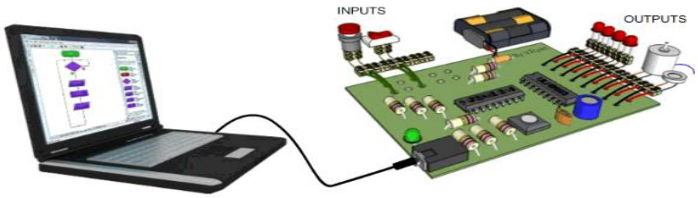


RQF LEVEL 5

TRADE: ELECTRONIC SERVICES

MODULE CODE: ELSMC501

TEACHER'S GUIDE

Module name: APPLY BASIC MICROCONTROLLER

Table of Contents

Acronyms

AVR: advanced virtual RISC

EEPROM: Electrically Erasable Programmable Read Only Memory

CISC: Complex Instruction Set Computer

LCD: Liquid Crystal Display

LED: Light Emitting Diode

LF: Library Function

I/O: Input/output

IC: Integrated Circuit

GND: Ground

PWM: Pulse Width Modulation

MC: Microcontroller

TX: Transmitter

RX: Receiver

RISC: Reduced Instruction Set Computer

RST: Reset

TTL: Transistor Logic

UDF: User Defined Function

USB: Universal Serial Bus

Introduction

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 a broader sense, the components which constitute a microcontroller are the memory, peripherals and most crucially a processor. Microcontrollers are present in devices where the user has to use a degree of control. They are designed and implemented to execute a specific function such as displaying integers or characters on an LCD display module of a home appliance. Application of microcontrollers is many. In simpler terms, any equipment which has to deal with the functions such as measuring, controlling, displaying and calculating the values consist of a microcontroller chip inside it. They are present in almost all the present day home appliances, toys, traffic lights, office instruments and various day-to-day appliances.

Module Code and Title: ELSMC501-APPLYING BASIC MICROCONTROLLER

Learning Units:

1. Select tools, materials and equipment
2. Apply basic C programming
3. Make microcontroller based circuit (Arduino)

Learning Unit 1: Select tools, materials and equipment

Picture/s reflecting the Learning unit 1

STRUCTURE OF LEARNING UNIT

Learning outcomes:

1.1: Identify tools, materials and equipment
1.2: Describe the microcontroller
1.3: Describe the sensors

Learning outcome 1.1: Identify tools, materials and equipment

Duration:3hours

Learning outcome 1 objectives:

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

The identification of materials, tools and equipment

Resources

Equipment	Tools	Materials
➢ Microcontrollers ➢ PC ➢ Projector ➢ Multimeter ➢ Soldering station ➢ PPE	➢ Whiteboard ➢ Pliers ➢ Screwdrivers ➢ Whiteboard	➢ Wires and cables ➢ Soldering tin ➢ Breadboard/PCB ➢ Electronic components ➢ Markers

Advance preparation:

- Availability of Materials, tools and equipment
- Proper preparation of working place
- Availability of electricity

- Identification of materials, tools and equipment

✓ Multimeter

A Multimeter or a multitester is an electronic measuring instrument that combines several measurement functions in on unit like voltage, current, resistance, capacitance, continuity, etc.

It can also be used to measure the configuration of diodes, capacitance, inductance, transistors, etc.
A Multimeter consists of two types: Analog and digital Multimeter

Analog Multimeter uses a micro ammeter with a moving pointer to display readings.

Digital Multimeter has a numeric display, and may also show a graphical bar representing the measured value.

- ✓ Microcontrollers

The microcontroller is a small computer on a single metal-oxide semi-conductor (MOS) integrated circuit(IC) chip. It is economical programmable logic control that can be interfaced with external devices in order to control the devices from a distance.

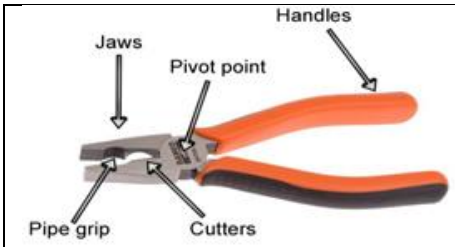
- ✓ Wires and cables

A *cable* is a thick *wire*, or a group of *wires* inside a rubber or plastic covering, which is used to carry electricity or electronic signals.

A wire is a single conductor (material most commonly being copper or aluminium) while cable is two or more insulated wires wrapped in one jacket. Multiple conductors that have no insulation around would be classified as a single conductor.

- ✓ Pliers

Pliers (plyers) are handheld, manually-powered hand tools, often employing serrated jaws, designed for holding, cutting, bending, or manipulation of tough or difficult materials such as wire, sheet metal, or fine machine components.



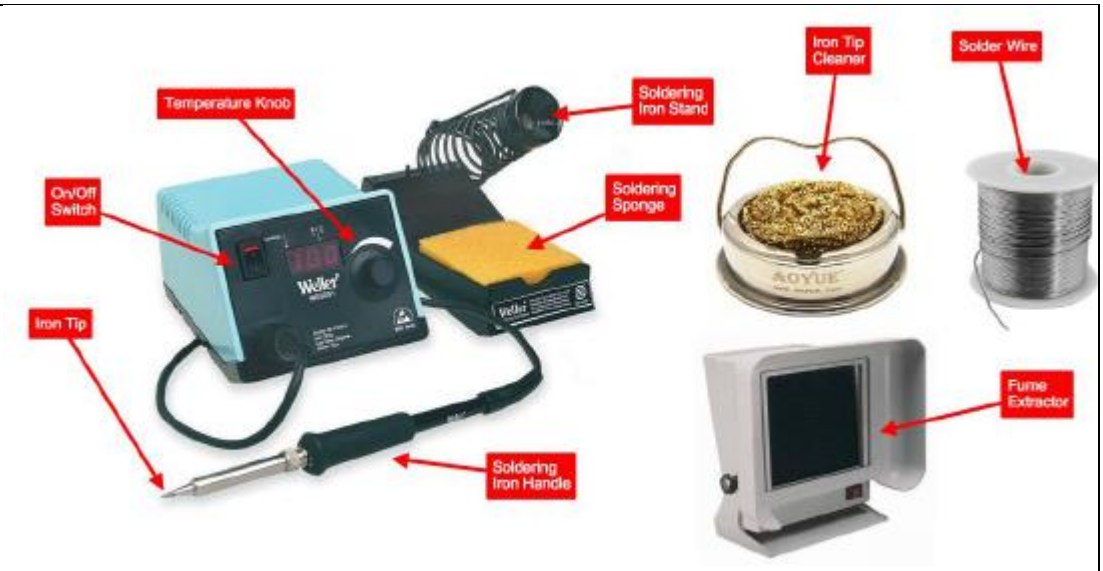
- Screwdrivers

A screwdriver is a tool, manual or powered, for screwing and unscrewing (inserting and removing) screws.

Screwdriver, tool, usually hand-operated, for turning screws with slotted heads. For screws with one straight diametric slot cut across the head, standard screwdrivers with flat blade tips and in a variety of sizes are used.

✓ Soldering station

A soldering station is *a* multipurpose power soldering device designed for electronic components soldering.

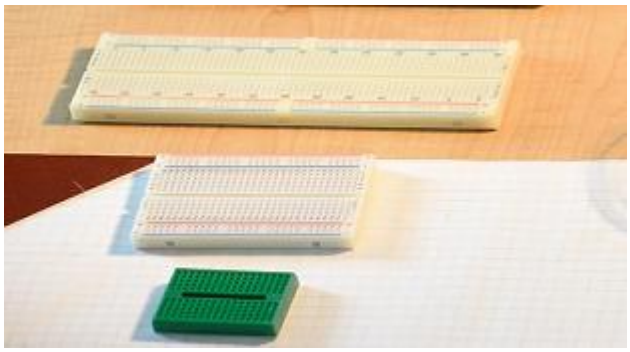


- ✓ Soldering tin

Soldering tin is a metal or metallic alloy used when melted to join metallic surfaces especially: an alloy or compound of lead and **tin** so used.

- ✓ Breadboard/PCB

Breadboard is a board for making an experimental model of an electric circuit.
It allows you to place components and connections on the board to make circuits without soldering.



- ✓ Electronic components

  An electronic components is any physical entity in an electronic system used to affect the electrons or their associated fields in a manner consistent with the intended function of the electronic system. It is often categorized as active (e.g diodes, transistors, thyristors, integrated circuit, etc) or passive (e.g resistors, inductors, capacitors ,etc).

- ✓ PPE

PPE is the personal protective equipment that will protect the user against health or safety risks. They can include items such as safety helmets, gloves, eye protection, hazmat suits, high-visibility clothing, safety footwear, safety tie together, ear plugs, ear defenders and respiratory protective equipment (RPE). In appropriate situations disposable PPE may be provided; e.g. single-use coveralls. Employers have duties concerning the provision and use of personal protective equipment at work.

Summary for the trainer related to the indicative content (key notes using bullets such as ticks etc)

A Multimeter or a multitester is an electronic measuring instrument that combines several measurement functions in on unit like voltage, current, resistance, capacitance, continuity.

A Multimeter consists of two types: Analog and digital Multimeter.


Theoretical learning Activity

1. Choose the right answer.

The following items are the main parts of digital Multimeter

a) Settings, display and ports
b) Selector, display and settings

Answer

a) Settings, display and ports

2. Enumerate three (3) electronic components should use while programming microcontroller

Answer

The three (3) electronic components are: Diodes, resistors and transistors.


Practical learning Activity

Trainees in pair Identify material, tools and equipment.

Points to Remember (Take home message)

- Passive components
- Active components
- Identification of tools, materials and equipment

Learning outcome 1 formative assessment

Written assessment

1. Reply by true or false

The following items are equipment used to program microcontroller:

a) Jumper wires and resistors

b) PC and microcontrollers

Answer

a) False
b) True

2. Why is it necessary to use PPE while you are in workshop?

Answer

PPE is used to protect against hazards that can occur while you are in workshop.

 **Please mix different assessment tools for triangulation and relevancy of assessment**

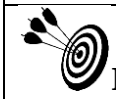
Practical assessment

Identify material, tools and equipment.

Checklist	**Score**	
	YES	**NO**
Indicator: Materials, tools and equipment are well identified		
Microcontrollers		
Multimeter		
Wires and cables		
Pliers		
Screwdrivers		
Soldering station		
Soldering tin		
Breadboard/PCB		
Electronic components		
PPE		
Observation		

Learning outcome 1.2: Describe the microcontrollers

 **Duration:4hours**

 Learning outcome 1 objectives:

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

1. Introduction to microcontroller
2. Classification of microcontrollers
3. Microcontroller parts
4. Advantages of microcontrollers
5. Applications of microcontrollers

Resources

Equipment	Tools	Materials
➢ Microcontrollers ➢ PC ➢ Projector	➢ Whiteboard	➢ Markers ➢ Connecting cables ➢ Jumper wires

Advance preparation:

- Availability of Materials, tools and equipment
- Proper preparation of working place
- Availability of electricity

Indicative content 1.2.1: Introduction to microcontroller

- Introduction to microcontroller

The microcontroller is defined as a small computer on a single metal-oxide semi-conductor (MOS) integrated circuit(IC) chip. It is economical programmable logic control that can be interfaced with external devices in order to control the devices from a distance.
In 1971, the first microcontroller was invented by two engineers Gary Boone and Michael Cochran at Texas, which was a 4-bit microcontroller with built-in ROM and RAM.To understand why microcontrollers form an important part of our modern electronics world, just look at any electronic device accessible to you. Microcontrollers are at the heart of any embedded system today. Therefore, the microcontroller has played a fundamental role in the technological revolution that has formed modern life. Microcontrollers are small, flexible, inexpensive devices that can be successfully implemented and programmed not only by experienced electrical engineers but also by students and professionals' people.

A Microcontroller is a VLSI (Very Large Scale Integration) Integrated Circuit (IC) that contains electronic computing unit and logic unit (combinedly known as CPU), Memory (Program Memory and Data Memory), I/O Ports (Input / Output Ports) and few other components integrated on a single chip and used in automatic control systems including security systems, office machines, power tools, alarming system.It is economical programmable logic control that can be interfaced with external devices in order to control the devices from a distance.

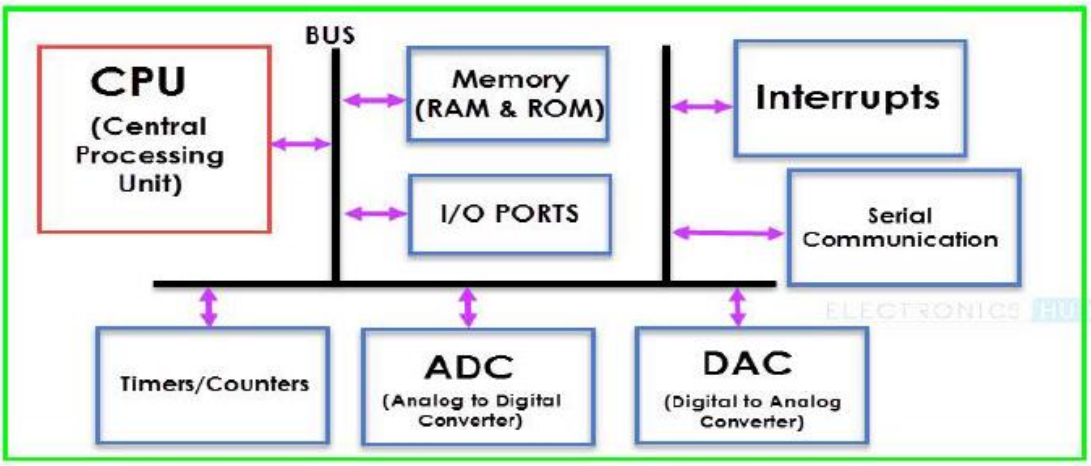
- It was specially built for embedded system and consisted of read write memory, read only memory, I/O ports, processor and built in clock.
- C and assembly languages are used to program the microcontrollers.
- There are also other languages available to program the microcontroller but at the start learning a microcontroller programming with C and

assembly language is a great choice, both are easy to learn and provide a clear concept about microcontroller.

- Technology have been evolved in an amazing way and made our lives easier more than ever before.
- Few years ago, making the elevator in running condition was a hell of task which involved complex programming and circuitry.
- Now, you are capable of not only controlling elevator from microcontroller but you can also move the submarine with the proper instructions directed into a single microcontroller.
- Any application which involves measuring, controlling, and displaying contains a microcontroller chip inside it.
- Microcontrollers come with a wide range of applications but it depends on you to decide what task you want to achieve with the help of microcontroller because it will only take instructions in the form of programming language.
- You can build, upload and execute any program depending on your priorities.

Basic Structure of a Microcontroller

The following image shows the Basic Structure of a Microcontroller.



