



The Byte Attic's

# Agon light™

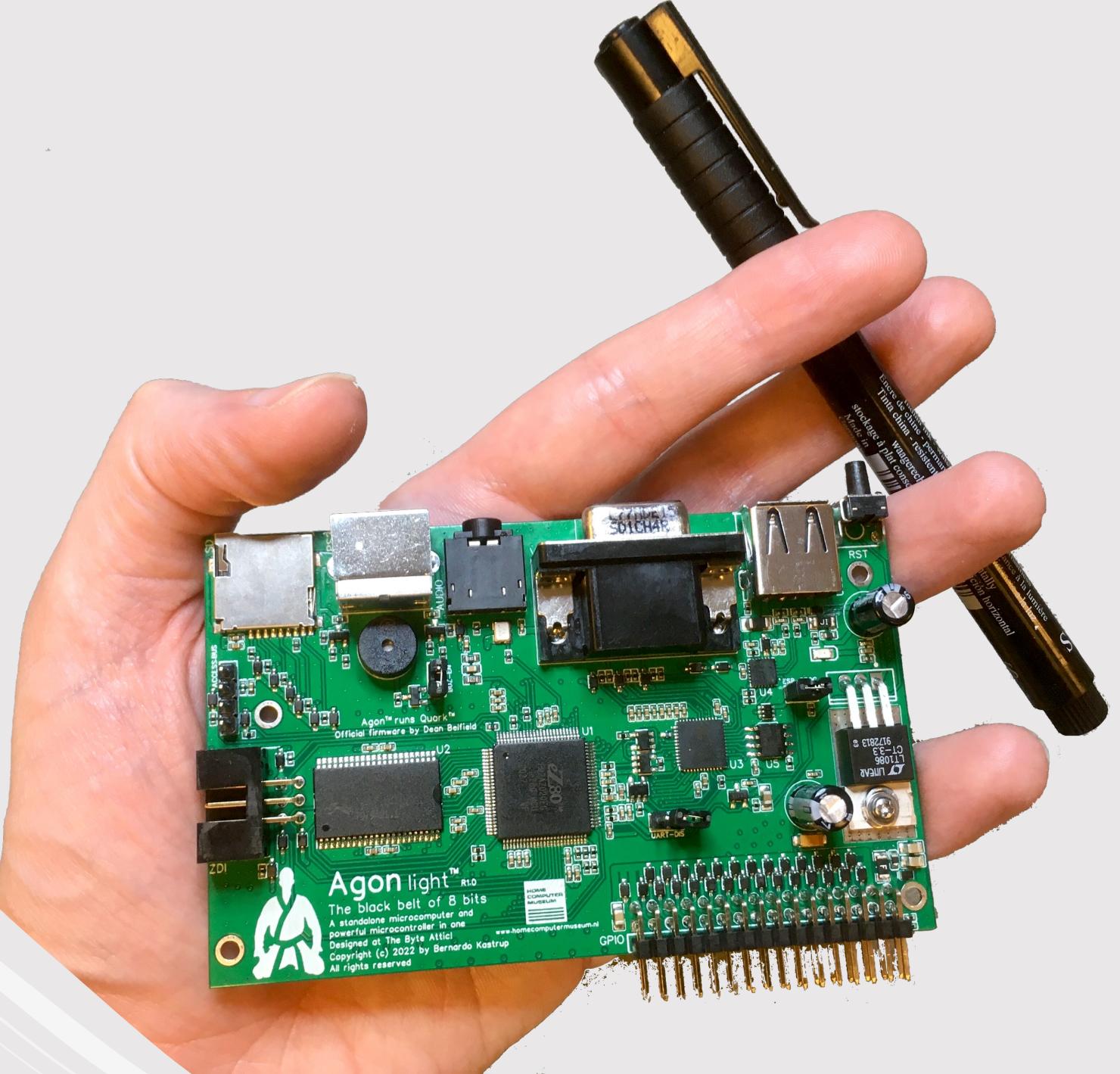
## Manual

© 2022 by Bernardo Kastrup. All rights reserved.

Last updated on: 7/22/22 7:19:24 PM

Provided as-is, expressly without warranties or representations of any kind.

The author disclaims all responsibility for damages incurred as a direct or indirect result of the use of this manual or the system it describes.



# What is Agon light™?

- A modern 8-bit microcomputer and microcontroller in one small, low-cost board
- Requires no host PC: Agon light puts out its own video (VGA), audio (2 identical mono channels), accepts a PS/2 keyboard and has µSD-card storage
- Features a control port with SPI, I<sup>2</sup>C, 20 distinct GPIOs, a system clock output, as well as power (3.3V and 5V) and ground rails
- Features a separate ACCESS.bus header for e.g. an optional status display
- Aims at the best possible trade-off across performance, cost and flexibility with cutting-edge technology
- There are no FPGAs and no emulation in Agon™: the 'bare wires' are exposed directly to the firmware programmer
- Agon light is powered by USB and runs internally at 3.3V

# What is so unique and attractive about it?

- Instant-on, stand-alone, BASIC-programmed\* microcontroller: no host PC or sketch compilation required
- Control your whole house from the immediacy of a BASIC prompt\*
- Say goodbye to assembly:
  - C-programmable audio/video coprocessor firmware with freely available tooling
  - C-programmable CPU firmware with freely available tooling
- A hardware canvas for you to make of it your own dream, firmware-customized microcomputer
- A laboratory for computer science experimentation
- The most advanced 8-bit microcomputer to date
- The best balance of cost, performance and programmability
- Agon light is an open-hardware and open-source project, so you get *all* the information about the system

\* Requires installation of Quark™ firmware



The Byte Attic's

# Agon light™

Technical  
overview and  
specifications



# Architecture and specifications

- Two subsystems:
  - The *processor subsystem*
  - The *terminal subsystem*
- The *processor subsystem* comprises:
  - CPU (eZ80F92 running at 18.432MHz)
  - System memory (512KB, 10ns, parallel SRAM)
  - µSD-card port (as main storage)
  - ZDI port (for programming the firmware of the CPU)
  - Control port (including 20 GPIOs) to control your projects from BASIC\*
- The *terminal subsystem* comprises:
  - Audio/video coprocessor (ESP32-PICO-D4 running at 240MHz)
  - Terminal memory (8MB, 133MHz, serial pSRAM)
  - Keyboard port (PS/2)
  - VGA port
  - Audio jack (2x mono)
  - USB 2.0 port (for power and programming the ESP32's firmware)
- The two subsystems communicate with each other via full-duplex high-speed serial link (384 kilobits per second), featuring flow control

