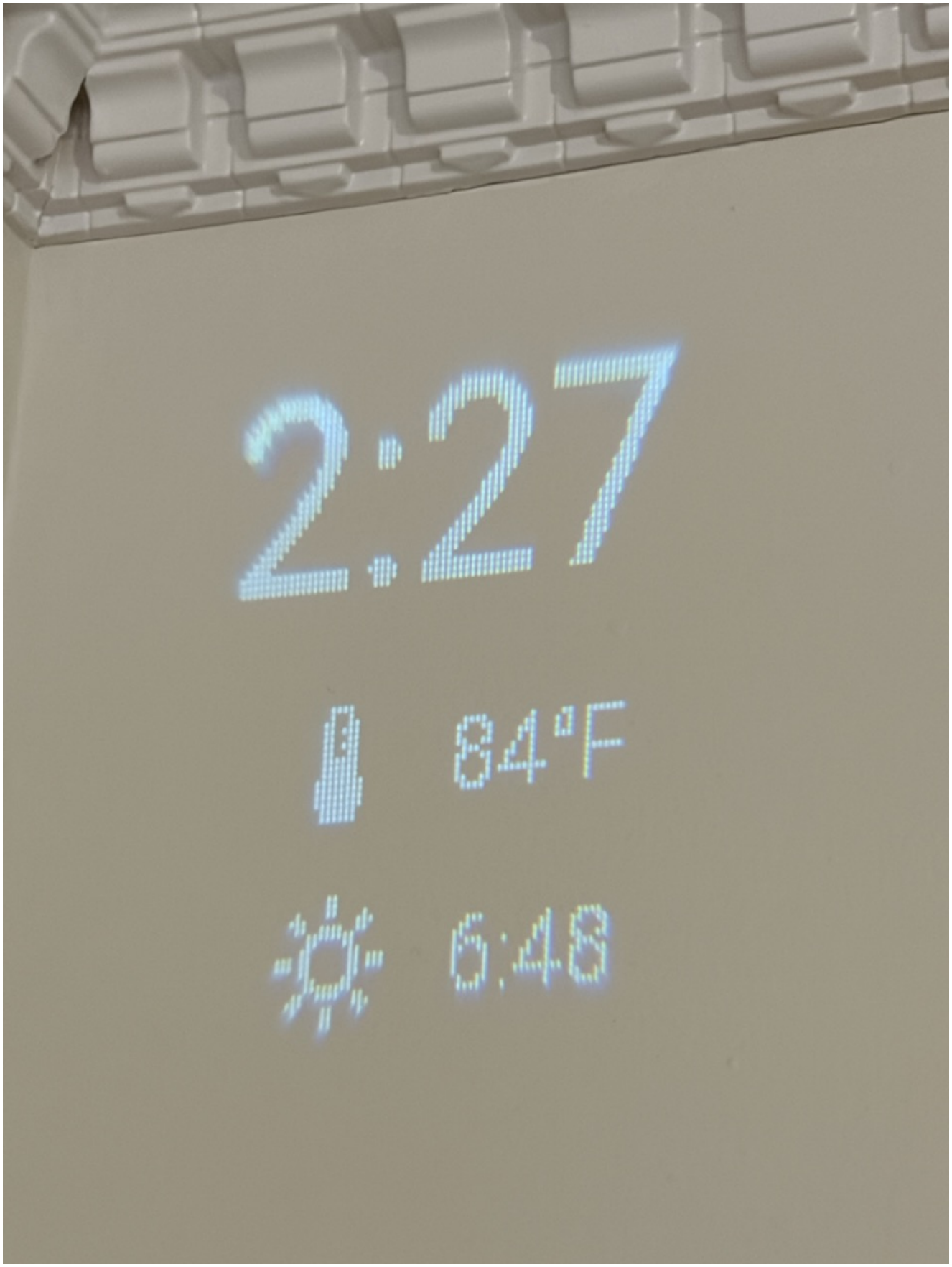
ESP32 Projection Clock (ESPHome + Home Assistant)

I built a **day/night projection clock** built on an **ESP32** with **ESPHome.** The projector is lit by a high-power LED driven by a constant-current board controlled with a PWM signal from the ESP32. The light is directed through a collimator, a color TFT display, and a focusing lens

I can project Time, temperature, sunrise data onto a ~12in image onto the ceiling ~70 in away. LED brightness is exposed and controlled through Home Assistant. The entire apparatus runs off of the ESP32 usb-c port at just under a 1A at full brightness.

