A model design in DMD technology: Smart Scrolling Display at 10Hz in 64×128 LED Array.

Abhijit Banerjee[1,*], Paramita Mondal[2], and Soham Chatterjee[3]
[1]*State Aided College Teacher, Department of Computer Science, New Alipore College (Affiliated to University of Calcutta), Block-L, New Alipore, Kolkata-700053*
[2]*State Aided College Teacher, Department of Electronics, New Alipore College (Affiliated to University of Calcutta), Block-L, New Alipore, Kolkata-700053*
[3]*M.Sc., Department of Computer Science, Institute of Management Study (Affiliated to Maulana Abul Kalam Azad University of Technology), EM Bypass, 93 Mukundapur, Kolkata-700099*

***Abstract*— Information displaying is the communicational part of the electronics plays a key role in the present era. Digital advertisement is the current requirement of modern businesses and the information world. Although Light Emitting Display (LED) based Dot Matrix Display (DMD) is an effective mode on displaying information but the complicated task is to make the message dynamic as the users have to change the message content according to this specific requirement. In case of changing the message content user has to connect the LED display to the master system, so that the display board cannot be placed elsewhere because of dedicated and complex connectivity. The present work provides the facility of real time message displaying by the user. So the modifiable message content via Serial control can be displayed. A mobile is used to send the message to the LED display and the dual core processor does all the processing and displaying output of this message in different patterns at 10Hz. and in 64×128 LED array.**

***Index Terms*—DMD, LED, ESP32, Serial, HUB75.**

I. INTRODUCTION

The human expressions and communications in terms of audio or video is very innate. Early humans recorded or displayed information by petroglyphs. As time passed digital storage devices and displays (after the invention of Diodes) were developed. Fritz Karl Preikschat filed five patent applications for the "PKT printer", a dot matrix teletypewriter built between 1954 and 1956 in Germany [1]. Later, between 1966-1967 an improved transistorized design became the basis for a portable dot matrix facsimile machine, which was prototyped and evaluated for military use by Boeing [2-3]. IBM marketed its first dot matrix printer in 1957, the same year that the dye-sublimation printer entered the market [4]. Nick Holonyak invented the first practical LED in 1962 at General Electric [5]. In 1968, Hewlett-Packard (HP) developed the first LED display [6] and in 1969 introduced the HP Model 5082-7000 numeric Indicator which became the first intelligent LED display using integrated circuit technology [7]. This innovation replaced the Nixie tube and was a major milestone in LED technology, forming the foundation for future LED displays widely used in electronics and signage [7]. Early models were monochromatic by design. The efficient Blue LED completing the color triad did not commercially arrive until the late 1980s [8]. Mark Fisher's design for U2's 1997 PopMart Tour introduced an innovative use of LED technology for large-scale displays. He utilized wide pixel spacing for long-distance viewing, creating a 52m wide by 17m high screen with 150,000 pixels.

One of such Dot Matrix Display (DMD) based on P10 (10mm pixel pitch Panel) board developed in recent years [9]. In the present work we have implemented various processing and displaying functions by using DMD-P10 board comprised of LED with Red, Green and Blue (RGB) using a 160Mhz. frequency based dual core processor viz. Expressif's-32 (ESP32) and analyses its test characteristics. In compare to earlier designs in this respect this work significantly analyses the functions viz. potential developed for the arrangement of each alphabets/symbols, potential for phrases passed though the output and time of activation of pixels during occurrence of the information. This introduces a critical analyses when reviewed after the estimation of the above stated functions at the post completion and during the final stage of testing.

II. METHODOLOGY

A. Required Tools & Modules in brief

1. ESP32 Development Board: The ESP32 as shown in Figure 1 has two Xtensa LX6

processor cores, allowing one core to handle real-time tasks and the other for communication or other operations like receiving commands via serial also serves as the controller for the P10 panels. This division of labour ensures smooth and uninterrupted operation of the display.

Operating at up to 160-240 MHz, the ESP32 provides sufficient speed to handle high refresh rates for large LED displays, even with computationally intensive tasks like multiplexing, colour rendering and animation processing.

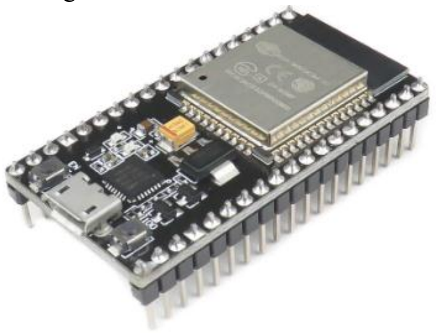
Figure 1. ESP-32-Xtensa LX6 processor cores

The ESP32 can output data rapidly through its Serial Peripheral Interface (SPI) interfaces and General Purpose Input Output (GPIO) pins, essential for driving the HUB75 protocol of the P10 panels.

