**INTERNET OF THINGS ASSIGNMENT RECORD**

**Subject code : BTCS-AMDS-009T**

|  |  |
| --- | --- |
| Name: | Meheak Sahu |
| Registration Number: | FET-BAML-2022-2026-025 |
| Course: | B.Tech CSE AIML |
| Semester: | 5th |
| Faculty: | Mr. Biswajeeban Mishra &  Mr. Pritam Nanda |

|  |  |
| --- | --- |
| Remarks |  |
| Signature |  |



SRI SRI UNIVERSITY

Bidyadharpur, Cuttack, Odisha.

Index

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl. No.** | **Experiment/Case Study** | **Page No.** | **Remark** |
| **1** | Prototyping and Arduino UNO R3. |  |  |
| **2** | Encoding formats. |  |  |
| **3** | Basic Structure of an Arduino Program. |  |  |
| **4** | The Architecture of Modern Computers. |  |  |
| **5** | Communication Protocol. |  |  |
| **6** |  |  |  |
| **7** |  |  |  |
| **8** |  |  |  |
| **9** |  |  |  |
| **10** |  |  |  |
| **11** |  |  |  |
| **12** |  |  |  |
| **13** |  |  |  |
| **14** |  |  |  |
| **15** |  |  |  |
| **16** |  |  |  |

***Assignment No.:1 Date: 9-8-2024***

**Q1. *What is a Prototype?***

***What are Open source and closed source prototype platforms?***

**A1.**

* **Prototype**

A prototype is a preliminary model, sample, or version of a product or system used to test a concept or process. It's a tangible representation of an idea, allowing for evaluation, feedback, and refinement before full-scale development. Prototypes can be physical, digital, or a combination of both.

* **Open Source and Closed Source Prototype Platforms**
* Open Source Prototype Platforms: These platforms provide the underlying code and design freely accessible to the public. Users can modify, distribute, and build upon the platform. Examples include Arduino, Raspberry Pi, and open source CAD software like FreeCAD.
* Closed Source Prototype Platforms: These platforms keep the source code proprietary, restricting access and modification. Users typically pay for licenses to use the platform. Examples include many commercial 3D printing software, electronic design automation (EDA) tools, and rapid prototyping machines.

**Q2. *What is Arduino?***

**A2. Arduino** is an open-source electronics platform based on easy-to-use hardware and software. It's designed for artists, designers, hobbyists, and anyone interested in creating interactive objects or environments. Arduino boards are microcontroller-based, meaning they have a small computer on board that can be programmed to control various electronic components.

**Q3. *Write down the Arduino Uno R3 Key Specifications.***

**A3.**

* **Main Processor :**

ATmega328P

* **Memory :**

SRAM: 2 KB

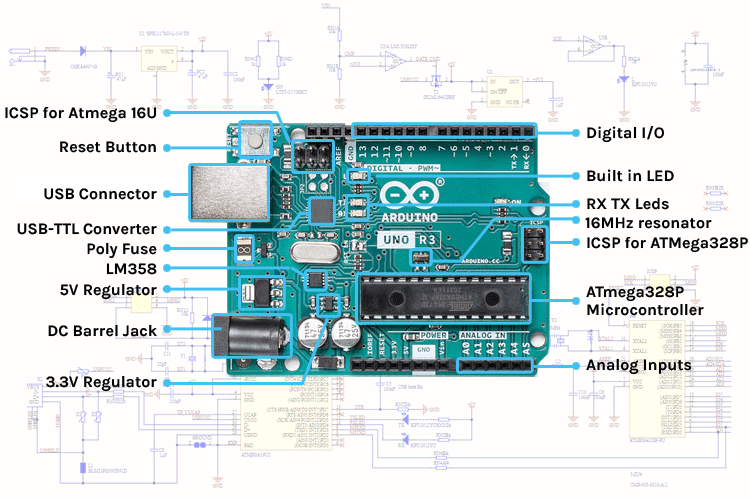
FLASH MEMORY: 32 KB (0.5 KB used by bootloader)

EEPROM: 1 KB

* **I/O Pins :**

Digital I/O pins: 14 (of which 6 can be used as PWM outputs)

Analog input pins: 6



***Assignment No.:2 Date: 9-8-2024***

**Q1. *What is an Encoding format List down encoding formats for various types of data  ( Text, Number, Photo, Audio, Video).***

**A1.** An encoding format is a standardized method for converting data into a specific digital format for efficient storage and transmission. It translates information from human-readable form into a format that computers can interpret. Here’s how encoding formats are used in different contexts.