This repository documents the design, 3d printing, and YAML code used to build the clock.


2:27
84°F
6:48

Objectives:
- Network-synchronized time, outdoor temperature, and sunrise time display
- Smooth, flicker-free brightness control via Home Assistant
- Single 5VDC USB cable power supply

This combination of parts is unique because it is powered by a single USB cable at 5VDC. A brighter LED would need a buck boost converter and more power.

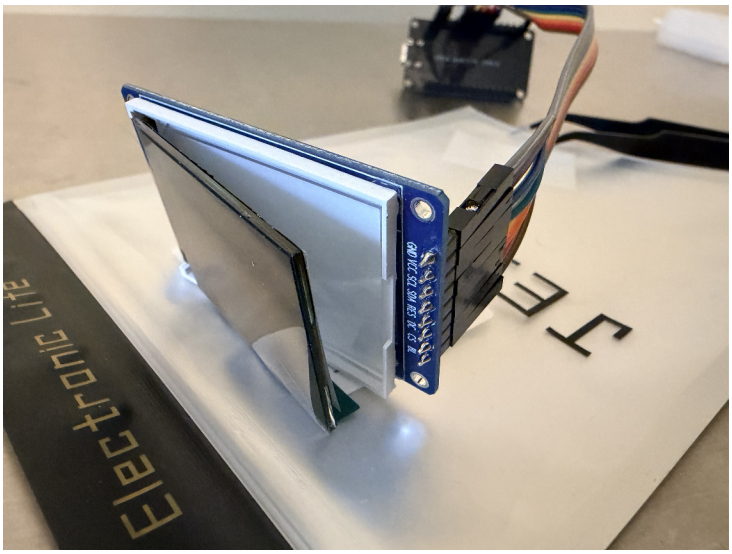
Parts List:

Component	**Description**	**Source**	**Price (as of 12-25)**
ESP32	Dual core, Wifi, Bluetooth	https://www.amazon.com/dp/B0D8T53CQ5	$20 for (3)
Screen	1.8 Inch TFT LCD, 128x160 ST7735	https://www.amazon.com/dp/B0D31BGJWF	$10
High Powered LED	3W White 3535 SMD, 700mA, 3.5V forward voltage, 6500k color temp	https://www.amazon.com/dp/B0CL4J79VS	$10 for (5)
LED Driver	constant current (LD1500SB)	https://www.amazon.com/dp/B0DCZFPHLY?th=1	$9 for (4)
Collimator Lens	53mm diameter	https://www.amazon.com/dp/B06XKV9PWY	$9
LED Heatsink	1.57” square	https://www.amazon.com/dp/B07SMX1DB7	$8.50 for (4)
Plano convex lens	50 mm dia, 150mm focal length	https://www.amazon.com/dp/B0714B1L2L	$9
Wires	22AWG or better. Dupont connectors work for a while. JST better		~$5
Grub screws	2x M4x6. Other sizes can work too.		~$1
3D printing filament	~500g of PLA or PETG		~$10

Making the TFT Screen transpaent
All TFT screens are transparent, but most are sold with a reflective backing and edge-lit backlight. In order to project light through the screen, the backing and backlight must be removed.

Here is the video where I learned how to do this: https://youtu.be/c2OQyfeLn_w?t=480

