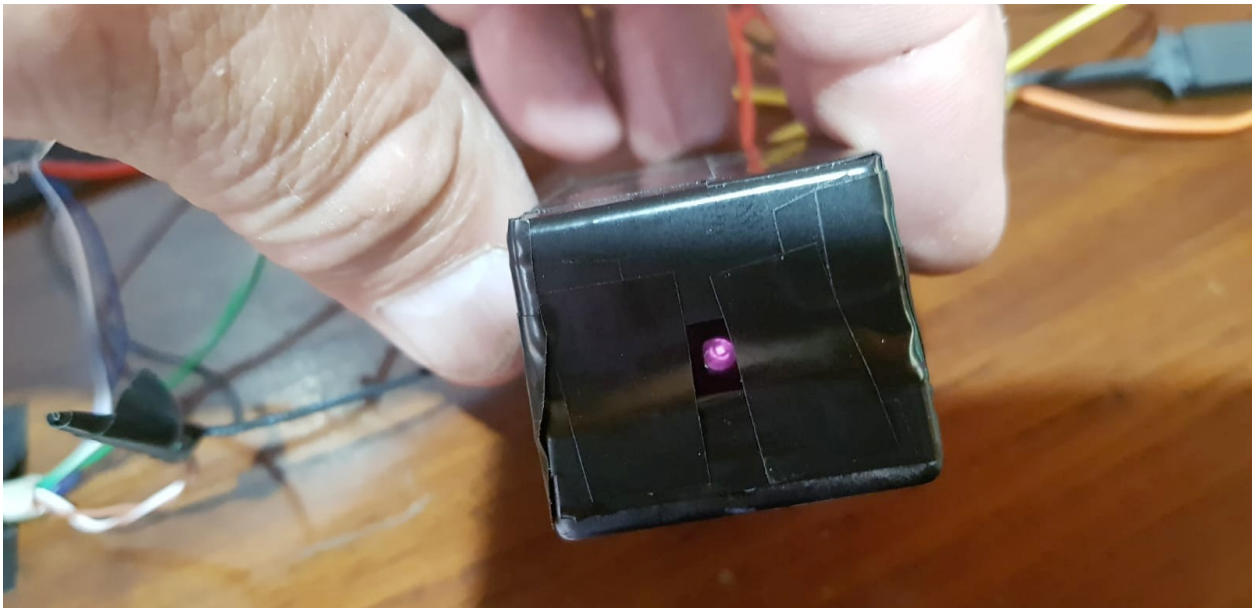
Garage Door Guard

The Guard will be an IFR beam across the door, up to 5 meters long and is not visible, use a Cell Camera to see it working.

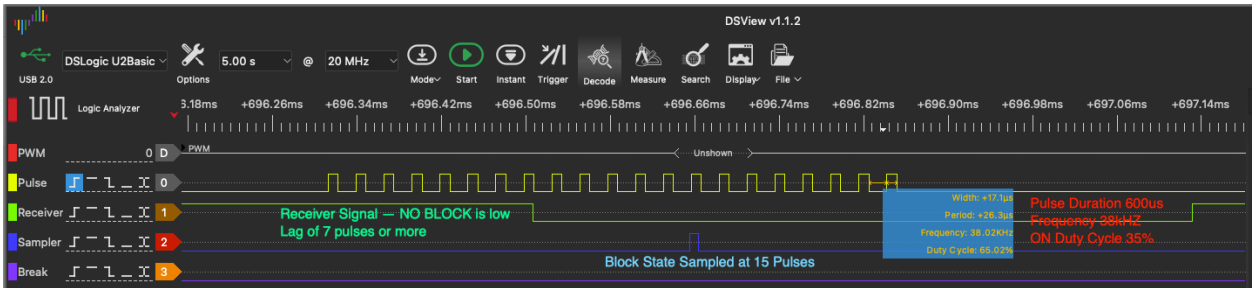


Cannot be a CONTINUOS beam because the Emitter (beam) will burn, so a Pulsing Beam with several parameters is defined.

Below are the Basic signals involved as seen by a Logic Analyzer

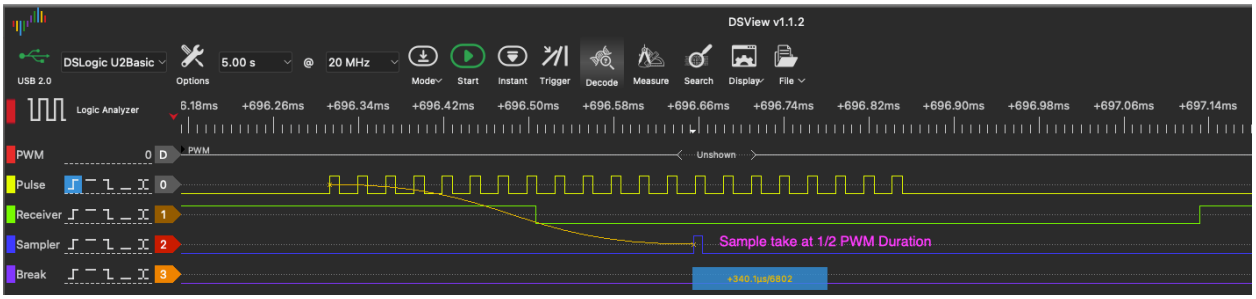
- The frequency depends on the RECEIVER sensor, in this case a VISHAY TSOP4308 which requires it to be at 38kHZ. The Pulse itself (yellow line) defined at 38kHZ and that will be created via a PWM at X duty cycle (theConf.ciclo) for Y microseconds duration ON (theConf.timers[TPWON]) and Z microseconds OFF ((theConf.timers[TPWOFF]) is a VISHAY TSAL6200.
    - In this case, Period of 28.8us, Frequency of 38kHZ and *off* duty cycle of 65% (35% **On** Cycle).
- The green signal is the RECEIVERs state, LOW is NOT BLOCKED (verifying it is receiving the emitters pulses) and HIGH is Blocked (not receiving pulses).
    - There is a "lag" in the receiver to start returning a valid Blocked/Not Blocked status. Per manual it is between 7 and 15 pulses.
    - Therefore, a Sampler time is needed to read the signal at an appropriate moment for a valid response.
        - If too early will return a BLOCK state
    - The timer is set to ½ the PWM ON Pulse duration (theConf.timers[TPWON]) and ONLY during the ON phase
- The blue line is used for debugging in the Logic Analyzer and shows when the RECEIVERs signal is sampled to return a 0=Not Blocked or 1=Blocked state.

Regarding the timing events as seen by the Logic Analyzer you will see a difference between theorical settings values and "logic analyzer" real timings. Much of this is due to the ESP32 RTOS overheads so it will not be exactly the value required. Also, 38Khz is not a power of 2 and therefore the ESP32 adjusts certain pulses and seem longer than the others.



Here is a detail diagram of the sampler timing. A timer is set to read the Receiver signal at this point and consider it a Block/NO Block state.

Cannot use Interrupts because of the lag and if blocked there would be no edge change and hence no Interrupt.



The configuration structure is the following:

```
enum
timerNames{TOPEN,TCLOSE,TREPEAT,TRELAY,TREPEATW,TDISPLAY,TBREAK,TCOOL,TPWMON,TPWMOFF,T
BREATH,TIFR,TFREQ,TLAST};

typedef struct config {
    uint32_t    centinel;
    uint8_t     rel;
    uint32_t    waithal;
    bool        hal,paja;
    int         ciclo;
    char        name[20];
    bool        traceTask,traceBoot,traceCmdq,traceISR,traceRmaker,traceTimers;
    int         reboots;
    uint16_t    timers[16];
    uint8_t     fadebr,fadew,fadesteps,arrowbr,arrowpace;
} config_flash;
```

The pulse controller has an ON and OFF timer duration, usually the same but can be different and the OFF must be greater or equal to the ON. Units are in MICROSECONDS us, not MILISECONDS ms

Diagram of the ON duration of 600us. Parameter theConf.timers[TPMWON]

