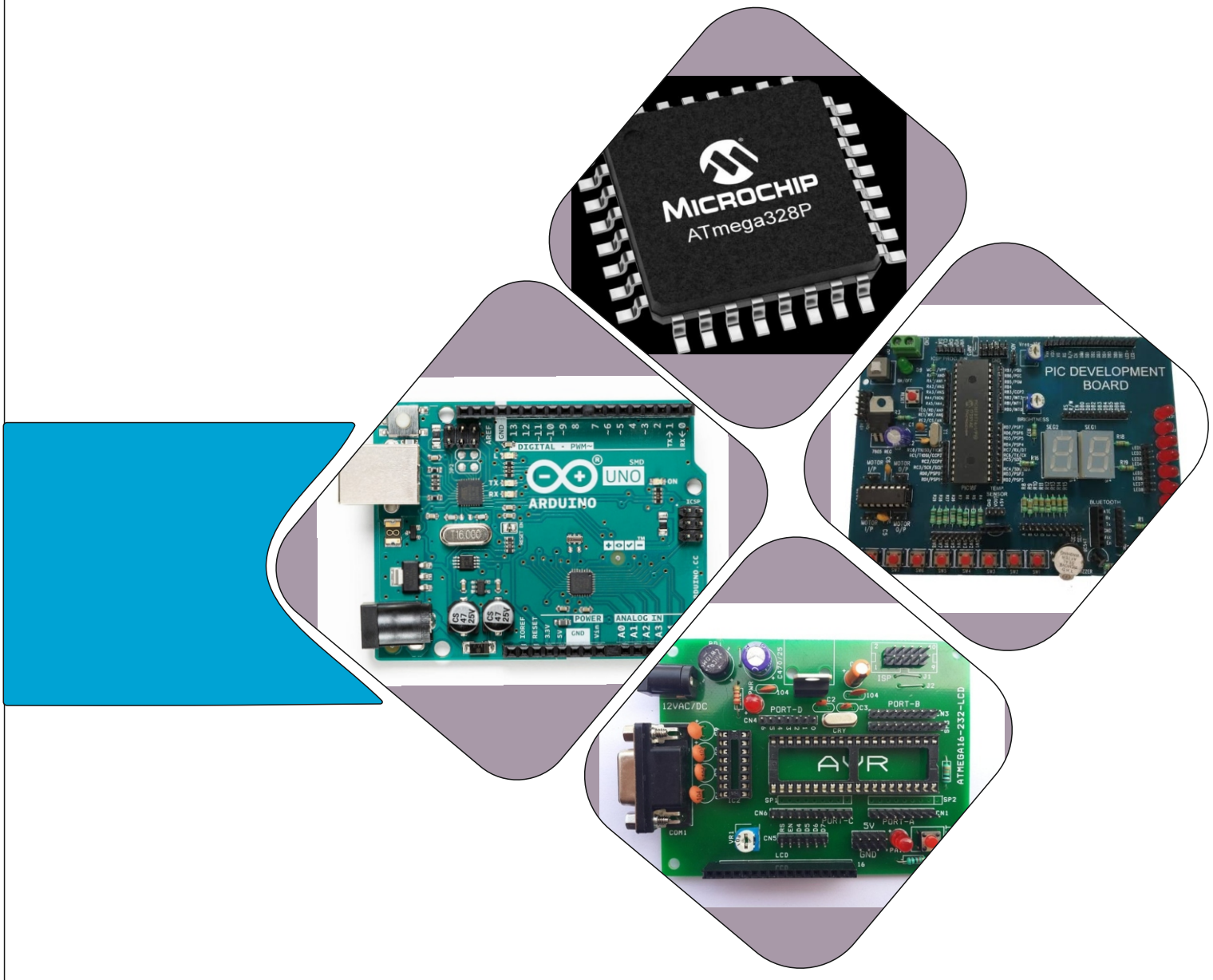
SCHEME :K

Name :_____
Roll No.:_____ Year : 20 ___ 20 __
Exam Seat No. :_____

LABORATORY MANUAL FOR

EMBEDDED SYSTEM (315338)



ELECTRONICS ENGINEERING GROUP

MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION, MUMBAI
(Autonomous)(ISO21001:2018)(ISO/IEC27001:2013)

VISION

To ensure that the Diploma level Technical Education constantly matches the latest requirements of technology and industry and includes the all-round personal development of students including social concerns and to become globally competitive, technology led organization.

MISSION

To provide high quality technical and managerial manpower, information and consultancy services to the industry and community to enable the industry and community to face the changing technological and environmental challenges.

QUALITY POLICY

We, at MSBTE, are committed to offer the best in class academic services to the students and institutes to enhance the delight of industry and society. This will be achieved through continual improvement in management practices adopted in the process of curriculum design, development, implementation, evaluation and monitoring system along with adequate faculty development programmes.

CORE VALUES

MSBTE believes in the followings:

- Education industry produces live products.
- Market requirements do not wait for curriculum changes.
- Question paper is the reflector of academic standards of educational organization
- Well-designed curriculum needs effective implementation too.
- Competency based curriculum is the backbone of need based program.
- Technical skills do need support of life skills.
- Best teachers are the national assets.
- Effective teaching learning process is impossible without learning resources.

A Laboratory Manual for

Embedded System

(315338)

Semester-V

(AO/ DE/ EJ/ EK/ ET/ EX/ IE/ TE)

Maharashtra State

Board of Technical Education, Mumbai

(Autonomous) (ISO 21001:2018)(ISO/IEC 27001:2013)

Maharashtra State

Board of Technical Education, Mumbai

(Autonomous) (ISO 21001:2018)(ISO/IEC 27001:2013)

4th Floor, Government Polytechnic Building, 49, Kherwadi, Bandra (East), Mumbai - 400051.

MAHARASHTRA STATE
BOARD OF TECHNICAL EDUCATION
Certificate

This is to certify that Mr. / Ms. …… Roll No.…… of Fifth Semester of Diploma in ..... of Institute……..…….…… (Code: ……) has attained pre-defined practical outcomes (PROs) satisfactorily in course **Embedded System (315338)** for the academic year 20…... to 20..... as prescribed in the curriculum.

Place: …… Enrollment No: …….

Date: …… Exam. Seat No: ……

Course Teacher **Head of the Department** **Principal**

Seal of the Institute

Preface

The primary focus of any engineering laboratory/field work in the technical education system is to develop the much needed industry relevant competencies and skills. With this in view, MSBTE embarked on this innovative 'K' Scheme curricula for engineering diploma programmes with outcome- based education as the focus and accordingly, a relatively large amount of time is allotted for the practical work. This displays the great importance of laboratory work, making each teacher, instructor and student realize that every minute of the laboratory time needs to be effectively utilized to develop these outcomes, rather than doing other mundane activities. Therefore, for the successful implementation of this outcome-based curriculum, every practical has been designed to serve as a ***'vehicle'*** to develop this industry identified competency in every student. The practical skills are difficult to develop through 'chalk and duster' activity in the classroom situation. Accordingly, the 'K' scheme laboratory manual development team designed the practicals to ***focus*** on the ***outcomes***, rather than the traditional age old practice of conducting practicals to 'verify the theory' (which may become a byproduct along the way).

This laboratory manual is designed to help all stakeholders, especially the students, teachers and instructors to develop in the student the predetermined outcomes. It is expected from each student that at least a day in advance, they have to thoroughly read through the concerned practical procedure that they will do the next day and understand the minimum theoretical background associated with the practical. Every practical in this manual begins by identifying the competency, industry relevant skills, course outcomes and practical outcomes which serve as a key focal point for doing the practical. The students will then become aware about the skills they will achieve through the procedure shown there and necessary precautions to be taken, which will help them to apply in solving real-world problems in their professional life.

This manual also provides guidelines to teachers and instructors to effectively facilitate student-centered lab activities through each practical exercise by arranging and managing necessary resources in order that the students follow the procedures and precautions systematically ensuring the achievement of outcomes in the students.

The field of embedded systems plays a vital role in the modern technological world, where intelligent electronic devices are increasingly embedded into everyday objects, from household appliances to advanced industrial automation. The text is designed to bridge the gap between theory and practice by offering a balanced approach that covers essential topics such as microcontrollers, programming in embedded C, interfacing techniques and real-time systems. It aims to equip students with the skills required to understand the architecture, design, and implementation of embedded solutions through practical examples, diagrams, and hands-on exercises.

Special emphasis has been placed on the development of problem-solving abilities, hardware-software integration, and the use of modern development tools, such as Arduino and other microcontroller platforms.

Programme Outcomes (POs) and Program Specific Outcomes (PSOs) to be achieved through Practical's of this Course

Following programme outcomes are expected to be achieved through the practical of the course.

PO1: Basic and Discipline specific knowledge: Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the broad based Electronics engineering problems.

PO2: Problem analysis: Identify and analyze well-defined Electronics engineering problems using codified standard methods.

PO3: Design/ development of solutions: Design solutions for well-defined technical problems and assist with the design of Electronics systems components or processes to meet specified needs.

PO4: Engineering Tools, Experimentation and Testing: Apply modern Electronics engineering tools and appropriate technique to conduct standard tests and measurements.

PO5: Engineering practices for society, sustainability and environment: Apply appropriate Electronics technology in context of society, sustainability, environment and ethical practices.

PO6: Project Management: Use Electronics engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well- defined engineering activities.

PO7: Life-long learning: Ability to analyze individual needs and engage in updating in the context of Electronics technological changes.

List of Industry Relevant Skills

The following industry relevant skills of the competency 'Develop simple applications based on embedded system' are expected to be developed in the students by undertaking the laboratory work in this practical manual.

a. To efficiently write hardware-level code for microcontrollers.
b. Able to write and debug embedded C code for real-time control and automation tasks.
c. To write basic programs for AVR microcontrollers, Arduino to control devices.
d. Hands-on skills in interfacing sensors (e.g. temperature sensors), actuators (LEDs, motors), and other components in embedded projects.
e. To understand serial communication protocols such as UART, I2C, and SPI for device-to-device communication.

Practical- Course Outcome matrix

Course Outcomes (COs)

CO1 - Select the relevant microcontrollers for various industrial applications.
CO2 - Choose appropriate family of microcontroller for different applications.
CO3 - Interpret the communication standards of embedded systems.
CO4 - Analyze the features of Real Time Operating System.
CO5 - Develop the basic applications using Arduino.

Sr. No.	Title of the Practical	CO 1	CO 2	CO 3	CO 4	CO 5
1.	*Identification of pins of AVR and PIC Microcontroller	✔	✔	-	-	-
2.	Use an IDE for ATmega 168/328 programming	-	✔	-	-	-
3.	*Write C program to perform various arithmetic operations	-	✔	-	-	-
4.	*Interface LED matrix with AVR microcontroller	-	✔	-	-	-
5.	Serial Communication using USB	-	-	✔	-	-
6.	*Installation of Arduino IDE for Windows / MacOS/Linux operating Systems	-	-	-	-	✔
7.	Building and Testing switch and LED interface using Arduino	-	-	-	-	✔
8.	*Programs to perform arithmetic operations on Arduino	-	-	-	-	✔
9.	*LCD Interfacing to Arduino board	-	-	-	-	✔
10.	Temperature sensor interfacing to Arduino board	-	-	-	-	✔

Guidelines to Teachers

1. Teacher should provide the guideline with demonstration of practical to the students with all features.
2. Teacher shall explain prior concepts to the students before starting of each practical.
3. Involve students in the performance of each experiment.
4. Teacher should ensure that the respective skills and competencies are developed in the students after the completion of the practical exercise.
5. Teachers should give opportunities to students for hands-on experience after the demonstration.
6. Teacher is expected to share the skills and competencies to be developed in the students.
7. Teacher may provide additional knowledge and skills to the students even though not covered in the manual but are expected of the students by the industry.
8. Finally give practical assignments and assess the performance of students based on tasks assigned to check whether it is as per the instructions.
9. Teacher is expected to refer complete curriculum document and follow guidelines for implementation
10. At the beginning of the practical, which is based on the simulation, teacher should make the students acquainted with any simulation software environment.

Instructions for Students

1. Listen carefully to the lecture given by the teacher about course, curriculum, learning structure, skills to be developed.
2. Organize the work in the group and make a record of all observations.
3. Do the calculations and plot the graph wherever it is required in the practical
4. Students shall develop maintenance skills as expected by industries.
5. Student shall attempt to develop related hand-on skills and gain confidence.
6. Student shall develop the habits of evolving more ideas, innovations, skills etc. those included in scope of manual
7. Student should develop the habit to submit the practical on date and time.
8. Student should prepare well while submitting a write-up of exercise.

Content Page

List of Practical's and Progressive Assessment Sheet

Sr. No.	Title of the practical	Page No.	Date of performance	Date of submission	Assessment marks (25)	Dated sign. of teacher	Remarks (if any)
1.	*Identification of pins of AVR and PIC Microcontroller	1					
2.	Use an IDE for ATmega 168/328 programming	15					
3.	*Write C program to perform various arithmetic operations	24					
4.	*Interface LED matrix with AVR microcontroller	34					
5.	Serial Communication using USB	43					
6.	*Installation of Arduino IDE for Windows / MacOS/Linux operating Systems	55					
7.	Building and Testing switch and LED interface using Arduino	64					
8.	*Programs to perform arithmetic operations on Arduino	72					
9.	*LCD Interfacing to Arduino board	86					
10.	Temperature sensor interfacing to Arduino board	96					

Practical No.1: Identification of pins of AVR and PIC Microcontroller.

I Practical Significance

Pin identification is important for accurate connection of the microcontroller in a circuit, which in turn ensures correct circuit design and effective implementation. AVR and PIC microcontrollers are widely used in embedded systems due to their low power consumption, ease of programming and comprehensive range of integrated features.

II Industry/Employer Expected Outcome

- Develop simple applications based on embedded system.

III Course Level Learning Outcome

- Select the relevant microcontrollers for various industrial applications.
- Choose appropriate family of microcontroller for different applications.

IV Laboratory Learning Outcome

- Identify pins and functions of AVR and PIC microcontroller.

V Relevant Affective Domain related outcomes

1. Demonstrate working as a leader or a team member.
2. Follow ethical practices.

VI Relevant Theoretical Background

AVR Microcontroller:

AVR was developed in 1996 by Atmel Corporation. AVR derives its name from its developers and stands for Alf-EgilBogen Vegard Wollan RISC microcontroller, also known as Advanced Virtual RISC.

The AVR microcontrollers are based on the advanced RISC architecture and consist of 32 x 8-bit general purpose working registers. Within one single clock cycle, AVR can take inputs from two general purpose registers and put them to ALU for carrying out the requested operation, and transfer back the result to an arbitrary register. The ALU can perform arithmetic as well as logical operations over the inputs from the register or between the register and a constant. Single register operations like taking a complement can also be executed in ALU.

AVR follows Harvard Architecture format in which the processor is equipped with separate memories and buses for Program and the Data information. Here while an instruction is being executed, the next instruction is pre-fetched from the program memory.

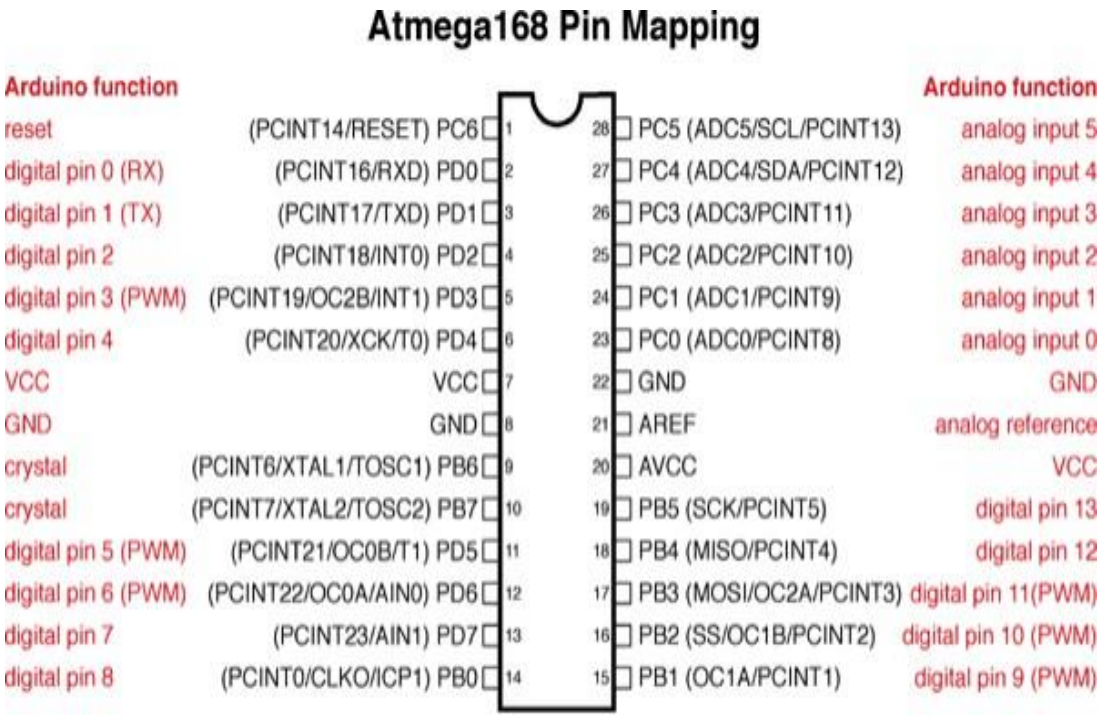


Figure 1.1 Pin-diagram of ATmega168

The ATmega168 is a widely used 8-bit microcontroller from the AVR family developed by Atmel (now part of Microchip Technology). It is widely used in embedded systems and microcontroller -based projects due to its simple architecture, reliability.

Pins and pin functions of ATmega168 (in 28-pin DIP package):

Pin	Name	Description
1	PC6 (RESET)	Reset (active low) / General-purpose I/O
2	PD0 (RXD)	UART Receive / Digital I/O
3	PD1 (TXD)	UART Transmit / Digital I/O
4	PD2	External Interrupt 0 (INT0) / Digital I/O
5	PD3	External Interrupt 1 (INT1) / Digital I/O
6	PD4	Timer/Counter2 Compare Match A (OC2A) / Digital I/O
7	VCC	Supply voltage
8	GND	Ground
9	PB6	Crystal Oscillator pin (XTAL1) / I/O
10	PB7	Crystal Oscillator pin (XTAL2) / I/O
11	PD5	Timer/Counter0 Compare Match B (OC0B) / Digital I/O
12	PD6	Timer/Counter0 Compare Match A (OC0A) / Digital I/O
13	PD7	Digital I/O
14	PB0	SPI Slave Select (SS) / Digital I/O
15	PB1	SPI MOSI / PWM (OC1A) / Digital I/O
16	PB2	SPI MISO / PWM (OC1B) / Digital I/O

continued…

Pin	Name	Description
17	PB3	SPI Clock (SCK) / Digital I/O
18	PB4	Digital I/O
19	PB5	Digital I/O
20	AVCC	Supply voltage for analog components (connect to VCC)
21	AREF	Analog Reference for ADC
22	GND	Ground
23	PC0 (ADC0)	ADC Channel 0 / Digital I/O
24	PC1 (ADC1)	ADC Channel 1 / Digital I/O
25	PC2 (ADC2)	ADC Channel 2 / Digital I/O
26	PC3 (ADC3)	ADC Channel 3 / Digital I/O
27	PC4 (ADC4)	ADC Channel 4 / SDA (I^2C data)
28	PC5 (ADC5)	ADC Channel 5 / SCL (I^2C clock)

Block diagram of ATmega168 microcontroller:

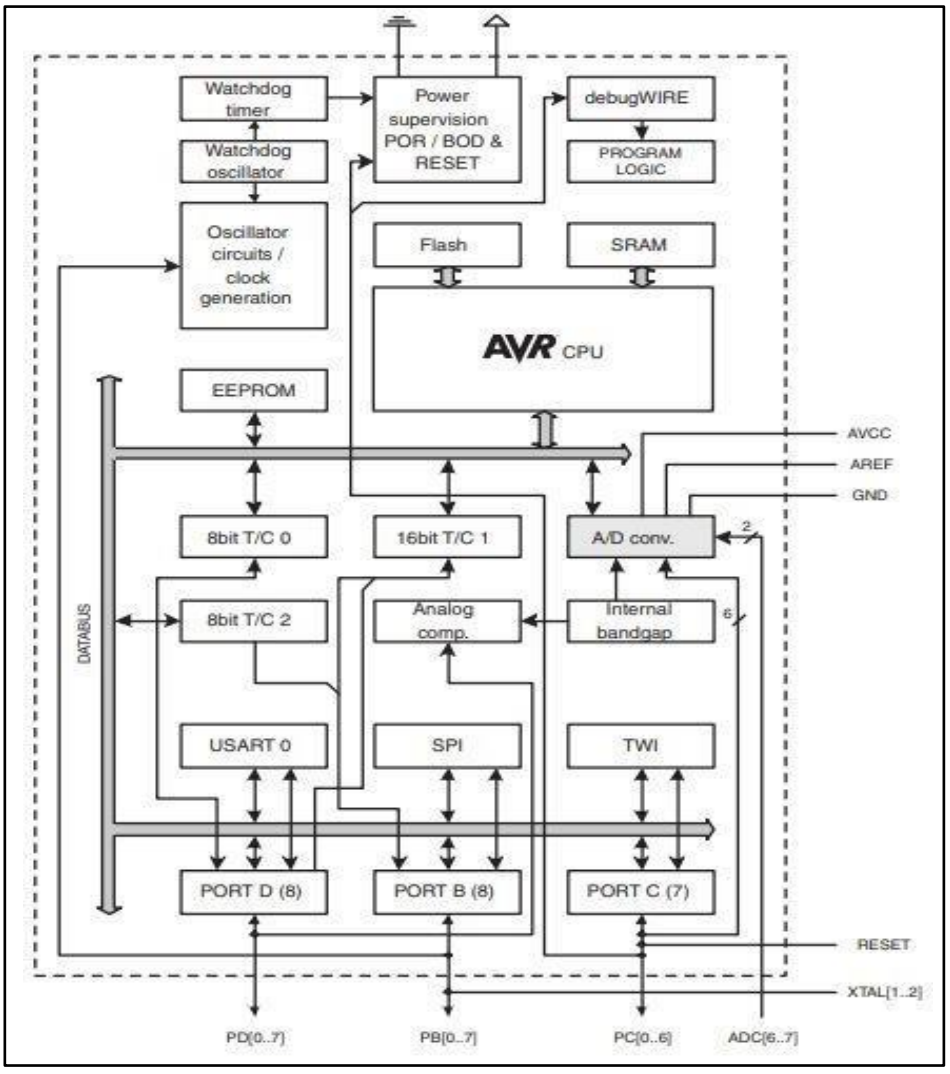


Figure 1.2 Block diagram of ATmega168 microcontroller

Features of ATmega168 microcontroller-

Feature	Description
Architecture	8-bit AVR RISC
Operating Voltage	2.7V – 5.5V
CPU Speed	Up to 20 MHz
Flash Memory	16 KB (for program storage)
SRAM	1 KB
EEPROM	512 Bytes
I/O Pins	23 programmable I/O lines
Timers	2 x 8-bit, 1 x 16-bit
ADC	10-bit, 6 channels
Communication Interfaces	UART, SPI, I2C (TWI)
PWM Channels	6
Package Types	DIP, TQFP, QFN

Pin diagram of ATmega328 Microcontroller:

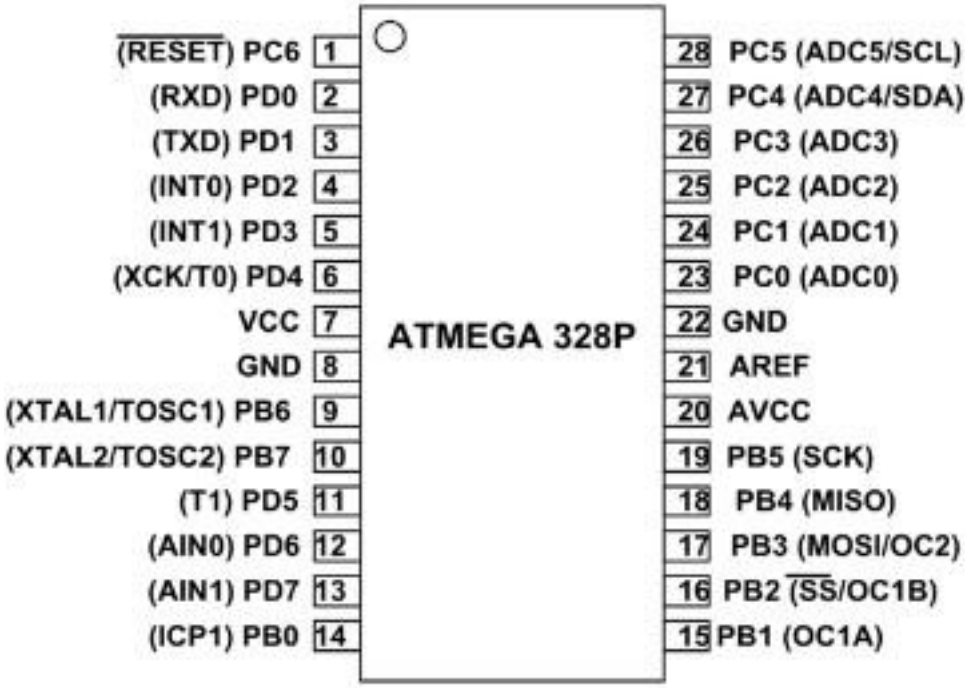


Figure 1.3 Pin-diagram of ATmega328

The **ATmega328P** is a popular 8-bit microcontroller from the AVR family, commonly used in Arduino boards such as the **Arduino Uno**. ATMEGA328P is a 28 pin chip as shown in the pin diagram above. Many pins of the chip here have more than one function. We will describe functions of each pin in the table below.