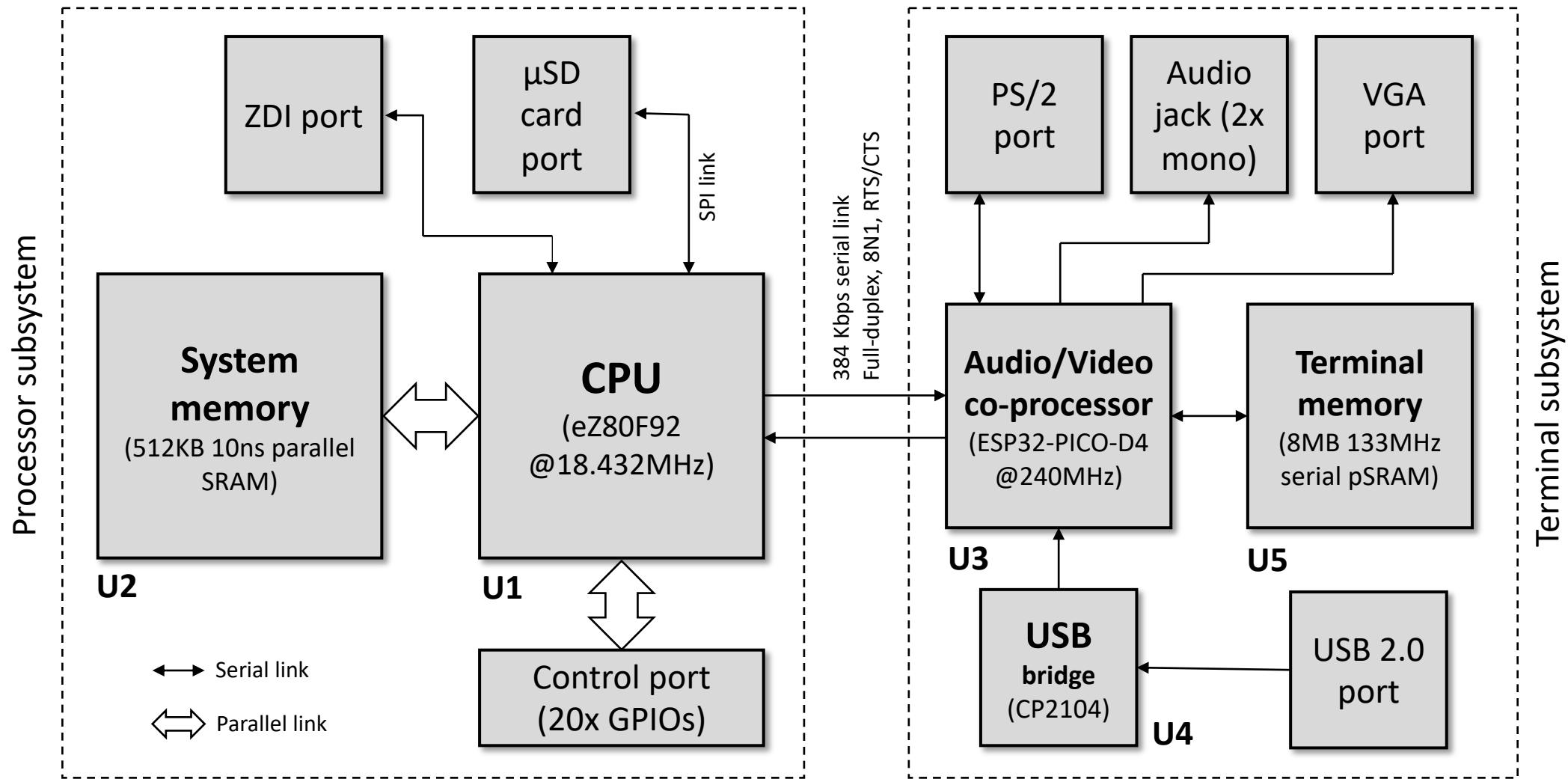
\* Requires installation of Quark™ firmware

# Theory of operation

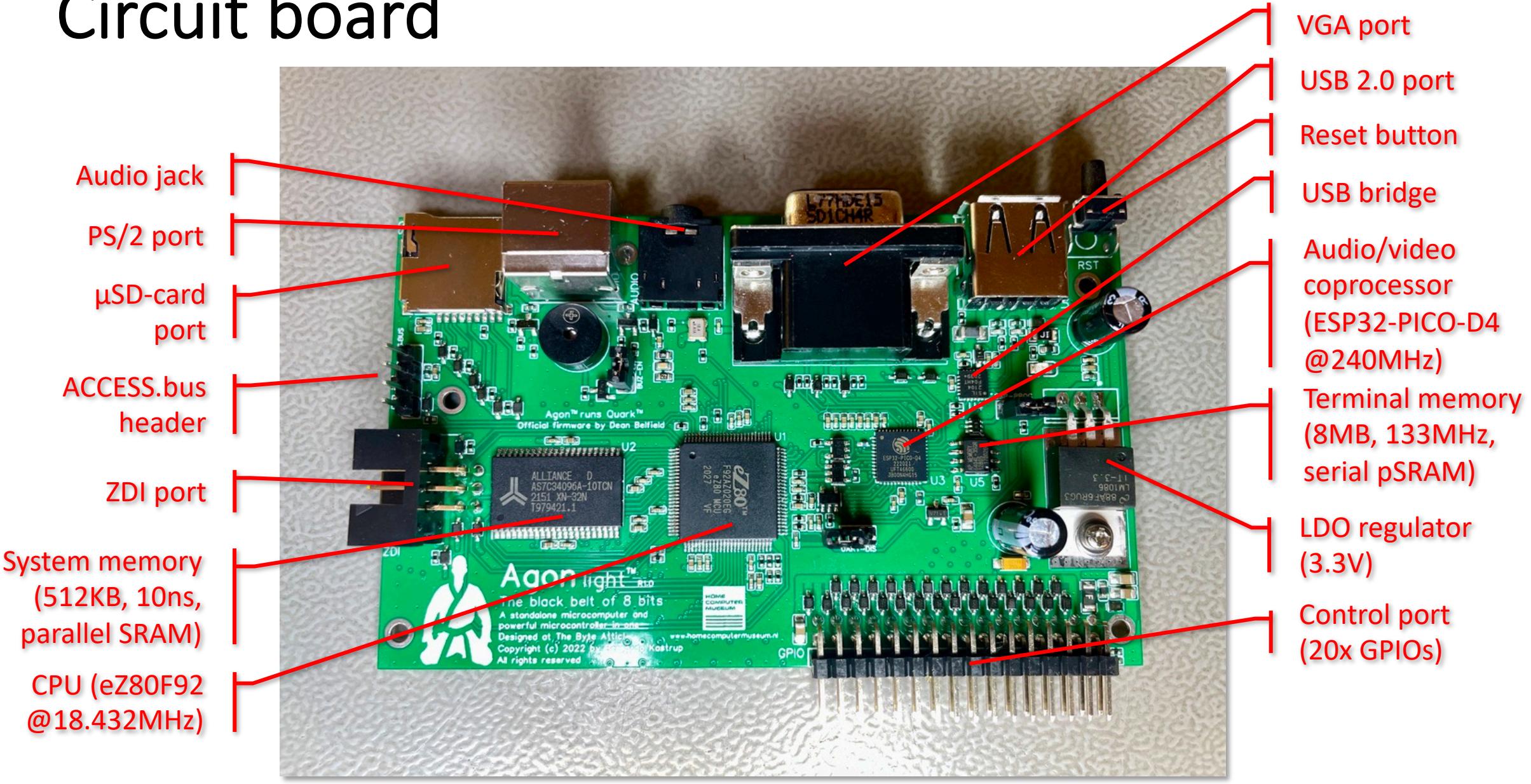
- The *terminal subsystem*:
  - Reads out the (PS/2) keyboard and sends the corresponding keypress tokens to the CPU via a high-speed serial link
  - Generates the screen based on display-list commands issued by the CPU and sent to the ESP32 via a high-speed serial link
  - Produces the VGA & audio signals
  - Supports the FabGL™ library
- The *processor subsystem*:
  - Runs the BIOS and BASIC interpreter\*
  - Executes application code
  - Drives the GPIOs based on the application code
  - Drives the *terminal subsystem* by issuing display-list and audio-related commands to the ESP32 via a high-speed serial link
  - Manages storage ( $\mu$ SD-card)