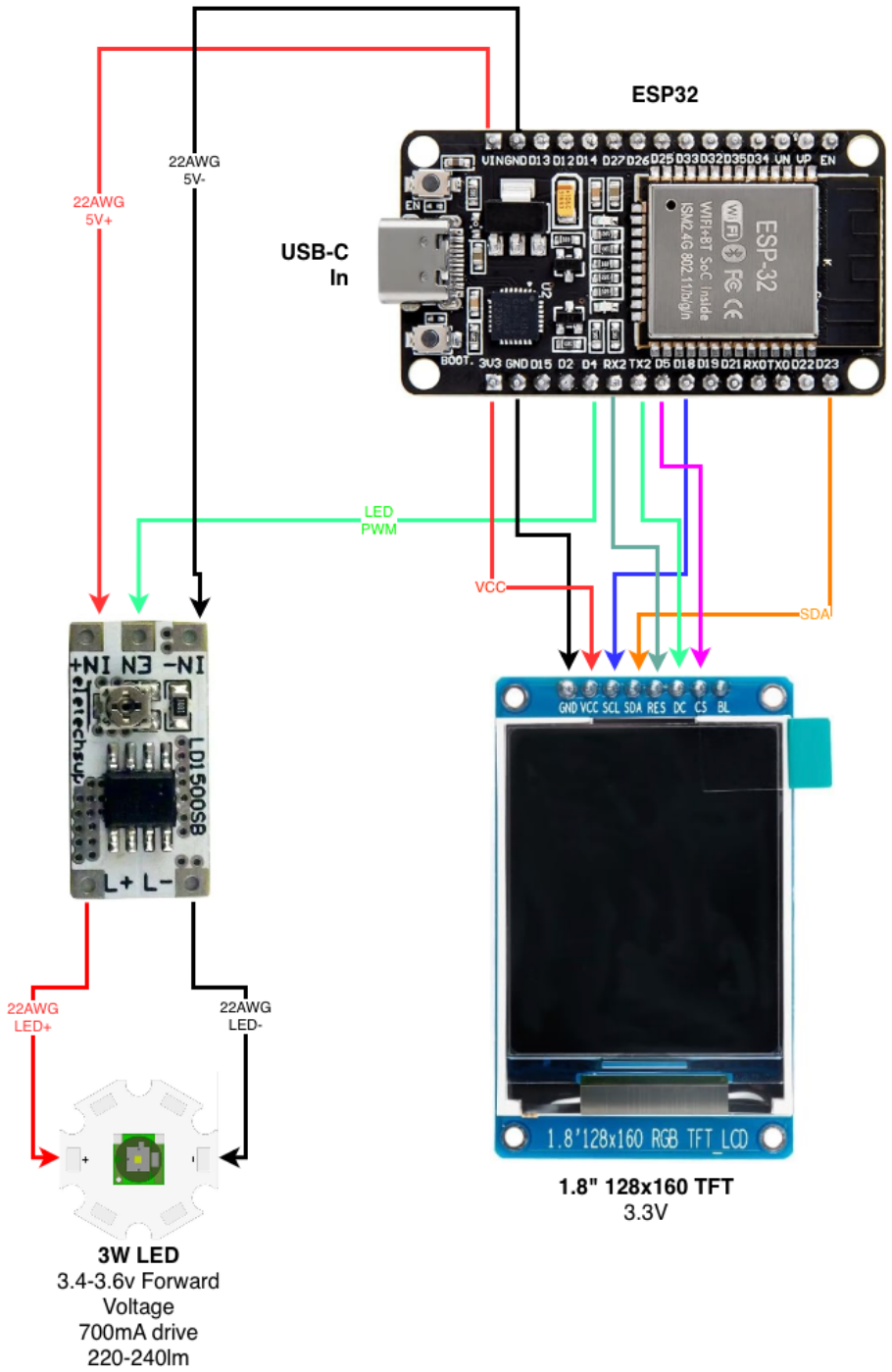
1. Carefully peel back the screen from the board



2. Remove the various backing films with tweezers. Cut the backlight cable and remove the backlight frame.

3. Once separated, light will pass through white sections of the screen.

Wiring Diagram

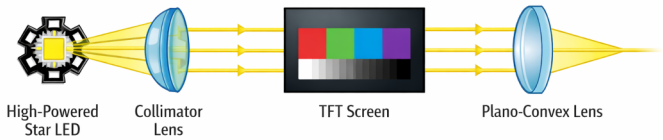


TFT Wiring Pin detail

TFT Pin	**ESP32 Pin**
GND	**GND**
VCC	**3.3V**
SCL/CLK/SCK - SPI Clock	**GPIO18**
SDA / MOSI / SPI Data	**GPIO23**
RES/RST, Reset	**GPIO16**
DC/A0, Data Command	**GPIO17**
CS, Chip select	**GPIO5**
BL/LED, backlight	**3.3V** or **GPIO4** (PWM backlight)

Projecting and focusing the image

I wanted a ~12 in projection onto the ceiling ~70 in away. LED lights like the star LED in the figure below produce diffuse omnidirectional light. The collimator lens focuses this light into a large even spot and the final plano-convex lens projects and focuses a ~12 in image on the ceiling about 70 in away.



