The Journal of Open Engineering

A 320x256 RGB LED Electronic Display Controlled via WiFi from Mobile Application

**Kevin Etta, Caroline Gilger, Andrew Maatman, Tuoxuan Ren,
Logan Wedel, Chandana Tamma, Cristinel Ababei**

Published on: Feb 06, 2020

DOI: 10.21428/9d720e7a.9d49c2cd

License: [Creative Commons Attribution 4.0 International License \(CC-BY 4.0\)](https://creativecommons.org/licenses/by/4.0/)

ABSTRACT

We present the design of a 320x256 RGB LED electronic display board intended to be used as a platform to present information of current and future events in schools or office buildings. The proposed design can display up to eight different images, which may include text and graphics. The rendering of these images is done periodically, with the duration of the period being controlled by the user. The images can be created from slides of a power point presentation or directly from stored photos. Loading of new images to the display board is done wirelessly via WiFi from an Android App running on a portable device such as a tablet or a smartphone. We present experimental results on a completed and fully operational prototype system, which includes the RGB LED display board together with the custom-built microcontroller board as well as the Android App. While the current use-application is as an informative message board in a school set-up, the proposed system can be used in any other indoor application. For outdoor usage, the system would require an additional weather-proof enclosure and fixtures.

Metadata Overview

Main design files: https://github.com/drewsum/Electronic_Display (archived version)

Target group: undergraduate students, trained engineers, scientists in embedded systems and image processing

Skills required: Hardware assembly – easy; surface mount PCB – advanced; embedded C firmware – easy; Android programming – easy

Keywords

RGB LED; Electronic display; WiFi communication; Open source

1. Introduction

There are generally three main categories of electronic displays. In the first category, simpler and more traditional, one can find the systems that use character LCD displays with two or more rows and 10, 20 or other number of characters per row, such as the 16x2 character LCD [1]. This type of display can typically display static or moving text messages [2]. The second category includes all systems that use graphic LCD displays. Examples in this category include small embedded LCD and touch-screen TFT displays (e.g., 320x240 pixels), digital photo displays, and systems that use LCD monitors as the primary method to display text and images [3]. Finally, in the third category, we can find RGB LED displays. Examples in this category include mostly outdoor scoreboard displays as well as highway and business side billboards. Their complexity and costs are high and depend on the size, which is typically large; see for example the commercial products from Daktronics [4]. However, one can find smaller

size examples created by hobbyists, including designs that use just one 64x64 RGB LED matrix panels [5] or just one 32x32 matrix panel [6]. The advantages of RGB LED displays compared to the color LCD displays include better visibility in sunlight conditions, better contrast for crisper details, purer colors, and reduced reflection. However, their power consumption is typically large and their cost relatively high. Finally, while most of these example designs are controlled by microcontrollers, there are solutions that use field programmable gate arrays (FPGAs) as well for improved speeds [7].

Literature Review

The problem of designing and prototyping large RGB LED displays has been less researched in the past literature. That is because this is more a problem of development rather than of theoretical research and because there exists already many commercial offerings of such displays from established companies, some of which were mentioned in the previous paragraph. Nevertheless, one can find prior research studies on power supplies for RGB LED displays or especially on new circuit solutions for LED drivers. For example, the study in [10] proposed a microcontroller-based driver whose objective was to maximize the LED efficiency. They reported a prototype for a 3x3 LED matrix and achieved efficiency greater than 92%. A dimmable multi-channel RGB LED driver was reported and studied with a nine LED string in [11]. Energy savings was the objective of the LED driver studied in [12]. The driver uses a multilevel pulse width modulation (MPWM) driving method combined with simple video signal conversion techniques. It was demonstrated for a 32x32 LED pixels display. The study in [13] addresses the challenges of active image adjustments and processing, considering the non-linearity of the LED display element. They propose a method for the nonlinear adjustment of the images contrast. The advantage of the proposed method is that the images can be displayed on OLED or LED displays with the unified luminance signal, for grayscale images, or with separate RGB controls. Other studies focused on the design of individual LEDs; such an example is the study in [14] that proposed solutions based on dielectric compound parabolic concentrators (CPCs) in order to increase brightness. Such studies are outside the scope of our paper, which focuses on the system integration design aspects, both hardware and software, and prototyping of large RGB LED displays.

Contribution

In this paper, we present the completely open-source design of a 320x256 RGB LED electronic display board intended to be used a platform to present information of current and future events in schools or office buildings. While the current use-application is as an informative message board in a school set-up, the proposed system can be used in any other indoor or outdoor application (for outdoor usage, the system will need weather-proof casing though). Designed to be cost-effective yet extendable, we hope this design will be adopted and improved in other practical and educational settings.