Microcontroller characteristics

- In modern technologies, some microcontrollers devices constitute a complex design and are capable of having word length more than 64 bit.
- Microcontroller consists of built in components including EPROM, EEPROM, RAM, ROM, timers, I/O ports and reset button. RAM is used for data storage while ROM is used for program and other parameters storage.
- Modern microcontroller are designed using CISC (complex instruction set computer) architecture which involves Marco-type instructions.
- Single macro type instruction is used to replace the number of small instructions.
- Modern microcontrollers operate at much lower power consumption as compared to older ones.
- They can operate at a lower voltage ranging from 1.8 V to 5.5 V.
- Flash memory like EPROM and EEPROM are very liable and advanced features in latest microcontrollers which set them apart from older microcontrollers.

- EPROM is faster and quick than EEPROM memory. It allows to erase and write cycles as many times as you want which makes it user friendly

Comparison between Microprocessor and Microcontroller

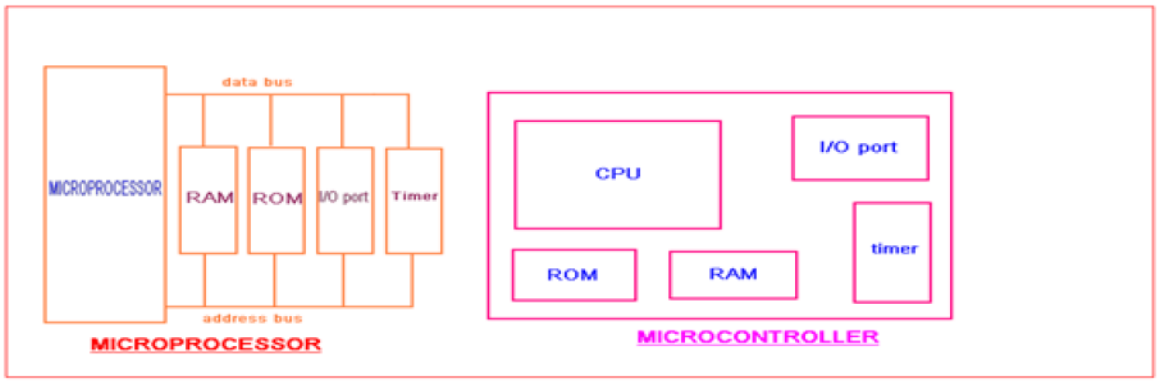
Microprocessor

- It has only the CPU inside; ie the processing powers such as Intel's Pentium 1,2,3,4 core 2 duos, i3, i5 etc.
- Don't have RAM, ROM and other peripheral on the chip. The system designer has to add them externally to make them functional.
- Application includes desktop PCs, laptops, notepads etc.
- Applications are where tasks are unspecific like developing software, games, websites, photo editing, creating documents etc.
- Since microprocessors cannot be used stand alone as it needs RAM, ROM and other peripherals the system that uses microprocessors is costlier than a microcontroller.
- The clock speed of the microprocessor is quite high as compared to the microcontroller. This can operate above 1 GHz as they perform complex tasks.

Microcontroller

- In a microcontroller CPU, RAM, ROM, and other peripherals are embedded on a single chip.
- At times it is termed a mini computer or a computer on a single chip.
- Some giants in the manufacturing business of microcontrollers are ATMEL, microchip, TI, Freescale, Philips, Motorola etc.
- to perform specific tasks. ie, the relationship between the input and output is defined.
- Since the applications are very specific, they need small resources like RAM, ROM, I/O ports and hence can be embedded on a single chip.
- The clock speed of a microcontroller varies from a few MHz to 30-50 MHz.

The main comparison between microprocessor and microcontroller shown in figure and table below:



Comparison of Microprocessor and Microcontroller

NO	Microprocessor	Microcontroller
1	It is only a general purpose computer CPU	It is a microcomputer itself
2	Memory, I/O ports, timers, interrupts are not available inside the chip	All are integrated inside the microcontroller chip
3	This must have many additional digital components to perform its operation	Can function as a microcomputer without any additional components.
4	Systems become bulkier and expensive.	Make the system simple, economic and compact
5	Not capable for handling Boolean functions	Handling Boolean functions
6	Higher accessing time required	Low accessing time
7	Very few pins are programmable	Most of the pins are programmable

8	Very few number of bit handling instructions	Many bit handling instructions
E.g.	INTEL 8086,INTEL Pentium series	INTEL8051,89960,PIC16F877

Indicative content 1.2.2: Classification of microcontrollers

Summary for the trainer related to the indicative content (key notes using bullets such as ticks etc)

- Classification of microcontrollers

The microcontrollers are classified in the following different types:

✓ Based on manufacturer

Microcontrollers are the most used devices in every intelligent equipment in the world. The main work of the Microcontrollers is to send and receive data by using the I/O ports and the specially designed ports for specific operations.
Now days, there are 12 top microcontroller companies over the whole world.
12. Microchip – united states

11. Intel corporation – united states

10. Zilog – united states

9. Toshiba eelctronics – japan

8. Holtek – taiwan

7. Analog devices – united states

6. NXP – netherlands

5. Renesas electronics – japan

4. Texas instruments – united states

3. Silicon labs – united states

2. Rohm semiconductor – japan

1. Infineon technologies – germany

- ✓ Based on family

The microcontroller family is determined by the brand and the architecture. For example, the STM8 family is a range of 8-bit micros made by STmicro, while the STM32 family is a range of 32-bit micros made by the same manufacturer. Therefore, there are three types of microcontroller:

a. 8051 microcontrollers

b. PIC microcontroller

c. AVR microcontroller

a. 8051 Microcontroller

- Most commonly used microcontrollers are belonged to 8051 family.
- 8051 microcontrollers are considered as an ideal choice for most of the professionals.
- Invented by Intel, 8051 microcontroller consists of two members including 8052 and 8031.
- 8052 consists of 3 times and 256 bytes RAM. It encompasses same features as that of 8051 microcontrollers.
- You can also consider 8051 as a subset of 8052 microcontroller.
- Similarly, 8031 exhibits same features as that of 8051 except ROM.
- However, external ROM of 64k bytes can be incorporated in this chip for execution of instructions.

8051 Microcontroller Architecture

- 8051 microcontroller is a 40 pin 8 bit microcontroller invented by Intel in 1981.
- 8051 comes with 128 bytes of RAM and 4KB of built in ROM.
- Based on priorities, 64 KB external memory can be incorporated with the microcontroller.

- A crystalline oscillator is embedded on this microcontroller which comes with a frequency of 12 MHz.
- Two 16 bit timers are integrated in this microcontroller that can be used as a timer as well as a counter.
- 8051 consists of 5 interrupts including External interrupt 0, external interrupt 1, Timer interrupt 0, timer interrupt 1 and Serial port interrupt. It also consists of four 8 bits programmable ports.

b.PIC Microcontroller

- Micro-chip technology invented Peripheral Interface Controller (PIC) which is very common among most of the professionals and experts.
- Micro-chip Technology is very concerned with the needs and requirements of the customers so they constantly keep upgrading their products in order to provide top notch service.
- Low cost, serial programmable ability, and wide availability make this microcontroller stand out of the party.

PIC Microcontroller Architecture

- PIC microcontroller supports Harvard architecture.
- It consists of ROM, CPU, serial communication, timers and counters, oscillators, interrupts, I/O ports and set of registers that also behave as a RAM.
- Special purpose registers are also incorporated on chip hardware.
- Low power consumption makes this controller an ideal choice for an industrial purpose.
- Every PIC brings into play “stack” that is capable of saving return addresses.
- In the older version of PIC microcontrollers, stack could not be accessed by programming, but later versions can be easily accessed by programming
- Low specification computer is enough to run the software that is capable of programming the PIC microcontroller circuit.
- Serial port or USB port is used to connect he computer with the microcontroller.

c. AVR Microcontroller

- AVR stand for ADVANCED VIRTUAL RISC.
- AVR micro controllers is family of RISC microcontrollers from Atmel.
- AVR was one of the first microcontroller families to use on- chip flash memory for program storage.
- AVR microcontrollers are very popular, used in numerous applications, particularly in project prototyping and also in embedded devices

- It supports Harvard Architecture in which program and data is stored in different spaces of microcontroller and can easily be accessed.
- It is considered as earlier types of controllers in which on-chip flash is used for storing program.
- Types of AVR microcontrollers AVR microcontrollers are obtainable in three categories:
  1.Tiny AVR
  2.Mega AVR
  3. Xmega AVR

TINY AVR: This microcontroller has Less memory, small in size, only for simpler applications. • 0.5–16 KB program memory. • 6–32-pin package. • Small in size.

MEGA AVR: This microcontroller is the most popular having a good amount of memory up to 256KB, higher no. of inbuilt peripherals and fit for modest to difficult applications. 4–256 KB program memory • 28–100-pin package • Extended instruction set.

XMEGA AVR: This microcontroller is used commercially for compound applications, which need large program memory and also high speed. • 16–384 KB program memory • 44-100 pin • Extended performance features such as DMA, Event System.

AVR architecture

- AVR architecture was produced by Vegard Wollan and Alf-Egil Bogen.
- The AT90S8515 was the first controller that was based on AVR architechture.
- However, AT90S1200 was the first AVR microcontroller that was commercially available in 1997.
- This controller has a watchdog timer and many power saving sleep modes that make this controller reliable and user-friendly.

✓ 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. 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.

- ✓ Based on memory/Memory devices

The memory devices are divided into two types, they are

- 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.

- ✓ Instruction set

The instruction set consists of CISC and RISC.

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.

A complex instruction set computer is a computer where single instructions can perform numerous low-level operations like a load from memory, an arithmetic operation, and a memory store or are accomplished by multi-step processes or addressing modes in single instructions, as its name propose "Complex Instruction Set".

RISC systems shorten execution time by reducing the clock cycles per instruction and CISC systems shorten execution time by reducing the number of instructions per program. The RISC gives a better execution than the CISC.

A reduced instruction set computer is a computer which only uses simple commands that can be divided into several instructions which achieve low-level operation within a single CLK cycle, as its name propose "Reduced Instruction Set "

RISC Architecture

The term RISC stands for "Reduced Instruction Set Computer". It is a CPU design plan based on simple orders and acts fast.

This is small or reduced set of instructions. Here, every instruction is expected to attain very small jobs. In this machine, the instruction sets are modest and simple, which help in comprising more complex commands. Each instruction is

about the similar length; these are wound together to get compound tasks done in a single operation. Most commands are completed in one machine cycle. This pipelining is a crucial technique used to speed up RISC machines.

Comparison Between RISC and CISC

RISC stands for 'Reduced Instruction Set Computer Whereas, CISC stands for Complex Instruction Set Computer. The RISC processors have a smaller set of instructions with few addressing nodes. The CISC processors have a larger set of instructions with many addressing nodes.

Indicative content 1.2.3: **Microcontroller parts**

Summary for the trainer related to the indicative content (key notes using bullets such as ticks etc)

- Microcontroller parts

Basically, a Microcontroller consists of the following components:

✓ Central processing unit (processor)

Central Processing Unit or 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.

✓ Memory (RAM, ROM)

Any Computational System requires two types of Memory: Program Memory and Data Memory. Program Memory, as the name suggests, contains the program i.e. the instructions to be executed by the CPU. Data Memory on the other hand, is required to store temporary data while executing the instructions.

Usually, Program Memory is a Read Only Memory or ROM and the Data Memory is a Random Access Memory or RAM. Data Memory is sometimes called as Read Write Memory (R/W M).

- ✓ I/O ports

The interface for the Microcontroller to the external world is provided by the I/O Ports or Input/Output Ports. Inputs device like Switches, Keypads, etc. provide information from the user to the CPU in the form of Binary Data.
The CPU, upon receiving the data from the input devices, executes appropriate instructions and gives response through Output Devices like LEDs, Displays, Printers, etc.

- ✓ Timers & counters

One of the important components of a Microcontroller are the Timers and Counters. They provide the operations of Time Delays and counting external events. Additionally, Timers and Counters can provide Function Generation, Pulse Width Modulation, Clock Control, etc.

- ✓ ADC/DAC converters

ADC (Analog to Digital Converter)

Analog to Digital Converter or ADC is a circuit that converts Analog signals to Digital Signals. The ADC Circuit forms the interface between the external Analog Input devices and the CPU of the Microcontroller. Almost all sensors are analog devices and the analog data from these sensors must be converted in to digital data for the CPU to understand.

DAC (Digital to Analog Converter)

DIgital to Analog Converter or DAC is a circuit, that works in contrast to an ADC i.e. it converts Digital Signals to Analog Signals. DAC forms the bridge between the CPU of the Microcontroller and the external analog devices.

✓ Serial interfacing ports

One of the important requirements of a Microcontroller is to communicate with other device and peripherals (external). Serial Port proves such interface through serial communication. Most common serial communication implemented in Microcontrollers is UART.

✓ Communication interfaces

Communications Interfaces means the interfaces and protocols that enable software installed on other computers (including servers and handheld devices) to interoperate with the Microsoft Platform Software on a Personal Computer.

✓ Pins description

A piece of wood, metal, or plastic used especially for fastening things together or for hanging one thing from another. Something that resembles a pin especially in long slender form. a pin that makes an electrical connection.

Indicative content 1.2.4: Advantages of microcontrollers

Summary for the trainer related to the indicative content (key notes using bullets such as ticks etc)

- Advantages of microcontrollers

✓ Automate process

The Development of Automated Energy Saving System using Microcontroller is the project to save the energy power. This project are used various sensors,

controller and display component. By using this system, the output of the system can be controlled automatically by using the Thermistor and relay.

- ✓ Single on board chip

A *single-board microcontroller* is a *microcontroller* built onto a *single* printed circuit *board.*

This single chip microcomputer will have a CPU, memory and I/O interfaces, timers, ADC/DACs etc. on a single chip itself. This component is called a Single Chip Microcontroller. Nowadays it is also being referred to as an Embedded System or to some extent it could also be called at System-on-Chip (SoC).

- ✓ Efficient

When operating at a low clock frequency, many microcontrollers suffer from high baseline power consumption which results in high total power consumption. A majority of MCUs achieve their optimal power consumption per MHz at a high frequency, performing significantly worse at a lower frequency.

Generally, microcontrollers have many advantages include:

a) No need for any external interfacing of basic components like Memory, I/O Ports, etc.

b) Microcontrollers doesn't require complex operating systems as all the instructions must be written and stored in the memory. (RTOS is an exception).

c) All the Input/Output Ports are programmable.

d) Integration of all the essential components reduces the cost, design time and area of the product (or application).

e) Usage of a microcontroller is simple, easy to troubleshoot and system maintaining.

f) Low time required for performing operations.

Disadvantages of Microcontrollers

The microcontrollers have also many advantages include:

a) Microcontrollers have got more complex architecture than that of microprocessors.

b) Only perform a limited number of executions simultaneously.

c) Mostly used in micro-equipments.

d) Cannot interface high power devices directly.

e) The amount of memory limits the instructions that a microcontroller can execute.

f) No Operating System and hence, all the instruction must be written.