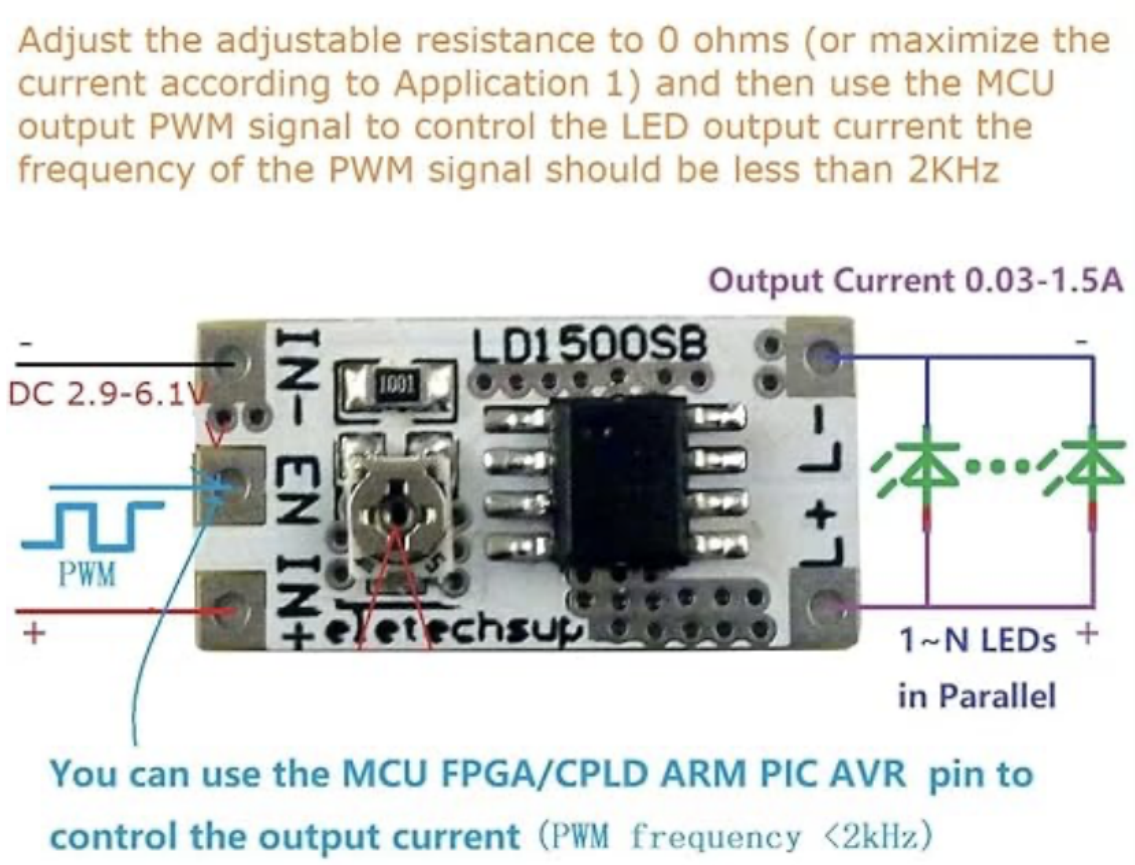
Some helpful notes:
1. When viewed from the front, the LCD screen should look upside down and backwards so the projected image will look correct on the wall/ceiling.
2. The Lens tube has some adjustment for closer of further projection distances.

LB1500SB tuning

Note this step can only be completed when the ESP32 is flashed and running

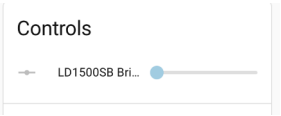
The LB1550SB constant current LED driver board has one variable resistor that will override the PWM signal sent by the ESP32. In order to sent the right amount of current to the LED, the variable resistor and the PWM signal must be tuned.

1. Turn the variable resistor to minimum resistance or maximum current. See image below



2. Hook up a multimeter in line with the L- pin on the board. Change the leads to measure current and set to meter to read Amps.