\* Requires installation of Quark™ firmware

# System diagram



# Circuit board

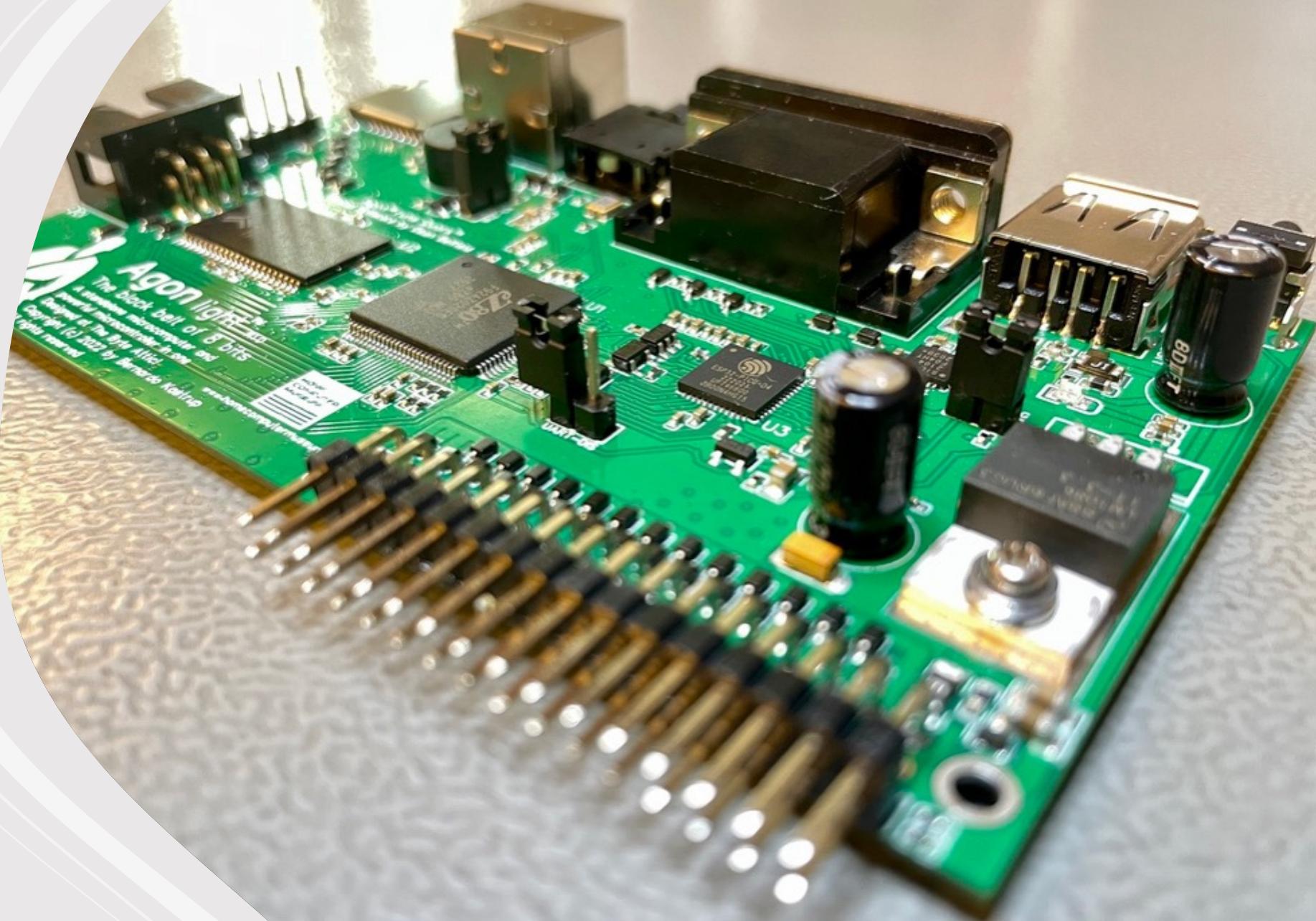




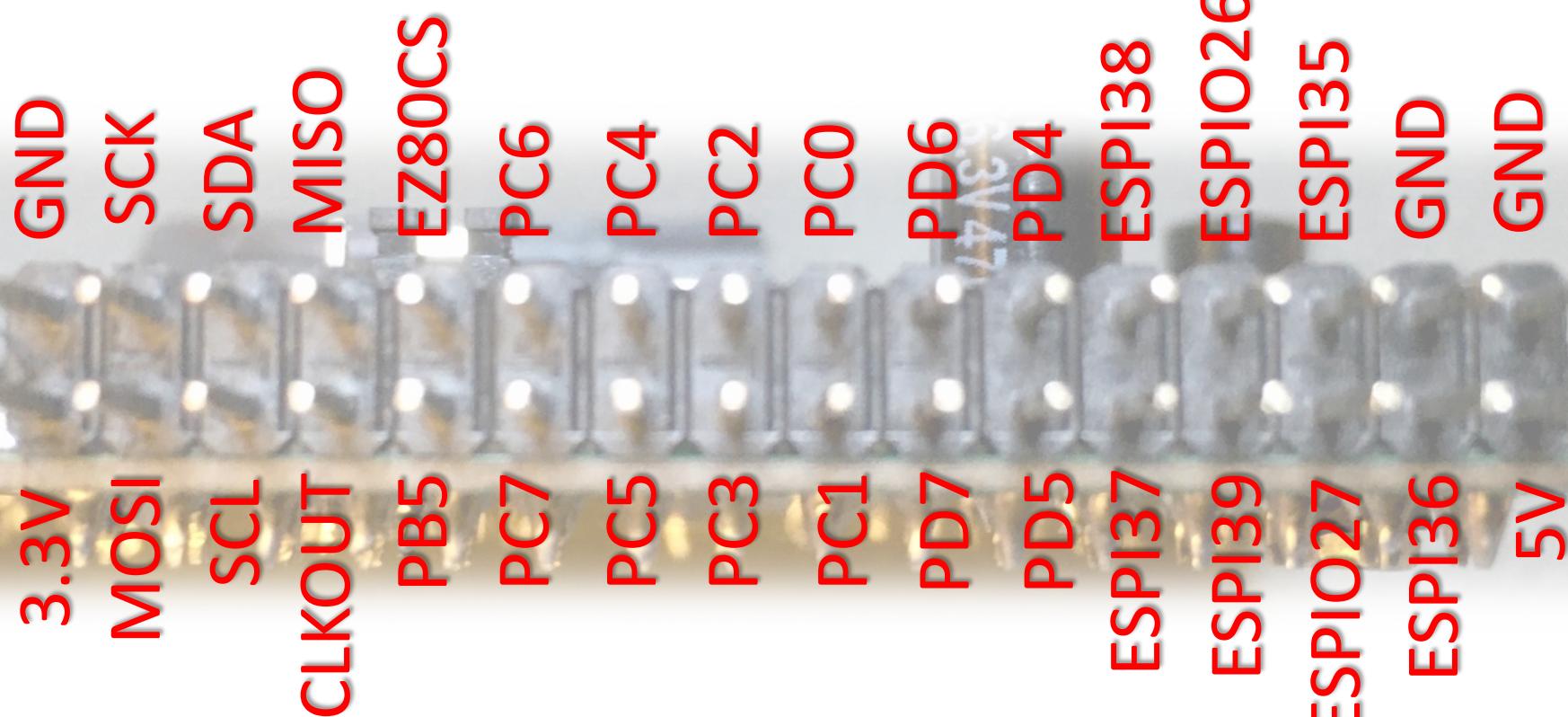
The Byte Attic's

# Agon light™

User's  
guide

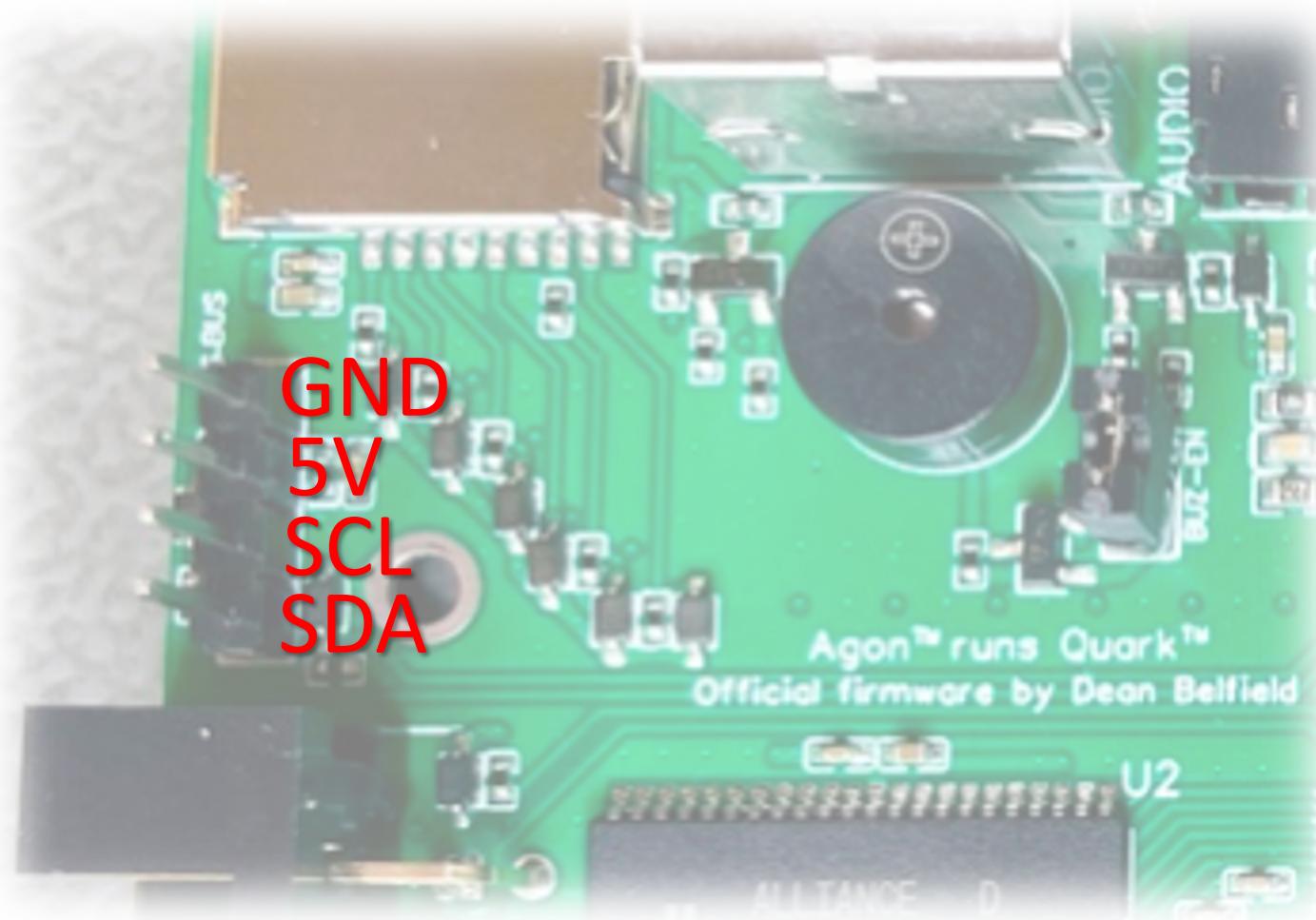


# Control port pinout



See schematics for signal references

# ACCESS.bus header pinout



# Pinout of serial link between CPU and ESP32

- On the *eZ80F92*'s side:
  - Pin 68 (**PD0/TXD0/IR\_TXD**) is the transmitter
  - Pin 69 (**PD1/RXD0/IR\_RXD**) is the receiver
  - Pin 70 (**PD2/!RTS0**) is RTS (signal '*eZ80RTS*' in the schematics)
  - Pin 71 (**PD3/!CTS0**) is CTS (signal '*eZ80CTS*' in the schematics)
- On the *ESP32-PICO-D4*'s side:
  - Pin 10 (**IO34**) is the receiver (connected to signal '*eZ80TxD*' in the schematics)
  - Pin 22 (**IO2**) is the transmitter (connected to signal '*eZ80RxD*' in the schematics)
  - Pin 17 (**IO14**) is CTS (connected to signal '*eZ80RTS*' in the schematics)
  - Pin 20 (**IO13**) is RTS (connected to signal '*eZ80CTS*' in the schematics)