Indicative content 1.2.5: Applications of microcontrollers

Summary for the trainer related to the indicative content (key notes using bullets such as ticks etc)

- Applications of microcontrollers

There are huge number of applications of Microcontrollers. The following are some applications of Microcontrollers:

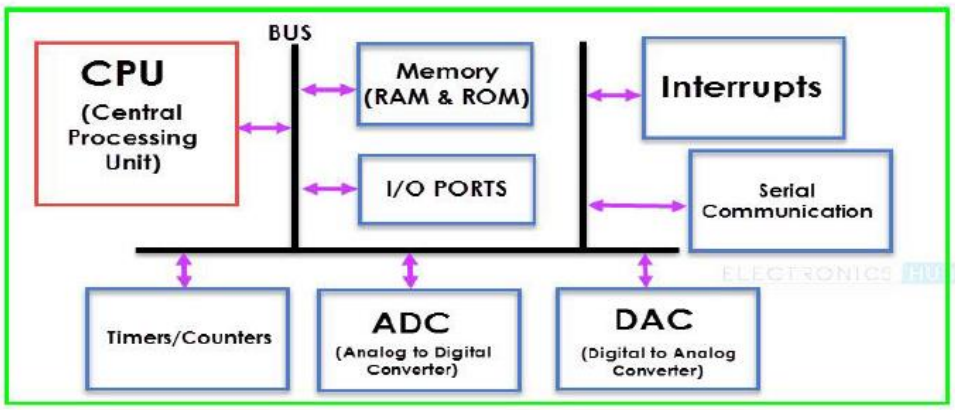
- ✓ Home appliances control
- ✓ Home automation control system
- ✓ Industrial machines control
- ✓ Agriculture activities monitoring
- ✓ Climate change monitoring
- ✓ Front Panel Controls in devices like Oven, washing Machine etc.
- ✓ Smoke and Fire Alarms
- ✓ Peripheral controller of a PC
- ✓ Robotics and Embedded systems
- ✓ Bio-medical equipment
- ✓ Communication and power systems
- ✓ Automobiles and security systems
- ✓ Implanted medical equipment

- ✓ Fire detection devices
- ✓ Temperature and light sensing devices
- ✓ Industrial automation devices

Summary for the trainer related to the indicative content(key notes using bullets such as ticks etc)

The microcontroller is defined as a small computer on a single metal-oxide semi-conductor (MOS) integrated circuit(IC) chip. It is economical programmable logic control that can be interfaced with external devices in order to control the devices from a distance.

Basic structure of microcontroller



Theoretical learning Activity

- ✓ Ask trainees to brainstorm about types of microcontroller within groups
- ✓ Ask trainees to brainstorm about comparison between microprocessor and microcontroller within groups

Practical learning Activity

- ✓ Trainees in pair identify the microcontroller parts and classify microcontroller types.

 Points to Remember (Take home message)

- Description of the three types of microcontroller
- Classification of microcontroller
- Advantages and disadvantages of microcontroller
- Microcontroller parts
- Basic structure of microcontroller

 Learning outcome 2 formative assessment

Written assessment

1. Choose the right answer.

The following items are applications of microcontroller, except:

a) Home appliances control
b) Industrial machines control
c) Agriculture activities monitoring
d) Climate change monitoring
e) Smoke and Fire Alarms
f) Robotics and Embedded systems
g) Communication and power systems
h) Automobiles and security systems
i) Fire detection devices
j) Success and failing
k) Temperature and light sensing devices

Answer

j) Success and failing

2. Enumerate six (6) main parts of microcontroller

Answer

1) Memory (RAM&ROM)
2) I/O ports
3) CPU
4) Timer/counters
5) Serial communication
6) ADC/DAC

 **Please mix different assessment tools for triangulation and relevancy of assessment**

Practical assessment

- ❖ Identify the microcontroller parts
- ❖ Classify microcontroller

Checklist	**Score**	
	YES	**NO**
Indicator: Microcontroller Hardware parts are well identified		
central processing unit (processor)		
Memory (RAM, ROM)		
I/O ports		
Timers & counters		
ADC/DAC converters		
Serial interfacing ports		
Communication interfaces		
Indicator :Microcontroller are well classified		
Based on manufacturer		
Instruction set		
Based on bits		
Based on memory		
Based on family		
Observation		

Learning outcome 1.3: Describe sensors

Duration: 3hours

Learning outcome 3 objectives:

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

1. Introduction to sensors
2. Types of sensors
3. Applications of sensors

Resources

Equipment	Tools	Materials
➢ PC ➢ Projector ➢ Microcontrollers ➢ Sensors	➢ Whiteboard ➢ PCB	➢ Markers ➢ Wires

Advance preparation:

- Availability of Materials, tools and equipment
- Proper preparation of working place
- Availability of electricity

- **Introduction to sensors**

A sensor is defined as a device or a module that helps to detect any changes in physical quantity like pressure, force or electrical quantity like current or any other form of energy**.** After observing the changes, sensor sends the detected input to a microcontroller or microprocessor. Hence, a sensor is a device that responds to a physical stimulus.

A sensor also called: transducer, detector, etc.

Some useful definitions:

Transducer

- A device that converts energy of one form into energy of another form.

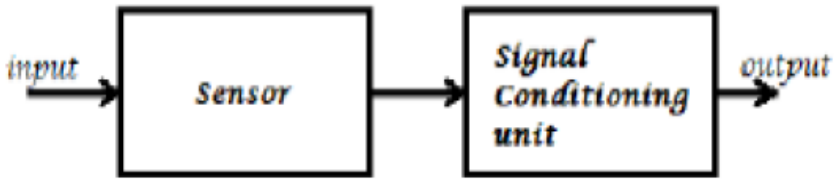
Actuator

- A device or mechanism capable of performing a physical action.

Stimulus

- The quantity that is sensed.
- Finally, a sensor produces a readable output signal, which can be either optical, electrical, or any form of signal that corresponds to change in input signal.

The sensor can be defined as a device which can be used to sense/detect the physical quantity like force, pressure, strain, light etc and then convert it into desired output like the electrical signal to measure the applied physical quantity. In few cases, a sensor alone may not be sufficient to analyze the obtained signal. In those cases, a signal conditioning unit is used in order to maintain sensor's output voltage levels in the desired range with respect to the end device that we use.



In signal conditioning unit, the output of the sensor may be amplified, and filtered or modified to the desired output voltage. For example, if we consider a microphone it detects the audio signal and converts to the output voltage (is in terms of millivolts) which becomes hard to drive an output circuit. So, a signal conditioning unit (an amplifier) is used to increase the signal strength. But the signal conditioning may not be necessary for all the sensors like photodiode, LDR etc.

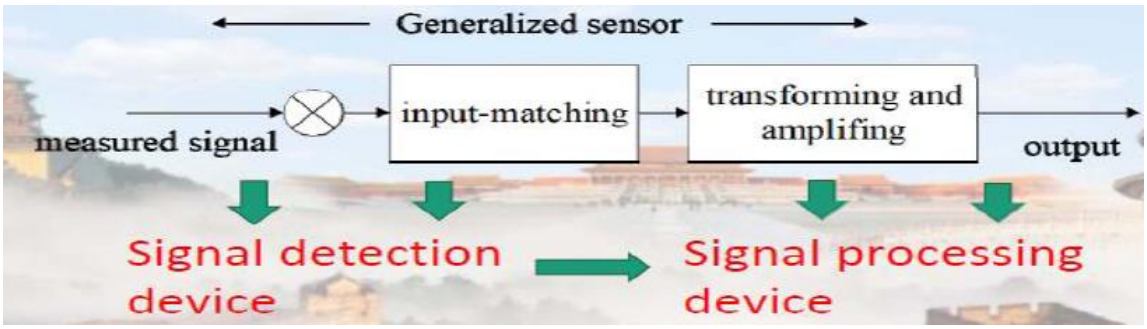


Figure: Block diagram of generalized sensor

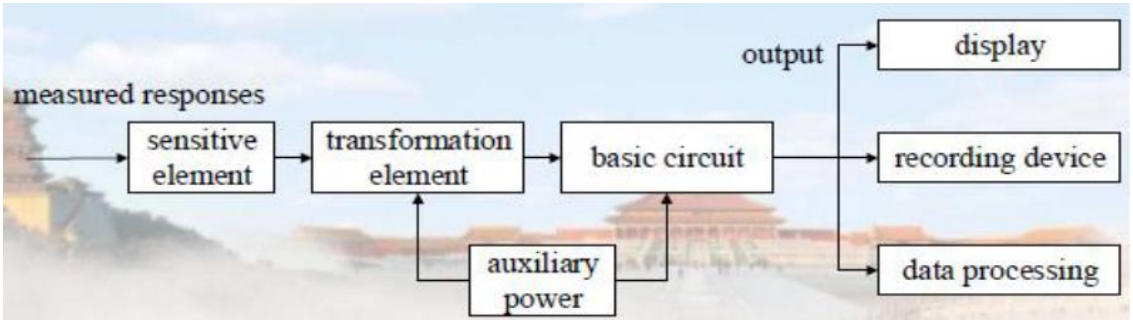


Figure: Composition of sensor

Indicative content 1.3.2: **Types of sensors**

Summary for the trainer related to the indicative content (key notes using bullets such as ticks etc)

- Types of sensors

Sensors are classified in three (3) classifications:

1. Active and Passive Sensors
2. Analog and Digital Sensor
3. Contact and non-contact sensors

Active and Passive Sensors

Active Sensors: are the type of sensors that produces output signal with help of external excitation supply. The own physical properties of the sensor vary with respect to the applied external effect. Therefore, it is also called as parametric sensors

Examples: the carbon microphone, strain gauge (device which measures pressure or stress)

Passive Sensors: are the type of sensors that produces output signal without the help of external excitation supply. They do not need any extra stimulus or voltage**.** Other name Self Generating Sensors.

Example: Magnetic microphones, piezoelectric sensors, thermocouple (which generates a voltage value corresponding to the heat, applied. It does not require any external power supply).

Analog and Digital Sensors

Analog Sensors: An analog sensor measures continuously the variable and detects any proportional value between 100% and 0%. For this

reason, the measure provided by the analog sensor is more precise than the one provided by the digital sensor.

The analog output generated is proportional to the measured or the input given to the system. Generally, analog voltage in the range of 0 to 5 V or current is produced as the output Examples for continuous signals: light sensors, temperature sensors (that measures between 0°C and 100°C.)

Digital sensors

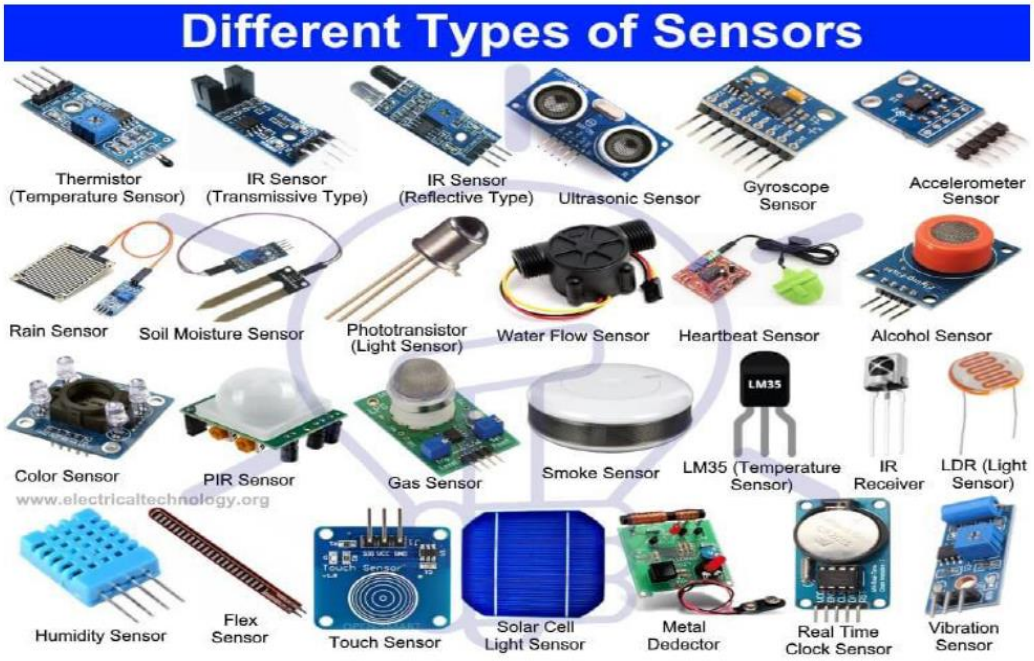
A digital sensor only detects two possible status: if it is working at 100% or at 0%.

A digital sensor can only measure if it is working over 50°C (displaying 1) or under 50°C (displaying 0).

Contact and noncontact sensors

Contact sensor: a sensor that requires physical contact with the stimulus**.** Examples: strain gauges, most temperature sensors

Non-contact sensor: requires no physical contact**.** Examples: most optical and magnetic sensors, infrared, thermometers, etc.



Types of sensors:

- Temperature & Thermocouple Sensors
- Light sensor
- Proximity Sensor
- Pressure Sensor
- Metal detector, Water Flow & Heartbeat Sensor
- Radiation sensor
- Motion sensor
- Smoke, Fog, Gas, Ethanol & Alcohol Sensor
- Humidity, Soil Moisture & Rain Sensor
- Etc.

Light sensor

LDR As the name itself specifies that the resistor that depends upon the light intensity. It works on the principle of photoconductivity which means the conduction due to the light. When light falls on the LDR, *its resistance decreases and acts similar to a conductor and when no light falls on it, its resistance is almost in the range of MΩ or ideally it acts as an open circuit.* One note should be considered with LDR is that it won't respond if the light is not exactly focused on its surface

Temperature sensor

Temperature sensors/detectors/transducers are electronic devices that detects thermal parameters provide signals to the inputs of control and display devices. Temperature sensor typically relies on an RTD or thermistor to measure temperature and convert it to an output voltage.

Radiation sensor

Radiation sensors/detectors are electronic devices that sense the presence of alpha, beta, or gamma particles and provide signals to counters and display devices. Key specifications include sensor type and minimum detectable energies. Radiation detectors are used for surveys and sample counting.

Proximity sensor

Proximity sensors are electronic devices used to detect the presence of nearby objects through non-contacting means. A proximity sensor can detect the presence of objects usually within a range of up to several millimetres and doing so, produce a usually DC output signal a controller.

Pressure sensor

Pressure sensors/detectors/transducers are electro-mechanical devices which detect forces per unit in gases or liquids are provide the signals to the inputs of control and display devices.

Motion sensor

Motion sensors/detectors/transducers are electronic devices that can sensor the movement or stoppage of parts, people, etc.

Metal sensor

Metal detectors are electronic or electro-mechanical devices used to sense the presence of metal in a variety of situations ranging from packages to people. Metal detectors can be permanent or portable and rely on a number of sensor technologies with electromagnetics being popular.

