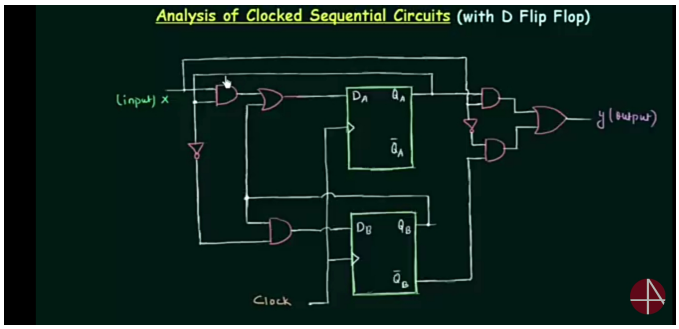
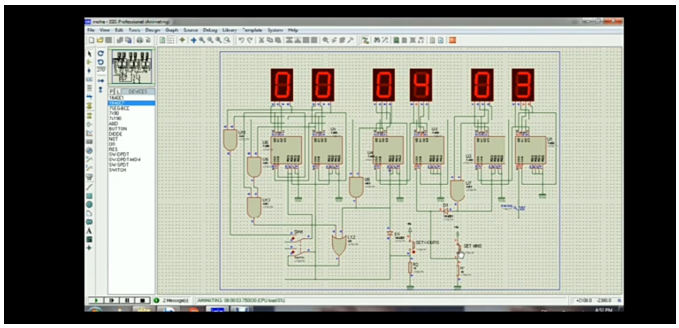
# DAY 10 REPORT

| | | | |
|---|---|---|---|
| Date: | 29/05/20 | Name: | ANKITHA C C |
| Course: | Logic design | USN: | 4AL16EC004 |
| Topic: | 1. Analysis of clocked sequential circuits<br>2. Digital clock design | Semester & Section: | 8th & 'A' SECTION |
| Github Repository: | ankitha-course | | |

FORENOON SESSION DETAILS

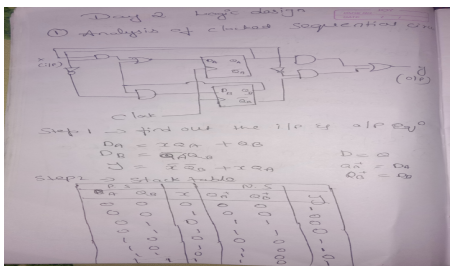
Image of session

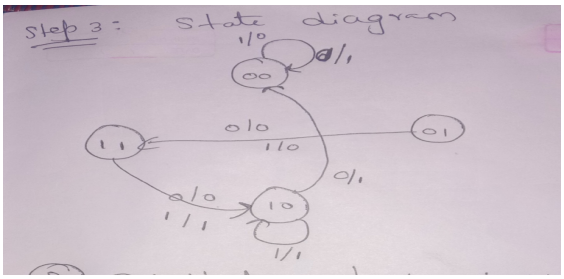


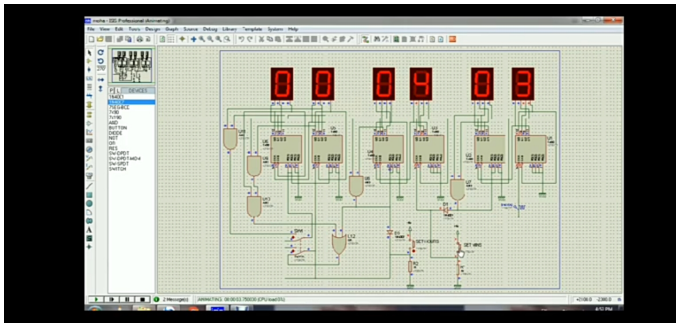
Report – Report can be typed or hand written for up to two pages.

## 1. Analysis of clocked sequential circuits





## 2. Digital clock signal



A digital clock is a one kind of clock used to display the time in the form of digital includes symbols or numerals. These clocks are frequently connected with electronic drives, but the term digital refers only to the LCD display, not to the drive mechanism. The digital clock circuit uses the 50-60hz oscillation of AC power.Most digital alarm clocks display the hour of the day in the form of 12 hours or 24 hours with an indication of AM or PM. Most digital alarm clocks use LCD display, seven segment display or VFD.

