Department of Electrical Engineering

College of Engineering

Trivandrum

Lab Manual

Microprocessors and Embedded Systems Lab

(2024 scheme)

Department of Electrical Engineering

College of Engineering Trivandrum

Prepared By	Verified By	Approved By
Dr. Lekshmi Mohan	**Prof. Vishal M J**	**HOD**

VISION

National Level Excellence and International Visibility in Every Facet of Engineering Research and Education.

MISSION

To facilitate quality transformative education in Engineering and Management.

To foster innovations in Technology and its application for meeting global challenges. To pursue and disseminate Quality Research. To equip, enrich and transform students to be responsible professionals for better service to humanity.

DEPARTMENT OF ELECTRICAL ENGINEERING

VISION

Be a centre of excellence and higher learning in Electrical Engineering and allied areas.

MISSION

To impart quality education in Electrical Engineering and bring-up professionally competent engineers.

To mould ethically sound and socially responsible Electrical Engineers with leadership qualities.

To inculcate research attitude among students and encourage them to pursue higher studies.

Program Outcomes

PO1	Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems
PO2	Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
PO3	Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations
PO4	Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions
PO5	Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
PO6	The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
PO7	Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development
PO8	Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO9	Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings
PO10	Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
PO11	Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments
PO12	Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change

Program Specific Outcomes

PSO1	Apply engineering knowledge to analyse, model, design and operate modern systems for generation, transmission, distribution and control of electrical power.
PSO2	Design, develop and test modern hardware and software systems for signal processing, measurement, instrumentation and control applications.

Course Objectives

1. Achieve proficiency in 8085 microprocessor assembly language programming
2. Acquire practical experience with Arduino.

Course Outcomes (COs)

At the end of the course students should be able to:

Course Outcome		**Bloom's Knowledge Level (KL)**
CO1	Develop and execute assembly language programs for solving arithmetic and logical problems using microprocessor	K4
CO2	Design and Implement systems with interfacing circuits for various applications	K4
CO3	Execute projects as a team using microprocessor for real life applications.	K3
CO4	Design an Arduino based system with the help of various interfacing devices	K6

Note: K1-Remember, K2-Understand, K3-Apply, K4-Analyse, K5-Evaluate, K6-Create

CO-PO Mapping (Mapping of Course Outcomes with Program Outcomes)

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2
CO1	3	3	2	3	2			3	2	3	3	2		3
CO2	3	3	2	3	2			3	2	3	3	2		2
CO3	3	3	2	2	2	3		3	3	3	3	2		2
CO4	3	3	3	3	3							3		

1:Slight (Low), 2:Moderate(Medium), 3:Substantial (High),-:No Correlation

List of Experiments

Exp. No	Title
1.	Study of Internal Architecture of 8085 Microprocessor and Pin diagram
2.	Data Transfer using Different Addressing Modes and Block Transfer
3.	Arithmetic Operations in Binary and BCD: Addition and Subtraction
4.	Arithmetic Operations: Multiplication and Division
5.	Binary to BCD Conversion and BCD to Binary Conversion
6.	Logical Operations
7.	Digital I/O using PP1-Square Wave Generation
8.	Interfacing D/A Converter : Generation of Simple Waveforms- Triangular, Ramp
9.	Familiarization of Arduino IDE
10.	Blinking Internal LED of Arduino UNO Module
11.	Arduino Based Voltage Measurement
12.	Temperature and Humidity Monitoring over ThingSpeak using Arduino UNO
13.	Arduino Based DC Current Measurement using Hall Effect Current Sensor
14.	Directional Control of the DC Motor using Arduino
15.	Interfacing of Relay with Arduino
16.	Building Intrusion Detection System with Arduino and Ultrasonic Sensor

EXPERIMENT 1

STUDY OF INTERNAL ARCHITECTURE OF 8085 MICROPROCESSOR AND PIN DIAGRAM

A microprocessor is a multipurpose, programmable logic device that reads binary instructions from a storage device called memory, accepts binary data as input and processes data according to those instructions and provides result as output. It includes an Arithmetic / Logic unit (ALU), a control unit and an array of registers as a small internal memory for holding data while it is being manipulated or processed. It is a general-purpose device which may be used for different purposes in different applications. Configuration of the system is flexible.

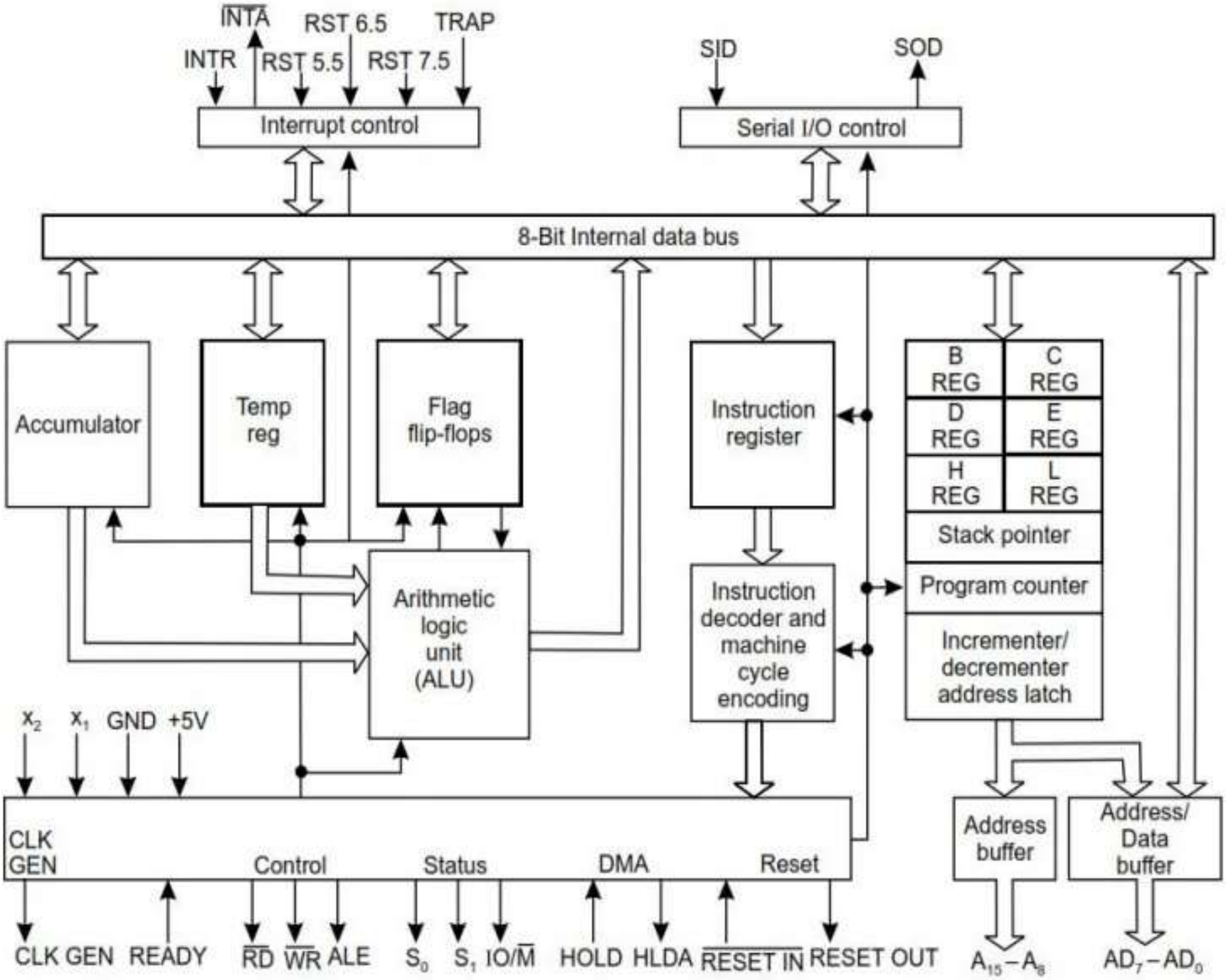


Fig 1.1 Hardware Architecture of 8085

INTERNAL ARCHITECTURE

ALU

The Arithmetic and logic unit (ALU) performs various arithmetic and logic operations like Addition, Subtraction, Logical AND, Logical OR, Logical exclusive OR, complement (Logical NOT), Increment (Add 1), Decrement (Subtract 1), Left shift (add input to itself) and clear (result is zero).

REGISTERS

Registers are small memories within the CPU. They are used by the microprocessor for temporary storage and manipulation of data and instructions. Data remain in the registers till they are sent to the memory or I/O devices.

Registers of 8085 are

- One 8-bit accumulator (ACC) ie, register A.
- Six 8-bit general purpose registers – B, C, D, E, H and L.
- One 16-bit program counter – PC.
- Instruction register – IR.
- Status register – Flag register
- One 16-bit Stack Pointer – SP.
- Temporary register – W and Z.

ACCUMULATOR

The accumulator, one of the most important 8 - bit registers of 8085, is mainly used for arithmetic, logic and rotate operations. The primary purpose of this register is to store temporary data and for the placement of final values of arithmetic and logic operations. It holds one of the operands.

GENERAL PURPOSE REGISTER

There are 6 general purpose registers in the 8085 processor, i.e. B, C, D, E, H& L. Each register can hold 8-bit data. These registers can work in pairs to hold 16-bit data and their pairing combination is like B-C, D-E & H-L. The H-L pair works as a memory pointer.

FLAG REGISTERS

The flag register is a group of flip-flops used to give the status of the result of different operations. The flag register in 8085 is an 8–bit register which contains 5 bit positions. These five flags are 1-bit F/F and are known as sign, zero, auxiliary carry, parity and carry.

