SCTR's

PUNE INSTITUTE OF COMPUTER TECHNOLOGY

SECOND YEAR INFORMATION TECHNOLOGY (2019 COURSE)

LABORATORY MANUAL

FOR

PROGRAMMING SKILL DEVELOPMENT LABORATORY

SEMESTER - IV

[Subject code: 214455]

[Prepared By]

Mrs. Usha A. Jogalekar

Mr. Sandip R. Warhade

Ms. Reshma D. Kapadi

DEPARTMENT OF INFORMATION TECHNOLOGY

INSTITUTE VISION AND MISSION

VISION

Pune Institute of Computer Technology aspires to be the leader inhigher technical education and research of international repute.

MISSION

To be leading and most sought after Institute of education and research in emerging engineering and technology disciplines that attracts, retains and sustains gifted individuals of significant potential.

DEPARTMENT VISION AND MISSION

VISION

The department endeavors to be recognized globally as a center of academic excellence & research in Information Technology.

MISSION

To inculcate research culture among students by imparting information technology related fundamental knowledge, recent technological trends and ethics to get recognized as globally acceptable and socially responsible professionals.

Savitribai Phule Pune University, Pune

Second Year Information Technology (2019 Course)

214455: Programming Skill Development Lab

Teaching Scheme:	Credit Scheme:	Examination Scheme:
Theory(TH) :02hrs/week	01	PR: 25Marks TW: 25Marks

Prerequisites: Computer Organization and Architecture

Course Objectives:

1. To learn embedded C programming and PIC18FXXXmicrocontrollers.
2. To learn interfacing of real-world input and output devices to PIC18FXXX microcontroller

Course Outcomes:

On completion of this course student will be able to --

CO1: Apply concepts related to embedded C programming.

CO2: Develop and Execute embedded C program to perform array addition, block transfer, sorting operations

CO3: Perform interfacing of real-world input and output devices to PIC18FXXX microcontroller.

CO4: Use source prototype platform like Raspberry-Pi/Beagle board/Arduino.

Guidelines for Instructor's Manual

The faculty member should prepare the laboratory manual for all the experiments and it should be made available to students and laboratory instructor/Assistant. The instructor's manual should include prologue, university syllabus, conduction & Assessment guidelines, topics under consideration- concept, objectives, outcomes, algorithm, sample test cases etc.

Guidelines for Student's Lab Journal

1. The laboratory assignments should be submitted by students in the form of journal. The Journal consists of Certificate, table of contents, and write-up of each assignment (Title, Objectives, Problem Statement, Outcomes, software & Hardware requirements, Date of Completion, Assessment grade/marks and assessor's sign, Theory- Concept, circuit diagram, pin configuration, conclusion/analysis).
2. As a conscious effort and little contribution towards Green IT and environment awareness, attaching printed papers as part of program listing to journal may be avoided.
3. Use of Digital media like shared drive containing students' programs maintained by lab In-charge is highly encouraged.
4. Practical Examination will be based on the term work submitted by the student in the form of journal.
5. Candidate is expected to know the theory involved in the experiment.
6. The practical examination should be conducted if the journal of the candidate is completed in all respects and certified by concerned faculty and head of the department.
7. All the assignment mentioned in the syllabus must be conducted.

Guidelines for Lab /TW Assessment

1. Examiners will assess the term work based on performance of students considering the parameters such as timely conduction of practical assignment, methodology adopted for

implementation of practical assignment, timely submission of assignment in the form of write-up along with results of implemented assignment, attendance etc.

2. Examiners will judge the understanding of the practical performed in the examination by asking some questions related to theory & implementation of experiments he/she has carried out.
3. Necessary knowledge of usage of software and hardware of PIC18FXXX microcontrollers and its interfacing kits should be checked by the concerned faculty members.

Guidelines for Laboratory Conduction

The instructor is expected to frame the assignments by understanding the prerequisites, technological aspects, utility and recent trends related to the topic. The instructor may set multiple sets of assignments and distribute among batches of students. It is appreciated if the assignments are based on real world problems/applications.

Guidelines for Practical Examination

Both internal and external examiners should jointly set problem statements for practical examination. During practical assessment, the expert evaluator should give the maximum weightage to the satisfactory implementation of the problem statement. The supplementary and relevant questions may be asked at the time of evaluation to judge the student's understanding of the fundamentals, effective and efficient implementation. The evaluation should be done by both external and internal examiners.

Suggested List of Laboratory Assignments

Suggested List of Laboratory Assignments Group A (Any Three):

Mapping of Course Outcomes for Group A -- CO1 , CO2

1. Study of Embedded C programming language (Overview, syntax, One simple program like addition of two numbers).
2. Write an Embedded C program to add array of n numbers.
3. Write an Embedded C program to transfer elements from one location to another for following:

i) Internal to internal memory transfer

ii) Internal to external memory transfer

4. Write an Embedded C menu driven program for :

i) Multiply 8 bit number by 8 bit number

ii) Divide 8 bit number by 8 bit number

5. Write an Embedded C program for sorting the numbers in ascending and descending order.

Group B (Any Three):

Mapping of Course Outcomes for Group B -- CO3

6. Write an Embedded C program to interface PIC 18FXXX with LED & blinking it using specified delay.
7. Write an Embedded C program for Timer programming ISR based buzzer on/off.
8. Write an Embedded C program for External interrupt input switch press, output at relay.
9. Write an Embedded C program for LCD interfacing with PIC 18FXXX.

Group C (Any two):

Mapping of Course Outcomes for Group C -- CO3

10. Write an Embedded C program for Generating PWM signal for servo motor/DC motor.

11. Write an Embedded C program for PC to PC serial communication using UART.

12. Write an Embedded C program for Temperature sensor interfacing using ADC & display on LCD.

Group D:

Mapping of Course Outcomes for Group D -- CO4

13. Study of Arduino board and understand the OS installation process on Raspberry-pi.

14. Write simple program using Open source prototype platform like Raspberry-Pi/Beagle board/Arduino for digital read/write using LED and switch Analog read/write using sensor and actuators.

Reference Books :

1. Mazidi, Rolin McKinlay and Danny Causey, 'PIC Microcontroller and Embedded Systems using Assembly and C for PIC18", Pearson Education
2. "Raspberry Pi for Beginners", 2nd Edition book" e-book.
3. Peatman, John B, "Design with PIC Microcontroller", Pearson Education PTE,
4. Ramesh Gaonkar, "Fundamentals of Microcontrollers and Applications In Embedded Systems (with the PIC18 Microcontroller Family)"Thomson/Delmar Learning; 1 edition (January 8, 2007), ISBN:978-1401879143.

GROUP A: ASSIGNMENTS

Experiment No: 01

Title: Study of Embedded C programming language (Overview, syntax, One simple program like addition of two numbers).

Objective: To learn embedded C programming and PIC18F458 microcontroller.

Outcome: On completion of this Assignment student will be able to

- Understand the concepts of embedded C programming
- Develop and Execute embedded C program to perform simple operations on integers.

Prerequisites: Computer Organization and Architecture

Lab facility:MPLAB X IDE simulator, XC8 Compiler

Theory:

Embedded C programming plays a key role to make the microcontroller run & perform the preferred actions. At present, we normally utilize several electronic devices like mobile phones, washing machines, security systems, refrigerators, digital cameras, etc. The controlling of these embedded devices can be done with the help of an embedded C program. For example, in a digital camera, if we press a camera button to capture a photo then the microcontroller will execute the required function to click the image as well as to store it.
Embedded C programming builds with a set of functions where every function is a set of statements that are utilized to execute some particular task. Both the embedded C and C languages are the same and implemented through some fundamental elements like a variable, character set, keywords, data types, declaration of variables, expressions, statements. All these elements play a key role while writing an embedded C program.