Digital clocks run with mains electricity and must be reset the time when the power is off. Most of the clocks don't have a battery back up, so this will cause to fail to generate an alarm sound at the fixed time. To overcome this problem, many digital alarm clocks are available to operate with a battery during the power outage. Commercial digital clocks are generally more consistent than consumer clocks. Because, these clocks give backup to maintain the time using multi decade battery during power off.

8051 Microcontroller based Digital Alarm Clock with LCD Display

The required components of this 8051 microcontroller based digital clock circuit with LCD display mainly include LCD display, AT89C51 Microcontroller, Preset, piezo buzzer and speaker. The function of each and every component of this project is discussed below.

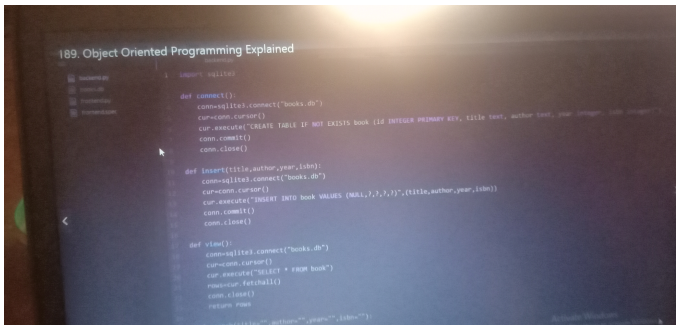
LCD Display

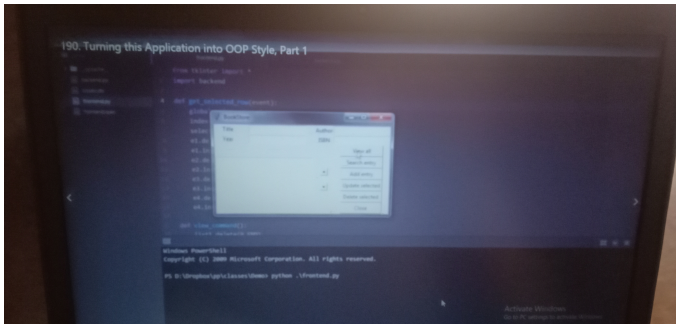
A 16×2 LCD display is an electronic display and it is used in a wide range of applications.These kind of displays is used in a multi segment LEDs an 7-segment displays. In this LCD display, each character is shown in 5×7 pixel matrix. This LCD display consists of two registers, they are data register and command register. A command register is an order for LCD display to do a task like clearing of its screen, initializing, controlling of display and cursor position setting. The data (ASCII value of the character) register is used to display the stored data on LCD display.

| | | | |
|---|---|---|---|
| Date: | 29/5/20 | Name: | Ankitha c c |
| Course: | Python | USN: | 4al16ec004 |
| Topic: | 1. Objects orientation | Semester & Section: | 8th a |

AFTERNOON SESSION DETAILS

Image of session




Report – Report can be typed or hand written for up to two pages.

# 1. Objects orientation

GUI in OOP Design (Practice)

Alter the frontend.py script containing the GUI code by changing its functional-oriented design into an OOP design.

For your convenience, the files frontend.py, backend.py (in OOP style), and the book.db files are attached in this article's resources.

Resources for this lecture

exercise-files.zip

Solution

Here are the frontend.py and backend.py scripts in OOP style. To execute this program you should execute the frontend.py file.

