Industrial Training Report

Place of training: Cranes Varsity, Bengaluru, Karnataka

Period of training: 25th July to 20th January 2024

Submitted to

Department of Electrical Engineering

Summer Training In-charge at TINJRIT: **Mr. Rajkumar Soni**

By

Puneet Jain : 20ETCEE006

(Batch 2020-2024)

Branch: Electrical Engineering

Techno India NJR Institute of Technology

Plot-T, Bhamashah (RIICO) Industrial Area,

Kaladwas, Udaipur-313001, Rajasthan

Contents

1. Certificate

Cranes Varsity Private Limited

Correspondence Address:
#82, Presidency Building, 3rd & 4th Floor,
St. Marks Road, Bengaluru - 560 001. Karnataka

Ph: +91 80 6764 4800/4848
Fax: +91 80 6764 4888
Email: training@cranessoftware.com

19-12-2023

TO WHOMSOEVER IT MAY CONCERN

This is to certify that **Mr Puneet Jain** was registered for **Placement Oriented Program in IoT and Embedded Systems.** His registration number is **U20230725344688.** He joined in July 2023 and completed his course in December 2023 at Cranes Varsity Pvt Ltd. During his tenure, we found him to be a very interactive and dedicated student. We wish him all the very best of luck in his future endeavors.

With Best Regards
For Cranes Varsity Pvt Ltd

[illegible]

Cranes Varsity Private Limited Bengaluru

Mr. Harsha G H

Manager

Training Operations

This is to certify that Puneet Jain, Bachelor of Electrical Engineering has completed Industrial Training in Embedded and IOT module from CRANES VARSITY as partial fulfillment of Bachelor of Engineering EE. The Industrial Training Report, Presentation, and Project are genuine work done by him and the same is being submitted for evaluation.

Signature

Mr. Rajkumar Soni

HOD EE

2. ACKNOWLEDGMENT's

I take this opportunity to express my profound gratitude and deep regards to my guide **Mr. Rajkumar Sharma** (Head of EE) for his exemplary guidance, monitoring, and constant encouragement throughout the training. The blessing, help, and guidance given by him occasionally shall carry me a long way in the journey of life on which we are about to embark.

I especially take the opportunity to thank our coordinator, Mr. Yogendra Singh Solanki, for his valuable information and guidance which helped me in completing this task through various stages. I also take this opportunity to express a deep gratitude to all my teachers of the Electrical Engineering Department for their coordinated support.

I am obliged to the staff members of the CRANES VARSITY for their valuable information provided by them in their respective fields. I am grateful for the cooperation provided by them during my training period.

I am thankful to the almighty and our parents for their moral support and to my friends with whom I shared my day-to-day experience and received lots of suggestions that improved my quality of work.

3. INTRODUCTION TO THE INSTITUTION: Cranes Varsity, a division of Cranes Software International Ltd, has been a pioneering force in empowering professionals through the seamless integration of technology and education since its inception in 1998. Positioned as a leading EdTech platform, we specialize in delivering impactful, hands-on training to a diverse audience, including graduates, universities, working professionals, and corporate and defense sectors. Our overarching goal is to bridge the gap between technology, academia, and industry.

With over 25 years of dedicated service, Cranes Varsity has transitioned from a trailblazing Technical Training institute to a dynamic EdTech Platform, offering state-of-the-art technology education services. We take pride in fostering trusted partnerships with more than 5000 esteemed academia, corporate, and defense organizations. This collaborative approach has enabled Cranes Varsity to successfully train over 1 Lakh engineers and facilitate the placement of 70,000+ professionals.

At the heart of our success lies the guiding principle, "We Assist Until We Place," which has served as the cornerstone of our growth and unwavering commitment to excellence over the past two decades. This ethos drives our continuous efforts to provide unparalleled support to our learners and uphold our position as a leader in the ever-evolving landscape of technology education.

4. General information about Embedded and IoT

Embedded systems and the Internet of Things (IoT) are closely related concepts in the field of technology, each playing a significant role in shaping the modern digital landscape. Here's a general overview of both:

Embedded Systems:

1. Definition:

- An embedded system is a dedicated computing device designed to perform specific functions or tasks within a larger system.

