MU-MatriX

User's Guide

Caroline Gilger, Logan Wedel, Drew Maatman, Kevin Etta, Tuoxuan Ren

Using the Android App:

2 BIG things to do before clicking "upload project to device"

1. Connect to the WiFi network "AI-THINKER_5D90B5"
    a. All the ESP modules use the same AI-THINKER_xxxxxx naming convention out of box so there may be others powered on if students are working on projects nearby
    b. The display board must be turned on and given a couple seconds for this network to become available.
2. Make sure you have mobile data on your device turned off.
    a. For some reason the app will try to send its packets through here to reach the internet instead of talking to the private WiFi network its connected to, so mobile data must be turned off.

Where the app is available

Install from the Google Play Store. Name: MU-MatriX

What to do to get started after install

- Create new projects containing the pictures you want and the time delay between image switching.
- Use "Device Control" to remotely adjust brightness or turn the screen on/off.
- Use "Upload Project" once you have completed building a project to select both that project and the device and upload them to the display board.

What is "Default Display Board"?

By default, the information for the display we built should be preloaded as "Default Display Board". In case this gets deleted, the information set in the embedded firmware to configure this connection is:

- Name it was ever you want
- 192.168.4.1
- Port 333

So create a new device with these characteristics.

Why are images changing background color for aspect ratio changes?

The code that fixes the aspect ratio applies a black background fill on the image for the empty gaps along the top/bottom or left/right sides. Because of this, any image that doesn't have a background to it (some of the logo png's are like this where the only data contained in the file is the colored part of the image) will turn black when the "fixed aspect ratio" is applied. If you want a white background on these images instead either:

- Use the stretch to fit option (not ideal in some cases)
- Recreate the image with the white background on it, such as screenshotting the image pulled up on a computer or something of that nature.

Tips to ensure project upload goes smoothly:

- Make sure you do the “2 big things before clicking upload project to device”
- After starting the upload, don’t go do other things on your phone or close the app
- Just let it sit and you can watch the progress bar go to work. Your phone screen should not turn off during this action.
- Expect about 1 minute 30 seconds per image (had to send them in small chunks)

What happens if you only have 3 images or 5 images to display (versus the maximum 8)?

This is fine. The microcontroller state machine switching between images will only switch between those contained in the project. Any old images stored on the device are erased from flash memory upon upload.

What to expect during normal operation:

During normal device operation, the LED display screen should cycle through images that users have programmed into the device through the android app and WiFi interface at a periodic rate. The LEDs in the center column of status LEDs on the blue logic control PCB situated on the back of the screen should all be lit while the screen is on. While the screen is cycling through displayed images, the SPI Flash Access LEDs located on the bottom right side of the blue logic board should illuminate on and off periodically, as the device accesses data stored in Flash memory. When programming new images through the Android App interface (outlined above), the screen should turn off and the WiFi TX/RX status LED should flicker (not blink) of the logic board. The user should see the progress bar on the upload screen in the android app slowly rising as images are being transferred over WiFi. **While images are being transferred over WiFi, do not exit the app.**

Restarting the device if issues occur during image programming:

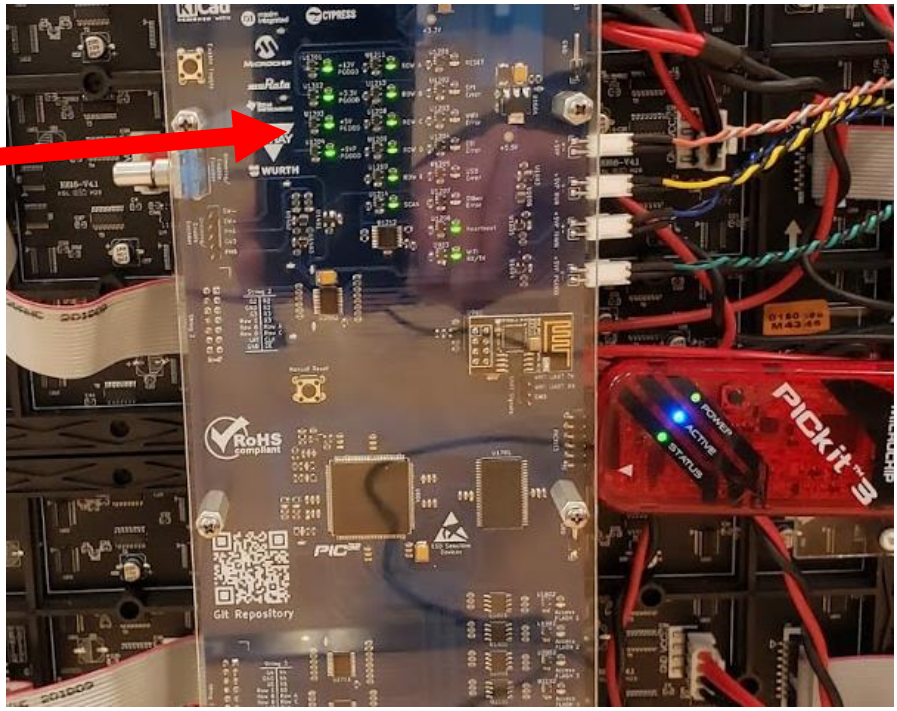
Many failsafe mechanisms were put into place to keep a WiFi connection open during image programming. With that being said, should the image transfer become “stuck”, the user could exit this state by unplugging the device from the wall and allowing it to power off, waiting around 15 seconds before plugging the device back into the wall. While the device is unplugged, the user should exit the Andoird App completely, and try the image transfer again once the device has been restarted. If WiFi transfers continuously get stuck, the user can see WiFi traffic through the USB debugging interface, outlined below.

Adjusting screen brightness and turning the screen on and off:

The screen brightness can be adjusted by turning the silver knob on the side of the screen. Turning the knob clockwise will raise the brightness, turning it counterclockwise will lower the brightness. Pressing the knob inward will turn the screen on and off and restart the slideshow of images programmed into the device.

Logic Board Status LEDs:

Debug Status LEDs



The logic board has status LEDs that give some indication of what's going on. The leftmost column of status LEDs should all be illuminated green while the LED display is operating, besides the lowest LED. These LEDs show the state of the Power-Good (PGOOD) signals. These LEDs indicate the state of the power supplies on the circuit board. If any LEDs are flickering or not illuminated, this points to an issue with a power supply.

The center column of debug LEDs should all be illuminated while images are being shown on the screen. These 6 LEDs show the states of key logic signals used to drive the display. If any of the LEDs are flickering or not illuminated while the screen is turned on, this is most likely due to a clocking issue with the microcontroller on the circuit board. This should not happen.

The rightmost column of status LEDs signal device faults and other status. The red LEDs in this column should never be illuminated during normal operation except on startup, when the device is plugged into the wall and power is applied. The error LEDs, such as USB error, SPI error, EBI error, and WiFi error, will illuminate if these respective subsystems in the device malfunction. Finally, the other error LED will illuminate if other more diverse errors occur. All of these error states can be read through the USB debugging interface, explained below.

Using the USB Debugging Features:

The logic board contains an extensive USB debugging interface. Connect a mini-USB cable to the USB connector on the top of the logic board, and connect the other end of the cable to a windows computer with a serial port interface software such as PuTTY or TeraTerm. FTDI USB drivers for windows may or may not be required, depending on your system. When connecting to the logic board through your computer's serial port, the following serial settings must be used:

- 115.2 kbps
- 8 data bits

- 1 stop bit
- No parity
- No flow control

These settings are also printed on the silkscreen on the circuit board near the USB connector.