CY - Carry flag, it is set when carry is generated and otherwise, it is reset.
Z – Zero flag is set if the result of an operation is zero otherwise it is reset.
S – Sign flag, Signed number is negative if S = 1 and positive if S = 0.
P – Parity flag, it is set for even parity and reset for odd parity.
AC - Auxiliary Carry flag is used for BCD operations. It is set when a carry is generated by digit D3 and passed to D4.

TEMPORARY REGISTER

There are 2 temporary registers, W and Z. It is also called operand register (8-bit). 8085 uses them internally to hold data temporarily during the execution of some instructions.

SPECIAL PURPOSE REGISTERS

It consists of three 16 bit registers – Program counter, Stack pointer, Incrementer / Decrementer Latch.

PROGRAM COUNTER

It holds the address of the next instruction to be executed to save time.

STACK POINTER

Stack is a portion of memory (RAM), that works in the LIFO concept. The stack pointer maintains the address of the last byte that is entered into the stack. Each time when the data is loaded into the stack, the Stack pointer gets decremented.

INCR/ DECR LATCH

It is used to increment or decrement the content of program counter and stack pointer register.

ADDRESS / DATA BUFFER and ADDRESS BUFFER

The contents of the stack pointer and program counter are loaded into the address buffer and address – data buffer. These buffers are then used to drive the external address bus and address–data bus. As the memory and I/O chips are connected to these buses, the CPU can exchange desired data to memory and I/O chips. The address data buffer can both send and receive data from internal data bus.

CONTROL UNIT

It performs data transfer and decision-making operations.

It consists of :

- Instruction Register
- Instruction Decoder
- Timing and Control unit

INSTRUCTION REGISTER

When an instruction like adding two data, moving a data, copying a data etc is fetched from memory, it is directed to the instruction register. So instruction registers are specifically to store the instructions that are fetched from memory.

INSTRUCTION DECODER

It decodes the information present in the instruction register for further processing. It then sends the decoded information to the timing and control unit.

TIMING AND CONTROL UNIT

It synchronizes the registers and flow of data through various registers and other units. This unit consists of an oscillator and sends control signals needed for internal and external control of data and other units. The oscillator generates clock signals.

Signals that are associated with this unit are:

- Control signals: READY, $\overline{RD}$, $\overline{WR}$, ALE
- Status signals: S0, S1, IO/$\overline{M}$
- DMA signals: HOLD, HLDA
- Reset signals: $\overline{RESETIN}$, RESET OUT

CONTROL AND STATUS SIGNALS

- RD – Read (active low) – Indicate that I/O or memory selected is to be read and data are available on the bus.

- WR – Write (active low) – Indicate that data available on the bus are to be written to memory or I/O ports.
- IO/$\overline{M}$ - Differentiate I/O operation or memory operations.
  0 – indicates a memory operation
  1 – indicates an I/O operation
- S1 and S0 – Status signals, tells current operation.

<u>INTERRUPT CONTROLLER</u>

Interrupt signals present in 8085 are:

1. INTR
2. TRAP
3. RST 7.5
4. RST 6.5
5. RST 5.5

Whenever the interrupt signal is enabled or requested, the microprocessor shifts the control from the main program to process the incoming request. After the request is completed, the control goes back to the main program.

<u>SERIAL I/O CONTROL</u>

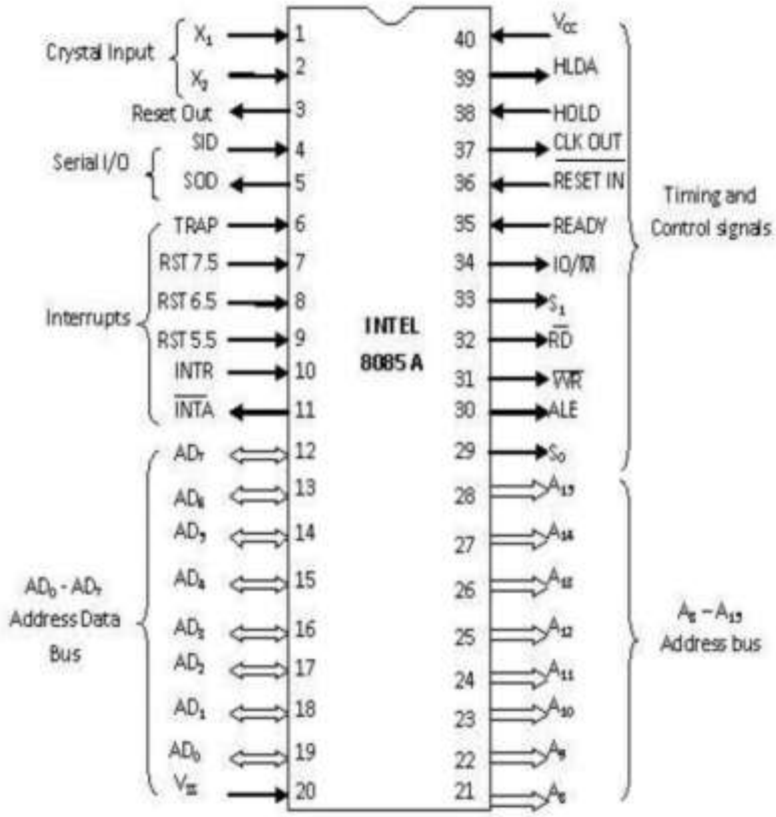
The input and output of serial data can be carried out using two instructions in 8085:

1. SID - Serial input data
2. SOD - Serial output data

Data on these line is accepted or transferred under software control by serial I/O control block, by using special instructions RIM & SIM.

8085 PIN DIAGRAM

8085 is an 8–bit, NMOS microprocessor. It is available as a 40-pin IC package fabricated on a single LSI chip. It uses a single +5V DC supply for its operation. 8085 microprocessor has a clock speed of about 3 MHz and the clock cycle is of 320ns. It has about 6500 transistors. It has 80 basic instructions and 246 opcodes. It consists of three main sections, arithmetic and logic unit, timing and control unit and several registers.



A8-A15 (Output):-
These are address bus and used for the most significant bits of memory address.

AD0-AD7 (Input/Output):-
These are time-multiplexed address data bus. These are used for the least significant 8 bits of the memory address during first clock cycle and then for data during the second and third clock cycle.

ALE (Address Latch Enable):-
It goes high during the 1st clock cycle of a machine. It enables the lower 8 bits of address to be latched either in the memory or external latch.

IO/M:-
It is status signal, when it goes high; the address on address bus is for I/O device, otherwise for memory.

S0, S1:-
These are status signals to distinguish various types of operation.

S1	S0	Operations
0	0	Halt
0	1	Write
1	0	Read
1	1	Opcode Fetch

RD (output):-
It is used to control read operation.

WR (output):-
It is used to control write operation.

HOLD (input):-
It is used to indicate that another device is requesting the use of the address & data bus.

HLDA (output):-
It is an acknowledgement signal used to indicate HOLD request has been received.

INTR (input):-
When it goes high, the microprocessor suspends its normal sequence of operations.

INTA (output):-
It is an interrupt acknowledgement signal sent by the microprocessor after INTR is received.

RST 5.5, 6.5, 7.5 and TRAP:-
These are various interrupt signals. Among them, TRAP is having highest priority.

RESET IN (input):-
It resets the PC to zero.

RESET OUT(output):-
It indicates that the CPU is being reset.

X1, X2 (input):-
This circuitry is required to produce a suitable clock for the operation of microprocessor. .

Clk (output):-
It is clock output for the user. Its frequency is the same at which the processor operates.

SID (input):-
It is used for data line for serial input.

SOD (output):-
It is used for data line for serial output.

Vcc:-
+5 volts supply.

Vss:-
Ground reference.

8085MICROPROCESSOR TRAINER KIT M85-03

M85-03 kit is a single-board Microprocessor training kit based on 8085 microprocessor. It provides monitor EPROM and user's RAM with battery backup. The kit has 28 keys hexadecimal keyboard and six digit seven segment displays for display. The kit also has the capability of interacting with a PC through an RS-232C serial link. The Input/Output structure of M85-03 provides 48 programmable I/O lines using 8255.

PROCEDURE

EXMEM(Examine memory) keyboard command is used to examine the memory locations. To examine the contents of the location for 2500 and 2501, the following key sequence has to be used.

RESET→EXMEM→2500→NEXT→2501

<u>To enter the program</u>

RESET→ EXMEM→ Enter Starting address of program → NEXT → Enter the machine code → NEXT

<u>To execute the program</u>

RESET →GO→ Starting address of program→. (Dot)(Fill Key)

<u>To check the result</u>

RESET→EXMEM→ Enter the address of the result location

<u>To check the register content</u>

Shift → EXREG → A/ B/C/D/E/H/L

EXPERIMENT 2

DATA TRANSFER USING DIFFERENT ADDRESSING MODES AND BLOCK TRANSFER

1. Write an ALP for loading registers A, B, C, D, E, H and L with single-byte data using immediate addressing and observe the register contents.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	3E, 01	START:	MVI A,01H	Load A with 01
2002	06, 02		MVI B,02H	Load B with 02
2004	0E, 03		MVI C,03H	Load C with 03
2006	16, 04		MVI D,04H	Load D with 04
2008	1E, 05		MVI E,05H	Load E with 05
200A	26, 06		MVI H,06H	Load H with 06
200C	2E, 07		MVI L,07H	Load L with 07
200E	EF	END:	RST 05	Return to monitor program

2. Write an ALP for loading registers B, C, D, E, H and L with the same data using register addressing and observe the register contents.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	3A, 50, 20	START:	LDA 2050H	Load accumulator with content of 2050
2003	47		MOV B, A	Move the content of A to B
2004	4F		MOV C, A	Move the content of A to C

2005	57		MOV D, A	Move the content of A to D
2006	5F		MOV E, A	Move the content of A to E
2007	67		MOV H, A	Move the content of A to H
2008	6F		MOV L, A	Move the content of A to L
2009	EF	END:	RST 05	Return to monitor program