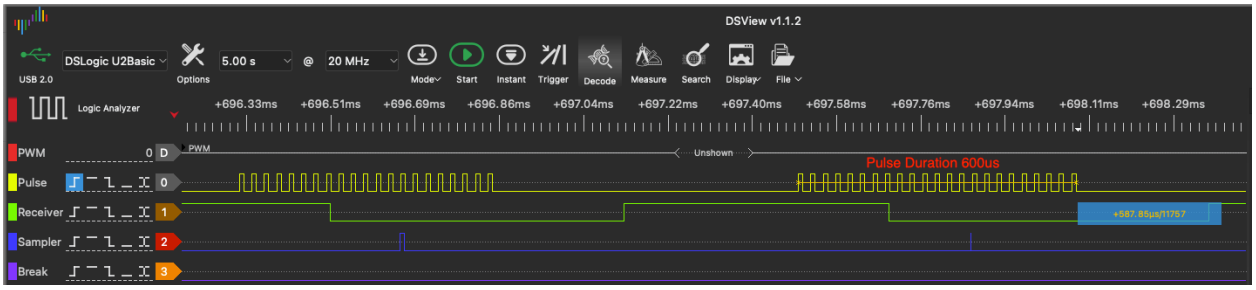
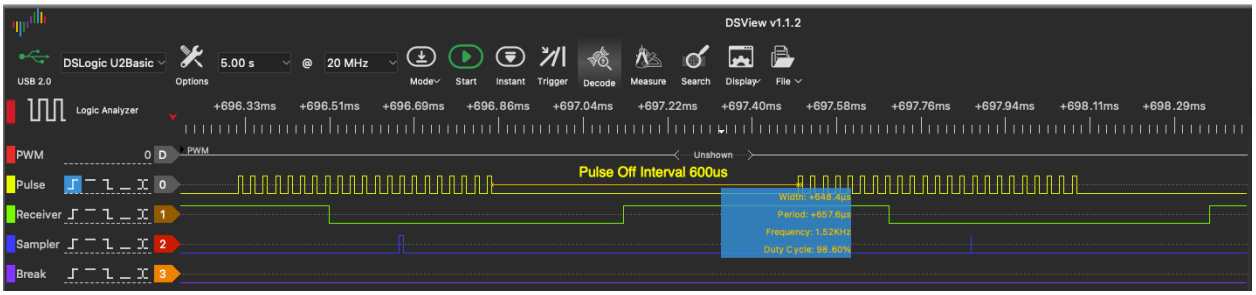
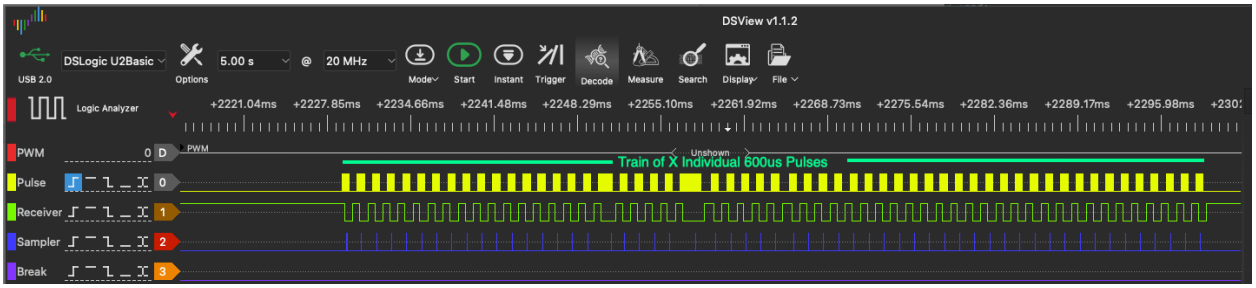


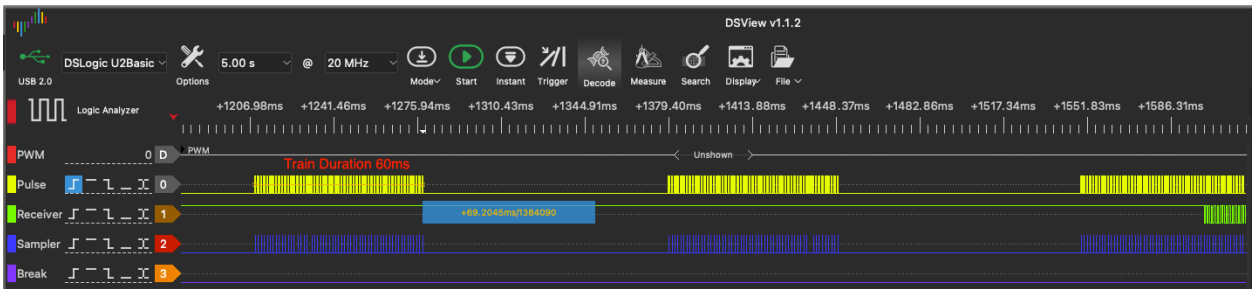
Diagram of the OFF duration of 600us, theConf.timers[TPMWOFF]



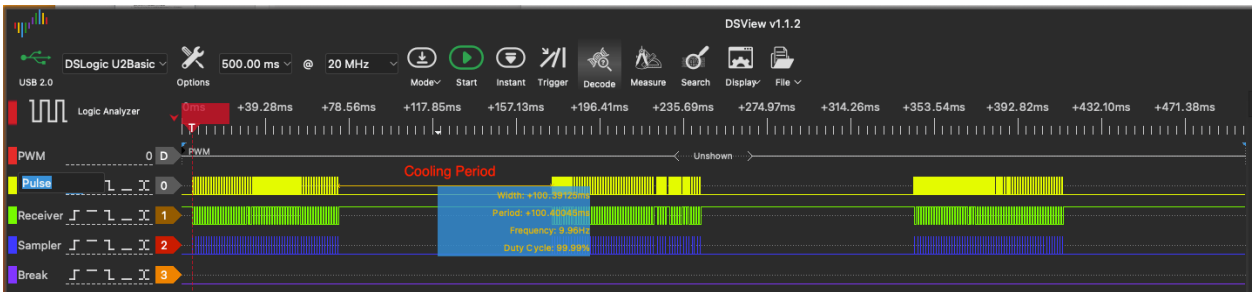
Due to overheating of the Emitter, we use a set of pulses, aka TRAIN, and it has the following characteristics:

A **Train** of pulses is used to increase emitter cooling; theConf.timers[TBREATH] in miliseconds in this case 60ms which allows for 50 pulses of ON/OFF times (600us+600us).

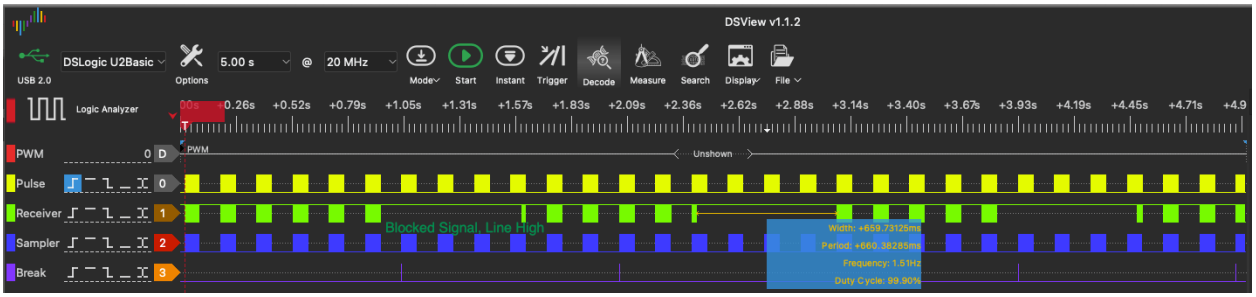




After which there is a COOLING period of X ms which is the parameter theConf.timers[TCOOL] and in this example its 100ms. All to not overheat the Emitter but still get a Blocking/Non-Blocking signal(below).



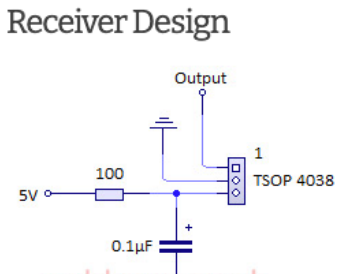
In the diagram below we can see the Blocked and Unblocked signal during the active pulses. When blocked there is no return from the Receiver but a HIGH signal.



Hardware

The Receiver is very simple. Vss, GND and Signal. Vss can be 5V or 3.3v but from a different source than the ESP32 due to lack of power (need 400-600mA vs 40mA from the ESP32). A 100ohm resistor to the Vss signal and a 0.1uF cap. Cable distance is not a problem.

TSOP4038, check the datasheet.