2. Characteristics:

- Real-Time Operation: Many embedded systems are designed to respond to stimuli in real-time.

- Single-Purpose: Typically, they are tailored for specific applications and functions.

- Low Power Consumption: Often, embedded systems are constrained by power requirements.

- Reliability: They need to be highly reliable, as they perform critical tasks.

3. Applications:

- Found in a wide range of devices: consumer electronics, automotive systems, medical equipment, industrial machines, etc.

- Examples include microcontrollers in washing machines, automotive engine control units, and industrial automation systems.

4. Development Tools:

- Specific to the microcontroller or processor used.

- Common programming languages include C and assembly.

Internet of Things (IoT):

1. Definition:

- The IoT refers to the network of interconnected devices embedded with sensors, software, and network connectivity, allowing them to collect and exchange data.

2. Characteristics:

- Connectivity: Devices in the IoT are interconnected, enabling communication and data sharing.

- Data Collection: Sensors gather data from the environment or the device itself.

- Remote Monitoring and Control: Allows users to monitor and control devices remotely.

- Scalability: Can range from small-scale deployments to large-scale, global networks.

3. Applications:

- Smart homes, healthcare, agriculture, industrial automation, smart cities, etc.

- Examples include smart thermostats, wearable fitness trackers, and industrial sensors.

4. Technologies:

- Communication Protocols: MQTT, CoAP, HTTP, etc.

- Wireless Technologies: Wi-Fi, Bluetooth, Zigbee, LoRa, etc.

Intersection of Embedded Systems and IoT:

1. Embedded Systems in IoT:

- Embedded systems form the core of many IoT devices, providing the necessary computing power and control.

- They manage sensors, actuators, and communication modules in IoT devices.

2. Challenges:

- Security: Both embedded systems and IoT face challenges related to securing devices and data.

- Interoperability: Ensuring seamless communication among devices from different manufacturers.

3. Development Tools:

- Tools used for embedded systems development are also essential for IoT device development.

In summary, embedded systems are the building blocks, providing the necessary computing capabilities, while IoT extends the capabilities by connecting these embedded systems into a network for data exchange and remote control. The synergy between embedded systems and IoT continues to drive innovation across various industries.

5. Overview of the Embedded and IOT Modules

5.1 Python Programming: Python programming is a versatile and widely used programming language known for its readability and simplicity. It is used in various domains, including web development, data science, artificial intelligence, machine learning, automation, and more. Learning Python can be a rewarding experience, and the time it takes to complete depends on various factors, such as your prior programming experience, the depth of knowledge you want to acquire, and the learning resources you use.

Fig.1 Python

Here is a general overview of Python programming and factors that may influence the time it takes to learn:

Python Basics:

1. Syntax and Basics:

   - Variables, Data Types, and Operators

   - Control Flow (if statements, loops)

   - Functions and Modules

2. Data Structures:

   - Lists, Tuples, Sets, Dictionaries

3. Object-Oriented Programming (OOP):

   - Classes and Objects

   - Inheritance, Encapsulation, Polymorphism

Intermediate Python:

1. File Handling:

   - Reading and Writing Files

2. Error Handling:

   - Exception Handling

3. Advanced Data Structures:

   - Advanced usage of Lists, Sets, Dictionaries

4. Functional Programming:

   - Lambda Functions, Map, Filter, Reduce

Advanced Python:

1. Modules and Libraries:

   - Popular libraries (e.g., NumPy, Pandas, Matplotlib)

   - Understanding third-party modules

2. Web Development (Optional):

   - Flask or Django for web development

3. Database Integration (Optional):

   - Connecting to and manipulating databases

4. Concurrency and Asynchronous Programming (Optional):

   - Asyncio for asynchronous programming

5. Testing and Debugging:

   - Writing unit tests

   - Debugging techniques

Time Estimates:

- Beginner with no programming experience: Several weeks to a few months.

- Intermediate knowledge in another programming language: A few weeks to a couple of months.