In embedded system programming, C is preferred over other languages. Due to the following reasons:

- Easy to understand
- HighReliability
- Portability
- Scalability

Steps to Build an Embedded C Program

There are different steps involved in designing an embedded c program like the following.

- Multiline Comments Denoted using/*……*/
- Single Line Comments Denoted using//
- Preprocessor Directives #include<…> or #define
- Global Variables Accessible anywhere in the program
- Function Declarations Declaring Function
- MainFunction Main Function, execution begins here
  {
  Local Variables Variables confined to main function
  Function Calls Calling other Functions
  Infinite Loop Like while(1) or for(;;)
  Statements . . . . .

….
….
}

- Function Definitions Defining the Functions
  {
  Local Variables Local Variables confined to this Function
  Statements . . . . .
  ….
  ….
  }

Main Factors of Embedded C Program

The main factors to be considered while choosing the programming language for developing an embedded system include the following.

Program Size

Every programming language occupies some memory where embedded processor like microcontroller includes an extremely less amount of random-access memory.

Speed of the Program

The programming language should be very fast, so should run as quickly as possible. The speed of embedded hardware should not be reduced because of the slow-running software.

Portability

For the different embedded processors, the compilation of similar programs can be done.

- Simple Implementation
- Simple Maintenance
- Readability

Advantages

- It is very simple to understand.
- It executes simply a single task at once
- The cost of the hardware used in the embedded c is typically low.
- The applications of embedded are extremely appropriate in industries.
- It takes less time to develop an application program.
- It reduces the complexity of the program.
- It is easy to verify and understand.
- It is portable from one controller to another.

Disadvantages

- At a time, it executes only one task but can't execute the multi-tasks
- If we change the program then need to change the hardware as well
- It supports only the hardware system.
- It has a scalability issue
- It has a restriction like limited memory otherwise compatibility of the computer.

Applications of Embedded C Program

- Embedded C programming is used in industries for different purposes

- The programming language used in the applications is speed checker on the highway, controlling of traffic lights, controlling of street lights, tracking the vehicle, artificial intelligence, home automation, and auto intensity control.

What is MPLAB?

MPLAB is a software program that runs on your PC to provide a development environment for your embedded system design. In other words, it is a program package that makes writing and developing a program easier. It could best be described as developing environment for a standard program language that is intended for programming microcontrollers.

Get started to**MPLAB X Programming IDE.**

Step1: Creating a new project

- Go to the FileTab.
- Click on NewProject.
- Step1: Choose Project:
- Select: Microchip Embedded -> Standalone Project.ClickNext.

Step2: Select Device:

- Select: Family -> Advanced 8 Bit MCU(PIC18).
- Select: Device: PIC18F458. ClickNext.

Step3: Select Tool: Simulator. Click Next.**Step4**: Select Compiler ->XC8. Click Next.**Step5**: Select Project Name and Folder.

- Give Project Name.
- Select project Location using BrowseButton.
- Uncheck Set as main projectoption.
- Click Finish.

Step6: Creating a new Source file and Header File.

- Go to the Project location in the Projectwindow.
- Click the + sign to open the projectspace.
- Right Click on the Source Files folder (for a C file) andHeaderfiles (for a .hfile).
- New - > C Source file / or C HeaderFile.

Step7: Opening an existing project.

- Go to the FileTab.
- Select OpenProject.
  Browse to the location and select the project name.X file (project file). Click on Open Project

Input: Two numbers
Output: Addition/subtraction operations performed on given numbers

Write – up:

a. Draw Block diagram and pin diagram of PIC18F458.

b. Explain STATUS register.

c. Write a note on MPLAB.

d. Consider two 8-bit numbers stored in registers TMR0 and TMR1. Calculate TMR2=TMR0+TMR1. Choose the values such that after addition STATUS register is

i. 00H
ii. 04H
iii. 10H
iv. 05H
v. 18H
vi. 01H
vii. 02H
viii. 03H
ix. 09H
x. 13H
xi. 11H

e. Repeat d for subtraction operation.

Conclusion: Based on the understanding of the basic concepts of Embedded C Programming along with its environment setup used in a simple program for addition of twonumbers.

Experiment No: 02

Title: Write an Embedded C program to add an array of n numbers.

Objective: To calculate the sum of the array elements.

Outcome: On completion of this Assignment student will be able to

- Understand the concepts of embedded ed C programming
- Develop and Execute embedded C program to perform array addition

Pre-requisites : Fundamental knowledge of Microcontroller, Logic and knowing the basics of C language

Lab facility: MPLAB X IDE simulator, XC8 Compiler, PIC18FXXXmicrocontroller kit

Theory:

There is actually not much difference between C and Embedded C apart from few extensions and the operating environment. Both C and Embedded C are ISO Standards that have almost same syntax, datatypes, functions, etc.

Embedded C is basically an extension to the Standard C Programming Language with additional features like Addressing I/O, multiple memory addressing and fixed- point arithmetic, etc. C Programming Language is generally used for developing desktop applications, whereas Embedded C is used in the development of Microcontroller based applications.
C arrays are declared in the following form type name [number ofelements]; For example, if we want an array of five integers , we write inC:
int numbers[5];

For a five character array → char letters[5];

type name [number of elements]={comma-separated values}
For example, if we want to initialize an array with five integers, with 1, 3, 5, 0, 9, as the initial values: int number[5]={1,3,5,0,9};

Let's see the logic to calculate the sum of the array elements. Suppose **arr** is an integer array of size N (arr[N]), the task is to write the C Program to sum the elements of an array.

Logic to calculate the sum of the array elements:

1. Create an intermediate variable 'sum'.
2. Initialize the variable 'sum' with 0.
3. To find the sum of all elements, iterate through each element, and add the current element to the sum.

```
sum = sum + arr[i];
```

Input: Array of size N (arr[N])

Output: Sum of the elements of an array

Conclusion: We have implemented sum of elements of an array using embedded C programming.

Experiment No: 03

Title: Write an Embedded C menu driven program for:

i) Multiply 8-bit number by 8-bitnumber

ii) Divide 8-bit number by 8-bitnumber

Objective: To learn arithmetic operation i.e., multiplication and division in embedded C programming and PIC18FXXXmicrocontrollers.

Outcome: On completion of this Assignment student will be able to

- Understand the concepts of embedded C programming
- Develop and Execute embedded C program to perform multiplication and division

Pre-requisites: Fundamental knowledge of Microcontroller, Logic and knowing the basics of C language

Lab facility : MPLAB X IDE simulator, XC8 Compiler, PIC18FXXXmicrocontroller kit

Theory:

Embedded C Programming Language, which is widely used in the development of Embedded Systems, is an extension of C Program Language. The Embedded C Programming Language uses the same syntax and semantics of the C Programming Language like main function, declaration of datatypes, defining variables, loops, functions, statements, etc.

The extension in Embedded C from standard C Programming Language include I/O Hardware Addressing, fixed point arithmetic operations, accessing address spaces, etc.

C examples – with standard arithmetic operators

Input:8-bit number stored in SFR

Output: Result of multiplication stored in 2 SFRs

Result of division (quotient and remainder) stored in 2 SFRs

Conclusion: Arithmetic operations like multiplication and division of 8-bit number with 8-bit number are studied here

GROUP B: ASSIGNMENTS

Experiment No: 04

Title: Write an Embedded C program to interface PIC 18FXXX with LED & blinking it using specified delay.