3. Write an ALP for loading register pairs BC, DE and HL with 16-bit data using immediate addressing and observe the register pair contents.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	01, 50, 21	START:	LXI B, 2150H	Load BC register pair with data 2150
2003	11, 51, 21		LXI D, 2151H	Load DE register pair with data 2151
2006	21, 52, 21		LXI H, 2152H	Load HL register pair with data 2152
2009	EF	END:	RST 05	Return to monitor program

4. Write an ALP to copy a block of 8-bit data from 4 memory locations (2250-2253) to another 4 memory locations (2254-2257) using direct addressing.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	3A, 50, 22	START:	LDA 2250H	Load data in 2250 to accumulator
2003	32, 54, 22		STA 2254H	Accumulator content stored in 2254
2006	3A, 51, 22		LDA 2251H	Load data in 2251 to accumulator
2009	32, 55, 22		STA 2255H	Accumulator data stored in 2255

200C	3A, 52, 22		LDA 2252H	Load data in 2252 to accumulator
200F	32, 56, 22		STA 2256H	Accumulator data stored in 2256
2012	3A, 53, 22		LDA 2253H	Load data in 2253 to accumulator
2015	32, 57, 22		STA 2257H	Accumulator data stored in 2257
2018	EF	END:	RST 05	Return to monitor program

5. Write an ALP to copy a block of 8-bit data from 4 memory locations (2250-2253) to another 4 memory locations (2254-2257) using 16-bit data transfer addressing mode direct addressing.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	2A, 50, 22	START:	LHLD 2250H	Data in 2250 to L register and datain 2251 to H
2003	22, 54, 22		SHLD 2254H	L register content to 2254 and H content to 2255
2006	2A, 52, 22		LHLD 2252H	Data in 2252 to L register and datain 2253 to H
2009	22, 56, 22		SHLD 2256H	L register content to 2256 and H content to 2257
200C	EF	END:	RST 05	Return to monitor program

6. Write an ALP to transfer a block of 8-bit data from 4 memory locations (2250-2253) to another 4 memory locations (2254-2257) using indirect addressing.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	21, 50, 22	START:	LXI H, 2250H	Set up HL as a pointer of source.
2003	11, 54, 22		LXI D, 2254H	Set up DE as a pointer of destination
2006	06, 04		MVI B, 04	Set up the counter
2008	7E	LOOP:	MOV A, M	Get data from source to accumulator

2009	12		STAX D	Store data in destination
200A	23		INX H	Pointer to next source location
200B	13		INX D	Pointer to next destination location
200C	05		DCR B	Decrement counter
200D	C2, 08, 20		JNZ LOOP	If the transfer is not over, continue
2010	EF	END:	RST 05	Return to monitor program

EXPERIMENT 3

ARITHMETIC OPERATIONS IN BINARY AND BCD: ADDITION AND SUBTRACTION

1. Write an ALP to add two 8-bit numbers, sum 8 bits.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	21, 00, 25	START:	LXI H,2500H	Initialize memory pointer
2003	7E		MOV A, M	Load the first operand from memory to register A
2004	23		INX H	Increment content of H-L pair
2005	46		MOV B, M	Load the second operand from memory to register B
2006	80		ADD B	Add 1st and 2nd numbers
2007	23		INX H	Pointer to store the result
2008	77		MOV M, A	Store result to memory
2009	EF	END:	RST 05	Return to Monitor program

2. Write an ALP to add two 8-bit numbers, sum 16 bits.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	21, 01, 25	START:	LXI H, 2501H	Address of 1st number in H-L pair.
2003	7E		MOV A, M	1st number in accumulator.
2004	23		INX H	Address of 2nd number 2502 in H-L pair.
2005	46		MOV B, M	Load the second operand from memory to register B
2006	0E, 00		MVI C, 00H	MSBs of sum in register C. Initial value = 00.

2008	80		ADD B	1^{st} number + 2^{nd} number.
2009	D2, 0D, 20		JNC AHEAD	Is carry? No, go to the label AHEAD.
200C	0C		INR C	Yes, increment C.
200D	23	AHEAD:	INX H	Increment content of H-L pair
200E	77		MOV M, A	Move the result from A to memory.
200F	23		INX H	Increment content of H-L pair.
2010	71		MOV M, C	Move the result from C to memory.
2011	EF	END:	RST 05	Return to Monitor program

3.Write an ALP to add two16 bit numbers, sum 16 bits or more.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	2A, 01, 25	START:	LHLD 2501H	1^{st} 16-bit number in H-L pair.
2003	EB		XCHG	Get 1^{st} number in D-E pair.
2004	2A, 03, 25		LHLD 2503H	2^{nd} 16-bit number in H-L pair.
2007	0E, 00		MVI C, 00H	MSBs of sum in register C. Initial value = 00.
2009	19		DAD D	1^{st} number + 2^{nd} number.
200A	D2, 0E, 20		JNC AHEAD	Is carry? No, go to the label AHEAD.
200D	0C		INR C	Yes, increment C.
200E	22, 05, 25	AHEAD:	SHLD 2505 H	Store LSBs of sum in 2505 and 2506 H.
2011	79		MOV A, C	MSBs of sum in accumulator
2012	32, 07,25		STA 2507H	MSBs of sum in 2507 H.

2015	EF	END:	RST 05	Return to Monitor program

4. Write an ALP to subtract two 8-bit numbers, difference 8 bits.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	21, 00, 25	START:	LXI H,2500H	Initialize memory pointer
2003	7E		MOV A, M	Load the first operand from memory to register A
2004	23		INX H	Increment content of H-L pair
2005	46		MOV B, M	Load the second operand from memory to register B
2006	90		SUB B	Subtract 2nd number from 1st number
2007	23		INX H	Pointer to store the result
2008	77		MOV M, A	Store result to memory
2009	EF	END:	RST 05	Return to Monitor program

5. Write an ALP for the decimal addition of two 8-bit numbers, sum 16 bits.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	21, 01, 25	START:	LXI H, 2501H	Address of 1st number in H-L pair.
2003	7E		MOV A, M	1st number in accumulator.
2004	23		INX H	Address of 2nd number 2502 in H-L pair.
2005	46		MOV B, M	Load the second operand from memory to register B
2006	0E, 00		MVI C, 00H	MSBs of sum in register C. Initial value = 00.
2008	80		ADD B	1st number + 2nd number.

2009	27		DAA	Decimal adjust
200A	D2, 0E, 20		JNC AHEAD	Is carry? No, go to the label AHEAD.
200D	0C		INR C	Yes, increment C.
200E	23	AHEAD:	INX H	Increment content of H-L pair
200F	77		MOV M, A	Move the result from A to memory.
2010	23		INX H	Increment content of H-L pair.
2011	71		MOV M, C	Move the result from C to memory.
2012	EF	END:	RST 05	Return to Monitor program

6. Write an ALP to add a series of 8-bit numbers, sum 16 bits.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	21, 00, 25	START:	LXI H, 2500H	Load the address of count to HL pair
2003	4E		MOV C, M	Load C with the count value.
2004	3E, 00		MVI A, 00H	LSBs of sum = 00 (initial value)
2006	47		MOV B, A	MSBs of sum = 00 (initial value)
2007	23	LOOP:	INX H	Point to next location.
2008	86		ADD M	Add memory content with accumulator.
2009	D2, 0D, 20		JNC AHEAD	When carry flag is 0, skip next task.
200C	04		INR B	Yes, add carry to MSBs of sum.
200D	0D	AHEAD:	DCR C	Decrement C register by 1.
200E	C2, 07, 20		JNZ LOOP	When Zero flag is not set, go to Loop.

2011	32, 50, 24		STA 2450H	Store LSBs of the sum in 2450 H.
2014	78		MOV A, B	Get MSBs of sum in accumulator.
2015	32, 51, 24		STA 2451H	Store MSBs of the sum in 2451 H.
2018	EF	END:	RST 05	Return to Monitor program

7. Write an ALP to add a series of 8-bit decimal numbers, sum 16 bits.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	21, 00, 25	START:	LXI H, 2500H	Load the address of count to HL pair
2003	4E		MOV C, M	Load C with the count value.
2004	3E, 00		MVI A, 00H	LSBs of sum = 00 (initial value)
2006	47		MOV B, A	MSBs of sum = 00 (initial value)
2007	23	LOOP:	INX H	Point to next location.
2008	86		ADD M	Add memory content with accumulator.
2009	27		DAA	Decimal adjust
200A	D2, 0E, 20		JNC AHEAD	When carry flag is 0, skip next task.
200D	04		INR B	Yes, add carry to MSBs of sum.
200E	0D	AHEAD:	DCR C	Decrement C register by 1.
200F	C2, 07, 20		JNZ LOOP	When Zero flag is not set, go to Loop.
2012	32, 50, 24		STA 2450H	Store LSBs of the sum in 2450 H.
2015	78		MOV A, B	Get MSBs of sum in accumulator.
2016	32, 51, 24		STA 2451H	Store MSBs of the sum in 2451 H.

2019	EF	END:	RST 05	Return to Monitor program

8. Write an ALP to shift an 8-bit number left by 1 bit.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	3A, 01, 25	START:	LDA 2501H	Get data in accumulator.
2003	87		ADD A	Shift it left by one bit.
2004	32, 02, 25		STA 2502H	Store result in 2502 H
2007	EF	END:	RST 05	Return to monitor program

9. Write an ALP to shift an 8-bit number left by 2 bits.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	3A, 01, 25	START:	LDA 2501H	Get data in accumulator.
2003	87		ADD A	Shift it left by one bit.
2004	87		ADD A	Shift it left again by one bit.
2005	32, 02, 25		STA 2502H	Store result in 2502 H
2008	EF	END:	RST 05	Return to monitor program

10. Write an ALP to shift a 16-bit number left by 1 bit.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	2A, 01, 25	START:	LHLD 2501H	Get 16 bit data in HL pair.
2003	29		DAD H	Shift it left by one bit.
2004	22, 03, 25		SHLD 2503H	Store the result in 2503 and 2504 H.

