INTERNET OF THINGS ASSIGNMENT RECORD

Subject code : BTCS-AMDS-SP8P

Name:	**Suraj Singha**
Registration Number:	**FET-BAML-2022-2026-030**
Course:	**Btech CSE AIML**
Semester:	**5th**
Faculty:	**Mr. Biswajeeban Mishra & Mr. Pritam Nanda**

Remarks	
Signature	

SRI SRI UNIVERSITY

Bidyadharpur, Cuttack, Odisha.

Index

Sl. No.	Date	Experiment/Case Study	Page No.	Remark
1		Prototyping and Arduino uno r3		
2		Encoding formats.		
3		Basic Structure of an Arduino Program.		
4		Difference between UART, I²C, SPI, CAN, and USB		
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				

IoT & Automation Lab.

Assignment-1

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

A prototype is a basic, rough version of a product created to test ideas and show how it works. It helps understand if the design is right and gathers feedback before making the final version.

Open-source software is software whose code is open to everyone. Everybody can access, use, modify, or distribute it freely. This fosters community inputs and helps in team working. Some examples of these are Firefox, MySQL, and Arduino.

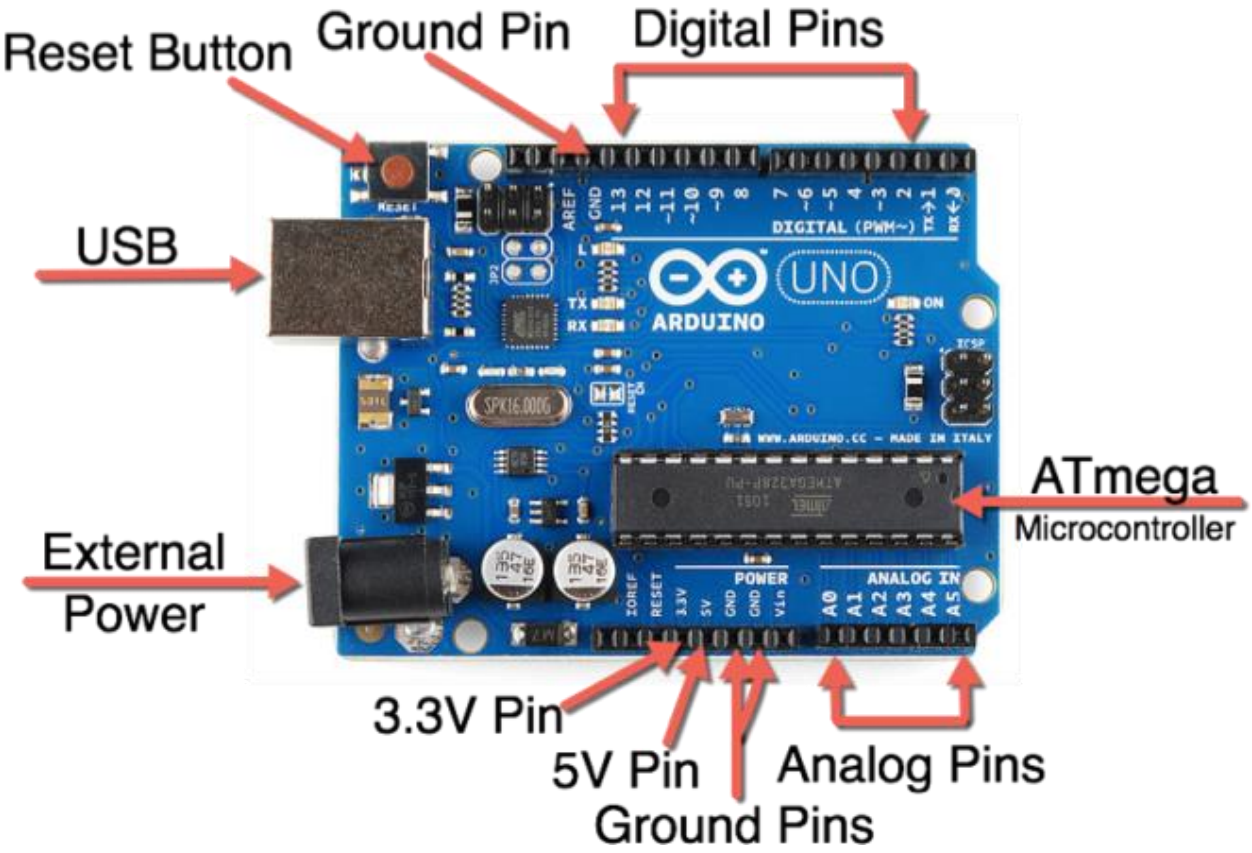
Closed-source software is just software whose code is kept private and owned by a particular company. The public cannot access or modify it, and more often than not, some license or subscription fees are paid for it. Examples include Skype, Microsoft Office, and Adobe Reader.