Pins and pin functions of ATmega328P microcontroller:

Pin No.	Pin Name	Description
1	PC6 (RESET)	External Reset Input
2	PD0 (RXD)	UART Receive (Serial Communication)
3	PD1 (TXD)	UART Transmit (Serial Communication)
4	PD2 (INT0)	External Interrupt 0
5	PD3 (INT1/OC2B)	External Interrupt 1 / PWM Output
6	PD4 (T0/XCK)	Timer/Counter 0 Input / Clock Output
7	VCC	Supply Voltage
8	GND	Ground
9	PB6 (XTAL1)	External Oscillator Input
10	PB7 (XTAL2)	External Oscillator Output
11	PD5 (OC0B/T1)	PWM Output / Timer Input
12	PD6 (OC0A/AIN0)	PWM Output / Analog Comparator Input
13	PD7 (AIN1)	Analog Comparator Input
14	PB0 (ICP1)	Input Capture for Timer 1
15	PB1 (OC1A)	PWM Output
16	PB2 (SS/OC1B)	SPI Slave Select / PWM Output
17	PB3 (MOSI/OC2A)	SPI Master Out / PWM Output
18	PB4 (MISO)	SPI Master In Slave Out
19	PB5 (SCK)	SPI Clock
20	AVCC	Power Supply for ADC
21	AREF	Analog Reference for ADC
28	PC0–PC5	ADC0 to ADC5 / General Purpose I/O

Multifunctional Pins: Many I/O pins serve multiple roles such as digital I/O, PWM, communication or analog input.

Features of ATmega328P microcontroller-

Feature	Specification
Architecture	8-bit AVR RISC
Flash Memory	32 KB (0.5 KB used by bootloader)
SRAM	2 KB
EEPROM	1 KB
Operating Voltage	1.8V – 5.5V
Clock Speed	Up to 20 MHz
Digital I/O Pins	23 programmable pins
PWM Channels	6 (on digital pins)
ADC Channels	6 (10-bit resolution)
Communication	UART, SPI, I2C
Timers	2 × 8-bit, 1 × 16-bit
Package Types	DIP-28, QFN-32, TQFP-32

PIC microcontroller:

PIC microcontroller was developed in the year 1993 by microchip technology. The term PIC stands for Peripheral Interface Controller. Initially this was developed for supporting PDP computers to control its peripheral devices, and therefore, named as a peripheral interface device. These microcontrollers are very fast and easy to execute a program compared with other microcontrollers. PIC Microcontroller architecture is based on Harvard architecture. PIC microcontrollers are very popular due to their ease of programming, wide availability, easy to interfacing with other peripherals, low cost, large user base and serial programming capability (reprogramming with flash memory), etc.

PIC Microcontroller Nomenclature:

Example: PIC16F877A

PIC: Personal/Peripheral Interface Controller.

16: Series of PIC microcontrollers

F: Flash Memory

877: Series No.

A: Advanced version of PIC-F16877

The **PIC16F877A** is a popular 40-pin microcontroller from Microchip's PIC16 family, widely used in embedded systems.

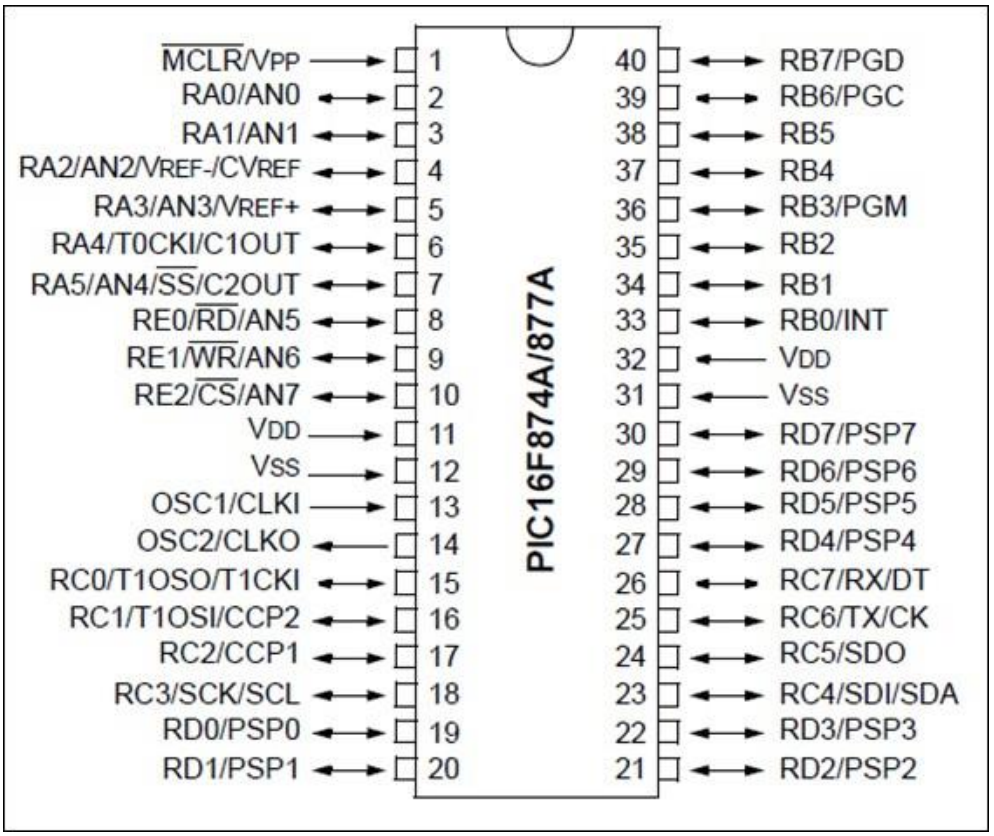


Figure 1.4 Pin-diagram of PIC16F877A microcontroller

Pins and pin functions of PIC 16F877A:

Pin No.	Pin Name	Function(s)
1	MCLR/VPP	**Master Clear (Reset)** input (active low); also used for programming voltage
2	RA0/AN0	**Port A bit 0**, Analog input 0
3	RA1/AN1	**Port A bit 1**, Analog input 1
4	RA2/AN2/VREF-	**Analog input 2**, or voltage reference minus for ADC
5	RA3/AN3/VREF+	**Analog input 3**, or voltage reference plus for ADC
6	RA4/T0CKI	**Digital I/O**, **Timer0 clock input** (T0CKI), open-drain output
7	RA5/AN4	**Digital I/O**, Analog input 4
8	RE0/AN5	**Port E bit 0**, Analog input 5
9	RE1/AN6	**Port E bit 1**, Analog input 6
10	RE2/AN7	**Port E bit 2**, Analog input 7
11	VSS	**Ground**
12	OSC1/CLKIN	**Oscillator input** / External clock input
13	OSC2/CLKO	**Oscillator output** / Clock output
14	RC0/T1OSO/T1CKI	Digital I/O / Timer1 oscillator output / Timer1 clock input
15	RC1/T1OSI	Digital I/O / Timer1 oscillator input
16	RC2/CCP1	Digital I/O / **Capture/Compare/PWM1**
17	RC3/SCK/SCL	Digital I/O / SPI clock / I²C clock
18	RD0	Digital I/O
19	RD1	Digital I/O
20	RD2	Digital I/O
21	RD3	Digital I/O
22	RD4	Digital I/O
23	RD5	Digital I/O
24	RD6	Digital I/O
25	RD7	Digital I/O
26	VSS	**Ground**
27	VDD	**+5V Supply Voltage**
28	RB0/INT	Digital I/O / **External interrupt input**
29	RB1	Digital I/O
30	RB2	Digital I/O
31	RB3/PGM	Digital I/O / Low-voltage programming
32	RB4	Digital I/O
33	RB5	Digital I/O
34	RB6	Digital I/O / **In-circuit programming (ICSP) clock**
35	RB7	Digital I/O / **ICSP data**
36	RC4/SDI/SDA	Digital I/O / SPI data in / I²C data
37	RC5/SDO	Digital I/O / SPI data out
38	RC6/TX/CK	Digital I/O / UART TX / Synchronous clock
39	RC7/RX/DT	Digital I/O / UART RX / Synchronous data
40	VDD	**+5V Supply Voltage**

Key Features of PIC16F877A:

- **Ports:** 5 Ports – A, B, C, D, and E (33 I/O pins)
- **Analog Inputs:** 8 channels (AN0–AN7)
- **Timers:** 3 timers – Timer0 (8-bit), Timer1 (16-bit), Timer2 (8-bit)
- **PWM:** 2 CCP modules (PWM1 on RC2, PWM2 on RC1)
- **USART:** Full-duplex serial communication
- **SPI/I2C:** Available on PORTC
- **Watchdog Timer:** Built-in with on-chip RC oscillator
- **EEPROM:** 256 bytes of internal EEPROM
- **Flash Memory:** 14 KB Program memory
- **RAM:** 368 bytes
- **ADC Resolution:** 10-bit

PIC16F877A Architecture:

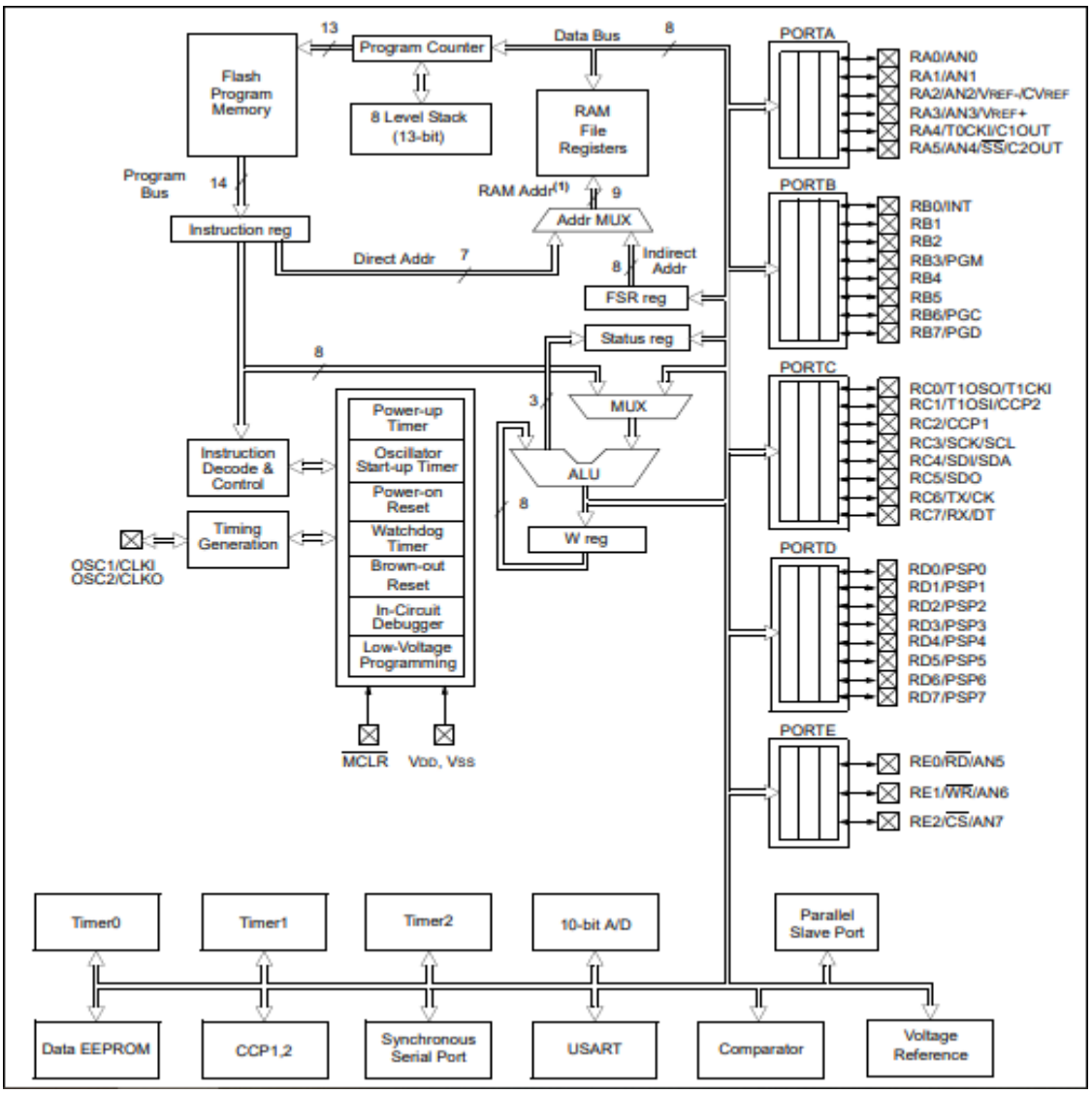


Figure 1.5 PIC16F877A Architecture

VII Actual Circuit diagram used in laboratory with related equipment rating-

This practical is related to identification of pins of AVR and PIC Microcontroller, so PIC microcontroller development board is included here.

A) Sample Setup Diagram-

AVR Microcontroller development board:

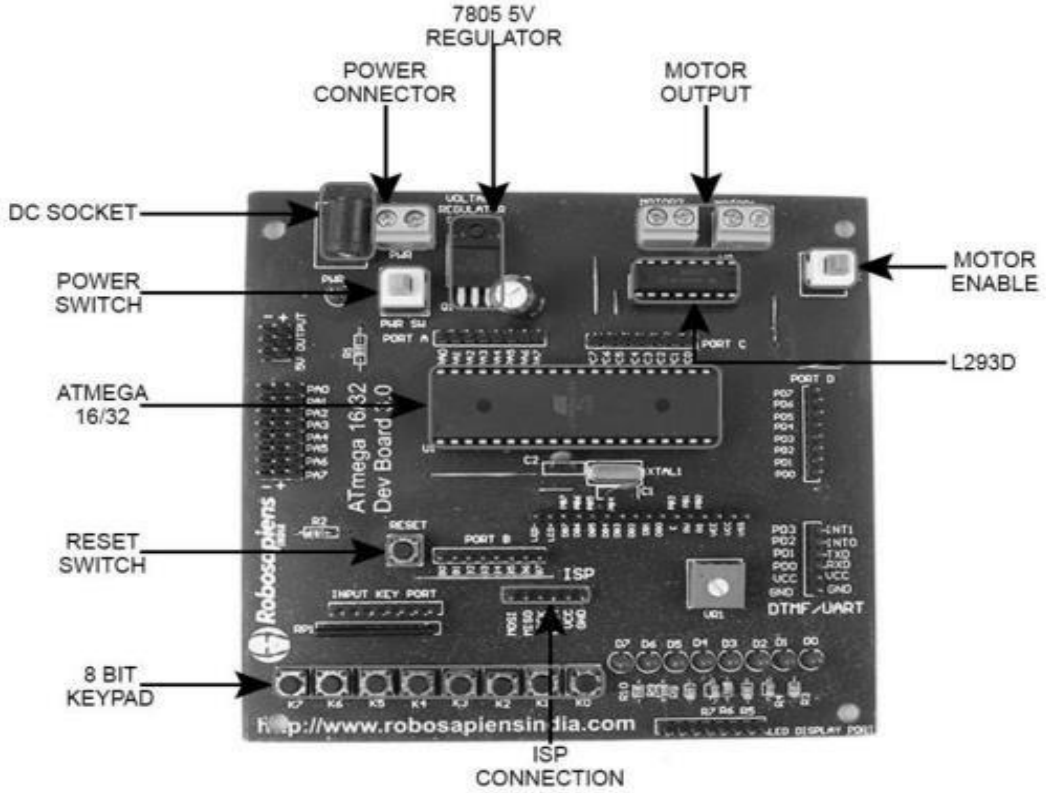


Figure 1.6 AVR Microcontroller development board

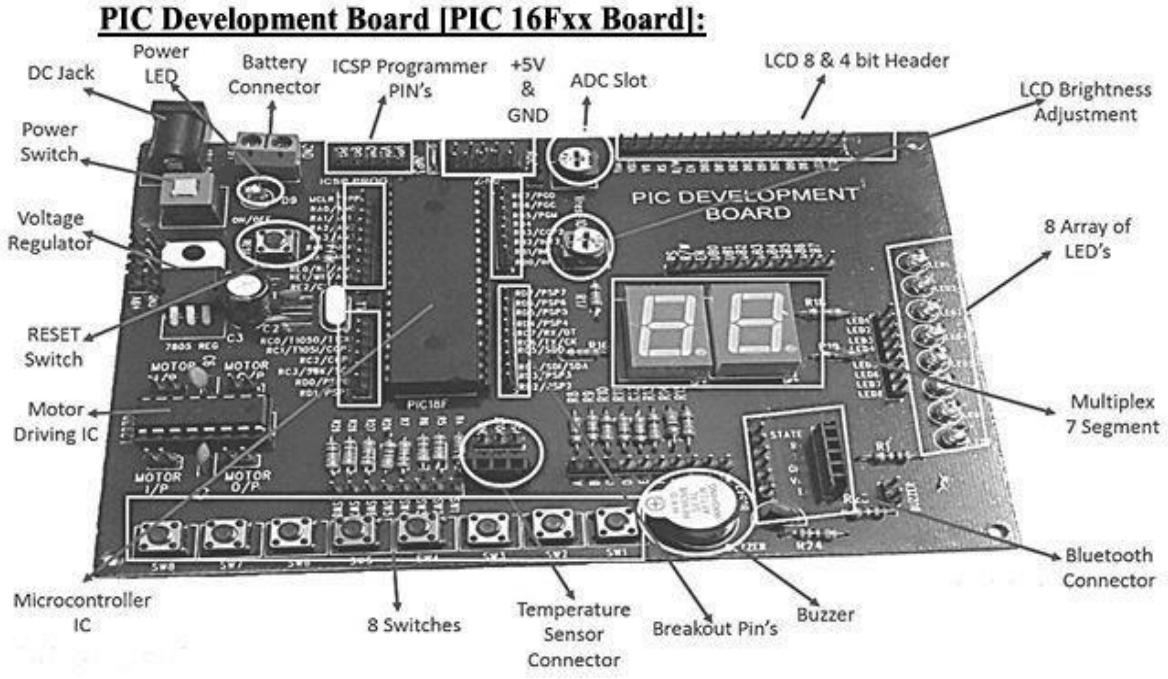


Figure 1.7 PIC Microcontroller development board

B) Actual Setup Diagram used in Laboratory -

VIII Required Resources/apparatus/equipment with specifications-

Sr. No.	Name of Resource	Specification	Quantity
1.	8051 Microcontroller kit	Single board system with 8K RAM, ROM memory with battery backup, 16X4, 16X2LCD display, PC keyboard interfacing facility, Hex keypad facility, single user cross c-compiler, RS-232, USB, interfacing facility with built in power supply.	1 No.
2.	AVR Microcontroller kit	Single Board system with IKB SRAM, 512 Bytes EPROM, SPI Interface, Onboard 8 Keypad, Onboard ISP, Onboard L293D Motor driver, on board DC power supply.	1 No.
3.	PIC Microcontroller kit	Single board system with 4X4 matrix keyboard, 16X2LCD display, two analog inputs, seven segment display, RS-232 cable, USB, interfacing facility with built in power supply, LM 35 temperature sensor, USART, SPI EEPROM.	1 No.

IX Precautions to be followed

1. Do not power up the development board when identifying pins.
2. Refer the Data Sheets for the given microcontroller.

X Procedure

1. Obtain the datasheet of **ATmega168/ATmega 328 and PIC16F877A microcontroller**
2. Observe the IC or development board.
3. View pin diagrams of specific AVR and PIC microcontrollers.
4. Understand pin functions.
5. Learn how to interface peripherals using specific pins.

XI Resources used

Sr. No.	Name of Resource	Specification	Quantity
1			
2			
3			

XII Actual Procedure (use blank sheet provided if space not sufficient)

……..……
……..……
..……
……..……
……..……
……..……
……..……
……..……

XIII Observation Table (use blank sheet provided if space not sufficient)

Observe pin out diagram and list various pins of AVR and PIC microcontroller and write their functions.

Table 1- AVR Microcontroller

Sr. No.	Pins	Function
1.	PD1 (TXD)	
2.	PB6	
3.	PD6	
4.	AREF	
5.	PC5 (ADC5)	

Table 2- PIC Microcontroller

Sr. No.	Pins	Function
1.	RA2/AN2/VREF	
2.	RB0/INT	
3.	RB6	
4.	RC4/SDI/SDA	
5.	RC6/TX/CK	

XIV Practical related questions

Note: Below given are a few sample questions for reference. Teachers must design more such questions so as to ensure the achievement of identifying CO.

1. State the pins used for I2C communication in ATmega328P?
2. Identify PORT B on the PIC16F877A and explain its use.
3. Identify the pins on ATmega328P used for SPI communication (MISO, MOSI, SCK, SS)?
4. State the pin(s) can be used as CCP (Capture/Compare/PWM) on PIC16F877A?
5. State the pins which are used for PWM output and explain their function?

[Space for Answers]

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

XV References/Suggestions for further reading

1. https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-9365-Automotive-Microcontrollers-ATmega88-ATmega168_Datasheet.pdf

2. https://www.farnell.com/datasheets/2047865.pdf

3. https://ww1.microchip.com/downloads/en/devicedoc/39582b.pdf

4. https://nptel.ac.in/courses/117104072

XVI Assessment Scheme

Performance Indicators		Weightage
Process Related : 15 Marks		**60 %**
1	Selection of ICs and power supply	25%
2	Testing of ICs	25%
3	Follow ethical practices	10%
Product Related: 10 Marks		**40%**
4	Identifying the correct function of pins	20%
5	Practical related questions	15%
6	Timely submission	05%
Total: 25 Marks		**100 %**

Marks Obtained			Dated signature of Teacher
Process Related (15)	**Product Related (10)**	**Total (25)**	

Practical No. 2: Use an IDE for ATmega 168/328 programming.

I Practical Significance

Use of an IDE for ATmega168/328 programming helps to simplify the development process by using features like syntax highlighting, code suggestions, and error detection. It has integrated compiler and programmer, which allow users to write, compile, and upload code. Built-in debugging tools helps to identify and fix errors efficiently, which improving code reliability. IDEs also provides access to libraries and sample codes, speeding up development. This practical will demonstrate the students to install and use Integrated Development Environment (IDE) tool for developing C programs of ATmega 168/328.

II Industry/Employer expected outcome

Develop simple applications based on embedded system.

III Course Level Learning Outcome(s)

- Choose appropriate family of microcontroller for different applications.

IV Laboratory Learning Outcome(s)

Use an integrated development environment tool for developing C Programs for ATmega 168/328

V Relevant Affective domain related Outcome(s)

1. Demonstrate working as a leader or a team member.
2. Follow ethical practices.

VI Relevant Theoretical Background

Microchip Studio is an Integrated Development Environment (IDE) for developing and debugging AVR® and SAM microcontroller applications. It has all the features and functionality of Atmel Studio into Microchip's well-supported portfolio of development tools to gives user a seamless and easy-to-use environment for writing, building and debugging applications written in C/C++ or assembly code. Microchip Studio can also import users Arduino® sketches as C++ projects to provide with a simple transition path from makerspace to marketplace.

VII Required Resources /apparatus/equipment with specifications

Sr. No.	Instrument /Components	Specification	Quantity
1	Desktop PC	Loaded with open-source IDE, simulation and program downloading software	1 No.