- Experienced programmer transitioning to Python: A few weeks.

Factors Affecting Learning Time:

1. Consistency: Regular, dedicated practice is key.

2. Learning Resources: The quality of tutorials, courses, and books can significantly impact learning speed.

3. Projects: Applying your knowledge through small projects can reinforce learning.

4. Prior Experience: Your background in programming can affect how quickly you grasp concepts.

5. Practice and Application: Hands-on coding and real-world projects enhance understanding.

5.2 C Programming Language: General Information

C is a general-purpose, procedural programming language created by Dennis Ritchie in the early 1970s at Bell Labs. It has become one of the most widely used programming languages due to its efficiency, versatility, and close-to-hardware functionality. Here is some general information about the C programming language:

Fig 2. C Language

Key Features:

1. Procedural Language:

- C follows a procedural programming paradigm, emphasizing functions and structured programming.

2. Low-Level Access:

- C provides low-level access to memory, making it suitable for system programming and developing applications where performance is critical.

3. Portability:

- C programs are highly portable and can be executed on different platforms with minimal modification.

4. Efficiency:

- C allows for direct manipulation of hardware, making it efficient for tasks that require low-level memory access.

5. Structured Language:

- C supports structured programming with features like functions, loops, and conditionals.

6. Extensibility:

- C can be easily extended through the use of libraries, and it serves as the foundation for many other programming languages.

7. Rich Standard Library:

- C comes with a standard library that provides functions for common operations, enhancing code reusability.

Basic Syntax and Structure:

- Hello World Program:

```c
#include <stdio.h>


int main() {
    printf("Hello, World!\n");
    return 0;
}
```

- Key Concepts:
  - Variables and Data Types
  - Functions and Control Flow (if statements, loops)
  - Arrays and Pointers
  - Structures and Unions
  - File Handling (I/O operations)

Memory Management:

- Pointers:

  - C supports pointers, allowing direct manipulation of memory addresses.

- Dynamic Memory Allocation:

  - Functions like `malloc()` and `free()` enable dynamic memory allocation.

Standard Libraries:

- stdio.h:

  - Input/output functions like `printf` and `scanf`.

- stdlib.h:

  - Functions for memory allocation (`malloc`, `free`) and other utility functions.

- string.h:

  - String manipulation functions.

- math.h:

  - Mathematical functions.

Applications:

1. System Software:

   - Operating systems, device drivers, and embedded systems.

2. Application Software:

   - Text editors, compilers, databases, and graphics software.

3. Game Development:

   - Many game engines and graphics libraries are written in C.

4. Network Programming:

   - C is widely used for network programming due to its low-level capabilities.

5. IoT (Internet of Things):

   - C is used in embedded systems and IoT devices.

Development Environments:

- IDEs:

  - Common Integrated Development Environments include Code::Blocks, Dev-C++, and Eclipse with CDT.

- Compilers:

  - Popular compilers include GCC (GNU Compiler Collection), Microsoft Visual C++, and Clang.

Learning Resources:

- Books:

  - "The C Programming Language" by Brian Kernighan and Dennis Ritchie (K&R) is a classic.

  - "C Programming Absolute Beginner's Guide" by Perry and Miller.

- Online Courses:

  - Platforms like Coursera, edX, and Udemy offer C programming courses.

C remains a foundational language in computer science and is still widely used in various domains. Learning C provides a solid understanding of programming fundamentals and is an excellent starting point for those exploring software development.

5.3 C++ or CPP language: C++ is a powerful and versatile programming language developed by Bjarne Stroustrup at Bell Labs in the early 1980s. It is an extension of the C programming language, incorporating both procedural and object-oriented programming paradigms. Known for its efficiency, performance, and flexibility, C++

is widely used in various domains, including system programming, game development, embedded systems, and application software.

