MUFFAKHAM JAH
COLLEGE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF
ELECTRONICS AND COMMUNICATION ENGINEERING

EMBEDDED SYSTEMSAND IOT LAB (PC752EC)

Vision and Mission of the Institution

Vision

To be part of universal human quest for development and progress by contributing high calibre, ethical and socially responsible engineers who meet the global challenge of building modern society in harmony with nature.

Mission

- To attain excellence in imparting technical education from the undergraduate through doctorate levels by adopting coherent and judiciously coordinated curricular and co-curricular programs
- To foster partnership with industry and government agencies through collaborative research and consultancy
- To nurture and strengthen auxiliary soft skills for overall development and improved employability in a multi-cultural work space
- To develop scientific temper and spirit of enquiry in order to harness the latent innovative talents
- To develop constructive attitude in students towards the task of nation building and empower them to become future leaders
- To nourish the entrepreneurial instincts of the students and hone their business acumen.
- To involve the students and the faculty in solving local community problems through economical and sustainable solutions.

Vision and Mission of ECE Department

Vision

To be recognized as a premier education center providing state of art education and facilitating research and innovation in the field of Electronics and Communication.

Mission

We are dedicated to providing high quality, holistic education in Electronics and Communication Engineering that prepares the students for successful pursuit of higher education and challenging careers in research, R& D and Academics.

Program Educational Objectives of B. E (ECE) Program:

1. Graduates will demonstrate technical competence in their chosen fields of employment by identifying, formulating, analyzing and providing engineering solutions using current techniques and tools
2. Graduates will communicate effectively as individuals or team members and demonstrate leadership skills to be successful in the local and global cross-cultural working environment
3. Graduates will demonstrate lifelong learning through continuing education and professional development
4. Graduates will be successful in providing viable and sustainable solutions within societal, professional, environmental and ethical contexts

MUFFAKHAM JAH
COLLEGE OF ENGINEERING AND TECHNOLOGY
BANJARA HILLS, ROAD NO-3, TELANGANA-500034

LABORATORY MANUAL

EMBEDDED SYSTEMS AND IOT LAB

Prepared by: Dr.Afshan Kaleem

Approved by: Dr. Ayesha Naaz

MUFFAKHAM JAH COLLEGE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF ELECTRONICS AND COMMUNICATIONS ENGINEERING

(Name of the Subject/Lab Course): EMBEDDED SYSTEMSAND IOT LAB (PC752EC)

Code: PC752EC	**Programme: UG**
Branch: ECE	**Version No: 1**
Year: IV	**Updated on:28-9-24**
Semester: VII	**No. of Pages: 82**

Classification Status (Unrestricted/restricted): Unrestricted
Distribution List: Department, Lab, Library, Lab Incharge

Prepared by: 1) Name:	**1) Name:**
2) Sign :	**2) Sign :**
3) Designation:	**3) Designation:**
4) Date :	**4) Date :**
Verified by: 1) Name:	*** For Q.C Only**
2) Sign :	**1) Name:**
3) Designation:	**2) Sign :**
4) Date:	**3) Designation:**
	4) Date :

Approved by: (HOD) 1) Name:
2) Sign :
3) Date :

PC752EC ***w.e.f Academic year 2023-2024***

EMBEDDED SYSTEMS AND IOT APPLICATIONS LAB

PC 752EC Instruction: 3 periods per week **Duration of SEE:- 3 hours**
CIE: 25 marks SEE:- 50 Marks
Credits: 1

Prerequisites: Microprocessor &amp; Microcontroller Lab(**PC752EC**)

PART-A

Interfacing Programs using embedded C on ARM Microcontroller Kit:

1. Program to interface 8-Bit LED and switch interface.
2. Program to implement Buzzer interface on IDE environment.
3. Program to display message in a 2-line x 16 characters LCD display.
4. Program to interface stepper motor and rotate in clockwise and anticlockwise direction.
5. Program to interface a Temperature sensor LM35, read the values, and display them.
6. Program to demonstrate serial communication, i.e., to transmit from the kit and receive from the PC using a serial port.

PART-B

Interfacing Programs using C/Python Programming on Arduino/Raspberry Pi Kit for IoT Applications:

7. Interface a Push button with Arduino/Raspberry Pi and write a program to turn ON LED when the push button is pressed.
8. Interface Digital sensor (IR/LDR) with Arduino/Raspberry Pi and write a program to turn ON LED upon sensor detection.
9. Interface LCD with Arduino/Raspberry Pi and write a program to display a message on it.
10. Interface DHT11 sensor with Arduino/Raspberry Pi and write a program to print temperature and humidity readings.
11. Interface motor using relay with Arduino/Raspberry Pi and write a program to turn ON the motor when the push button is pressed.
12. Interface Bluetooth unit with Arduino/Raspberry Pi and write a program to turn LED ON/OFF when "1"/"0" is received from a smartphone using Bluetooth.
13. Program to upload temperature and humidity data to ThingSpeak cloud using Arduino/Raspberry Pi.

Note: A minimum of 10 experiments to be performed and at least
3experiments fromeach part to be performed.