2. **What is Arduino?**

Arduino is an open-source electronic prototyping platform allowing easy creation and development of both hardware and software. This helps in making a project interactive by reading inputs example,

press a button and making outputs, such as turning on a light. It has found wide application in the education sector, prototyping new ideas, and DIY projects, such as home automation and wearable technology.

3. Write down Arduino Uno R3 Key Specifications:



Main Processor:

- **Microcontroller:** ATmega328P (8-bit RISC processor)

Memory:

- **SRAM:** 2 KB (for temporary data)

- **Flash Memory:** 32 KB (for storing the application code; 0.5 KB used by the bootloader)
- **EEPROM:** 1 KB (non-volatile memory for storing data even after power-off)

I/O Pins:

- **Digital I/O Pins:** 14 (6 can provide PWM output)
- **Analog Input Pins:** 6
- **PWM Output Pins:** 6 (Pins 3, 5, 6, 9, 10, 11)
- **UART:** 1 (Serial communication on pins 0 (RX) and 1 (TX))
- **Built-in LED:** Pin 13

Assignment-2

What is an Encoding format? List down encoding formats for various types of data (Text, Number, Photo, Audio, Video).

Encoding format is a standardized way to convert data into a format that computers can read and process. It's similar to translating human language into a language that computers can understand.

Different Encoding Formats:
Text Encoding:
ASCII: Stands for 128 characters, to be more precise letters, numbers, and other punctuation marks.
Unicode: A general type of character set standard that characterize most of the commonly used language as well as the ASCII.

UTF-8: A kind of variable encoded character code that is maintained between ASCII and the web_standards.

Number Encoding:
Binary: Functions only with 0s and 1s only.
Decimal: The system that we us in our daily lives for instance, the base-10 system.
Hexadecimal: Alphabetic characters represented by erotiske by 16 digits from 0 to 9 and A-F.
Floating-point: Can be real numbers written in hindu numerals with a decimal point.

Image Encoding:
JPEG: Some examples include lossy compression, which is best used for photos.
PNG: It is a type of compression which is best used for images having clear cut edges and texts.
GIF: Supports animation and transparency and is used for the simple images.
BMP: A format with no compression which hence means that the size of the file is large.
TIFF: There is no information loss when the images are compressed, it supports different image bit depths.

Audio Encoding:
MP3: Lossy compression the most used format for audio, especially for music.
AAC: Broadcasting and iTune's uses lossy compression.
WAV: Imperfect but will keep all the data as close to original:

lossless compression, high-quality sound.
FLAC: This is free of loss type of compression, it does not reduce on the quality of the audio.

Video Encoding:
MP4: Regular type of a video file storage and sharing.
AVI: Supports various codecs.
MOV: Apple's video format.
WMV: Microsoft's video format.

Assignment-3

Explain Basic Structure of an Arduino Program.

- There are two required parts or functions that enclose blocks of statements.
- **setup()** is the preparation, **loop()** is the execution.
- Both functions are required for the program to work.

```
void setup()
{
statements;
}
void loop()
{
statements;
```

}

The setup() function is called once, when the Arduino board is first turned on or reset. It is used to initialize the board and set up the hardware.

The setup function should follow the declaration of any variables at the very beginning of the program. It is **the first function to run in the program**, is run only once, and is used to set **pinMode** or initialize serial communication.

Setting pinMode: This tells the Arduino **whether a specific pin is going to be used for input** (reading data) or **output** (sending data).

For example, if you have an LED connected to pin 13, you would specify in the setup() function that pin 13 is an output pin.