VIII Precautions to be Followed

1. Check rules / syntax of assembly programming.

IX Procedure

1. Click on the link https://www.microchip.com/en-us/tools-resources/develop/microchip-studio for installing microchip studio.
2. Click on Download Microchip Studio for installing

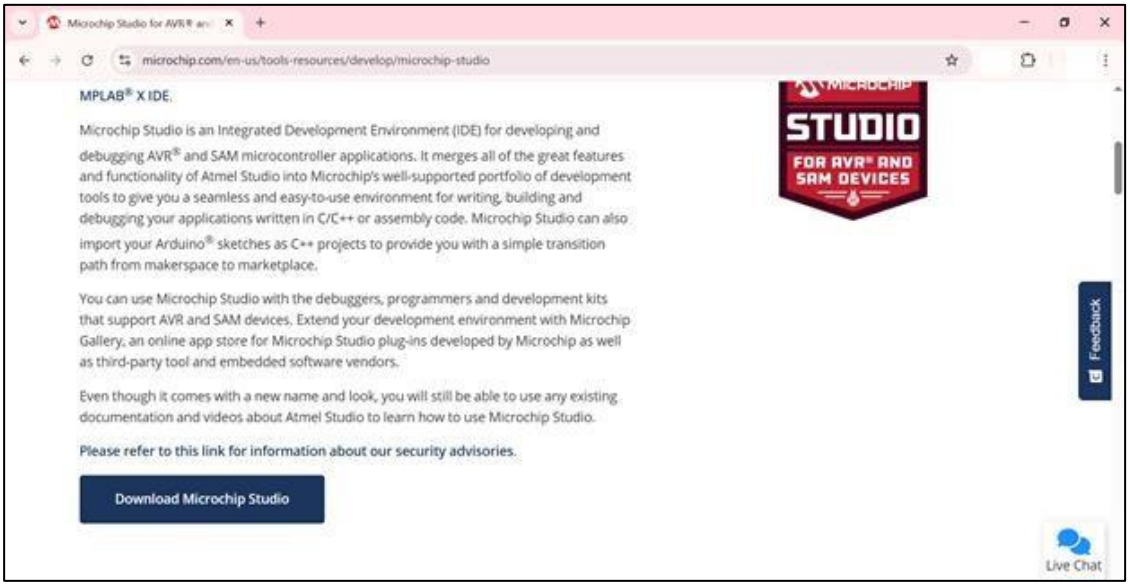


Fig 2.1: Initial Window

3. It provides two options for installation.

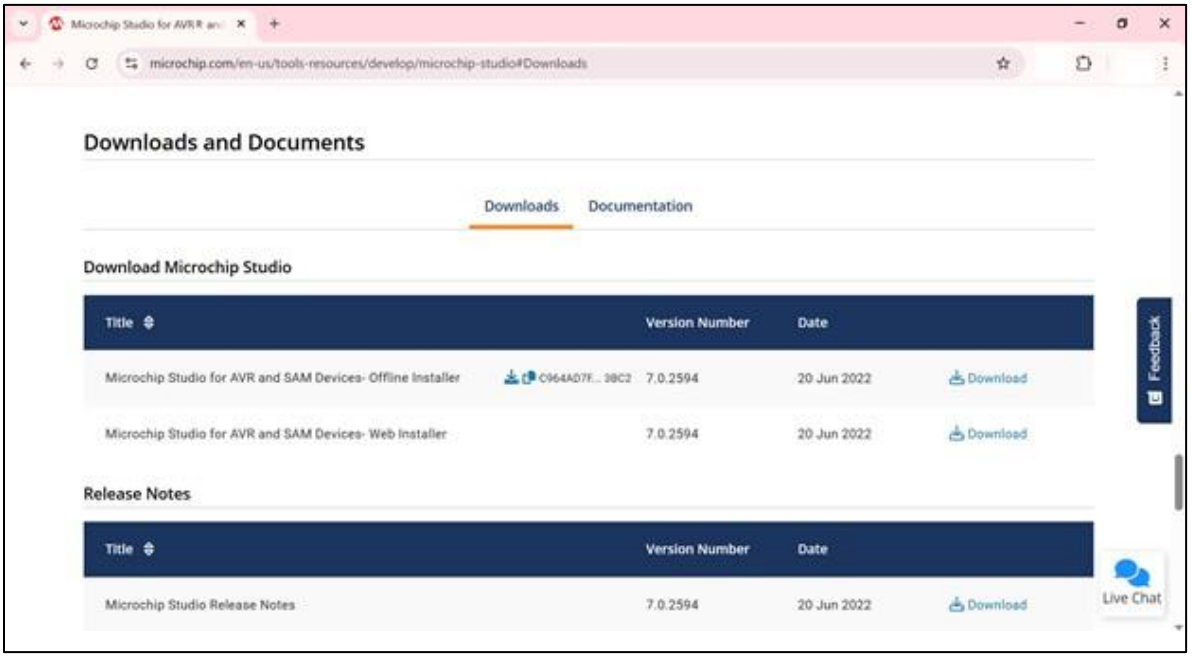


Fig 2.2: Installation options

4. Click Web installer for installion.
5. Double click the installer executable file and follow the installation wizard.

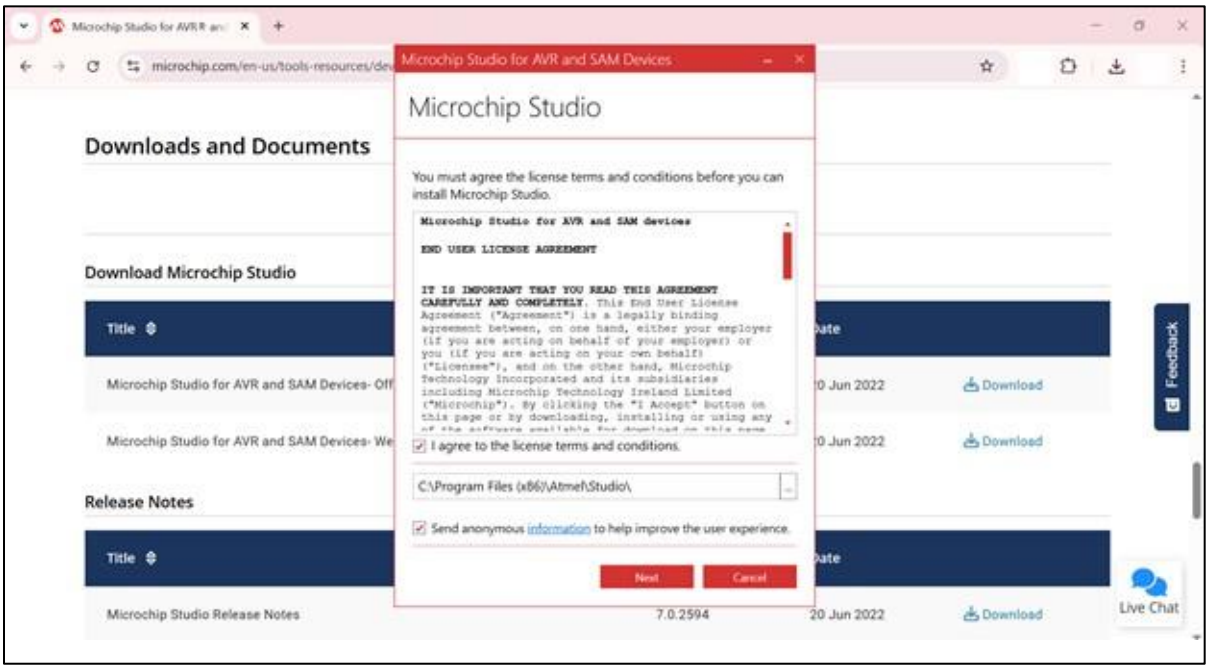


Fig 2.3: Download licence agreement

6. Select the Microchip Studio for AVR and SAM Devices and also select extension for advanced software framework and Example projects.

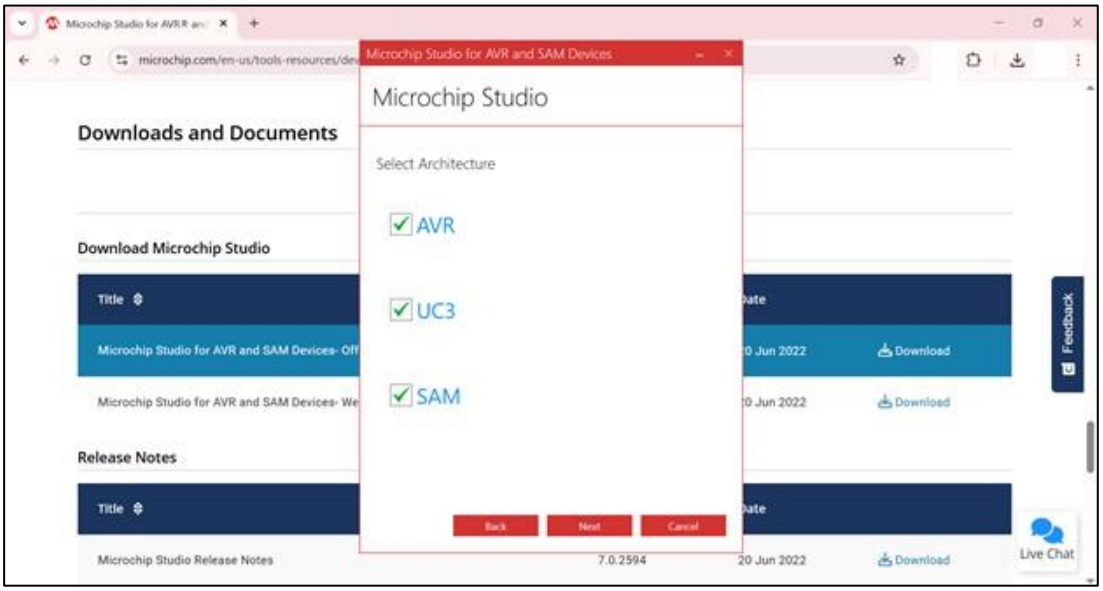


Fig 2.4: Selecting devices before installation

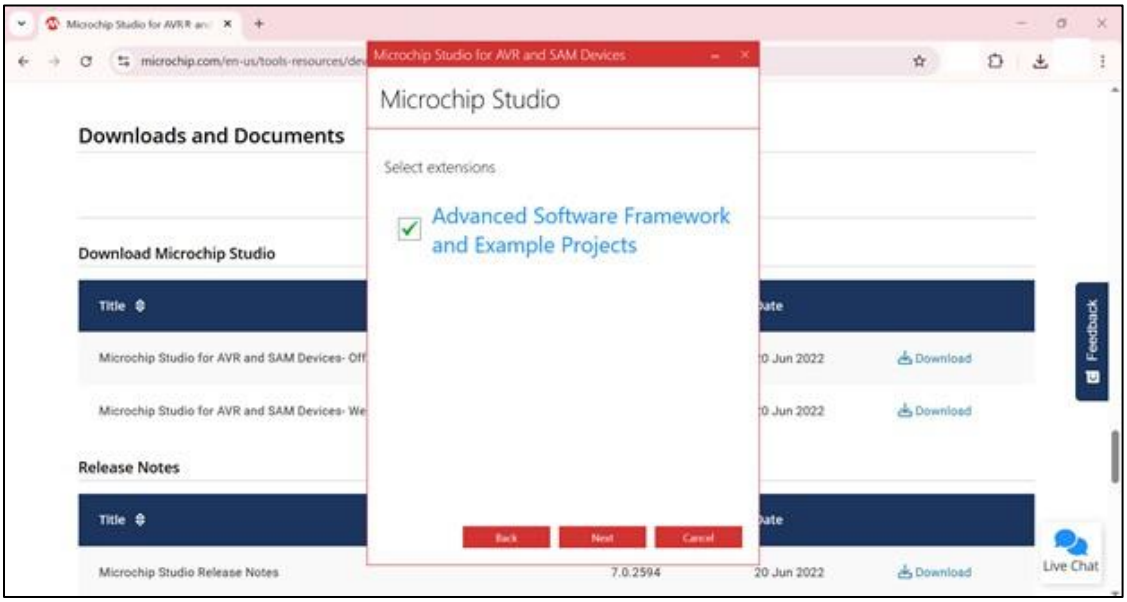


Fig 2.5: Adding Example project window

7. It will validate the system.

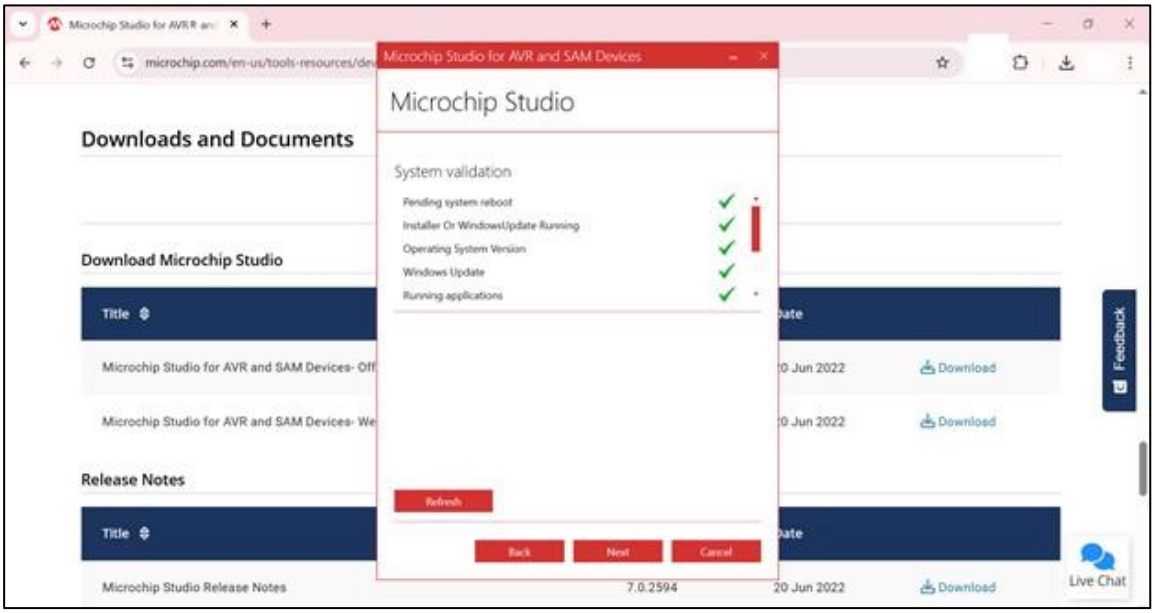


Fig 2.6: System Validation

8. It will now install the Microchip Studio.

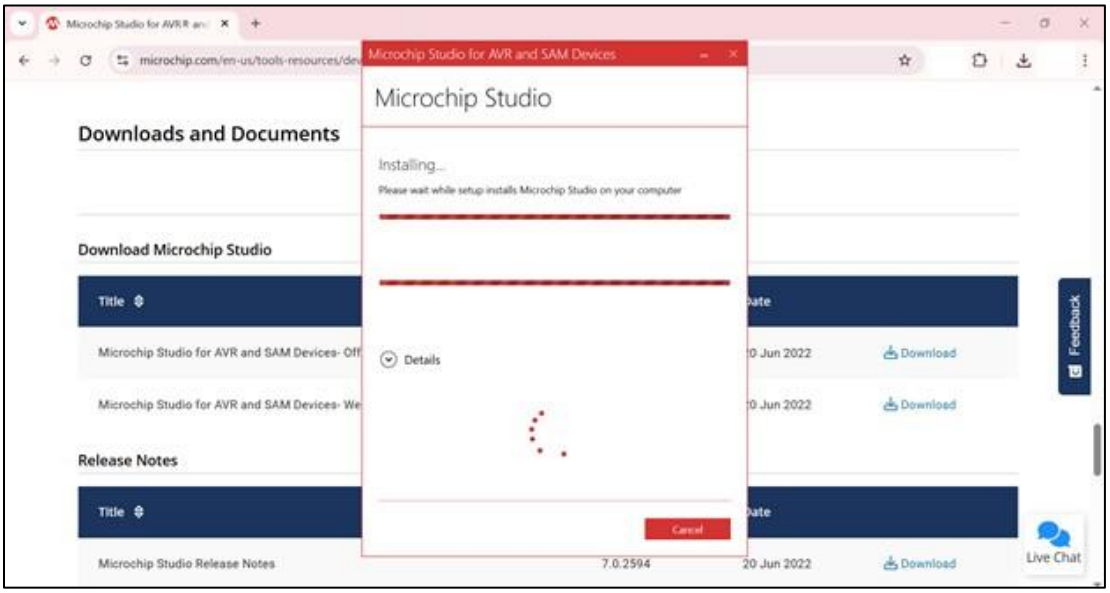


Fig 2.7: Microchip Installation

9. Once finished, the installer displays an option to Start Atmel Studio after completion.
10. **Creating a new Project**: click the New Project option.

Fig 2.8: Creating new project window

11. The project wizard appears. Select GCC C Executable Project

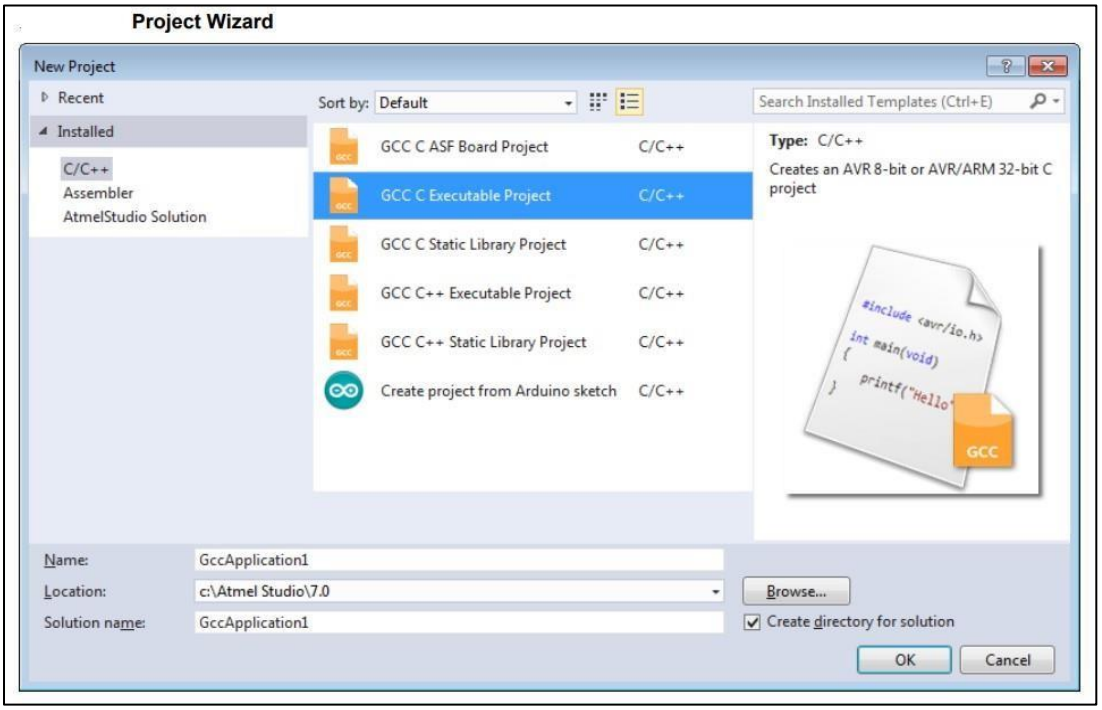


Fig 2.9: Selecting the file extension window

12. In the Name box, type a name for the new project.
13. In the Location box, select a save location.
14. When a new project is created, the Device Selection dialog is displayed and you will be prompted to select the project target device. The device selection dialog lists all supported devices for the current project type. To narrow down the selection of devices, select the device family in the Device Family field, or use the Search for Device field to view a filtered list of devices matching your search string.

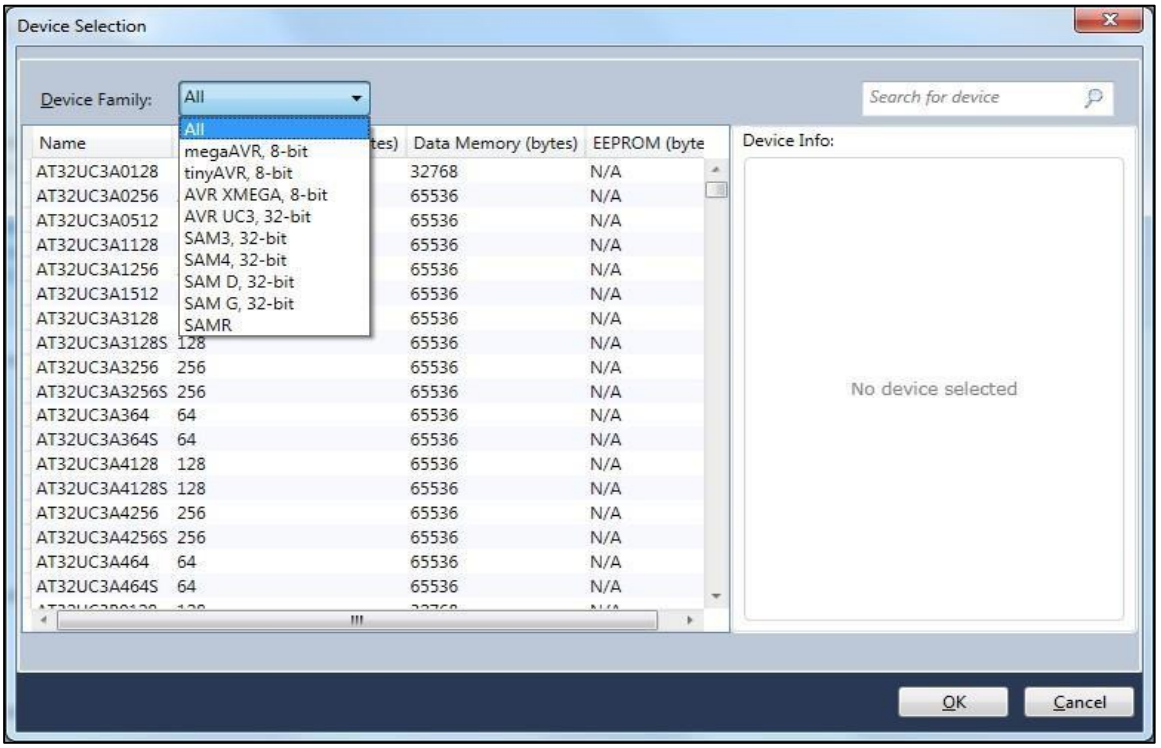


Fig 2.10: Selecting the device window

15. Select a device

16. In the Device Selection dialog, select ATxmega128 or any equivalent .
17. Click OK.
18. Writing and Compiling Code: Solution and project has been created. User can now start editing your application.

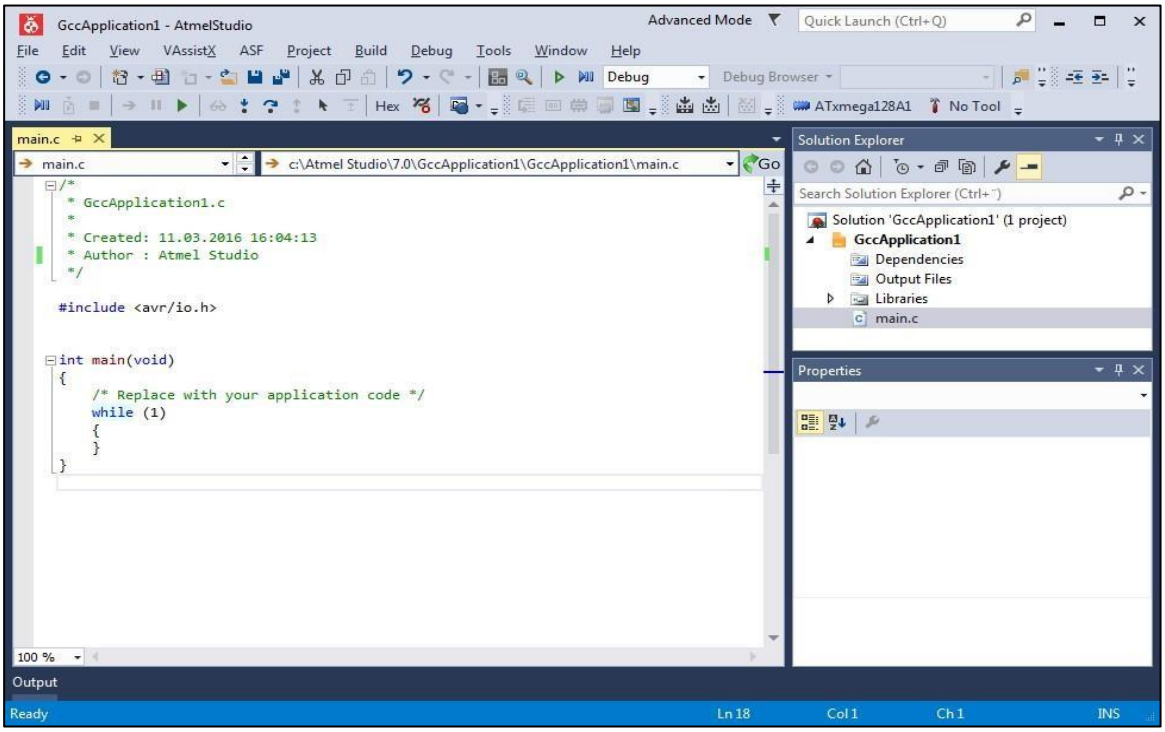


Fig 2.11: Creating and compiling code window

19. Create and build a simple application: Replace the original main function with the source code
20. To compile the project, press F7 key or select Build Solution from the Build menu.
21. Atmel Studio now builds the application. All output from the compiler is listed in the output window.
22. The programme can be further debugged using the AVR simulator.

X Resources used/apparatus/equipment with specifications:

S. No.	Instrument /Components	Specification	Quantity
1.			

XI **Actual Procedure Followed** (use blank sheet provided if space not sufficient)

……

……

……

……

……

……

……

……

……

……

……

……

……

XII Conclusions and Recommendation

……

……

……

……

XIII Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO

1. List the features of ATmega 168.
2. Compare ATmega168 and ATmega 328.
3. List the applications of AVR microcontrollers.

[Space for Answers]

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

XIV References / Suggestions for further reading

1. https://www.microchip.com/en-us/tools-resources/develop/microchip-studio#Downloads
2. https://www.microchip.com/en-us/products/microcontrollers-and-microprocessors/8-bit-mcus/avr-mcus https://ww1.microchip.com/downloads/en/DeviceDoc/ATmega48A-PA-88A-PA-168A-PA-328-P-DS- DS40002061A.pdf

XV Assessment Scheme

The given performance indicators should serve as a guideline for assessment regarding process and product related marks:

Performance indicators		Weightage
Process related: 15 Marks		**60%**
1	Identifying the given IDE	20%
2	Identifying the features of IDE	30%
3	Follow ethical practices.	10%
Product related: 10 Marks		**40%**
4	Correct Procedure followed for installation of IDE	20%
5	Answer to sample questions	15%
6	Timely Submission	05%
	Total: 25 Marks	**100 %**

Marks Obtained			Dated signature of Teacher
Process Related (15)	**Product Related (10)**	**Total (25)**	

Practical No.3: Write C program to perform various arithmetic operations.

I Practical Significance

This Practical will help the students to develop skills to write AVR based embedded C programs. This experiment provides knowledge in embedded system programming using the AVR microcontroller. By performing arithmetic operations (addition, subtraction and multiplication) on constant data and displaying the results through a microcontroller port, students gain hands-on experience with basic syntax and structure of C programs.

II Industry/Employer Expected Outcome

- Develop simple applications based on embedded system.

III Course Level Learning Outcome

- Choose appropriate family of microcontroller for different applications.

IV Laboratory Learning Outcome

- Develop AVR C program to perform addition, subtraction, and multiplication operations on two constant data and output the result to port with some delay between each output.

V Relevant Affective Domain related outcomes

1. Demonstrate working as a leader or a team member.
2. Follow ethical practices.

VI Relevant Theoretical Background

AVR programming involves writing code to control AVR microcontrollers like ATmega16, ATmega32, ATmega328P, etc., using Embedded C. AVR microcontrollers are 8-bit RISC-based microcontrollers developed by Atmel (now Microchip Technology).

Data Types:

Data Type	Size	Range	Use
char	1 byte	-128 to 127	Signed 8-bit integer
unsigned char / uint8_t	1 byte	0 to 255	8-bit data (e.g., ports, LEDs)
int	2 bytes	-32,768 to 32,767	Signed 16-bit integer
unsigned int / uint16_t	2 bytes	0 to 65,535	16-bit counters, timers
long	4 bytes	Large integers	Rarely used in 8-bit AVR due to size
float	4 bytes	Decimal numbers	Used with ADC, sensor readings
double	4 bytes (same as float)	Decimal numbers	Same as float in AVR-GCC

Data Operators:

Operator	Name	Example	Description
+	Addition	c = a + b;	Adds two operands
-	Subtraction	c = a - b;	Subtracts right operand from left operand
*	Multiplication	c = a * b;	Multiplies two operands
/	Division	c = a / b;	Divides left operand by right (integer)
++	Increment	a++; or ++a;	Increases operand by 1
--	Decrement	a--; or --a;	Decreases operand by 1

VII Required Resources/apparatus/equipment with specifications-

Sr. No.	Name of Resource	Suggested Broad Specification	Quantity
1	Desktop PC	Core i3/i5 Processor, 32/64-bit windows operating system, 2GB/Higher RAM	1
2	Open source software	AVR microcontroller - AVR studio 7 installed on PC	1

VIII Precautions to be followed

1. Check the rules/syntax of C programming
2. Check the circuit connections before power ON.
3. Avoid short circuits by checking wiring before powering the board

IX Procedure

1. Start Microchip/AVR Studio by double clicking on the icon.
2. Create a new project.
3. Select device for Target.
4. Type the program in text editor and save
6. Build the target.
7. Check for any errors in the output window and remove if any.
8. Click on Debug and start simulation and start/stop debug session.
9. Run the program step by step.
10. Observe the output on the project window. It will display all internal registers and their contents.
11. Note the contents of the registers in the observation table.