With up to 520 KB of SRAM, the ESP32 can store frame buffers, design patterns and text strings in memory.

For a 64x128 (: row×column) array RGB display, the managing of the individual pixel colours requires a frame buffer. Assuming 1 byte per colour channel (3 bytes/pixel), the ESP32 can manage this buffer efficiently.

ESP32 modules come with 4 MB or more of flash storage, which can store pre-defined text, designs, animations, or custom fonts.

2. Four 64x32- P10- RGB-LED panels: P10 panels as shown in Figure 2 have to be arranged in a 2x2 grid to achieve a 64x128 resolution.

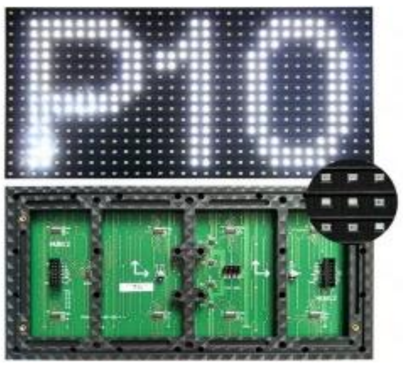
Figure 2. Frontal and back view of Four 64x32-P10- RGB-LED panels

3. 5-Volt Power Supply:
   The ESP32 has multiple power modes, but for driving P10 panels, it typically operates in full power mode.

Figure 3. Compact view of SMPS

   The GPIO pins must handle high-speed switching to maintain display fidelity.
   Proper heat management is essential, as sustained operation at full load generates heat in both the ESP32 and the P10 panel's power circuitry.
   The DL brand Switch mode power supply (SMPS) shown above in Figure 3 is capable of providing at least 10-15A (each P10 panel typically consumes ~2.5-3A at full brightness).
4. HUB75 Connectors: Connects ESP32 to P10 panels. HUB75 connector shown in Figure 4.

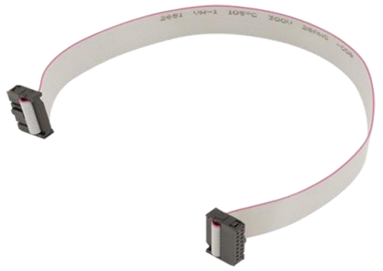
Figure 4. HUB75 connector

5. Wires and Connectors: For power and data connections. AC plug, cord and FF/MM/MF connecting cables are included with heath shrink tubes.

B. Panel Setup and Connections

1. Arrange the Panels:
   Place the P10 panels in a 2x2 configuration (e.g., top-left, top-right, bottom-left, bottom-right).
2. Daisy-Chaining:
   Using HUB75 interface used by P10 panels requires precise control over 13 signals. Data flows from the ESP32 to the top-left panel, then cascades to the top-right, bottom-left, and finally bottom-right as shown in Figure 5.

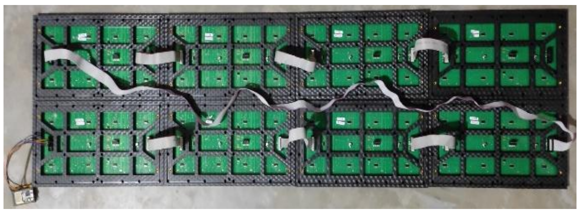
Figure 5. Connection between HUB75 interface and P10 panels

C. ESP32 to P10 Pin Mapping

Connect the ESP32 GPIO pins to the HUB75 input of the first panel as follows:

Table 1: HUB75 and ESP32 connectors

HUB75 Signal	ESP32 GPIO Pin
R1	GPIO 25
G1	GPIO 26
B1	GPIO 27
R2	GPIO 14
G2	GPIO 12
B2	GPIO 13
A	GPIO 32
B	GPIO 33
C	GPIO 15
D	GPIO 4
CLK	GPIO 18
LAT	GPIO 19
OE	GPIO 23

RGB Data Lines (R1, G1, B1, R2, G2 and B2): Control the colours for the two rows being scanned simultaneously.

Row Select Lines (A, B, C, D): Determine which row is active during the current scan cycle.

Control Lines (CLK, LAT, OE):

CLK (Clock): Synchronizes data shifts.

