Assignment 1 :

1. What is a Prototype? What are Open source and closed source prototype platforms?

- A prototype is an early sample, model, or release of a product built to test a concept or process.

- Open Source Prototype Platforms: These platforms have their design, software, and specifications publicly available.
  This allows users to modify, distribute, and use the designs without restriction. Examples include Arduino and Raspberry Pi.

- Closed Source Prototype Platforms: These platforms keep their design, software, and specifications proprietary.
  Users cannot legally modify or share the design. An example is the BASIC Stamp microcontroller by Parallax.

2. What is Arduino?

- Arduino is an open-source electronics platform based on easy-to-use hardware and software.
It consists of a microcontroller, an integrated development environment (IDE), and a standard programming language.

3. Write down Arduino Uno R3 Key Specifications:

- Main Processor: ATmega328P
- Memory:
  - SRAM: 2 KB
  - FLASH MEMORY: 32 KB (ATmega328P) of which 0.5 KB used by bootloader
  - EEPROM: 1 KB
- I/O Pins:
  - Digital I/O Pins: 14 (of which 6 provide PWM output)
  - Analog Input Pins: 6

Assignment 2 :

Encoding - A data encoding format is a standardized method for converting data into a specific format for efficient storage, transmission, and interpretation by computers.

Encoding format for different types of data :--

1. Text - ASCII - for basic text encoding
UTF-8 - for unicode text encoding

BASE64 - for encoding binary to ASCII

2. Number - Binary - its the representation of number in Binary format
   IEEE 754 - for floating point numbers
   Hexadecimal - base 16 representation of numbers , etc..

3. Photo - 1. JPEG
   2. PNG
   3. Gif

4. Audio - 1. MP3
   2. WAV
   3. FLAC

5. Video - 1. MP4
   2. AVI (older version of video format)
   3. MKV (open mainly by VLC media player)

Assignment 3 :

Q. Explain the basic structure of the arduino program ?

Ans:-- Arduino program used GCC based compiler and mainly consist of 2 main function :-

1. void setup() :-

   where the code is run only one time when the program start running , it is where all the variables set up are kept .

2. void loop() :--

   this part of the code runs infinitely till the power is on or available

Example of blinking of an led :-

```
// Here we can include the header for the types of project we are working on
// #include<dhtt11.h> - this is for working with dhtt11 sensor
// also we can define the macro program that need to be initialised before the program compile
// define ()
etc...

void setup(){
        pinMode(LED_BUILTIN , OUTPUT);
}

void loop(){
        digitalWrite(LED_BUILTIN, HIGH); // This turn on the light
        delay(1000); // the light is on for 1sec i.e 1000 ms
        digitalWrite(LED_BUILTIN, LOW); // This turn off the light
        delay(1000); // the light is off for 1sec i.e 1000 ms
  }
```

Assignment 4 :

Q. How do UART, I^2C, SPI, CAN, and USB communication protocols differ in terms of data transmission speed, complexity, pin usage, and device-to-device communication? What are the key features that make each protocol suitable for specific applications, and in what types of embedded systems would each be most commonly used ?

1. UART (Universal Asynchronous Receiver/Transmitter)

- Its data transmission speed is slow normally between (9600 bps to 1 Mbps).
- It is simple to use .
- It uses 2 pins TX and RX.
- It mainly communicates between 1 to 1 device . But multiple can be achieved using additional hardware.
- Some of the key feature of UART are simple , low pin count , reliable for low speed , 1-1 communication and serial communication.
- Some of the application of UART are : Basic microcontroller projects, wireless communication modules, low-power devices, debugging interfaces

2. I^2C (Inter-Integrated Circuit)

- I^2C is a low-to-moderate speed protocol, operating at standard mode (100 kbps), fast mode (400 kbps), fast mode plus (1 Mbps), and high-speed mode (up to 3.4 Mbps).
- It requires two pins, SDA (data) and SCL (clock), along with pull-up resistors, making it relatively simple to implement.
- I^2C is designed for communication between multiple devices, supporting up to 127 devices on a single bus through an addressing scheme, which makes it ideal for connecting multiple peripherals within an embedded system.
- The protocol is popular for its simplicity and low pin usage, making it particularly suitable for sensor networks and peripheral interfaces on a single PCB.
- Typical applications include sensor connections (like temperature and pressure sensors), LCD displays, EEPROMs, and small modules in compact, low-power systems such as wearables, portable medical devices, and consumer electronics.

3.SPI (Serial Peripheral Interface)

- SPI is a high-speed communication protocol typically operating between 1 Mbps and 50 Mbps, making it much faster than I^2C.
- It requires four main pins—MOSI (Master Out Slave In), MISO (Master In Slave Out), SCLK (Clock), and CS/SS (Chip Select/Slave Select).
- Each slave device requires its own CS/SS line, which can add to pin complexity in systems with multiple devices.

- SPI is full-duplex, allowing simultaneous data transmission and reception, making it suitable for applications that need high-speed, continuous data flow.
- This protocol’s high speed and simplicity make it ideal for applications that require fast data transfer, such as SD cards, flash memory, displays, and ADCs/DACs in embedded systems.
- SPI is commonly used in IoT devices, data acquisition systems, and consumer electronics like smart thermostats, gaming devices, and industrial control systems where real-time data processing is essential.

4. CAN (Controller Area Network)

- CAN operates at moderate speeds, typically from 125 kbps to 1 Mbps, with CAN FD supporting up to 5 Mbps.
- It uses two pins, CAN_H and CAN_L, for differential signaling, which helps maintain reliable communication in electrically noisy environments.
- CAN’s message-based, multi-master architecture allows multiple devices to communicate on the same bus with prioritized messages, ensuring critical data is transmitted first.
- CAN’s fault tolerance, error detection, and prioritization make it highly reliable, making it the go-to protocol for safety-critical applications.
- It’s widely used in automotive systems (engine control units, airbag systems), industrial automation, and robotics where reliability and real-time control are essential.
- CAN is also common in medical devices where safety and data integrity are paramount.

5. USB (Universal Serial Bus)

- USB offers high data transfer rates, ranging from 1.5 Mbps (USB 1.0) to 5 Gbps (USB 3.0 and higher).
- While it’s complex, requiring advanced protocol management and power handling, USB is a plug-and-play standard, providing both power and data transmission over four main pins (VCC, GND, D+, and D-).
- USB primarily uses a host-device communication model but supports multiple devices connected through hubs, allowing hot-swapping of devices.
- The protocol’s speed, ease of connectivity, and ability to power peripheral devices make USB ideal for consumer electronics, personal computers, data acquisition devices, and portable medical devices.
- It’s commonly used for connecting external storage, peripherals like keyboards and mice, and charging and data transfer for mobile devices.
- USB’s wide adoption makes it a standard choice for high-speed data transfer and peripheral communication.