Sample Program- To perform addition, subtraction and multiplication of two 8 bit numbers.

STEP1: Algorithm-

1. **Start**
2. **Initialize variables**
   Declare two input variables a and b.
   Declare result variables: sum, diff, prod.
3. **Perform arithmetic operations**
   Calculate sum = a + b
   Calculate diff = a - b
   Calculate prod = a * b
4. **Store the result**
   Store the result of addition, subtraction and multiplication in PORTB
5. **End**

STEP2: Program-

Program	Comments

#include <xc.h>
#define F_CPU 16000000UL
#include <avr/io.h>
#include <util/delay.h>
	// Define clock frequency (16 MHz)
	// AVR input/output definitions
	// For _delay_ms()

int main(void)
{
const uint8_t a = 10;
const uint8_t b = 5;
uint16_t sum,prod;
uint8_t diff;
DDRB = 0xFF;
while(1)
{
// Addition
sum = a + b;
PORTB = sum;
_delay_ms(100);
// Subtraction
diff = a - b;
PORTB = diff;
_delay_ms(100);
// Multiplication
prod = a * b;
PORTB = prod;
_delay_ms(100);
}
}
	// Define constant data
	
	// To store operation results
	// Set PORTB as output
	// All pins of PORTB set as output
	
	// Calculate sum
	// Output result to PORTB
	// 100 m second delay
	
	// Calculate subtraction
	// Output result to PORTB
	// 100 m second delay
	
	// Calculate multiplication
	// Output result to PORTB
	// 100 m second delay

Output window:

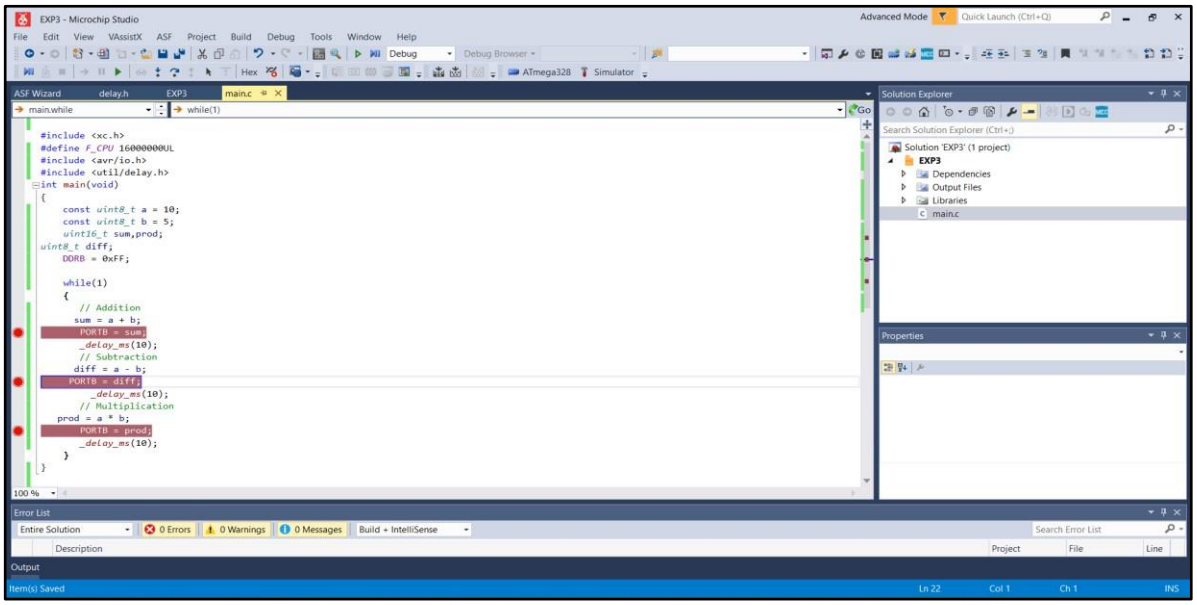


Figure 3.1 Editor window

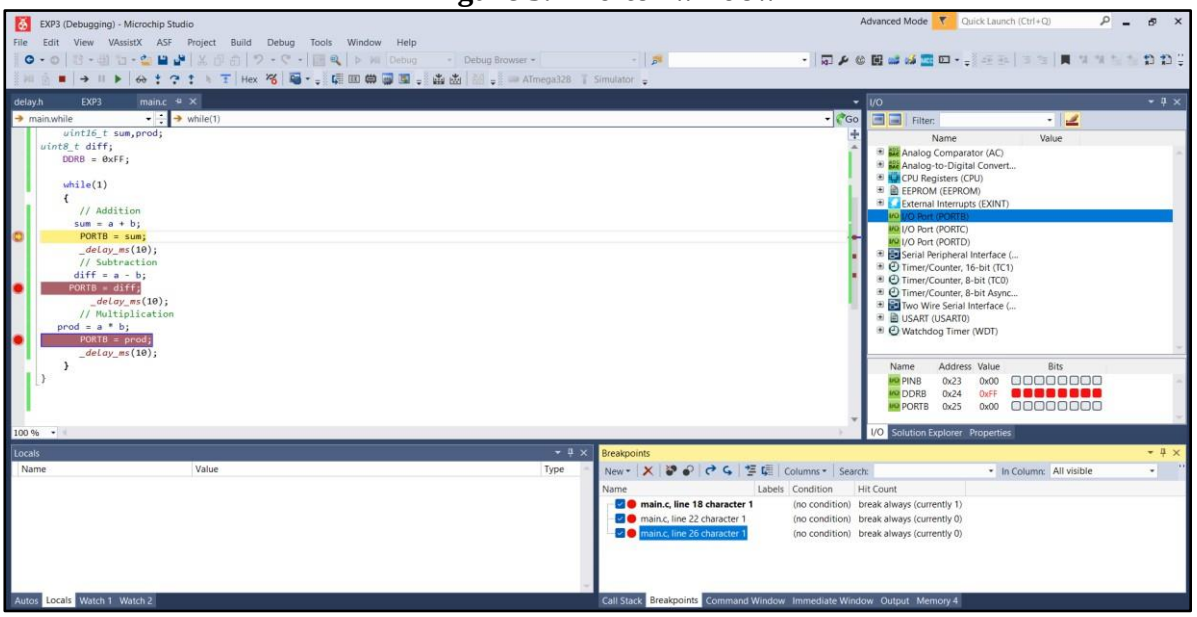


Figure 3.2 Initial value of PORT B

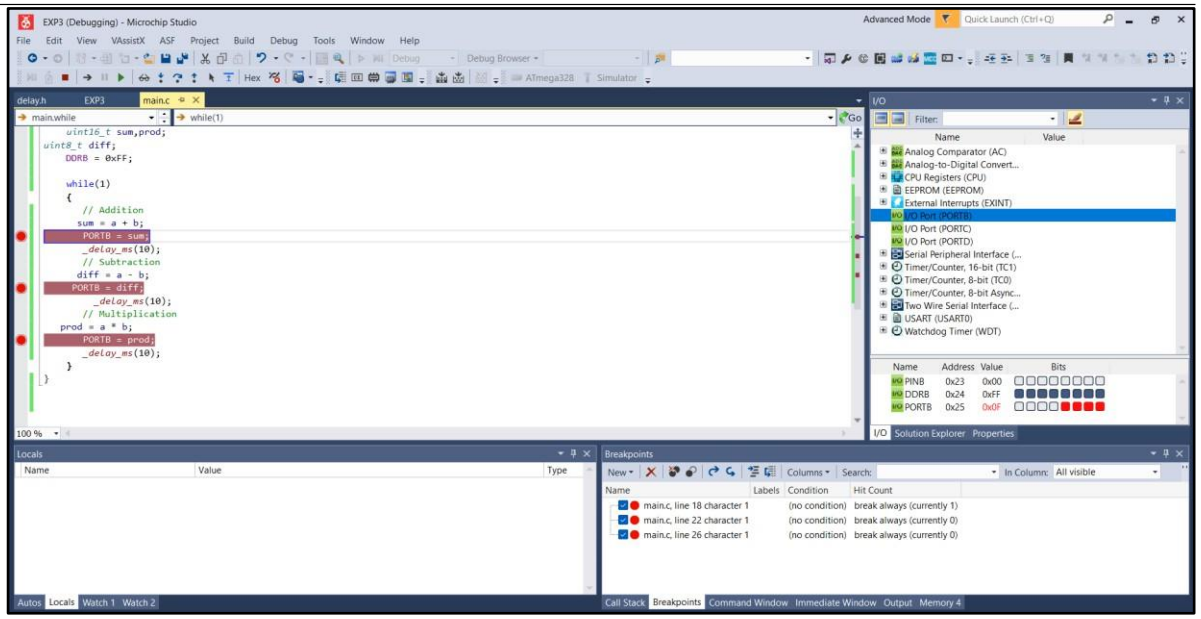


Figure 3.3 Result of addition on PORT B

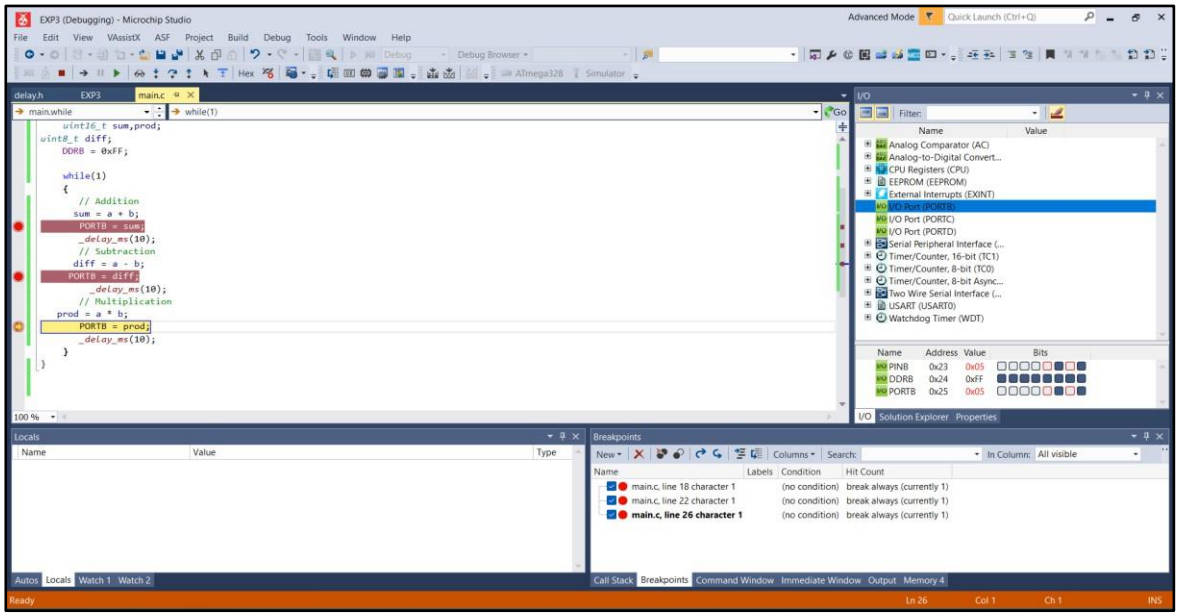


Figure 3.4 Result of subtraction on PORT B

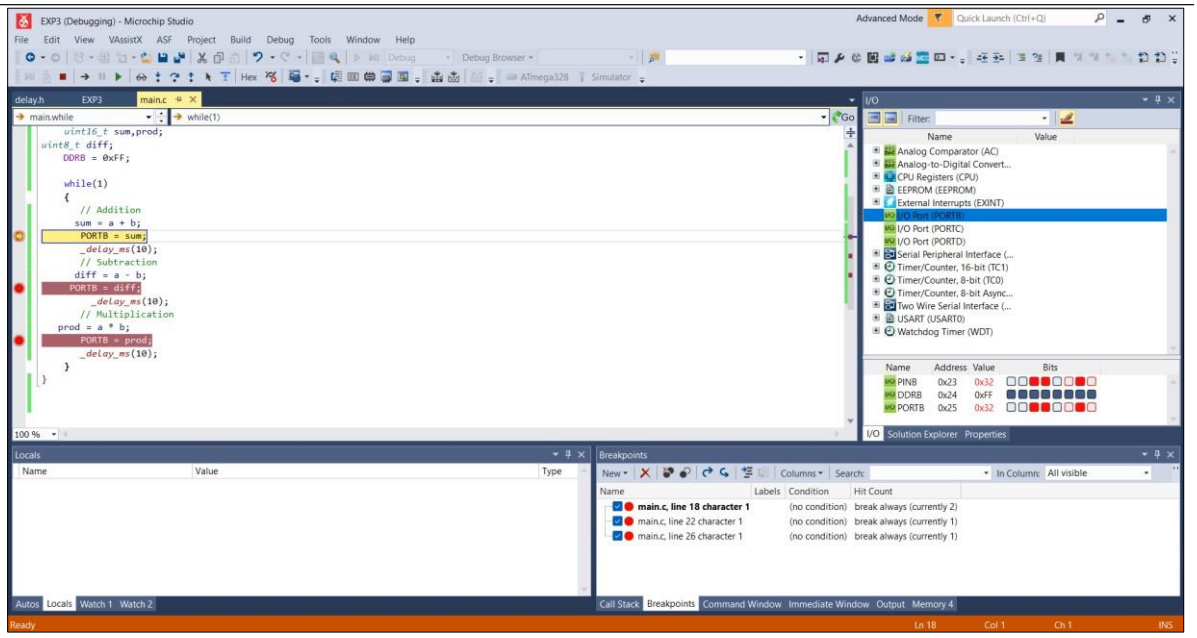


Figure 3.5 Result of multiplication on PORT B

Problem statement for student: Write an AVR program that accepts two numbers from PORTA and performs addition, subtraction, and multiplication. Display the result on PORTB with proper indication for each operation.

Program

Program	Comments

X Resources used

Sr. No.	Name of Resource	Specification	Quantity
1			
2			
3			

XI Actual Procedure (use blank sheet provided if space not sufficient)

……..……

……..……

……

……..……

……..……

……..……

……..……

XII Observation Table for the program for students (use blank sheet provided if space not sufficient)

Table 1-

Arithmetic Operation	Data Byte 1	Data Byte 2	Result after execution
Addition			
Subtraction			
Multiplication			

XIII Results (Output of the program)

……

……

……

……

XIV Interpretation of results (Give meaning of the above obtained results)

……

……

……

XV Conclusion and recommendation (Actions/decisions to be taken based on the interpretation of results)

……

……

……

……

……

XVI Practical related questions

Note: Below given are a few sample questions for reference. Teachers must design more such questions so as to ensure the achievement of identifying CO.

1. Define constants and variables in AVR C for arithmetic operations?
2. State the header files essential for the AVR program for problem statement for students?
3. State the importance of configuring data direction registers (DDRx) in AVR.
4. Write an AVR program to perform division of two 8 bit numbers.
5. Can PORTC be used simultaneously for indication while PORTB displays result? How?
6. Will the result be affected if one operand is changed from uint8_t to int8_t? Justify the answer

[Space for Answers]

……

……

……

……

……

……

……

……

……

……

……

……

……

……

XVII References/Suggestions for further reading

1. https://www.microchip.com/en-us/tools-resources/develop/microchip-studio#Downloads

2. https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-9365-Automotive-Microcontrollers-ATmega88-ATmega168_Datasheet.pdf

3. https://www.farnell.com/datasheets/2047865.pdf

4. https://ww1.microchip.com/downloads/en/devicedoc/39582b.pdf

5. https://nptel.ac.in/courses/117104072

6. https://youtu.be/s_AETeLbKwk

XVIII Assessment Scheme

Performance Indicators		Weightage
Process Related : 15 Marks		**60 %**
1	Coding and Debugging ability	30%
2	Making connections of hardware	20%
3	Follow ethical practices	10%
Product Related: 10 Marks		**40%**
4	Correct program of student activity	20%
5	Relevance of output of the problem definition	10%
6	Conclusion and Practical related questions	10%
Total : 25 Marks		**100 %**

Marks Obtained			Dated signature of Teacher
Process Related (15)	**Product Related (10)**	**Total (25)**	

Practical No. 4: Interface LED matrix with AVR microcontroller

I Practical Significance

Interfacing an LED matrix with an AVR microcontroller helps students understand how to control multiple LEDs using digital outputs. It demonstrates practical use of ports, bit patterns, and timing. This enhances their skills in embedded systems and microcontroller programming. It is widely used in display systems like digital clocks and signboards.

II Industry/Employer expected outcome

Develop simple applications based on embedded system.

III Course Level Learning Outcome(s)

- Choose appropriate family of microcontroller for different applications

IV Laboratory Learning Outcome(s)

- Interface 4 x4 LED matrix with AVR.
- Develop C program to display various patterns.

V Relevant Affective domain related Outcome(s)

1. Demonstrate working as a leader or a team member.
2. Follow ethical practices.

VI Relevant Theoretical Background

AVR is a family of microcontrollers developed by Atmel (now part of Microchip Technology). AVR microcontrollers are based on the RISC (Reduced Instruction Set Computing) architecture, which allows them to execute instructions in a single clock cycle, making them fast and efficient. They are 8-bit microcontrollers commonly used in embedded systems, robotics, and automation projects.

Popular AVR microcontrollers include ATmega8, ATmega16, and ATmega328. They support serial communication protocols like UART, SPI, and I²C, and have timers, counters, and ADCs (Analog-to-Digital Converters). AVR microcontrollers can be programmed using C language through development environments such as Atmel Studio or Arduino IDE.

Interfacing an LED matrix with an ATmega8 microcontroller allows the display of characters, numbers, and patterns using LEDs arranged in rows and columns. The ATmega8 controls the LEDs by sending signals to specific row and column pins, lighting up the desired LEDs. Multiplexing is used to reduce the number of I/O pins by turning on one row or column at a time very quickly, giving the illusion that all LEDs are on. The microcontroller outputs binary bit patterns through ports like PORTB, PORTC, or PORTD to control the matrix. Current-limiting resistors are connected to prevent LED damage. Bitwise operations are used in programming (C or Assembly) to send the right patterns to the LED matrix.

VII Actual Circuit diagram used in laboratory with related equipment rating-

A) Sample Circuit Diagram-

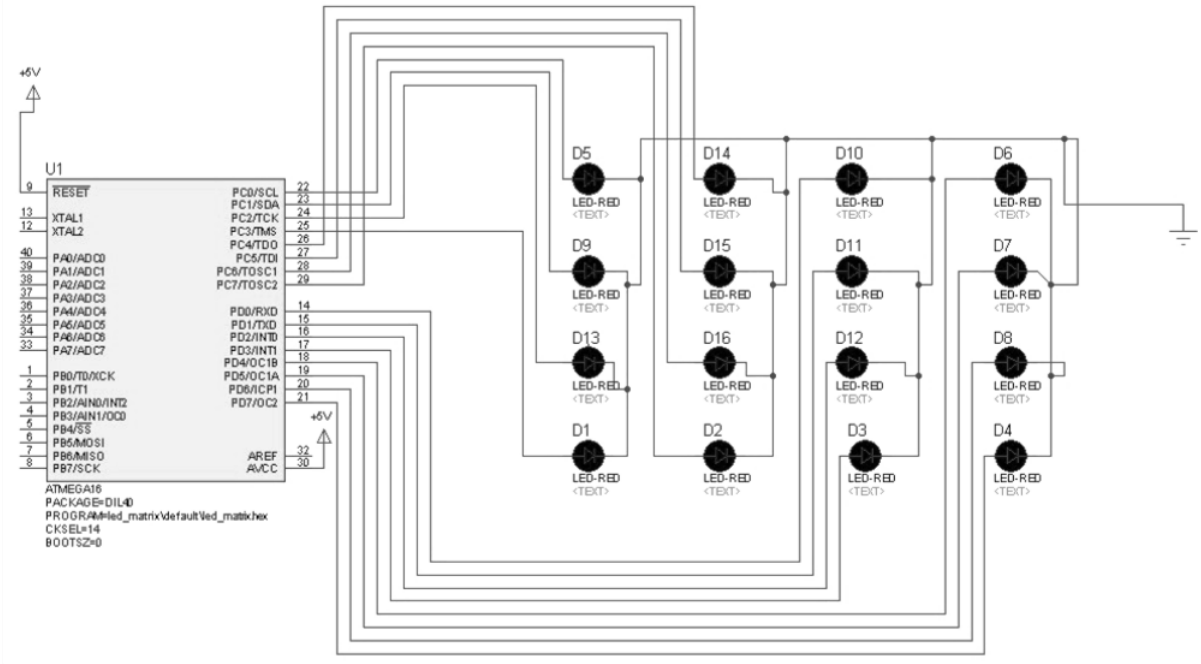


Fig 4.1 Experimental Set up

B) Actual Circuit Diagram used in Laboratory

VIII Required Resources /apparatus/equipment with specifications

Sr. No.	Instrument /Components	Suggested Broad Specification	Quantity
1.	Desktop PC	Loaded with Arduino open-source IDE, simulation and program downloading software	1 No.
2.	Microchip Studio IDE	• Supported Microcontrollers: AVR (e.g., ATmega series) and ARM Cortex-M based SAM microcontrollers • Programming Languages: C, C++, and Assembly • Compiler: Includes AVR GCC and ARM GCC compilers • Debugger Support: Supports on-chip debugging using tools like Atmel-ICE, JTAGICE3, AVR Dragon, etc. • Project Management: Provides project wizards, code templates, and solution explorer for easy management • Device Programming: Supports In-System Programming (ISP), JTAG, and debugWIRE • Simulator: Built-in software simulator for testing code without hardware	1 No.
3.	AVR Microcontroller	• Architecture: 8-bit RISC (Reduced Instruction Set Computing) • Operating Voltage: 2.7V to 5.5V • Clock Speed: Up to 20 MHz (depending on model) • Flash Memory: 4KB to 256KB (for program storage) • SRAM: 128 bytes to 8KB (for data storage) • EEPROM: 64 bytes to 4KB (for permanent data storage) • I/O Pins: 15 to 86 general-purpose input/output pins • Timers/Counters: 8-bit and 16-bit timers available • ADC (Analog-to-Digital Converter): 10-bit resolution, multiple input channels • Communication Interfaces: Supports UART, SPI, and I²C protocols • Interrupts: Internal and external interrupt support • Watchdog Timer: For system reset in case of software failure	1 No.
4.	Breadboard	-	1 No.

Sr. No.	Instrument /Components	Suggested Broad Specification	Quantity
5.	LED	Red/Green	16 No.
6.	Resistor	150-ohm	As required
7.	Connecting Wires	-	As required

IX Precautions to be Followed

1. Check rules /syntax of AVR microcontroller programming.
2. Check the circuit connections before power ON.
3. Avoid short circuits by checking wiring before powering the board

X Procedure

1. Start Microchip/AVR Studio by double clicking on the icon.
2. Create a new project.
3. Select device for Target.
4. Type the program in text editor and save
5. Build the target.
6. Check for any errors in the output window and remove if any.
7. Click on Debug and start simulation and start/stop debug session.
8. Run the program step by step.
9. Observe the output.

Sample Program- Interface 4 x4 LED matrix with AVR and develop C program to display various patterns

STEP1: Algorithm-

1. Make the circuit connections as per given diagram.
2. Start the program.
3. Set PORT C and PORT D of ATmega16 as output ports.
4. Clear the ports by sending 0x00 before displaying zero.
5. Send the bit pattern to light up LEDs to show the number zero.
6. Wait for a short time (add delay).
7. Clear the ports again with 0x00 before displaying one.
8. Send the bit pattern to light up LEDs to show the number one.
9. Repeat from step 4 continuously.

STEP 2: Program-

Program	Comments

#define F_CPU 1000000UL
#include <avr/io.h>
#include <util/delay.h>
	// Sets the CPU frequency to 1 MHz
	// Includes the AVR input/output register definitions
	// Includes delay functions

int main(void)
{
DDRC = 0xFF;
DDRD = 0xFF;
 	
while(1)
{
PORTC = 0x00;
PORTD = 0x00;
 	
PORTC = 0x9F;
PORTD = 0xF9;
 	
_delay_ms(10000);
 	
PORTC = 0x00;
PORTD = 0x00;
 	
PORTC = 0xA0;
PORTD = 0x8F;
 	
_delay_ms(10000);
}
}
	
	// Sets all PORTC pins as output
	// Sets all PORTD pins as output
	
	// Infinite loop
	
	// Clear PORTC (turn off all rows/columns)
	// Clear PORTD
	
	// Send pattern to PORTC
	// Send pattern to PORTD (displays digit or pattern on LED matrix)
	// Delay of 10 seconds
	
	// Clear PORTC again
	// Clear PORTD
	
	// Send a new pattern to PORTC
	// Send a new pattern to PORTD (displays another digit/pattern)
	// Delay of 10 seconds

Problem statement 1 for student: Write a program to display A, V and R on 4 x 4 matrix LED after certain delay.

Program	Comments

Program	Comments

XI Resources used/apparatus/equipment with specifications:

S. No.	Instrument /Components	Specification	Quantity
1.			
2.			
3.			
4.			
5.			
6.			

XII Actual Procedure Followed (use blank sheet provided if space not sufficient)

……

……

……

……

……

……

……

……

……

……

……

……

……

XIII Observations for Student Activity program (use blank sheet provided if space not sufficient)

……

……

……

XIV Results (Output of the Program)

……

……

……

……

XV Interpretation of Results (Give meaning of the above obtained results)

……

……

……

……

XVI Conclusions and Recommendation (Actions/decisions to be taken based on the interpretation of results).