GENERAL GUIDELINES AND SAFETY INSTRUCTIONS

1. Sign in the log register as soon as you enter the lab and strictly observe your lab timings.
2. Strictly follow the written and verbal instructions given by the teacher / Lab Instructor. If you do not understand the instructions, the handouts and the procedures, ask the instructor or teacher.
3. **Never work alone!** You should be accompanied by your laboratory partner and / or the instructors / teaching assistants all the time.
4. It is mandatory to come to lab in a formal dress and wear your ID cards.
5. Do not wear loose-fitting clothing or jewelry in the lab. Rings and necklaces are usual excellent conductors of electricity.
6. Mobile phones should be switched off in the lab. Keep bags in the bag rack.
7. Keep the labs clean at all times, no food and drinks allowed inside the lab.
8. Intentional misconduct will lead to expulsion from the lab.
9. Do not handle any equipment without reading the safety instructions. Read the handout and procedures in the Lab Manual before starting the experiments.
10. Do your wiring, setup, and a careful circuit checkout before applying power. Do not make circuit changes or perform any wiring when power is on.
11. Avoid contact with energized electrical circuits.
12. Do not insert connectors forcefully into the sockets.
13. **NEVER** try to experiment with the power from the wall plug.
14. Immediately report dangerous or exceptional conditions to the Lab instructor/ teacher: Equipment that is not working as expected, wires or connectors are broken, the equipment that smells or “smokes”. If you are not sure what the problem is or what's going on, switch off the Emergency shutdown.
15. Never use damaged instruments, wires or connectors. Hand over these parts to the Lab instructor/Teacher.
16. Be sure of location of fire extinguishers and first aid kits in the laboratory.
17. After completion of Experiment, return the bread board, trainer kits, wires, CRO probes and other components to lab staff. Do not take any item from the lab without permission.
18. Observation book and lab record should be carried to each lab. Readings of current lab experiment are to be entered in Observation book and previous lab experiment should be written in Lab record book. Both the books should be corrected by the faculty in each lab.
19. Handling of Semiconductor Components: Sensitive electronic circuits and electronic components have to be handled with great care. The inappropriate handling of electronic component can damage or destroy the devices. The devices can be destroyed by driving to high currents through the device, by overheating the device, by mixing up the polarity, or by electrostatic discharge (ESD). Therefore, always handle the electronic devices as indicated by the handout, the specifications in the data sheet or other documentation.
20. Special Precautions during soldering practice
    a. Hold the soldering iron away from your body. Don't point the iron towards you.
    b. Don't use a spread solder on the board as it may cause short circuit.
    c. Do not overheat the components as excess heat may damage the components/board.
    d. In case of burn or injury seek first aid available in the lab or at the college dispensary.

MUFFAKHAM JAH COLLEGE OF ENGINEERING & TECHNOLOGY

BE 4/4 Year VIII Semester ECE

EMBEDDED SYSTEMS AND IOT APPLICATIONS LAB

List of ExperimentsPART A

[Interfacing Programs using Embedded C on ARM Microcontroller Kit]

PART-A

Interfacing Programs using embedded C on ARM Microcontroller Kit:

1. Program to interface 8-Bit LED and switch interface.
2. Program to implement Buzzer interface on IDE environment.
3. Program to display message in a 2-line x 16 characters LCD display.
4. Program to interface stepper motor and rotate in clockwise and anticlockwise direction.
5. Program to interface a Temperature sensor LM35, read the values, and display them.
6. Program to demonstrate serial communication, i.e., to transmit from the kit and receive from the PC using a serial port.

PART B

Interfacing Programs using C/Python Programming on Arduino/Raspberry Pi Kit for IoT Applications:

7. Interface a Push button with Arduino/Raspberry Pi and write a program to turn ON LED when the push button is pressed.
8. Interface Digital sensor (IR/LDR) with Arduino/Raspberry Pi and write a program to turn ON LED upon sensor detection.
9. Interface LCD with Arduino/Raspberry Pi and write a program to display a message on it.
10. Interface DHT11 sensor with Arduino/Raspberry Pi and write a program to print temperature and humidity readings.
11. Interface motor using relay with Arduino/Raspberry Pi and write a program to turn ON the motor when the push button is pressed.
12. Interface Bluetooth unit with Arduino/Raspberry Pi and write a program to turn LED ON/OFF when "1"/"0" is received from a smartphone using Bluetooth.
13. Program to upload temperature and humidity data to ThingSpeak cloud using Arduino/Raspberry Pi.

PART - A

Interfacing Programs using Embedded C on ARM Microcontroller Kit

ARM Microcontroller

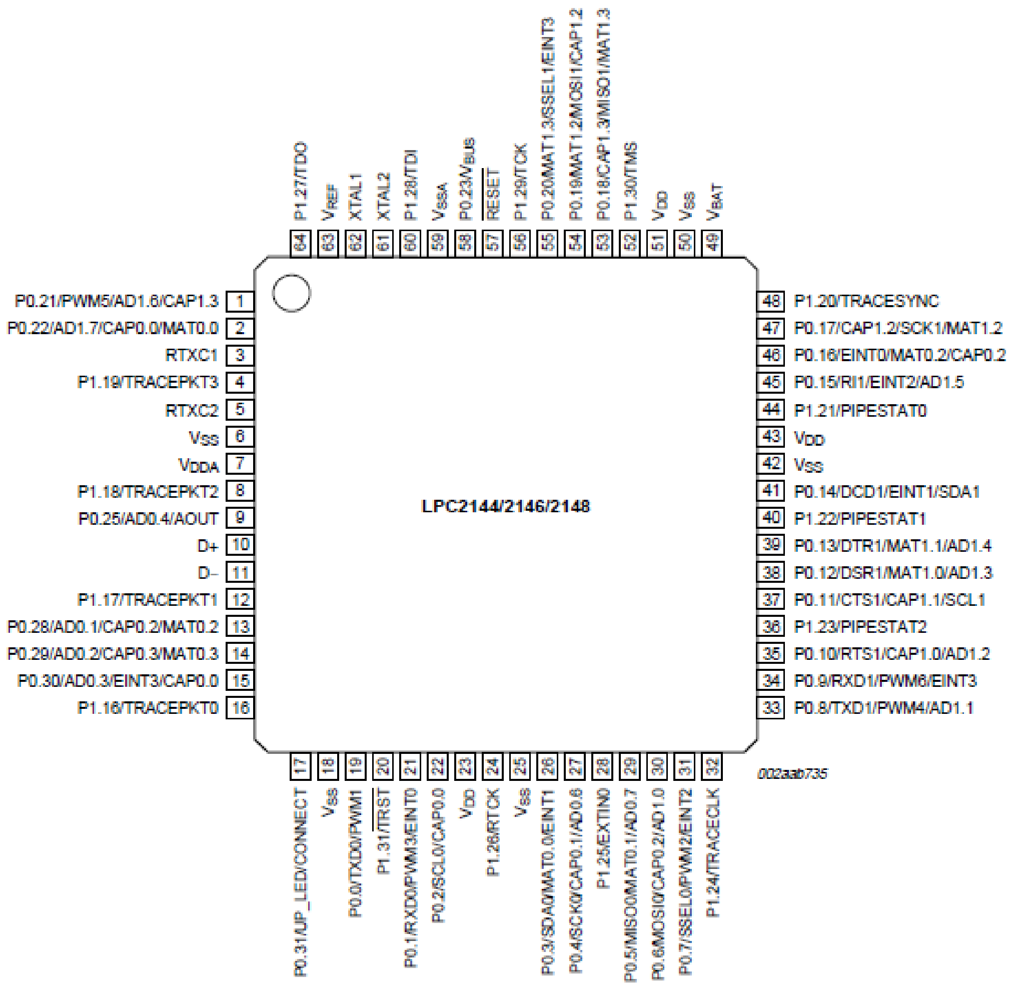
The LPC2141/2/4/6/8 microcontrollers are based on a 32/16-bit ARM7TDMI-S CPU with real-time emulation and embedded trace support, that combines the microcontroller with embedded high speed flash memory ranging from 32 kB to 512 kB. A 128-bit wide memory interface and unique accelerator architecture enable 32-bit code execution at the maximum clock rate. For critical code size applications, the alternative 16-bit Thumb mode reduces code by more than 30 % with minimal performance penalty.