Objective: Connect LED to the microcontroller and it should start blinking.

Outcome: On completion of this Assignment student will be able to

- Understand the concepts writing PIC program using embedded C programming and write the .exe code on the kit or use the Proteus software to blink the LED (real or virtual)

Pre-requisites: Embedded C programming, MPLAB, Proteus

Lab facility: MPLAB X IDE simulator, XC8 Compiler, PIC18FXXX microcontroller kit, Proteus, LED

Theory: Timer is used to control the output on a bit of a port.

There are 4 timers in PIC18. Each have corresponding timer registers TMRxH and TMRxL where x is the timer number that ranges from 0 to 3

There is a control register correspond to every timer TxCON. In this program Timer 0 is used. Following is the T0CON register.

TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0

TMR0ON: Timer0 On/Off Control bit

1 = Enables Timer0
0 = Stops Timer0

T08BIT: Timer0 8-bit/16-bit Control bit

1 = Timer0 is configured as an 8-bit timer/counter
0 = Timer0 is configured as a 16-bit timer/counter

T0CS: Timer0 Clock Source Select bit

1 = Transition on T0CKI pin
0 = Internal instruction cycle clock (CLKO)

T0SE: Timer0 Source Edge Select bit

1 = Increment on high-to-low transition on T0CKI pin
0 = Increment on low-to-high transition on T0CKI pin

PSA: Timer0 Prescaler Assignment bit

1 = TImer0 prescaler is not assigned. Timer0 clock input bypasses prescaler.
0 = Timer0 prescaler is assigned. Timer0 clock input comes from prescaler output.

T0PS2:T0PS0: Timer0 Prescaler Select bits

111 = 1:256 Prescale value
110 = 1:128 Prescale value
101 = 1:64 Prescale value
100 = 1:32 Prescale value
011 = 1:16 Prescale value
010 = 1:8 Prescale value
001 = 1:4 Prescale value
000 = 1:2 Prescale value

The program is written such that a LED is connected to a pin of any port (PORTB)

and that pin is toggled after a cycle controlled by the timer. This is repeated for infinite time. Thus when the port bit is 0, LED is off and when it is 1, LED is on. So the LED blinks after every cycle.

ALGORITHM:

Step1: Reset TRISB so that PORTB is in output mode Step2:

Reset PORTB

Step3: Call the delay

Step4: Set PORTB

Step5: goto Step3

Input: make all the required connections

Output: LED starts blinking with an equal interval

Conclusion: The student is able to understand the working of the timer, LED, and ports.

Experiment No: 05

Title: Write an Embedded C program for Timer programming ISR based buzzer on/off.

Objective: Connect buzzer to the microcontroller and it should be on or off as per the user input.

Outcome: On completion of this Assignment student will be able to

- Understand the concepts writing PIC program using embedded C programming and write the .exe code on the kit or use the Proteus software to switch the buzzer on or off (real orvirtual)

Pre-requisites: Embedded C programming, MPLAB, Proteus

Lab facility: MPLAB X IDE simulator, XC8 Compiler, PIC18FXXX microcontroller kit, Proteus, Buzzer

Theory: Timer is used to generate the delay for which the buzzer should be on. In this program Timer 1 is used. Following is the T1CON register.

RD16	---	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON

RD16 D7 16-bit read/write enable bit
1 = Timer1 16-bit is accessible in one 16-bit operation.
0 = Timer1 16-bit is accessible in two 8-bit operations.

D6 Not used

T1CKPS2:T1CKPS0 D5 D4 Timer1 prescaler selector
0 0 = 1:1 Prescale value
0 1 = 1:2 Prescale value
1 0 = 1:4 Prescale value
1 1 = 1:8 Prescale value

T1OSCEN D3 Timer1 oscillator enable bit
1 = Timer1 oscillator is enabled.
0 = Timer1 oscillator is shutoff.

T1SYNC D2 Timer1 synchronization (used only when TMR1CS = 1 for counter mode to synchronize external clock input)
If TMR1CS = 0 this bit is not used.

TMR1CS D1 Timer1 clock source select bit
1 = External clock from pin RC0/T1CKI
0 = Internal clock (Fosc/4 from XTAL)

TMR1ON D0 Timer1 ON and OFF control bit
1 = Enable (start) Timer1
0 = Stop Timer1

The program is written such that a buzzer is connected to a pin of any port (PORTB) and that pin is set when the switch is pressed. The duration for which the pin should be set is controlled by the timer. This is repeated for infinite time.

ALGORITHM:

Step1: Reset TRISB so that PORTB is in output mode Step2:

Reset PORTB

Step3: check the switch is pressed

Step4: if yes then

Step5: Set PORTB

Step6: Call the delay

Step7:end if

Step8: goto Step3

Input: make all the required connections

Output: The buzzer is on when the switch is pressed.

Conclusion: The student is able to understand the working of the timer, buzzer, and ports.

Experiment No: 06

Title: Write an Embedded C program for External interrupt input switch press, output at relay.

Objective: To understand the working of interrupts in PIC18

Outcome: When PIC18 is interrupted, using RB0, the data at PORTD will be toggled

Pre-requisites: Embedded C programming, MPLAB, Proteus

Lab facility: MPLAB X IDE simulator, XC8 Compiler, PIC18FXXX microcontroller kit

Theory:

Sometimes External devices are connected with microcontroller. If that external device has to send some information to microcontroller, then microcontroller needs to know about this situation to get that information. An example of such an external device is the digital thermometer. It measures the temperature and at the end of measurements transmits results to the microcontroller. Now the purpose of this article to explain the fact that how does the microcontroller knows to get the required information from an external device.

Types of interrupts

There are two methods of communication between the microcontroller and the external device:

- By using Polling
- By using Interrupts

INTERRUPTS

Interrupt is the signal which is sent to the microcontroller to mark the event that requires immediate attention. This signal requests the microcontroller to stop to perform the current program temporarily time to execute a special code. It means when external device finishes the task imposed on it, the microcontroller will be notified that it can access and receive the information and use it.

INTERRUPT SOURCES in microcontrollers

The request to the microcontroller to stop to perform the current program temporarily can come from various sources:

- Through external hardware devices like pressing specific key on the keyboard, which sends Interrupt to the microcontroller to read the information of the pressed key.
- During execution of the program, the microcontroller can also send interrupts to itself to report an error in the code. For example, division by 0 will causes an interrupt.
- In the multi-processor system, the microcontrollers can send interrupts to each other to communicate. For example, to divide the work between them they will send signals between them.

INTERRUPT TYPES in pic microcontrollers

There are 2 types of interrupts for PIC microcontroller that can cause break.

Software Interrupt: It comes from a program that is executed by microcontroller or we can say that it is generated by internal peripherals of the microcontroller and requests the processor to hold the running of program and go to make an interrupt.

Hardware Interrupt: These interrupts are sent by external hardware devices at certain pins of microcontroller.