……

……

……

……

XVII Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO

a. How many I/O pins are required to control a 4x4 LED matrix?
b. Give the significance of DDRC = 0xFF and DDRD = 0xFF in the program.
c. What is the purpose of using a current-limiting resistor with LEDs?

[Space for Answers]

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

XVIII References / Suggestions for further reading

1. http://vlabs.iitkgp.ac.in/rtes/exp11/index.html#
2. https://www.vskills.in/certification/tutorial/interfacing-switch-and-led-with-arduino/

XIX Assessment Scheme

The given performance indicators should serve as a guideline for assessment regarding process and product related marks:

Performance indicators		Weightage
Process related: 15 Marks		**60%**
	Use of IDE tools for programming	20%
	Coding and Debugging ability	30%
	Follow ethical practices.	10%
Product related: 10 Marks		**40%**
	Correctness of Program for student activity	20%
	Relevance of output of the problem definition	15%
	Timely Submission of report, Answer to sample questions	05%
	Total: 25 Marks	**100 %**

Marks Obtained			Dated signature of Teacher
Process Related (15)	**Product Related (10)**	**Total (25)**	

Practical No.5: Serial Communication using USB.

I Practical Significance

Through this experiment, students gain hands-on experience in data transmission and reception between a microcontroller (or development board like Arduino) and a PC using USB. Microcontrollers typically use UART (Universal Asynchronous Receiver Transmitter) protocol for serial communication. USB-to-Serial converters act as bridges between the PC's USB and the microcontroller's UART.

II Industry/Employer Expected Outcome

- Develop simple applications based on embedded system.

III Course Level Learning Outcome

- Interpret the communication standards of embedded systems.

IV Laboratory Learning Outcome

- Configure USB protocol on PC.

V Relevant Affective Domain related outcomes

1. Demonstrate working as a leader or a team member.
2. Follow ethical practices.

VI Relevant Theoretical Background

USB (Universal Serial Bus) has become the standard interface for connecting peripheral devices to computers and microcontrollers due to its reliability, high speed, and plug-and-play capability. USB, a serial protocol that also adheres to the RS-232 standard, was introduced as a more advanced and standardized way for devices to communicate through handshaking, device detection, auto speed negotiation, etc. UART is still used today, but is primarily seen in GPS and Bluetooth modules, Arduino, and other microcontrollers.

Serial communication involves transmitting data one bit at a time over a single wire, making it simple and efficient for many embedded applications. Parameters like baud rate, data bits, stop bits, and parity must be correctly configured on both ends. Each USB device uses the standard A type connector to the USB host or Hub through A type receptacle. The other end of the cable has series B connector which is used to plug into the B type receptacle. A connector is used for the upstream connection towards the host and B connector for the downward stream to the USB device. When the device is connected to the PC, it activates the host to recognize it. The PC detects the device and manages a control flow between the device and computer.

PC manages the data transfer between the device and PC. Once detected, the PC sends data to the USB system software to recognize it which then identifies the device and assigns an address. This address is used to detect the particular USB device. The software controls the

input and output data between the PC and device. If the software fails to assign the address, the PC will not detect the device.

Features of USB:

1. Plug and Play: Devices can be connected or disconnected without restarting the computer or system.
2. Up to 127 devices can be connected to a single USB host using hubs.
3. Host-Controlled Communication: All data transfer is initiated and managed by the host (e.g. a PC); devices cannot communicate directly with each other.
4. Supplies power to connected devices (e.g., charging phones or powering peripherals), typically 5V with varying current levels.
5. Devices can enter low-power states or automatically shut down when not in use.
6. Provides universal connectors such as USB-A, USB-B, USB-C, and micro/mini variants.

Electrical Characteristics:

USB uses a differential transmission pair for data.

- On low and full speed devices, a differential '1' is transmitted by pulling D+ over 2.8V and D- under 0.3V with appropriate pull up/pull down registers.
- A differential '0' on the other hand is a D- greater than 2.8V and a D+ less than 0.3V with the same appropriate pull down/up resistors
- Physical wire consists of 4 pins
- Two wires D+ and D- are used to carry the signal
- Vbus is normally +5V at the source.
- USB cable connectors have power pins longer than signal pins so that power is applied before the signals.

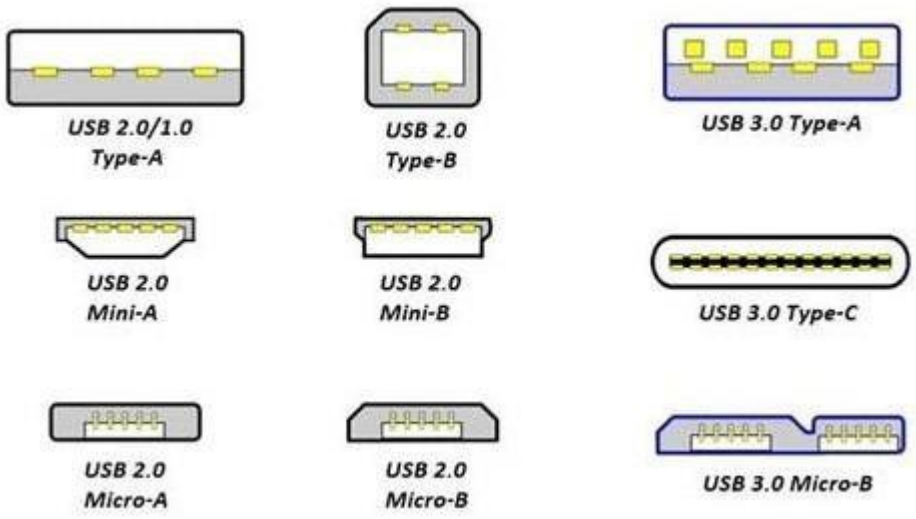


Figure 5.1 Types of USB connectors

VII Actual Circuit diagram used in laboratory with related equipment rating-

A) Sample Setup Diagram-

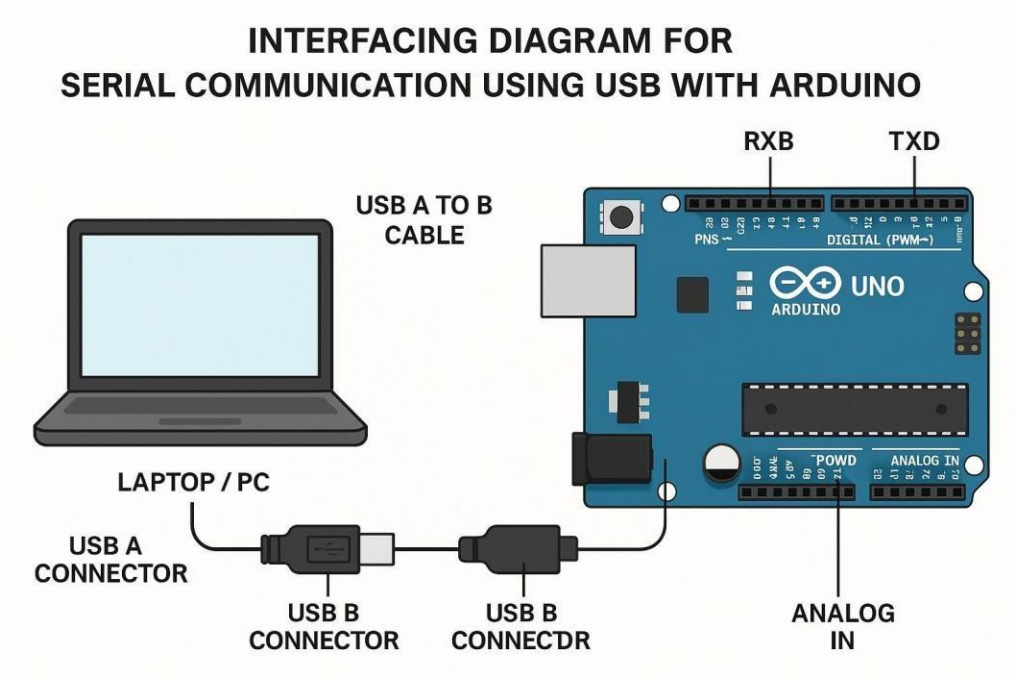


Figure 5.2 Serial Communication using USB with Arduino

B) Actual Setup Diagram used in Laboratory -

VIII Required Resources/apparatus/equipment with specifications-

Sr. No.	Name of Resource	Suggested Broad Specification	Quantity
1	Desktop PC	Core i3/i5 Processor, 32/64-bit windows operating system, 2GB/Higher RAM With USB port	1
2	Microcontroller Board	Arduino Uno / ATmega328P / STM32 / PIC etc.	1
3	USB Cable	USB A to B / USB to micro-USB	1
4	USB-to-Serial Converter (if needed)	FT232 / CP2102 / CH340G	1
5	Serial Communication Software	Arduino IDE-Latest version (from Arduino.cc)	1

IX Precautions to be followed

1. Check the rules/syntax of C programming
2. Check the circuit connections before power ON.
3. Avoid short circuits by checking wiring before powering the board

X Procedure

1. Connect the microcontroller board to the PC using a USB cable.
2. Open the serial monitor or terminal software on the PC.
3. Configure the baud rate (e.g. 9600 bps) and communication settings (8 data bits, no parity, 1 stop bit).
4. Write and upload a code to the microcontroller.
5. Observe the communication in the terminal window.
6. Modify the program to send sensor data or interact with digital inputs.
7. Verify bidirectional communication by sending data from PC to MCU and vice versa.

Sample Program- To perform and understand serial communication between a microcontroller and a PC using USB interface.

STEP1: Algorithm-

1. Start the program.
2. In the setup() function:
   Initialize serial communication using Serial.begin(9600) to set the baud rate to 9600 bits per second.
3. Enter the loop() function (this runs continuously).
4. Send the message "MSBTE" to the serial monitor using Serial.println().
5. Wait for 1 second using delay(1000).
6. Check if any data has been received from the serial port using Serial.available().
7. If data is available:
   Read one character from the serial buffer using Serial.read().
   Print the message "You typed: " followed by the received character using Serial.print() and Serial.println().
8. Repeat steps 4 to 7 continuously.
9. End.

STEP2: Program-

Program	Comments

void setup() {
  Serial.begin(9600);
}
void loop() {
  Serial.println("MSBTE");
  delay(1000);
  if (Serial.available()) {
  char c = Serial.read();
Serial.print("You typed: ");
  Serial.println(c);
  }
}
	// Start serial communication at 9600 baud
	
	//Transmit data serially- “MSBTE”
	// Wait for 1 second
	
	// Read character from serial port
	// Echo received character

Output window:

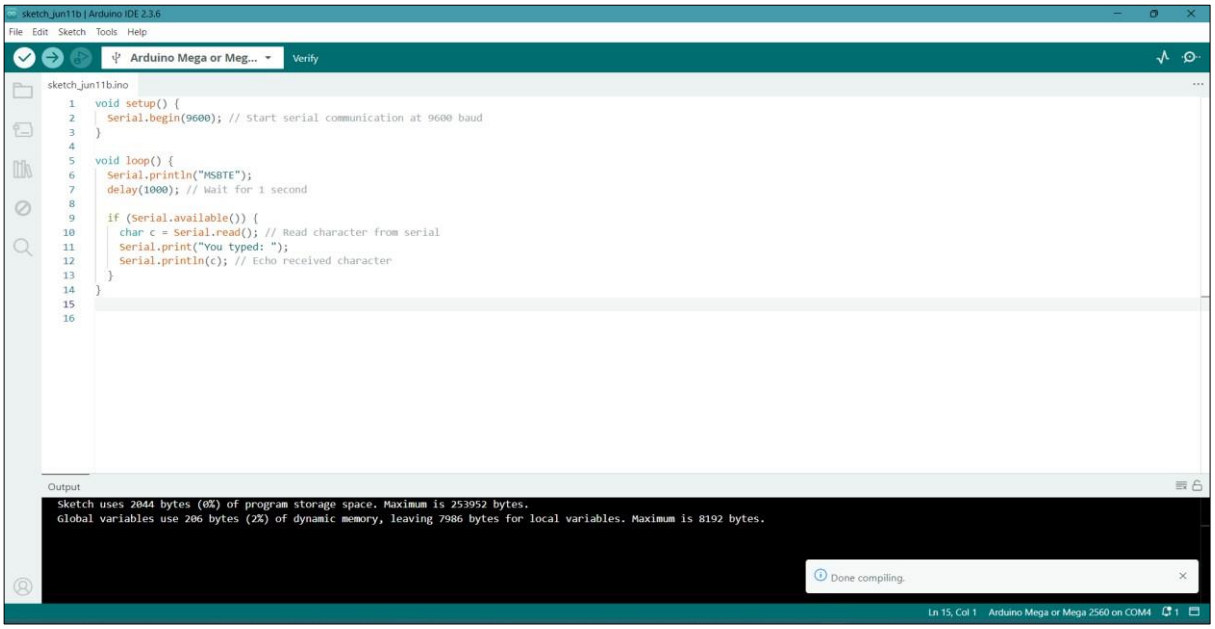


Figure 5.3 Editor window of the program- Compiling the program in Arduino IDE.

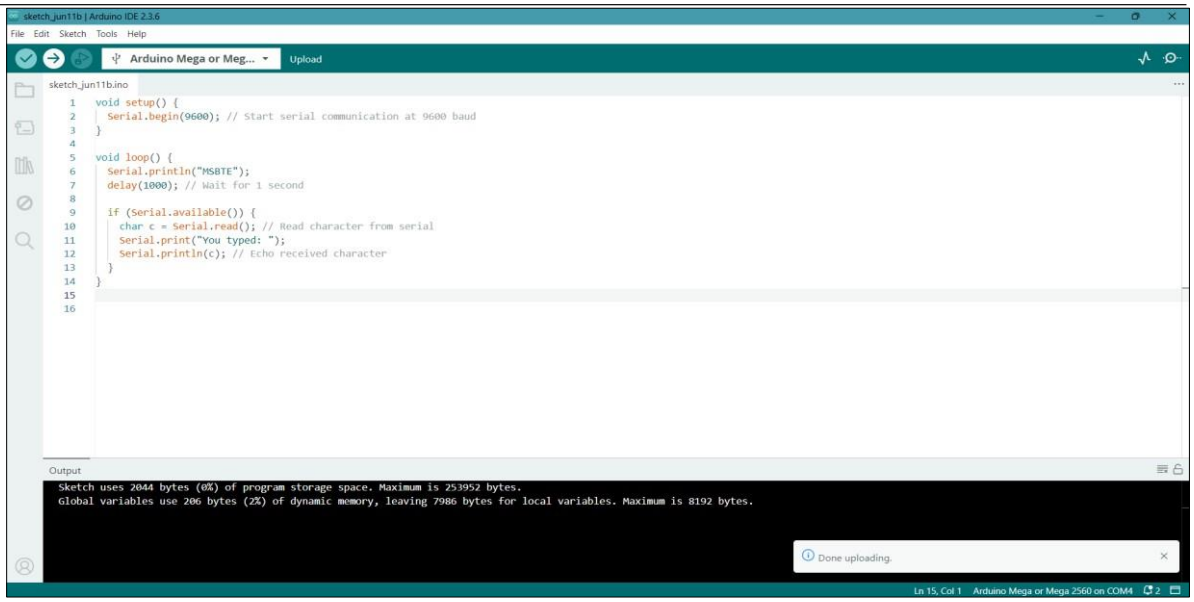


Figure 5.4 Output window of the program- Uploading the program on Arduino.

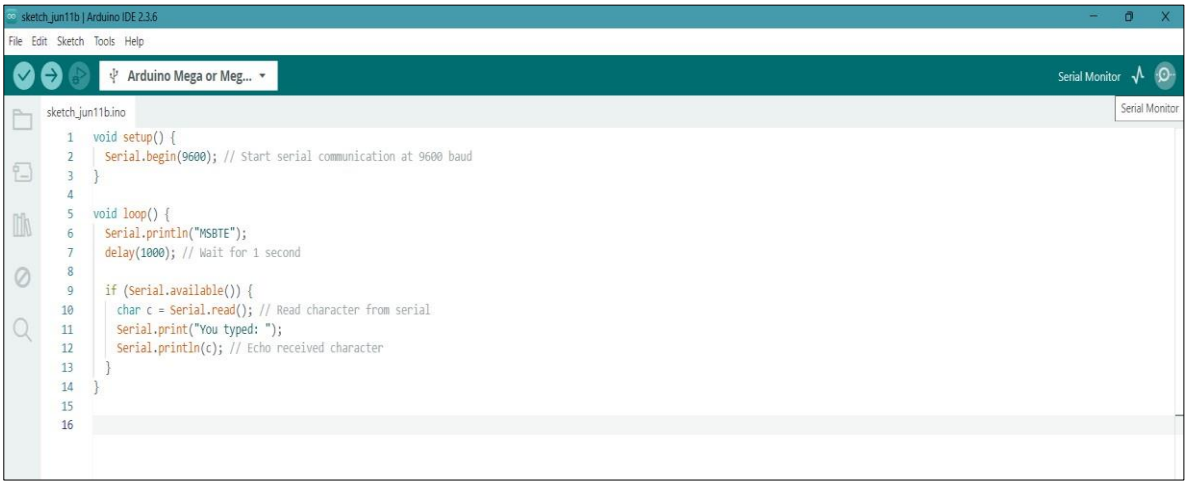


Figure 5.5 Output window of the program- Selecting serial monitor

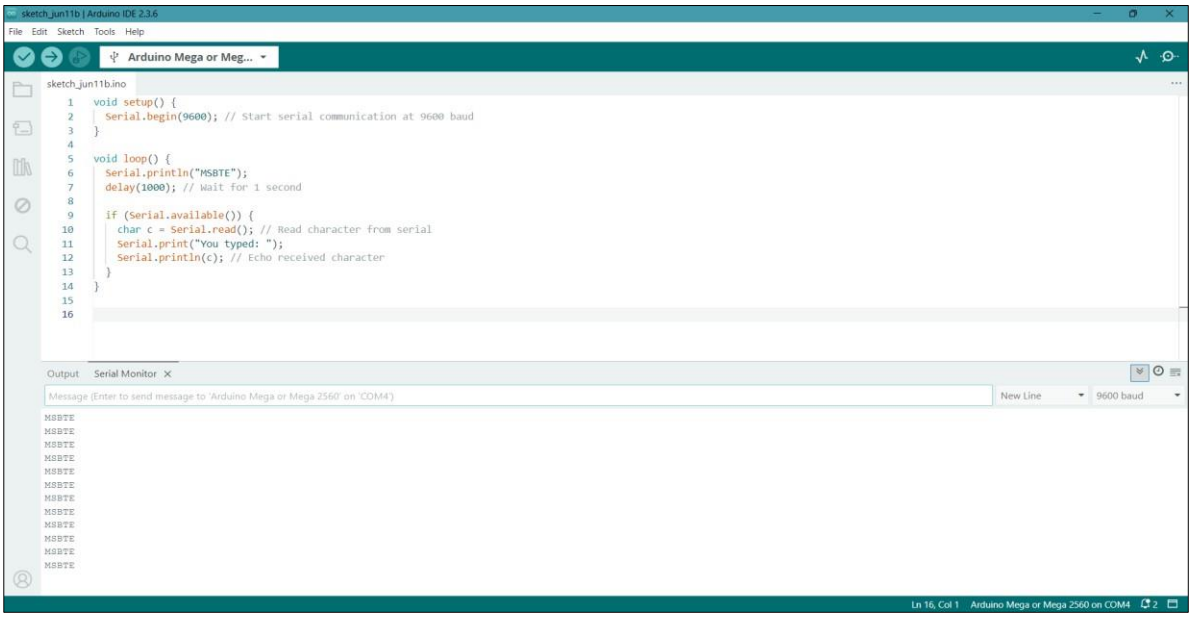


Figure 5.6 Output window of the program- sending "MSBTE" from Arduino

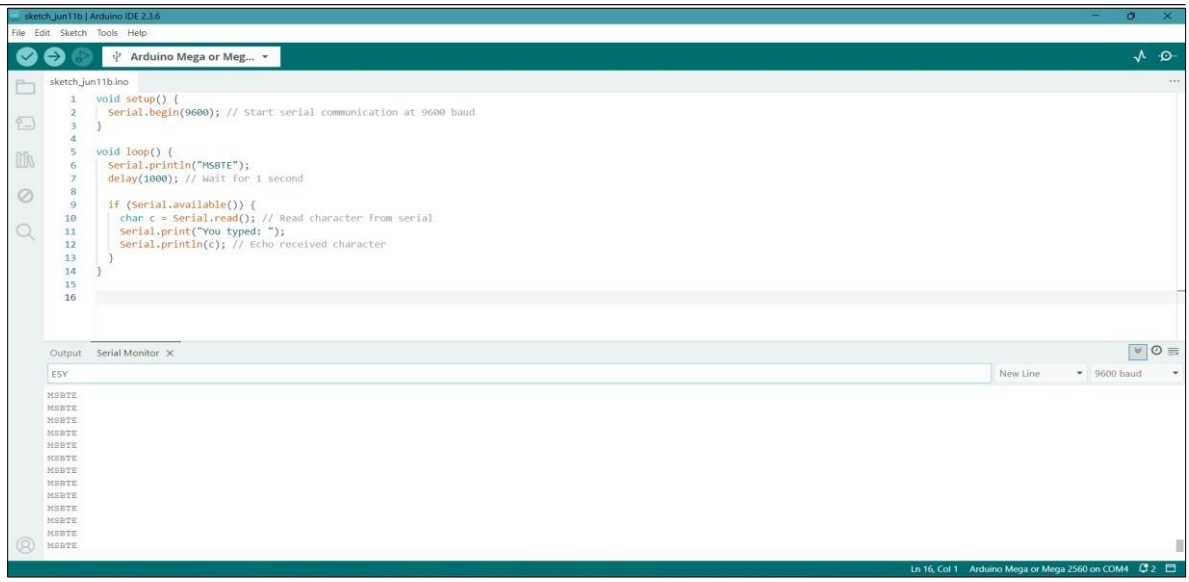


Figure 5.7 Output window of the program- sending "ESY" from PC

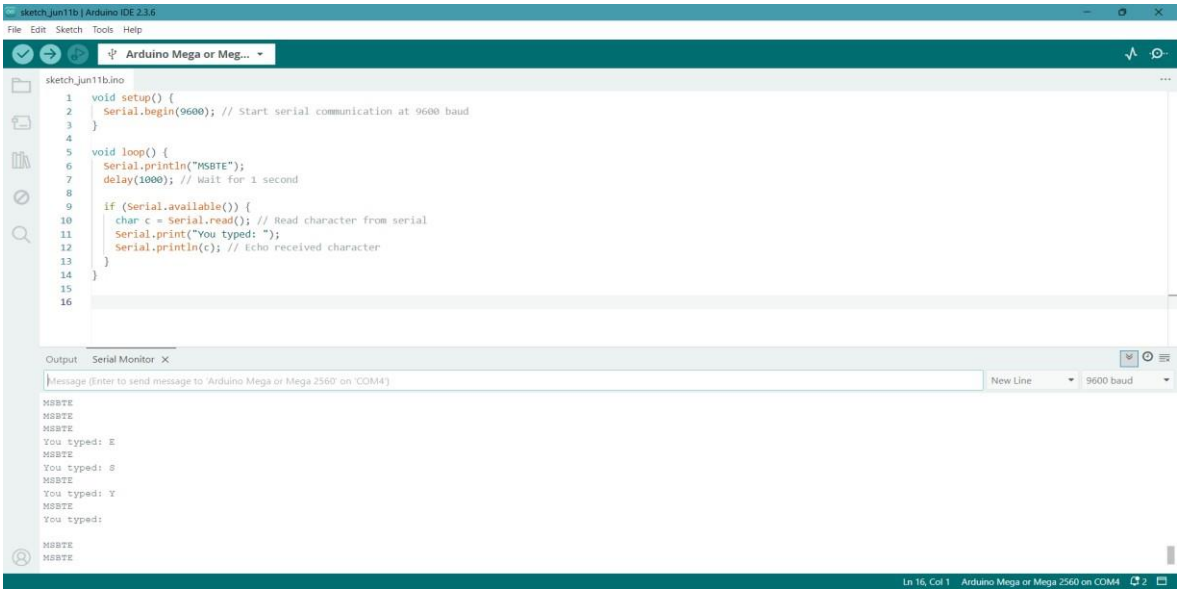


Figure 5.8 Output window of the program- receiving "ESY" by Arduino.

Problem statement for student: Write a program to control an LED connected to an Arduino by sending commands ('1' to turn ON, '0' to turn OFF) from a PC using serial communication over USB.

Program

Program	Comments

XI Resources used

Sr. No.	Name of Resource	Specification	Quantity
1			
2			
3			
4			
5			

XII Actual Procedure (use blank sheet provided if space not sufficient)

……..

…….

..……

……..

……..

……..

……..

……..

XIII Observations for the program for students (use blank sheet provided if space not sufficient)

…….

……

..……

……..……

…… ……

XIV Results (Output of the program)

…….

……

…….

……

XV Interpretation of results (Give meaning of the above obtained results)

…….

……

…….

…….

XVI Conclusion and recommendation (Actions/decisions to be taken based on the interpretation of results)

……

…….

…….

XVII Practical related questions

Note: Below given are a few sample questions for reference. Teachers must design more such questions so as to ensure the achievement of identifying CO.

1. Give the difference between USB and UART?
2. Explain the role of a USB-to-Serial converter.

3. Give the basic steps to configure USB-to-Serial communication on a PC?
4. What are common baud rates used in serial communication?
5. What are Serial.read( ) and Serial.println( ) used for?
6. Give the function used to start serial communication in Arduino.