Due to their tiny size and low power consumption, LPC2141/2/4/6/8 are ideal for applications where miniaturization is a key requirement, such as access control and point-of-sale. A blend of serial communications interfaces ranging from a USB 2.0 Full Speed device, multiple UARTs, SPI, SSP to I2Cs, and on-chip SRAM of 8 kB up to 40 kB, make these devices very well suited for communication gateways and protocol converters, soft modems, voice recognition and low end imaging, providing both large buffer size and high processing power. Various 32-bit timers, single or dual 10-bit ADC(s), 10-bit DAC, PWM channels and 45 fast GPIO lines with up to nine edge or level sensitive external interrupt pins make these microcontrollers particularly suitable for industrial control and medical systems.

LPC2148 Chip Features:

1. 16-bit/32-bit ARM7TDMI-S microcontroller in a tiny LQFP64 package.
2. 8 kB to 40 kB of on-chip static RAM and 32 kB to 512 kB of on-chip flash memory.
3. 128-bit wide interface/accelerator enables high-speed 60 MHz operation.
4. In-System Programming/In-Application Programming (ISP/IAP) via on-chip boot loader software. Single flash sector or full chip erase in 400 ms and programming of 256 bytes in 1 ms.
5. Embedded ICE RT and Embedded Trace interfaces offer real-time debugging with the on-chip Real Monitor software and high-speed tracing of instruction execution.
6. USB 2.0 Full-speed compliant device controller with 2 kB of endpoint RAM.
7. Single 10-bit DAC provides variable analog output (LPC2142/44/46/48 only).
8. Two 32-bit timers/external event counters (with four capture and four compare channels each), PWM unit (six outputs) and watchdog.
9. Low power Real-Time Clock (RTC) with independent power and 32 kHz clock input.

10. Multiple serial interfaces including two UARTs (16C550), two Fast I2C-bus (400 kbit/s).
11. SPI and SSP with buffering and variable data length capabilities.
12. Vectored Interrupt Controller (VIC) with configurable priorities and vector addresses.
13. Up to 45 of 5 V tolerant fast general purpose I/O pins in a tiny LQFP64 package.
14. Up to 21 external interrupt pins available.
15. 60 MHz maximum CPU clock available from programmable on-chip PLL with settling time of 100 ms.
16. On-chip integrated oscillator operates with an external crystal from 1 MHz to 25 MHz.
17. Power saving modes include Idle and Power-down.
18. Individual enable/disable of peripheral functions as well as peripheral clock scaling for additional power optimization.
19. Processor wake-up from Power-down mode via external interrupt or BOD.
20. Single power supply chip with POR and BOD circuits:
21. CPU operating voltage range of 3.0 V to 3.6 V (3.3 V ± 10 %) with 5 V tolerant I/O pads.