Indicative content 1.3.3: **Applications of sensors**

Summary for the trainer related to the indicative content (key notes using bullets such as ticks etc)

- Applications of sensors
  - ✓ Temperature
  - ✓ detection
  - ✓ Pressure detection
  - ✓ Light detection
  - ✓ Proximity detection
  - ✓ Chemical
  - ✓ concentration
  - ✓ detection
  - ✓ Metal detection
  - ✓ Humidity detection
  - ✓ Smoke detection

Summary for the trainer related to the indicative content(key notes using bullets such as ticks etc)

Sensor is a device that responds to a physical stimulus.

Types of sensors

Sensors are classified in three (3) classifications:

1. Active and Passive Sensors
2. Analog and Digital Sensor
3. Contact and non-contact sensors

Theoretical learning Activity

- ✓ ask trainees to brainstorm about types of sensors within groups
- ✓ ask trainees to brainstorm about application of sensor within groups

Practical learning Activity

- ✓ Trainees in pair perform to Identify and select sensors application

Points to Remember (Take home message)

- ➢ Definition of sensor
- ➢ The types of sensors
- ➢ Application of sensor


Learning outcome 3 formative assessment

Written assessment

1. Distinguish active and passive sensors

Answer

Active Sensors: are the type of sensors that produces output signal with help of external excitation supply.

Passive Sensors: Are the type of sensors that produces output signal without the help of external excitation supply.

2. Complete the following statements using the types of sensors

Answer

a).....a sensor that requires physical contact with the stimulus**.** Examples: strain gauges, most temperature sensors

b).....requires no physical contact**.** Examples: most optical and magnetic sensors, infrared, thermometers, etc.

Answer

Contact sensor: A sensor that requires physical contact with the stimulus**.** Examples: strain gauges, most temperature sensors

Non-contact sensor: Requires no physical contact**.** Examples: most optical and magnetic sensors, infrared, thermometers, etc.

 **Please mix different assessment tools for triangulation and relevancy of assessment**

Practical assessment

Identify and select sensors application

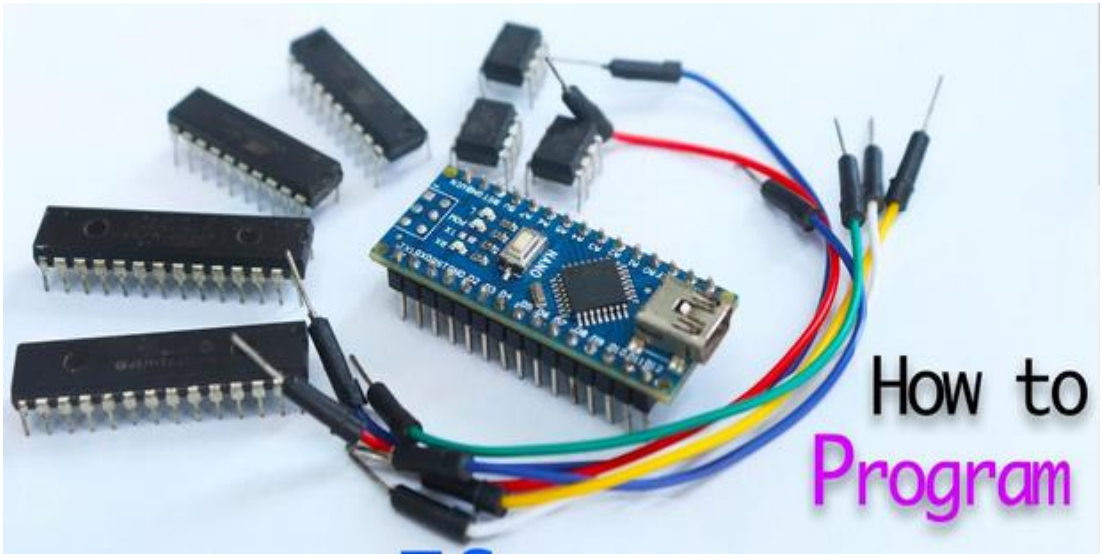
Checklist	**Score**	
	YES	**NO**
Indicator :Applications of sensors are well identified and selected		
Temperature detection		
Smoke detection		
Humidity detection		
Metal detection		
Chemical concentration detection		
Proximity detection		
Light detection		
Pressure detection		
Observation		

References:

- Lipovski, G. J. (1999). Single and Multi-Chip Microcontroller Interfacing: For the Motorola 6812.
- https://www.slideshare.net/redwan1006066/microcontroller-presentation
- https://www.electronicshub.org/microcontrollers/
- https://www.entcengg.com/criteria-choosing-microcontroller/

Learning Unit 2: Apply basic C programming

Picture/s reflecting the Learning unit 2



STRUCTURE OF LEARNING UNIT

Learning outcomes:

2.1: Describe the C language
2.2: Describe the microcontroller programming languages
2.3: Test the microcontroller codes (sketches)

Learning outcome 2.1: Apply the C programming language

Duration:10hrs

Learning out come 2 objectives:

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

Apply C programming Language

Resources

Equipment	Tools	Materials
➢ PC ➢ Projector	➢ C compiler ➢ Whiteboard	➢ Markers ➢ Chalk

Advance preparation:

- Availability of Materials, tools and equipment
- Proper preparation of working place
- Availability of electricity

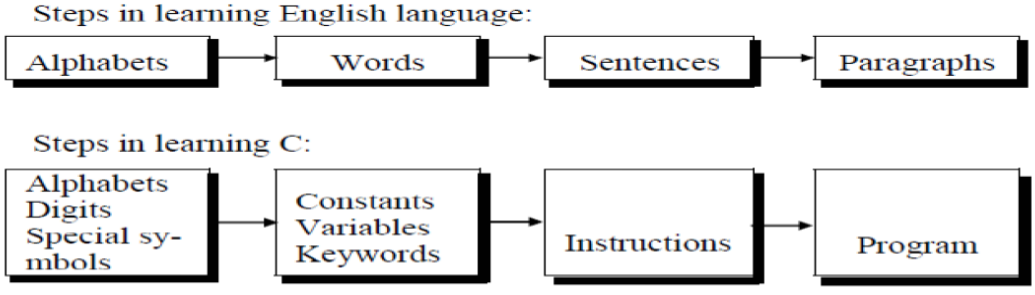
✓ Apply C programming Language

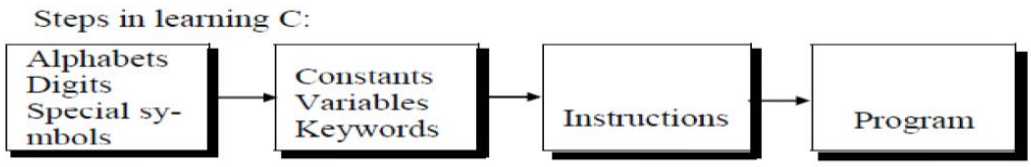
Introduction to C programming

C is a programming language developed at AT & T's Bell Laboratories of USA in 1972. It was designed and written by a man named Dennis Ritchie. It was mainly developed as a system programming language to write an operating system.

C is a general-purpose programming language that is extremely popular, simple and flexible. It is machine-independent, structured programming language which is used extensively in various applications.
There is a close analogy between learning English language and learning C language. The classical method of learning English is to first learn the alphabets used in the language, then learn to combine these alphabets to form words, which in turn are combined to form sentences and sentences are combined to form paragraphs.

Learning C is similar and easier. Instead of straight-away learning how to write programs, we must first know what alphabets, numbers and special symbols are used in C, then how using them constants, variables and keywords are constructed, and finally how are these combined to form an **instruction**. A group of instructions would be combined later on to form a **program**.





A computer **program** is just a collection of the instructions necessary to solve a specific problem. The basic operations of a computer system form what is known as the computer's **instruction set.** And the approach or method that is used to solve the problem is known as an ***algorithm***.

Features of C Programming Language

1. Programs written in C are efficient and fast.

2.C is highly portable, programs once written in C can be run on other machines with minor or no modification.

3. C is a collection of C library functions; we can also create our function and add it to the C library.

4. C is easily extensible.

Advantages of C

1. C is a general purpose programming language. You can generate games, business software, utilities, mathematical models, word processors, spreadsheets and other kinds of software.

2. C is a structured programming language. It uses structured statements such as while, for loops in place of goto statements which cause bugs (error) in the program.

3. System independence- C does not require any services from the operating system, it runs independently. C can run on any operating system.

4. High efficiency- C compilers are generally able to translate source code into efficient machine instructions. C language data and control

mechanisms are well matched to most small computers and microcomputers.

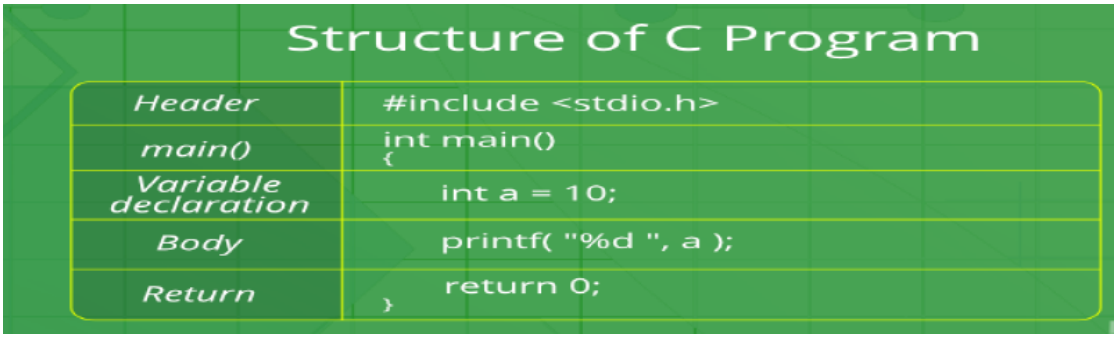
5. System programming- C is used for system programming i.e. writing operating systems. The UNIX operating system is also rewritten from C.

Disadvantages of C

1. C does not provide Object Oriented Programming (OOP) concepts.
2. There are no concepts of Namespace in C.
3. C does not provide binding or wrapping up of data in a single unit.
4. C does not provide Constructor and Destructor.

- ✓ C program structure

1) Comment line
2) header
3) main function ()
{
4) Global variable declaration
5) body
6) return
}



1) Comment line

It indicates the purpose of the program. It is represented as

/*.....*/

2) Header /Preprocessor Directive: • stdint.h – Defines exact width integer types.

• stdio.h – Defines core input and output functions

• math.h – Defines common mathematical functions

3) main()

Comment line is used for increasing the readability of the program. It is useful in explaining the program and generally used for documentation. It is enclosed within the decimeters. Comment line can be single or multiple line but should not be nested. It can be anywhere in the program except inside string constant & character constant.

A header file is a file with extension .h which contains C function declarations and macro definitions to be shared between several source files.

Some of C Header files:

Syntax to include a header file in C:

#include

#include<stdio.h> tells the compiler to include information about the standard input/output library. It is also used in symbolic constant such as #define PI3.14 (value).

The next part of a C program is to declare the main () function. The syntax to declare the main function is:

Syntax to Declare main method:

int main()

{}

The program execution starts with opening braces and end with closing brace. And in between the two braces declaration part as well as executable part is mentioned. And at the end of each line, the semi-colon is given which indicates statement termination.

It is the user defined function and every function has one main () function from where actually program is started and it is encloses within the pair of curly braces.
The main ( ) function can be anywhere in the program but in general practice it is placed in the first position.

4) Global Declaration:

This is the section where variable is declared globally so that it can be access by all the functions used in the program. It refers to the variables that are to be used in the function. Please note that in the C program, no variable can be used without being declared. Also in a C program, the variables are to be declared before any operation in the function.**Therefore, local variable** is declared inside a function whereas **global variable** is declared outside the function. **Local variables** are created when the function has started execution and is lost when the function terminates, on the other hand, **Global variable** is created as execution starts and is lost when the **program** ends.

Example:

```
int main()
{
int a;
```

5) Body

Body of a function in C program, refers to the operations that are performed in the functions. It can be anything like manipulations, searching, sorting, printing, etc.

Example:

```
int main()
```

{

int a;

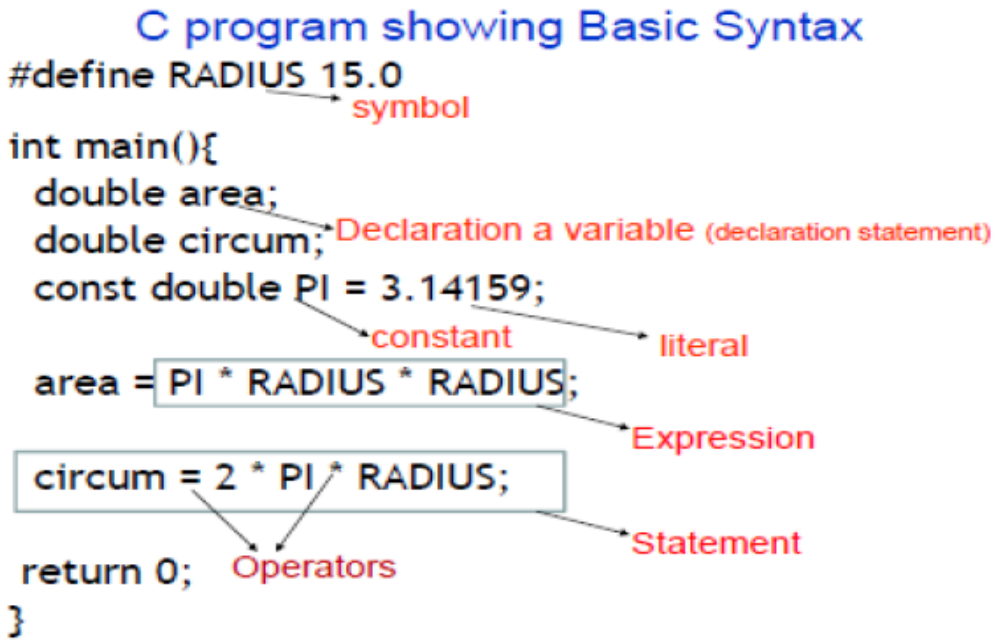
printf("%d", a);

6) Return Statement

The last part in any C program is the return statement. The return statement refers to the returning of the values from a function. This return statement and return value depend upon the return type of the function. For example, if the return type is void, then there will be no return statement. In any other case, there will be a return statement and the return value will be of the type of the specified return type.

Example:

```
int main()
{
int a;
printf("%d", a);
return 0;
}
```



```
/*First c program with return statement*/
#include <stdio.h>
int main (void)
{
printf ("welcome to c Programming language.\n");
return 0;
}
```

Output: welcome to c programming language.

Exanple 1

```
#include <stdio.h>
Int main()
{
Printf( “welcome to c!\n”);
return 0;
}
```

Output :

Welcome to C!

Example 2

```
#include <stdio.h>
Int main()
{
Printf( “welcome\nto\nc!\n”);
return 0;
}
```

Output

Welcome

To

C!