Following interrupts sources are present in PIC18F452

- **Timer overinterrupt**
- Pins RB0, RB1, RB2 for external hardware interrupts (INT0, INT1,INT2)
- PORTB Change interrupts (any one of the upper four Port B pins.RB4-RB7)
- ADC (**analog-to-digital converter**)Interrupt
- CCP (compare capture pulse-width-modulation)Interrupt
- **Serial communication's USARTinterrupts (receive and**transmit)
- Reset, Brown-Out Reset, Watch-dog Reset, Power OnReset
- Parallel Port Read/WriteInterrupt
- Master Synchronous Serial PortInterrupt
- Data**EEPROMWrite Complete**Interrupt

REGISTER CONFIGURATION for external interrupt

These are the registers for interrupt operation and minimum 1 register can be used to control the interrupt operation in PIC18F452 which are:

- RCON (Reset ControlRegister)
- INTCON, INTCON2, INTCON3 (Interrupt ControlRegisters)
- PIR1, PIR2 (Peripheral Interrupt RequestRegisters)
- PIE1, PIE2 (Peripheral Interrupt EnableRegisters)

RCON Register:

- Reset controlregister
- IPEN bit to enable interrupt priority scheme, 1= enable priority leveloninterrupts
- Other bits used to indicate the cause ofresetRI (Reset Instruction flag), TO (Watchdog Time Out flag), PD (Power on Detection flag), POR (Power on Reset status) and BOR (Brown Out Reset status bit)

INTCON Register:

- 3 Interrupt control registers INTCON, INTCON2,INTCON3
- Readable and writable register which contains various enable and flagbits
- Interrupt flag bits get set when an interrupt conditionoccurs
- Contain enable, priority and flag bits for external interrupt, port B pin change and TMR0 overflow interrupt

PIE Register:

- Peripheral Interrupt Enable register
- May be multiple register (PIE1, PIE2), depending on the number of peripheral interrupt sources
- Contain the individual bits to enable/disable Peripheral interrupts for use

PIR Register:

- Peripheral Interrupt Flag register
- May be multiple register (PIR1, PIR2), depending on the number of peripheral interrupt sources
- Contain bits to identify which interrupt occurs(flags)
- Corresponding bits are set when the interrupt occurred

EXTERNAL INTERRUPT registers setting

INTCON registers are just used to configure the external PIC interrupts.

INTCON REGISTER:

GIE: Global InterruptEnable

This bit is set high to enable all interrupts of PIC18F452. 1
= Enable all interrupts
0 = Disable all interrupts

PEIE: Peripheral Interrupt Enable

This bit is set high to enable all the peripheral interrupts (Internal interrupts) of the microcontroller.
1 = Enable all peripheral interrupts 0
= Disable all peripheral interrupts

T0IE: TMR0 Overflow Interrupt Enable

This bit is set high to enable the External Interrupt 0. 1
= Enable TMR0 overflow interrupt
0 = Disable TMR0 overflow interrupt

INTE: INT External Interrupt Enable

This bit is set high to enable the external interrupts. 1
= Enables the INT external interrupt
0 = Disables the INT external interrupt

RBIE: RB Interrupt Enable

This bit is set high to enable the RB Port Change interrupt pin. 1 =
Enables the RB port change interrupt
0 = Disables the RB port change interrupt

T0IF: TMR0 Overflow Interrupt Flag

1 = TMR0 register has overflowed (it must be cleared in software) 0 =
TMR0 register has not overflowed

INTF: INT External Interrupt Flag

1 = The INT external interrupt occurred (it must be cleared in software) 0 =
The INT external interrupt did not occur

RBIF: RB Port Change Interrupt Flag

1 = At least one of the RB7:RB4 pins changed the state (must be cleared in software)
0 = None of RB7:RB4 pins have changed the state

ALGORITHM:

In this program, we configure the External Interrupt 0 (INT0). PORT D is used as output port and it is monitored through a set of 8 LEDs. Reset is given through pin 1.

Push button is connected to RB0 for external interrupt source.

Step1: Enable the External Interrupt 0 by setting INT0IE bit high (INTCON=0x10).
Step2: Set the interrupt on falling edge by setting the INTEDG0 in INTCON2 to zero (INTCON2=0)
Step3: Set the Global Interrupt Enable in INTCON to high (INTCON.GIE=1) Step4: Initialize PORTD with certain value (LATD=0xAA)
Step5: Write the ISR (Interrupt Service Routine) for the interrupt
Step6: Clear the INT0IF bit of INTCON (INTCON.INT0IF=0) Step7: Invert or toggle the value at PORTD (LATD=~LATD) Step8: go to step3.

Input: press the push button
Output: the LED's connected to the PORTD are toggled when the button is pressed. Different patterns should be generated using LEDs.

Conclusion: The student is able to understand the working of the interrupts and able to write the ISR (Interrupt Service Routine).

Experiment No: 07

Title: Write an Embedded C program for Generating PWM signal for servo motor/DC motor.

Objective: To study DC motor control with PIC18F PWM concept.

Outcome: On completion of this Assignment student will be able to understand the controlling of DC motor speed using PWM technique of PIC18F microcontroller.

Pre-requisites: Embedded C programming, MPLAB X IDE

Lab facility: MPLAB X IDE simulator, XC8 Compiler, PIC18FXXX microcontroller kit, DC motor.

Theory:

Pulse Width Modulation (PWM) is a technique by which the width of a pulse is varied while keeping the frequency of the wave constant.

A period of a pulse consists of an **ON** cycle (5V) and an **OFF** cycle (0V). The fraction for which the signal is ON over a period is known as a **duty cycle**.

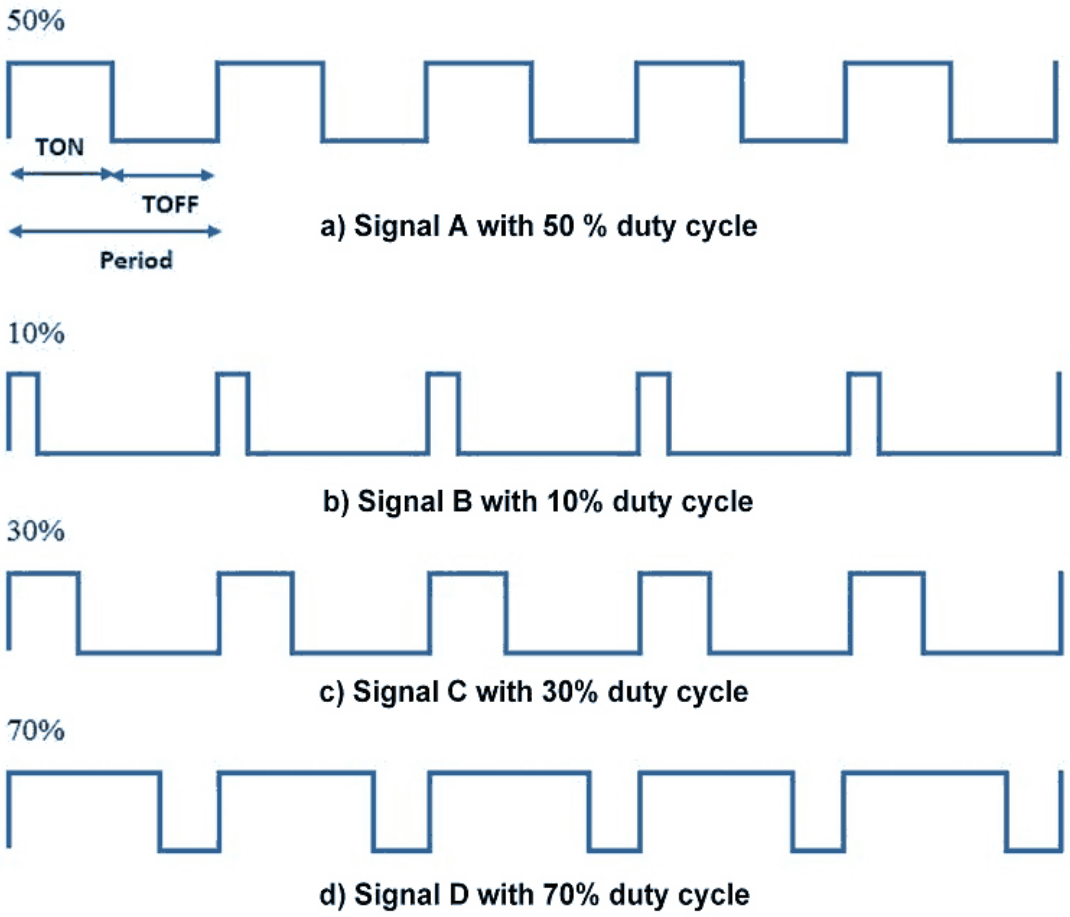
$$DutyCycle(percentage) = \frac{Ton}{Total\,Period} * 100$$