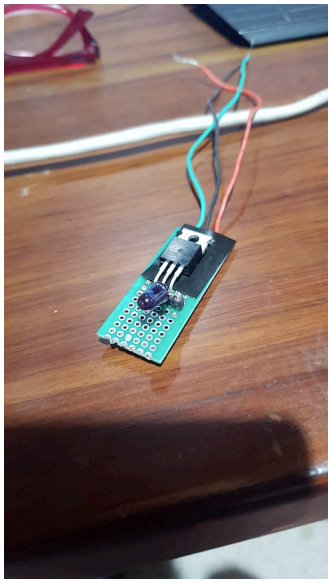


The Emitter must be switched so a Mosfet or NPN Transistor can be used to energize it. Vss is *direct from the power source* and the GND comes from the Drain or Collector. The Transistor/Mosfet must have a 150 ohm resistor to its base and get the Source/Emitter from ground from the ESP32. Don't forget to tie the grounds from external power source and esp32 power source.

Emitter is TSAL6200 by Vishay.

Example with **NPN Transistor** S8050 up to 1.5A.

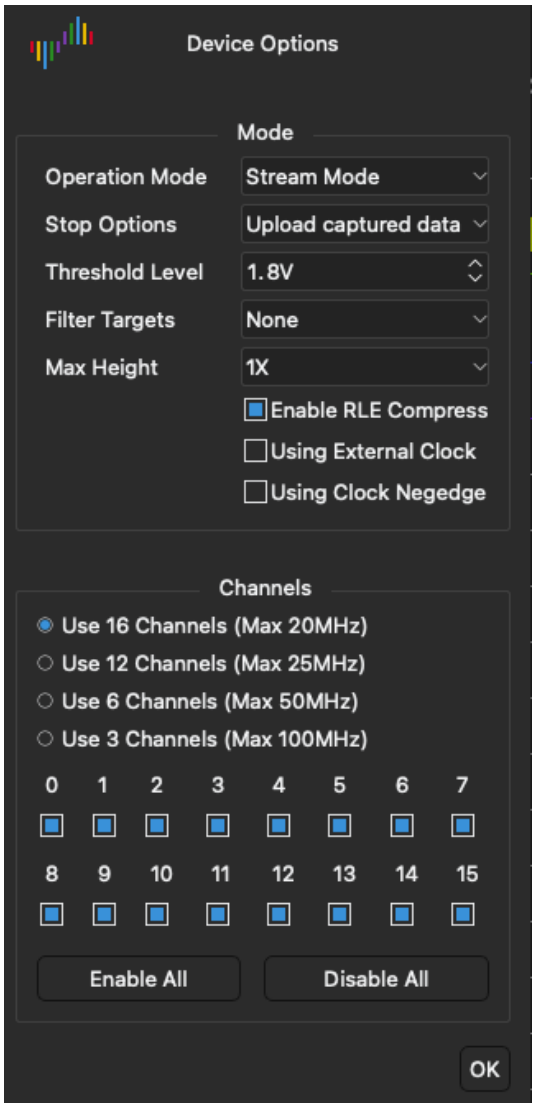
MOSFET version IRF630N or anything for 1Amp



DSView Setup

Setting up the **DSViewer** for the logic 0 requires a change in the Options to set the voltage to something higher than 1.5volts (Threshold Level).

Pulse signal to the Base/Gate(yellow IO22 first pin), another to the Receiver Signal (green IO23 2nd Pin), to the Sample test lead (Blue IO32 3rd Pin) and there is the IO33(Purple 4th pin) when a Break is encountered.



The PCB Board

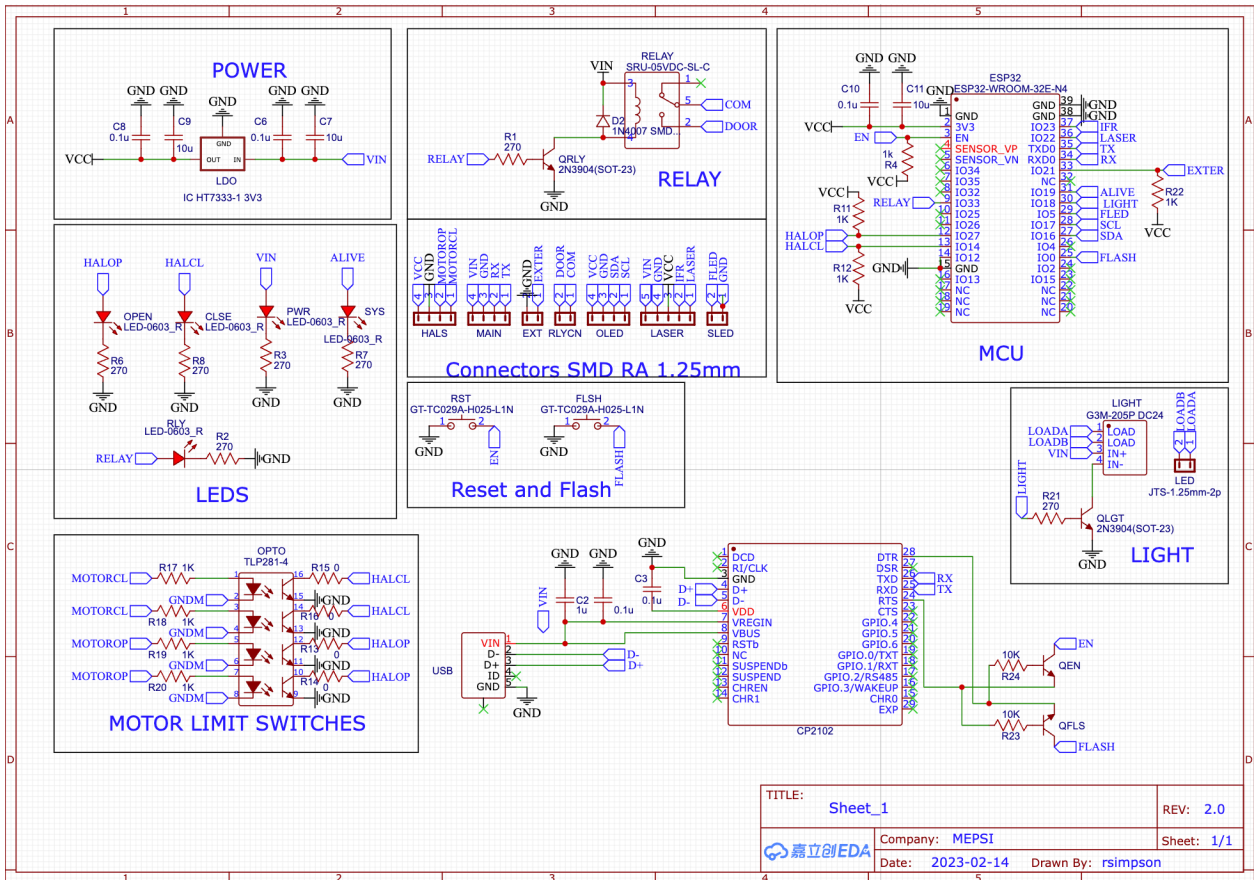
A PCB board was designed to hold all elements required for the Garage Door using EasyEda

Schematic link https://easyeda.com/editor#id=ae46ed4ded0d474eb67743ef520c68af with a 2-layer and 4-layer option. Either should work well.