2. Overall implementation and design

2.1 Hardware description

System-level Diagram

The system-level diagram of the overall RGB LED display system is shown in Fig. 1. The major components include the 320x256 LED display created by an array of 5x4 of individual 64x64 LED panels – as the hardware (HW) – and the Android App – as the software (SW).

Figure 1: System-level diagram of the proposed electronic display.

Block Diagram of the Electronic Display System

The block diagram of the electronic display is shown in Fig. 2. The actual 320x256 display is constructed as an array with five rows and 4 columns, 5x4, of individual 64x64 RGB LED panels. The main controller algorithm is executed on a PIC32MZ2048EFH144 microcontroller unit (MCU) running at 252MHz, with 2MB of flash memory, 512KB of SRAM, and 32-bit bus width.

Figure 2: Block diagram of the electronic display system.

The integrated 2MB of flash memory of the MCU is used as the primary frame buffer that stores the currently displayed image on the display. It is shown as the “Frame Buffer 2” in Fig. 2. An additional external 2MB parallel SRAM chip is used to store the image to be displayed next. This memory, indicated as “Frame Buffer 1” in Fig. 2, effectively plays the role of a cache between the MCU and the memory system formed by eight different Flash memory chips indicated as “Flash 1..8” in Fig. 2. The eight 2MB flash chips store the current presentation slides or “project”, with one slide per flash chip. The flash chips—physically placed close to the microcontroller to reduce communication delays—were chosen due to their low cost, ease of use, and large number of write cycles (100,000 cycles typically). High performance is achieved by the MCU’s 50MHz External Bus Interface (EBI) feature that is exploited to fetch image data into the primary frame buffer, “Frame Buffer 2”. The WiFi module is the popular and low cost ESP8266 module, which provides its own connection to the device that runs the Android App and thus it does not require a separate Internet service. A new “project” represents eight different images, which are uploaded via the WiFi connection from the App to the eight flash chips.

For debugging and programming purposes, an FT234 USB-UART bridge is included in the design, with its ground isolated from the rest of the system’s electric circuits. To power the RGB LED panels, an off-the-shelf 5V 50A power supply is used while the controller board is powered by a separate 12V 50A off-the-shelf power supply.

Controller Board

The logic or controller board is responsible with the control of the 320x256 pixel array of the electronic display. As described earlier, the control algorithm is executed on a 252MHz, 144 pins, 32 bit PIC microcontroller. This microcontroller was selected because of its fast clock frequency and its high pin count, which provided satisfactory performance for 320x256 pixel display. The front and bottom layout snapshots created by the KiCAD layout tool together with a photograph of the prototype board completely populated with components and mounted on the back of the display system are shown in Fig. 3.

Figure 3: Images of the logic/controller board. Front side of layout (top), back side of layout (middle), and photograph of the actual prototype installed on the back of the panels (bottom).

2.2 Software description

Embedded Software

The microcontroller executes the control algorithm that drives the 5x4 array of 64x64 pixels LED panels. Each panel is separated into two halves, with each half having three 64-bit long shift registers for red, green and blue data. The data loaded into these shift registers is used to drive the LEDs on the panel depending on which row is selected using the row select signals. The panels are arranged into 4 rows (or strings) and each row is five panels long (i.e., has five columns). This arrangement requires 24 shift registers, each of which are 320 bits long, for a 320x256 pixel display. The row select control lines

set where the data shifted into these shift registers appear on the display, vertically. The microcontroller's speed is vital here since if data is moving too slowly onto the panels, a visible flicker will be noticed. The microcontroller simply shifts the data within the byte array onto the RGB pins and updates the row select signals as fast as it can. It changes the rows that are displayed at a rate of about 12KHz. The flow chart describing the algorithm for driving the display is shown in Fig. 4.

Figure 4: Flow chart of the MCU algorithm used to control—shifting and displaying—the LEDs on the panels over time.

In addition to driving the display, the microcontroller also has several other tasks to service. During normal operation, up to eight images are stored across eight 256KB serial flash memory chips. This external non-volatile memory acts as long-term image storage, and stores image data across MCU resets and power cycles. It is accessed through a Serial Peripheral Interface (SPI). The MCU can read and write entire images worth of data to a flash chip in about three seconds. When a project is programmed into the system, each image is stored in the SPI flash, while the number of images for the project and the cycle time between images is stored into the microcontroller's internal flash.