E.g. A pulse with a period of 10ms will remain ON (high) for 2ms.Therefore, the duty cycle will be

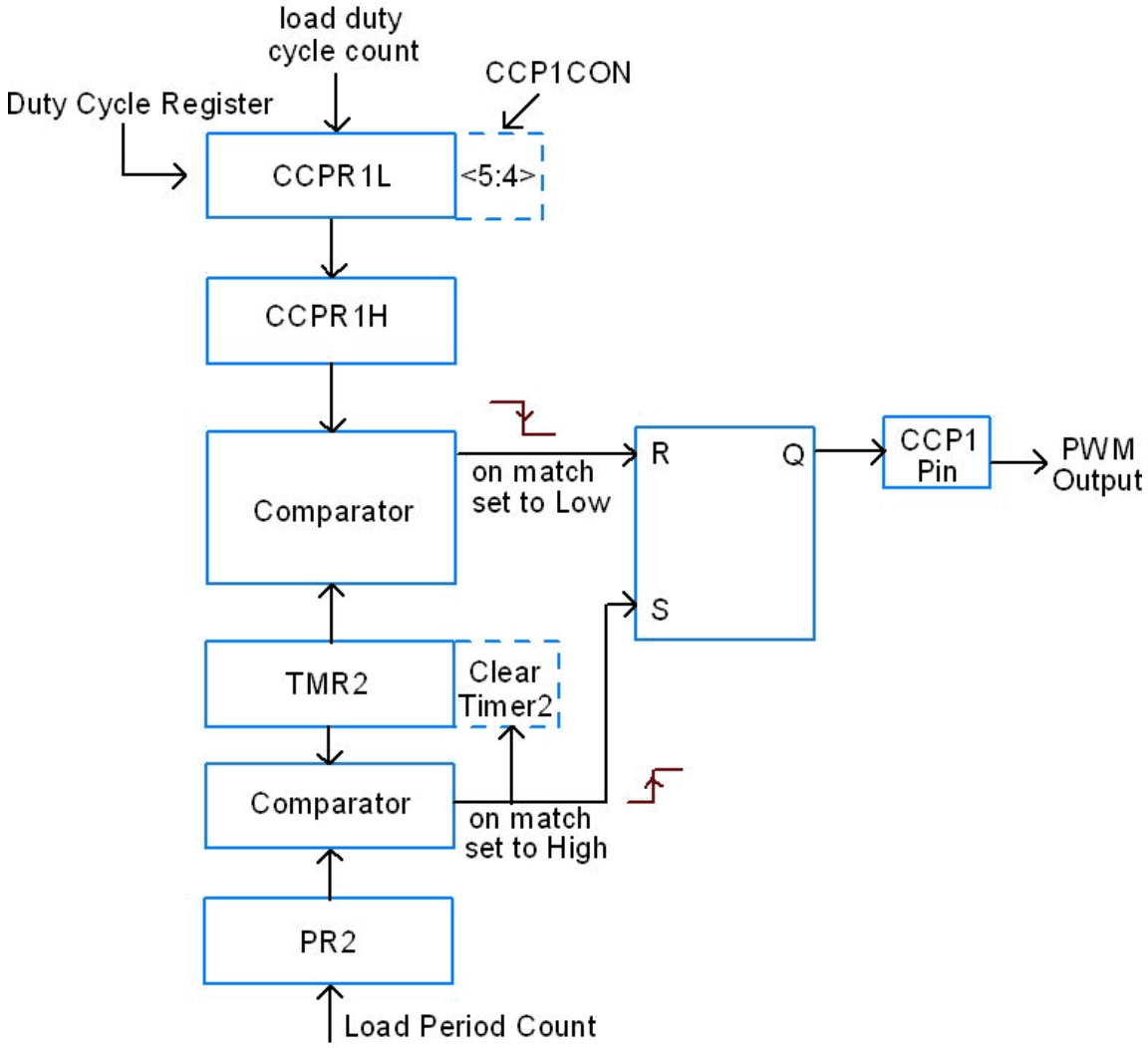
D = 2ms / 10ms = 20%

Through the PWM technique, we can control the power delivered to the load by using the ON-OFF signal. The PWM signals can be used to control the speed of DC motors and to change the intensity of the LED. Moreover, it can also be used to generate sine signals.

Pulse Width Modulated signals with different duty cycle are shown below



PIC18F4550 controller has an in-built 10-bit PWM module known as the CCP module. The pin CCP1 (RC2) is used for generating PWM signals. It needs to be configured as an output.



Working of PWM in CCP module

Steps for Programming

1. Load the PR2 value which will decide the period of thepulse.
2. Set the duty cycle by loading value in the CCPR1L: CCP1CON<5:4>
3. Configure the CCP1CON register for setting a PWMmode.
4. Initialize the pin CCP1as an output pin which will give PWM output.
5. Configure the T2CON register and enable TMR2 using T2CON

Input: Make all the connections and start

Output: DC motor shaft will rotate with different speeds.

Conclusion: The student is able to understand the working of the PIC18F micro controllers PWM technique to control speed of DC motor.

Experiment No: 08

Title: Write an Embedded C program for LCD interfacing with PIC 18FXXX.

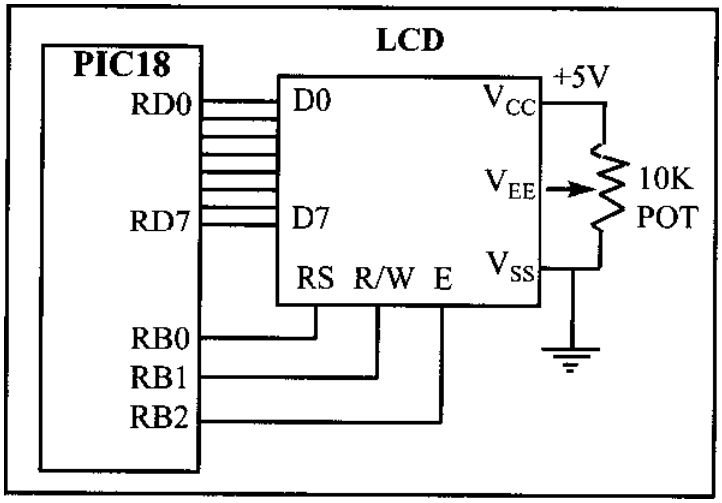
Objective: To study LCD connection with PIC18 and communication.