2007	EF	END:	RST 05	Return to monitor program

11.Write an ALP to shift a 16-bit number left by 2 bits.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	2A, 01, 25	START:	LHLD 2501H	Get 16 bit data in HL pair.
2003	29		DAD H	Shift it left by one bit.
2004	29		DAD H	Shift it left again by one bit.
2005	22, 03, 25		SHLD 2503H	Store the result in 2503 and 2504 H.
2008	EF	END:	RST 05	Return to monitor program

EXPERIMENT 4

ARITHMETIC OPERATIONS: MULTIPLICATION AND DIVISION

1. Write an ALP to multiply two 8-bit numbers stored at locations 2500H and 2501H and the product is stored at 2502H and 2503H.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	21, 00, 25	START:	LXI H, 2500H	Load H-L pair with address 2500H
2003	46		MOV B, M	Get the first number in the B register
2004	23		INX H	Increment H-L pair
2005	4E		MOV C, M	Get the second number in the C register
2006	3E, 00		MVI A, 00H	Initialise accumulator with 00H
2008	16, 00		MVI D,00H	Initialise D register with 00H
200A	80	LOOP:	ADD B	Add content of Accumulator to register B.
200B	D2, 0F, 20		JNC AHEAD	Jump on no carry to AHEAD
200E	14		INR D	Increment D register if carry present
200F	0D	AHEAD:	DCR C	Decrement content of register C
2010	C2, 0A, 20		JNZ LOOP	Jump on not zero to LOOP
2013	23		INX H	Increment H-L pair
2014	77		MOV M, A	Move the result from accumulator to memory location 2502H
2015	23		INX H	Increment H-L pair
2016	72		MOV M, D	Move the carry from D register to memory location 2503H
2017	EF	END:	RST 05	Return to monitor program

2. Write an ALP to multiply a 16-bit number by an 8-bit number. Multiplicand is stored at locations 2100H and 2101H and the multiplier is in 2102H. The product is to be stored at 2103H and 2104H.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	21, 02, 21	START:	LXI H, 2102H	Initialize memory pointer with 2102H
2003	46		MOV B, M	Load multiplier in B register
2004	11, 00, 00		LXI D, 0000H	Initialise the DE pair with 0000H
2007	2A, 00, 21		LHLD 2100H	Load multiplicand in H-L pair
200A	EB		XCHG	Exchange DE with HL pair
200B	19	BACK:	DAD D	Add DE and HL contents
200C	05		DCR B	Decrement register B
200D	C2, 0B, 20		JNZ BACK	If not zero, go to BACK
2010	22, 03, 21		SHLD 2103H	Store the product in HL pair to 2103H and 2104H
2013	EF	END:	RST 05	Return to monitor program

3. Write an ALP for binary division. The 8-bit divisor and dividend are stored at memory locations 2100H and 2101H respectively. The remainder and quotient should be stored at 2102H and 2103H respectively.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	21, 00, 21	START:	LXI H, 2100H	Initialize HL pair as memory pointer
2003	46		MOV B, M	Load divisor in B register
2004	23		INX H	Increment HL pair
2005	7E		MOV A, M	Load dividend to accumulator

2006	23		INX H	Increment HL pair
2007	0E, 00		MVI C, 00H	Initialize quotient as 00H
2009	B8		CMP B	Is dividend less than divisor?
200A	DA, 13, 20		JC AHEAD	If yes, jump to AHEAD
200D	90	BACK:	SUB B	Subtract divisor from dividend
200E	0C		INR C	Increment C register
200F	B8		CMP B	Is dividend less than divisor
2010	D2, 0D, 20		JNC BACK	If no carry, jump to BACK
2013	77	AHEAD:	MOV M, A	Store remainder at 2102H
2014	23		INX H	Increment HL pair
2015	71		MOV M, C	Store quotient at 2103H
2016	EF	END	RST 05	Return to monitor program

EXPERIMENT 5

BINARY TO BCD CONVERSION AND BCD TO BINARY CONVERSION

1. Write an ALP to convert BCD to Binary

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	3A, 00, 25	START:	LDA 2500H	Load accumulator with content of address 2500H
2003	47		MOV B, A	Move data from accumulator to reg. B
2004	E6, F0		ANI F0H	AND F0 with accumulator content
2006	0F		RRC	Rotate accumulator content right by 1 bit
2007	0F		RRC	Rotate accumulator content right by 1 bit
2008	0F		RRC	Rotate accumulator content right by 1 bit
2009	0F		RRC	Rotate accumulator content right by 1 bit
200A	57		MOV D, A	Move data from accumulator to reg. D
200B	0E, 0A		MVI C, 0AH	Initialise C register with 0AH
200D	97		SUB A	Subtract A from A (clearing accumulator)
200E	82	BACK:	ADD D	Add D with A
200F	0D		DCR C	Decrement C register
2010	C2, 0E, 20		JNZ BACK	Jump if not zero to BACK
2013	57		MOV D, A	Move data from accumulator to reg D
2014	78		MOV A, B	Move data from reg B to accumulator
2015	E6, 0F		ANI 0FH	AND 0F with accumulator content
2017	82		ADD D	Add D with A

2018	32, 01, 25		STA 2501H	Store accumulator content in 2501H
201B	EF	END:	RST 05	Return to monitor program

2. Write an ALP to convert Binary to BCD

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	16,00	START:	MVI D, 00H	Initialise D with 00H
2002	1E,00		MVI E, 00H	Initialise E with 00H
2004	21, 00, 24		LXI H, 2400H	Load H-L pair with address 2400H
2007	7E		MOV A, M	Move data from memory to accumulator
2008	FE, 64	HUND:	CPI 64H	Compare data in accumulator with 64H
200A	DA 13, 20		JC TEN	Jump on carry to label TEN
200D	1C		INR E	Increment E register
200E	D6, 64		SUI 64H	Subtract 64H from accumulator
2010	C3, 08, 20		JMP HUND	Jump to label HUND
2013	FE, 0A	TEN:	CPI 0AH	Compare data in accumulator with 0AH
2015	DA, 1E, 20		JC UNIT	Jump if carry to label UNIT
2018	14		INR D	Increment D register
2019	D6, 0A		SUI 0AH	Subtract 0AH from accumulator
201B	C3, 13, 20		JMP TEN	Jump to label TEN
201E	23	UNIT:	INX H	Increment H-L pair
201F	73		MOV M,E	Move data from reg. E to memory

2020	4F		MOV C,A	Move data from accumulator to reg. C
2021	7A		MOV A,D	Move data from reg. D to accumulator
2022	07		RLC	Rotate accumulator content left by 1 bit
2023	07		RLC	Rotate accumulator content left by 1 bit
2024	07		RLC	Rotate accumulator content left by 1 bit
2025	07		RLC	Rotate accumulator content left by 1 bit
2026	81		ADD C	Add C with A
2027	23		INX H	Increment H-L pair
2028	77		MOV M, A	Move data from accumulator to memory
2029	EF	END:	RST 05	Return to monitor program

EXPERIMENT 6

LOGICAL OPERATIONS

1. Write an ALP to find the larger of two numbers.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	21, 01, 25	START:	LXI H, 2501H	Address of 1st number in H-L pair.
2003	7E		MOV A, M	1st number in accumulator.
2004	23		INX H	Address of 2nd number in H-L pair.
2005	BE		CMP M	Compare 2nd number with 1st number. Is the 2nd number >1st ?
2006	D2, 0A, 20		JNC AHEAD	No, larger number is in accumulator. Go to AHEAD.
2009	7E		MOV A, M	Yes, get 2nd number in accumulator.
200A	32, 03, 25	AHEAD:	STA 2503H	Store larger number in 2503H.
200D	EF	END:	RST 05	Return to monitor program

2. Write an ALP to find the smaller of two numbers.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	21, 01, 25	START:	LXI H, 2501H	Address of 1st number in H-L pair.
2003	7E		MOV A, M	1st number in accumulator.
2004	23		INX H	Address of 2nd number in H-L pair.
2005	BE		CMP M	Compare 2nd number with 1st number. Is the 2nd number >1st ?
2006	DA, 0A, 20		JC AHEAD	Yes, smaller number is in accumulator. Go to AHEAD.

2009	7E		MOV A, M	No, get 2^{nd} number in accumulator.
200A	32, 04, 25	AHEAD:	STA 2504H	Store smaller number in 2504H.
200D	EF	END:	RST 05	Return to monitor program

3. Write an ALP to find the largest number in an array of 8-bit numbers.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	21, 00, 25	START:	LXI H, 2500H	Address for count in H-L pair.
2003	4E		MOV C, M	Count in register C.
2004	23		INX H	Address of 1^{st} number in H-L pair.
2005	7E		MOV A, M	1^{st} number in accumulator.
2006	0D		DCR C	Decrement count.
2007	23	LOOP:	INX H	Address of next number.
2008	BE		CMP M	Compare next no. with previous maximum. Is next no. > previous?
2009	D2, 0D, 20		JNC AHEAD	No, larger number is in accumulator. Go to the label AHEAD.
200C	7E		MOV A, M	Yes, get larger number in accumulator.
200D	0D	AHEAD:	DCR C	Decrement count.
200E	C2, 07, 20		JNZ LOOP	Jump if not zero.
2011	32, 50, 24		STA 2450H	Store result in 2450H.
2014	EF	END:	RST 05	Return to monitor program

4. Write an ALP to find the smallest number in an array of 8-bit numbers.

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	21, 00, 25	START:	LXI H, 2500H	Address for count in H-L pair.
2003	4E		MOV C, M	Count in register C.
2004	23		INX H	Address of 1st number in H-L pair.
2005	7E		MOV A, M	1st number in accumulator.
2006	0D		DCR C	Decrement count.
2007	23	LOOP:	INX H	Address of next number.
2008	BE		CMP M	Compare next no. with previous maximum. Is next no. > previous?
2009	DA, 0D, 20		JC AHEAD	Yes, smaller number is in accumulator. Go to the label AHEAD.
200C	7E		MOV A, M	No, get smaller number in accumulator.
200D	0D	AHEAD:	DCR C	Decrement count.
200E	C2, 07, 20		JNZ LOOP	Jump if not zero.
2011	32, 51, 24		STA 2451H	Store result in 2451H.
2014	EF	END:	RST 05	Return to monitor program

5. Write an ALP to sort an array of 8-bit numbers in the descending order.

MEMORY ADDRESS	MACHINE CODES	LABEL	MNEMONICS	COMMENTS
2000	21, 00, 26	START:	LXI H, 2600H	Set pointer for array.