LAT (Latch): Latches the current data into the panel's registers.

OE (Output Enable): Controls brightness and turns the LEDs on/off during the scan.

D. Embedded Software Setup

1. Development Environment: Install Arduino IDE.
2. Add support for the ESP32 board (via the ESP32 Arduino core).
3. In Arduino IDE, go to Tools → Manage Libraries and search for PxMatrix, Adafruit GFX, ESPAsyncSerial and Arduino Core for ESP32 library.
4. Install the PxMatrix, Adafruit GFX, ESPAsyncSerial and Arduino Core for ESP32 library.

The PxMatrix library is specifically designed to control LED matrix displays (including HUB75 P10 panels) using MHz processor like the ESP32.

Adafruit GFX library provides graphics rendering on processor displays. It provides a high-level Application Programming Interface (API) for drawing shapes, text and images.

Arduino Core for ESP32 allows programming the ESP32 in the Arduino IDE. It includes drivers for GPIO, Wi-Fi, Bluetooth, Pulse width modulation (PWM) and more.

ESPAsyncSerial: For serial communication to update text and animations dynamically.

5. Features of the Code (as per requirement)

Serial Commands:

The ESP32's hardware UARTs allow for serial communication thus enabling users to send commands (e.g., text updates) via the serial monitor.

The Universal Asynchronous Receiver Transmitter (UART) operates at high baud rates (e.g., 115200), ensuring low latency for updates.

TEXT:<message>: Displays the given <message> on the panel.

BRIGHTNESS:<value>: Adjusts brightness (0-255).

CLEAR: Clears the display.

DESIGN:<type>: Displays a predefined design (e.g., SMILEY, BOX).

6. Dynamic Updates:

Text and designs can be updated in real-time via the serial monitor.

Input from the serial monitor is parsed to distinguish between text, brightness adjustments or custom designs.

Efficient parsing routines ensure that the display refresh task remains uninterrupted.

7. Basic Designs:

Supports simple shapes like smiley faces or rectangles as examples. It can be extended to include more complex patterns.

8. Analyses:

The final stage outputs are analyzed through the real time checking voltage vs. time of activations by downloading the runtime ".csv" file from the IDE platform for each pass (iterations) and the validating responses from the serial monitor.

The python programming platform are utilized for evaluations and plotting of the different measures of the parameters related to pixel activation for alphabets scrolling and thus by determination of the transiency in the response.

III. FLOW OF WORK

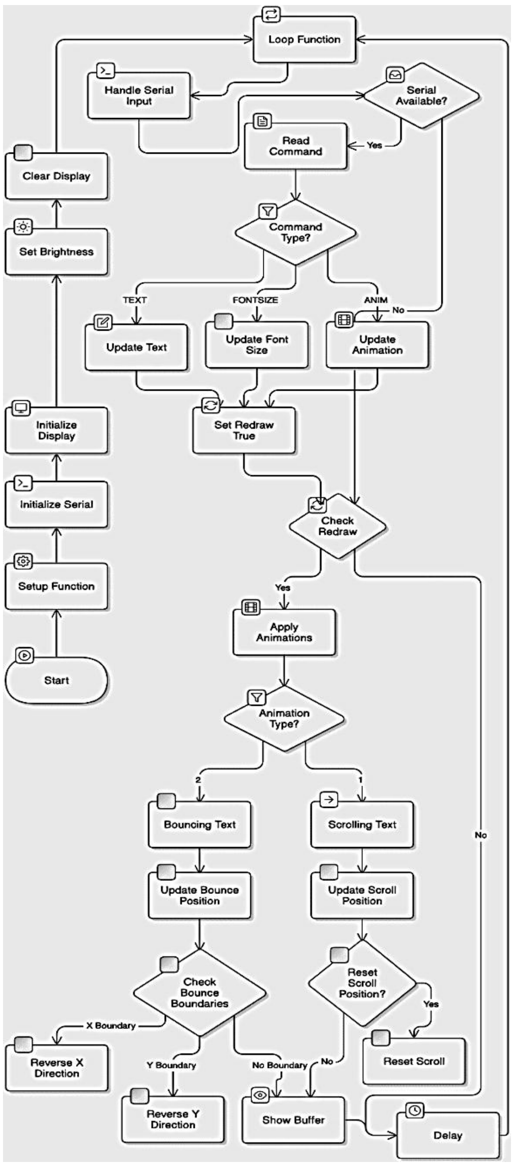

Figure 6. Process flow chart

A. The *process* begins when the ESP32 is powered on and the code starts execution as shown in flowchart Figure 6.

B. *Setting-up* begins with
Initialize Serial Communication: Begin communication with the PC for serial commands.
Initialize Display: Configure the P10 LED panel with the PxMatrix library.
Set Brightness and Clear Display: Ensure the display starts with the default brightness and no content.