Outcome: On completion of this Assignment student will be able to understand the communication from PIC18 with LCD

Pre-requisites: Embedded C programming, MPLAB, Proteus

Lab facility: MPLAB X IDE simulator, XC8 Compiler, PIC18FXXX microcontroller kit, LCD, Proteus

Theory:



The above figure shows the connection of LCD with PIC18.

The resistor R1 is used for giving the contrast to the LCD. The crystal oscillator of 12 MHz is connected to the OSC1 and OSC2 pins of Pic microcontroller PIC18F4550 for system clock. The capacitor C2 and C3 will act filters to the crystal oscillator. You can use different ports or pins for interfacing the LCD before going to different ports please check the data sheet whether the pins for general purpose or they are special function pins.

1	V_{SS}	--	Ground
2	V_{CC}	--	+5 V power supply
3	V_{EE}	--	Power supply to control contrast
4	RS	I	RS = 0 to select command register, RS = 1 to select data register
5	R/W	I	R/W = 0 for write, R/W = 1 for read
6	E	I/O	Enable
7	DB0	I/O	The 8-bit data bus
8	DB1	I/O	The 8-bit data bus
9	DB2	I/O	The 8-bit data bus
10	DB3	I/O	The 8-bit data bus
11	DB4	I/O	The 8-bit data bus
12	DB5	I/O	The 8-bit data bus
13	DB6	I/O	The 8-bit data bus
14	DB7	I/O	The 8-bit data bus

The above figure shows pin description of LCD.

1	Clear display screen
2	Return home
4	Decrement cursor (shift cursor to left)
6	Increment cursor (shift cursor to right)
5	Shift display right
7	Shift display left
8	Display off, cursor off
A	Display off, cursor on
C	Display on, cursor off
E	Display on, cursor blinking
F	Display on, cursor blinking
10	Shift cursor position to left
14	Shift cursor position to right
18	Shift the entire display to the left
1C	Shift the entire display to the right
80	Force cursor to beginning of 1st line
C0	Force cursor to beginning of 2nd line
38	2 lines and 5x7 matrix

The above figure shows the hex codes and their corresponding instruction to LCD register. The characters are sent ot the LCD without checking the busy flag. We need to wait 5-10 ms between issuing each character to the LCD. In programming an LCD we need a long delay for the power-up process.

Initializing the LCD function:

lcdcmd(0x38);//Configure the LCD in 8-bit mode,2 line and 5×7 font
lcdcmd(0x0C);// Display On and Cursor Off

lcdcmd(0x01);// Clear display screen
lcdcmd(0x06);// Increment cursor
lcdcmd(0x80);// Set cursor position to 1st line,1st column

Sending command to the LC:

- rs=0; Register select pin islow.
- rw=0; Read/write Pin is also for writing the command to the LCD.
- en=1;enable pin ishigh.

Sending data to the LCD:

- rs=1; Register select pin ishigh.
- rw=0; Read/write Pin is also for writing the command to theLCD.
- en=1; enable pin ishigh.

ALGORITHM:

Step1: Reset TRISD

Step2: Reset TRISE

Step3: store the messages in two character arrays

Step4: store 0x0F in ADCON1 register

Step5: initialize LCD Step6:

call the delay Step7: write

the first string Step8: call

the delay

Step9: move the cursor to the second line in LCD

Step10: write the second string

Input: Make all the connections and start

Output: two input strings are displayed on the LCD.

Conclusion: The student is able to understand the working of the PIC18 with LCD and able to write the codes for the initialization of LCD, writing to LCD etc.

Experiment No:10

Title: Study of Arduino board and understand the OS installation process on Raspberry-pi.

Objective:1) To study different Arduino families and Arduino UNO.
2) To study OS installation process on Raspberry-pi.

Outcome: Introductory understanding of open source hardware Arduino and Raspberry-pi operating system installation.

Theory:

Arduino hardware, including the Nano, MKR and Classic families.

Classic Family

In the Classic Family, you will find boards such as the legendary Arduino UNO and other classics such as the Leonardo & Micro. These boards are considered the backbone of the Arduino project, and has been a success for many years (and more to come).

Arduino UNORev3	Arduino Mega2560Rev3	Arduino Leonardo	Arduino UNO MiniLimited Edition

Arduino Due	Arduino Micro	Arduino Zero	Arduino UNO WiFi Rev2

Nano Family

The Nano Family is a set of boards with a tiny footprint, packed with features. It ranges from the inexpensive, basic Nano Every, to the more feature-packed Nano BLE Sense / Nano RP2040 Connect that has Bluetooth® / Wi-Fi radio modules.
These boards also have a set of embedded sensors, such as temperature/humidity, pressure, gesture, microphone and more. They can also be programmed with MicroPython and supports Machine Learning.

Arduino Nano 33 IoT	Arduino Nano RP2040Connect		Arduino Nano BLE Sense
Arduino Nano 33BLE	Arduino NanoEvery	Arduino Nano	Arduino Nano MotorCarrier

MKR Family

The MKR Family is a series of boards, shields & carriers that can be combined to create amazing projects without any additional circuitry. Each board is equipped with a radio module (except MKR Zero), that enables Wi-Fi, Bluetooth®, LoRa®, Sigfox, NB-IoT communication. All boards in the family are based on theCortex-M032-bit SAMD21low power processor, and are equipped with a crypto chip for secure communication.

The MKR Family shields & carriers are designed to extend the functions of the board: such as environmental sensors, GPS, Ethernet, motor control andRGB matrix.

Arduino Unois a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator (CSTCE16M0V53-R0), a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.. You can tinker with your Uno without worrying too much about doing something wrong, worst case scenario you can replace the chip for a few dollars and start over again.