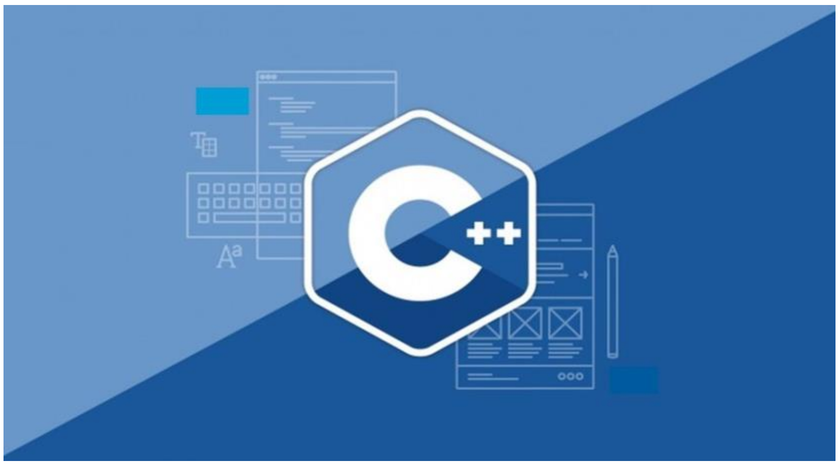

Fig 3. C++ language

One of the distinctive features of C++ is its strong support for object-oriented programming (OOP). It provides features such as classes, objects, inheritance, polymorphism, and encapsulation, allowing developers to create modular and reusable code. C++ also supports procedural programming, making it suitable for a wide range of applications.

The language includes the Standard Template Library (STL), a powerful set of generic algorithms and data structures. The STL simplifies complex programming tasks with pre-built components like vectors, queues, and sorting algorithms. This contributes to code reusability and enhances productivity.

Memory management in C++ is explicit, allowing developers to allocate and deallocate memory manually using operators like `new` and `delete`. While this provides fine-grained control, it also necessitates careful management to avoid memory leaks and errors.

C++ is a multi-paradigm language, accommodating procedural, object-oriented, and generic programming styles. This flexibility enables developers to choose the most appropriate paradigm for a given task, enhancing code organization and maintainability.
Portability is another key aspect of C++.

Programs written in C++ can be compiled and executed on different platforms with minimal adjustments, contributing to its widespread use and longevity.
The language prioritizes performance, making it suitable for applications demanding computational efficiency, such as game development and system-level programming.
Development environments for C++ include popular Integrated Development Environments (IDEs) like Visual Studio, Code::Blocks, and Eclipse. Additionally, several compilers, including GCC, Microsoft Visual C++, and Clang, support C++.

Learning resources for C++ range from authoritative books like "Programming: Principles and Practice Using C++" by Bjarne Stroustrup to online courses on platforms like Coursera and Udemy. Aspiring programmers can benefit from the language's rich ecosystem and community support.

In summary, C++ stands as a robust and enduring programming language with a diverse range of applications. Its combination of procedural and object-oriented features, along with support for generic programming and efficient memory management, makes it a preferred choice for developers tackling diverse challenges in the software development landscape.

5.4 Basic Electronics: Basic Electronics: Understanding Analog and Digital electronics.

Basic electronics is the foundation of modern technology, encompassing the study of electrical components, circuits, and systems. It can be broadly categorized into two main domains: analog and digital electronics. Understanding these concepts is crucial for anyone delving into the field of electronics.