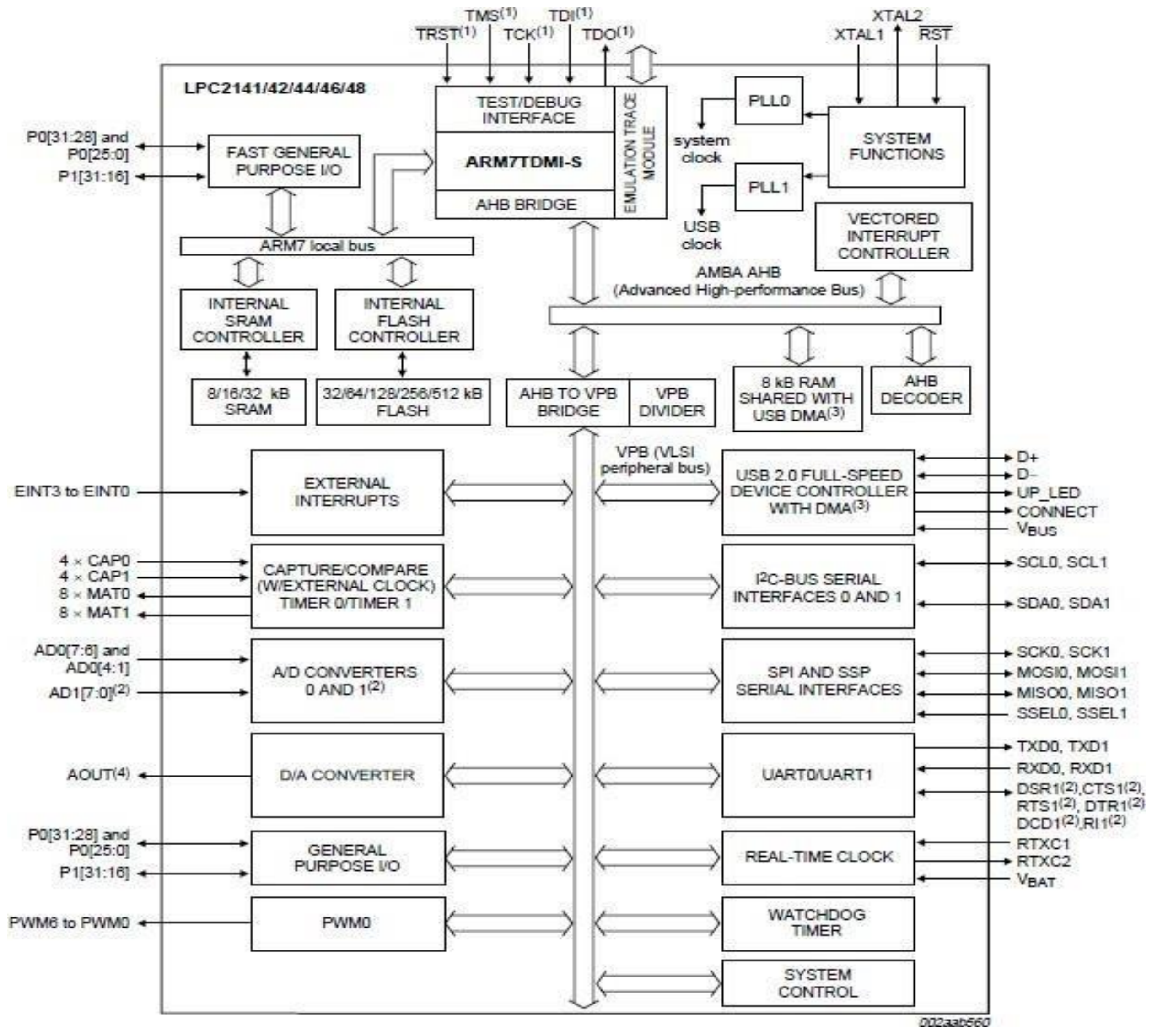


Pin Diagram of ARM LPC2148

Architectural overview

The LPC2141/2/4/6/8 consists of an ARM7TDMI-S CPU with emulation support, the ARM7 Local Bus for interface to on-chip memory controllers, the AMBA Advanced High-performance Bus (AHB) for interface to the interrupt controller, and the ARM Peripheral Bus (APB, a compatible superset of ARM's AMBA Advanced Peripheral Bus) for connection to on-chip peripheral functions. The LPC2141/24/6/8 configures the ARM7TDMI-S processor in little-endian byte order.

The connection of on-chip peripherals to device pins is controlled by a Pin Connect Block. This must be configured by software to fit specific application requirements for the use of peripheral functions and pins.



Architecture of ARM LPC2148

LPC 2144/6//8 consists of 45 GPIO functionality is in 2 ports which are:

1. Port0 (P0.0 to P0.31)- 24,26,27 are invisible pins, remaining 29 are visible i/o pins.
2. Port1 (P1.16 to P0.31)- 16 pins are visible and 16 pins are invisible(P1.0-P1.15)

It consists of 19 different peripherals such as:

	FUNCTION	PIN	TYPE & DESCRIPTION
1.	D+	10	INPUT/OUTPUT(USB bidirectional D+ line)
2.	D-	11	INPUT/OUTPUT(USB bidirectional D- line)
3.	XTAL1	62	
4.	XTAL2	61	
5.	RTXC1	3	INPUT(Input to the RTC oscillator circuit)
6.	RTXC2	5	OUTPUT(output to the RTC oscillator circuit)
7.	VSS	6, 18 **,**25,42,50	
8.	VSSA	52	INPUT(Analog Ground: 0 V reference)
9.	VDD	23, 43, 51	(power supply)
10.	VDDA	7	INPUT(analog power supply)
11.	VREF	63	INPUT(**A/D Converter Reference**)
12.	VBAT	49	INPUT(RTC power supply)

Register description

The Pin Control Module contains 3 registers. Pin selection register are used to select the different functionalities of LPC2148 i/o pins.

- PINSEL0 Pin function select
  - Read/Write 0x0000 0000 (P0.0-P0.15)
- PINSEL1 Pin function select
  - Read/Write 0x0000 0000 (P0.16-P0.31)
- PINSEL2 Pin function select
  - Read/Write 0x0000 0000 (P1.16-P1.31)

GPIO port Direction register (IODIR)

- IODIR Register is used to configure the i/o pins, either input and output pins
- IODIR is a 32-pin register.
- IODIRx=0x00000000-i/p config.

- IODIRx=0xffffffff-o/p config.

GPIO port pin value register (IOPIN)

- This register provides the value of port pins that are configured to perform only digital functions.
- IOPIN register is used to read the current state of every GPIO pin

GPIO port Set register (IOSET)

- This register is used to produce a HIGH level output at the port pins configured as GPIO in an OUTPUT mode.
- Writing 1 produces a HIGH level at the corresponding port pins.
- Writing 0 has no effect.

GPIO port Clear register (IOCLR)

- This register is used to produce a LOW level output at port pins configured as GPIO in an OUTPUT mode.
- Writing 1 produces a LOW level at the corresponding port pin and clears the corresponding bit in the IOSET register.
- Writing 0 has no effect.

EXPERIMENT – 1

1) Write an Embedded C program to blink single LED on ARM LPC2148microcontroller kit.

PROGRAM:

```
#include<LPC214X.h>
void delay(unsigned int);
int main()
{
IODIR0=0X0010000;
while(1)
{
IOSET0=0X0010000;
delay(20);
IOCLR0=0X0010000;
delay(20);
}
}

void delay(unsigned int i)
{
int j,k;
for(j=0;j<i;j++)
for(k=0;k<1275;k++);
}
```

2) **Write an Embedded C program to toggle 8 LEDs on ARM LPC2148 Microcontroller Kit**.

<u>PROGRAM:</u>

```
#include<LPC214X.h>
void wait (void)
{                                              /* wait function */
  int  d;
  for (d = 0; d < 1000000; d++);            /* only to delay for LED flashes */
}

int main()
{
IODIR1 = 0x00FF0000;
while(1)
{
IOSET1 = 0x00FF0000;
wait ();
IOCLR1 = 0x00FF0000;
wait ();
}
}
```

3) **Write an Embedded C program to toggle alternate LEDs on ARM LPC2148Microcontroller Kit**.