The microcontroller also has an external parallel 256KB SRAM. It is able to read and write data to this SRAM through the External Bus Interface (EBI). EBI allows external memory to be attached to the MCU and it treats it just like any other virtual memory address space. This allows data in EBI SRAM to be treated like a normal byte array in C firmware. During normal operation, while an image is being

shown on the screen, the next image to be displayed is moved from SPI Flash into EBI SRAM. When the image is ready to be displayed, it is moved all at once from EBI SRAM into the internal frame buffer and shown on the screen. In this memory structure, EBI SRAM acts as a data cache, since moving data from EBI SRAM into internal RAM is much faster than moving data from SPI Flash into internal RAM. Using this technique, image transitions appear seamless.

The MCU has a serial interface with a WiFi module included on the logic board. This is the main interface between the Android app (described in the next section) and the system. Through the WiFi interface, the user can turn the screen on and off, adjust the brightness of the screen, and load new projects into the system. The WiFi module and the MCU communicate through a UART interface. A ring buffer in MCU firmware allows massive amounts of data to be transferred at a time from the WiFi module into the MCU.

Smartphone Application

A custom Android App, named MU-MatriX (available already for download on Google Play) was developed to allow the user to create and upload to the display system new “projects”. A “project” consists of eight images that form a presentation displayed periodically at a configured interval. The app stores each project’s name, set of images, and duration between image switch into its internal application storage. After naming a new project, a user can populate the project with images, selecting them one-by-one from their device. The images can maintain the aspect ratio of the original image or stretch to fit the size of the display board. There is also the option to rotate images in 90 degree intervals before adding them into the project. In the background, any selected image is saved off into a 320x256 bitmap with black background filling in any gaps created by the aspect ratio lock. As images are being added to a project, they can be reordered by simple up and down arrows next to their thumbnail on the “Edit Project” screen. A spinner widget provides a drop-down selection for the project-level property of duration between slides.

The default display board device is automatically stored under the user’s device list on app installation. The IP address and port number used by default for the ESP8266 WiFi module (on the logic board) are simply placed under a device name “Default Display Board”. The app allows more than just this one display board configuration to be saved in memory, allowing for this system to scale to multiple screens all managed under one app. A device control screen gives the app simple commands to send the Wifi module on the Logic Board: Power On, Power Off, and a screen brightness slider that sends a numeric value between 5 and 100. Simple WiFi messages holding single-line commands are sent over the network using the TCP protocol. Uploading a project to the Logic Board from the application takes a little bit more work and is given its own screen within the app. After selecting one of the projects, one of the hardware devices, and pressing “Upload”, a progress bar and message will become visible on

the screen tracking the conversion and transmission of each image within that project. After all the images have been sent, a message will notify the user of the upload completion.

The conversion step of the project upload performs a variety of operations to turn the 320x256 bitmap into a 245,760 byte long array. This contains the hexadecimal values which represent the intensity of color for each LED in the pixel's RGB matrix. The transmission operation starts by telling the Logic board to stop multiplexing images across its screen (to allow it to focus on receiving the new data) and clear the parallel RAM chip (about to be filled with the image data). The app separates the long byte array of data into smaller chunks that it sends across a TCP connection to the ESP8266 WiFi module on a new thread. This thread sends a chunk of the image data to the ESP module, and after receiving confirmation from the microcontroller that the message arrived safely, it updates the progress bar on the main thread via a Handler. After one full image is sent, commands are sent to specify which non-volatile flash chip to store the image into. After the full project is sent, a message containing the number of images the project contains and the time duration between images is sent over WiFi again to be stored into non-volatile program flash and used during the state machine operations.

The Android App was developed using Google's Android Studio, meaning the code behind the application is mostly Java, xml files, and some Gradle files to define the package and build configurations. With simplicity in mind, a drawer navigation system was used with menu items each navigating to a different fragment within the main activity. A few buttons also provide navigation shortcuts that help the user follow the flow of operation. A separate activity was built to handle image selection from the image gallery and to handle the image editing tools at the user's disposal. Due to the lightweight nature of the data being stored outside the images, SharedPreferences were used to store much of the project and device information that needed to be persisted. A couple of screenshots of the app in action are shown in Fig. 5.

Figure 5: Several snapshots of the custom app during normal operation.