2003	4E		MOV C, M	Load the Count.
2004	0D		DCR C	Decrement Count.
2005	51	REPEAT:	MOV D, C	Count the number of Comparisons in register D.
2006	21, 01, 26		LXI H, 2601H	Load starting address of data array.
2009	7E	LOOP:	MOV A, M	Copy content of memory location to Accumulator.
200A	23		INX H	Increment content of HL pair
200B	BE		CMP M	Compare the number with next number.
200C	D2, 14, 20		JNC SKIP	Jump to skip if carry not generated.
200F	46		MOV B, M	Copy content of memory location to B Register.
2010	77		MOV M, A	Copy content of Accumulator to memory location.
2011	2B		DCX H	Decrement content of HL pair
2012	70		MOV M, B	Copy content of B Register to memory location.
2013	23		INX H	Increment content of HL pair
2014	15	SKIP:	DCR D	Decrement D register
2015	C2, 09, 20		JNZ LOOP	Jump to LOOP if not Zero.
2018	0D		DCR C	Decrement C register
2019	C2, 05, 20		JNZ REPEAT	Jump to REPEAT if not Zero.
201C	EF	END:	RST 05	Return to monitor program

6. Write an ALP to sort an array of 8-bit numbers in the ascending order.

MEMORY ADDRESS	MACHINE CODES	LABEL	MNEMONICS	COMMENTS
2000	21, 00, 26	START:	LXI H, 2600H	Set pointer for array.
2003	4E		MOV C, M	Load the Count.
2004	0D		DCR C	Decrement Count.
2005	51	REPEAT:	MOV D, C	Count the number of Comparisons in register D.
2006	21, 01, 26		LXI H, 2601H	Load starting address of data array.
2009	7E	LOOP:	MOV A, M	Copy content of memory location to Accumulator.
200A	23		INX H	Increment content of HL pair
200B	BE		CMP M	Compare the number with next number.
200C	DA, 14, 20		JC SKIP	Jump to skip if carry generated.
200F	46		MOV B, M	Copy content of memory location to B Register.
2010	77		MOV M, A	Copy content of Accumulator to memory location.
2011	2B		DCX H	Decrement content of HL pair
2012	70		MOV M, B	Copy content of B Register to memory location.
2013	23		INX H	Increment content of HL pair
2014	15	SKIP:	DCR D	Decrement D register
2015	C2, 09, 20		JNZ LOOP	Jump to LOOP if not Zero.
2018	0D		DCR C	Decrement C register

2019	C2, 05, 20		JNZ REPEAT	Jump to REPEAT if not Zero.
201C	EF	END:	RST 05	Return to monitor program

EXPERIMENT 7

DIGITAL I/O USING PPI- SQUARE WAVE GENERATION

Aim

To generate a pulse train of frequency 200 Hz and duty cycle 50%.

Theory

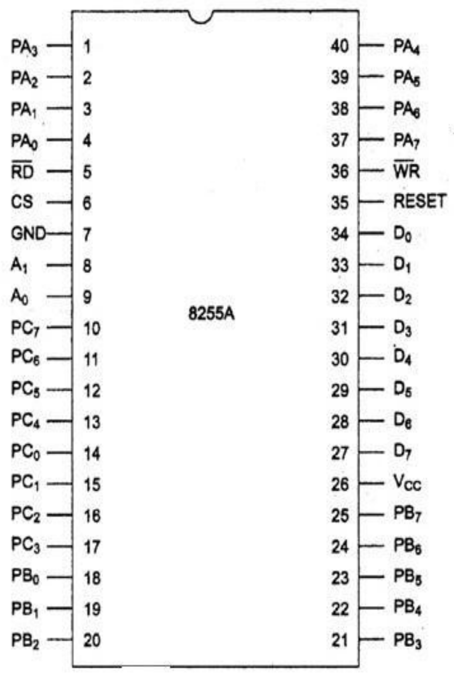
Waveform generation using microprocessor requires input-output ports interfaced to it. Programmable peripheral interface (PPI) 8255 is a general purpose programmable I/O device designed to interface the CPU with its outside world such as ADC, DAC, keyboard etc. IC 8255 provides 3 nos. of 8-bit ports (Port A, Port B and Port C). IC 8255 needs to be initialized before use. Initialization includes setting mode of 8255 (Input-Output or Bit Set Reset) and data direction in case of IO mode (Input or Output) for individual ports. In this case, 8255 is setup in I/O mode with all ports as output ports. So the initialization control word is 80H.

The data send out from microprocessor to 8255 may be directed to Port A, B, C or Control Word Register (CWR). Each register mentioned above is given an 8-bit address.

I/O address range

Port A = 00H; Port B = 01H; Port C = 02H; Control word Register CWR = 03H

To initialize the 8255, load control word (80H) in Accumulator and send it to CWR (03H)



8255-I CONNECTOR-CN4			
PIN	**SIGNALS**	**PIN**	**SIGNALS**
1	**P1C4**	14	**P1B1**
2	**P1C5**	15	**P1A6**
3	**P1C2**	16	**P1A7**
4	**P1C3**	17	**P1A4**
5	**P1C0**	18	**P1A5**
6	**P1C1**	19	**P1A2**
7	**P1B6**	20	**P1A3**
8	**P1B7**	21	**P1A0**
9	**P1B4**	22	**P1A1**
10	**P1B5**	23	**P1C6**
11	**P1B2**	24	**P1C7**
12	**P1B3**	25	**GND**
13	**P1B0**	26	**VCC**

- **PA0 – PA7 –** Pins of port A
- **PB0 – PB7 –** Pins of port B
- **PC0 – PC7 –** Pins of port C
- **D0 – D7 –** Data pins for the transfer of data
- **RESET –** Reset input
- **RD' –** Read input
- **WR' –** Write input
- **CS' –** Chip select
- **A1 and A0 –** Address pins

Algorithm- Square Wave Waveform

Delay Calculation

The delay time required for frequency of 200 Hz is 2500µs for low and high states. Time delay subroutines load a value in a register or register-pair and decrement it. When the value equals 0, it returns.

The statements from DCX D to JNZ REP is repeated N times (where N is loaded in DE register-pair). LXI D, N and RET are executed only once.

Total T states in time delay = 24N +17

System frequency = 3.072 MHz

Thus, 1 T state = 0.3255µs

Time delay, td = (24N + 17) x 0.3255µs

For a delay of 2500 µs,

$2500 \times 10^{-6} = (24 \times N + 17) \times 0.3255 \times 10^{-6}$

$N = 319_{10} = 013FH$

Algorithm:

1. Start
2. Set control word (Port A as output port)
3. Set port A low
4. Call delay
5. Set port A high
6. Call delay
7. Go to Step 3

Program

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	3E, 80		MVI A, 80H	Load A with immediate data 80H
2002	D3, 03		OUT 03H	Send content of A to CWR
2004	3E, 00	LOOP:	MVI A, 00H	Load A with immediate data 00H
2006	D3, 00		OUT 00H	Send content of Acc to output port A
2008	CD, 15, 20		CALL DELAY	Call the delay subroutine
200B	3E, FF		MVI A, FFH	Load A with immediate data FFH
200D	D3, 00		OUT 00H	Send content of Acc to output port A
200F	CD, 15, 20		CALL DELAY	Call the delay subroutine
2012	C3, 04, 20		JMP LOOP	Jump to LOOP to repeat
DELAY SUBROUTINE				
2015	11, 3F, 01	DELAY:	LXI D, 013FH	Load DE register pair with value of N
2018	1B	REP:	DCX D	Decrement D
2019	7A		MOV A, D	Move content of D to A
201A	B3		ORA E	OR the value of E with A and store the result in A
201B	C2,18, 20		JNZ REP	Jump on non zero to REP
201E	C9		RET	Return to main program

Procedure

Enter the program for square wave generation from memory location 2000H onwards. Execute the program and observe the waveform available at the pins of port A. (Connect the probe to the corresponding Port A signal pins of CN4 connector)

Result

EXPERIMENT 8

INTERFACING D/A CONVERTER : GENERATION OF SIMPLE WAVEFORMS-TRIANGULAR, RAMP

i. Triangular Waveform

Aim

To generate a triangular wave of suitable amplitude using DAC interface.