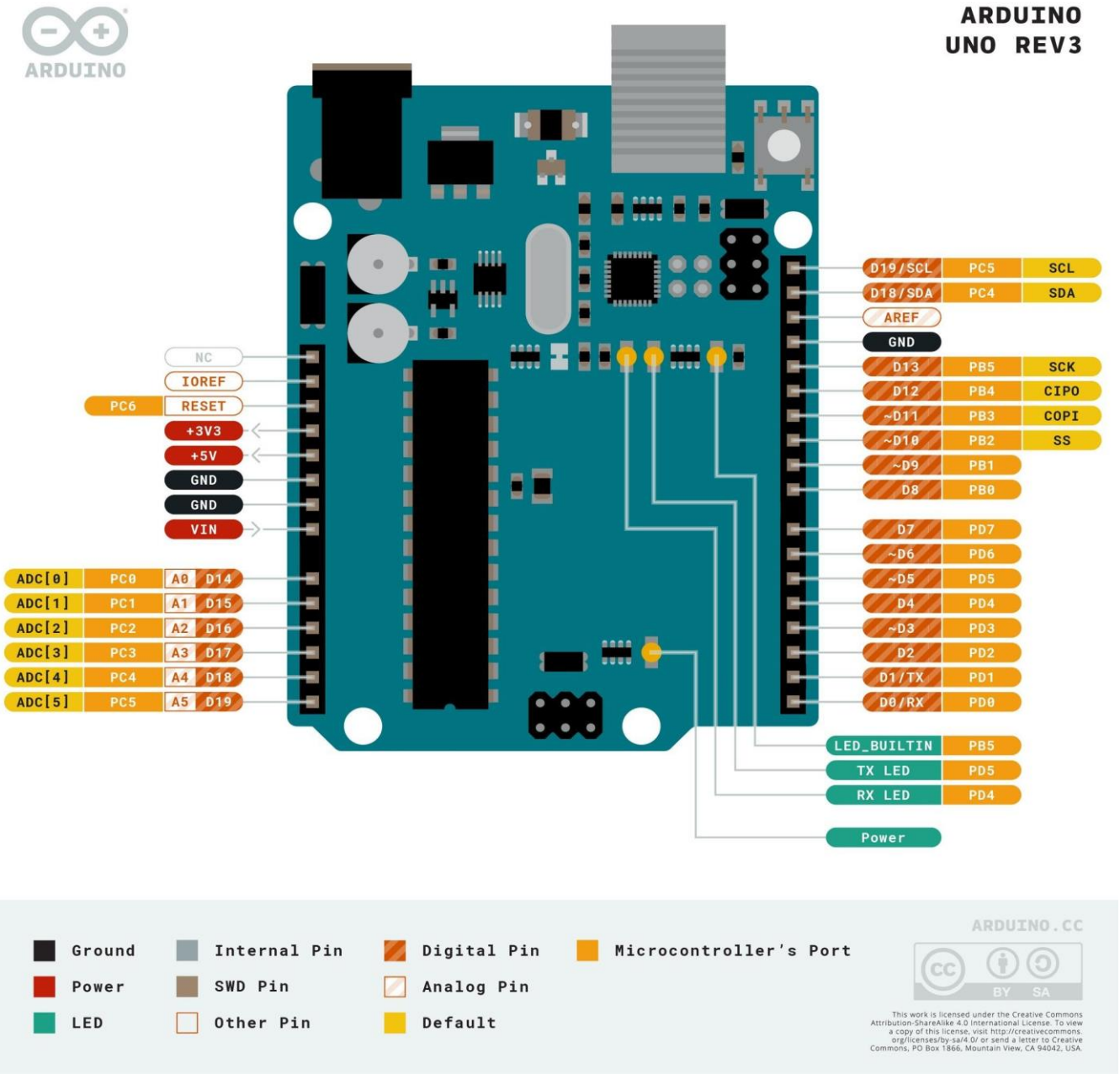
"Uno" means one in Italian and was chosen to mark the release of ArduinoSoftware (IDE) 1.0. The Uno board and version 1.0 of Arduino Software (IDE) were the reference versions of Arduino, now evolved to newer releases. The Uno board is the first in a series of USB Arduino boards, and the reference model for the Arduino platform; for an extensive list of current, past or outdated boards see the Arduino index ofboards.

Arduino UNO Tech specs

MICROCONTROLLER	ATmega328P
OPERATINGVOLTAGE	5V
INPUT VOLTAGE(RECOMMENDED)	7-12V
INPUTVOLTAGE(LIMIT)	6-20V
DIGITALI/OPINS	14 (of which 6 provide PWMoutput)
PWM DIGITALI/OPINS	6
ANALOGINPUTPINS	6
DC CURRENT PERI/OPIN	20mA
DC CURRENT FOR3.3VPIN	50mA
FLASHMEMORY	32 KB (ATmega328P) of which 0.5 KB usedby bootloader
SRAM	2 KB(ATmega328P)
EEPROM	1 KB(ATmega328P)
CLOCKSPEED	16 MHz
LED_BUILTIN	13
LENGTH	68.6mm
WIDTH	53.4mm
WEIGHT	25 g

Arduino UNO

Pinout Diagram



Installing the Operating System on Raspberry Pi

Raspberry Pi recommend the use ofRaspberry Pi Imagerto install an operating system on your SD card. You will need another computer with an SD card reader to install the image.

SD Cards for Raspberry Pi

Raspberry Pi computers use a micro SD card, except for very early models which use a full-sized SD card.

Using an SD card of 8GB or greater capacity with Raspberry Pi OS. If you are using the lite version of Raspberry Pi OS, you can use a 4GB card. Other operating systems have different requirements: for example, LibreELEC can run from a smaller card. Please check with the supplier of the operating system to find out what capacity of card they recommend.

Using Raspberry Pi Imager

Raspberry Pi have developed a graphical SD card writing tool that works on Mac OS, Ubuntu 18.04, and Windows calledRaspberry Pi Imager; this is the easiest option for most users since it will download the image automatically and install it to the SD card.

Download the latest version ofRaspberry Pi Imagerand install it. If you want to use Raspberry Pi Imager from a second Raspberry Pi, you can install it from a terminal using`sudo apt install rpi-imager`. Then:

- Connect an SD card reader with the SD cardinside.
- Open Raspberry Pi Imager and choose the required OS from thelistpresented.
- Choose the SD card you wish to write your imageto.
- Review your selections and click on theWritebutton to begin writing datatothe SDCard.

Downloading an Image

If you are using a different tool than Raspberry Pi Imager to write to your SD Card, most require you to download the image first, then use the tool to write it to the card. Official images for recommended operating systems are available to download from the Raspberry Pi websitedownloads page. Alternative operating systems for Raspberry Pi computers arealso availablefrom some third-party vendors.

You may need to unzip the downloaded file (.zip) to get the image file (.img) you need to write to the card.

Configuration on First Boot

If you have not already configured your operating system using theAdvancedMenuof Raspberry Pi Imager when Raspberry Pi OS starts up for the first time you will be guided through initial setup.

The Raspberry Pi OS configuration wizard will run on the first boot. The wizard starts off by allowing you to configure international settings and your timezone information.

After hitting "Next" you'll be prompted to create a user account. Here you can choose your username, and a password.

If you want to you can set your username to the old default username ofpi, which was used on older versions of Raspberry Pi OS.

However, if you do choose to create this account you will trigger a warning message, and we'd advise you to avoid the old default password ofraspberry.

After creating an user account you can configure your screen, and your wireless network.

Once your wireless network is configured and your Raspberry Pi has access to the Internet you will be prompted to update the operating system to the latest version. This will automatically download any patches and updates needed to bring your new operating system right up to date.

Once the operating system is updated you will be prompted to reboot your Raspberry Pi.

Conclusion: The student is able to understand different families kits, boards and shields of Arduino and installation of OS on Raspberry-Pi SBC.

Experiment No:11

Title: Write simple program using Open-source prototype platform like Raspberry-Pi/Beagle board/Arduino for digital read/write using LED and switch Analog read/write using sensor and actuators.

Objective:1) To study Arduino / Raspberry-pi GPIO programming

Outcome: Introductory understanding of GPIO programming using C / Python programming language.

Theory :

Tinkercad -https://www.tinkercad.comis an excellent tool that allows you to simulate Arduino-based systems and a lot more. We can simulate all exercises and even your own designs before trying them on real hardware. It also allows us to do programming using blocks. We can download / copy-paste the generated code later into Arduino IDE to program the real Arduino board, rather than having to write it from scratch.

Hardware -In Components Basic, you can select Arduino Uno R3. We can add more components and wire them up as desired. Clicking on the lead of a component allows you to start a connecting wire from there. Clicking on a wire allows you to change itscolor.