PROGRAM:

```
#include<LPC214X.h>
void delay(unsigned int);
int main()
{
IODIR0=0X00FF0000;
while(1)
{
IOSET0=0X00AA0000;
IOCLR0=0X00550000;
delay(20);
IOSET0=0X00550000;
IOCLR0=0X00AA0000;
delay(20);
}
}

void delay(unsigned int i)
{
int j,k;
for(j=0;j<i;j++)
for(k=0;k<1275;k++);
}
```

```
#include <LPC21xx.H>                          /* LPC21xx definitions */
void wait (void)
{                                             /* wait function */
  int  d;
  for (d = 0; d < 1000000; d++);              /* only to delay for LED flashes */
}

int main (void)
{
  unsigned int i;                             /* LED var */
  IODIR1 = 0x00FF0000;                        /* P1.16..23 defined as Outputs */
  while (1)
{                                             /* Loop forever */
    for (i = 1<<16; i < 1<<23; i <<= 1)
{                                             /* Blink LED 0,1,2,3,4,5,6 */
      IOSET1 = i;                             /* Turn on LED */
      wait ();                                /* call wait function */
      IOCLR1 = i;                             /* Turn off LED */
    }
    for (i = 1<<23; i > 1<<16; i >>=1 )
{                                             /* Blink LED 7,6,5,4,3,2,1 */
      IOSET1 = i;                             /* Turn on LED */
      wait ();                                /* call wait function */
      IOCLR1 = i;                             /* Turn off LED */
    }
  }
}
```

EXPERIMENT – 2

1) Write an Embedded C program to rotate stepper motor in Clockwise direction continuously by interfacing it to ARM LPC2148 Microcontroller kit.

PROGRAM:

```
#include <LPC214X.H>
void delay(unsigned int count)
{
  unsigned int i ,j= 0;
  for(i = 0; i <= count; i ++)
  for(j = 0; j <= 5; j ++);
}
unsigned long i;
int main()
{
  IODIR1 = 0x00FF0000;
  i = 0x00110000;
  while(1)
  {
          IOSET1 = 0x00110000;
          delay(10000);
          IOCLR1 = 0x00FF0000;
          IOSET1 = 0x00220000;
          delay(10000);
          IOCLR1 = 0x00FF0000;
          IOSET1 = 0x00440000;
          delay(10000);
          IOCLR1 = 0x00FF0000;
          IOSET1 = 0x00880000;
          delay(10000);
          IOCLR1 = 0x00FF0000;
  }
}
```

2) Write an Embedded C program to rotate stepper motor in Anti-Clockwise direction continuously by interfacing it to ARM LPC2148 Microcontroller kit.

PROGRAM:

```
#include <LPC214X.H>
void delay(unsigned int count)
{
  unsigned int i ,j= 0;
  for(i = 0; i <= count; i ++)
  for(j = 0; j <= 5; j ++);
}
unsigned long i;
int main()
{
  IODIR1 = 0x00FF0000;
  i = 0x00110000;
  while(1)
  {
          IOSET1 = 0x00880000;
          delay(10000);
          IOCLR1 = 0x00FF0000;
          IOSET1 = 0x00440000;
          delay(10000);
          IOCLR1 = 0x00FF0000;
          IOSET1 = 0x00220000;
          delay(10000);
          IOCLR1 = 0x00FF0000;
          IOSET1 = 0x00110000;
          delay(10000);
          IOCLR1 = 0x00FF0000;
  }
}
```

3) Write an Embedded C program to rotate stepper motor 2 times in Clockwise direction & 2 times in Anti-clockwise direction continuously by interfacing it to ARM LPC2148 Microcontroller kit.

PROGRAM:

```
#include <LPC214X.H>
void int (void)
int d;
  for(d = 0; d<1000000; d ++)
}

int main(void)
unsigned int i,j,k;
  IODIR1 = 0x00FF0000;
  while(1)
  {
For (j=0;j<100;j++)
}
For (i=16;i<=19;i++)
{
            IOSET1 =1<<i;
            wait ( )
            IOCLR1 =1<< i;
}
}

Wait ( )
  For(k=0;k<100;k++)
{
For(i=19;i>=16;i--)
{
 IOSET1 =1<<
Wait ( )
IOCLR1 = 1<< i;
  }
  }
  }
```

EXPERIMENT – 3

1) Write an Embedded C program to display a message on LCD screen by interfacing it to ARM LPC2148 Microcontroller kit.

PROGRAM:

```
/* Interface program for LCD Interface for MCB2140 board */
#include <LPC21xx.H>
unsigned char disp1[] = {'M','J','C','E','T',' ','E','C ','E',' ','D','E ','P','T','*',0x00};
void delay(unsigned int x)
{
  unsigned int i,j;
  for(i = 0; i < 10; i ++)
     for(j = 0; j < x; j ++);
}
void delay1()
{
  unsigned int i,j;
  for(i = 0; i < 0x50; i ++)
    for(j = 0; j < 0x4fb; j ++);
}
/* Function to display the character entered from the keyboard */
void Disp_Key(unsigned long c)
{
  unsigned int x;
  unsigned long l;
  l = c << 16;
  IOCLR1 = 0x00FF0000;
  IOSET1 = l;
  IOCLR0 = 0x003F8000;                              /* E    R/W* RS */
  IOSET0 = 0xfa << 15;                              /* 0     1      0  */
  IOCLR0 = 0x003F8000;
  IOSET0 = 0xf9 << 15;                              /* 0     0      1  */
  IOCLR0 = 0x003F8000;
  IOSET0 = 0xfd << 15;                              /* 1     0      1  */
  IOCLR0 = 0x003F8000;
  IOSET0 = 0xf9 << 15;                                    /* 0     0     1  */
  IOCLR0 = 0x003F8000;
  x = 0x50;
  delay(x);
}
/* Function to issue the series of commands to the LCD module */
void Command_Write(unsigned long com_word)
{
  unsigned int x;
  IOCLR1 = 0x00FF0000;
  IOSET1 = com_word;          /* Send commands to the LCD through GPIO1 */
  IOCLR0 = 0x003F8000;                                    /* E    R/W* RS */
  IOSET0 = 0xfb << 15;                                    /* 0     1    1  */
  IOCLR0 = 0x003F8000;
  IOSET0 = 0xf8 << 15;                                    /* 0     0    0  */
  IOCLR0 = 0x003F8000;
  IOSET0 = 0xfc << 15;                                    /* 1     0    0  */
  IOCLR0 = 0x003F8000;
  IOSET0 = 0xf8 << 15;                                    /* 0     0    0  */
  IOCLR0 = 0x003F8000;
  x = 0x50;
  delay(x);
}
/* Function to read characters from a string and displaying the same by calling the
Disp_key fucntion */
void Disp_String1(void)
{
  int i;
  i = 0;
  while(disp1[i] != 0x00)
  {
    Disp_Key(disp1[i]);
    i ++;
  }    }
/* Function to initialize the LCD module */
void LCD_init(void)
{
  int j;
  unsigned long command;
   j = 0x9f;                              /* Delay given for the voltage rise from 0V to 5V */
  delay(j);
  command = 0x38 << 16;                   /* Issue software reset for the LCD module */
  Command_Write(command);
  j = 0x13F;
  delay(j);
   command = 0x38 << 16;                          /* Function set 8-bit, 1 line 5*7 font */
  Command_Write(command);
  command = 0x0c << 16;                   /* Display on off control. Diplay on, cursor off, */
  Command_Write(command);                         /* blink off */
  command = 0x01 << 16;                           /* Clear the LCD display */
  Command_Write(command);
}
  int main()
{
  PINSEL0 = 0x00050000;                          /* Enable RxD1 and TxD1 */
  PINSEL1 = 0x00000000;
  PINSEL2 = 0x00000004;
  IODIR0 = 0x007FB0FC;                    /* Configure the GPIO0 and GPIO1 as O/P ports */
  IODIR1 = 0x00FF0000;                       /* Initialize LCD module */
  LCD_init();
  delay1();
Disp_String1();                /* Displays the "MJCET ECE DEPT*" message
```

