The system should turn on lights when someone enters a room and turn them off after a certain period of inactivity.

END

References

All About Circuits – PCB Design. (n.d.). Retrieved from AllAboutCircuits.com: https://www.allaboutcircuits.com

Assembly, P. D. (n.d.). EDN Network. Retrieved from https://www.edn.com/pcb-design-tips-for-assembly/

Camenzind, H. (2005). Designing Analog Chips. In H. Camenzind.

Circuit Cellar – Embedded Design. (n.d.). Retrieved from CircuitCellar.com: https://circuitcellar.com

Givargis, V. a. (October 2001). Embedded System Design. In V. a. Givargis.

How to Assemble a PCB. (n.d.). Retrieved from Newbury Electronics: https://www.newburyelectronics.co.uk/blog/how-to-assemble-a-pcb

Introduction, A. U. (n.d.). Wikipedia – Embedded System. Retrieved from Wikipedia.org: https://en.wikipedia.org/wiki/Embedded_system

Jones, D. L. (2011). PCB Design Tutorial. In D. L. Jones.

Lab, G. –S. (n.d.). Retrieved from GeeksforGeeks.org: https://www.geeksforgeeks.org

Marwedel, P. (2021). Embedded Systems Foundations of Cyber-Physical Systems, and the Internet of Things".

Microcontroller Fundamentals for Engineers and Scientists. (2021). In S. F. Pack, Microcontroller Fundamentals for Engineers and Scientists.

Microprocessor Tutorial. (n.d.). Retrieved from Javatpoint.com: https://www.javatpoint.com/microprocessor

Microprocessors, M. v. (n.d.). Retrieved from IBM.com: https://www.ibm.com

N. Senthil Kumar, M. S. (2016). Microprocessors and Microcontrollers.

Organizing Electronics Labs for Development. (n.d.). Retrieved from ResearchGate.net: https://www.researchgate.net

PCB, H. t. (n.d.). Retrieved from www.Instructables.com: https://www.instructables.com/How-to-Create-a-PCB/

Scheible, J. L. (2024). Fundamentals of Layout Design for Electronic Circuits. In J. L. Scheible.

Tinkercad – Circuits. (n.d.). Retrieved from Tinkercad.com: https://www.tinkercad.com/circuits

Wilmshurst, T. (2019). Designing Embedded Systems with PIC Microcontrollers.

www.Wikipedia.org/microcontroller. (n.d.). Retrieved from www.Wikipedia.org: https://en.wikipedia.org/wiki/Microcontroller

October, 2024