Programming and Simulation

To program the Arduino,

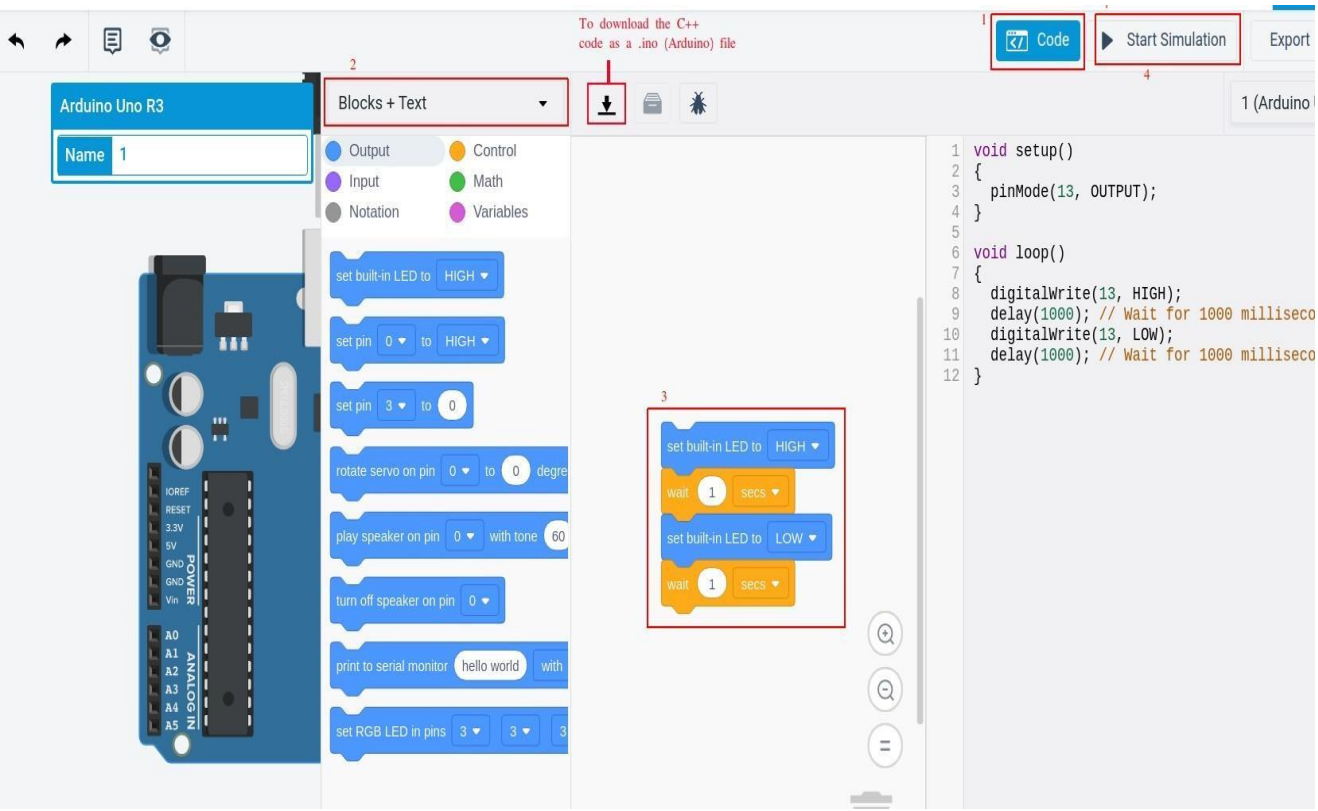
1.Click on Code

1. You can choose Blocks or Blocks+Text or Text. For beginners, it isrecommended to use Blocks +Text.

a. This allows you to see the C++ code generated corresponding to yourblocks.

b. You can copy this code later into Arduino IDE to program the real Arduino,rather than having to write it fromscratch.

c. You can also download the code as an Arduino-compatible .ino file. 3.You can code by selecting the blocks and connecting themappropriately. 4.You can start the simulation by clicking StartSimulation.



Debugging-

More often than not, the code written by a programmer does not work as expected the very first time he/she runs it. We need to find out the logical flaws in our code and fix them before we are able to achieve full functionality. Figuring out flaws usually boil down to inspecting variable values at various points in our code, and comparing it with the expected values at those points based on the program logic and data inputs. The usual way Arduino programmers do it is by printing out the variable values toSerialconsole.

Tinkercard allows for debugging without having to print the values you want to inspect through Serial. The example below shows debugging of theUltrasonicDistance Sensorexample.

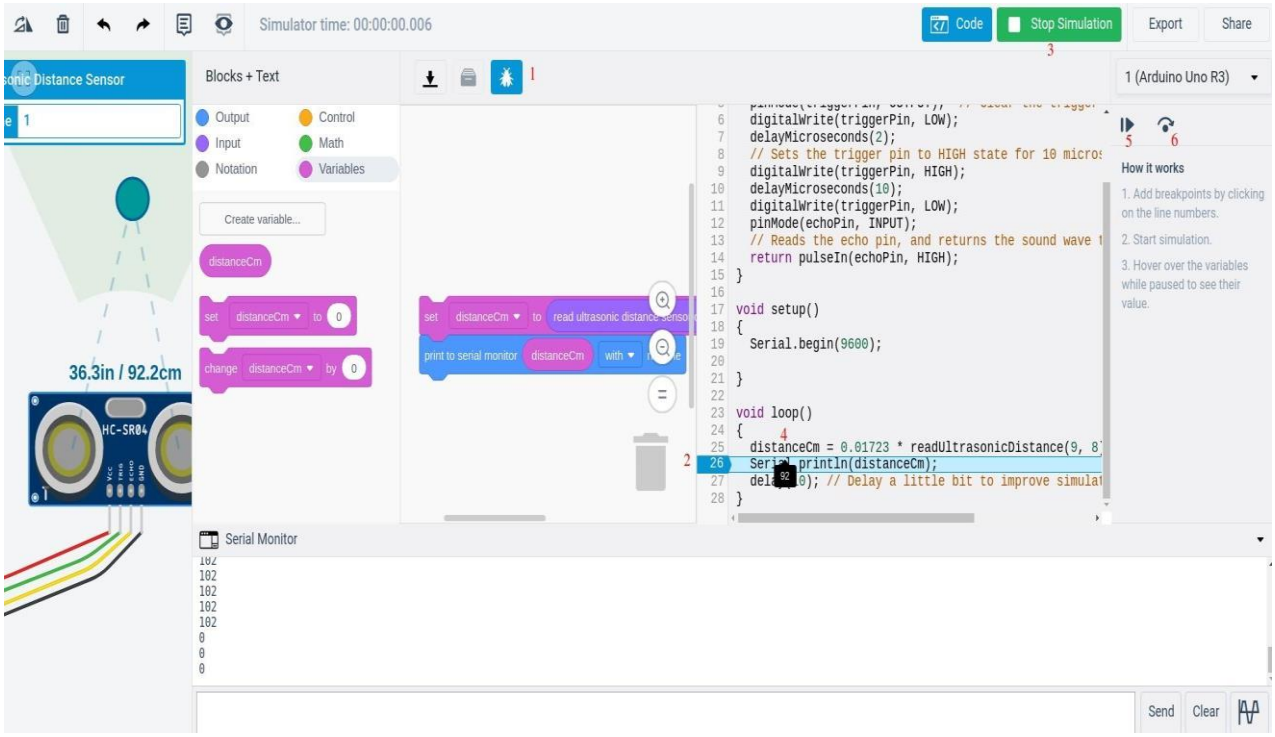
Press the Debugger button.

1. Select the line(s) where you want the execution is to be paused. Such a linewhere you wish to pause execution is called abreakpoint.

2. Click StartSimulation.

3. Hover over the variable values you want to inspect, and determine if the values are along the expected lines. If not, there is something wrong, and use your logicto determine what could bewrong.

4.You can press the Resume execution button to run until the next breakpoint. 5.You can also step line by line by clicking the Step Over Next Functionbutton.



Conclusion : The student is able to understand GPIO programming of Arduino by using sensors and actuators.