**Text:**

* [**ASCII (American Standard Code for Information Interchange)**](https://en.wikipedia.org/wiki/ASCII)
* [**UTF-8 (Unicode Transformation Format - 8-bit)**](https://en.wikipedia.org/wiki/UTF-8)
* [**UTF-16 (Unicode Transformation Format - 16-bit)**](https://en.wikipedia.org/wiki/UTF-16)
* [**UTF-32 (Unicode Transformation Format - 32-bit)**](https://en.wikipedia.org/wiki/UTF-32)
* [**ISO 8859-1 (Latin-1)**](https://en.wikipedia.org/wiki/ISO/IEC_8859-1)

**Numbers:**

* [**Binary (Base-2)**](https://en.wikipedia.org/wiki/Binary_number)
* [**Decimal (Base-10)**](https://en.wikipedia.org/wiki/Decimal)
* [**Hexadecimal (Base-16)**](https://en.wikipedia.org/wiki/Hexadecimal)
* [**IEEE 754**](https://en.wikipedia.org/wiki/IEEE_754)
* [**BCD (Binary-Coded Decimal)**](https://en.wikipedia.org/wiki/Binary-coded_decimal)

**Photos or Images:**

* [**JPEG (Joint Photographic Experts Group)**](https://en.wikipedia.org/wiki/JPEG)
* [**PNG (Portable Network Graphics)**](https://en.wikipedia.org/wiki/Portable_Network_Graphics)
* [**GIF (Graphics Interchange Format)**](https://en.wikipedia.org/wiki/GIF)
* [**BMP (Bitmap)**](https://en.wikipedia.org/wiki/BMP_file_format)
* [**TIFF (Tagged Image File Format)**](https://en.wikipedia.org/wiki/TIFF)

**Audio:**

* [**MP3 (MPEG-1 Audio Layer III)**](https://en.wikipedia.org/wiki/MP3)
* [**WAV (Waveform Audio File Format)**](https://en.wikipedia.org/wiki/WAV)
* [**AAC (Advanced Audio Coding)**](https://en.wikipedia.org/wiki/Advanced_Audio_Coding)
* [**FLAC (Free Lossless Audio Codec)**](https://en.wikipedia.org/wiki/FLAC)
* [**OGG (Ogg Vorbis)**](https://en.wikipedia.org/wiki/Vorbis)

**Video:**

* [**MP4 (MPEG-4 Part 14)**](https://en.wikipedia.org/wiki/MPEG-4_Part_14)
* [**AVI (Audio Video Interleave)**](https://en.wikipedia.org/wiki/Audio_Video_Interleave)
* [**MKV (Matroska Video)**](https://en.wikipedia.org/wiki/Matroska)
* [**MOV (QuickTime Movie)**](https://en.wikipedia.org/wiki/QuickTime_File_Format)
* [**WMV (Windows Media Video)**](https://en.wikipedia.org/wiki/Windows_Media_Video)

***Assignment No.:3 Date: 9-8-2024***

***Q1. Explain Basic Structure of an Arduino Program.***

***A1.*** An Arduino program is also called a sketch.

**Basic Structure:**

void setup()

{

statements;

}

void loop()

{

statements;

}

It consists of two main functions:

*1) setup() function*

* Runs only once when the Arduino board starts or resets.
* Used to initialize hardware components, set pin modes (input or output), and define initial values.

*2) loop() function*

* Runs repeatedly after the setup() function completes.
* Contains the core logic of the program.
* This is where the Arduino performs actions and interacts with the environment.

Both functions are enclosed in curly braces {}

Every statement inside these functions is ended with a semicolon ‘ ; ’

**Basic Sketch:**

This simple sketch blinks an LED connected to pin 13.

void setup() {

pinMode(13, OUTPUT); // Set pin 13 as an output

}

void loop() {

digitalWrite(13, HIGH); // Turn LED on

delay(1000); // Wait for 1 second

digitalWrite(13, LOW); // Turn LED off

delay(1000); // Wait for 1 second

}

***Assignment No.:4:***

***Q. The Architecture of Modern Computers***

**1. Parallelism in Modern CPU Architectures (e.g., x86-64 Architecture)**

Modern CPUs like those using the x86-64 architecture are designed to handle various forms of parallelism to enhance performance and efficiency:

* Instruction-Level Parallelism (ILP): Through techniques like *pipelining*, *superscalar execution*, and *out-of-order execution*, multiple instructions can be processed in parallel on a single core. This significantly boosts CPU efficiency by allowing multiple instructions to execute simultaneously.
* Data-Level Parallelism (DLP): Single Instruction, Multiple Data (SIMD) extensions, such as AVX in x86-64, enable the CPU to apply the same operation to multiple data points in parallel, which is especially useful in tasks like multimedia processing.
* Thread-Level Parallelism (TLP): Modern CPUs can process multiple threads simultaneously with multithreading and multicore designs, improving application responsiveness and system throughput.

Implications for Software Design: These parallelism techniques encourage software developers to utilize concurrency models and optimize for scalability to maximize performance on modern CPUs.