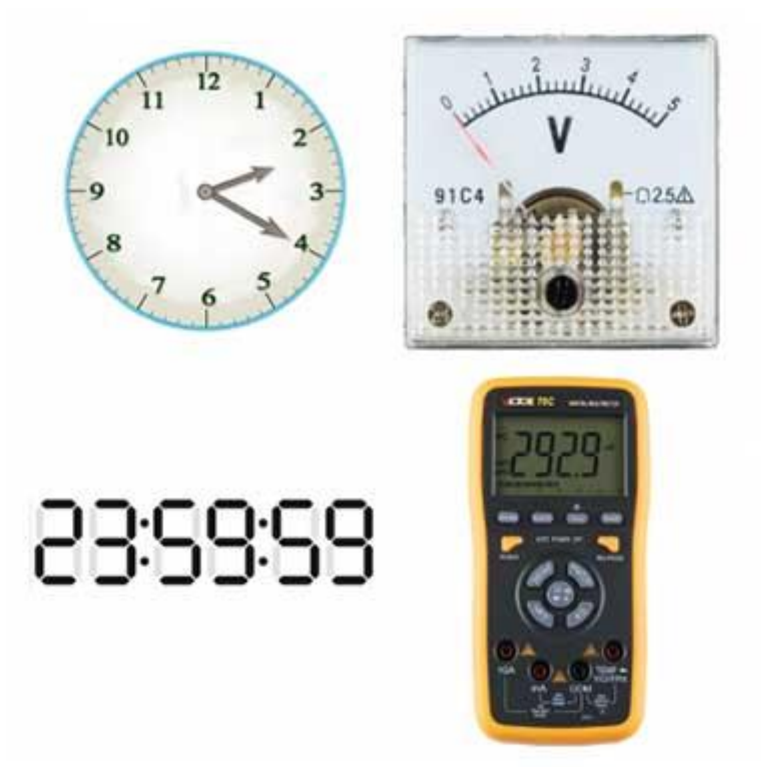

Fig 4. Analog v/s Digital electronics

5.4.1 Analog Electronics:

1. Definition:

Analog electronics deals with continuous signals, where information is represented by varying voltage or current levels. It involves the study of analog circuits that process and manipulate continuous signals.

2. Basic Components:

- Resistor: Restricts the flow of electric current.

- Capacitor: Stores and releases electrical energy.

- Inductor: Stores energy in a magnetic field.

- Diode: Allows current to flow in one direction.
- Transistor: Amplifies or switches electronic signals.

3. Operational Amplifiers (Op-Amps):

- Op-Amps are fundamental components in analog electronics used for amplification, signal conditioning, and mathematical operations.

4. Analog Signals:

- Analog signals represent information using continuously varying voltage or current levels.
- Examples include audio signals, temperature readings, and analog sensors.

5. Analog Circuits:

- Amplifiers: Increase the strength of signals.
- Filters: Allow certain frequencies to pass while blocking others.
- Oscillators: Generate periodic waveforms.

6. Applications:

- Audio systems, analog sensors, analog communication systems, and analog signal processing.

5.4.2 Digital Electronics:

1. Definition:

Digital electronics deals with discrete signals represented by binary digits (bits), using combinations of 0s and 1s. It involves the study of digital circuits that process and manipulate digital signals.

2. Basic Components:

- Logic Gates: Fundamental building blocks for digital circuits (AND, OR, NOT, XOR).
- Flip-Flops: Store binary information.
- Registers: Group of flip-flops used for data storage.
- Microcontrollers/Microprocessors: Process digital information.

3. Digital Signals:

- Digital signals represent information using discrete levels of voltage or current, typically denoted as 0 and 1.
- They offer advantages in terms of noise immunity and ease of processing.

4. Digital Circuits:

- Adders and Subtractors: Perform arithmetic operations.
- Multiplexers and Demultiplexers: Select and route digital data.

- Counters and Shift Registers: Sequential logic circuits.

5. Binary System:

- Information in digital electronics is represented using the binary system, where each digit is a binary number (0 or 1).

6. Applications:

- Computers, digital cameras, digital audio players, microcontrollers, and digital communication systems.

Comparison:

1. Signal Representation:

- Analog: Continuous signals with varying voltage or current levels.
- Digital: Discrete signals with binary representation (0s and 1s).

2. Noise Immunity:

- Analog: Prone to noise and interference.
- Digital: More immune to noise; signals can be accurately reconstructed.

3. Processing and Storage:

- Analog: Continuous processing and storage.
- Digital: Discrete processing and storage.

4. Accuracy:
   - Analog: Limited by components and noise.
   - Digital: High accuracy due to discrete levels.

5. Applications:
   - Analog: Audio systems, sensors, and communication.
   - Digital: Computing, data storage, and signal processing.