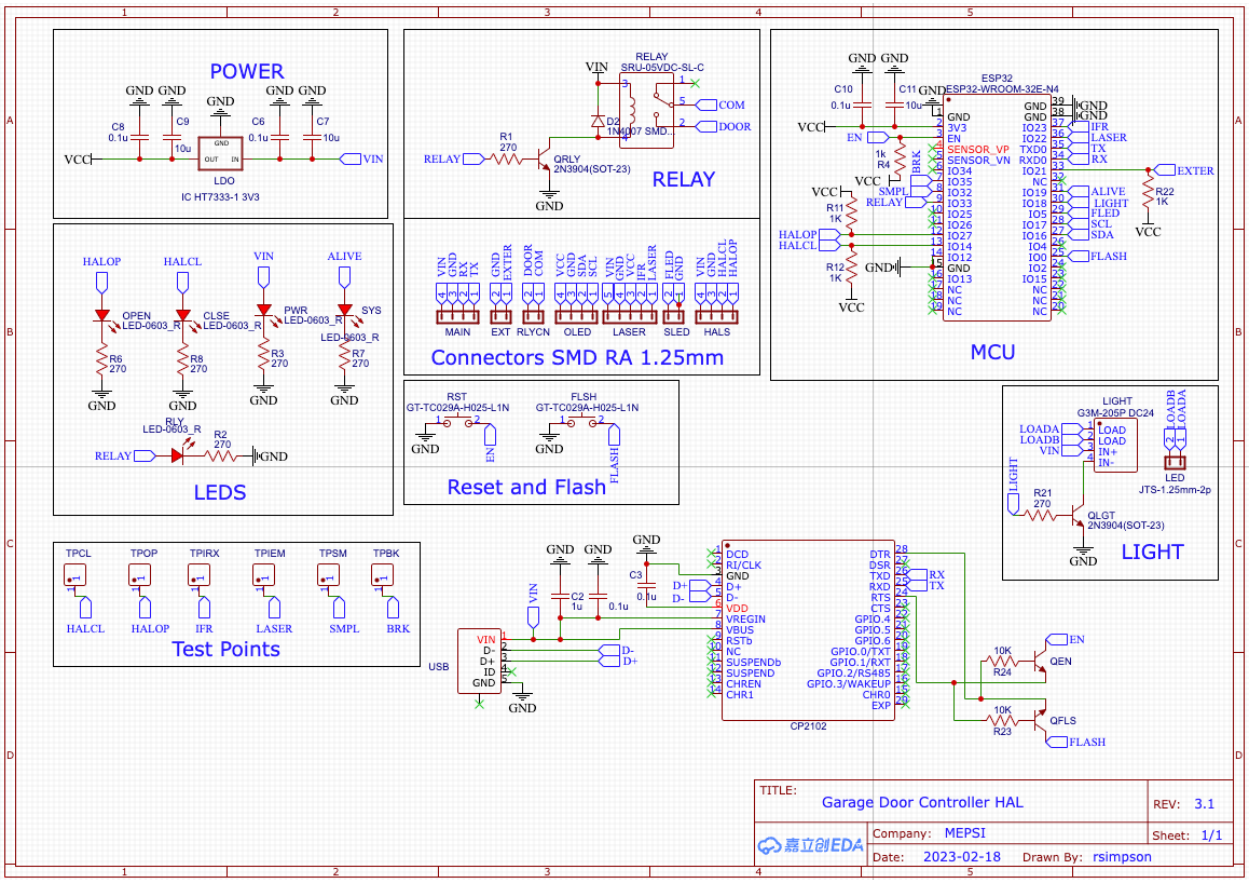
- MCU required
  - Esp32
- LDO required
  - HT7333
    - Pins order is important if substituted
- USB chip optional
  - CP2102

- Optocoupler for the “sensors” required
    - TLP281-4
        - UPDATE. The direct conection to the motor is not possible.
        - Just use HALs and forget the Optocoupler.
- Micro-usb connector Optional
- SMD connectors of several different pins
- Relay required
    - Songle 5VDC
- AC SSR optional
    - G3MB
- Some NPN Transistors SMD low amps SOT-23
-

Below Old schematic with Optocoupler



New schematic without Optocoupler and some testpoints.



The system considers it has a Relay to activate the Door Motor (as standard switch found in the Craftsman and others) to its convenience to close or open the door. RELAY pin.

The IFR module is an external HW (described above) managed by the LASER pin to send the Pulses and IFR pin to get the sensors response.

It will TURN ON/OFF a bulb/led or similar external light at EXTER pin. WARNING!!!! Using a typical LED LUMINAIRE with its POWER SOURCE creates a heck of EMFs that disrupt the system, so if using this the LED power source must be far away from the MCU.

Also, a Strip Led if available will be used to visualize the different states of the Virtual Machine, Closed/Opening/Opened/Closing and BLOCKED each with a different color palette. This is managed via the FLED pin to the Data pins in many strip leds like WS2812B etc. Number of leds is FIXED at 27. External Power Source 12+ or whatever volts must be used and Grounds tied together.

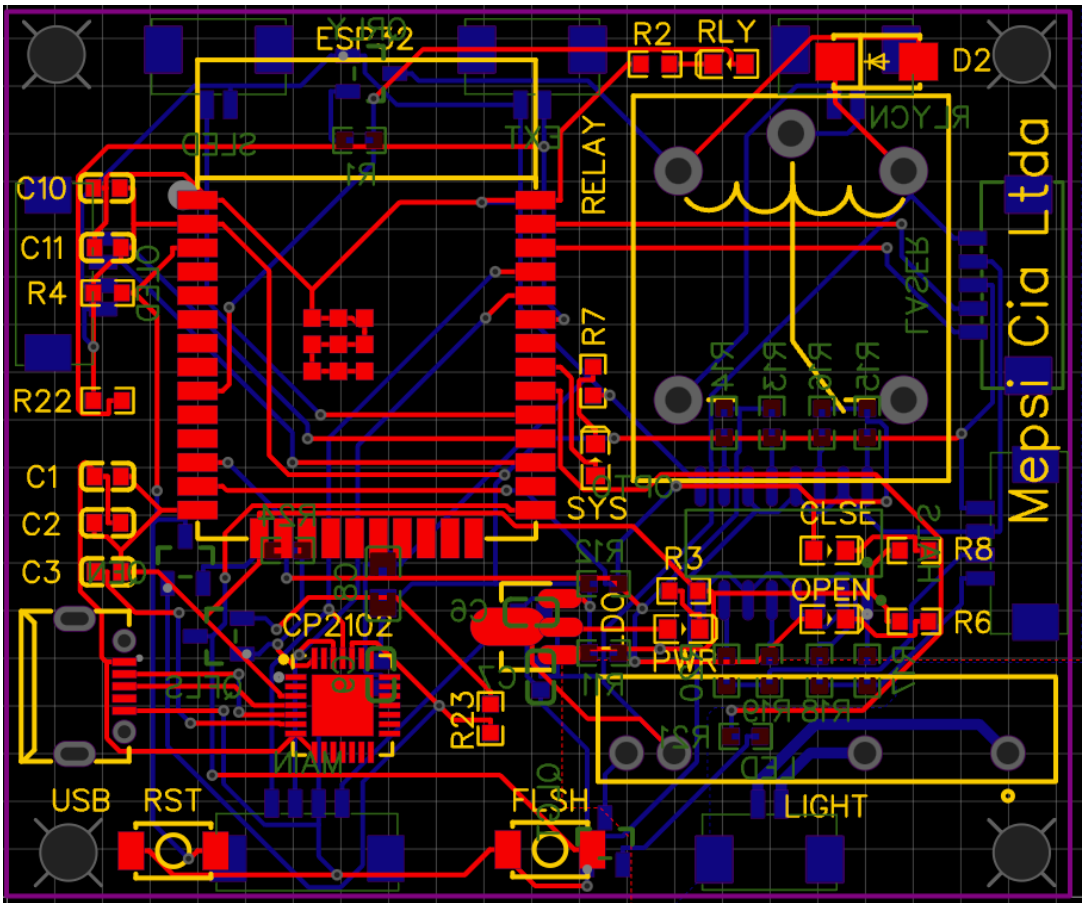
An OLED (128x64) screen if available will also update the VM states. SCL/SDA Pins.

The system is updatable via OTA but, if need be, the RX and TX pins allow for firmware update by pressing the RST and FLSH buttons as is traditional for the ESP32. Also if installed in HW, the USB is used a per standard NodeMCU.