Algorithm

1. To initialize 8255, load control word (80H) in Accumulator and sent it to CWR (03H)
2. Clear Accumulator
3. Send Accumulator content to output port A (00H)
4. Increment Accumulator data
5. If Accumulator content not equal to FFH, go to Step 3
6. Out Accumulator content to Port A (00H)
7. Decrement Accumulator
8. If Accumulator content not equal to 00H, go to Step 6
9. Go to Step 3

Program

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	3E, 80		MVI A, 80H	Load Acc with immediate data 80H
2002	D3, 03		OUT 03H	Send content of Acc to CWR
2004	AF		XRA A	Clear the accumulator
2005	D3, 00	LOOP1:	OUT 00H	Output the contents of Acc to the output port A
2007	3C		INR A	Increment Acc
2008	FE, FF		CPI FFH	Compare the content of Accumulator with maximum
200A	C2, 05, 20		JNZ LOOP1	Jump to LOOP1 if the result of comparison is not equal to zero

200D	D3, 00	LOOP2:	OUT 00H	Output the value in acc at the port A
200F	3D		DCR A	Decrement Acc
2010	C2, 0D, 20		JNZ LOOP2	Jump to LOOP2 if result of comparison is not equal to zero
2013	C3, 05, 20		JMP LOOP1	Jump to LOOP1 to repeat the process

ii. Ramp (Sawtooth) Waveform

Aim

To generate a sawtooth waveform of suitable amplitude using DAC interface.

Algorithm:

1. To initialize 8255, load control word (80H) in Accumulator and sent it to CWR (03H)
2. Clear Accumulator
3. Out Accumulator content to Port A
4. Increment Accumulator
5. Go to Step 3

(Since Accumulator is an 8-bit register, incrementing from FFH results in 00H)

Program

MEMORY ADDRESS	MACHINE CODE	LABEL	MNEMONICS	COMMENTS
2000	3E, 80		MVI A, 80H	Load A with immediate data 80H
2002	D3, 03		OUT 03H	Send the contents of A to output port
2004	AF		XRA A	EXOR the value of A with A itself. This resets/clear
2005	D3, 00	LOOP:	OUT 00H	Output the content of A to port
2007	3C		INR A	Increment A
2008	C3, 05, 20		JMP LOOP	Jump to LOOP

Procedure

Enter the program from memory location 2000H onwards. Connect the CN4 pins of 8255 with the DAC module. Execute the program and observe the output between X- Out and GND pins of the DAC.

Result

The following waveforms were generated using 8085.

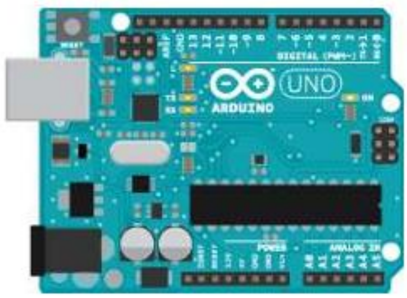
Triangular waveform

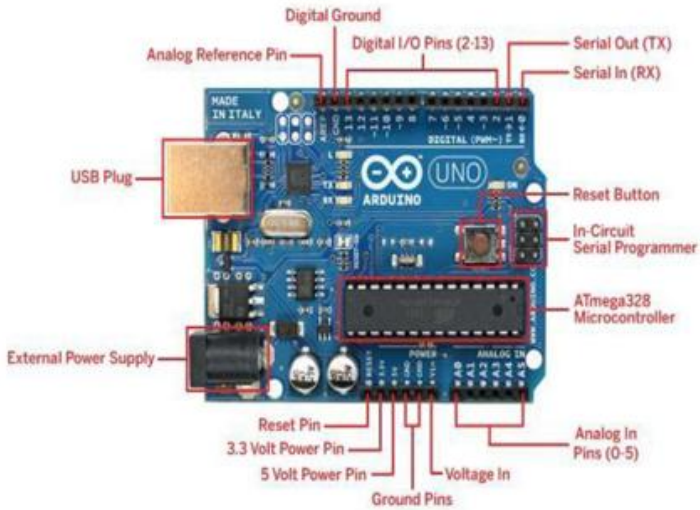
Ramp waveform

EXPERIMENT 9

FAMILIARIZATION OF ARDUINO IDE

Arduino is an open-source physical computing platform based on a simple microcontroller. It is a copyright held by the original team in Italy that developed the hardware, the IDE (Integrated Development Environment), and the software libraries. The Arduino development environment can run on Windows, Linux, or macOS at no cost other than the hardware itself. The software is freely downloadable as a single bundle from www.arduino.cc, the official website for all things Arduino. Arduino can sense its environment by receiving input from various sensors and can interact with its surroundings by controlling lights, motors, and other actuators.





The Arduino board comes with a single LED, often called the Pin 13 LED because it is electrically connected to Digital Pin 13. This LED is the board's only built-in indicator accessible to programs.

EXPERIMENT 10

BLINKING INTERNAL LED OF ARDUINO UNO MODULE

Aim

To blink internal LED of Arduino UNO

Procedure

In the menu of the Arduino IDE you can choose:

File ‣ Examples ‣ 01. Basics ‣ Blink

The IDE will open the code to blink the built in LED automatically.

Uploading code to the Arduino

Now our program is ready to upload to the Arduino. First we have to connect our Arduino to the computer with the USB cable. Make sure you've selected the correct board in the IDE:

Tools ‣ Board ‣ Arduino/Genuino UNO

and the correct port:

Tools ‣ Port

If you are not sure which port to use, try them all until you can successfully upload your code. Then verify your code for possible errors. The IDE only checks if it can read your code. It does not check if you have written correct code for what you are trying to program. If everything works, the IDE shows the Compiling completed message. You can now upload your code by pressing the upload button. The uploading is complete when the messages appear. Your program will immediately start after uploading. As a result, you should now see your Arduino LED blink with 1000ms intervals.

Program

```
void setup

{

// initialize digital pin LED_BUILTIN as an output.
```

pinMode(LED_BUILTIN, OUTPUT);

} // the loop function runs over and over again forever

void loop()

{

digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level) delay(1000); // wait for a second

digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW delay(1000); // wait for a second

}

Result

BLINKING EXTERNAL LED USING ARDUINO UNO MODULE

Aim

To blink an externally connected LED using Arduino UNO

Program

int LED = 8;

void setup()

{

// initialize digital pin LED as an output.

pinMode(LED, OUTPUT);

} // the loop function runs over and over again forever

void loop()

{

```
digitalWrite(LED, HIGH); // turn the LED on (HIGH is the voltage level)
delay(1000); // wait for a second
digitalWrite(LED, LOW); // turn the LED off by making the voltage LOW
delay(1000); // wait for a second }
```

Result

EXPERIMENT 11

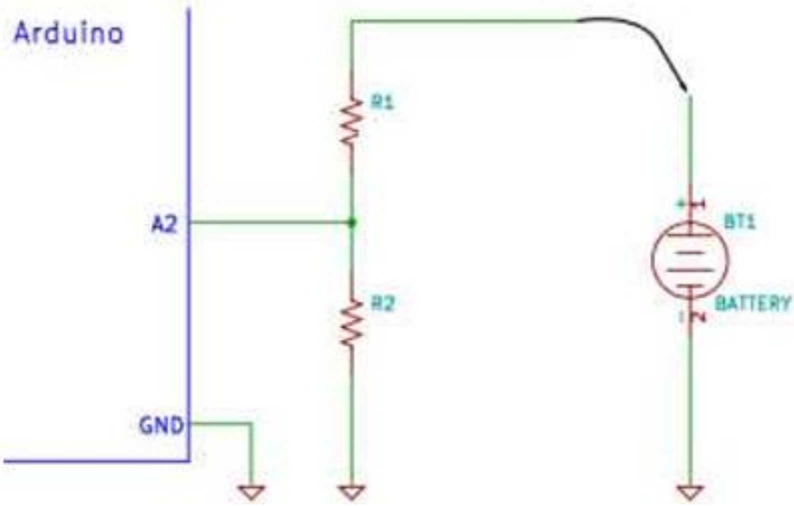
ARDUINO BASED VOLTAGE MEASUREMENT

Aim

To measure a DC voltage in range 0-9 V using Arduino UNO

Theory

A voltage divider circuit consisting of two resistors in series will divide the input voltage to bring it within the range of the Arduino analog inputs.



Design

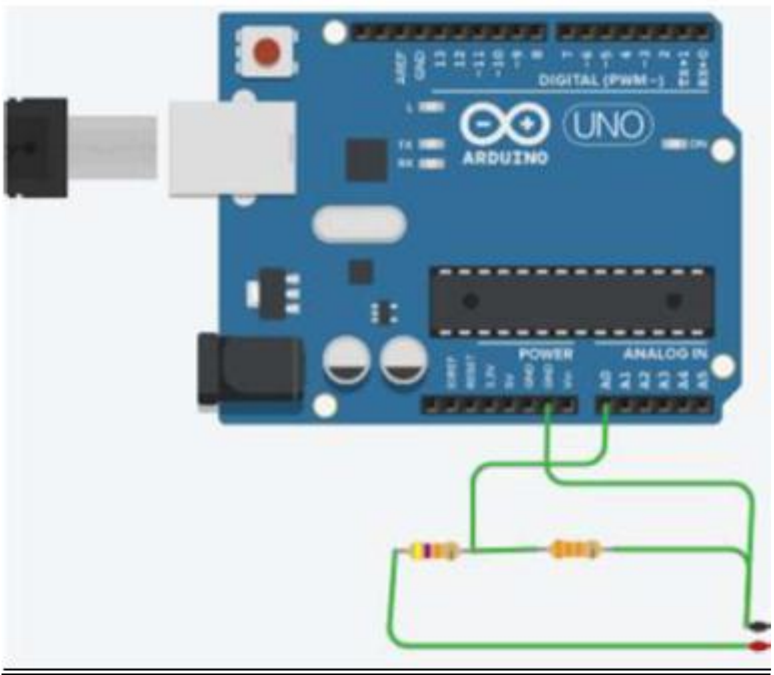
(Design the suitable values of R1 & R2, such that maximum voltage across R2 will be 5V, when actual maximum input voltage is applied across series combination, in this case, 9V)

Program

```
int value = 0;
float voltage;
float R1 = 100.0;
float R2 = 330.0;
void setup()
{
```

```
pinMode(A0, INPUT);
Serial.begin(9600);
}
void loop()
{
value = analogRead(A0);
voltage = value * (5.0/102)*((R1 + R2)/R2);
Serial.print("Voltage =");
Serial.println(voltage);
delay(500);
}
```

Circuit Diagram



Result

EXPERIMENT 12

TEMPERATURE AND HUMIDITY MONITORING OVER THINGSPEAK USING ARDUINO UNO

Aim

To send Temperature and humidity data to Thingspeak Using Arduino and ESP8266

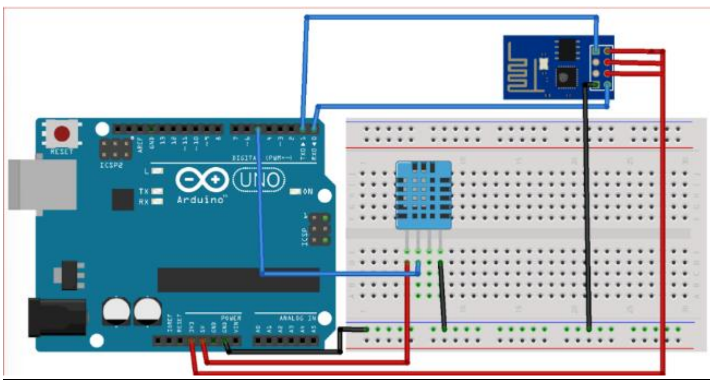
Theory

Arduino Uno reads the current temperature and humidity data from DHT11 and sends it to the ThingSpeak server for live monitoring from anywhere in the world.