2**. Differences Between RISC (Reduced Instruction Set Computing) and CISC (Complex Instruction Set Computing) Architectures**

* CISC (e.g., x86):
  + *Complex Instruction Set*: A large set of varied instructions, some of which execute multi-step tasks, reducing the overall instruction count.
  + *Variable-Length Instructions*: Instructions vary in length, resulting in more complex decoding but fewer instructions overall.
  + *Memory Access*: Supports memory-to-register operations, which allows direct memory access within instructions.
* RISC (e.g., ARM):
  + *Reduced Instruction Set*: Uses a smaller set of simpler instructions, resulting in higher instruction counts but efficient decoding and execution.
  + *Fixed-Length Instructions*: Simplifies pipelining, enabling faster execution and increased instruction throughput.
  + *Load/Store Architecture*: Memory access is restricted to explicit load and store instructions, making the design simpler and energy-efficient.

Impact on Operating Systems:

* *CISC OS Design*: CISC-based systems require fewer instructions per task but may involve complex decoding steps, which OS developers manage to optimize overall performance.
* *RISC OS Design*: RISC-based systems benefit from a straightforward pipeline and simpler design, which generally leads to higher throughput and modular OS designs.

**3. Comparing CISC vs. RISC Architectures and John von Neumann vs. Harvard Architectures**

* Von Neumann Architecture: Uses a unified memory for data and instructions, making it simpler and cost-effective but can lead to bottlenecks due to bus contention.
* Harvard Architecture: Separates memory for data and instructions, which reduces contention and allows parallel access, making it efficient for embedded and real-time processing.

Impact on Processor Performance: The Harvard architecture typically offers faster data handling and is ideal for specialized tasks, while the Von Neumann model is more widely used in general-purpose systems due to its simplicity.

Microcontrollers (e.g., Arduino Uno R3)

AVR vs. ARM Cortex-M Architecture

* AVR Architecture: Used in the Arduino Uno R3, an 8-bit RISC architecture known for low power consumption and a simple instruction set, ideal for basic applications.
* ARM Cortex-M Architecture: A 32-bit RISC architecture used in more advanced microcontrollers, which provides more processing power, larger memory access, and additional features suitable for complex applications.

8-bit vs. 16/32-bit Microcontrollers

* Memory Addressing: 8-bit microcontrollers have smaller address space, limiting memory access compared to 16/32-bit microcontrollers.
* Processing Power: Higher-bit microcontrollers, like 16- or 32-bit, offer more processing power and are suited for more demanding applications.
* Application Suitability: 8-bit microcontrollers are best for light-duty tasks, while 16- and 32-bit microcontrollers excel in more complex environments.

Memory Segmentation

Memory Segmentation in x86 Architecture

Memory segmentation divides the address space, facilitating backward compatibility with 16-bit applications, though it adds complexity compared to the flat memory models in modern 64-bit systems.

User Space vs. Kernel Space

* User Space: Where applications run, isolated to prevent direct access to system-critical code.
* Kernel Space: Reserved for the OS kernel to manage critical system processes and protect system stability and security.

Endianness

* Big-Endian: Stores the most significant byte first.
* Little-Endian: Stores the least significant byte first.

These differences are essential for data interpretation and correct system operation.

Related Wikipedia Articles:

* [Instruction-level parallelism](https://en.wikipedia.org/wiki/Instruction-level_parallelism)
* [RISC](https://en.wikipedia.org/wiki/Reduced_instruction_set_computing)
* [CISC](https://en.wikipedia.org/wiki/Complex_instruction_set_computing)
* [Von Neumann architecture](https://en.wikipedia.org/wiki/Von_Neumann_architecture)
* [Harvard architecture](https://en.wikipedia.org/wiki/Harvard_architecture)
* [Endianness](https://en.wikipedia.org/wiki/Endianness)

***Assignment No.:5:***

***Communication Protocols***

**UART (Universal Asynchronous Receiver-Transmitter)**

* *Simple, two-way communication* with only two wires (TX, RX), typically operating at low speeds (9600-115200 baud rate).
* Used in basic setups like connecting Arduinos to computers.

**I²C (Inter-Integrated Circuit)**

* A *two-wire protocol* (SDA and SCL) that can connect multiple devices (up to 127) and operates at medium speeds (100kHz to 400kHz).
* Commonly used for multiple sensors and chips on one board.

**SPI (Serial Peripheral Interface)**

* A fast protocol with multiple wires (MOSI, MISO, CLK, SS), ideal for high-speed data transfers between devices like SD cards or displays.

**CAN (Controller Area Network)**

* Uses *differential signaling* with two wires (CAN-H, CAN-L) to ensure reliable communication in noisy environments.
* Commonly found in automotive and industrial applications.

**USB (Universal Serial Bus)**

* A versatile, high-speed protocol supporting both data transfer and power supply, widely used in computers, phones, and other devices.

Related Wikipedia Articles:

* [UART](https://en.wikipedia.org/wiki/Universal_asynchronous_receiver-transmitter)
* [I²C](https://en.wikipedia.org/wiki/I%C2%B2C)
* [SPI](https://en.wikipedia.org/wiki/Serial_Peripheral_Interface)
* [CAN bus](https://en.wikipedia.org/wiki/CAN_bus)
* [USB](https://en.wikipedia.org/wiki/USB)