# Recommended configuration of serial link between CPU and ESP32

---

*Channel:* full duplex, asynchronous

---

*Baud rate:* 384,000 bits per second

---

*Signal structure:* 1 start bit, 8 data bits, 1 stop bit,  
no parity bit (8N1)

---

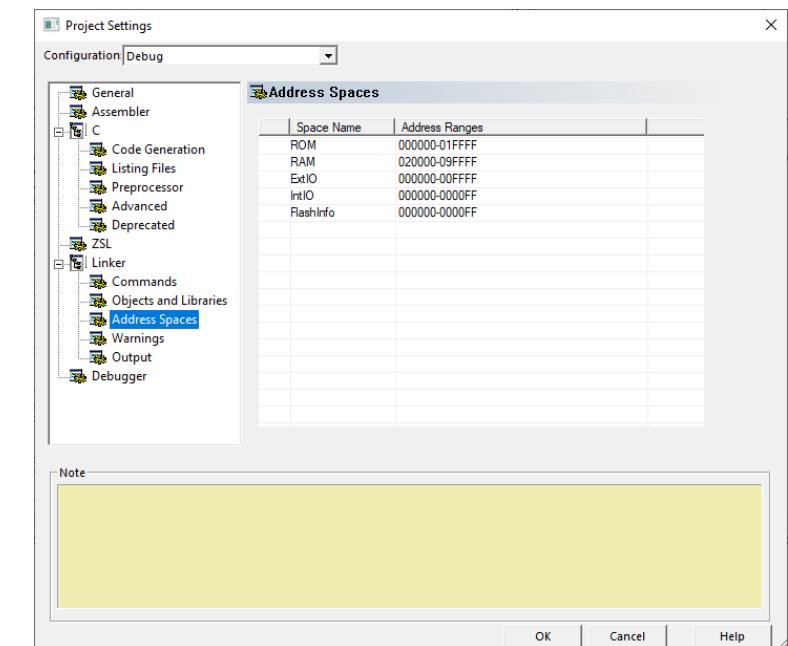
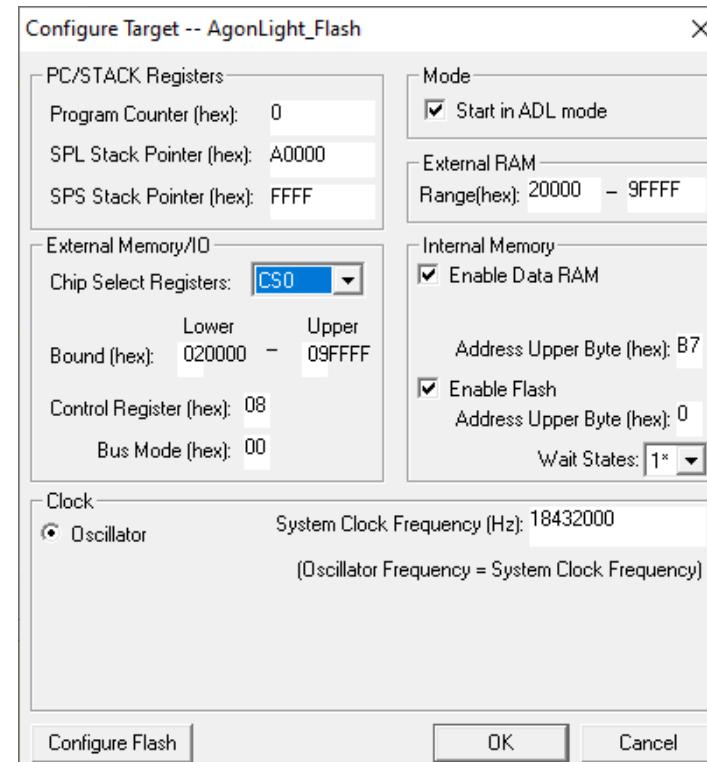
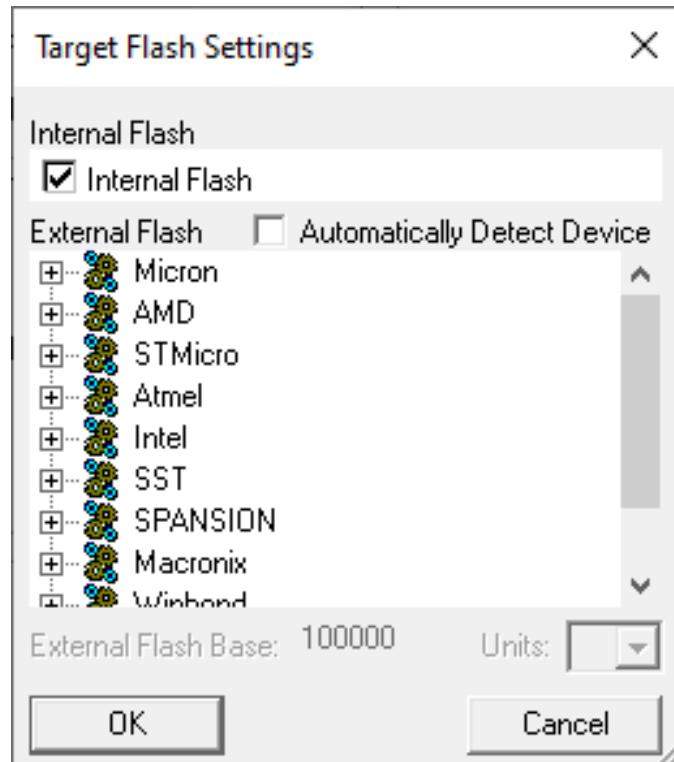
*Flow control:* CTS/RTS

---

Other possible baud rates are: 115200, 128000, 144000, 192000, 230400 and 288000 bps

# Developing firmware for the eZ80F92

- Use the freely-available Zilog ZDS-II™ IDE, downloadable from:  
[https://www.zilog.com/index.php?option=com\\_zcm&task=view&soft\\_id=38&Itemid=74](https://www.zilog.com/index.php?option=com_zcm&task=view&soft_id=38&Itemid=74)
- Documentation is provided in the Agon light Github repository, in the directory /Third party documentation
- Configure your project as per the figures below (CS1, CS2 and CS3 are *not* used in Agon light, so their settings don't matter)