Understanding both analog and digital electronics is essential as modern electronic systems often involve a combination of these technologies. Signal processing may start with analog sensors, which are then digitized for further processing in a digital system. This synergy between analog and digital concepts forms the backbone of modern electronic devices and systems. Students and enthusiasts in electronics often begin by learning the principles of basic components, circuits, and systems, gradually progressing to more complex applications in both analog and digital domains.

6. ARM Microcontroller: ARM microcontrollers are a family of microprocessors based on the ARM architecture, designed for embedded systems and applications requiring low power consumption. ARM, which stands for Advanced RISC Machine, is a Reduced Instruction Set Computing (RISC) architecture known for its efficiency and versatility.

ARM microcontrollers find widespread use in various industries, including automotive, consumer electronics, industrial automation, and Internet of Things (IoT) devices. One of the key advantages of ARM architecture is its scalability, allowing manufacturers to produce microcontrollers with varying performance levels to meet diverse application requirements.

These microcontrollers typically feature a 32-bit or 64-bit architecture, providing a balance between performance and power efficiency. The ARM Cortex-M series is particularly popular for microcontroller applications due to its emphasis on low power consumption and cost-effectiveness.

ARM microcontrollers offer a range of peripherals and features, including GPIO (General Purpose Input/Output) pins, timers, communication interfaces (such as UART, SPI, I2C), and analog-to-digital converters. This rich set of peripherals makes ARM microcontrollers suitable for a wide range of tasks, from simple control applications to more complex systems.

Programmers often use the C programming language to develop software for ARM microcontrollers. The ARM Cortex Microcontroller Software Interface Standard (CMSIS) provides a standardized framework for software development, ensuring portability across different ARM-based microcontroller platforms.

ARM microcontrollers are known for their energy efficiency, making them well-suited for battery-powered devices and applications where power consumption is a critical factor. The architecture's popularity has led to a vast ecosystem of development tools, compilers, and libraries, further facilitating the creation of software for these microcontrollers.

In summary, ARM microcontrollers are a versatile and widely adopted choice for embedded systems, offering a balance between performance, power efficiency, and scalability. Their prevalence across various industries attests to their adaptability for diverse applications in the rapidly evolving field of embedded computing.

6.1 LPC 2129 Microcontroller: The LPC2129 microcontroller is part of the LPC2100 series developed by NXP Semiconductors, formerly Philips Semiconductors. This series is based on the ARM7TDMI-S architecture, a 32-bit Reduced Instruction Set Computing (RISC) architecture, renowned for its efficiency in embedded systems. The LPC2129 is specifically designed for applications requiring advanced control and connectivity features.