[Space for Answers]

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

XVIII References/Suggestions for further reading

1. https://youtu.be/ODKOfLL7sB4
2. https://www.arduino.cc/
3. https://youtu.be/vzCNfBux1h0

4. https://youtu.be/eciWL2_SDsI
5. https://www.youtube.com/watch?v=CysNp724Ldg
6. https://youtu.be/SX8z3-BEuWQ
7. https://youtu.be/ynAvySNCi-0
8. https://youtu.be/EVDnYnAyktw

XIX Assessment Scheme

Performance Indicators		Weightage
Process Related : 15 Marks		**60 %**
1	Coding and Debugging ability	30%
2	Making connections of hardware	20%
3	Follow ethical practices	10%
Product Related: 10 Marks		**40%**
4	Correct program of student activity	20%
5	Relevance of output of the problem definition	10%
6	Conclusion and Practical related questions	10%
Total : 25 Marks		**100 %**

Marks Obtained			Dated signature of Teacher
Process Related (15)	**Product Related (10)**	**Total (25)**	

Practical No. 6: Installation of Arduino IDE for Windows/MacOS/Linux operating system.

I Practical Significance

Installing the Arduino IDE is practically significant because it enables students to write, upload, and debug code for Arduino microcontrollers. It provides an easy-to-use platform to learn embedded systems and electronics. With it, students can develop real-world IoT and automation projects. The IDE also supports various sensors and modules, enhancing hands- on learning.

II Industry/Employer expected outcome

Develop simple applications based on embedded system.

III Course Level Learning Outcome(s)

1. Develop the basic applications using Arduino.

IV Laboratory Learning Outcome(s)

1. Install Arduino IDE and its development tool for Windows/MacOS/Linux Operating systems.

V Relevant Affective domain related Outcome(s)

1. Demonstrate working as a leader or a team member.
2. Follow ethical practices.

VI Relevant Theoretical Background

Arduino is an open-source electronics platform based on easy-to-use hardware and software. It is widely used for prototyping and educational purposes. The Arduino Uno, one of the most common boards, is equipped with digital I/O pins that can interface with a variety of input and output devices, including LEDs and switches.

Arduino acts as a foundational tool for understanding embedded systems, computational thinking, and system design through hands-on experimentation. It simplifies complex microcontroller concepts into an accessible, modular platform that aligns with constructive learning theories, enabling learners to bridge the gap between theoretical knowledge and real-world application. By supporting open-source collaboration and abstracting hardware control through a user-friendly interface, Arduino fosters innovation, promotes algorithmic thinking, and facilitates the modelling of cyber-physical systems, making it a vital educational and developmental resource in modern technological studies.

VII Required Resources /apparatus/equipment with specifications

Sr. No.	Instrument /Components	Suggested Broad Specification	Quantity
1.	Desktop PC	Loaded with Arduino open-source IDE, simulation and program downloading software	1 No.
2.	Arduino Uno	Microcontroller: ATmega328P Operating Voltage: 5V Input Voltage: 7-12V Inout Voltage (limit): 6-20V Digital I/O Pins: 14 (6 provide PWM output) PWM Digital I/O Pins: 6 Analog Input Pins: 6 Flash Memory: 32 KB (ATmega328P) SRAM: 2 KB (ATmega328P) EEPROM: 1 KB (ATmega328P) Clock Speed: 16 MHz, LED_BUILTIN: 13	1 No.
3.	USB Cable for Arduino Board	• Arduino Uno, Mega, Leonardo → USB Type-B cable • Arduino Nano (older models) → Mini-USB cable • Arduino Nano (newer models), Micro, Due, MKR series → Micro-USB or USB-C cable	1 No.

VIII Precautions to be Followed

1. Check rules /syntax of Arduino programming.
2. Check the circuit connections before power ON.
3. Avoid short circuits by checking wiring before powering the board

IX Procedure

Step 1: Download and Install Arduino IDE

a) Visit the Arduino Software page and download the Arduino IDE for your operating system (Windows, macOS, or Linux).

b) Download the Arduino IDE software using the link:
https://www.arduino.cc/en/software/

c) Follow the installation instructions specific to your system:

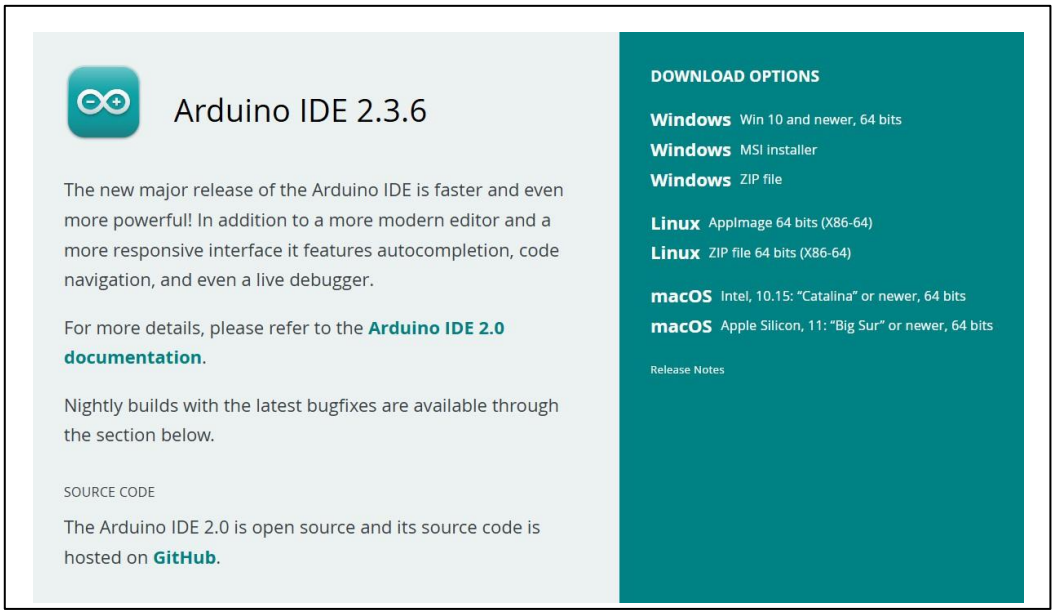


Fig 6.1: Arduino IDE download window

d) Windows: Run the installer and follow the prompts.
e) macOS: Drag the Arduino app to your Applications folder.
f) Linux: Unzip the downloaded file and run the install.sh script.
g) Once installed, open the Arduino IDE.

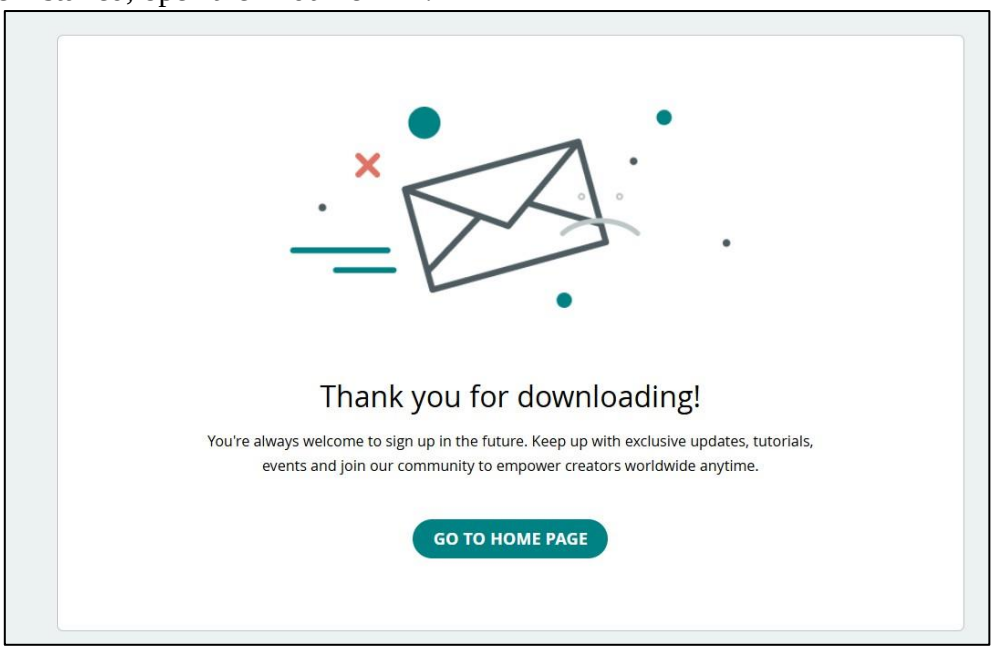


Fig 6.2: Downloading finished window

h) Accept the terms and conditions to proceed further by clicking I AGREE

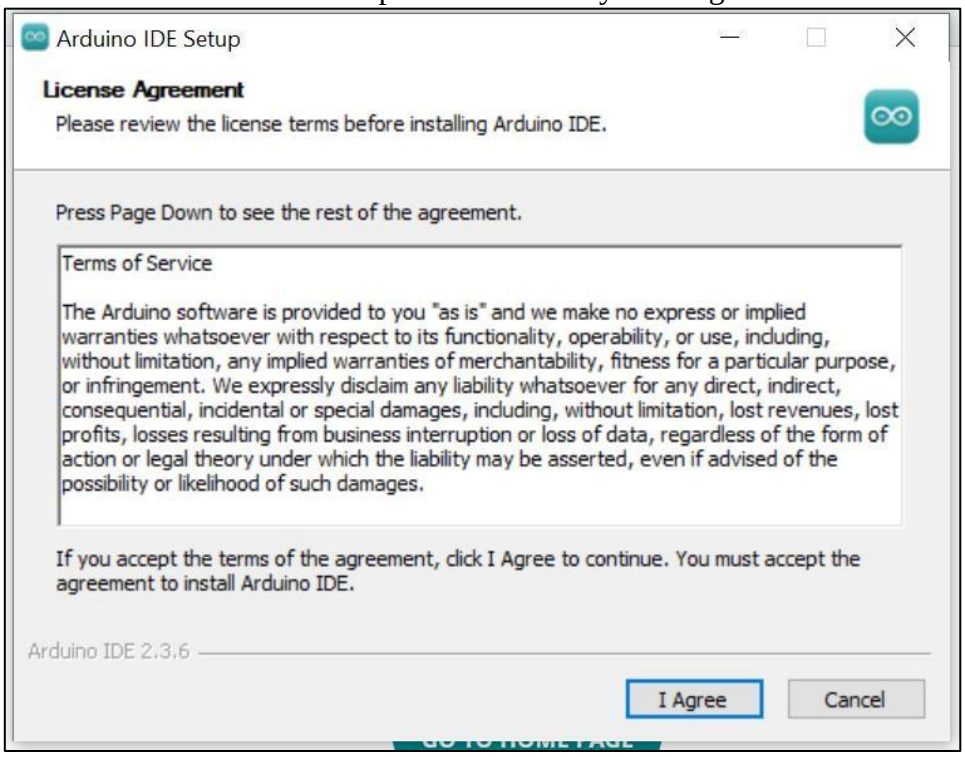


Fig 6.3: Installation Agreement Window

i) Choose the appropriate option and click NEXT

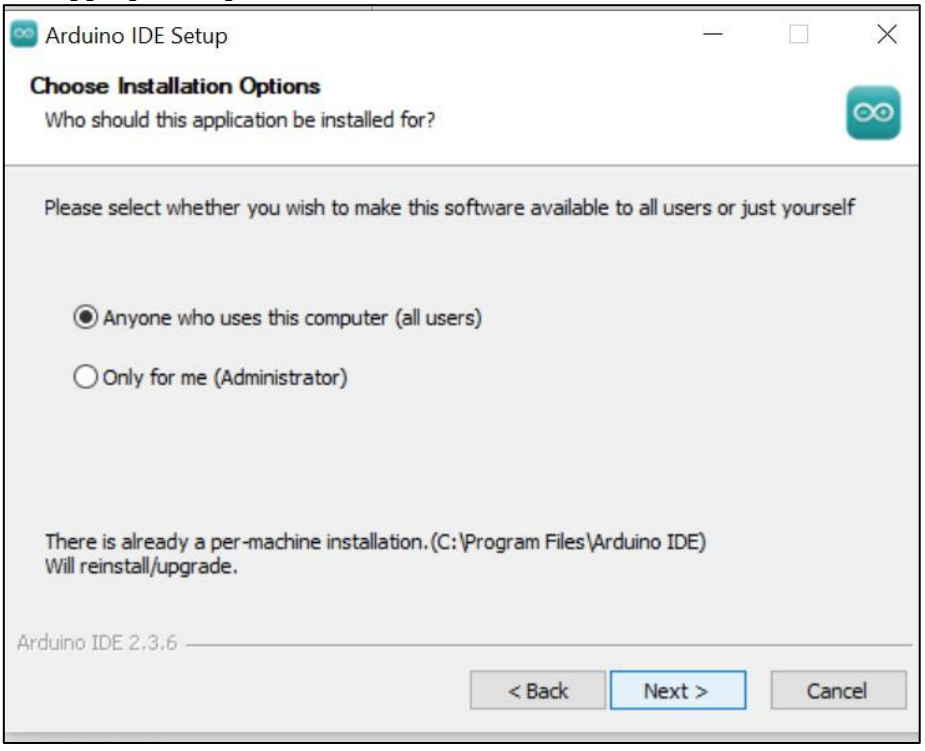


Fig 6.4: Installation Window option

j) Choose the folder for installation of Arduino IDE

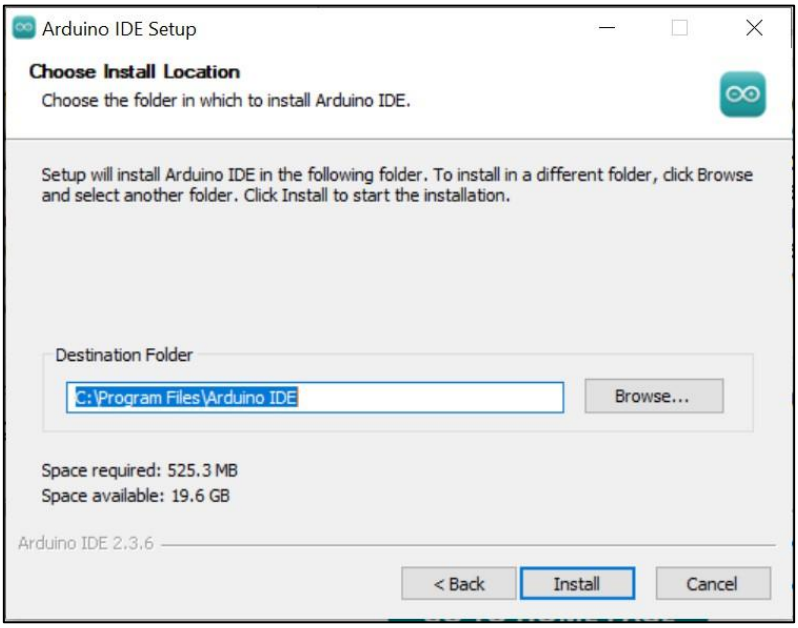


Fig 6.5: Selecting folder for installation window

k) The installation will start and progress

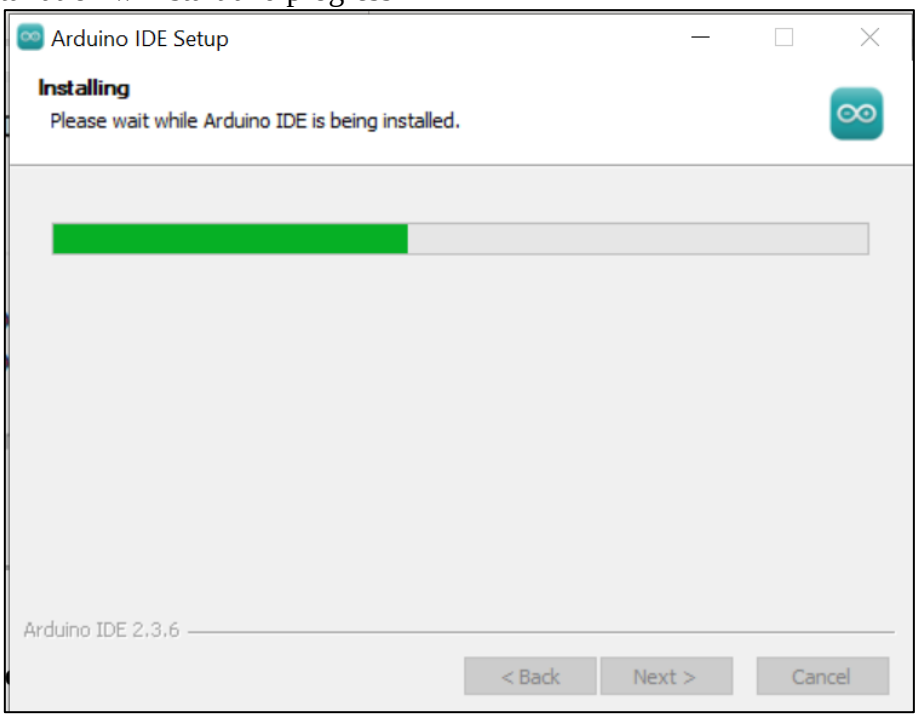


Fig 6.6: Installation Progress Window

l) Once installation is completed user can run the Arduino IDE

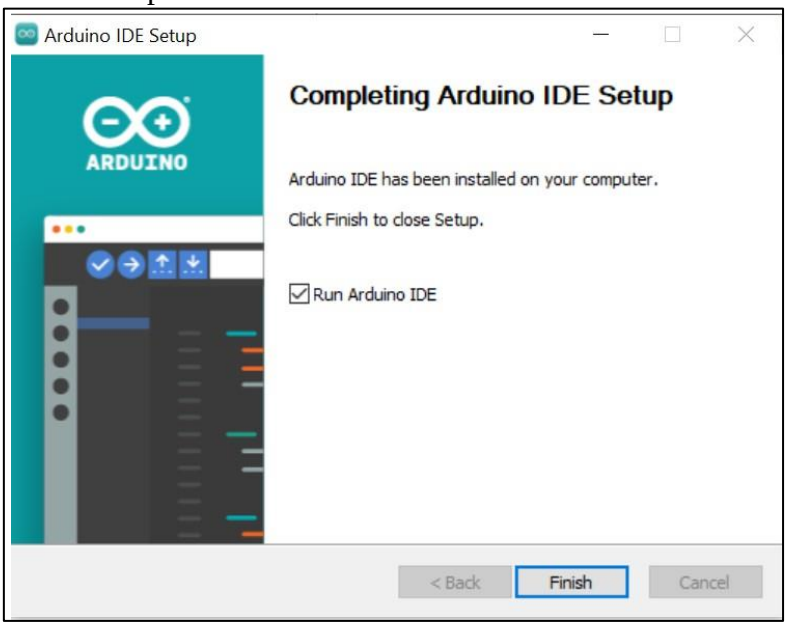


Fig 6.7: Installation Completed Window

Step 2: Connect Your Arduino Board

1. Plug the Arduino board into user's computer using the USB cable.
2. In the Arduino IDE, go to Tools > Board and select the Arduino model (e.g., Arduino Uno).
3. Then, go to Tools > Port and select the port to which the Arduino is connected.

Step 3: Install Arduino Drivers (If Needed)

- On Windows, user might need to install drivers manually. Go to Tools > Device Manager to check if the board is recognized.
- For macOS and Linux, drivers should install automatically when user connects the Arduino**.**

Step 4: Test Your Setup With the Blink Sketch

1. Open the Arduino IDE.
2. Go to File > Examples > 01.Basics > Blink.
3. This will load a simple sketch that blinks the onboard LED.
4. Click the Upload button (right-arrow icon) to send the sketch to Arduino.
5. User should see the onboard LED blinking on Arduino!

Step 5: Troubleshooting Tips

1. If Arduino board isn't recognized then :
2. Double-check the connection and the selected port in the Tools > Port menu.
3. Try a different USB cable.
4. If the upload fails, make sure user've selected the correct board model and port.

X Resources used/apparatus/equipment with specifications:

S. No.	Instrument /Components	Specification	Quantity
1.			
2.			
3.			

XI Actual Procedure Followed (use blank sheet provided if space not sufficient)

……
……
……
……
……
……
……
……
……
……
……
……
……

XIIConclusions and Recommendation

……

……

……

……

XIII Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO

1. List the different types of Arduino Boards available.
2. Compare setup() and loop() functions in Arduino.
3. How do you upload a program to an Arduino board?

[Space for Answers]

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

XIV References / Suggestions for further reading

1. https://www.arduino.cc/
2. https://www.arduino.cc/en/Guide/PortableIDE

XV Assessment Scheme

The given performance indicators should serve as a guideline for assessment regarding process and product related marks:

Performance indicators		Weightage
Process related: 15 Marks		**60%**
1	Identifying the given IDE	20%
2	Identifying the features of IDE	30%
3	Follow ethical practices.	10%
Product related: 10 Marks		**40%**
4	Correct procedure followed for installation of IDE	20%
5	Answer to sample questions	15%
6	Timely Submission	05%
	Total: 25 Marks	**100 %**

Marks Obtained			Dated signature of Teacher
Process Related (15)	**Product Related (10)**	**Total (25)**	

Practical No. 7: Building and Testing switch and LED interface using Arduino.

I Practical Significance

Building and testing a switch and LED interface using Arduino helps users to understand digital input-output operations. It also develops basic circuit design and programming skills using the Arduino IDE. This interface forms the foundation for building more high level embedded systems and automation systems and also enhances troubleshooting abilities through real-time testing and debugging. This practical will help student to turn ON/OFF LED by switch using Arduino.

II Industry/Employer expected outcome

Develop simple applications based on embedded system.

III Course Level Learning Outcome(s)

- Develop the basic applications using Arduino.

IV Laboratory Learning Outcome(s)

- Build the circuit using 4 switches and 4 LEDs to Arduino Board.
- Test the LED ON/OFF as per switch position.

V Relevant Affective domain related Outcome(s)

1. Demonstrate working as a leader or a team member.
2. Follow ethical practices.

VI Relevant Theoretical Background

Arduino is an open-source electronics platform based on easy-to-use hardware and software. It is widely used for prototyping and educational purposes. The Arduino Uno, one of the most common boards, is equipped with digital I/O pins that can interface with a variety of input and output devices, including LEDs and switches.

The LED and switch interface with Arduino illustrates simple digital output and input operations. An LED is employed as an output device that glows upon receiving current, and a switch is employed as an input device to make the circuit work.

The Arduino board senses the switch state through a digital input pin and drives the LED through a digital output pin. A resistor that limits current is put in series with the LED to limit current through it. A pull-down resistor is connected with the switch to provide a specific LOW state when the switch is not being pressed.

When the switch is pushed, Arduino sees a HIGH reading and activates the LED. pinMode() is used to initialize the pins, while digitalRead() and digitalWrite() are employed in reading input and writing output.

Debouncing methods are typically used to eliminate switch bounce false triggers. This interface is essential in the study of embedded systems and digital logic control with Arduino.

VII Actual Circuit diagram used in laboratory with related equipment rating-

A) Sample Setup Diagram -

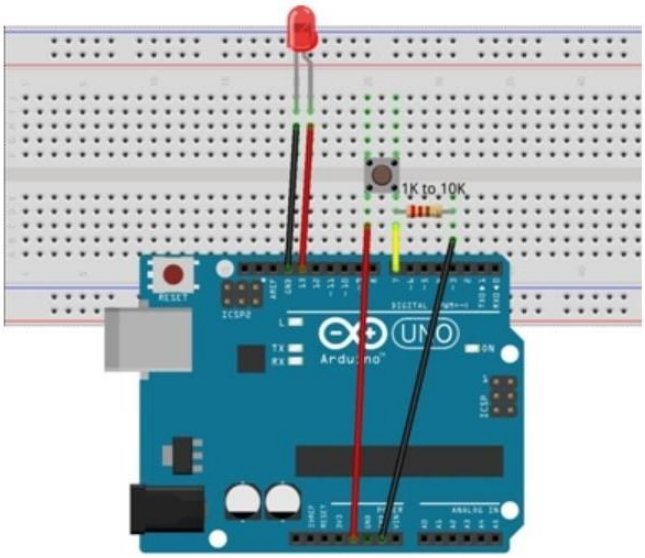


Fig 7.1 Experimental Set up

B) Actual Setup Diagram used in Laboratory

VIII Required Resources/apparatus/equipment with specifications-

Sr. No.	Instrument /Components	Suggested Broad Specification	Quantity
1.	Desktop PC	Loaded with Arduino open-source IDE, simulation and program downloading software	1 No.
2.	Arduino Uno	Microcontroller: ATmega328P Operating Voltage: 5V Input Voltage: 7-12V Inout Voltage (limit): 6-20V Digital I/O Pins: 14 (6 provide PWM output)	1 No.

Sr. No.	Instrument /Components	Suggested Broad Specification	Quantity
		PWM Digital I/O Pins: 6 Analog Input Pins: 6 Flash Memory: 32 KB (ATmega328P) SRAM: 2 KB (ATmega328P) EEPROM: 1 KB (ATmega328P) Clock Speed: 16 MHz, LED_BUILTIN: 13	
3.	USB Cable for Arduino Board	• Arduino Uno, Mega, Leonardo → USB Type-B cable • Arduino Nano (older models) → Mini-USB cable • Arduino Nano (newer models), Micro, Due, MKR series → Micro-USB or USB-C cable	1 No.
4.	Breadboard	-	1 No.
5.	LED	Red/Green	1 No.
6.	Resistor	1k to 10k	1 No.
7.	Push Button	-	1 No.
8.	Connecting Wires	-	As required

IX Precautions to be Followed

1. Check rules /syntax of Arduino programming.
2. Check the circuit connections before power ON.
3. Avoid short circuits by checking wiring before powering the board

X Procedure

1. Launch Arduino IDE.
2. Make the circuit as given in connection diagram
3. Open your first project.
   i. To create a new project, select File → New.
   ii. To open an existing project, select File → Example → Basics →Blink.
4. Select your Arduino board.
   i. Go to Tools → Board and select your board.
5. Select your serial port.
6. Upload the program to your board.
7. Note the contents of the registers/Ports in the observation table.

Sample Program: To test the LED ON/OFF as per switch position

Step 1: Algorithm

1. Make the circuit connections as per given diagram.
2. Connect LED anode to pin 13 of Arduino via current limiting resistor and cathode of LED to ground.
3. Connect one terminal of push button switch to digital pin 7 and other terminal to ground.