C. Loop: (Here the codes repeats)
Check Serial Input: If serial input is available, handle it in handleSerialInput().
Commands such as TEXT, FONTSIZE or ANIM are processed.

Update the current Text, font Size, and animation Type variables dynamically.

Redraw Display: If the redraw flag is set call apply Animations () to update the display.
Delay: A small delay ensures smooth operation and controlled refresh rates.

D. Handle Serial Commands
Parse and process commands received via the serial monitor:
Update text: TEXT:<text>
Update font size: FONTSIZE:<size>
Update animation: ANIM:<type>
Set the redraw flag to ensure animations are refreshed.

E. Apply Animations
Animation Type 1: Scrolling Text
Display text scrolling horizontally.
Adjust the position of the text for smooth scrolling.
Animation Type 2: Bouncing Text
Display text bouncing diagonally.
Reverse direction when hitting the screen edges.
Update Display Buffer: Render the current frame to the P10 display.

F. The process loops continuously allowing real-time updates.

IV. RESULT AND DISCUSSION

A. Data Obtained from P10 DMD for unishade text
ROW REFRESH: Activates one row at a time and sends data for that row.
SCROLLING: Dynamically shifts the pixel data for smooth motion effects.
SET, Color = BLUE HEX: #0000FF RGB: 0, 0, 255
Text = NEW ALIPORE COLLEGE
Brightness (in Voltage) = 1.8-3.3 for 0-255
OUTPUT Table 2:
Where,
P_c is Characters in Pixels
A_{LED} Number of Active LED`s
V_T Total Voltage for Active LED`s

Table 2. Active LED vs. Voltage per characters

P_c	A_{LED}	V_T
N	12	21.58V
E	10	18.01V
W	12	21.60V
A	10	17.97V
L	8	14.38V
I	4	7.21V
P	10	18.02V
O	12	21.62V
R	10	17.96V
E	10	18.00V
C	8	14.43V

O	12	21.55V
L	8	14.40V
L	8	14.35V
E	10	18.00V
G	10	17.95V
E	10	18.15V

Where, P_n = Pixel Number

and, $V_T = P_1 + P_2 + P_3 + \ldots\ldots + P_n$

SET, DISPLAY Frequency = 60Hz

OUTPUT Table 3:

Where,

P_c is Characters in Pixels

T_s Time Frame in Seconds

V_T Total Voltage for Active LED`s

Table 3. Time frame vs. voltage per character

P_c	V_T	T_s
N	21.58V	0.17
E	18.01V	0.34
W	21.60V	0.51
		0.60
A	17.97V	0.85
L	14.38V	1.02
I	7.21V	1.19
P	18.02V	1.36
O	21.62V	1.53
R	17.96V	1.7
E	18.00V	1.87
		1.99
C	14.43V	2.21
O	21.55V	2.38
L	14.40V	2.55
L	14.35V	2.72
E	18.00V	2.89
G	17.95V	3.06
E	18.15V	3.23

N.B. Occurrence of blanks in the table signifies ASCII character as Space

B. Graph Obtained from the Tables above

1. Graph in Figure 9 is obtained from above estimated Table 2.

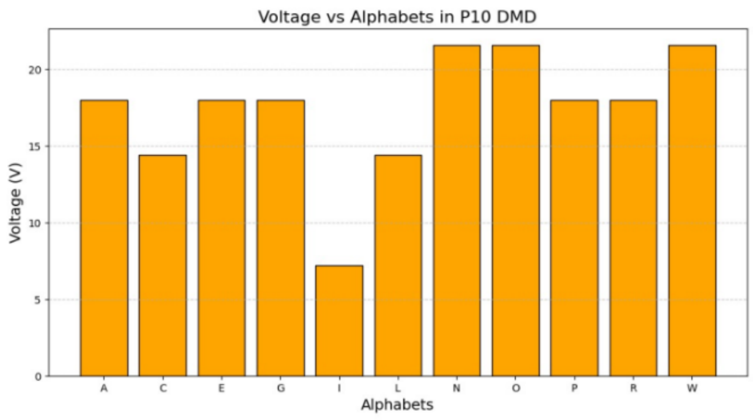

Figure 9. Average variation of Voltage vs. Alphabets with P10 DMD for 30 passes (iterations)

The X-axis shows voltage (in v) for each row refresh. The Y-axis shows the number of active pixels (lit LEDs) for each row during the scrolling of the phrase "NEW ALIPORE COLLEGE".