Initializing Serial Communication: This is like **opening a communication channel** between your Arduino and your computer or another device. This is useful for **sending data back and forth**.

The loop() function is called repeatedly, until the Arduino board is turned off or reset. It is where the Arduino program does most of its work.

The loop function follows next and includes the code to be executed continuously – reading inputs, triggering outputs, etc.

Assignment-4

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 ?

UART (Universal Asynchronous Receiver/Transmitter)

It's simple, reliable, and doesn't need much to work—just a couple of lines (TX and RX). There's no separate clock line, so each side just has to agree on how fast they'll be sending messages.

- **Speed:** Not lightning fast, usually 9600 bps to 1 Mbps, but it gets the job done.
- **Setup Complexity:** Super simple—like setting up a walkie-talkie line. Just two lines (TX and RX).
- **Communication:** It's mostly one-on-one, like chatting with one friend on the other end. You can set it up for more, but that's a bit finicky.
- **Where it Fits:** This is for simpler, short-distance communication—like talking to a GPS module, Bluetooth, or debugging between a microcontroller and PC.

Why Use UART? It's the go-to for simple, straightforward communication and is great for low-data tasks, logging, or basic debugging.

I^2C (Inter-Integrated Circuit)

I^2C can connect a handful of devices, and they each have a unique address, so it's clear who's talking to whom. It only needs two lines: one for data (SDA) and one for clock (SCL).

- **Speed:** A bit slower, typically up to 3.4 Mbps on the high end.
- **Setup Complexity:** A tad more complex than UART since you're coordinating multiple devices on one bus, but not too tricky.
- **Communication:** Multi-master and multi-slave, so it's open for multiple devices, each with a unique address.
- **Where it Fits:** This is the pick for connecting things like sensors, EEPROMs, and RTCs (real-time clocks). Think of a few low-speed peripherals needing simple data like temperature or pressure.

Why Use I^2C? When we have got multiple peripherals (like a few sensors) and want to keep wiring to a minimum, I^2C works well. It's efficient and doesn't hog many GPIO pins.

SPI (Serial Peripheral Interface)

SPI is quick, but we need a lot of lines to keep things moving smoothly. We can see it in setups that need fast, full-duplex data exchange.

- **Speed:** Can handle 10-50 Mbps, so it's fast.
- **Setup Complexity:** Moderate. Needs four lines for each slave device (clock, data in, data out, and chip select), so wiring can add up if you have a lot of devices.

- **Communication:** One master with multiple slaves, but each slave needs its own “hello” (chip select line), which can be a bit bulky.
- **Where it Fits:** SPI’s great for things like displays, SD cards, or audio codecs, where you need high-speed data transfer and don’t mind a few extra wires.

Why Use SPI? When we need speed and we can spare the pins, SPI’s a solid choice. We see it often in high-speed situations, like SD cards or display interfaces.

CAN (Controller Area Network)

CAN is designed to keep everyone in line even in noisy environments (like cars or industrial settings). It’s used mostly in cars, where sensors and modules need to talk reliably.

- **Speed:** It’s moderate at about 1 Mbps (CAN FD goes up to 5 Mbps).
- **Setup Complexity:** Higher complexity; CAN has built-in error-checking and collision management.
- **Communication:** Multi-master, meaning any device on the bus can start talking as long as it has the right priority.
- **Where it Fits:** CAN is ideal for cars and industrial systems where you need reliable communication across many devices.

Why Use CAN? If reliability is non-negotiable and the environment’s a bit noisy, CAN’s a beast. That’s why it’s in vehicles, keeping everything coordinated and safe.

USB (Universal Serial Bus)

USB is like we have a host, and everything else connects to it for quick, structured communication. We already know it well from charging devices and transferring data.

- **Speed:** Blazing fast. From 12 Mbps (USB 1.1) to 5 Gbps or more (USB 3.0+).
- **Setup Complexity:** High. USB needs a detailed setup and strict adherence to protocol, but it's standard.
- **Communication:** Host-based. The host (usually a PC or similar) controls the conversation with each device connected.
- **Where it Fits:** USB is perfect for high-data applications, like external storage or communication with peripherals.

Why Use USB? When we need top-notch speed and a standardized plug-and-play experience, USB's the way to go. It's everywhere for a reason—easy to use and fast.