4. Connect the Arduino to the computer via USB cable.
5. Open the Arduino IDE and select the correct board and COM port.
6. Write the code and upload it to the Arduino.
7. Press the push button switch.
8. Check the status of LED as the push button is pressed and released.

STEP 2 PROGRAM:

Program	Comments

#define ledPin 13
#define switchPin 7
int val = 0;
	// choose the pin for the LED
	// choose the input pin (for a pushbutton)
	// variable for reading the pin status

void setup()
{
pinMode(ledPin, OUTPUT);
pinMode(switchPin, INPUT);
}
 	
void loop()
{
val = digitalRead(switchPin);
if (val == HIGH)
 	
{
digitalWrite(ledPin, LOW);
}
else
{
digitalWrite(ledPin, HIGH);
}
}
	
	// declare LED as output
	// declare pushbutton as input
	
	// read input value
	// check if the input is HIGH (button released)
	
	// turn LED OFF
	
	// turn LED ON

Problem statement 1 for student: Write a program to turn ON the LED and Turning OFF after a certain delay.

Program	Comments

XI Resources used /apparatus/equipment with specifications-:

S. No.	Instrument /Components	Specification	Quantity
1.			
2.			
3.			
4.			
5.			
6.			

XII Actual Procedure Followed (use blank sheet provided if space not sufficient)

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

XIII Observations for Student Activity program (use blank sheet provided if space not sufficient)

……

……

……

XIV Results (Output of the Program)

……

……

……

……

XV Interpretation of Results (Give meaning of the above obtained results)

……

……

……

……

XVI Conclusions and Recommendation (Actions/decisions to be taken based on the interpretation of results).

……

……

……

……

XVII Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO

a. Write the purpose of setup() and loop() function.
b. Give the significance of void loop() used in programming.
c. List various data types used in Arduino programming.

[Space for Answers]

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

XVIII References / Suggestions for further reading

a. https://www.arduino.cc/
b. https://www.vskills.in/certification/tutorial/interfacing-switch-and-led-with-arduino/
c. https://www.tutorialspoint.com/arduino/arduino_data_types.htm

XIX Assessment Scheme

The given performance indicators should serve as a guideline for assessment regarding process and product related marks:

Performance indicators		Weightage
Process related: 15 Marks		**60%(15)**
	Use of IDE tools for programming	20%
	Coding and Debugging ability	30%
	Follow ethical practices.	10%
Product related: 10 Marks		**40% (10)**
	Correctness of Program for student activity	20%
	Relevance of output of the problem definition	15%
	Timely Submission of report, Answer to sample questions	05%
	Total	**100 % (25)**

Marks Obtained			Dated signature of Teacher
Process Related (15)	**Product Related (10)**	**Total (25)**	

Practical No.8: Programs to perform arithmetic operations on Arduino.

I Practical Significance

This Practical will help the students to develop skills to write an Arduino based embedded C program. It reinforces understanding of basic math in real-time systems. It encourages logical thinking and problem-solving with constraints. It helps to bridge the gap between mathematical concepts and practical.

II Industry/Employer Expected Outcome

- Develop simple applications based on embedded system.

III Course Level Learning Outcome

- Develop the basic applications using Arduino.

IV Laboratory Learning Outcome

- Develop programs to perform arithmetic operation using math functions: constrain(), max(), min(), Pow(), sq(), sqrt() using Arduino.

V Relevant Affective Domain related outcomes

1. Demonstrate working as a leader or a team member.
2. Follow ethical practices.

VI Relevant Theoretical Background

Math Functions in Arduino

1. abs( )

Description: Calculates the absolute value of a number.

Syntax: abs(x)

Parameters: x: the number Returns: x: if x is greater than or equal to 0.

-(x): if x is less than 0.

2. constrain( )

Description : Constrains a number to be within a range.

Syntax : constrain(x, a, b)

Parameters: x: the number to constrain, a: the lower end of the range, b: the upper end of he range.

Allowed data types: all data types.

Limits a value between a minimum and maximum.

Use of the function: Prevent motor speeds or sensor values from going beyond safe limits.

Example: Limit temperature sensor reading to between 0°C and 100°C.

Returns: x: if x is between a and b. a: if x is less than a. b: if x is greater than b.

3. max( )

Description : Calculates the maximum of two numbers.

Syntax : max(x, y)

Parameters x: the first number. Allowed data types: any data type. y: the second number.

Allowed data types: any data type.

Returns : The larger of the two parameter values.

4. min( )

Description : Calculates the minimum of two numbers.

Syntax : min(x, y)

Parameters x: the first number.

Allowed data types: any data type. y: the second number.

Returns : The smaller of the two numbers.

max(a, b) and min(a, b), Finds the maximum or minimum of two numbers.

Use of the function: Decision making based on sensor thresholds. Example: Compare two temperature sensors and use the higher one for cooling system control.

5. sq( )

Description Calculates the square of a number: the number multiplied by itself.

Syntax: sq(x)

Parameters x: the number. Allowed data types: any data type.

Returns: The square of the number.

Use of the function: Speed or distance calculations, motion algorithms. Example: Compute squared error in PID control.

6. sqrt( )

Description Calculates the square root of a number.

Syntax: sqrt(x) Parameters x: the number.

Allowed data types: any data type.

Returns: the number's square root.

Data type: double.

Use of the function: Pythagorean theorem in navigation or distance calculation. Example: Compute the distance between two points using GPS or accelerometer data.

7. pow( )

Description Calculates the value of a number raised to a power. pow( ) can be used to raise a number to a fractional power. This is useful for generating exponential mapping of values or curves.

Syntax: pow(base, exponent)

Parameters base: the number.

Allowed data types: float. exponent: the power to which the base is raised.

Returns The result of the exponentiation.

Data type: double.

Use of the function: Calculations in physics-based simulations or energy computations.

Example: Compute energy as $E=m \cdot v^2$ (which involves powers).

VII Actual Circuit diagram used in laboratory with related equipment rating-

A) Sample Setup Diagram-

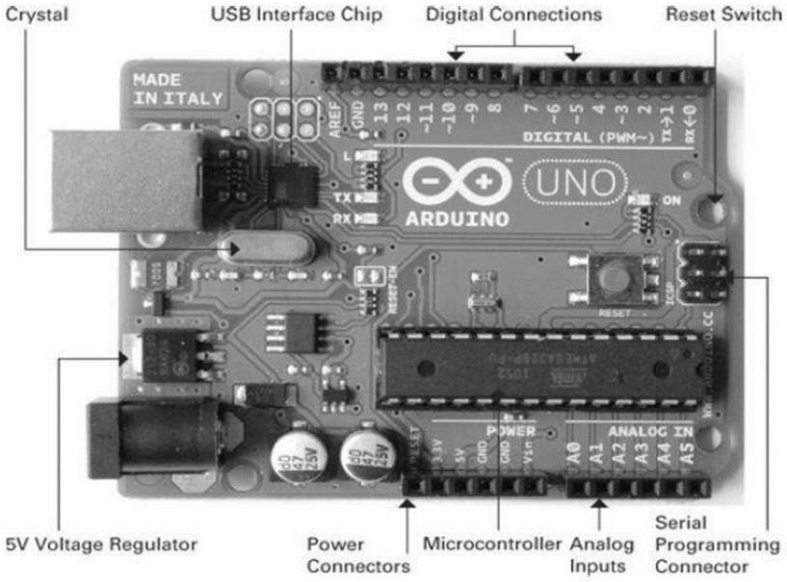


Figure 8.1 Arduino uno ATmega328P

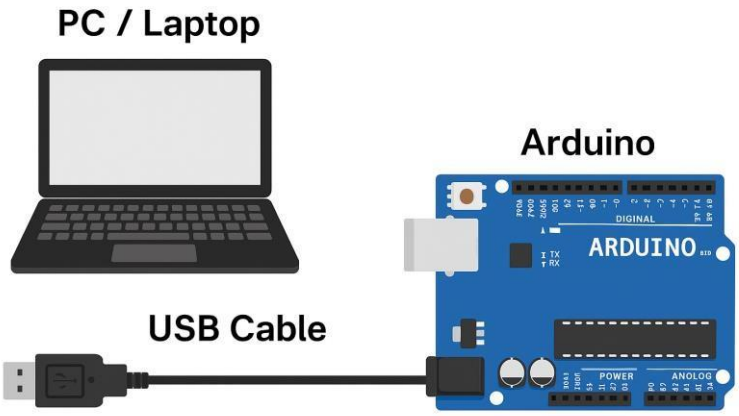


Figure 8.2 Interfacing Arduino with PC

B) Actual Setup Diagram used in Laboratory-

VIII Required Resources/apparatus/equipment with specifications-

Sr. No.	Name of Resource	Suggested Broad Specification	Quantity
1	Desktop PC	Core i3/i5 Processor, 32/64-bit windows operating system, 2GB/Higher RAM	1
2	Microcontroller Board	Arduino Uno / ATmega328P / STM32 / PIC etc.	1
3	Arduino IDE-	Latest version (from Arduino.cc)	1
4	USB Cable	USB A to B / USB to micro-USB	1
5	USB-to-Serial Converter (if needed)	FT232 / CP2102 / CH340G	1

IX Precautions to be followed

1. Check the rules/syntax of Arduino programming.
2. Check the circuit connections before power ON.
3. Avoid short circuits by checking wiring before powering the board

X Procedure

1. Install Arduino IDE on PC.
2. Make the circuit as given in connection diagram
3. Open the project.
    - To create a new project, select File → New
    - To open an existing project, select File.
4. Select used Arduino board
    - Go to Tools → Board and select your board
5. Select used serial port
6. Upload the program on connected board
7. Note the contents of the registers/Ports in the observation table.

Sample Program-

Perform basic arithmetic operations using Arduino mathematical functions: constrain(), max(), min(), pow(), sq() and sqrt().

STEP1: Algorithm-

1. Start the program.
2. Initialize serial communication at 9600 baud rate using Serial.begin(9600)
3. Declare and Assign Input Values
   - Declare two floating-point variables num1 and num2.
   - Assign values to them (e.g., num1 = 9.0, num2 = 4.0).
4. Apply constrain() Function
   - Use constrain(num1, 1.0, 10.0) to limit num1 to the range 1.0 to 10.0
   - Use constrain(num2, 1.0, 10.0) to limit num2 to the range 1.0 to 10.0
5. Calculate Maximum and Minimum
   - Use max(num1, num2) to find the maximum of the two constrained numbers.
   - Use min(num1, num2) to find the minimum of the two constrained numbers.
6. Calculate Power
   - Use pow(num1, num2) to compute num1 raised to the power of num2.
7. Calculate Squares
   - Use sq(num1) to calculate the square of num1.
   - Use sq(num2) to calculate the square of num2.
8. Calculate Square Roots
   - Use sqrt(num1) to calculate the square root of num1.
   - Use sqrt(num2) to calculate the square root of num2.
9. Display the original and computed values using Serial.print() on the Serial Monitor.
10. Stop Execution.

STEP2: Program-

Program	Comments

void setup() {
Serial.begin(9600);
delay(1000);
	// Initialize Serial Communication
	// Allow time for Serial Monitor to open

float num1 = 9.0; float num2 = 4.0;	// Declare and Assign Input Values

// Apply constrain() Function
float cNum1 = constrain(num1, 1.0, 10.0);
float cNum2 = constrain(num2, 1.0, 10.0);
	// Constrain num1 between 1.0 and 10.0
	// Constrain num2 between 1.0 and 10.0

// Calculate Maximum and Minimum
float maxValue = max(cNum1, cNum2);
float minValue = min(cNum1, cNum2);
	// Calculate Maximum and Minimum

// Calculate Power float powerResult = pow(cNum1, cNum2);	// Calculate Power

// Calculate Square
float square1 = sq(cNum1);
float square2 = sq(cNum2);
	// Calculate Square of num1 and num2

// Calculate Square Root
float sqrt1 = sqrt(cNum1);
float sqrt2 = sqrt(cNum2);
	// Calculate Square Root

// Display All Results on the Serial Monitor
Serial.println("-------Arithmetic Results using Math Functions ----- ");
 	
Serial.print("Original num1: ");
Serial.println(num1);
Serial.print("Original num2: ");
Serial.println(num2);
Serial.print("Constrained num1 (1.0 - 10.0): ");
Serial.println(cNum1);
Serial.print("Constrained num2 (1.0 - 10.0): ");
Serial.println(cNum2);
Serial.print("Maximum Value: ");
Serial.println(maxValue);
Serial.print("Minimum Value: ");
Serial.println(minValue);
Serial.print("Power (num1 ^ num2): ");
Serial.println(powerResult);
	// Display All Results on the Serial Monitor

Program	Comments
Serial.print("Square of num1: "); Serial.println(square1); Serial.print("Square of num2: "); Serial.println(square2); Serial.print("Square Root of num1: "); Serial.println(sqrt1); Serial.print("Square Root of num2: "); Serial.println(sqrt2); Serial.println("----------"); }	

void loop()
{
// Stop Execution — do nothing
}
	// Empty loop - program runs only once

Output window: (Serial Monitor window)

Figure 8.3 Editor window of the program in Arduino IDE.

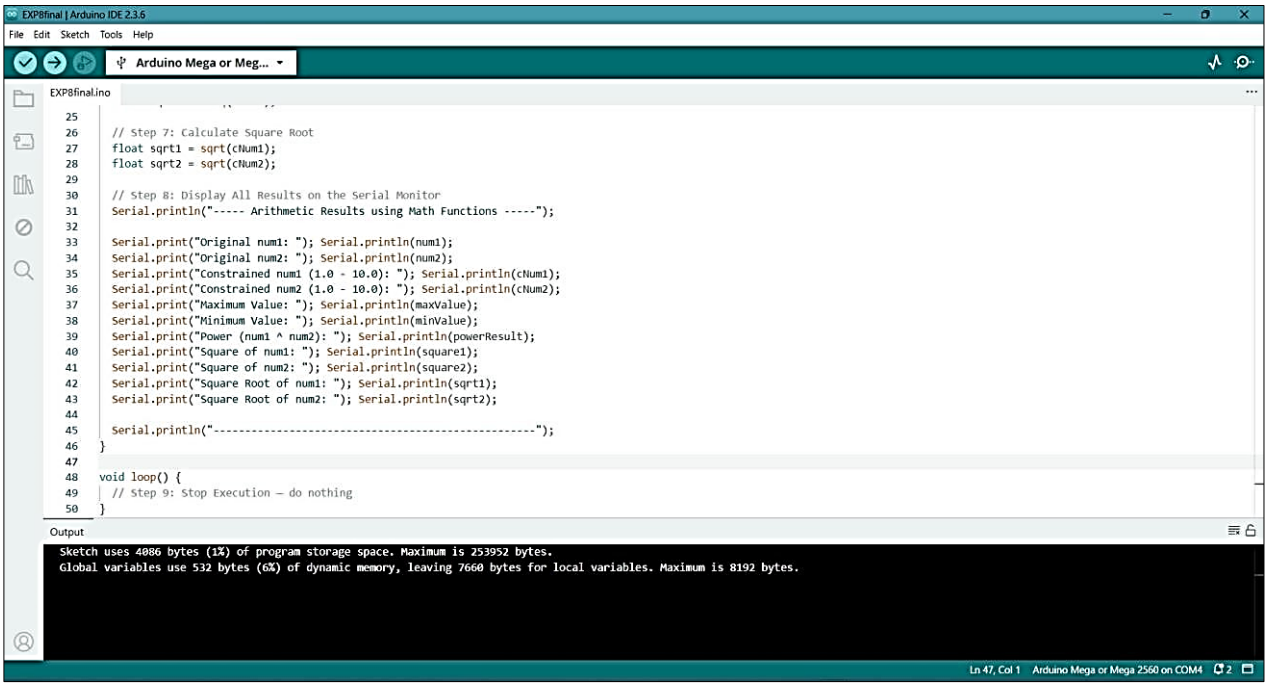


Figure 8.4 Editor window of the program in Arduino IDE.

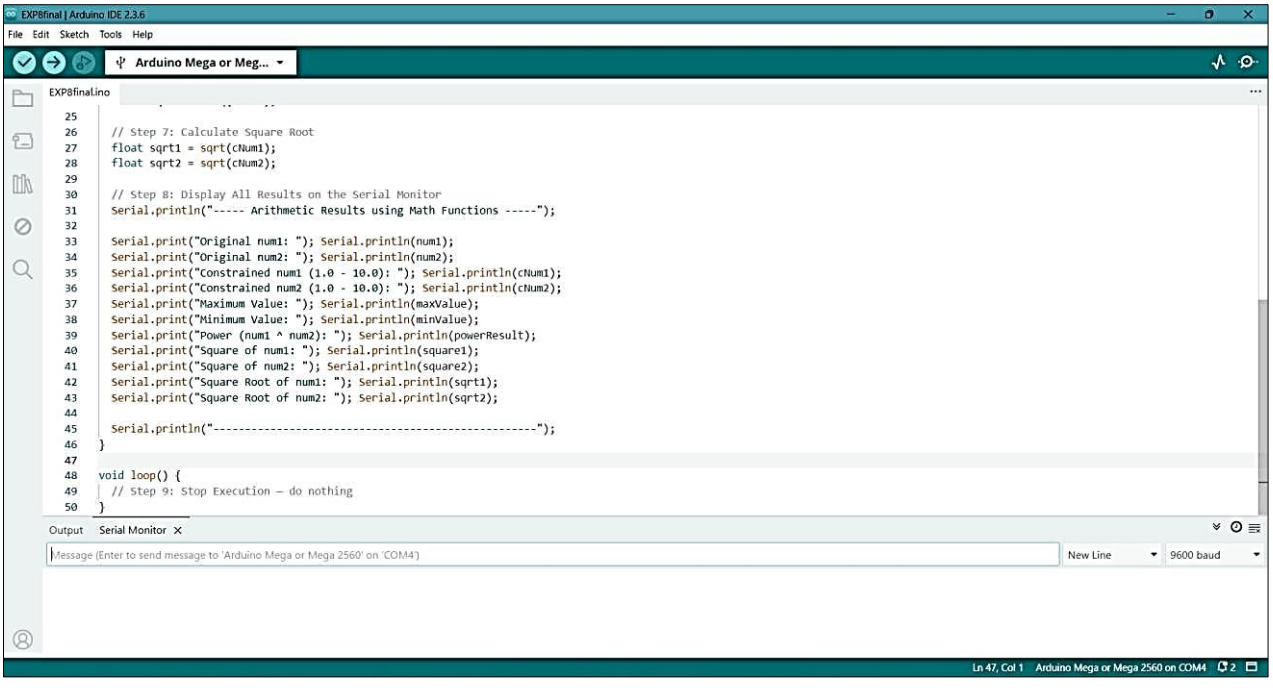


Figure 8.5 Editor window of the program in Arduino IDE.

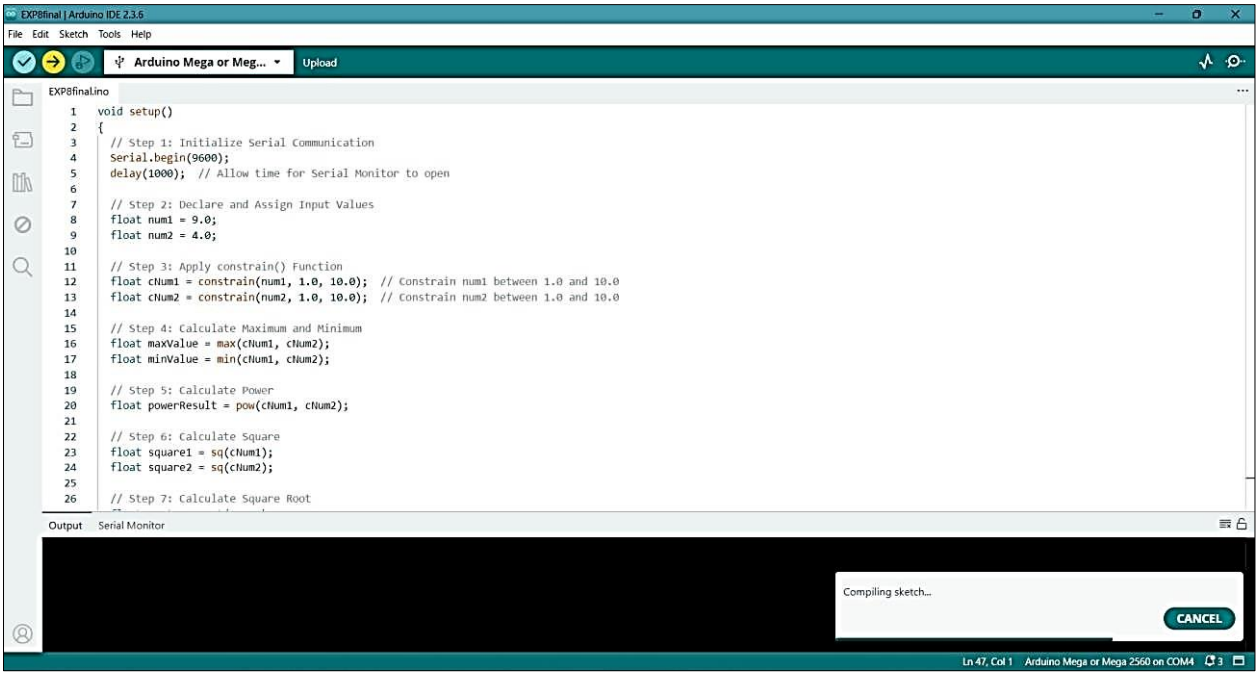


Figure 8.6 Editor window of the program- Compiling and uploading the program.

Figure 8.7 Output window of the program- Uploading the program on Arduino.

Output on serial monitor:

----- Arithmetic Results using Math Functions -----
Original num1: 9.00
Original num2: 4.00

Constrained num1 (1.0 - 10.0): 9.00
Constrained num2 (1.0 - 10.0): 4.00

Maximum Value: 9.00
Minimum Value: 4.00

Power(num1^num2):6561:00

Square of num1: 81.00
Square of num2: 16.00

Square Root of num1: 3.00
Square Root of num2: 2.00

Problem statement for student:

Design and implement an Arduino program to perform basic arithmetic operations. The program should accept two user inputs (via serial monitor or sensors) for the operations. Calculate the maximum and minimum of the two inputs using max() and min(). The power of one input raised to the other using pow(). The square and square root of each input using sq() and sqrt(). Display the results on the serial monitor.

Step 3- Program

Program	**Comments**

XI Resources used

Sr. No.	Name of Resource	Specification	Quantity
1			
2			
3			
4			
5			

XII Actual Procedure Followed (use blank sheet provided if space not sufficient)

……..……

……..……

...……

……..……

……..……

……..……

XIII Observation Table for the program for students (use blank sheet provided if space not sufficient)

Table 1-

Sr. No.	Num 1	Num 2	Function	Output Value for the specified function
1			max( )	
2			min( )	
3			pow( )	
4			sq( )	
5			sqrt( )	

XIV Results (Output of the program)

……..……

..……..……

...……

……..……

XV Interpretation of results (Give meaning of the above obtained results)

……

……

……

XVI Conclusion and recommendation (Actions/decisions to be taken based on the interpretation of results)

……

……

……

XVII Practical related questions

Note: Below given are a few sample questions for reference. Teachers must design more such questions so as to ensure the achievement of identifying CO.

1. List the practical examples of math functions max ( ), min ( ) in use?
2. Explain the condition if the initial values of num1 and num2 are outside the range specified in constrain()?
3. If num1 = -4.0 and num2 = -2.0, write the result after applying constrain(num1, 1.0, 10.0) and constrain(num2, 1.0, 10.0)?
4. **If num1 = 3.0 and num2 = 5.0, what are the values of:**
   - constrain(num1, 1.0, 10.0)

- max(num1, num2)
- pow(num1, num2)
- sqrt(num1)

[Space for Answers]

……

……

……

XVIII References/Suggestions for further reading

1. https://youtu.be/ODKOfLL7sB4

2. https://www.arduino.cc/

3. https://youtu.be/vzCNfBux1h0

4. https://youtu.be/eciWL2_SDsI

5. https://www.youtube.com/watch?v=CysNp724Ldg

6. https://youtu.be/SX8z3-BEuWQ

7. https://youtu.be/ynAvySNCi-0

XIX Assessment Scheme

Performance Indicators		Weightage
Process Related : 15 Marks		**60 %**
1	Coding and Debugging ability	30%
2	Making connections of hardware	20%
3	Follow ethical practices	10%
Product Related: 10 Marks		**40%**
4	Correct program of student activity	20%
5	Relevance of output of the problem definition	10%
6	Conclusion and Practical related questions	10%
Total : 25 Marks		**100 %**

Marks Obtained			Dated signature of Teacher
Process Related (15)	**Product Related (10)**	**Total (25)**	

Practical No.9: LCD Interfacing to Arduino board.

I Practical Significance

In traditional Arduino LCD display projects, there are often limitations in terms of available pins, especially when using Arduino Uno. Additionally, wiring and connections can become quite complex. To address these issues, an I2C 16x2 Arduino LCD display is introduced, which utilizes the I2C communication interface. This means that it only requires 4 pins to connect the LCD display, including VCC, GND, SDA, and SCL. By adopting the I2C interface, we can save at least 4 digital/analog pins on the Arduino, making the project's connections simpler and more convenient.

II Industry/Employer Expected Outcome

- Develop simple applications based on embedded system.

III Course Level Learning Outcome

- Develop the basic applications using Arduino.

IV Laboratory Learning Outcome

- Interface two 16 x 2 LCD modules with Arduino using I2C serial communication protocol.

V Relevant Affective Domain related outcomes

1. Demonstrate working as a leader or a team member.
2. Follow ethical practices.

VI Relevant Theoretical Background

LCD stands for **Liquid Crystal Display.** LCD is a flat-paneled display**.** It uses liquid crystals combined with polarized to display the content. LCD uses the light modulation property of LCD. LCD is available both in Monochrome and Multicolor. It cannot emit light directly without a backlight. In some LCDs, it displays the content only with the help of a backlight in a dark place.

The LCDs have a parallel interface, meaning that the microcontroller has to manipulate several interface pins at once to control the display. The interface consists of the following pins:

A **Register Select (RS)** pin that controls the transmitting command code for LCD initialization or LCD data to display. The data register can be selected, which holds what goes on the screen, or an instruction register, which is where the LCD's controller looks for instructions on what to do next. If RS=0, it means that the command register is selected and of RS=1, then the data register is selected.