It was found out that V_T for same characters are non-identical for difference of voltages between two or more pixels.

2. Graph in Figure 8 is obtained from above Table 3.

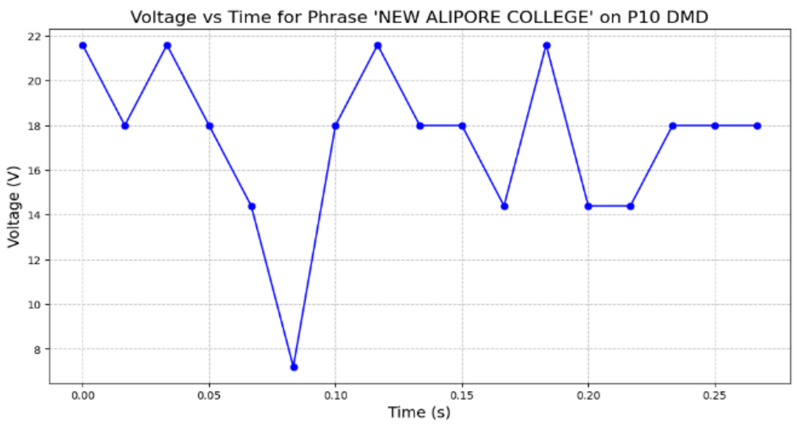

Figure 10. Average variation of voltage vs. Time for Phrase “New Alipore College” on P10-DMD for 30 passes (iteration)

The X-axis shows time (in milliseconds) for each row refresh. The Y-axis shows the number of active pixels (lit LEDs) for each row during the scrolling of the phrase "NEW ALIPORE COLLEGE".
It was found out that T_s for whitespace characters and real characters are different as the refresh rate due to no processing of pixel is different.

3. Graph in Figure 11 is obtained from combining the Table 2 and 3 considering all the passes (iteration) attain previously.

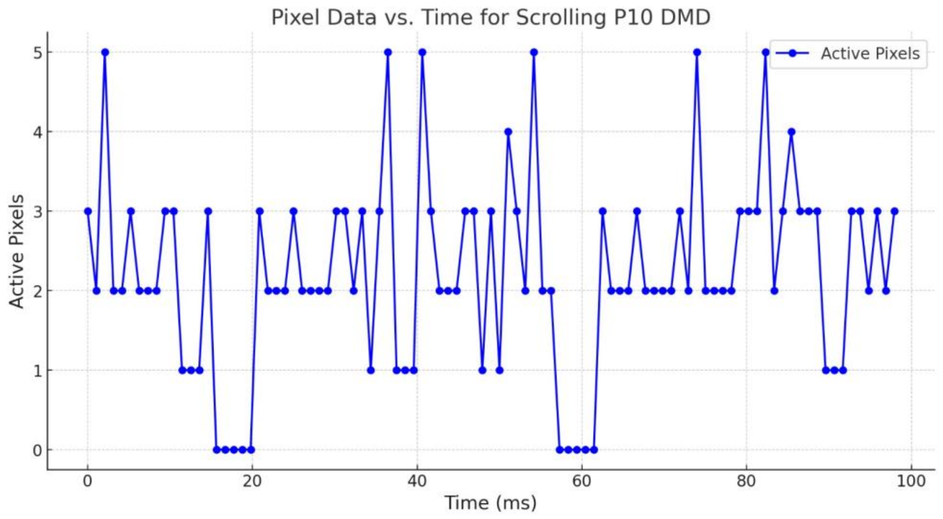

Figure 11. Active Pixel vs. Time for Scrolling P10 DMD

The X axis shows number of Active Pixels (P_A)
The Y axis shows Time (in Milliseconds)
C. Panel Outputs:
Font: Default
Font Thickness: 2PT

Some of the outputs are given in Figure 12 for Real time displaying of the phrase “NEW ALIPORE COLLEGE” in P10 DMD through frame 1-3.