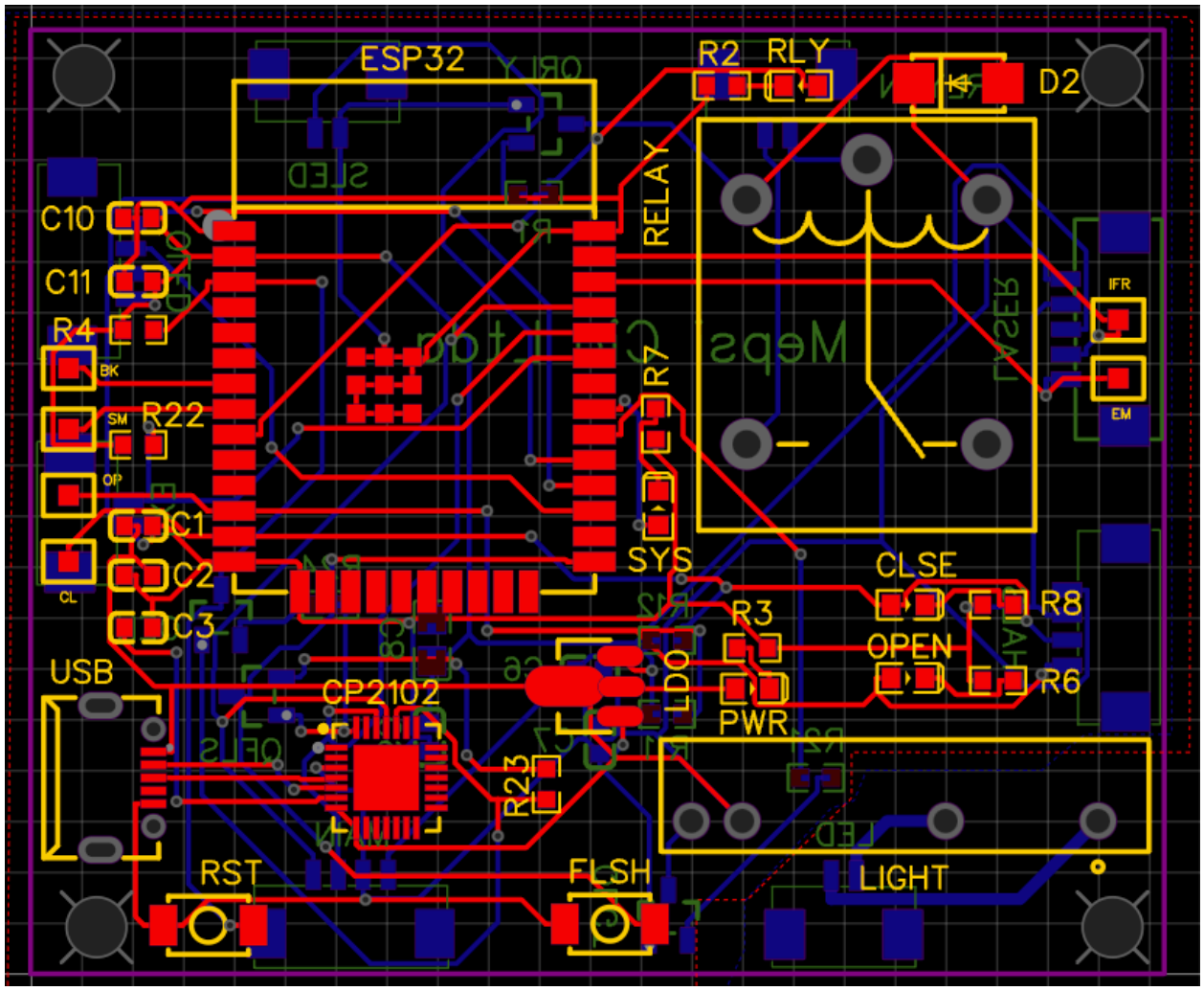
The HALOP and HALCL pins are for the Open and Close sensors, be it Hal sensors or direct connection from the Door Controller via the Optocoupler. *UPDATE, now directly connected toi the ESP32 from the external HALs.*

The Optocoupler works as the Hals sensors, LOW when active(open). HALOP and HALCL lines are pulled high by default and when active will go low. This optocoupler is 4 channels but we only use 2. The other 2 are “spare” for whatever reasons and hence all outputs have a 0-ohm resistor in the two used. Other 2 are left unconnected.