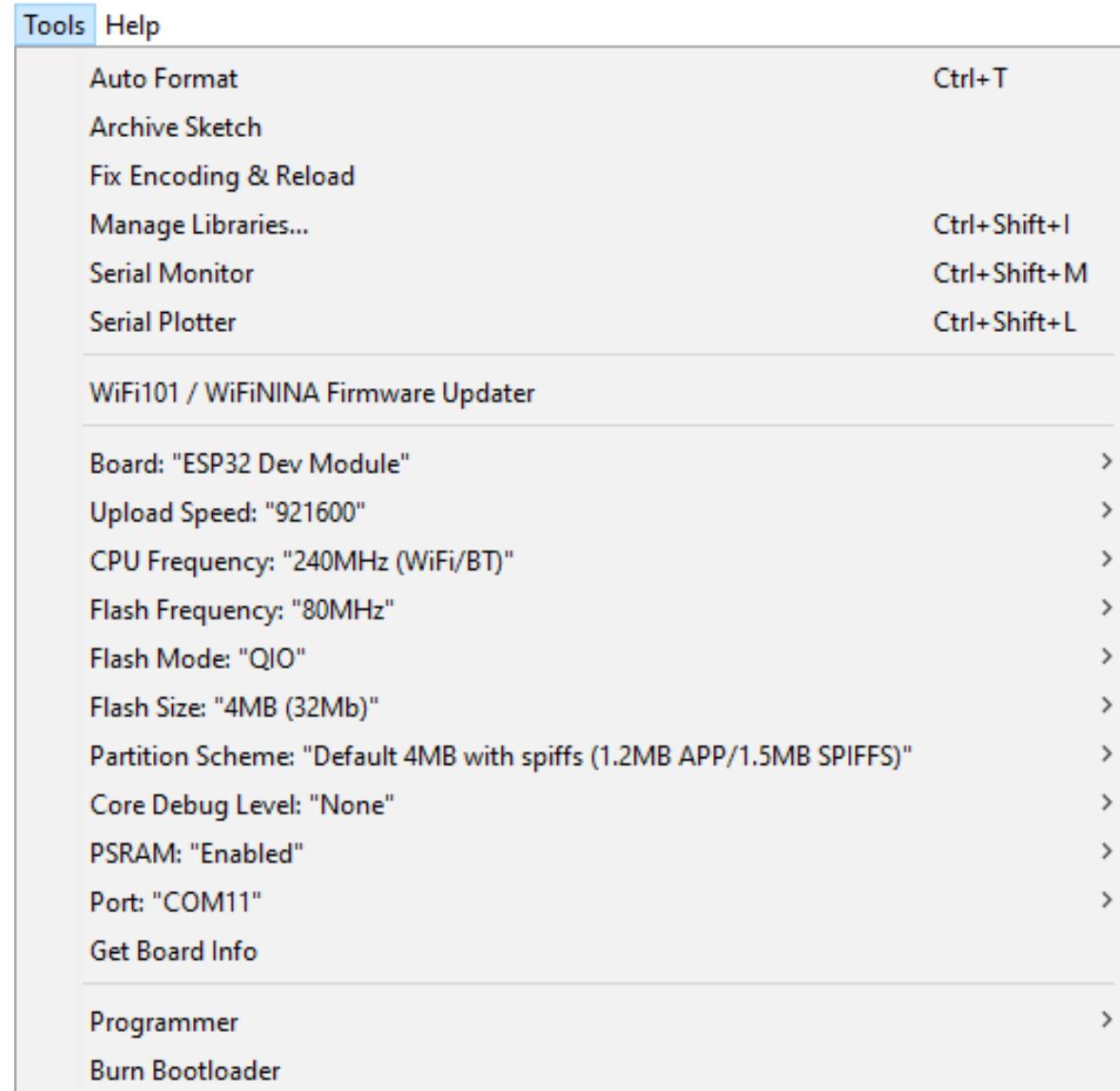
# Required programming/debugging USB smart capable

- To upload firmware into the eZ80F92 CPU, from within the ZDS-II IDE, you will need a Zilog opto-isolated *USB Smart Cable*, shown in the photos
- Zilog product number:  
ZUSBSC00100ZACG
- There seems to be homebrew alternatives to this cable with plans available online, but I have not tested any of them



# Developing firmware for the ESP32-PICO-D4

- Use the freely-available Arduino™ IDE
- Install the FabGL™ library as per instructions available online
  - Link to the FabGL library:  
<http://www.fabgl.org/index.html>
  - Link to installation tutorial:  
<https://youtu.be/8OTaPQISTas>
- The figure to the right illustrates a suitable configuration for loading an Arduino sketch into the ESP32
  - Change the port number to the one active in your case



# Power supply and signal level considerations

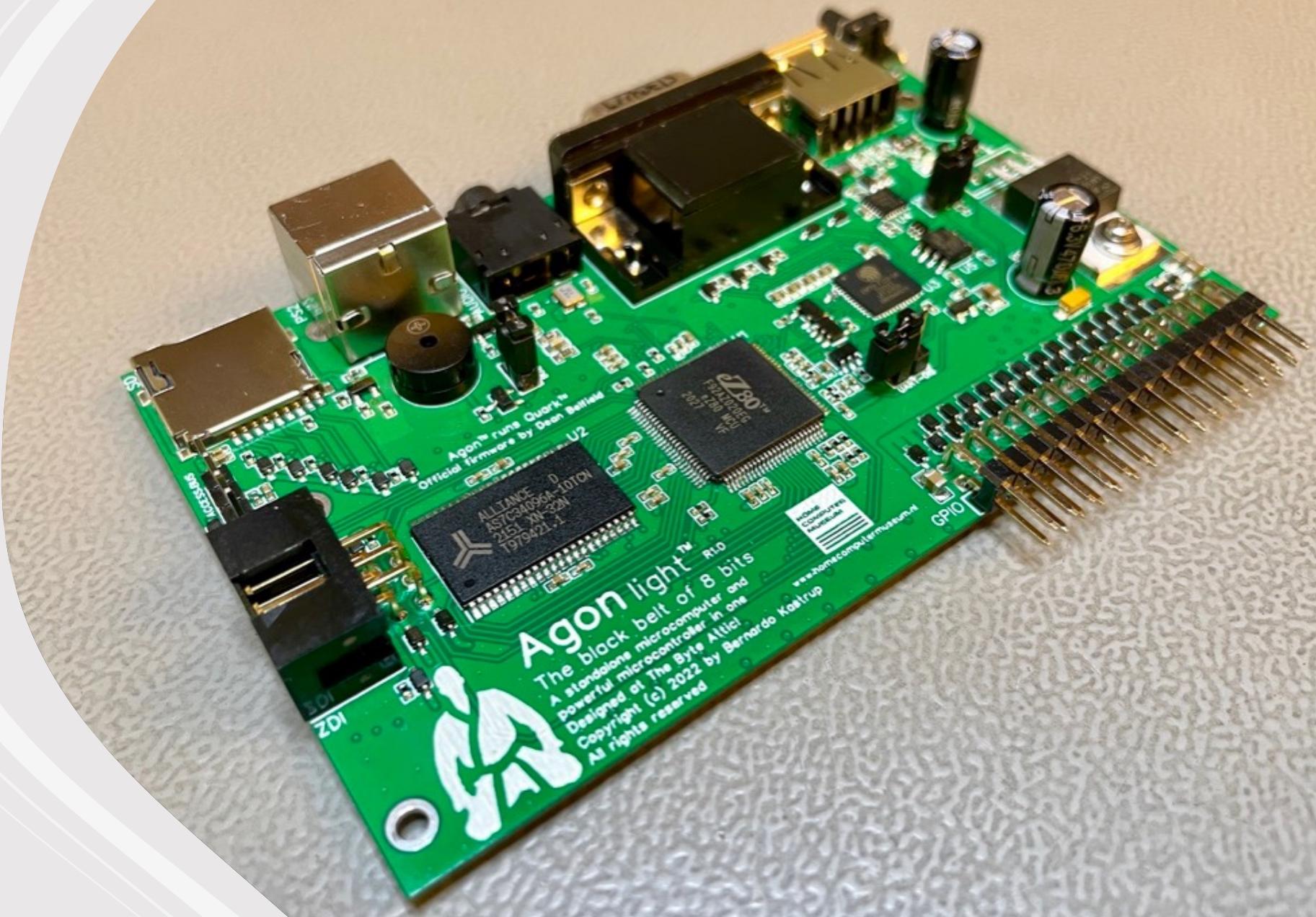
- Agon light can be powered (5V) *either* from its USB port *or* from the 5V pin in its control port
- If Agon light is powered from the USB port, then the 5V pin in the control port can be used to power an external circuit connected to Agon light
- Similarly, the 3.3V pin in the control port can be used to power an external circuit, *but it cannot be used to power Agon light*
- The on-board LDO regulator can provide up to 1.5A of current at 3.3V
  - Maximum *total* current for Agon light's internal use *and* devices powered from the 3.3V pin in the control port
  - It assumes that the USB device powering Agon light can deliver 1.5A; otherwise, that device becomes the bottleneck
- All logic signals on the control port (including all GPIOs) are referenced to 3.3V and, therefore, are *not* TTL-level