Components Required

- Arduino Uno
- ESP8266 WiFi Module
- DHT11 Sensor
- Breadboard
- Jumper Wires

Circuit Diagram



Connections

S.NO.	Pin Name	Arduino Pin
1	ESP8266 VCC	3.3V
2	ESP8266 RST	3.3V
3	ESP8266 CH-PD	3.3V
4	ESP8266 RX	TX
5	ESP8266 TX	RX
6	ESP8266 GND	GND
7	DHT-11 VCC	5V
8	DHT-11 Data	5
9	DHT-11 GND	GND

Step 1: ThingSpeak Setup for Temperature and Humidity Monitoring

Step 2: Create a Channel for Your Data

Step 3: API Key

Program

Programming Arduino for Sending data to ThingSpeak

```
AT
AT+CWMODE=1
AT+CWJAP="CircuitLoop","circuitdigest101"
AT+CIPSTART="TCP","184.106.153.149",80
AT+CIPSEND=56
AT+CIPCLOSE
AT+CIPSTART="TCP","184.106.153.149",80
AT+CIPSEND=56
GET /update?key=9B6ILVOYMUSVOADA&field1=19.7&field2=50
AT+CIPSTART="TCP","184.106.153.149",80
AT+CIPSEND=56
GET /update?key=9B6ILVOYMUSVOADA&field1=19.7&field2=50
AT+CIPSTART="TCP","184.106.153.149",80
AT+CIPSEND=56
GET /update?key=9B6ILVOYMUSVOADA&field1=19.7&field2=50
AT+CIPSTART="TCP","184.106.153.149",80
AT+CIPSEND=56
GET /update?key=9B6ILVOYMUSVOADA&field1=19.7&field2=50
AT+CIPSTART="TCP","184.106.153.149",80
AT+CIPSEND=56
GET /update?key=9B6ILVOYMUSVOADA&field1=19.8&field2=50
AT+CIPSTART="TCP","184.106.153.149",80
AT+CIPSEND=56
```

Arduino code

```
#include <stdlib.h>
#include <DHT.h>
#define DHTPIN 5         // DHT data pin connected to Arduino pin 5
#define DHTTYPE DHT11     // DHT11 (DHT Sensor Type )
DHT dht(DHTPIN, DHTTYPE); // Initialize the DHT sensor
#define SSID "WiFi Name"     // "WiFi Name"
#define PASS "WiFi Password"       // "Password"
#define IP "184.106.153.149"// thingspeak.com ip
String msg = "GET /update?key=Your API Key"; //change it with your key...
float temp;
int hum;
String tempC;
int error;
void setup()
{Serial.begin(115200); // use default 115200.
```

```
Serial.println("AT");
  delay(5000);
  if(Serial.find("OK")){
    connectWiFi();}
}
void loop(){
   start:
  error=0;
  temp = dht.readTemperature();
  hum = dht.readHumidity();
  char buffer[10];
  tempC = dtostrf(temp, 4, 1, buffer);
  updateTemp();
  if (error==1){
    goto start;
  }
 delay(5000);
}
void updateTemp(){
  String cmd = "AT+CIPSTART=\"TCP\",\"";
  cmd += IP;
  cmd += "\",80";
  Serial.println(cmd);
  delay(2000);
  if(Serial.find("Error")){
    return;}
  cmd = msg ;
  cmd += "&field1=";
  cmd += tempC;

cmd += "&field2=";
```

```
  cmd += String(hum);
  cmd += "\r\n";
  Serial.print("AT+CIPSEND=");
  Serial.println(cmd.length());
  if(Serial.find(">")){
    Serial.print(cmd);
  }
  else{
    Serial.println("AT+CIPCLOSE");
    //Resend...
    error=1;
  }
}
  boolean connectWiFi(){
  Serial.println("AT+CWMODE=1");
  delay(2000);
  String cmd="AT+CWJAP=\"";
  cmd+=SSID;
  cmd+="\",\"";
  cmd+=PASS;
  cmd+="\"";
  Serial.println(cmd);
  delay(5000);
  if(Serial.find("OK")){
    return true;
  }else{
    return false;
  }
}
```

Result

EXPERIMENT 13

ARDUINO BASED DC CURRENT MEASUREMENT USING HALL EFFECT CURRENT SENSOR

Aim

To measure the DC current using Hall effect current sensor and display the value using the 12C LCD module

Theory

ACS712 is a hall effect current sensor. Hall effect will convert the magnetic field created by the current that flows through the sensor to voltage directly proportional to it and makes this sensor able to generate voltage linear with the current that flows through this sensor.

Components Required

Arduino UNO

USB cable for Arduino

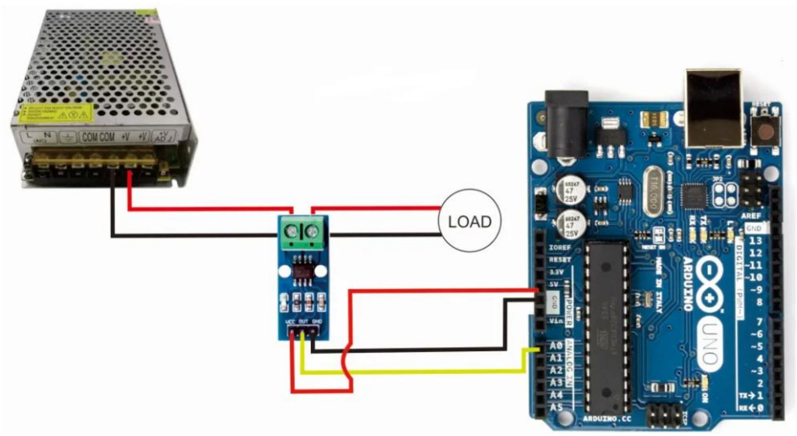
ACS712 sensor

DuPont wire set

Breadboard

LCD display

Circuit Diagram



Program Code

```
void setup() {
// put your setup code here, to run once:
Serial.begin(9600);
}
void loop() {
// put your main code here, to run repeatedly:
int adc = analogRead(A0);
float voltage = adc*(5/1023.0);
float current = (voltage-2.5)/0.185;
Serial.print("Current : ");
Serial.println(current);
delay(300);
}
```

To display on LCD

```
#include "ACS712.h"
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);
ACS712 sensor(ACS712_05B, A0);
//ACS712_20A for 20 Amp type
//ACS712_30A for 30 Amp type
void setup() {
Serial.begin(9600);
sensor.calibrate();
lcd.init();
lcd.backlight();
```

```
lcd.setCursor(0, 0);
lcd.print("ACS712 Demo");
}
void loop() {
float I = sensor.getCurrentAC();
if (I < 0.09) {
I = 0;
}
lcd.setCursor(0, 0);
lcd.print("Current : ");
lcd.print(I);
lcd.print(" A");
delay(300);
}
```

Result

EXPERIMENT 14

DIRECTIONAL CONTROL OF THE DC MOTOR USING ARDUINO

Aim

Directional Control of DC motor using Arduino

Theory

There is very basic to control the dc motor direction. all we know there is two wires in the DC motor. one is positive and other is negative. when we connect positive wire to positive terminal and negative wire to negative terminal the motor will rotate clock wise.

and when we connect negative wire of motor with positive terminal and positive wire with the negative terminal then the motor will rotate anti clock wise.