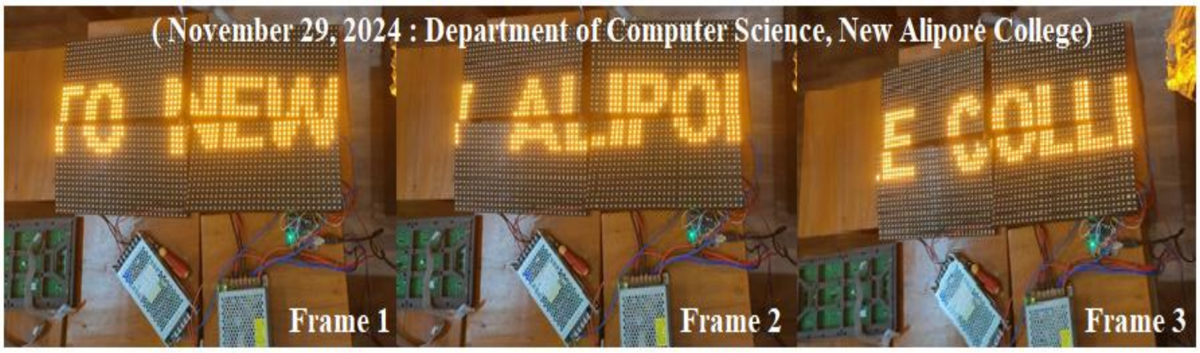
(November 29, 2024 : Department of Computer Science, New Alipore College)
TO NEW
ALIPO
E COLLI
Frame 1
Frame 2
Frame 3

Figure 12. A view of real time display of the phrase “NEW ALIPORE COLLEGE” on P10 DMD via frame 1-3

V. CONCLUSION

Programming a 64x128- P10- RGB LED panel using an ESP32 provides a flexible and cost-effective solution for creating visually appealing, real-time customizable displays. The system's ability to accept serial commands for scrolling text, font type, font size, font animations and graphic animations makes it highly versatile and user-friendly. It demonstrates the power of integrating hardware like the ESP32 with libraries such as PxMatrix and Adafruit GFX to achieve smooth performance and rich graphics on LED displays.

This project highlights how advancements in microcontroller technology can enable scalable and interactive solutions for dynamic visual communications. The serial interface ensures adaptability allowing users to customize the display in real-time without reprogramming the device.
Future Scope:
Wireless Connectivity: Implement wireless control using Wi-Fi (via a web interface) or Bluetooth to eliminate the need for a physical serial connection.

Real-time Data Integration: Display real-time data such as weather updates, stock prices or IoT sensor readings fetched from APIs over the internet.
Multi-panel Synchronization: Develop solutions for synchronizing multiple P10 panels across larger display setups, creating seamless video walls.
Mobile App Integration: Create a mobile app that allows users to control text, animations and designs intuitively.
Font and Language Expansion: Add multi-language support and allow users to upload custom fonts or emoji.
Interactive Features: Incorporate touch sensors or external inputs for interactive applications such as voting systems or real-time message boards.

ACKNOWLEDGMENT

We sincerely acknowledge New Alipore College, affiliated to University of Calcutta for granting financial support and special laboratory set up for the present work.
The open-source embedded processors and integrated development environment (IDE) of Arduino, Esspresif, and others organization are used due course of this work for which we are highly thankful.
Prototyping development kits and boards are brought from Robu.in. Credit for the images, of devices partly utilized here for showcasing the elements present in the circuitry is given to robu.in.

REFERENCES

[1] Preikschat Fritz Karl, “Working papers on dot matrix teletypewriter”, Published: 1961.
[2] “Facsimile transponder prototype at Boeing”, Published: 06/02/1967.
[3] “Drawings of portable fax machine for Boeing”, Published: 30/08/1966.
[4] "1957: IBM introduces the first dot-matrix printer", CNN, by Mary Brandel on May 12, 1999.
[5] Kramer Bernhard, “Advances in Solid State Physics”, 2003, Springer Science & Business Media, pp.40, ISBN 9783540401506.
[6] “Hewlett-Packard 5082-7000”, The Vintage Technology Association, 1969.
[7] Borden, Howard C and Gerald P, "Solid-State Displays", February 1969.
[8] "An Introductory Guide to LED Display Technology and its Uses & Benefits". Dynamo LED Displays. By Reynolds Daniel.
[9] "News – Press Centre | LG Display" by lgdisplay.com.
[10] “Light Emitting Diode Television Screen”, The prototype and scientific paper, EXHIBIT #635.
[11] “Announcement of Awards”, published by the Science Service from 29th ISEF pp. Dated: May 1978.
[12] Mitchell's modular LED x-y (horizontally and vertically digitally scanned array system) was cited in the 29th International Science and Engineering Exposition “book of abstracts”, pp. 97, published by the "Science Service", Washington D.C. Dated: May 1978.