The Byte Attic's

# Agon light™

Assembly  
guide

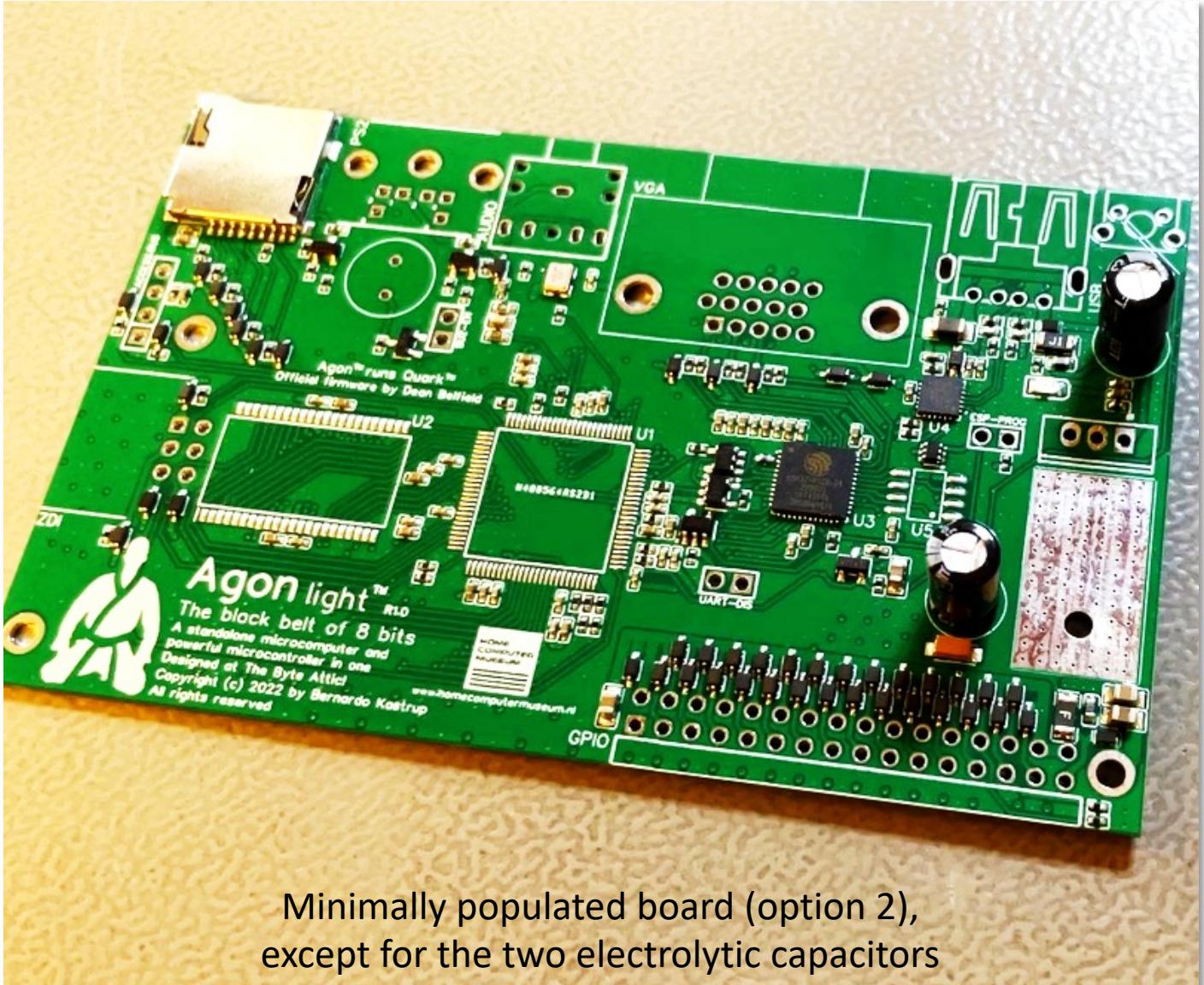


# Assembly options

- There are four options:
  1. You buy the bare PCB and fully populate it yourself (requires a stencil and reflow oven)
  2. You buy a PCB minimally populated with the small parts and the two QFN ICs (U3 and U4), which are hard to solder by hand
  3. You buy a PCB with all SMD parts already populated from factory, only the through-hole parts still needing to be soldered
  4. You buy a fully-populated board, so you need not do any soldering yourself
- Options (1), (3) and (4) will not be discussed further: if you choose option (1) you know what you are doing, option (3) is easy enough, and option (4) requires nothing of you
- Option (2) requires though-hole and fine-pitch QFP drag-soldering. Here are the instructions for doing it properly:  
<https://youtu.be/k9TF2ZCngoE>
- Reasons for choosing option (2): PCB makers charge a premium (usually 50% of the parts' costs) for procuring parts for you, and there are multiple import fees involved. It's cheaper (and better, if you know how to do it) to buy and populate the most expensive parts (U1, U2 and U5) yourself, especially if you are building Agon light to sell it commercially