A **Read/Write (R/W)** pin that selects reading mode or writing mode. RW=0; writes to the LCD. RW=1; Reads from the LCD.

An **Enable** pin that enables writing to the registers. It is one of the control pins used to interface with microcontrollers and other devices. It is falling edge triggered

8 data pins (D0 -D7)- The states of these pins (high or low) are the bits that you're writing to a register when you write, or the values you're reading when you read.

There's also a display contrast pin (VEE), power supply and ground pins (+5V and GND)

and LED Backlight (Bklt+ and BKlt-) pins that you can use to power the LCD, control the display contrast, and turn on and off the LED backlight, respectively.

The process of controlling the display involves putting the data that form the image of what to display into the data registers, then putting instructions in the instruction register.

I2C or IIC stands for Inter-Integrated Communication. I2C is a serial communication interface to communicate with other I2C devices. I2C uses a multi-master / multi slave method. I2C uses 2 lines named **SCL** and **SDA** for transmission/reception and another 2 lines for power supply and ground. Each and every I2C device has an I2C address to identify. I2C addresses of multiple devices may have the same address. The address is in the format of "0x20" (Example address).

- The serial Clock (SCL) pin is to synchronize the transmitter and receiver.
- Serial Data (SDA) pin is to transfer data.

I2C LCD uses I2C communication interface to transfer the information required to display the content. The I2C LCD requires only 2 lines (SDA and SCL) for transferring the data. So, the complexity of the circuit is reduced.

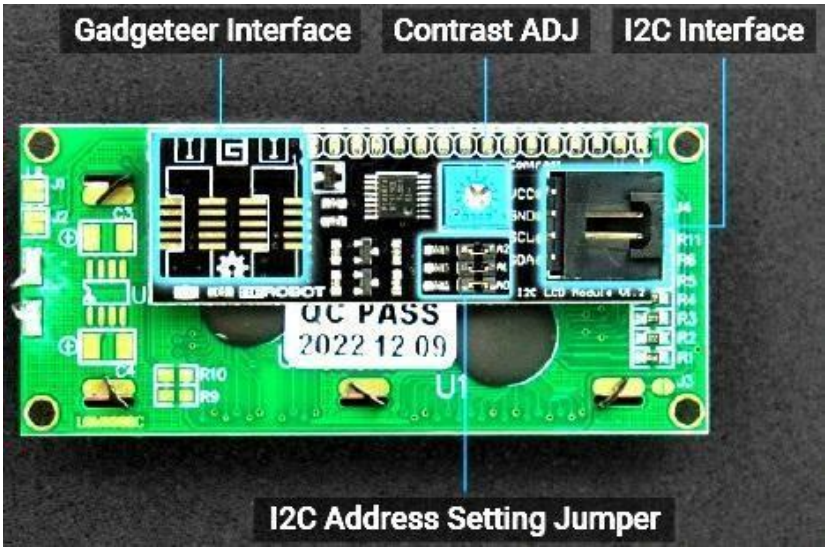


Figure 9.1 I2C communication interface

Arduino is an open-source electronics platform based on easy-to-use hardware and software. These are able to read inputs – light on a sensor, a finger on a button, or a Twitter message – and turn it into an output – activating a motor, turning on an LED, publishing something online. Arduino designs use a variety of microprocessors and controllers. The boards are equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (shields) and other circuits. The boards feature serial communications interfaces, including Universal Serial Bus (USB) on some models, which are also used for loading programs from personal computers.

VII Actual Circuit diagram used in laboratory with related equipment rating-

A) Sample Setup Diagram-

I2C LCD can be connected to the Arduino directly with SDA pin to SDA pin and SCL pin to SCL pin as per the circuit diagram given below.

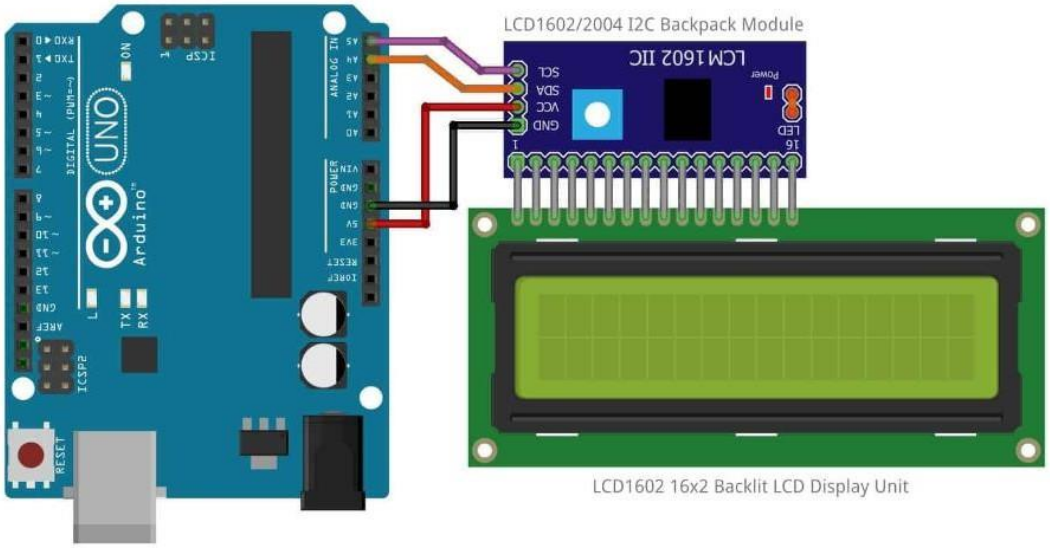


Figure 9.2 Interfacing I2C LCD with Arduino

B) Actual Setup Diagram used in Laboratory -

VIII Required Resources/apparatus/equipment with specifications-

Sr. No.	Name of Resource	Suggested Broad Specification	Quantity
1	Desktop PC	Core i3/i5 Processor, 32/64-bit windows operating system, 2GB/Higher RAM With USB port	1
2	Microcontroller Board	Arduino Uno / ATmega328P / STM32 / PIC etc.	1
3	USB Cable	USB A to B / USB to micro-USB	1
4	USB-to-Serial Converter (if needed)	FT232 / CP2102 / CH340G	1
5	Arduino IDE	Latest version (from Arduino.cc)	1
6	I2C LCD	16x2 LCD and I2C communication interface (PCF8574/ MCP23008)	1 each
7	Connecting wires	-	As required

IX Precautions to be followed

1. Check the rules/syntax of Arduino programming.
2. Check the circuit connections before power ON.
3. Avoid short circuits by checking wiring before powering the board

X Procedure

1. Connect the Arduino board to the PC using a USB cable.
2. Install the "LiquidCrystal I2C"library for LCD display in Arduino IDE.
3. Import "LiquidCrystal_I2C.h" header file in the code.
4. Connect the display device to Arduino.
   - Connect the SDA pin of an LCD display to the SDA pin of the Arduino.
   - Connect the SCL pin of an LCD display to the SCL of the Arduino.
   - Connect VCC to 5V pin
   - Connect GND to GND pin.
5. Find and put the I2C Address of the display device.
6. Run the code to display the data on the LCD.
7. Write and upload a code to the microcontroller.
8. Above procedure can be done in any simulation software like Tinkercad, Proteus etc. And then hardware connection can be done as shown in the connection diagram.

Sample Program- Write an Arduino program to interface a 16x2 I2C LCD with Arduino and display the text "MSBTE" on the LCD.

STEP1: Algorithm-

1. Include the LCD library to communicate with an I2C-enabled LCD.
2. Create an LCD object with the I2C address (32), 16 columns, and 2 rows.
3. In the setup() function:
   - Initialize the LCD.
   - Turn on the LCD backlight.
4. In the loop() function:
   - Set the cursor to the beginning (0,0).

- Print "MSBTE" on the LCD.
- Add a small delay for simulation stability.

STEP2: Program-

Program	Comments

#include <LiquidCrystal_I2C.h>
 	
LiquidCrystal_I2C lcd_1(32, 16, 2);
 	
void setup()
{
  lcd_1.init();
  lcd_1.backlight();
}
 	
void loop()
{
  lcd_1.setCursor(0, 0);
 	
  lcd_1.print("MSBTE");
  delay(1000);
  lcd_1.clear();
}
	
	// Initialize the LCD
	// Turn on the LCD backlight
	
	// Set cursor position to top-left corner (row 0,
	// column 0)
	
	// Wait for 1000 millisecond(s)
	// Clear the screen

Output window:

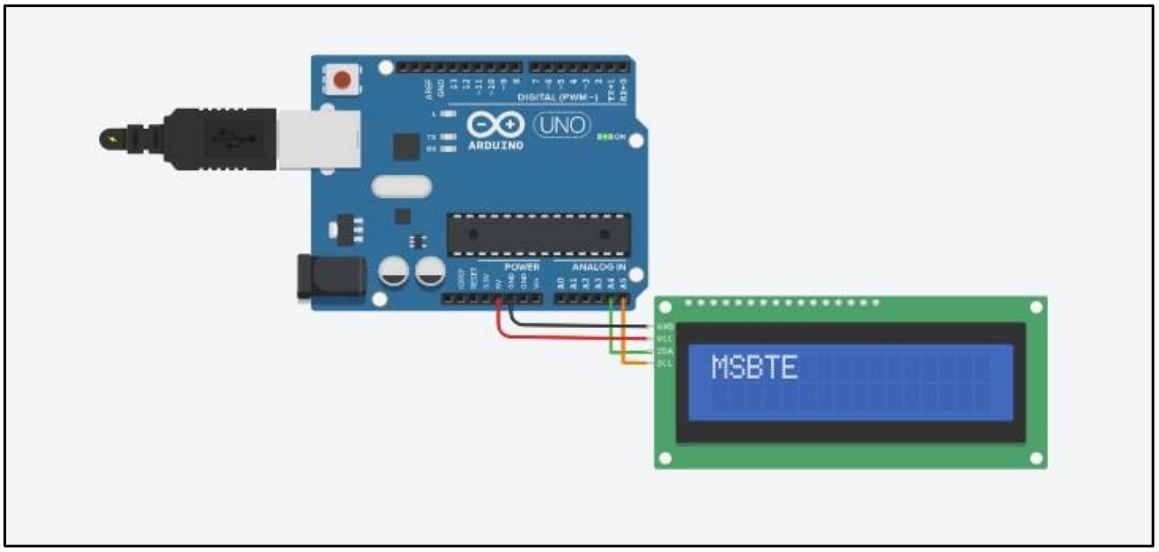


Figure 9.3 Output window-4 of the program displaying “MSBTE” on LCD

Problem statement for student: Design an Arduino-based system to interface two I2C 16x2 LCD displays and control their output using a digital input pin (pin 2). The system should display different messages on the two LCDs based on the state of a switch or digital input.

Program

Program	Comments

XI Resources used

Sr. No.	Name of Resource	Specification	Quantity
1			
2			
3			
4			
5			
6			
7			

XII Actual Procedure (use blank sheet provided if space not sufficient)

……..

……

...……

……..

……..

……..

……..

……..

XIII Observation Table for the program for students (use blank sheet provided if space not sufficient)

Table 1-

Sr. No.	Digital input condition	Message displayed on LCD-1	Message displayed on LCD-2
1	0 (LOW)		
2	1 (High)		

XIV Results (Output of the program)

……..

……

...……

……..

……..

……

XV Interpretation of results (Give meaning of the above obtained results)

……..

……

...……

……..

XVI Conclusion and recommendation (Actions/decisions to be taken based on the interpretation of results)

……..

……

...……

……..

……..

XVII Practical related questions

Note: Below given are a few sample questions for reference. Teachers must design more such questions so as to ensure the achievement of identifying CO.

1. Explain the purpose of using the LiquidCrystal_I2C library in this program?
2. Explain the significance of two different I2C addresses (39 and 32) used for the LCDs?
3. If two LCDs connected on the same I2C bus, then explain how the Arduino differentiate between them?
4. If the LCD doesn't display anything, then explain the steps to troubleshoot it?
5. Draw the interfacing diagram of LCD with Arduino without using I2C means in the traditional way.

[Space for Answers]

……..
……
..……
……..
……..
……
……..
……
..……
……..
……..
……
……..
……
..……
……..
……..
……
……..
……
..……
……..
……..
……
……..
……
..……

……..

……..

……

……..

……

..……

……..

……..

……

……..

……..

……

..……

……..

……..

……

……..

……

..……

……..

……..

……

……..

……

..……

……..

……..

……

XVIII References/Suggestions for further reading

1. https://www.arduino.cc/

2. https://youtu.be/SX8z3-BEuWQ

3. https://youtu.be/SHORUnKgpKE

4. https://youtu.be/wEbGhYjn4QI

5. https://www.geeksforgeeks.org/electronics-engineering/how-to-interface-i2c-lcd-display-with-arduino/

6. https://youtu.be/-7ZahqXoYf4?si=bhN7Si5BuVtDFg9m

7. https://youtu.be/sSHUWcgFw90

XIX Assessment Scheme

Performance Indicators		Weightage
Process Related : 15 Marks		**60 %**
1	Coding and Debugging ability	30%
2	Making connections of hardware	20%
3	Follow ethical practices	10%
Product Related: 10 Marks		**40%**
4	Correct program of student activity	20%
5	Relevance of output of the problem definition	10%
6	Conclusion and Practical related questions	10%
Total : 25 Marks		**100 %**

Marks Obtained			Dated signature of Teacher
Process Related (15)	**Product Related (10)**	**Total (25)**	

Practical No. 10: Temperature Sensor Interfacing to Arduino Board

I Practical Significance

Interfacing a temperature sensor like LM35 with an Arduino board develops skills in reading analog sensor data, converting it into temperature values, and displaying it using a serial monitor or LCD. The practical demonstrates real-time data acquisition and basic embedded programming. It serves as a foundation for IoT and automation applications.

II Industry/Employer expected outcome

Develop simple applications based on embedded system.

III Course Level Learning Outcome(s)

- Develop the basic applications using Arduino.

IV Laboratory Learning Outcome(s)

- Develop program to read the data from the temperature sensor through Arduino and display on LCD.

V Relevant Affective domain related Outcome(s)

1. Demonstrate working as a leader or a team member.
2. Follow ethical practices.

VI Relevant Theoretical Background

A temperature sensor, like LM35 or other equivalent senor, is an analog device which produces a voltage output proportional to the temperature. The LM35 generates output of 10 mV per °C, enabling to convert voltage readings to temperature values. When interfaced with an Arduino board, the analog output of the sensor is read using the Analog-to-Digital Converter (ADC) which is built in the Arduino.
The Arduino then processes this data and based on the sensor's characteristics converts it into temperature values using simple arithmetic calculations. This setup is commonly used in various applications like climate monitoring, smart agriculture, and industrial automation.

To convert the voltage to temperature:

As the LM35 produces 10 mV (or 0.01 V) for every degree Celsius , to calculate the temperature, divide the voltage by 0.01:

Temperature (°C) = Vout / 0.01

Temperature (°C) = Vout × 100

For example suppose analogRead() function returns the value 100. Then to find the temperature:

1. First, calculate the voltage: Vout = 100 × (5 / 1024) = 0.48828125 V
2. Now convert that voltage to temperature: Temperature = 0.488 × 100 = 48.8°C

Therefore, temperature can be calculated by using formula:

Temperature = analogRead value x [(5 / 1024) x 100] = **analogRead value x 0.48828125**

VII Actual Circuit diagram used in laboratory with related equipment rating-

A) Sample Circuit Diagram-

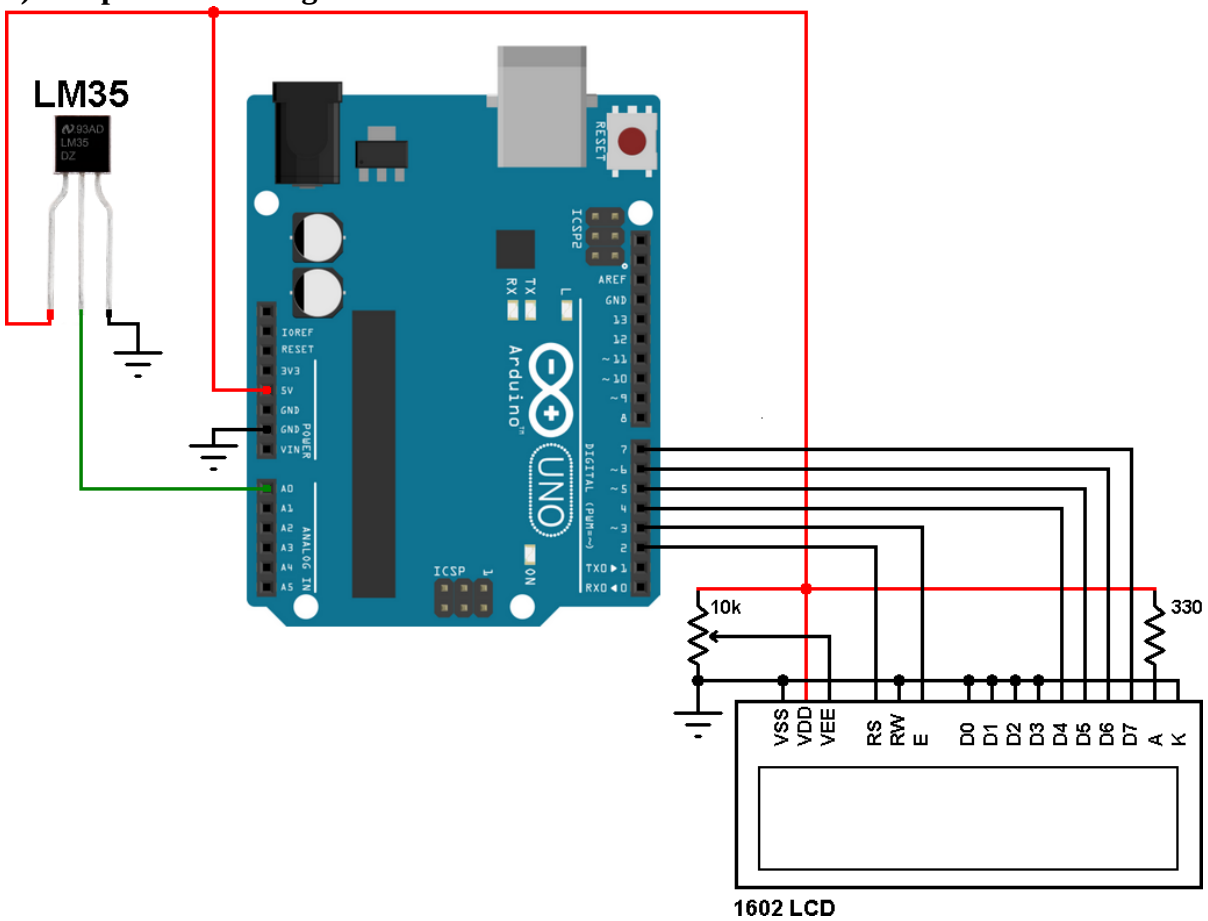


Fig 10.1 Experimental Set up

B) Actual Circuit Diagram used in Laboratory

VIII Required Resources/ apparatus/equipment with specifications

Sr. No.	Instrument /Components	Suggested Broad Specification	Quantity
1.	Desktop PC	Loaded with Arduino open-source IDE, simulation and program downloading software	1 No.
2.	Arduino Uno	Microcontroller: ATmega328P Operating Voltage: 5V Input Voltage: 7-12V Inout Voltage (limit): 6-20V Digital I/O Pins: 14 (6 provide PWM output) PWM Digital I/O Pins: 6 Analog Input Pins: 6 Flash Memory: 32 KB (ATmega328P) SRAM: 2 KB (ATmega328P) EEPROM: 1 KB (ATmega328P) Clock Speed: 16 MHz, LED_BUILTIN: 13	1 No.
3.	USB cable for Arduino Board	• Arduino Uno, Mega, Leonardo → USB Type-B cable • Arduino Nano (older models) → Mini-USB cable • Arduino Nano (newer models), Micro, Due, MKR series → Micro-USB or USB-C cable	1 No
4.	Breadboard	-	1 No.
5.	LM 35 Temperature Sensor	1. Output Voltage: 10 mV per °C 2. Temperature Range: 3. LM35: 0°C to +100°C (standard) 4. LM35A: −55°C to +150°C (extended range) 5. Operating Voltage: 4V to 30V DC 6. Accuracy: ±0.5°C at room temperature 7. Current Consumption: Less than 60 µA 8. Low Self-Heating: Less than 0.1°C in still air 9. Output Type: Analog (linear voltage output) 10. Package Types: TO-92, SO-8, TO-220	1 No.
6.	LCD 16 x 2	1. Display Type: Alphanumeric LCD 2. Characters Displayed: 16 characters per line, 2 lines (total 32 characters) 3. Interface: Parallel (typically 4-bit or 8-bit mode using HD44780 controller) 4. Operating Voltage: 4.7V to 5.3V DC 5. Backlight: Usually LED, available in various colors (e.g., blue, green) 6. Character Size: Approximately 5x8 dot matrix 7. Current Consumption: Around 1 mA (without backlight), 10–20 mA (with backlight)	1 No.

Sr. No.	Instrument /Components	Suggested Broad Specification	Quantity
7.	Connecting Wires	-	As required

IX Precautions to be Followed

1. Check rules /syntax of Arduino programming.
2. Check the circuit connections before power ON.
3. Avoid short circuits by checking wiring before powering the board

X Procedure

a. Launch Arduino IDE.
b. Make the circuit as given in connection diagram
c. Open your first project.
   i. To create a new project, select File → New.
   ii. To open an existing project, select File → Example → Basics →Blink.
d. Select your Arduino board.
   i. Go to Tools → Board and select your board.
e. Select your serial port.
f. Upload the program to your board.
g. Note the contents of the registers/Ports in the observation table.

Sample Program:

STEP 1: Algorithm:

1. Make the LM35 Sensor connections as per given steps
   a) VCC (left pin) → Arduino 5V
   b) GND (right pin) → Arduino GND
   c) Output (middle pin) → Arduino Analog Pin A0
2. Connect 16x2 LCD (Parallel Interface using HD44780 Controller) as per given steps
   a) VSS → GND
   b) VDD → 5V
   c) VEE → Middle pin of 10k potentiometer (contrast adjustment), other ends to 5V and GND
   d) RS → Arduino Digital Pin 7
   e) RW → GND (write mode)
   f) E → Arduino Digital Pin 6
   g) D4 → Arduino Digital Pin 5
   h) D5 → Arduino Digital Pin 4
   i) D6 → Arduino Digital Pin 3
   j) D7 → Arduino Digital Pin 2
   k) A (LED Anode) → 5V through a 330Ω resistor
   l) K (LED Cathode) → GND
3. Write the Arduino Code for interfacing of temperature sensor LM35 and display on LCD.
4. Compile the Arduino code (sketch)
5. Connect via USB or external 9V power supply
6. Upload code and read analog voltage from LM35 on A0
7. Convert it to temperature (in °C) using the formula: Temp = (analogValue * 5.0 * 100) / 1024

8. Observe the output and display the temperature on the 16x2 LCD

- **STEP 2: Sample Program**: Develop program to read the data from the temperature sensor through Arduino and display on LCD.

Program	Comment
#include <LiquidCrystal.h> LiquidCrystal lcd(13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3); float temperature; int ADC_value int tempPin =A0;	// 8-bit mode

void setup()
{
lcd.begin(16, 2);
 	
lcd.clear();
 	
}
void loop()
{
lcd.setCursor(0,0);
lcd.print("TEMPERATURE :");
ADC_value = analogRead(tempPin);
 	
lcd.print(temperature);
lcd.print("ºC");
delay(1000);
}
	// set up the LCD's number of columns and rows:
	// clears the LCD screen and positions the cursor in the upper-left corner.
	
	// Set cursor to column 0 row
	// Print a message to the LCD.
	
	// read analog volt from sensor and save to variable temp
	temperature = ADC_value * 0.48828125;
	and convert the analog volt to temperature
	
	// display temperature value
	
	// update sensor reading each one second

Problem statement 1 for student: Write a program to read the data from DHT11 Temperature and Humidity sensor and display on serial monitor.

Program	Comment

XI Resources used /apparatus/equipment with specifications:

S. No.	Instrument /Components	Specification	Quantity
1.			
2.			
3.			
4.			
5.			
6.			
7.			

XII Actual Procedure Followed (use blank sheet provided if space not sufficient)

……

……

……

……

……

……

……

……

……

……

……

……

……

XIII Observations for Student Activity program (use blank sheet provided if space not sufficient)

……

……

……

XIV Results (Output of the Program)

……

……

……

……

XV Interpretation of Results (Give meaning of the above obtained results)

……

……

……

……

XVI Conclusions and Recommendation (Actions/decisions to be taken based on the interpretation of results).

……

……

……

……

XVII Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO

a. State the significance of baud rate while displaying data on serial monitor.
b. Give the formula to convert LM35 analog output to temperature in °C.
c. Give the function of the Analog-to-Digital Converter (ADC) in Arduino.

[Space for Answers]

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

XVIII References / Suggestions for further reading

a. https://www.arduino.cc/
b. https://www.hackatronic.com/arduino-with-temperature-sensor-interfacing-lcd-and-lm35/
c. https://projecthub.arduino.cc/ejshea/displaying-temperature-and-humidity-on-an-lcd-91bc36
d. https://projecthub.arduino.cc/onatto22/dht11-humidity-temperature-sensor-with-16x2-lcd-display-7bf46c

XIX Assessment Scheme

The given performance indicators should serve as a guideline for assessment regarding process and product related marks:

Performance indicators		Weightage
Process related: 15 Marks		**60%**
1	Use of IDE tools for programming	20%
2	Coding and Debugging ability	30%
3	Follow ethical practices.	10%
Product related: 10 Marks		**40%**
4	Correctness of program for student activity	20%
5	Relevance of output of the problem definition	15%
6	Timely Submission of report, Answer to sample questions	05%
	Total: 25 Marks	**100 %**

Marks Obtained			Dated signature of Teacher
Process Related (15)	**Product Related (10)**	**Total (25)**	