#frontend.py

```
from tkinter import *
from backend import Database
database=Database("books.db")
class Window(object):
    def __init__(self,window):

   self.window = window
   self.window.wm_title("BookStore")

    l1=Label(window,text="Title")
      l1.grid(row=0,column=0)

      l2=Label(window,text="Author")
      l2.grid(row=0,column=2)

      l3=Label(window,text="Year")
      l3.grid(row=1,column=0)

      l4=Label(window,text="ISBN")
      l4.grid(row=1,column=2)

      self.title_text=StringVar()
      self.e1=Entry(window,textvariable=self.title_text)
      self.e1.grid(row=0,column=1)

      self.author_text=StringVar()
      self.e2=Entry(window,textvariable=self.author_text)
      self.e2.grid(row=0,column=3)
```

```
        self.year_text=StringVar()
        self.e3=Entry(window,textvariable=self.year_text)
        self.e3.grid(row=1,column=1)

        self.isbn_text=StringVar()
        self.e4=Entry(window,textvariable=self.isbn_text)
        self.e4.grid(row=1,column=3)

        self.list1=Listbox(window, height=6,width=35)
        self.list1.grid(row=2,column=0,rowspan=6,columnspan=2)

        sb1=Scrollbar(window)
        sb1.grid(row=2,column=2,rowspan=6)

        self.list1.configure(yscrollcommand=sb1.set)
        sb1.configure(command=self.list1.yview)
         self.list1.bind('<<ListboxSelect>>',self.get_selected_row)

            b1=Button(window,text="View all", width=12,command=self.view_command)
            b1.grid(row=2,column=3)

            b2=Button(window,text="Search entry", width=12,command=self.search_command)
            b2.grid(row=3,column=3)

            b3=Button(window,text="Add entry", width=12,command=self.add_command)
            b3.grid(row=4,column=3)

            b4=Button(window,text="Update selected", width=12,command=self.update_command)
            b4.grid(row=5,column=3)
```

```
        b5=Button(window,text="Delete selected", width=12,command=self.delete_command)
        b5.grid(row=6,column=3)

        b6=Button(window,text="Close", width=12,command=window.destroy)
        b6.grid(row=7,column=3)

    def get_selected_row(self,event):
        index=self.list1.curselection()[0]
        self.selected_tuple=self.list1.get(index)
        self.e1.delete(0,END)
        self.e1.insert(END,self.selected_tuple[1])
        self.e2.delete(0,END)
        self.e2.insert(END,self.selected_tuple[2])
        self.e3.delete(0,END)
        self.e3.insert(END,self.selected_tuple[3])
        self.e4.delete(0,END)
        self.e4.insert(END,self.selected_tuple[4])
    def view_command(self):
        self.list1.delete(0,END)
        for row in database.view():
            self.list1.insert(END,row)

    def search_command(self):
        self.list1.delete(0,END)
        for row in database.search(self.title_text.get(),self.author_text.get(),self.year_text.get(),self.isbn_text.get()):
            self.list1.insert(END,row)

    def add_command(self):
```

```
            database.insert(self.title_text.get(),self.author_text.get(),self.year_text.get(),self.isbn_text.get())
            self.list1.delete(0,END)
            self.list1.insert(END,(self.title_text.get(),self.author_text.get(),self.year_text.get(),self.isbn_text.get()))


        def delete_command(self):
            database.delete(self.selected_tuple[0])


        def update_command(self):

database.update(self.selected_tuple[0],self.title_text.get(),self.author_text.get(),self.year_text.get(),self.isbn_text.get())
    window=Tk()
    Window(window)
    window.mainloop()
```

And below you will also find the backend.py script in OOP:

```
    #backend.py
    import sqlite3
    class Database:
        def __init__(self, db):
            self.conn=sqlite3.connect(db)
            self.cur=self.conn.cursor()
            self.cur.execute("CREATE TABLE IF NOT EXISTS book (id INTEGER PRIMARY KEY, title text, author text, year
integer, isbn integer)")
            self.conn.commit()
        def insert(self,title,author,year,isbn):
            self.cur.execute("INSERT INTO book VALUES (NULL,?,?,?,?)",(title,author,year,isbn))
            self.conn.commit()
        def view(self):
            self.cur.execute("SELECT * FROM book")
            rows=self.cur.fetchall()
```

```
            return rows
        def search(self,title="",author="",year="",isbn=""):
            self.cur.execute("SELECT * FROM book WHERE title=? OR author=? OR year=? OR isbn=?",
(title,author,year,isbn))
            rows=self.cur.fetchall()
            return rows
        def delete(self,id):
            self.cur.execute("DELETE FROM book WHERE id=?",(id,))
            self.conn.commit()
        def update(self,id,title,author,year,isbn):
            self.cur.execute("UPDATE book SET title=?, author=?, year=?, isbn=? WHERE id=?",(title,author,year,isbn,id))
            self.conn.commit()
        def __del__(self):
            self.conn.close()
```

Resources for this lecture

frontend.py