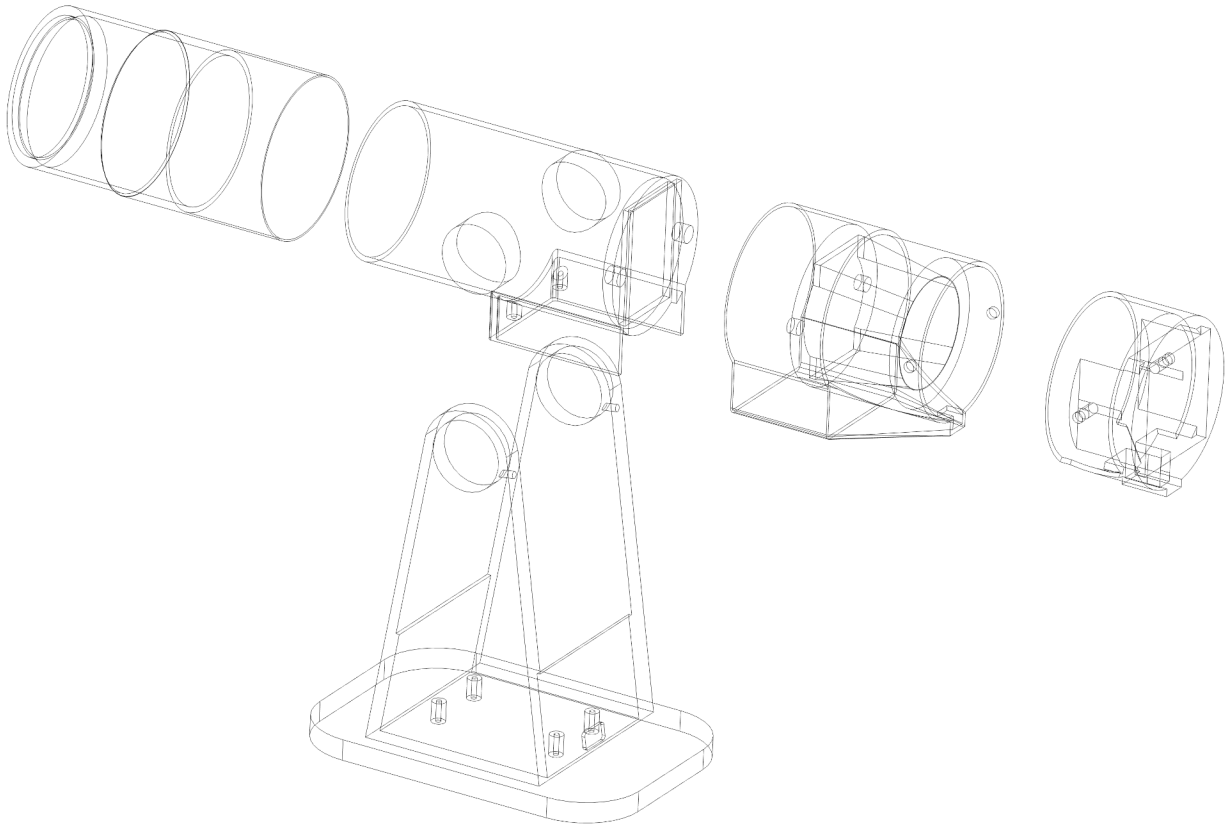
3. Gradually change the LED brightness in Home Assistant until you reach the recommended current for your LED. In my case, that was 700mA at ~80% brightness.



Thanks to Makerguides for this link with much more detail:
https://www.makerguides.com/drive-power-leds-with-ld1500sb-esp32/

Mechanical Design

The clock is composed of three slip together tubes, an end cap that houses the heatsink, and pivoting base. The whole assembly looks like a little telescope.