PART -B

Experiment 1: Blinking LED

Tools Required:

- Arduino Uno
- LED
- Resistor (220 ohms)
- Jumper wires
- Breadboard

Code:

```
// Pin number where the LED is connected

#define ledPin 13

void setup() {

  // Set the LED pin as an output

  pinMode(ledPin, OUTPUT);

}

void loop() {

  // Turn the LED on

  digitalWrite(ledPin, HIGH);

  delay(1000); // Wait for 1 second

  // Turn the LED off

  digitalWrite(ledPin, LOW);

  delay(1000); // Wait for 1 second

}
```

Experiment 2: LDR With and Without Serial Monitor

Tools Required:

- Arduino Uno
- LDR (Light Dependent Resistor)
- Resistor (10k ohms)
- LED
- Jumper wires
- Breadboard

Without Serial Monitor Code:

```
const int ldrPin = A0; // Connect the LDR to analog pin A0

const int ledPin = 13; // Connect the LED to digital pin 13

void setup() {

  pinMode(ldrPin, INPUT);

  pinMode(ledPin, OUTPUT);

}

void loop() {

  int ldrValue = analogRead(ldrPin); // Read LDR value (0-1023)

  // If the LDR value is below 100, turn the LED on; otherwise, turn it off

  if (ldrValue < 100) {

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

  } else {

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

  }

}
```

With Serial Monitor Code:

```
const int ldrPin = A0; // Connect the LDR to analog pin A0

const int ledPin = 13; // Connect the LED to digital pin 13

void setup() {

  pinMode(ldrPin, INPUT);

  pinMode(ledPin, OUTPUT);

  Serial.begin(9600); // Initialize serial communication at 9600 bps

}

void loop() {

  int ldrValue = analogRead(ldrPin); // Read LDR value (0-1023)

  // If the LDR value is below 100, turn the LED on; otherwise, turn it off

  if (ldrValue < 100) {

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

    Serial.println("ITS DAY TIME LED ON");

    Serial.println(ldrValue);

  } else {

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

    Serial.println("ITS NIGHT TIME LED OFF");

    Serial.println(ldrValue);

  }

}
```

Experiment 3: LCD Screen

Tools Required:

- Arduino Uno
- 16x2 I2C LCD Display
- Jumper wires
- Breadboard

Code:

```
#include <Wire.h>

#include <LiquidCrystal_I2C.h>

// Initialize the library with the numbers of the interface pins

LiquidCrystal_I2C lcd(0x27, 16, 2);

void setup() {

  lcd.init();

  lcd.backlight();

  lcd.setCursor(3, 0);

  lcd.print("Hello WORLD");

  lcd.clear();

}

void loop() {

  // Empty loop

}
```

LCD for Continuous Display Code:

```
#include <Wire.h>

#include <LiquidCrystal_I2C.h>

// Initialize the library with the numbers of the interface pins

LiquidCrystal_I2C lcd(0x27, 16, 2);

void setup() {

  lcd.init();

  lcd.backlight();

}

void loop() {

  lcd.clear();

  lcd.print("Hello World!");

  delay(1000);

  lcd.clear()

  lcd.print("I'm powered by");

  lcd.setCursor(0, 1);

  lcd.print("Arduino!");

  delay(1000);

  lcd.clear();

  lcd.print("Goodbye!");

  delay(1000);

}
```

Experiment 4: DC Motor Using Relay Module

Tools Required:

- Arduino Uno
- DC motor
- Relay module
- Push button
- Jumper wires
- Breadboard

Code:

```
const int buttonPin = 3;

const int relayPin = 2;

int buttonState = 0;

void setup() {

  pinMode(buttonPin, INPUT_PULLUP);

  pinMode(relayPin, OUTPUT);

}

void loop() {

  buttonState = digitalRead(buttonPin);

  if (buttonState == HIGH) {

    digitalWrite(relayPin, HIGH);

  } else

{

    digitalWrite(relayPin, LOW);

  }

}
```

Experiment 5: Bluetooth - Controlling Built-in LED

Tools Required:

- Arduino Uno
- HC-05 Bluetooth module
- Jumper wires
- Breadboard

Code:

```
char data; // Variable for storing received data

void setup() {

  pinMode(LED_BUILTIN, OUTPUT); // Set the LED as an output

  Serial.begin(9600); // Initialize serial communication at 9600 bps

}

void loop() {

  if (Serial.available() > 0) { // Checks if data is coming from the serial port

    data = Serial.read(); // Read the data from the serial port

    if (data == '1') {

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

    } else if (data == '0') {

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

    }

  }

}
```