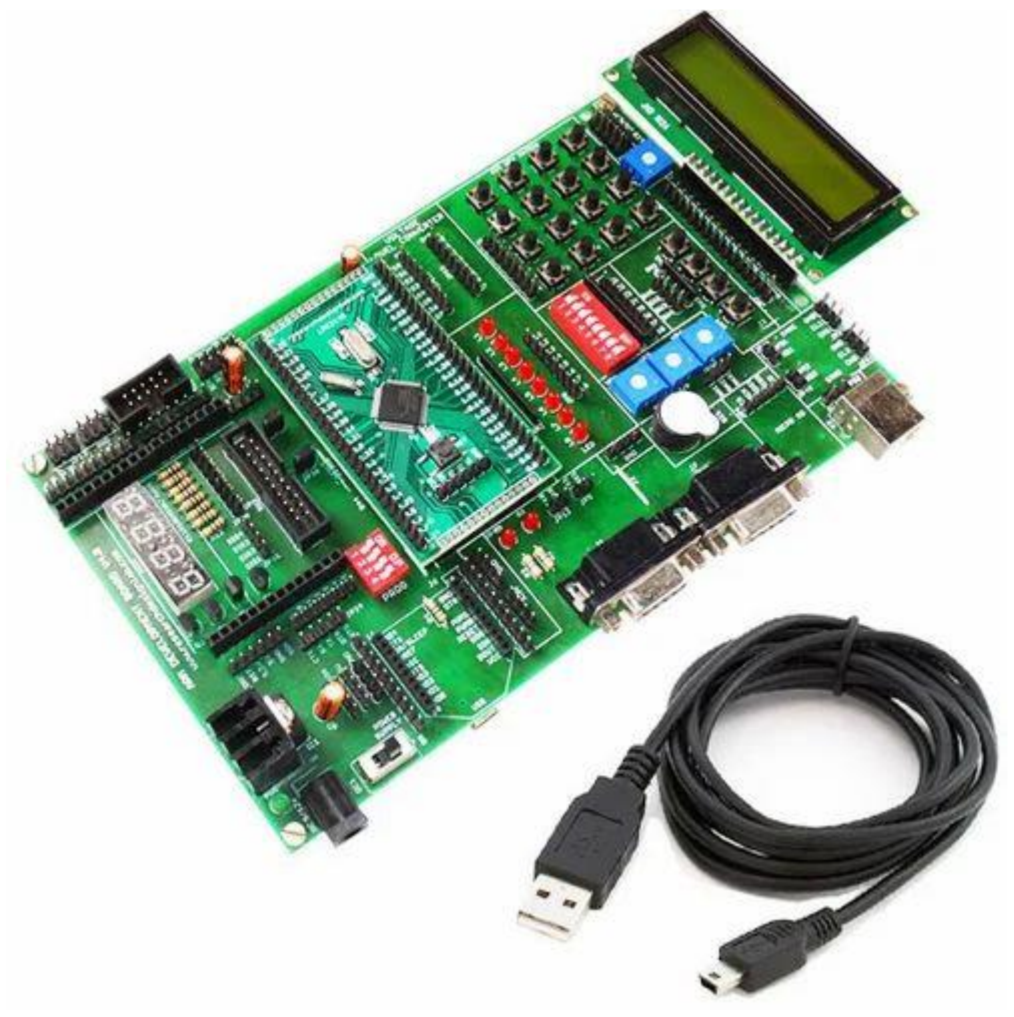

Fig.5 LPC 2129 Board

The LPC2129 features a 16/32-bit ARM7TDMI-S core running at a clock frequency of up to 60 MHz. It integrates on-chip Flash memory for program storage, ranging from 128 KB to 512 KB, and has SRAM ranging from 16 KB to 64 KB for data storage. This on-chip memory configuration reduces the need for external memory components, making it suitable for cost-effective designs.

Connectivity is a key strength of the LPC2129, with built-in peripherals including UARTs (Universal Asynchronous Receiver-Transmitters), SPI (Serial Peripheral Interface), I2C (Inter-Integrated Circuit), and CAN (Controller Area Network) interfaces. These features make it suitable for applications in communication, automotive, and industrial control systems.

The LPC2129 also includes a variety of timers/counters, GPIO (General Purpose Input/Output) pins, and analog-to-digital converters, enhancing its versatility for diverse embedded applications. Additionally, it supports real-time clock (RTC) functionality, enabling accurate timekeeping in applications requiring timestamping or scheduling.

Program development for the LPC2129 is typically done in C or assembly language using tools like Keil MDK (Microcontroller Development Kit) or other ARM development environments. The presence of a JTAG (Joint Test Action Group) interface facilitates debugging and programming of the microcontroller.

Overall, the LPC2129 microcontroller combines the efficiency of the ARM7TDMI-S architecture with advanced connectivity features, making it well-suited for applications where a balance of performance, connectivity, and cost-effectiveness is crucial. Its widespread use is evident in fields such as industrial automation, automotive control systems, and various embedded applications requiring robust processing capabilities.

7. Keil IDE:

KEIL™
Tools by ARM
µVision®5
Integrated Development Environment
KEIL is a trademark and µVision is a registered trademark of ARM Ltd. All rights reserved.
This product is protected by US and international laws.

Keil IDE (Integrated Development Environment) is a widely-used software development tool for embedded systems. Developed by Arm, it provides a comprehensive environment for writing, compiling, and debugging code for microcontrollers. Keil supports various microcontroller architectures, including ARM, 8051, and C166. With features like project management, code editing, and debugging tools, Keil simplifies the development process for embedded systems, making it a preferred choice among engineers and developers working on diverse microcontroller projects.