- ✓ Data types

Data types refer to an extensive system used for declaring variables or functions of different types before its use. The type of a variable

determines how much space it occupies in storage and how the bit pattern stored is interpreted. The value of a variable can be changed any time.

There are 4 classes of data types:

1. Primary data types (int, float, char)
2. User defined data types (enumerator, typedef)
3. Derived data types (array, function, pointers, structure, union)
4. Empty data sets or void data type. (void)

1. Primary data types

Integers

An integer is a numeric literal (associated with numbers) without any fractional or exponential part. There are three types of integer literals in C programming:

Decimal (base 10)

Octal (base 8)

Hexadecimal (base 16)

For example:

Decimal: 0, -9, 22 etc

Octal: 021, 077, 033 etc

Hexadecimal: 0x7f, 0x2a, 0x521 etc

Floating-point Literals

A floating-point literal is a numeric literal that has either a fractional form or an exponent form. For example:

-2.0

0.0000234

-0.22E-5

Characters

A character literal is created by enclosing a single character inside single quotation marks. For example: 'a', 'm', 'F', '2', '}' etc.

- ✓ Variables and constants)

Constants

Constant is any value that cannot be changed during program execution. If you want to define a variable whose value cannot be changed, you can use the const keyword. This will create a constant. For example,

const double PI = 3.14;

Here, PI is a symbolic constant; its value cannot be changed.

const PI = 3.14;

PI = 2.9; //Error

Variable

In programming, a variable is a container (storage area) to hold data.

To indicate the storage area, each variable should be given a unique name (identifer).

Variable names are just the symbolic representation of a memory location. For example:

int playerScore = 95;

Here, playerScore is a variable of int type. Here, the variable is assigned an integer value 95 .

The value of a variable can be changed, hence the name variable.

Rules for naming a variable

1. A variable name can have only letters (both uppercase and lowercase letters), digits and underscore.
2. The first letter of a variable should be either a letter or an underscore.

3. There is no rule on how long a variable name (identifer) can be. However, you may run into problems in some compilers if the variable name is longer than 31characters.

Note: You should always try to give meaningful names to variables. For example:

firstName is a better variable name than fn .C is a strongly typed language. This means that the variable type cannot be changed once it is declared. For example:

```
int number = 5; // integer variable
number = 5.5; // error
```

Here, the type of number variable is int . You cannot assign a floating-point (decimal) value 5.5 to this variable.

- ✓ Operators

Operators can be used to perform the required computations on the values.

Classes of operators:

Arithmetic operator.
1. Unary operators.(+)
2. Relational operators.
3. Assignment operators.
4. Equality operators.
5. Logical operators.
6. Conditional operators.

Arithmetic Operator

An arithmetic operator performs mathematical operations such as addition, subtraction, multiplication, division etc on numerical values (constants and variables).

Operator	Meaning of Operator
+	addition or unary plus
-	subtraction or unary minus
*	Multiplication
/	division
%	remainder after division (modulo division)

Assignment Operator

A value can be stored in a variable with the use of assignment operator. The assignment operator (=) is used in assignment statement and assignment expression.
Operand on the left-hand side should be variable and the operand on the right hand side should be variable or constant or any expression. When variable on the left hand side is occur on the right hand side then we can avoid by writing the compound statement. For example,
int x= y;
int Sum=x+y+z;

- ✓ Functions

A function is a self-contained block of codes or sub programs with a set of statements that perform some specific task or coherent task when it is called.
It is something like to hiring a person to do some specific task like, every six months servicing a bike and hand over to it. Any 'C' program contain at least one function i.e main ().

- ✓ Libraries

A **library** in **C** is a collection of header files, exposed for use by other programs. The **library** therefore consists of an interface expressed in a .h file (named the "header") and an implementation expressed in a .**c** file

Common Library Functions: stdio.h functions

There are basically two types of function:

1. Library function

2. User defined function

The standard library functions are built-in functions in C programming to handle tasks such as mathematical computations, I/O processing, string handling etc.

These functions are defined in the header file. When you include the header file, these functions are available for use. For example: The printf() is a standard library function to send formatted output to the screen (display output on the screen). This function is defined in "stdio.h" header file.

User-defined function

As mentioned earlier, C allow programmers to define functions. Such functions created by the user are called user-defined functions. You can create as many user-defined functions as you want.

Library function	**User-defined function**
Library functions (LF) are predefined functions	User defined functions (UDF) are the function which created by the user as per his own requirements.
LF are part of header file (MATH.h) which is called run time.	UDF are part of the program which compile run time
In LF it is given by developers	In UDF the name of function id decided by user.
LF name of function cannot changed	In UDF the name of function can be changed any time

- ✓ Conditions

Control Statement

Generally C program statement is executed in a order in which they appear in the program. But sometimes we use decision making condition for execution only a part of program,that is called control statement. Control statement defined how the control is transferred from one part to the other part of the program. There are several control statement like if...else, switch, while, do....while, for loop, break, continue, go to etc.

if statement

Statement execute set of command like when condition is true and its syntax is

If (condition)

Statement;

Or

The syntax of the if statement in C programming is:

```
1. if (test expression)
2. {
3. // statements to be executed if the test expression is true
4. }
```

The statement is executed only when condition is true. If the if statement body is consists of several statement then better to use pair of curly braces. Here in case condition is false then compiler skip the line within the if block.

```
void main()
{
int n;
printf (" enter a number:");
scanf("%d",&n);
If (n>10)
```

Printf(“ number is grater”);

}

Output:

Enter a number:12

Number is greater

if.....else ... Statement

it is bidirectional conditional control statement that contains one condition & two possible action. Condition may be true or false, where non-zero value regarded as true & zero value regarded as false. If condition are satisfy true, then a single or block of statement executed otherwise another single or block of statement is
executed. Its syntax is:-

```
if (condition)
{
Statement1;
Statement2;
}
else
{
Statement1;
Statement2;
}
```

Or

```
1. if (test expression) {
2. // statements to be executed if the test expression is true
3. }
4. else {
5. // statements to be executed if the test expression is false
6. }
```

Else statement cannot be used without if or no multiple else statement are allowed within one if statement. It means there must be a if statement with in an else statement.

Example:

```
/* To check a number is eve or odd */
void main()
{
int n;
printf (“enter a number:”);
scanf (“%d”, &n);
If (n%2==0)
printf (“even number”);
else
printf(“odd number”);
}
```

Output: enter a number:121
odd number

Example 2: if...else statement

```
1. // Check whether an integer is odd or even
2. #include <stdio.h>
3. int main()
4. {
5. int number;
6. printf("Enter an integer: ");
7. scanf("%d", &number);
8. // True if the remainder is 0
9. if (number%2 == 0)
10. {
11. printf("%d is an even integer.",number);
```

```
12. }
13. else
14. {
15. printf("%d is an odd integer.",number);
16. }
17. return 0;
18.}
```

Loops in C

Loop: It is a block of statement that performs set of instructions. In loops Repeating particular portion of the program either a specified number of time or until a particular no of condition is being satisfied.

There are five types of loops in c:

1.While loop

2.do while loop

3.for loop

4. Nested loop

5. Infinite loop

1. While loop

Syntax:

```
while(condition)
{
Statement 1;
Statement 2;
}
Or     while(test condition)
       Statement;
```

The test condition may be any expression .when we want to do something a fixed no of times but not known about the number of iteration, in a program then while loop is used.

Here first condition is checked if, it is true body of the loop is executed else, If condition is false control will be come out of loop.

Example:

```
/* wap to print 5 times welcome to C" */
#include<stdio.h>
void main()
{
int p=1;
While(p<=5)
{
printf("Welcome to C\n");
P=p+1;
}
}
```

2. do while loop

Output: Welcome to C
Welcome to C
Welcome to C
Welcome to C
Welcome to C

So as long as condition remains true statements within the body of while loop will get executed repeatedly.

This (do while loop) statement is also used for looping. The body of this loop may contain single statement or block of statement. The syntax for writing this statement is:

Syntax:-

```
Do
{
Statement;
}
while(condition);
```

Example:-

```
#include<stdio.h>
void main()
{
int X=4;
do
{
Printf("%d",X);
X=X+1;
}whie(X<=10);
Printf(" ");
}
```

Output: 4 5 6 7 8 9 10

3. For loop

There is minor difference between while and do while loop, **while loop** test the condition before executing any of the statement of loop. Whereas **do while loop** test condition after having executed the statement at least one within the loop.

for loop in a program, for loop is generally used when number of iteration are known in

advance. The body of the loop can be single statement or multiple statements. Its

syntax for writing is:

Syntax:

```
for(exp1;exp2;exp3)
{
Statement;
}
```

Or

```
for(initialized counter; test counter; update counter)
{
Statement;
}
```

Here exp1 is an initialization expression, exp2 is test expression or condition and exp3 is an update expression. Expression 1 is executed only once when loop started and used to initialize the loop variables. Condition expression generally uses relational and logical operators. And updation part executed only when after
body of the loop is executed.

Example:-

```
void main()
{
int i;
for(i=1;i<10;i++)
{
Printf(" %d ", i);
}
}
```

Output:-1 2 3 4 5 6 7 8 9

4. Nested loop

It consists one loop inside another loop

5. Infinite loop

It executes continuously without end stop.

 Summary for the trainer related to the indicative content(key notes using bullets such as ticks etc)

- **Features of C Programming Language**

1. Programs written in C are efficient and fast.

2. C is highly portable, programs once written in C can be run on other machines with minor or no modification.

3. C is a collection of C library functions; we can also create our function and add it to the C library.

4. C is easily extensible.

- The types of loops: For, while, do while, nested, and infinite loops.

 Theoretical learning Activity

- ✓ Ask trainees to brainstorm about control structure within groups

 Practical learning Activity

- ✓ Trainees in pair perform to Apply C programming language

 Points to Remember (Take home message)

- C program structure
- Control structure
- Feature of C programming language

 Learning outcome 1 formative assessment

Written assessment

1. What is the correct value to return to the operating system upon the successful completion of a program?
A. -1
B. 1
C. 0
D. Programs do not return a value.

Answer

C. 0

2. What is the only function all C programs must contain?
A. start ()
B. system()
C. main()
D. program()

Answer

C. main()

3. What punctuation is used to signal the beginning and end of code blocks?
A. { }
B. -> and <-
C. BEGIN and END
D. (and)

Answer

A. { }

4. What punctuation ends most lines of C code?
A. .
B. ;
C. :
D. '

Answer

B. ;

5. Which of the following is a correct comment?
A. */ Comments */
B. ** Comment **
C. /* Comment */
D. { Comment }

Answer

C. /* Comment */

6. Which of the following is not a correct variable type?
A. float
B. real
C. int
D. double

Answer

B. real

7. What is the final value of x when the code int x; for(x=0; x<=9; x++) {} is run?
A. 10
B. 9
C. 0
D. 1

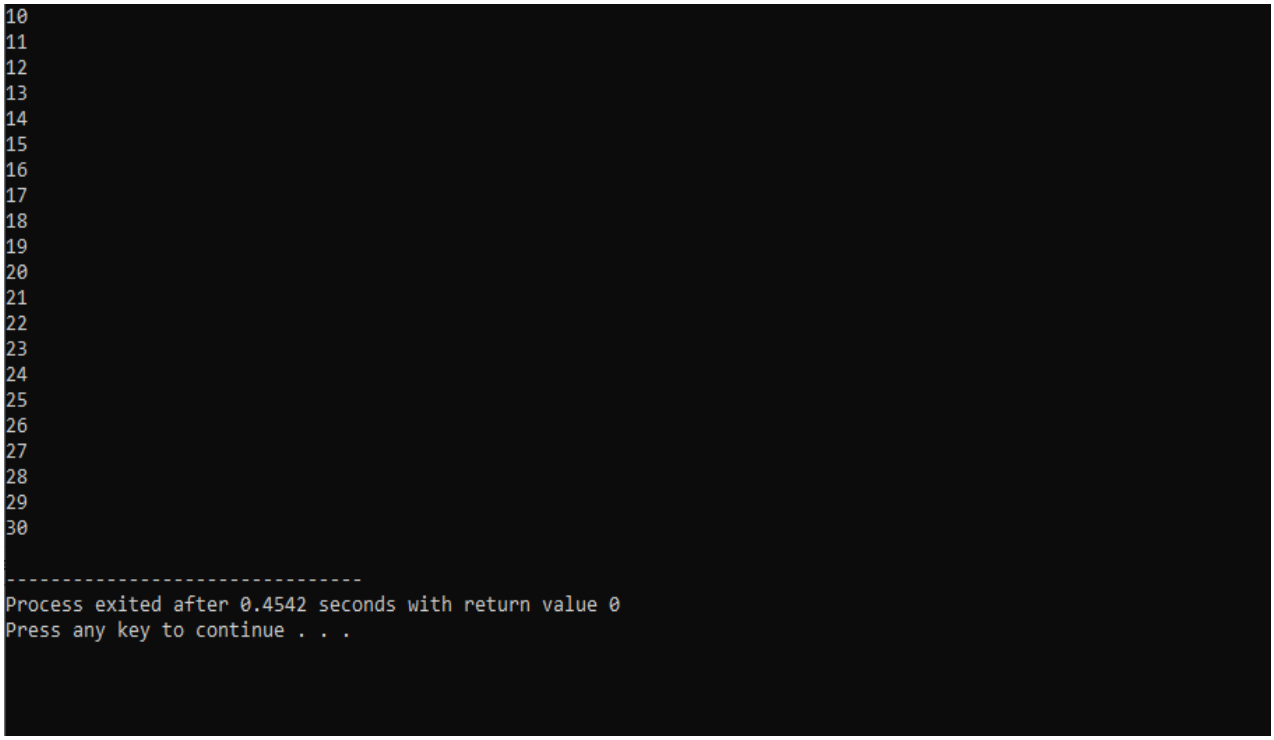
Answer

A. 10

Please mix different assessment tools for triangulation and relevancy of assessment

Practical assessment

1. Write a C program that could display the following output.



Answer

```
#include <stdio.h>
int main(int argc, char *argv[]) {

        int i=10;
        do
    {
    printf("%d\n",i);
        i++;

    }while(i<=30);
```

```
  printf( "   ");

        return 0;
}
```

2. Apply C programming language

Checklist	Score	
	YES	**NO**
Indicator: Programming language is well used		
Data types		
Variables (and constants)		
Operators		
Functions		
Conditions		
Libraries		
Variables (and constants)		
Observation		

Learning outcome 2.2: Program the microcontroller (Arduino)

Duration: 5hrs

Learning out come 2 objectives:

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

1. Software (tools) used to program microcontrollers
2. Program a microcontroller using C programming language concept
3. Program microcontroller Interfaces.

Resources

Equipment	Tools	Materials
➢ PC ➢ Projector ➢ Microcontrollers	➢ Software ➢ Whiteboard	➢ Markers

Advance preparation:

- Availability of Materials, tools and equipment
- Proper preparation of working place
- Availability of electricity

Indicative content 2.2.1: Software (tools) used to program microcontrollers

- Software (tools) used to program microcontrollers:
  - ✓ Code editor

C is what's referred to as a compiled **language**, meaning you have to use a compiler to turn the code into an executable file before you can run it. The

code is written into one or more text files, which you can open, read and edit in any text **editor.** The process of editing, compiling, running, and debugging programs is often managed by a single integrated application known as an Integrated Development Environment, or IDE for short.