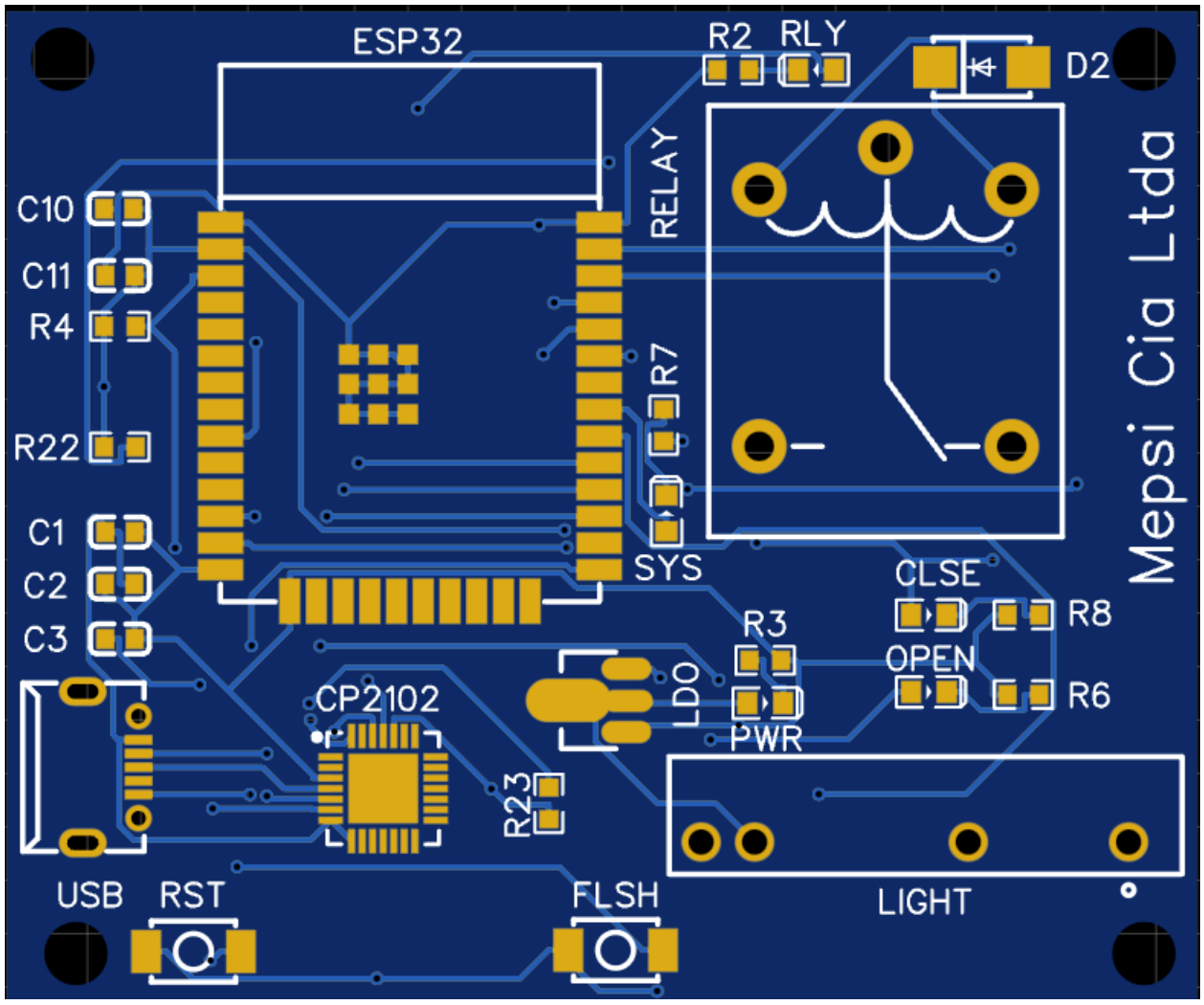
Below, old pcb with Optocoupler

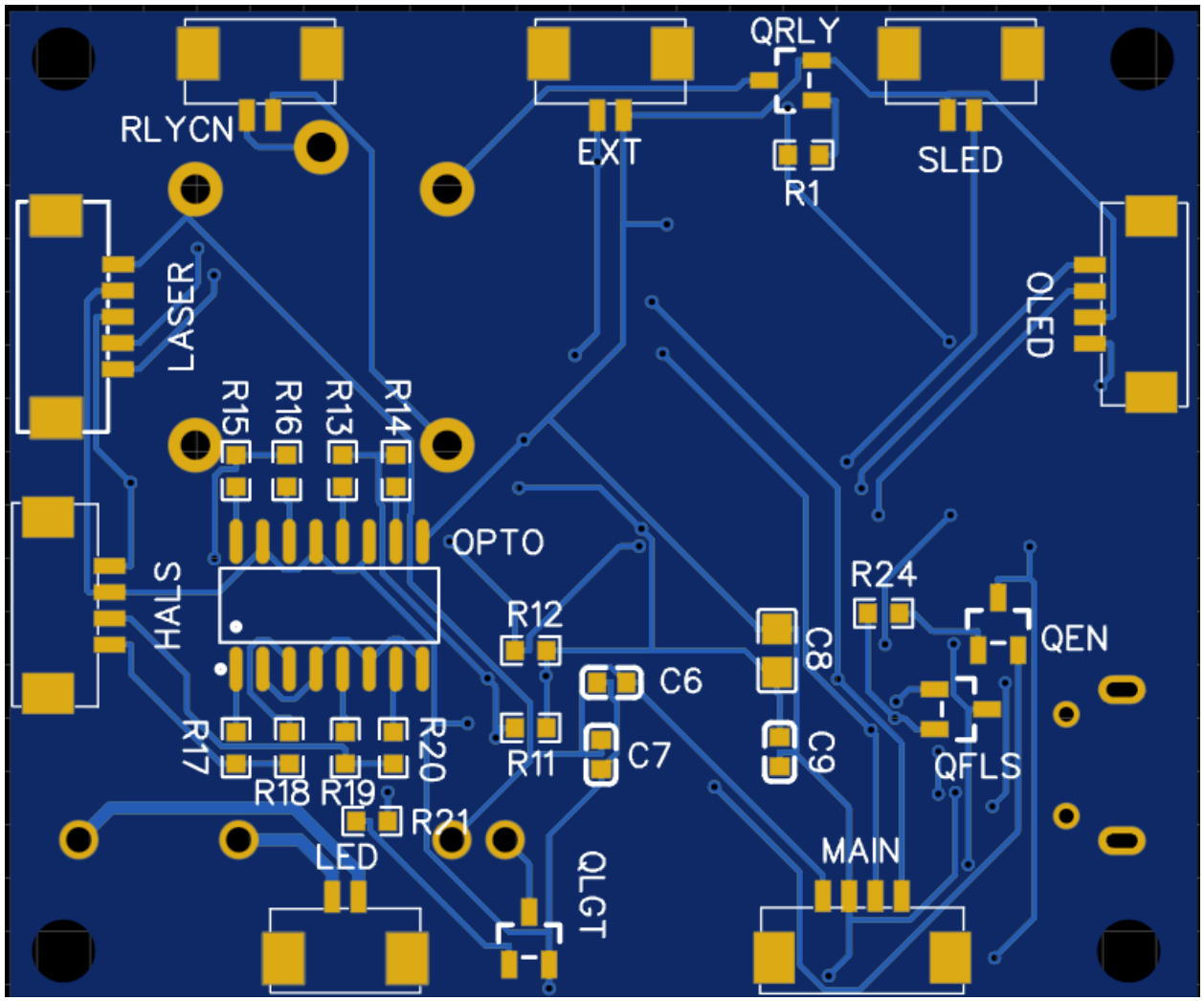


New pcb without Optocoupler

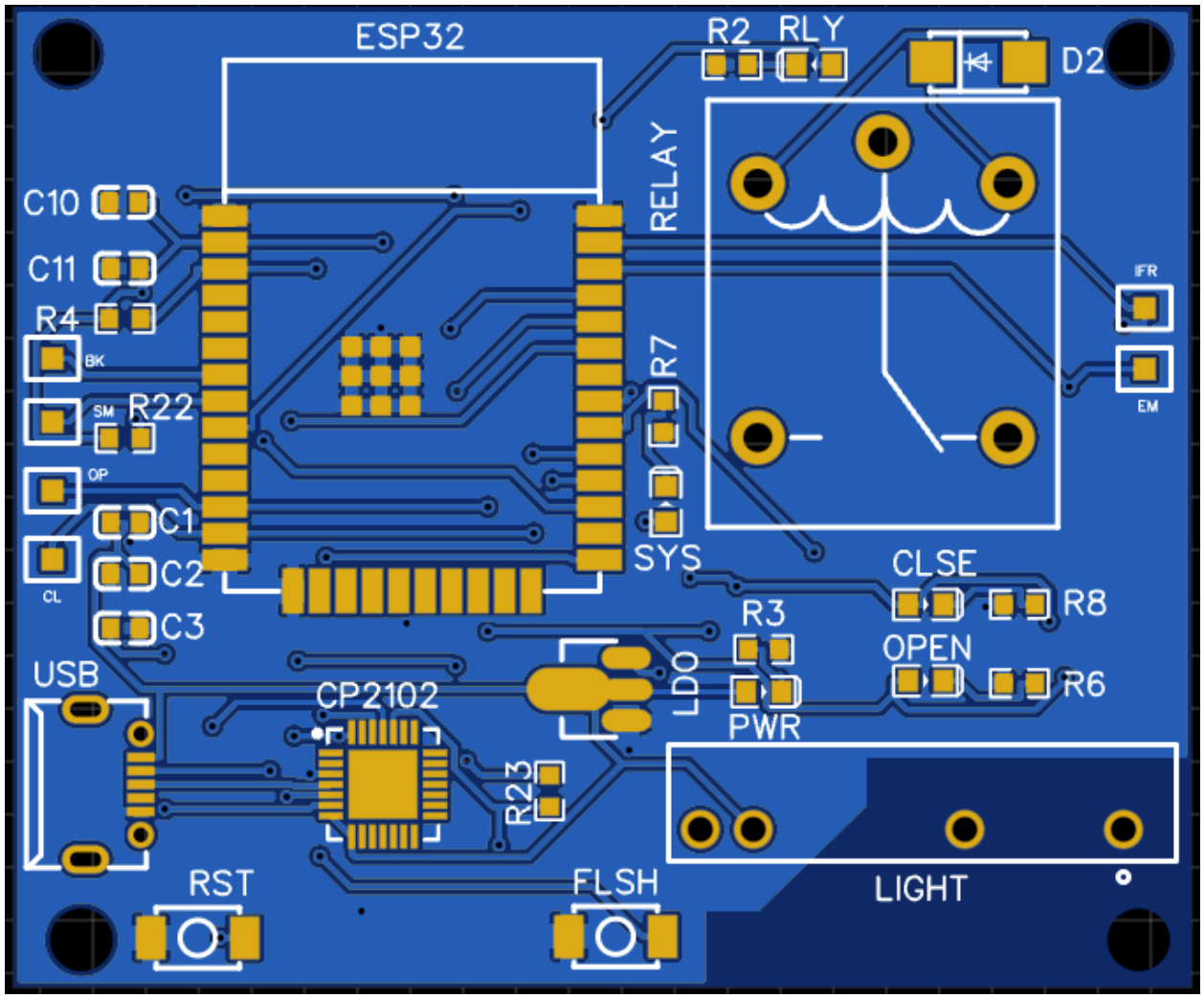


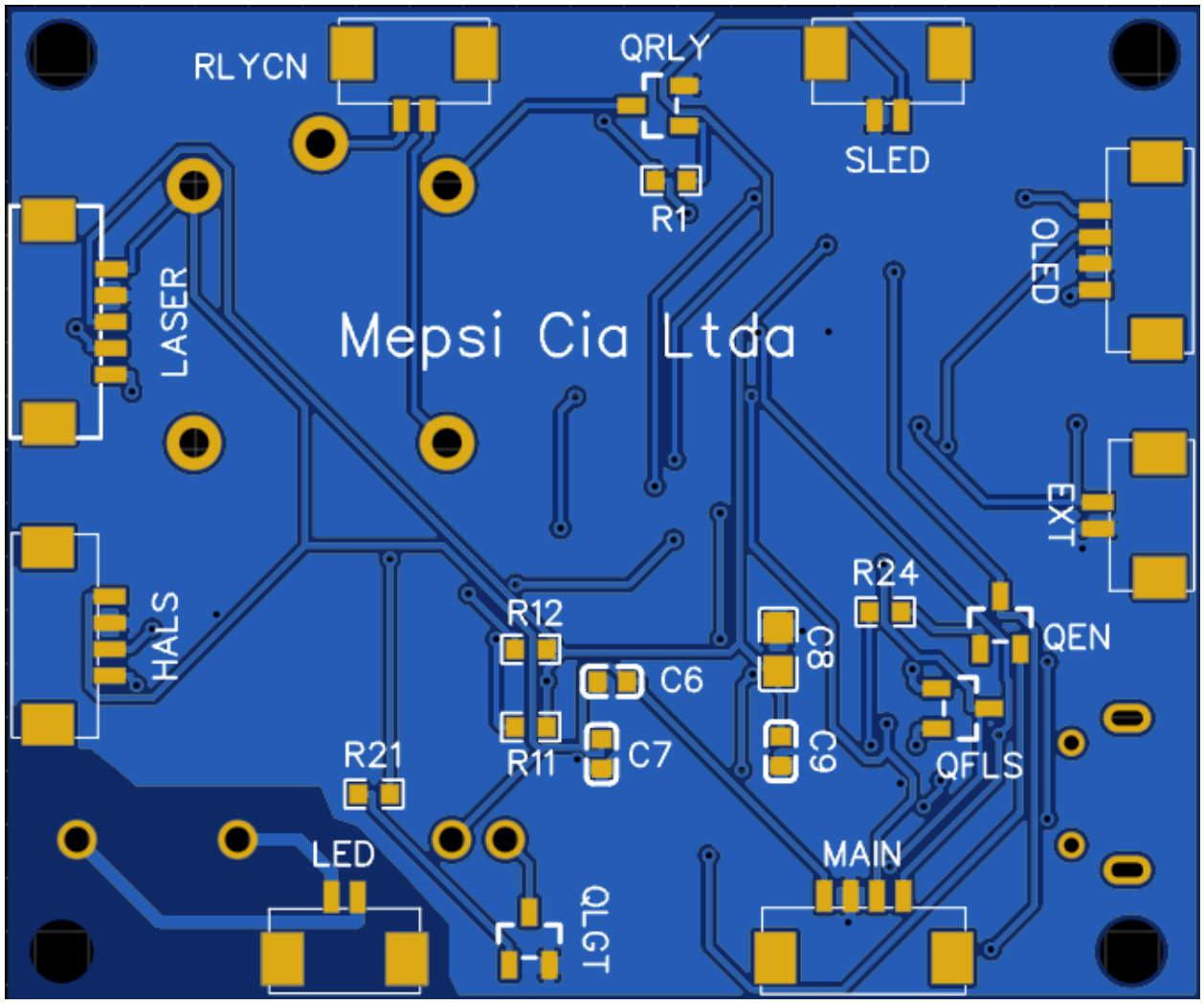
Old 2D with Opto


ESP32
R2
RLY
D2
RELAY
Mepsi Cia Ltda
C10
C11
R4
R22
C1