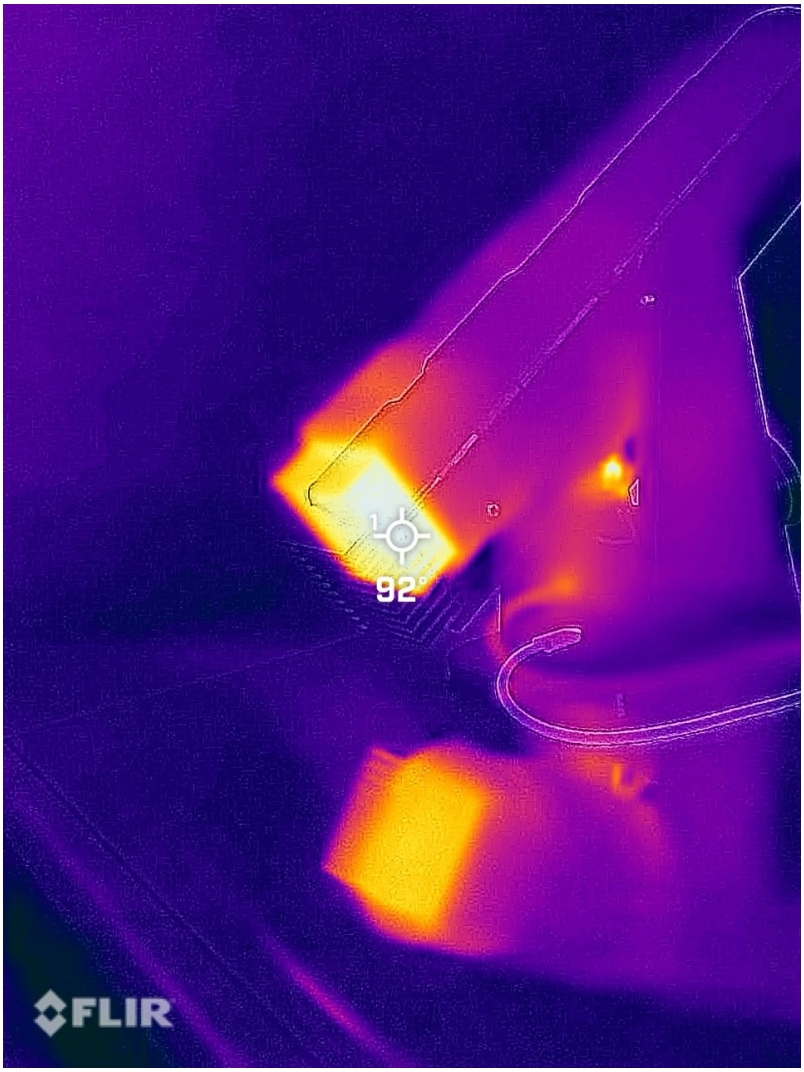


I used black PLA+ on an Ender 3 v3 SE. Supports are required for some parts and printing orientation should be obvious. You will need 5 total parts:

1. Lens tube
2. Screen tube
3. Base
4. Light tube
5. Heatsink endcap

Some boards are partially exposed for convenience, but there is little possibility of a short circuit. The Fusion 360, stp file, and 3mf files are provided for all the printable components.

Thermals are OK with the clock at full brightness


1
92°
FLIR

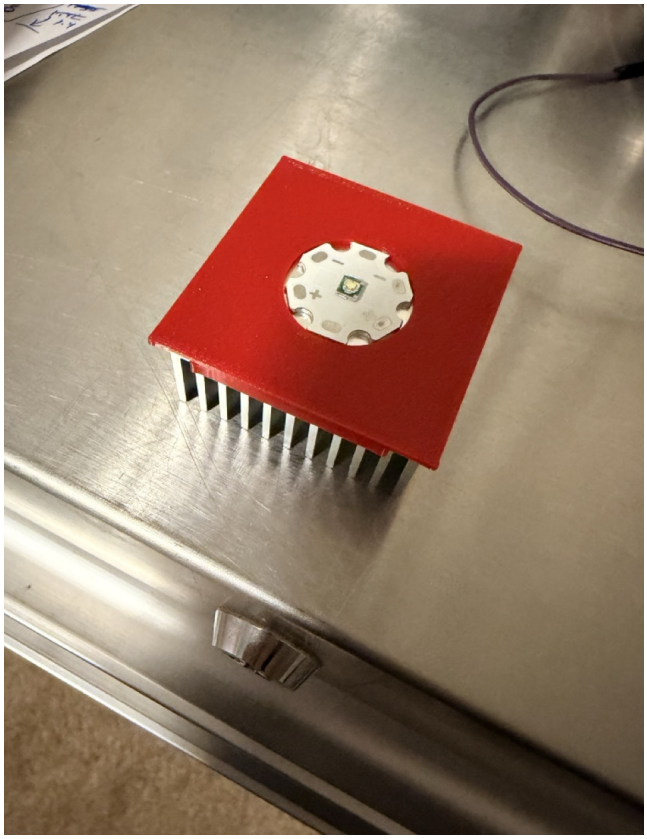
Assembly tips

1. Print the screen tube first. After making the screen transparent, carefully friction fit it into the opening with the cable hanging out the bottom. Mount the screen board with two little screws (I think I used M1.6).

2. Print the lens tube next. The lens fits in its opening via friction. I was going to design an end cap, but it seems secure enough as is. The lens tube slides onto the screen tube and provides a limited focus range.