- ✓ Assembler

An **assembler** is a program that translates assembly language (low-level language) source code into machine code (typically, in the form of object files). Assembly language is specific to the CPU architecture you're targeting, so there are many different assembly languages. And there are often multiple dialects (regional variety language) of assembly language for a specific CPU architecture.

- ✓ C compiler

A **compiler** is a special program that processes statements written in a particular programming language and turns them into machine language or "code" that a computer's processor uses. Typically, a programmer writes language statements in a language such as Pascal or **C** one line at a time using an editor.

- ✓ Programmer

Programming Languages

A program is a set of instructions that tells a computer what to do in order to come up with a solution to a particular problem. Programs are written using a programming language. A **programming language** is a formal language designed to communicate instructions to a computer. There are two major types of programming languages: low-level languages and high-level languages.

So for as programming language concern these are of two types.

1) Low level language
2) High level language

Low level languages:

Low level languages are **machine level** and **assembly level language**. In machine level language computer only understand digital numbers i.e. in the form of 0 and 1. So, instruction given to the computer is in the form binary digit, which is difficult to implement instruction in binary code. This type of program is not portable, difficult to maintain and also error prone. The **assembly language** is on other hand modified version of machine level language. Where instructions are given in English like word as ADD, SUM, MOV etc. It is easy to write and understand but not understand by the machine. So the translator used here is assembler to translate into machine level. Although language is bit easier, programmer has to know low level details related to low level language. In the assembly level language the data are stored in the computer register, which varies for different computer. Hence it is not portable

High level language:

These languages are machine independent, means it is portable. The language in this category is Pascal, Cobol, Fortran etc. High level languages are understood by the machine. So it need to translate by the translator into machine level. A translator is software which is used to translate high level language as well as low level language into machine level language.

Three types of translator are: **Compiler, interpreter and assembler .**

Compiler and interpreter are used to convert the high-level language into machine level language. The program written in high level language is known as source program and the corresponding machine level language program is called as object program.

Both compiler and interpreter perform the same task but there working is different. Compiler read the program at-a-time and searches the error and lists them. If the program is error free then it is converted into object program. When program size is large then compiler is preferred. Whereas interpreter read only one line of the source code and convert it to object code. If it check error, statement by statement and hence of take more time.

- **Compiler** A compiler is a language translator that converts high level programs into machine understandable machine codes. In this process, the compiler converts the whole program to machine code at a time. If there are any syntactic or semantic error, the compiler will indicate them. It checks the whole program and displays all errors. It is not possible to execute the program without fixing those errors.

- **Interpreter** An interpreter is also a language translator that converts high level programs into machine codes. Unlike compilers, interpreters convert the source code to machine code line by line. As it checks line by line, the scanning time is lower. But the overall execution time is higher. Interpreter displays an error at a time. The programmer should fix that error to interpret the next line. Programming languages such as Python, Ruby, PHP, Perl are some examples of interpreter-based languages.

- **Assembler** In addition to high level languages and machine language, there is another language called the assembly language. Assembly language is in between the high level languages and

machine language. It is closer to machine language than high level languages. It is also called low level language. This language is not easily readable and understandable by the programmer like a high level programming language. The assembler works as the translator in converting the assembly language program to machine code.

Difference Between Compiler Interpreter and Assembler

A compiler is a software that converts programs written in a high level language into machine language. An interpreter is a software that translates a high level language program into machine language while an assembler is a software that converts programs written in assembly language into machine language.

Therefore, Compiler, Interpreter and Assembler are language translators. The difference between compiler interpreter and assembler is that compiler converts whole high level language programs to machine language at a time while interpreter converts high level language programs to machine language line by line and assembler converts assembly language programs to machine language.

Difference between the high and the low level language

1. Low level language is machine readable form of program. Whereas the high level language will be in human readable form.
2. Low level language are difficult to write and compile but high level languages are easy to write as well as compile.
3. Low level language are compact and require less memory space. High level language uses compilers and interpreters which requires large memory space.

4. In high level language debugging (troubleshooting) .I.e. Finding and correcting errors are easier whereas debugging in the low level language is quite difficult.

5. Low level language coding and compiling is time consuming process whereas high level language coding and compiling is much easy and takes very less time to compile.

Indicative content 2.2.2: Program a microcontroller using C programming language concept

Summary for the trainer related to the indicative content (key notes using bullets such as ticks etc)

- Program a microcontroller using C programming language concept

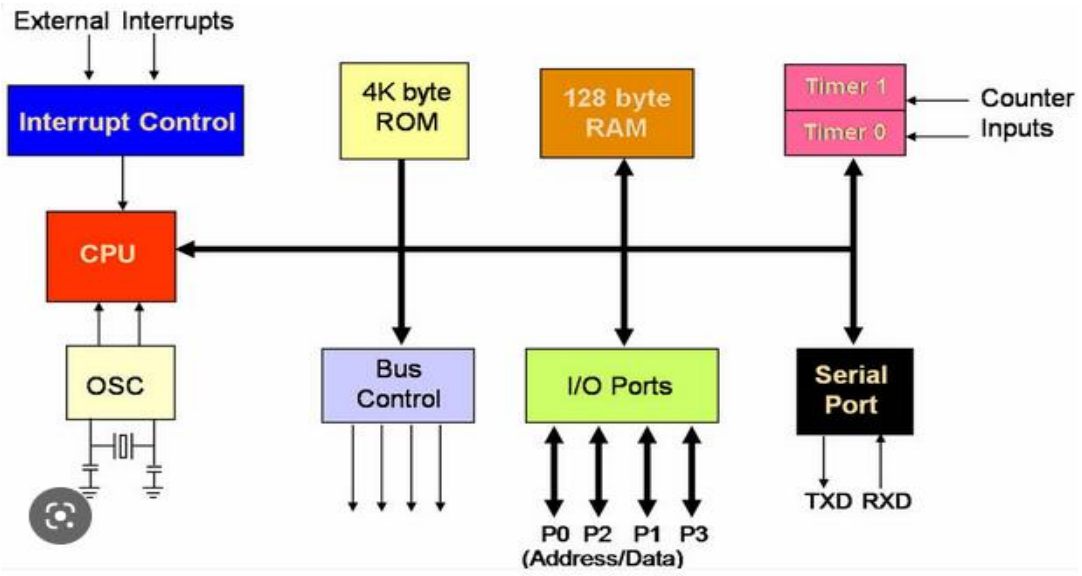
C programming is very important for small microcontrollers. Embedded C language is used to develop microcontroller-based applications. Embedded C is an extension to the C programming language including different features such as addressing I/O, fixed-point arithmetic, multiple-memory addressing, etc. In embedded C language, specific compilers are used. Therefore, C programming language, which is used to create programs for microcontrollers, has been extended to create embedded c However, you must comprehend the distinction between C and Embedded C if you intend to start programming embedded computers using embedded C.

Indicative content 2.2.3: Program microcontroller Interfaces

Summary for the trainer related to the indicative content (key notes using bullets such as ticks etc)

- Program microcontroller Interfaces

An interface is the circuitry that connects an embedded controller to the outside world.Interface is also defined as the path for communication between two components. Interfacing is of two types, **memory interfacing and I/O interfacing**.



Summary for the trainer related to the indicative content(key notes using bullets such as ticks etc)

The difference between low and high level language:

1. Low level language is machine readable form of program. Whereas the high level language will be in human readable form.
2. Low level language are difficult to write and compile but high level languages are easy to write as well as compile.

Theoretical learning Activity

- ✓ Ask trainees to brainstorm about discussion of low and high level language within groups

Practical learning Activity

- ✓ Trainees in pair perform to Program the microcontroller (Arduino)

Points to Remember (Take home message)

1.Software (tools) used to program microcontrollers 2. The difference between low and high level languages

Learning out come 2 formative assessment

Written assessment

1. The following items are software (tools) used to program microcontrollers, except:

a) C compiler

b) Assembler

c) Code editor

d) Arduino board

Answer

d) Arduino board

2. Differentiate low and high level languages

Answer

Low level language is machine readable form of program. Whereas the high level language will be in human readable form.

 **Please mix different assessment tools for triangulation and relevancy of assessment**

Practical assessment

1. Read carefully the following codes and correct errors if any.

```
void main
{
int n;
printf (“enter a number:”)
scanf (“%d”, n);
If (n%2==0);
printf (“even number”);
else
printf(“odd number”);
```

Answer

```
void main( )
{
int n;
printf (“enter a number:”);
scanf (“%d”, &n);
```

If (n%2==0)

printf (“even number”);

else

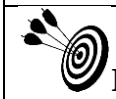
printf(“odd number”); }

2. Program the microcontroller (Arduino)

Checklist	Score	
	YES	**NO**
Indicator: Software (tools) used to program microcontrollers are well applied		
Code editor		
Assembler		
C compiler		
Observation		

Learning outcome 2.3: Test the microcontroller codes (Arduino)

 **Duration: 5hours**

 Learning out come 3 objectives:

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

Install the Arduino environment (IDE)

 **Resources**

Equipment	Tools	Materials
➢ PC ➢ Microcontroller ➢ Projector	➢ IDE Software ➢ Whiteboard	➢ USB cable ➢ Connecting cables ➢ Markers

Advance preparation:

Availability of Materials, tools and equipment

Proper preparation of working place

Availability of electricity

Indicative content 2.2.3: Install the Arduino environment (IDE)

- Install the Arduino environment (IDE)
  - ✓ Connect the board to
  - ✓ computer via UBS cable
  - ✓ Install the drivers
  - ✓ Launch the Arduino IDE
  - ✓ Select your board
  - ✓ Select your serial port
  - ✓ Write your program /sketch
  - ✓ Verify/ compile your sketch
  - ✓ Correct / debug errors

Summary for the trainer related to the indicative content(key notes using bullets such as ticks etc)

The steps used to install the Arduino environment (IDE):

1. Connect the board to
2. Computer via UBS cable
3. Install the drivers
4. Launch the Arduino IDE
5. Select your board
6. Select your serial port
7. Write your program /sketch
8. Verify/ compile your sketch
9. Correct / debug errors


Theoretical learning Activity

- ✓ Ask trainees to brainstorm about the procedures used to install the Arduino environment (IDE) within groups


Practical learning Activity

- ✓ Trainees in pair perform Arduino microcontroller codes testing.

 Points to Remember (Take home message)

- The steps used to install the Arduino environment (IDE)
- Develop different program/projects using Arduino

 Learning outcome 3 formative assessment

Written assessment

1. What IDE stands for?

Answer

IDE: Integrated Development Environment

2. Propose the steps used to install the Arduino environment (IDE)

Answer

The steps used to install the Arduino environment (IDE):

1. Connect the board to

2. Computer via UBS cable

3. Install the drivers

4. Launch the Arduino IDE

5. Select your board

6. Select your serial port

7. Write your program /sketch

8. Verify/ compile your sketch

9. Correct / debug errors

Practical learning Activity

- ✓ Trainees in pair perform Arduino microcontroller codes testing.

Please mix different assessment tools for triangulation and relevancy of assessment

Practical assessment

Test Arduino microcontroller codes

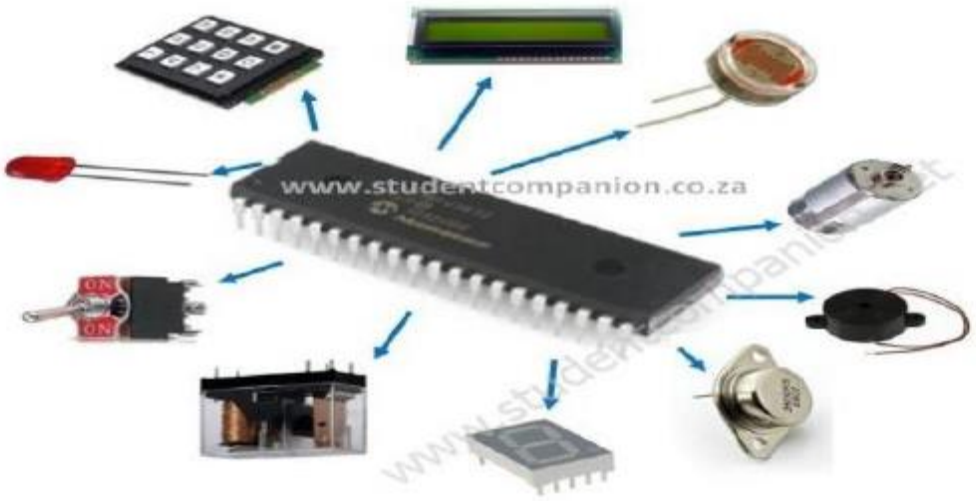
Checklist	**Score**	
	YES	**NO**
Indicator : Installation of the Arduino environment (IDE) is well done		
Connect the board to computer via UBS cable		
Install the drivers		
Launch the Arduino IDE		
Select your board		
Select your serial port		
Open the blink example		
Write your program / sketch		
Verify/ compile your sketch		
Correct / debug errors		
Observation		

References :

- Geier, M. J. (2011). How to diagnose and fix everything electronic. McGraw-Hill/TAB Electronics.
- Lipovski, G. J. (2004). Introduction to microcontrollers: architecture, programming, and interfacing for the Freescale 68HC12. Elsevier.