Components Required

Arduino Uno

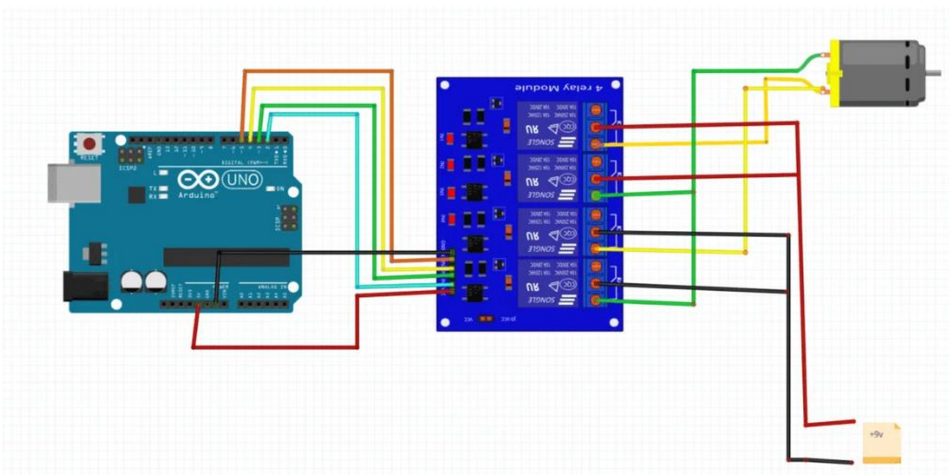
Potentiometer

5v Relay-4

Zero PCB board

Bc547

Circuit Diagram



Program Code

```
void setup()
{
 pinMode(2, OUTPUT);
pinMode(3, OUTPUT);
pinMode(4, OUTPUT);
pinMode(5, OUTPUT);
}
void loop()
{
digitalWrite(2, HIGH);
digitalWrite(3, HIGH);
digitalWrite(4, LOW);
digitalWrite(5, LOW);
delay(100000);
 digitalWrite(2, LOW);
digitalWrite(3, LOW);
digitalWrite(4, HIGH);
 digitalWrite(5, HIGH);
delay(5000);
}
```

Result

EXPERIMENT 15

INTERFACING OF RELAY WITH ARDUINO

Aim

To interface a relay with Arduino

Theory

Relay is an electromagnetic switch, which is controlled by small current, and used to switch ON and OFF relatively much larger current. Means by applying small current we can switch ON the relay which allows much larger current to flow.

Components Required

Arduino

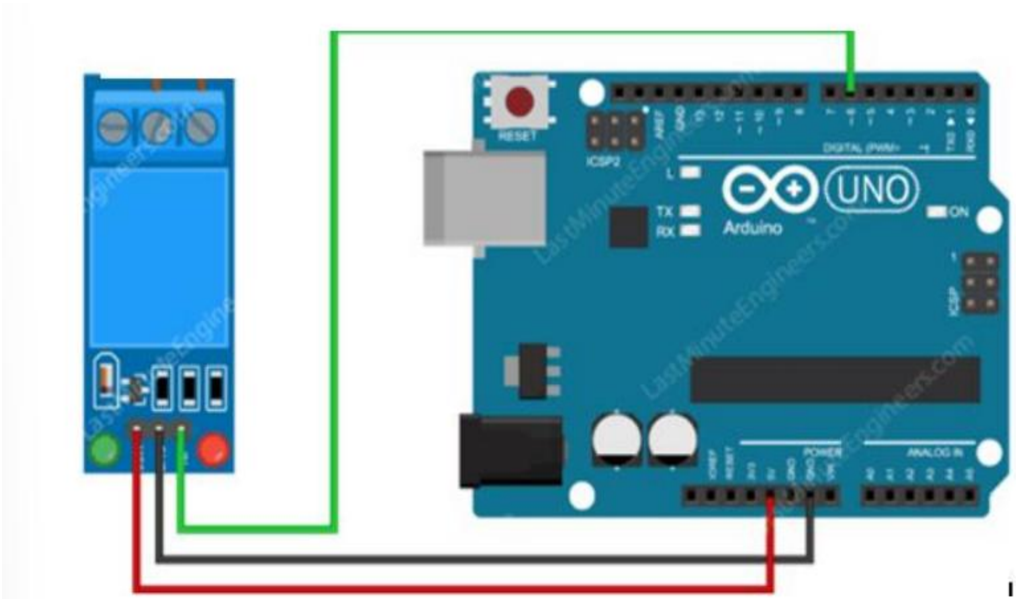
5V Relay

USB Cable

Breadboard

Jumper Wires

Connection Diagram



Steps of working:

1. The relay module connected with three pins. We will connect the relay module with Arduino in the normally open state. The black one of relay is usually the ground. Connect this to the Arduino GND.

2. Connect the red wire of relay module to 5V of the Arduino.

3. Connect the signal pin of relay module to a digital pin 6 of the Arduino.

4. Upload the code

5. Observe the clicking sound of the relay that states the ON and OFF constantly

Program Code

```
// Arduino Relay Control Code
int relayPin=6;
#define interval 2000
void setup() {
 pinMode(relayPin, OUTPUT);
}
void loop()
{
 digitalWrite(relayPin, HIGH);
 delay(interval);
 digitalWrite(relayPin, LOW);
 delay(interval);
}
```

Observations:

Sr no.	Delay	Relay Status
1.		
2.		

EXPERIMENT 16

BUILDING INTRUSION DETECTION SYSTEM WITH ARDUINO AND ULTRASONIC SENSOR

Aim

To build and intrusion detection system (IDS) with Arduino and Ultrasonic sensor.

Theory

The HC-SR04 ultrasonic sensor uses SONAR to determine the distance of an object just like the bats do. It offers excellent non-contact range detection with high accuracy and stable readings in an easy to-use package from 2 cm to 400 cm or 1" to 13 feet. It comes complete with ultrasonic transmitter and receiver module. The ultrasonic sensor uses the reflection of sound in obtaining the time between the wave sent and the wave received. It usually sent a wave at the transmission terminal and receives the reflected waves. The time taken is used together with the normal speed of sound in air (340ms-1) to determine the distance between the sensor and the obstacle. The Ultrasonic sensor is used here for the intruder detection. The sound via a buzzer occurs when an object comes near to the sensor. The distance to which the sensor will respond can be easily adjusted in the program.

Hardware Required

Arduino UNO

Red LED

Green LED

HC-SR04 Ultrasonic Sensor

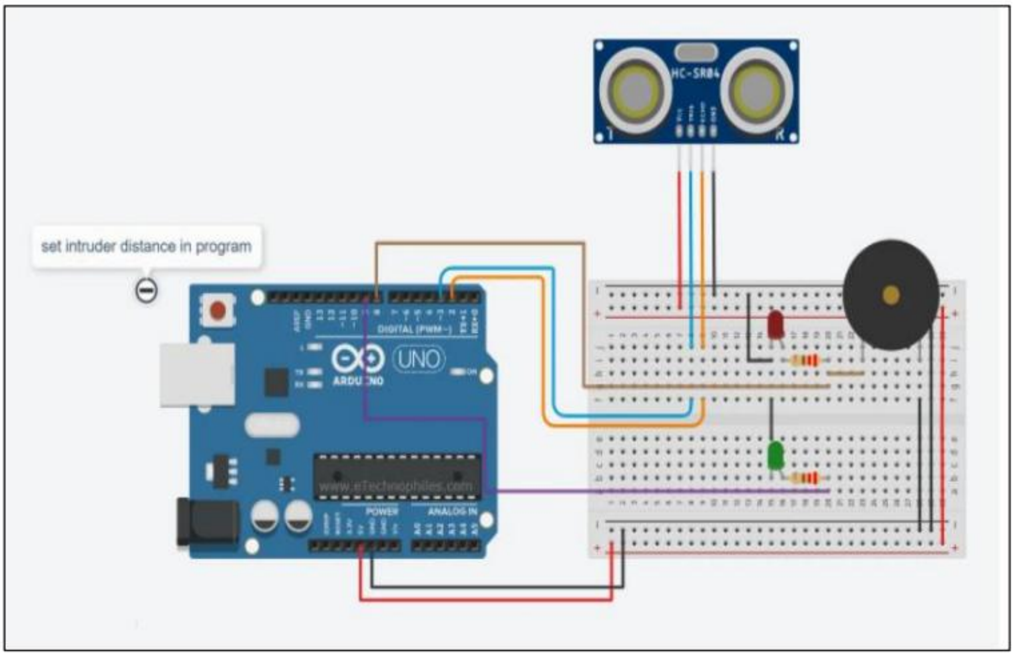
Buzzer

USB cable

Breadboard

Jumper wires

Connection Diagram



Steps of working

1. Insert the Ultrasonic sensor into your breadboard and connect its Echo pin to the digital pin 2 and the Trigger pin to digital pin 3 of the Arduino.
2. Insert the RED and Green LED into the breadboard. Attach the positive leg (the longer leg) of red LED to signal pin of the Buzzer via the 220-ohm resistor, and the negative leg to GND. The green LED is connected to digital pin 8 of the Arduino.
3. Upload the code.
4. Observe the LEDs and take some object in front of ultrasonic sensor.
5. Observe the changes in the LED and buzzer sound.

Program Code

```
/**********Intrusion Detection******/
```

```
#define echo 2
 #define trig 3
 #define outA 8 // Red LED
 #define outB 9 // Green LED
 #define outC 10 // Buzzer
 float duration; // time taken by the pulse to return back
 float distance; // one way distance travelled by the pulse
const int intruderDistance = 10; // the minimum distance up to which the
sensor is able to sense any object
 void setup() {
 pinMode(trig, OUTPUT);
 pinMode(echo, INPUT);
 pinMode(outA, OUTPUT);
 digitalWrite(outA, LOW);
 pinMode(outB, OUTPUT);
 digitalWrite(outB, LOW);
pinMode(outC, OUTPUT);
 digitalWrite(outC, LOW);
 Serial.begin(9600);
}
void loop() {
time_Measurement();
distance = (float)duration * (0.0343) / 2;
// calculate the one way distance travelled by the pulse
Serial.println(distance);
alarm_condition();
 }
 void time_Measurement()
 { // function to measure the time taken by the pulse to return back
```

```
 digitalWrite(trig, LOW);
 delayMicroseconds(2);
 digitalWrite(trig, HIGH);
 delayMicroseconds(10);
 digitalWrite(trig, LOW);
 duration = pulseIn(echo, HIGH);
 }
 void alarm_condition()
 { //function to execute the output commands based on the sensor
input
 if(distance<=intruderDistance)
 {
digitalWrite(outA,HIGH);
 digitalWrite(outB,LOW);
 analogWrite(outC,200);}
 else
 {
digitalWrite(outA,LOW);
digitalWrite (outB, HIGH);
analogWrite (outC,0);
}
}
```

Observation

Sr No.	Object Detected	LED	Buzzer
1			
2			

Result