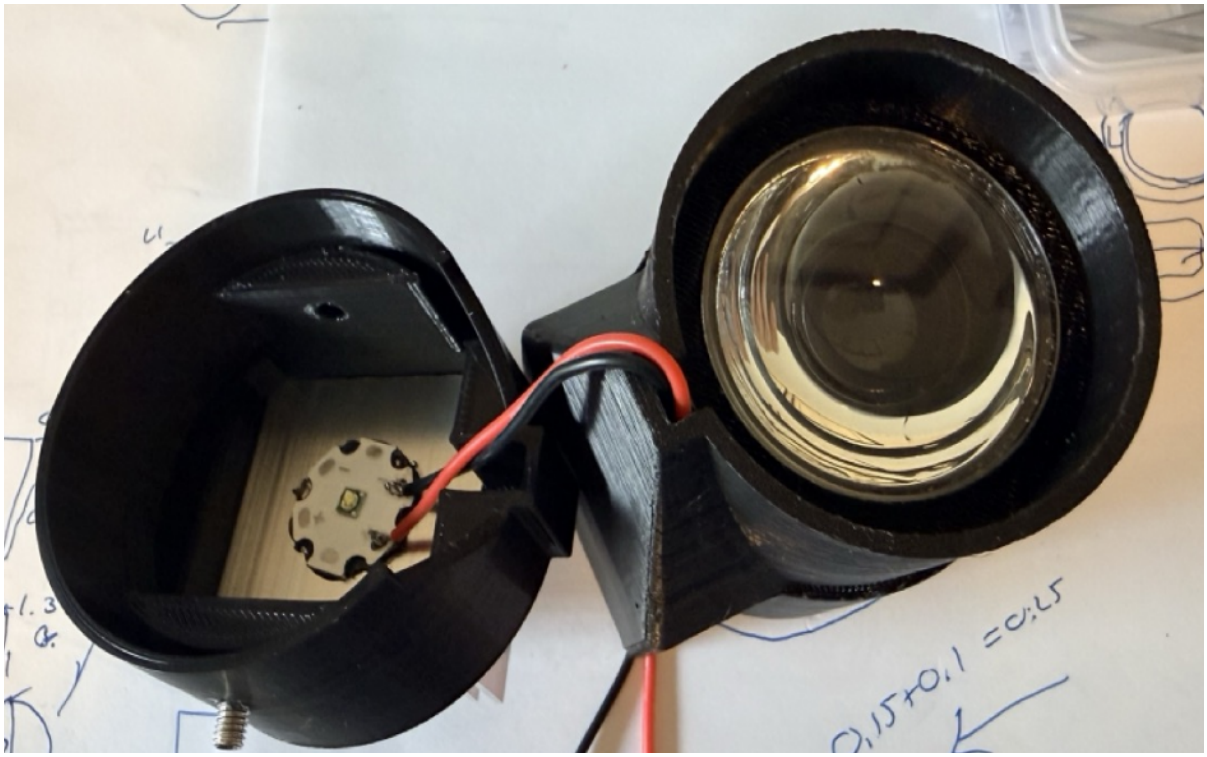
3. Print the base next and mount the ESP32 in place with slightly larger M2 screws. The M1.6s can slip through the board mounting holes. Carefully slip the screen and lens tube in position. Use grub screws in the proved holes to lock the position.

4. Mount the star LED onto the heatsink base. I used JB weld, but thermal epoxy would be better. Getting the LED in the center of the heatsink is very important. A 3D printed centering jig is included in the files. Before mounting solder 22AWG wires to the + and – pads with the wire facing down and parallel to the heat sink fins (see picture)



5. Print the light tube and heatsink cap. Slide the heatsink into the end cap. It should hold fine via friction. Route the wires through the chase.

6. Place the collimator in the light tube dome side down. Place the reflector on top with the wire recces facing down. Place the heatsink end cap assembly on top and press until the grub screws line up. Install two M4 grub screws.



7. Connect the screens board, ESP32, and LED driver board electrically. I attached a little raspberry pi heatsink to the driver board with a Kapton tape barrier and a heatshrink scarf. This may not be nesseccary.