3. Safety

The electronic display system consumes significant power due to the large number of LEDs. The power supply can provide up to 50 A. Thus, users must be cautious of the high currents supplied to the RGB LED panels. In our current prototype, connections are accessible on the back of the panel, so, there is the risk of immediate contact with such connections. We plan to enclose the entire panel inside a wood enclosure with a back lid such that no electrical contacts can be directly accessed (such an enclosure is not part of the current design and is not described in this paper or in the documentation).

As already mentioned earlier, the MCU on the controller board has many features to help with debugging. These features include a rugged USB debugging interface, which was implemented with a UART to USB bridge. This allows a user to connect to the microcontroller through a 115.2Kb serial connection from a software terminal such as Putty. This was a crucial debugging feature when developing the MCU firmware. The MCU's analog to digital converter was also used to be able to measure system level power supply voltages. The ADC can be initialized through the USB interface, and results can be printed through the serial interface. The ADC is only initialized when the user requests it; this allows for the CPU bandwidth to be maximized when the debug feature is not needed.

4. General testing

The following table lists the metrics or figures of merit that were tested and validated using the complete hardware prototype of the electronic display system as well as the fully operational Android App.

Table 1: Required product specifications vs. achieved performance.

Metric	Units	Target Specification	Test Result
Number of images (slides) to display	Slides	8	8
Color Resolution	Bits (colors)	12 (4096)	9 (512)
Connectivity	N/A	WiFi	WiFi connectivity works correctly
Refresh Rate	N/A	Image displayed correctly without noticeable flickering	30Hz refresh rate with no perceived flickering

Number of Images to Display

The objective of this experiment was to verify that eight images can be saved to the flash chips located on the logic board and that all images can be displayed correctly in order and repeatedly. The testing strategy was to cycle through all eight flash chips, write and then read the image data from, and drive the LED panels to display each of the images. First, all eight flash chips were written to with eight different images. Then, each image was read from the flash chips and displayed onto the panels. The images displayed on the panels were verified whether they were the desired image. This was repeated five times and all images were changed each repetition to verify the reliability of the entire process of writing and reading images into the flash chips. During this process, we monitored the error (as percentage of pixel data written or read incorrectly) during both writing to and reading from the flash chips. This test was successful with an outcome of 0% error.

Color Resolution

The objective for this experiment was to identify the optimal number of bits for the color representation used to render images. The constraints of this experiments included the amount of memory available inside the microcontroller to implement the buffer (stores the data of the image currently being displayed) as well as the time it takes to upload all eight images from the App to the board's flash chips. The desired color representation was 12-bit color code, which would be capable of generating 4096 different colors per pixel. However, due to how the 5x4 panels (64x64 pixels each) are driven by the only 8 PWM frames (giving 3 bits per color channel) with information about pixels forming an image, we arrived at the compromise of using only a 9-bit color code, which is still capable of generating 512 different colors per pixel. This color code ensures that images have reasonable color resolution to make them appear in natural colors onto the display. While 12-bit color code would obviously be preferable, the 9-bit color code is sufficient to be able to display rich colors, yet to restrict the amount of memory required, achieve better performance (as refresh rate), and to naturally match the PWM control of the 64x64 LED panels.

Connectivity

The objective of this experiment was to verify the Android App and the logic board can communicate successfully via WiFi. The ESP8266 module, mounted on the logic board, was used for this connection. The test plan for this experiment involved sending 20 images of data from the Android App to the logic board and verifying that no data was lost during the transfer. Testing to be sure that no data was lost in the transfer involved displaying the image onto the screen and verifying that the image appeared as intended. This test was successful with an outcome of 0% error.

Refresh Rate

The objective of this experiment was to verify that the microcontroller was fast enough to properly display the images without any flicker. This was performed by using an oscilloscope to measure the frequency of the clock and latch signals and seeing that the images displayed on the panels did not flicker. This procedure was repeated five times on five different images to verify that a range of different detailed images can be shown without flickering. This test was successful with an achieved refresh rate of 30Hz.

5. Use case

The LED electronic display system was built from scratch and then extensively tested. Several photographs of the final prototype are shown in Figs. 6 and 7. The built system met all the desired design requirements. The logic board was able to store successfully eight images on the non-volatile memory transmitted via WiFi from the Android App. The refresh rate of each image during normal operation was 60 fps, which is enough to be free of visible flickering. Only 9-bit color resolution was able to be supported because there are only 8 PWM frames able to display shading (i.e., color channel) of each color. This was found to be acceptable as images are still naturally displayed onto the panels. The main limitations of the proposed design include: 1) the fact that soldering the electronic components on the power and controller boards is time consuming and requires more than just beginner-level of skill, but still easily doable as the logic board is just a 2-layer PCB board, 2) the overall cost is relatively high at \$804, but still affordable for a project of this scale, and 3) the display cannot render videos.