Learning Unit 3: Make microcontroller based circuit (Arduino)

Picture/s reflecting the Learning unit 3



STRUCTURE OF LEARNING UNIT

Learning outcomes:

3.1: Describe the arduino microcontroller (boards)

3.2: Describe the arduino Integrated Development Environment (IDE)
3.3: Implement simple projects using arduino

Learning outcome 3.1: Describe the Arduino microcontroller hardware interface

Duration: 10hours

Learning out come 1 objectives:

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

Description of Arduino boards types

Resources

Equipment	Tools	Materials
➢ PC ➢ Projector ➢ Arduino Uno ➢ Arduino Nano ➢ Arduino Atmega	➢ Whiteboard ➢ Breadboard/PCB	➢ USB cable ➢ Markers

Advance preparation:

- Availability of Materials, tools and equipment
- Proper preparation of working place
- Availability of electricity

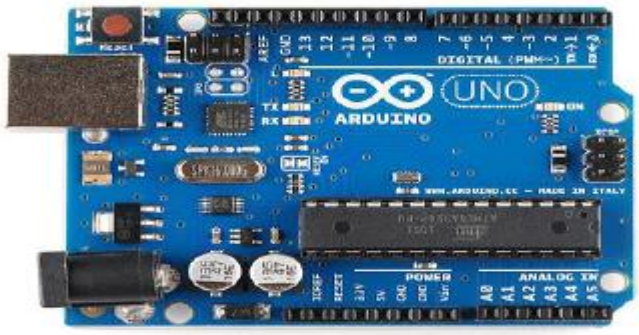
Indicative content 3.3.1: Description of Arduino boards types

- Description of Arduino boards types

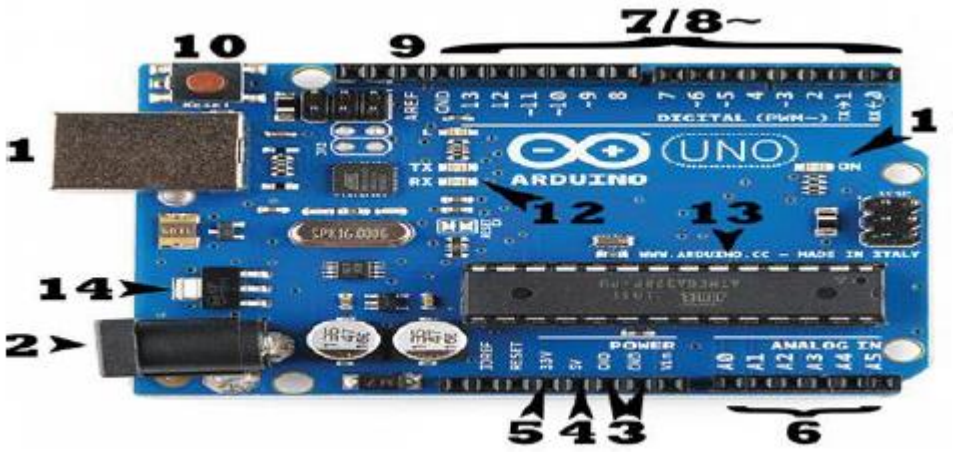
The Arduino microcontroller hardware interface

Arduino is an open-source platform used for building electronics projects. Arduino consists of both a physical programmable circuit board (often referred to as a microcontroller) and a piece of software, or IDE (Integrated Development Environment) that runs on your computer, used to write and upload computer code to the physical board.

The Arduino platform has become quite popular with people just starting out with electronics, and for good reason. Unlike most previous programmable circuit boards, the Arduino does not need a separate piece of hardware (called a programmer) in order to load new code onto the board -- you can simply use a USB cable. Additionally, the Arduino IDE uses a simplified version of C++, making it easier to learn to program. Finally, Arduino provides a standard form factor that breaks out the functions of the micro-controller into a more accessible package. The following Arduino is known as Arduino Uno.



The Uno is one of the more popular boards in the Arduino family and a great choice for beginners. We'll talk about what's on it and what it can do later in this teacher's guide.



Power (USB / Barrel Jack)

Every Arduino board needs a way to be connected to a power source. The Arduino UNO can be powered from a USB cable coming from your computer or a wall power supply (like this) that is terminated in a barrel jack. In the picture above the USB connection is labeled **(1)** and the barrel jack is labeled **(2)**.

The USB connection is also how you will load code onto your Arduino board. More on how to program with Arduino can be found in our Installing and Programming Arduino tutorial.

NOTE: Do NOT use a power supply greater than 20 Volts as you will overpower (and thereby destroy) an Arduino. The recommended voltage for most Arduino models is between 6 and 12 Volts.

Pins (5V, 3.3V, GND, Analog, Digital, PWM, AREF)

The pins on your Arduino are the places where you connect wires to construct a circuit (probably in conjunction with a breadboard and some wire. They usually have black plastic 'headers' that allow you to just plug a wire right into the board. The Arduino has several different kinds of pins, each of which is labelled on the board and used for different functions.

- **GND (3)**: Short for 'Ground'. There are several GND pins on the Arduino, any of which can be used to ground your circuit.
- **5V (4) & 3.3V (5)**: As you might guess, the 5V pin supplies 5 volts of power, and the 3.3V pin supplies 3.3 volts of power. Most of the simple components used with the Arduino run happily off of 5 or 3.3 volts.
- **Analog (6)**: The area of pins under the 'Analog In' label (A0 through A5 on the UNO) are Analog In pins. These pins can read the signal from an analog sensor (like a temperature sensor) and convert it into a digital value that we can read.
- **Digital (7)**: Across from the analog pins are the digital pins (0 through 13 on the UNO). These pins can be used for both digital input (like telling if a button is pushed) and digital output (like powering an LED).
- **PWM (8)**: You may have noticed the tilde (~) next to some of the digital pins (3, 5, 6, 9, 10, and 11 on the UNO). These pins act as normal digital pins, but can also be used for something called Pulse-Width Modulation (PWM).For now, think of these pins as being able to simulate analog output (like fading an LED in and out).
- **AREF (9)**: Stands for Analog Reference. Most of the time you can leave this pin alone. It is sometimes used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the analog input pins.

Reset Button

Just like the original Nintendo, the Arduino has a reset button **(10)**. Pushing it will temporarily connect the reset pin to ground and restart any code that is loaded on the Arduino. This can be very useful if your code doesn't repeat, but you want to test it multiple times. Unlike the original Nintendo however, blowing on the Arduino doesn't usually fix any problems.

Power LED Indicator

Just beneath and to the right of the word “UNO” on your circuit board, there’s a tiny LED next to the word ‘ON’ **(11)**. This LED should light up whenever you plug your Arduino into a power source. If this light doesn’t turn on, there’s a good chance something is wrong. Time to re-check your circuit!

TX RX LEDs

TX is short for transmit, RX is short for receive. These markings appear quite a bit in electronics to indicate the pins responsible for serial communication. In our case, there are two places on the Arduino UNO where TX and RX appear -- once by digital pins 0 and 1, and a second time next to the TX and RX indicator LEDs **(12)**. These LEDs will give us some nice visual indications whenever our Arduino is receiving or transmitting data (like when we’re loading a new program onto the board).

Main IC

The black thing with all the metal legs is an IC, or Integrated Circuit **(13)**. Think of it as the brains of our Arduino. The main IC on the Arduino is slightly different from board type to board type, but is usually from the ATmega line of IC’s from the ATMEL company. This can be important, as you may need to know the IC type (along with your board type) before loading up a new program from the Arduino software. This information can usually be found in writing on the top side of the IC. If you want to know more about the difference between various IC's, reading the datasheets is often a good idea.

Voltage Regulator

The voltage regulator **(14)** is not actually something you can (or should) interact with on the Arduino. But it is potentially useful to know that it is

there and what it’s for. The voltage regulator does exactly what it says -- it controls the amount of voltage that is let into the Arduino board. Think of it as a kind of gatekeeper; it will turn away an extra voltage that might harm the circuit. Of course, it has its limits, so don’t hook up your Arduino to anything greater than 20 volts.
The Arduino Family.

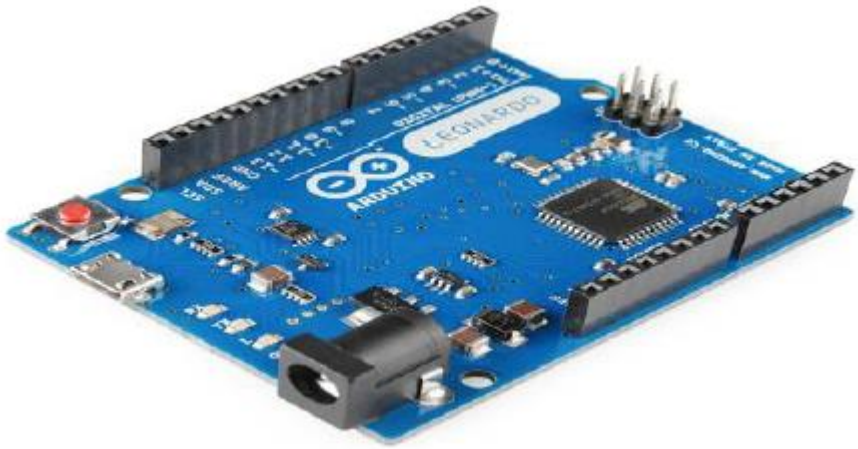
Arduino makes several different boards, each with different capabilities. In addition, part of being open source hardware means that others can modify and produce derivatives of Arduino boards that provide even more form factors and functionality.

Red Board

The RedBoard can be programmed over a USB Mini-B cable using the Arduino IDE. It'll work on Windows 8 without having to change your security settings (we used signed drivers, unlike the UNO). It's more stable due to the USB/FTDI chip we used, plus it's completely flat on the back, making it easier to embed in your projects. Just plug in the board, select "Arduino UNO" from the board menu and you're ready to upload code. You can power the RedBoard over USB or through the barrel jack. The on-board power regulator can handle anything from 7 to 15VDC.

Arduino Mega (R3)

The Arduino Mega is like the UNO's big brother. It has lots (*54!*) of digital input/output pins (14 can be used as PWM outputs), 16 analog inputs, a USB connection, a power jack, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The large number of pins make this board very handy for projects that require a bunch of digital inputs or outputs (like lots of LEDs or buttons).



There are three (3) main types of Arduino:

- ✓ Nano

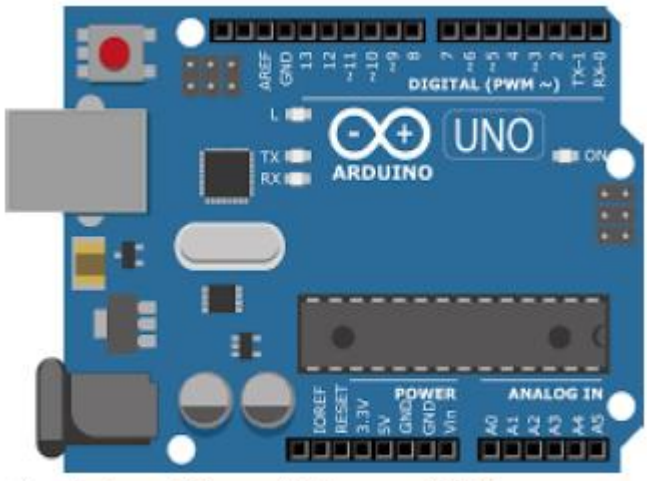
- ➢ **Arduino Nano** is a small, compatible, flexible and breadboard friendly Microcontroller board, developed by Arduino.cc in Italy, based on ATmega328p (Arduino Nano V3.x) / Atmega168 (Arduino Nano V3.x).
- ➢ It comes with exactly the same functionality as in Arduino UNO but quite in small size.

- It comes with an operating voltage of 5V, however, the input voltage can vary from 7 to 12V.
- **Arduino Nano Pin out** contains 14 digital pins, 8 analog Pins, 2 Reset Pins & 6 Power Pins.
- Each of these Digital & Analog Pins are assigned with multiple functions but their main function is to be configured as input or output.
- They are acted as input pins when they are interfaced with sensors, but if you are driving some load then use them as output.
- Functions like pinMode() and digitalWrite() are used to control the operations of digital pins while analogRead() is used to control analog pins.
- The analog pins come with a total resolution of 10bits which measure the value from zero to 5V.
- Arduino Nano comes with a crystal oscillator of frequency 16 MHz. It is used to produce a clock of precise frequency using constant voltage.
- There is one limitation using Arduino Nano i.e. it doesn't come with DC power jack, means you can not supply external power source through a battery.
- This board doesn't use standard USB for connection with a computer, instead, it comes with Mini USB support.
- Tiny size and breadboard friendly nature make this device an ideal choice for most of the applications where a size of the electronic components are of great concern.
- Flash memory is 16KB or 32KB that all depends on the Atmega board i.e Atmega168 comes with 16KB of flash memory while Atmega328 comes with a flash memory of 32KB. Flash memory is used for storing code. The 2KB of memory out of total flash memory is used for a bootloader.

Arduino Nano

- ✓ Uno

The **Arduino Uno** is one kind of microcontroller board based on ATmega328, and Uno is an Italian term which means one. Arduino Uno is named for marking the upcoming release of microcontroller board namely **Arduino Uno Board 1.0**. This board includes digital I/O pins-14, a power jack, analog i/ps-6, ceramic resonator-A16 MHz, a USB connection, an RST button, and an ICSP header. All these can support **the microcontroller** for further operation by connecting this board to the computer. The power supply of this board can be done with the help of an AC to DC adapter, a USB cable, otherwise a battery. This article discusses what is an **Arduino Uno microcontroller**, pin configuration, **Arduino Uno specifications or features**, and applications.



Arduino Uno ATmega328

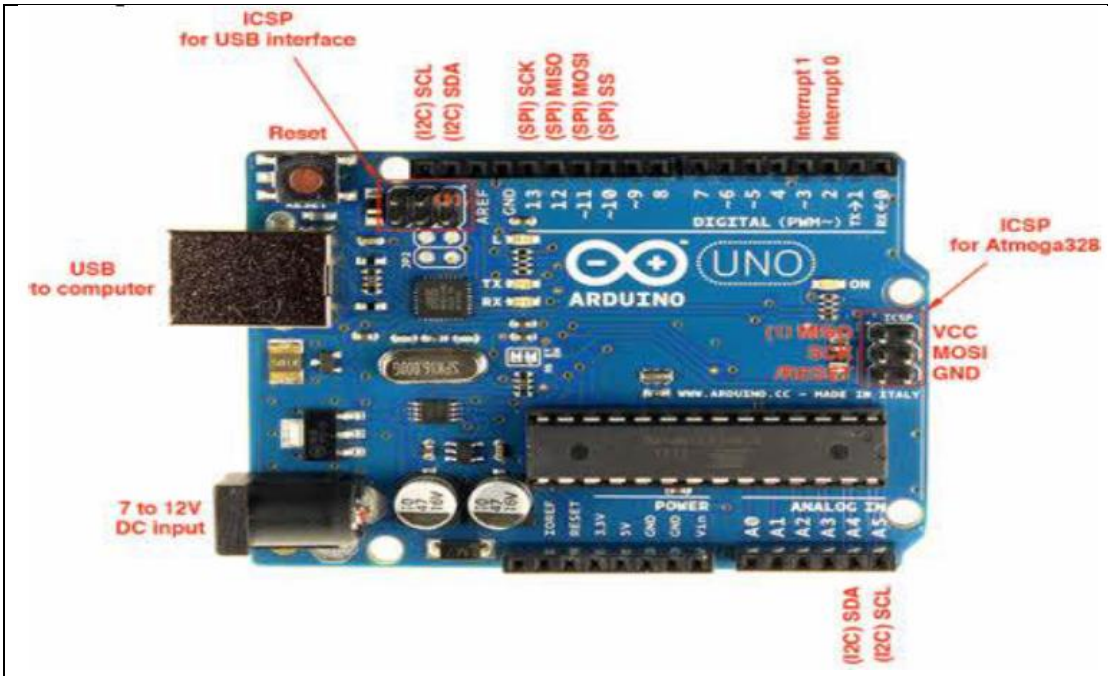
Features of Arduino Uno Board

The features of Arduino Uno ATmega328 includes the following:

- The operating voltage is 5V
- The recommended input voltage will range from 7v to 12V
- The input voltage ranges from 6v to 20V
- Digital input/output pins are 14
- Analog i/p pins are 6
- DC Current for each input/output pin is 40 mA
- DC Current for 3.3V Pin is 50 mA
- Flash Memory is 32 KB
- SRAM is 2 KB
- EEPROM is 1 KB
- CLK Speed is 16 MHz

Arduino Uno Pin Diagram

The Arduino Uno board can be built with power pins, analog pins, ATmegs328, ICSP header, Reset button, power LED, digital pins, test led 13, TX/RX pins, USB interface, an external power supply. The **Arduino UNO board description** is discussed below.