Bluetooth for Sending Serial Message Code:

```
#include <SoftwareSerial.h>
SoftwareSerial BTSerial(2, 3); // RX, TX
void setup() {
  Serial.begin(9600);
  BTSerial.begin(9600); // Default baud rate for HC-05 module
}
void loop() {
  if (BTSerial.available()) {
    char data = BTSerial.read();
    Serial.print("Received: ");
    Serial.println(data);
  }
}
```

Experiment 6: DHT-11 Sensor

Tools Required:

- Arduino Uno
- DHT-11 Temperature and Humidity Sensor
- Jumper wires
- Breadboard

Code:

```
#include <DHT.h>
#define DHTPIN 2     // Pin where the DHT is connected
#define DHTTYPE DHT11   // DHT 11
DHT dht(DHTPIN, DHTTYPE);
void setup() {
  Serial.begin(9600);
  dht.begin();
}
void loop() {
  float h = dht.readHumidity();
  float t = dht.readTemperature();

  // Check if any reads failed
  if (isnan(h) || isnan(t)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }
  Serial.print("Humidity: ");
  Serial.print(h);
  Serial.print(" %\t");
  Serial.print("Temperature: ");
  Serial.print(t);
  Serial.println(" *C");
  delay(2000);
}
```

Experiment 7: DHT-11 Sensor with LCD

Tools Required:

- Arduino Uno
- DHT-11 Sensor
- 16x2 I2C LCD
- Jumper wires
- Breadboard

Code:

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <DHT.h>
#define DHTPIN 2     // Pin where the DHT is connected
#define DHTTYPE DHT11   // DHT 11
DHT dht(DHTPIN, DHTTYPE);
LiquidCrystal_I2C lcd(0x27, 16, 2);
void setup() {
  lcd.init();
  lcd.backlight();
  dht.begin();
}
void loop() {
  float h = dht.readHumidity();
  float t = dht.readTemperature();
  lcd.clear();
  if (isnan(h) || isnan(t)) {
    lcd.setCursor(0, 0);
    lcd.print("Failed to read");
    return;
  }
  lcd.setCursor(0, 0);
  lcd.print("Temp: ");
  lcd.print(t);
  lcd.print(char(223));
  lcd.print("C");
  lcd.setCursor(0, 1);
  lcd.print("Humidity: ");
  lcd.print(h);
  lcd.print("%");
  delay(2000);
}
```

Experiment 8: Gas Sensor with Serial Monitor

Tools Required:

- Arduino Uno
- Gas sensor (MQ-2)
- Jumper wires
- Breadboard

Code:

```
int sensorPin = A0;
int sensorThreshold = 500;
void setup() {
  Serial.begin(9600);
  pinMode(sensorPin, INPUT);
}
void loop() {
  int sensorValue = analogRead(sensorPin);
  if (sensorValue > sensorThreshold) {
    Serial.print("Gas is too high: ");
    Serial.println(sensorValue);
  } else {
    Serial.print("Gas concentration: ");
    Serial.println(sensorValue);
  }
}
```

Experiment 9: Gas Sensor with LED and Buzzer

Tools Required:

- Arduino Uno
- Gas sensor (MQ-2)
- LED
- Buzzer
- Jumper wires
- Breadboard

Code:

```
int sensorPin = A0;
int sensorThreshold = 500;
const int ledPin = 13;
const int buzzer = 2;
void setup() {
  Serial.begin(9600);
  pinMode(sensorPin, INPUT);
  pinMode(ledPin, OUTPUT);
  pinMode(buzzer, OUTPUT);
}
void loop() {
  int sensorValue = analogRead(sensorPin)
if (sensorValue > sensorThreshold) {
    Serial.print("Gas is too high: ");
    Serial.println(sensorValue);
  } else {
    Serial.print("Gas concentration: ");
    Serial.println(sensorValue);
  }
}
```

Experiment 9: Gas Sensor with LED and Buzzer

Tools Required:

- Arduino Uno
- Gas sensor (MQ-2)
- LED
- Buzzer
- Jumper wires
- Breadboard

Code:

```
int sensorPin = A0;
int sensorThreshold = 500;
const int ledPin = 13;
const int buzzer = 2;
void setup() {
  Serial.begin(9600);
  pinMode(sensorPin, INPUT);
  pinMode(ledPin, OUTPUT);
  pinMode(buzzer, OUTPUT);
}
void loop() {
  int sensorValue = analogRead(sensorPin);
  if (sensorValue > sensorThreshold)
co{
    digitalWrite(ledPin, HIGH);
    digitalWrite(buzzer, HIGH);
    Serial.print("Warning! Gas concentration: ");
    Serial.println(sensorValue);
  } else {
    digitalWrite(ledPin, LOW);
    digitalWrite(buzzer, LOW);
    Serial.print("Gas concentration: ");
    Serial.println(sensorValue);
  }
}
```

CONTENT BEYOND SYLLABUS

Experiment 10: Traffic Light

Tools Required:

- Arduino Uno
- 3 LEDs (Red, Yellow, Green)
- Resistors (220 ohms)
- Jumper wires
- Breadboard

Code:

```
// Define pin numbers for LEDs
const int redLED = 8;
const int yellowLED = 10;
const int greenLED = 3;
void setup() {
  // Set the LED pins as output
  pinMode(redLED, OUTPUT);
  pinMode(yellowLED, OUTPUT);
  pinMode(greenLED, OUTPUT);
}

void loop() {
  digitalWrite(redLED, HIGH);
  delay(5000); // Red light for 5 seconds
  digitalWrite(redLED, LOW);

  digitalWrite(yellowLED, HIGH);
  delay(2000); // Yellow light for 2 seconds
  digitalWrite(yellowLED, LOW);

  digitalWrite(greenLED, HIGH);
  delay(5000); // Green light for 5 seconds
  digitalWrite(greenLED, LOW);
}
```

Experiment 11: Ultrasonic Sensor

Tools Required:

- Arduino Uno
- Ultrasonic sensor (HC-SR04)
- Jumper wires
- Breadboard

Code:

```
const int trig = 9;
const int echo = 10;
void setup() {
  Serial.begin(9600);  // Initialize serial communication
  pinMode(trig, OUTPUT);
  pinMode(echo, INPUT);
}
void loop() {
  digitalWrite(trig, LOW);
  delayMicroseconds(2);
  // Send a 10us pulse to trigger pin
  digitalWrite(trig, HIGH);
  delayMicroseconds(10);
  digitalWrite(trig, LOW);
  // Read the echo pin and calculate distance
  long duration = pulseIn(echo, HIGH);
  long distance = duration * 0.034 / 2;
  Serial.print("DISTANCE: ");
  Serial.print(distance);
  Serial.println(" cm");

  delay(1000); // Wait for 1 second before the next measurement
}
```

Experiment 12: Soil Moisture Sensor

Tools Required:

- Arduino Uno
- Soil moisture sensor
- Jumper wires
- LED
- Breadboard

Code:

```
int sensorPin = A0;     // Soil moisture sensor connected to analog pin A0
int sensorValue = 0;
int ledPin = 13;        // LED connected to digital pin 13
void setup() {
  pinMode(ledPin, OUTPUT);
  Serial.begin(9600);   // Initialize serial communication
}
void loop() {
  // Read the analog value from the soil moisture sensor
  sensorValue = analogRead(sensorPin);
  Serial.println(sensorValue);
  // If sensor value is above threshold (dry soil), turn on LED
  if (sensorValue > 400) {
    digitalWrite(ledPin, HIGH);
  } else {
    // If sensor value is below threshold (moist soil), turn off LED
    digitalWrite(ledPin, LOW);
  }
  delay(1000); // Wait for 1 second before the next reading
}
```

APPENDIX

LABORATORY COURSE ASSESSMENT GUIDELINES

i. The number of experiments/programs/sessions in each laboratory course shall be as per the curriculum in the scheme of instructions provided by OU.

ii. The students will maintain a separate note book for each laboratory course in whichall the related work would be done.

iii. In each session the students will complete the assigned tasks of process development, coding, compiling, debugging, linking and executing the programs.

iv. The students will then execute the programme and validate it by obtaining the correct output for the provided input. The course coordinator will certify the validation in the same session.

v. The students will submit the record in the next class. The evaluation will be continuous and not cycle-wise or at semester end.

vi. The internal marks of 25 are awarded in the following manner:

a.	Laboratory record	-	Maximum Marks 15
b.	Test and Viva Voce	-	Maximum Marks 10

vii. Laboratory Record: Each experimental record is evaluated for a score of 50. **Therubric parameters are as follows:**

a.	Write up format	-	Maximum Score 20
b.	Process development and coding	-	Maximum Score 10
c.	Compile, debug, link and execute program	-	Maximum Score 15
d.	Process validation through input-output	-	Maximum Score 5

While (a) is assessed at the time of record submission, (b), (c) and (d) are assessedduring the session based on the performance of the student in the laboratory session.Hence if a student is absent for any laboratory session but completes the program in another session and subsequently submits the record, it shall be evaluated for a score of 20 and not 50.

viii. The experiment evaluation rubric is therefore as follows :

ix.

Parameter	Max Score	Outstanding	Accomplished	Developing	Beginner	Points
Process Development and Coding	10					
Compilation, Debugging, Linking and Executing	15					
Process Validation	5					
Write up format	20					

x. The first page of the record will contain the following title sheet:

MUFFAKHAM JAH COLLEGE OF ENGINEERING AND TECHNOLOGHY
LABORATORY EXPERIMENT ASSESSMENT SHEET
ELECTRONICS AND COMMUNICATION ENGINEERING DEPARTMENT
B.E. 4/4 2023-2024
ELECTRONIC DESIGN & AUTOMATION LABORATORY

NAME: **ROLL NO.**

Exp. No.	Title of the Program	Date conducted	Date Submitted	Process Development and Coding (Max 10)	Compilation , Debugging, Linking and Executing (Max 15)	Process Validation (Max 5)	Write up format (Max 20)	Total Score (Max 50)
1								
2								
3								
4								
5								
6								
7								
8								
9								
10								
11								
12								
	TOTAL							

Date: **Signature of Course Coordinator**

xi. The 15 marks of laboratory record will be scaled down from the TOTAL of the assessment sheet.

xii. The test and viva voce will be scored for 10 marks as follows:

Internal Test	-	6 marks
Viva Voce / Quiz	-	4 marks

xiii. Each laboratory course shall have 5 course outcomes.

The proposed course outcomes would be as follows:

On successful completion of the course, the student will acquire the ability to:

1. Apply the design concepts for development of a process and interpret data
2. Demonstrate knowledge of programming environment, compiling, debugging, linking and executing variety of programs.
3. Demonstrate documentation and presentation of the algorithms / flowcharts / programs in a record form.
4. Validate the process using known input-output parameters.
5. Employ analytical and logical skills to solve real world problem and demonstrate oral communication skills.

xiv. The Course coordinators would prepare the assessment matrix in accordance with the guidelines provided above for the five course outcomes. The scores to be entered against each of the course outcome would be the sum of the following as obtained from the assessment sheet in the record:

a. Course Outcome 1: Sum of the scores under ‘Process Development and Coding’.
b. Course Outcome 2: Sum of the scores under ‘Compilation/Debugging/Linking and Executing’.
c. Course Outcome 3: Sum of the scores under ‘Write up format’.
d. Course Outcome 4: Sum of the scores under ‘Process validation’.
e. Course Outcome 5: Marks for ‘Internal Test and Viva voce’.

xv. Soft copy of the assessment matrix would be provided to the course coordinators.

xvi. There may be some laboratory courses based on proprietary software like MATLAB, AUTOCAD etc. for which the course coordinators and programme coordinators would formulate appropriate course outcomes.

MUFFAKHAM JAH COLLEGE OF ENGINEERING AND TECHNOLOGY

Program Outcomes of B.E (ECE) Program:

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, 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.
PO 12: 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 (PSOs) of ECE Department, MJCET

PSO1: The ECE Graduates will acquire state of art analysis and design skills in the areas of digital and analog VLSI Design using modern CAD tools.
PSO2: The ECE Graduates will develop preliminary skills and capabilities necessary for embedded system design and demonstrate understanding of its societal impact.
PSO3: The ECE Graduates will obtain the knowledge of the working principles of modern communication systems and be able to develop simulation models of components of a communication system.
PSO4: The ECE Graduates will develop soft skills, aptitude and programming skills to be employable in IT sector.