Figure 6: Photographs of the back side of the final electronic display prototype.

Figure 7: Photographs of the front side during normal operation.

6. Reuse potential and adaptability

The electronic display is completely reproducible as all the hardware components are readily available from various vendors and the complete design files are made publicly available. Designed to be cost-effective yet extendable, we hope this design will be adopted and improved upon in other practical and educational settings. At the minimum, the display can be replicated as presented and used out of the box. We envision two main ways to extend it. First, the display size could be increased by adding more 64x64 RGB LED panels to the rows and columns. This will potentially require power supplies capable of providing higher currents. Also, the firmware residing on the MCU of the controller board will need to be changed to account for display sizes different than the current 320x256 LEDs. Second, the system could be extended to add the capability of playing video. Such an extension will be more challenging. It could be done by keeping the display continuously connected to the App (or the Internet), which would stream desired videos. Another idea is to modify the controller board in order to include larger capacity flash memory or a micro SD card, which could store large video files. While doable, such changes are major changes to the current design.

We believe that our design has more unique value in educational contexts, because in business contexts, companies can purchase such display boards from established digital display power-houses such as Daktronics. In educational contexts, one can use the proposed design as an open-source flexible “workbench” for various studies in courses on embedded systems, digital signal processing and design, or wireless communications. For instance, in a typical embedded systems course, examples of such studies may include: 1) Investigation of the power consumption and of firmware techniques to reduce it, 2) Laboratory experiments where students are required to implement essential modules of the firmware. Examples of such modules can cover: the WiFi communication between the design and the mobile app, the module responsible for debugging features via the UART peripheral of the

microcontroller, etc. The existing design could serve as a platform to verify software development tasks covering microcontroller firmware and mobile app development as well.

7. Availability of materials and methods

The bill of materials for the electronic display system is listed in Table 2. Detailed specifics on each of the electronic components (i.e., resistors, capacitors, etc.) are not included here for brevity (the list is too long); however, they are available in the online repository.

Table 2: Bill of materials for the electronic display design.

Designator	Component	Num.	Cost per unit US\$	Total cost (US\$)	Source of materials	Material type
LED display	64x64 RGB LED Panels	20	\$18.70	\$374	Ali Express	Printed circuit board (PCB), Semiconducto r
Power supply 1	12V, 50A Power Supply	1	\$50	\$50	Amazon	NA
Power supply 2	5V, 50A Power Supply	1	\$20	\$20	Amazon	NA
Controller board	Logic Board PCB	1	\$100	\$100	JLC PCB	PCB
Electronic components	Logic Board Components	N/A	\$150	\$150	Digi-Key	Semiconducto r
Frame	Custom Frame	1	\$50	\$50	McMaster- Carr	NA
Connections	Cables, Wires, Misc.	N/A	\$60	\$60	Digi-Key	NA

8. Ease of build

Complete design files, both hardware and software, and documentation are made publicly available in the online repository. In the simplest scenario, one can simply duplicate the proposed electronic display system. The most challenging aspect in this process would be the soldering of all the SMD

components on the logic board, because it requires prior experience and SMD soldering skills as well as the availability of a reflow oven. Otherwise, everything is rather simple and straightforward to assemble and to program. Extensive efforts were put into implementing a comprehensive help and debugging capability for the logic board. To use this capability, one would only need to connect a serial terminal (such as Putty) to the USB port of the logic board, then, follow the prompts. More detailed information on this debug ability is provided in the documentation for the firmware.

Perhaps the most notable aspect regarding the technical implementation of this design is that it provided an excellent educational value for the duration of two semesters. More specifically, 1) it required us to develop our own method of “packetizing” the pixel data of all eight images transmitted from the mobile app to the display board via WiFi and 2) it provided a context where hardware and firmware designers needed to team-up with mobile app software developers and create a product that works reliably and repeatedly. This required good communication skills and collaboration to develop common testing strategies.

9. Operating software and peripherals

Special software is only needed at the time of programming of the microcontroller on the logic board. Specifically, the Microchip’s MPLAB is required to be able to compile and program the microcontroller. For programming, a programmer such as the PicKit 3 is necessary. The Android App is available on Google Play, so, no additional software is needed. However, if one would like to make changes to the source code of the App, Google’s Android Studio is required; it is available for download from Google. None of the above two software tools (MPLAB and Android Studio) need more disk space and memory than typical programs.