Arduino Uno Board Pin Configuration

Power Supply

The **Arduino Uno power supply** can be done with the help of a USB cable or an external power supply. The external power supplies mainly include AC to DC adapter otherwise a battery. The adapter can be connected to the Arduino Uno by plugging into the power jack of the Arduino board. Similarly, **the battery** leads can be connected to the Vin pin and the GND pin of the POWER connector. The suggested voltage range will be 7 volts to 12 volts.

Input & Output

The 14 digital pins on the Arduino Uno can be used as input & output with the help of the functions like pinMode(), digitalWrite(), & Digital Read(). **Pin1 (TX) & Pin0 (RX) (Serial):** This pin is used to transmit & receive TTL serial data, and these are connected to the ATmega8U2 USB to TTL Serial chip equivalent pins.

Pin 2 & Pin 3 (External Interrupts): External pins can be connected to activate an interrupt over a low value, change in value.

Pins 3, 5, 6, 9, 10, & 11 (PWM): This pin gives 8-bit PWM o/p by the function of analogWrite().

SPI Pins (Pin-10 (SS), Pin-11 (MOSI), Pin-12 (MISO), Pin-13 (SCK): These pins maintain SPI-communication, even though offered by the fundamental hardware, is not presently included within the Arduino language.

Pin-13(LED): The inbuilt LED can be connected to pin-13 (digital pin). As the HIGH-value pin, the light emitting diode is activated, whenever the pin is LOW.

Pin-4 (SDA) & Pin-5 (SCL) (I2C): It supports TWI-communication with the help of the Wire library.

AREF (Reference Voltage): The reference voltage is for the analog i/ps with analogReference().

Reset Pin: This pin is used for reset (RST) the microcontroller.

Memory

The memory of this Atmega328 Arduino microcontroller includes flash memory-32 KB for storing code, SRAM-2 KB EEPROM-1 KB.

- ✓ Atmega

The **ATmega328** is a single-chip microcontroller created by Atmel in the mega AVR family. It has a modified Harvard architecture 8-bit RISC processor core.

The Atmel 8-bit AVR RISC-based microcontroller combines 32 kB ISP flash memory with read-while-write capabilities, 1 kB EEPROM, 2 kB SRAM, 23 general purpose I/O lines, 32 general purpose working registers, three flexible timer/counters with compare modes, internal and external interrupts, serial programmable USART, a byte-oriented 2-wire serial interface, SPI serial port, 6-channel 10-bit A/D converter programmable watchdog timer with internal oscillator, and five software selectable power

saving modes. The device operates between 1.8-5.5 volts. The device achieves throughput approaching 1 MIPS per MHz.

ATmega328 AVR Microcontroller Chip

Summary for the trainer related to the indicative content(key notes using bullets such as ticks etc)

The three (3) main types of Arduino:

- Nano
- Uno
- Atmega

Theoretical learning Activity

- ✓ Ask trainees to brainstorm about discussion of Arduino board types within groups

Practical learning Activity

- ✓ Trainees in pair to Identify Arduino board types

 Points to Remember (Take home message)

Description of Arduino types

 Learning outcome 1 formative assessment

Written assessment

1. Why is it necessary to compile Arduino program before uploading.

Answer

Compiling before uploading help us to check and correct errors.

2. Choose the right answer.

The following items are types of Arduino boards except:

a. Uno

b. Nano

c. Atmega

d. ROM

Answer

d. ROM

 **Please mix different assessment tools for triangulation and relevancy of assessment**

Practical assessment

Identify Arduino board types

Checklist	**Score**	
	YES	**NO**
I**ndicator**: Arduino board types are described		
Uno		
Nano		
Atmega		
Observation		

Learning outcome 3.2: Interconnect the circuit components on the breadboard

Duration: 10hours

Learning out come 1 objectives:

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

1. Identify microcontroller based electronic circuit elements
2. Connect electronic components to the microcontroller
3. Sample projects

Resources

Equipment	Tools	Materials
➢ PC ➢ Microcontrollers ➢ Projector	➢ Whiteboard ➢ LCD ➢ RFID ➢ Keypad ➢ Motor ➢ Remote control	➢ Electronic components ➢ Sensors ➢ Lamps ➢ Markers ➢ USB cable

Advance preparation:

- Availability of Materials, tools and equipment
- Proper preparation of working place
- Availability of electricity

Indicative content 3.2.1: Identify microcontroller based electronic circuit elements

- Identify microcontroller based electronic circuit elements

  ✓ Power supply

There are three main DC voltage sources available to supply power for our microcontroller projects: **Batteries, wall adapters or the USB port of a computer**. Generally, the power level requirement is dictated by the requirements of the devices that you use to build the circuit.

  ✓ Arduino Microcontroller

Arduino consists of both **a physical programmable circuit board** (often referred to as a microcontroller) and a piece of software, or IDE (Integrated Development Environment) that runs on your computer, used to write and upload computer code to the physical board.

  ✓ Electronic components

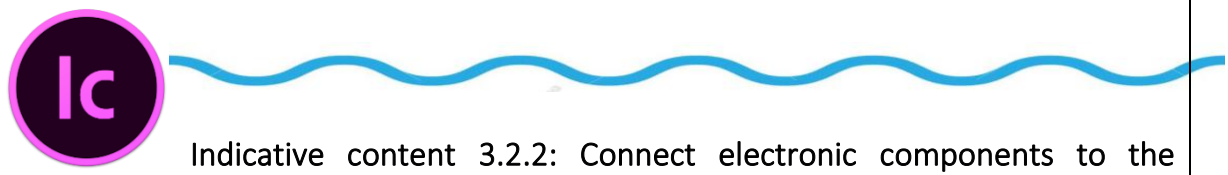
Nearly every microcontroller consists of five main components: the **CPU, RAM, Digital Converters, Serial Bus Interface, and Input/Output ports.**

  ✓ Connecting wires

The system bus is the connective wire that links all components of the microcontroller together. Therefore, wires used to connect microcontroller with its peripherals is called jumper wires.

✓ Load

A microcontroller's "load" varies depending on what you're doing on it. This means that if you're taking *it as a* resistive load.



Indicative content 3.2.2: Connect electronic components to the microcontroller

Summary for the trainer related to the indicative content (key notes using bullets such as ticks etc)

- Connect electronic components to the microcontroller

Use the USB cable to connect the microcontroller to your computer. Plug the USB-C end into the port on the microcontroller, and plug the USB end into a USB port on your computer. If your computer only has USB-C ports, you can use a USB-C to USB-C connector. Therefore,

A microcontroller is embedded inside of a system to control a singular function in a device. It does this by interpreting data it receives from its I/O peripherals using its central processor. The temporary information that the microcontroller receives is stored in its data memory, where the processor accesses it and uses instructions stored in its program memory to decipher and apply the incoming data. It then uses its I/O peripherals to communicate and enact the appropriate action.

Indicative content 3.2.3: Sample projects

Summary for the trainer related to the indicative content (key notes using bullets such as ticks etc)

- Sample projects

✓ Lighting control

Lighting controls can help save energy – and money – by automatically turning lights off when they're not needed, by reducing light levels when full brightness isn't necessary, or otherwise controlling the lighting in and around your home. Common types of lighting controls include dimmers. Lighting control provides users with the ability to manage energy consumption and support visual requirements. A basic commercial lighting control system will allow the end-user to control the light output, illuminate specific areas when needed, and automatically reduce lighting when a space becomes unoccupied.

✓ Heating control

Microcontroller takes signal from temperature sensor and compare with pre-set value of temperature then take decision when heating device or cooling device would be turned on and the duration of maintained temperature in system.

✓ Motors control

By sending energy pulses to the motor, we retain its torque while controlling the speed. It works because we are changing the average

amount of power to the motor, not its voltage. Microcontrollers are great for precision timing and thus make excellent PWM controllers.

- ✓ Proximity control

A proximity sensor is a noncontact type sensor that detects the presence of an object. Proximity sensors can be implemented using different techniques such as optical, ultrasonic, Hall effect, capacitive, and so on.

- ✓ LCD display

A liquid crystal display, better known as an LCD, is an excellent way for a microcontroller to present visible information. LCDs can display output from the μC such as time, date, and temperature; they can also be used to display the contents of memory, and aid in debugging programs.

- ✓ IR remote control

An IR Remote Control sends out infrared light signals. You can't see infrared lights with your eyes, however, it may be visible with the use of a digital camera, cellphone camera, or camcorder. IR remote controls have the (IR) symbol.

The input signals or the commands are sent from a transmitter using IR transmission and received by the IR receiver, processed and used to drive the loads. At both the transmitter and receiver, a microcontroller is used to process the signals.

- ✓ Keypad control

Keypads are widely used input devices being used in various electronics and embedded projects. They are used to take inputs in the form of numbers and alphabets, and feed the same into system for further processing. In this tutorial we are going to interface a 4x4 matrix keypad with 8051 microcontroller.

✓ RFID control

RFID stands for Radio Frequency Identification. RFID module can read or write small amount of data into a Passive RFID tag, which can be used in identification process in various systems like Attendance system, security system, voting system etc.

 Summary for the trainer related to the indicative content (key notes using bullets such as ticks etc)

Identification of microcontroller based electronic circuit elements:

- Power supply
- Arduino Microcontroller
- Electronic components
- Connecting wires
- Load

Sample projects

- Lighting control
- Heating control
- Proximity control
- LCD display
- Keypad control
- Etc


Theoretical learning Activity

- ✓ Ask trainees to brainstorm about basic elements used with microcontroller to create a programmable electronic circuit within groups


Practical learning Activity

- ✓ Trainees in pair perform to Interconnect the circuit components on the breadboard


Points to Remember (Take home message)

Basic elements used with microcontroller to create a programmable electronic circuit


Learning outcome 2 formative assessment

Written assessment

1. Reply by True and False

RFID in full words:

a) Radio Frequency Identification
b) Rwanda Federation Identity

Answer

a) True

b) False

2. Identify five (5) basic elements used with microcontroller to create a programmable electronic circuit

Answer

- Power supply
- Arduino Microcontroller
- Electronic components
- Connecting wires
- Load

 **Please mix different assessment tools for triangulation and relevancy of assessment**

Practical assessment

Interconnect the circuit components on the breadboard

Checklist	**Score**	
	YES	**NO**
Indicator: Microcontroller based circuit is well designed		
Power supply		
Microcontroller		
Electronic components		
Connecting wires		
Load		
Observation		

Learning outcome 3.3: Test the microcontroller based circuit functionality

Duration: 10hours

Learning out come 3 objectives:

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

Testing process

Resources

Equipment	Tools	Materials
➢ PC ➢ Microcontroller	➢ IDE software	➢ USB cable

Advance preparation:

- Availability of Materials, tools and equipment
- Proper preparation of working place
- Availability of electricity

Indicative content 3.3.1: Testing process

- Testing process

✓ Connect Arduino circuit to IDE
✓ Assign a port to microcontroller
✓ Upload a complied program to Arduino circuit
✓ Check the functionality of designed circui

Summary for the trainer related to the indicative content ((key notes using bullets such as ticks etc)

The testing process:

1. Connect Arduino circuit to IDE
2. Assign a port to microcontroller
3. Upload a complied program to Arduino circuit
4. Check the functionality of designed circuit

Theoretical learning Activity

✓ Ask trainees to brainstorm about the processes used to test microcontroller based circuit functionality within groups

Practical learning Activity

- ✓ Trainees in pair perform to test the microcontroller based circuit functionality

Points to Remember (Take home message)

The processes used to test microcontroller based circuit functionality

Learning out come 3 formative assessment

Written assessment

1. List two (2) equipment could you use when you want test microcontroller project.

Answer

a) PC
b) Microcontroller/Arduino board

2. The following statements are processes used to test microcontroller based circuit functionality, except:

a) Connect Arduino circuit to IDE
b) Assign a port to microcontroller
c) Downloaded software used to make a microcontroller project
d) Upload a complied program to Arduino circuit
e) Check the functionality of designed circuit

Answer

c) Downloaded software used to make a microcontroller project

Please mix different assessment tools for triangulation and relevancy of assessment

Practical assessment

Test the microcontroller based circuit functionality

Checklist	Score	
	YES	NO
Indicator: Testing processes are applied		
Connect Arduino circuit to IDE		
Assign a port to microcontroller		
Upload a complied program to Arduino circuit		
Check the functionality of designed circuit		
Observation		

Integrated assessment

Integrated Situation

UMURAVA Ltd is a company specialized in mining, wishes to modernize the transportation of minerals production from mine-hole to main stock. This last is located at ten meters. A big concern is that a good quantity of minerals is stolen during manhood-head transportation.

They need to construct the permanent conveyor-belt based transportation system to avoid that issue.

As long as they deposit production on conveyor, lamps on the way must light-up until it reaches to the stock. The stock door should be automated to open immediately and then closes after the production gets inside.

As one of their specialized computer technician, you are required to implement this conveyor (belt) system, using Arduino microcontroller, to facilitate the replacing of old head-carrying.

The required tools, materials and equipment are available.

Note: The work should be done within 4 Hours.

Assessment Criterion 1: Quality of Process

Checklist	Score	
	YES	**NO**
Indicator: Required Tools, equipment and materials are well selected		
Multimeter		
PPE		
Electronic components		
Breadboard/PCB		
Soldering tin		
Soldering station		
Screwdrivers		
Pliers		
Wires and cables		
Microcontrollers		
Indicator: Arduino IDE program is coded		
The program codes are compiled		
The program codes are uploaded to circuit		
Observation		

Assessment Criterion 2: Quality of product

Checklist	Score	
	YES	**NO**
Indicator: Conveyor belt is implemented		
Microcontroller is used		
Lamps are lighting		
Automated door is implemented		
Observation		

Assessment Criterion 3: Relevance

Checklist	Score	
	YES	**NO**
Indicator: Efficient use of materials		
No material is wasted		
Indicator: Tools are well used		
No tool is damaged		
Indicator: Circuit is well implemented		
Time is expected		
System run effectively		
Observation		

Assesment Criterion 4: Safety

Checklist	Score	
	YES	**NO**
Indicator: PPE are well used		
Gloves		
Overall		
Safety shoes		
Observation		

References:

- D.Ibrahim, PIC BASIC Projects, 30 Projects Using PIC BASIC and PIC BASIC PRO, 2006
- Geier, M. J. (2011). How to diagnose and fix everything electronic. McGraw-Hill/TAB Electronics.
- https://www.build-electronic-circuits.com/microcontroller-basics/