The USB debugging interface allows the user to see what is going on under the hood of the device. The user can see if any device faults have been recorded, the flow of data, the status of many of the microcontroller peripherals, and also set the screen to solid colors. To be able to see which commands are available, once a serial connection to the logic board has been established, **press the enter key twice and type 'Help'.** Data printed on the screen may appear scrambled while the device is drawing images on the screen and moving data during normal operation. To stop the microcontroller from moving data and drawing images on the screen, type 'Disable State Machine' into the serial terminal and press enter. The screen should be turned off, and data flow should be stopped. Now the user has full control over the logic board and can set the screen to solid colors, copy data from flash chips into RAM, etc. To turn the screen on and off, type 'Enable Muxing' or 'Disable Muxing'. To restart the state machine to normal operation, type 'Reset' (which resets the microcontroller) or type 'Enable State Machine'.

Other things you should know about the Logic Board (some already mentioned):

- There is a rotary encoder located on the side of the logic board. Turning it allows the user to adjust screen brightness. Pressing it inward turns the screen on and off.
- The logic board cannot be powered by both the auxiliary input voltage jack (+12VIN AUX) and the input voltage shanks at the same time
- The microcontroller on the logic board was programmed using a PIC KIT 3 programmer from Microchip. PIC KIT 4 may work as well, but has not been tested.
- The project Git repository (including hardware design files and app and embedded software) is located at this URL at the time of writing: https://github.com/drewsum/Electronic_Display. Contact Drew Maatman at drewmaatman@gmail.com should any issues arise accessing the project Git repository.
- The microcontroller on the logic board is programmed with firmware located in the project repository at /sfw/embedded_firmware/U601_firmware/
- The microcontroller can be reset with the reset pushbutton (Manual Reset)
- The logic board will shine a red reset LED when the MCU is in reset or is being programmed
- The microcontroller firmware was complied with Microchip's XC32 compiler, version 2.10, using the MPLAB X IDE, version 5.15
- **The microcontroller firmware was compiled at optimization level two. The command line argument -O2 must be added to the XC32 command line driver arguments section of the project settings. Failure to do so will result in screen flickering. Optimization level two may or may not require a Microchip XC32 Pro compiler license. We purchased the Pro license for $30 for a one month subscription when building the project.**
- The logic board has an extensive USB debugging interface
    - Connect a Micro USB cable to the USB jack on the top of the logic board
    - The serial port settings are printed in silkscreen on the logic board near the USB jack
        - 115.2 kbps

        - 8 data bits
        - 1 stop bit
        - No parity
        - No flow control
    - Use serial port software such as Tera Term or PuTTY to communicate with the logic board.
    - **When opening a serial connection, press the enter key twice. This enables USB communication. Skipping this will result in no serial output.**
    - After pressing enter twice, type “Help” and press enter. This will show all available serial commands.
    - To control what’s on the screen, type “Disable State Machine”. The user can now explicitly control what’s displayed on the screen. “Enable Muxing” and “Disable Muxing” enable and disable the screen. “Enable State Machine” resumes the normal system behavior
- The logic board has error state reporting that can be shown through the red error LEDs on the board and through the USB debugging interface
- If a CPU exception occurs, all red LEDs on the logic board will illuminate. This should *never* happen.
- The microcontroller watchdog timer and deadman timer should reset the microcontroller if a firmware fault occurs
- The logic board has green PGOOD LEDs that illuminate when the on board power supplies are within +/-10% of their programmed output voltages. This can show if the power supplies are operating correctly.
- Images that the screen displays can only be programmed through the Android App, but there are two hardcoded images stored in microcontroller flash. These can be displayed on the screen through the USB debugging interface. Serial commands for displaying these images can be shown through the ‘Help’ serial command
- The screen can be set to solid colors through the USB debugging interface
- The logic board has input undervoltage protection, input overvoltage protection, and input reverse voltage polarity protection. This means that **the logic board *must* be supplied with +12V DC, and +12V DC only. Using a higher or lower source voltage will not work, but should not damage the logic board.**
- When programming images from the android app, WiFi traffic can be monitored using the USB debugging interface