C2
C3
R7
SYS
CLSE
R8
R3
OPEN
R6
CP2102
LDO
PWR
R23
USB
RST
FLSH
LIGHT

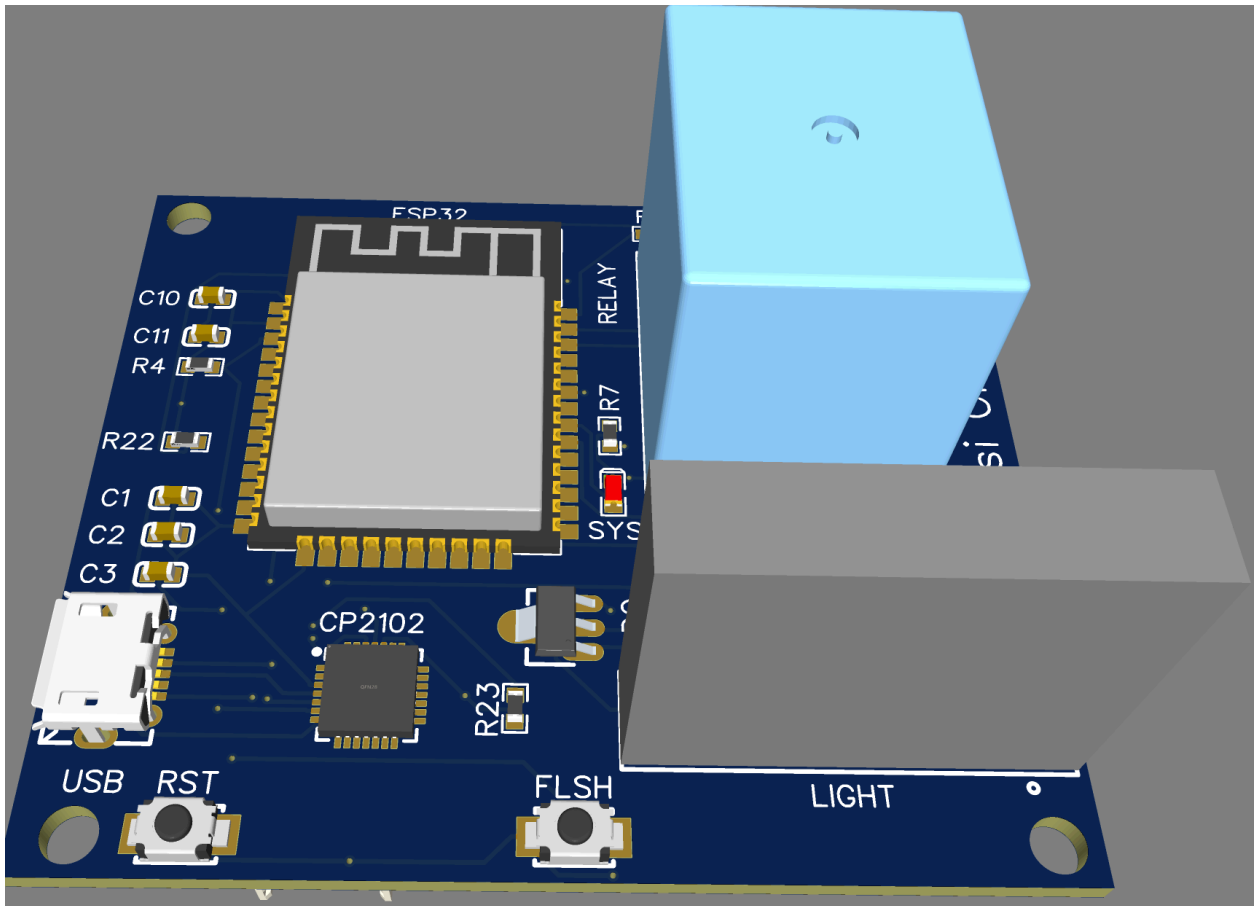


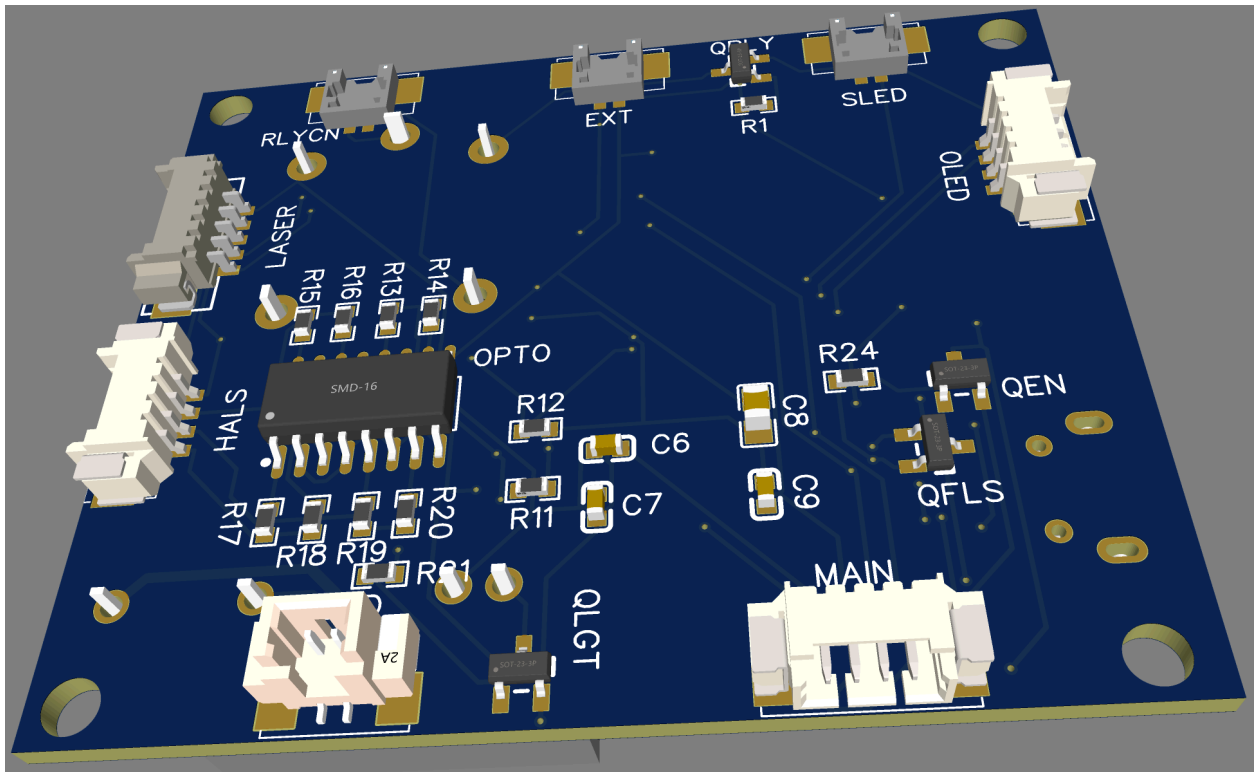
New 2D w/o Opto


ESP32
R2
RLY
D2
RELAY
C10
C11
R4
BK
SM
R22
OP
C1
C2
CL
C3
R7
SYS
IFR
EM
CLSE
R8
R3
OPEN
R6
USB
CP2102
LDO
PWR
R23
RST
FLSH
LIGHT