8. Compile and Flash the YAML code onto ESP32

9. Enjoy your projection clock

ESPHome YAML code

Be sure to substitute your OTA, AP, API, and sensor names where indicated.

```
# ESPhome Projection Clock, 12-25, BG
esphome:
  name: projection-clock
  friendly_name: Projection Clock

esp32:
  board: esp32dev
  framework:
    type: esp-idf

# Enable logging
logger:

# Enable Home Assistant API
api:
  encryption:
    key: "" //Insert your api key here

ota:
  - platform: esphome
    password: "" //Insert your ota password here

wifi:
  ssid: !secret wifi_ssid
  password: !secret wifi_password

  # Enable fallback hotspot (captive portal) in case wifi connection fails
  ap:
    ssid: "Projection-Clock"
    password: "" //Insert your ap password here

captive_portal:
```

```
# Backlight control. 1200 Hz below the 2khZ limit for the LD1500SB. 500Hz resulted in
filckering
output:
  - platform: ledc
    pin: GPIO4
    id: dim_pwm
    frequency: 1200 Hz

number:
  - platform: template
    name: "LD1500SB Brightness"
    min_value: 0
    max_value: 1
    step: 0.01
    set_action:
      - output.set_level:
          id: dim_pwm
          level: !lambda |
            if (x <= 0.01f) return 0.0f;      // TRUE OFF
            else if (x < 0.05f) return 0.05f; // Minimum floor
            else return x;
# Outdoor temperature sensor and dawn time. Substitute your sensor name for
"sensor.temp2_temperature."
sensor:
  - platform: homeassistant
    id: tempout
    entity_id: sensor.temp2_temperature

text_sensor:
  - platform: homeassistant
    id: sensor_sun_next_dawn
    entity_id: sensor.sun_next_dawn

# Home Assistant time
time:
  - platform: homeassistant
    id: ha_time

# SPI bus for display
spi:
  clk_pin: 18
  mosi_pin: 23
  id: spi_bus

# Fonts. Download Roboto and Materials and Symbols font and install .ttf in
/config/esphome/fonts
font:
  - file: "gfonts://Roboto"
    id: font_big
```

```
    size: 38
  - file: "gfonts://Roboto"
    id: font_small
    size: 14

  - file: "fonts/MaterialSymbolsOutlined-VariableFont_FILL,GRAD,opsz,wght.ttf"
    id: font_icons
    size: 20
    glyphs:
      - "\U0000E81A"   # Sunny
      - "\U0000E1FF"   # Thermometer

# Display configuration. st7735 is depricated so a day may come when this must change
display:
  - platform: st7735
    id: tft
    model: "INITR_GREENTAB"
    cs_pin: 5
    dc_pin: 17
    reset_pin: 16
    device_width: 128
    device_height: 160
    col_start: 2
    row_start: 1
    rotation: 180
    invert_colors: false
    update_interval: 1s
    lambda: |-
      // --- TIME (centered, no leading zero) ---
      char timebuf[6] = {};
      id(ha_time).now().strftime(timebuf, sizeof(timebuf), "%I:%M");

      const char* timestr = (timebuf[0] == '0') ? timebuf + 1 : timebuf;

      // Center X = 64 on a 128px-wide portrait display
      it.printf(64, 32, id(font_big), TextAlign::CENTER, "%s", timestr);

      // --- SUNRISE (centered, icon + time) ---
      auto s = id(sensor_sun_next_dawn);
      if (s->has_state()) {
        std::string full = s->state;            // "2025-12-07T13:04:59+00:00"
        std::string t = full.substr(11, 5);     // HH:MM
        int hour = std::stoi(t.substr(0,2)) - 6; // UTC → local (example CST)
        if (hour < 0) hour += 24;
        t = std::to_string(hour) + ":" + t.substr(3,2);

        // Centered icon + text pair on line 3 at y = 120
        it.printf(48, 97, id(font_icons), TextAlign::CENTER, "\U0000E81A");   // sunrise icon
        it.printf(80, 97, id(font_small), TextAlign::CENTER, "%s", t.c_str());
```

```
} else {
  // fallback (no icon)
  it.printf(64, 97, id(font_small), TextAlign::CENTER, "No Sun");
}

// --- TEMPERATURE (centered, icon + value) ---
auto t2 = id(tempout);
if (t2->has_state()) {
  char tempbuf[16];
  snprintf(tempbuf, sizeof(tempbuf), "%.0f°F", t2->state);

  // Centered icon + text pair on line 2 at y = 80
  it.printf(48, 70, id(font_icons), TextAlign::CENTER, "\U0000E1FF");   // thermometer icon
  it.printf(80, 70, id(font_small), TextAlign::CENTER, "%s", tempbuf);
} else {
  // fallback (no icon)
  it.printf(64, 70, id(font_small), TextAlign::CENTER, "No Temp");
}
```

If I ever made another clock, I would (and you should):

1. Use a square TFT, like a 128x128
2. Use a larger heatsink that could accommodate both the LED and driver board
3. Add weight to the lens tube for counterbalance