10. Dependencies

The entire design is made open source and publicly available. It does not have any dependencies and all code (firmware, App) was written from scratch. However, it does require coding frameworks such as Google’s Android Studio that are available for free.

11. Hardware documentation and files location:

The entire design is made open source and publicly available. The following repository contains both hardware and software design files. Additional information, including a project poster and a short video presentation is available at (project “RGB LED Electronic Display”):

<http://dejazzzer.com/hardware.html> ([archived version](#))

Name of repository: https://github.com/drewsum/Electronic_Display

Persistent identifier: NA

Licence: Open Source License for Hardware and Free and Open Source License for Software

Publisher: Andrew Maatman

Date published: 1/05/19

12. Conclusions

We presented a completely open source design of a 320x256 RGB LED electronic display board created from a 5x4 array of 64x64 pixels RGB off-the-shelf panels. The electronic display is intended to be used as a display board of information of current and future events in schools or office buildings. The information is presented under the form of eight images or slides that are displayed periodically by the board. The images can be uploaded to the board system via WiFi from an Android App that was custom developed for this purpose. A complete prototype was built and tested successfully. Despite its relatively high overall cost, the proposed electronic display system is extremely easy to use. The complete design files, both hardware and software, are made publicly available. As such, we hope that this design can serve as a starting point for other researchers and educators who are interested in developing more complex RGB LED electronic displays.

13. Future Work

As discussed in the Section “Reuse potential and adaptability” above, the proposed electronic display could be extended in several ways. Currently, we are planning to extend the design in order to be able to play video as well.

Acknowledgements

The authors would like to thank the generous support from the Dept. of Electrical and Computer Engineering in the OPUS College of Engineering, Marquette University.

Funding statement

This work was not funded through research grants.

Competing interests

The authors declare that they have no competing interests.

References

1. Character LCDs, Lady Ada, 2019, [Online]. Available: <https://cdn-learn.adafruit.com/downloads/pdf/character-lcds.pdf>

2. A Wirelessly Controlled, Arduino-Powered Message Board, 2019, [Online]. Available: <https://www.instructables.com/id/A-Wirelessly-Controlled-Arduino-Powered-Message-B/>
3. Open Frame LCD Monitors, EarthLCD, 2019, [Online]. Available: <https://earthlcd.com/product-category/open-frame-lcd-monitors/>
4. Digital Billboard Products, Daktronics, 2019, [Online]. Available: <https://www.daktronics.com/en-us/products/digital-billboards>
5. Live Spotify Album Art Display, SparkFun, 2019, [Online]. Available: <https://learn.sparkfun.com/tutorials/live-spotify-album-art-display>
6. PIXEL LED Art Frame, 2019, [Online]. Available: <http://creativeartsandtechnology.com/product/pixel-led-art-frame/>
7. Use the ezPixel FPGA Board to Control Giant Arrays of Smart LEDs, IEEE Spectrum, 19 Dec. 2017, [Online]. Available: <https://spectrum.ieee.org/geek-life/hands-on/use-the-ezpixel-fpga-board-to-control-giant-arrays-of-smart-leds>
8. LTpowerCAD and LTpowerPlanner, Analog Devices, 2019, [Online]. Available: <https://www.analog.com/en/design-center/ltpowercad.html>
9. LTspice, Analog Devices, 2019, [Online]. Available: <https://www.analog.com/en/design-center/design-tools-and-calculators/ltspice-simulator.html>
10. J. Hasan, D.H. Nguyen, and S.S. Ang, "A RGB-driver for LED display panels," IEEE Applied Power Electronics Conference and Exposition (APEC), 2010.
11. M.G.V. Bautista, W.-R. Liou, and M.-L. Yeh, "Dimmable multi-channel RGB LED driver," IEEE ECCE Asia, 2013.
12. X. Lv, K.H. Loo, Y.M. Lai, and C.K. Tse, "Energy-saving driver design for full-color large-area LED display panel systems," IEEE Trans. on Industrial Electronics, vol. 61, no. 9, 2014.
13. M. Andrei, S. Moldovanu, and N. Nistor, "Nonlinear transformations applied to image processing on the RGB LED display," Int. Symposium for Design and Technology in Electronic Packaging (SIITME), 2018.
14. C. Peng, X. Li, P. Zhang, L. Xiong, and X. Liu, "RGB high brightness LED modules for projection display application," Journal of Display Technology, vol. 7, no. 8, 2011.