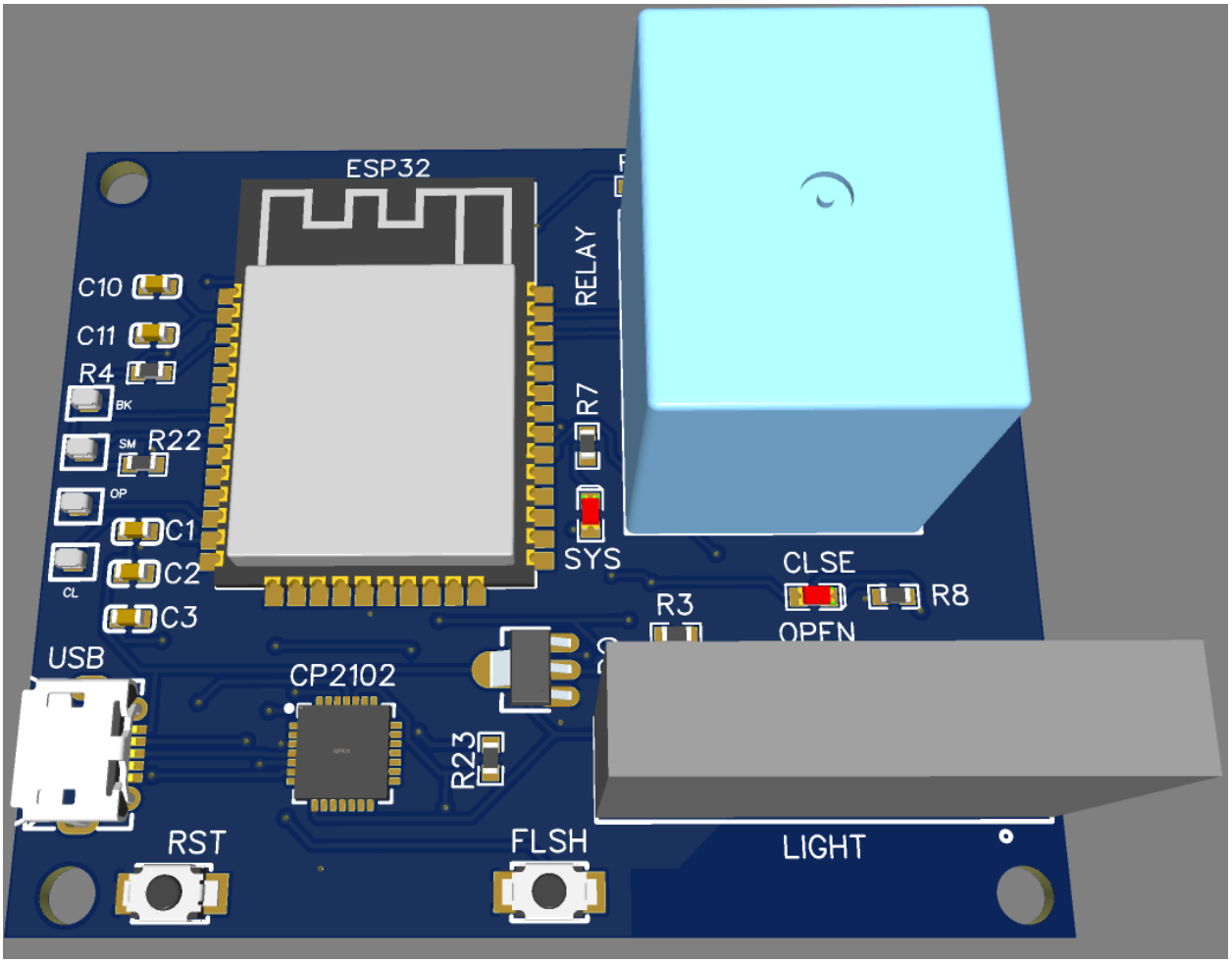


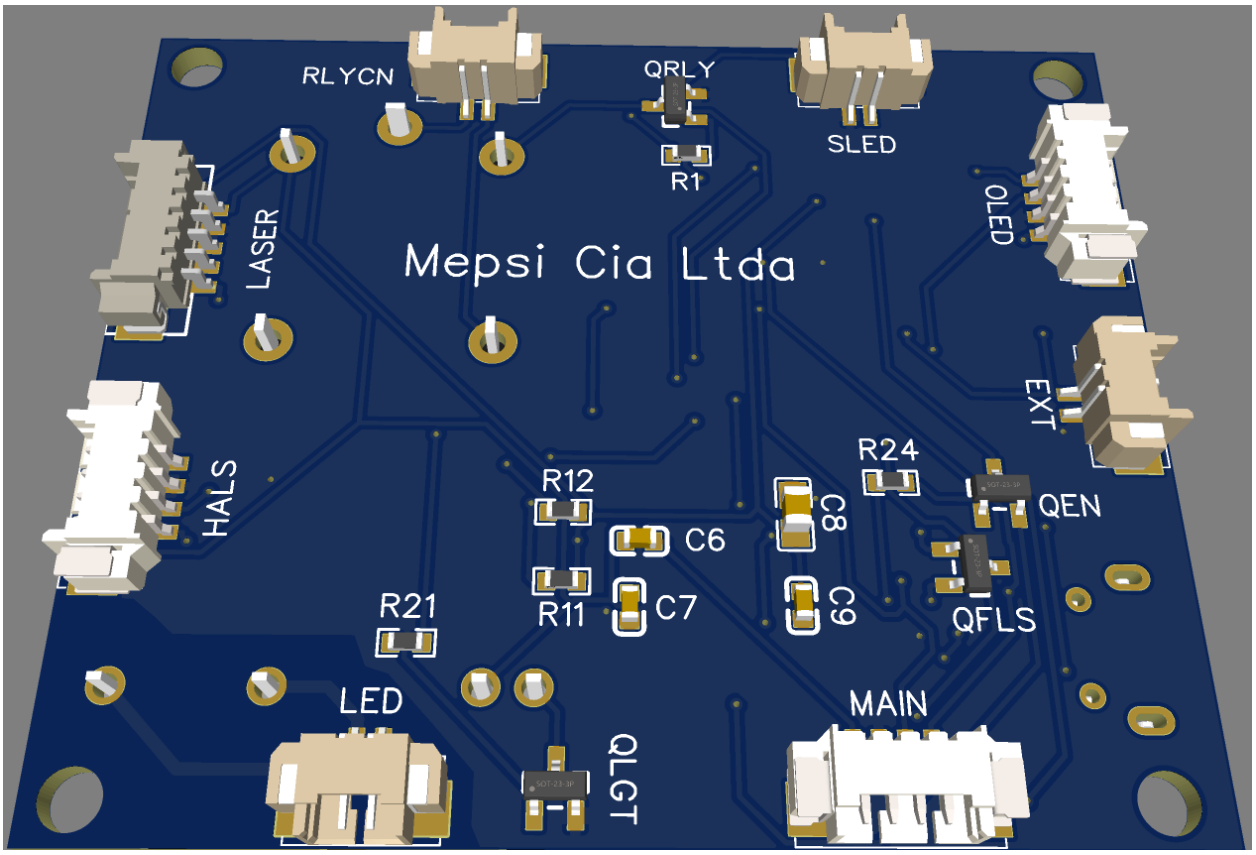
Old 3D with Opto


ESP32
RELAY
C10
C11
R4
R22
C1
C2
C3
R7
SYS
CP2102
R23
USB
RST
FLSH
LIGHT


RLYCN
EXT
R1
SLED
OLED
LASER
R15
R16
R13
R14
OPTO
SMD-16
HALS
R12
C6
C8
R24
QEN
R11
C7
C9
QFLS
R17
R18
R19
R20
R21
MAIN
QLGT
2A

New 3D w/o Opto




RLYCN
QRLY
R1
SLED
OLED
LASER
Mepsi Cia Ltda
EXT
HALS
R24
R12
C8
QEN
C6
R11
C7
C9
QFLS
R21
LED
MAIN
QLGT