# Manufacturing files

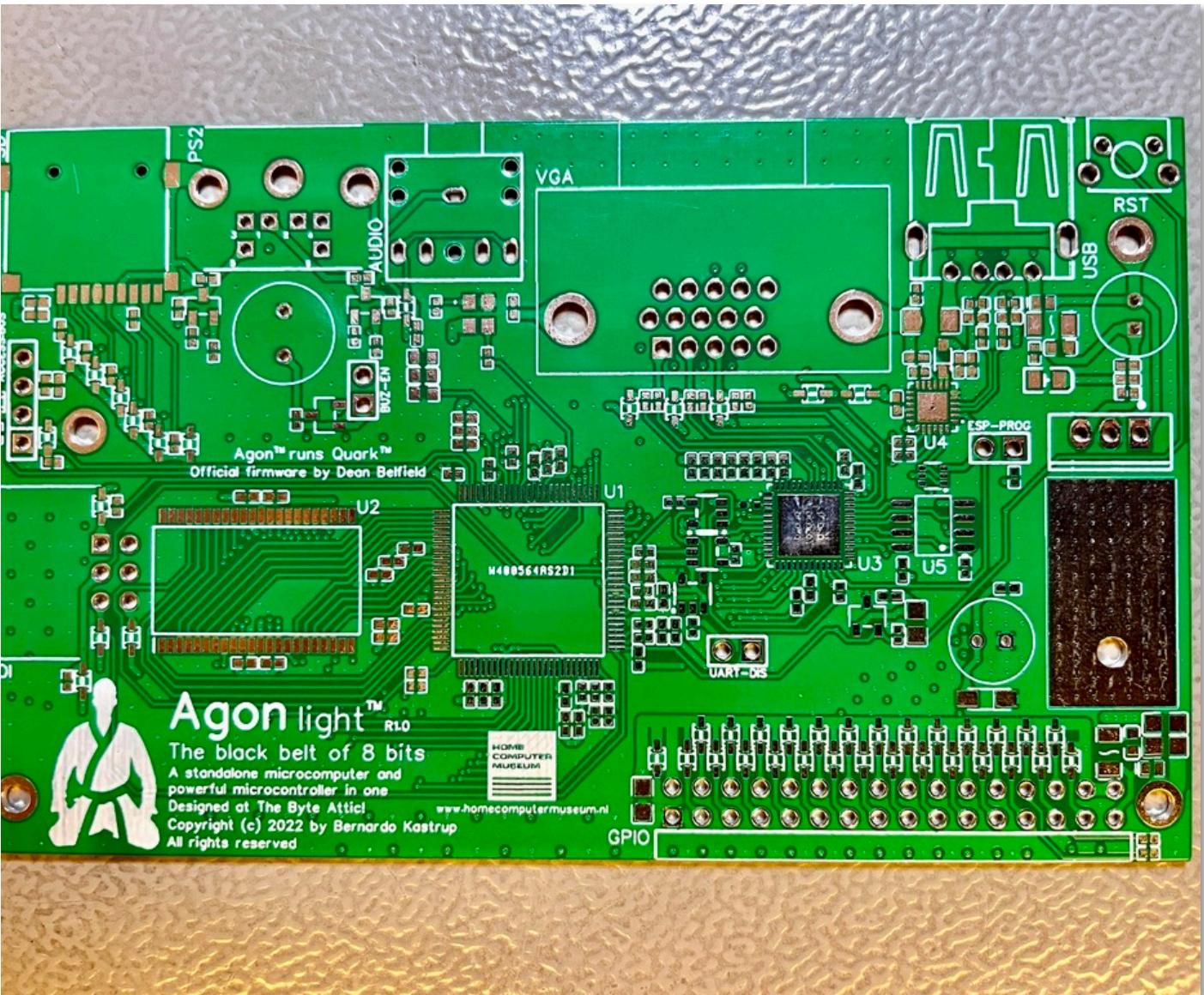
- All files are available in the /Manufacturing directory of Agon light's Github repository at:  
<https://github.com/TheByteAttic/AgonLight>
- For option (4), send the following files to your PCB manufacturer, next to the Gerber file (`Gerber_PCB_AgonLight_R1.0.zip`):
  - `PickAndPlace_PCB_AgonLight_R1.0.csv`
  - `BOM_PCB_AgonLight_R1.0.csv`
- For option (3), send these files:
  - `PickAndPlace_PCB_AgonLight_R1.0_NoTHT.csv`
  - `BOM_PCB_AgonLight_R1.0_NoTHT.csv`
- For option (2), send these:
  - `PickAndPlace_PCB_AgonLight_R1.0_MINIMAL.csv`
  - `BOM_PCB_AgonLight_R1.0_MINIMAL.csv`



Minimally populated board (option 2), except for the two electrolytic capacitors

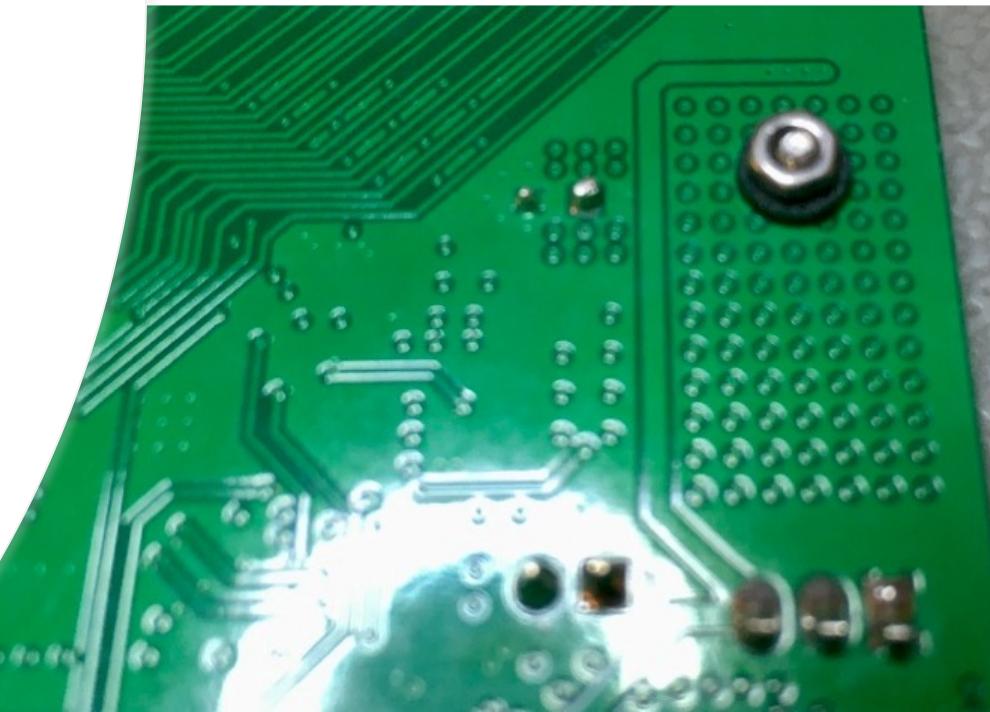
# PCB layer stack

- Agon light's PCB has four layers:
  - Two signal layers (top and bottom)
  - Two inner planes (GND and 3.3V)
- The stack is as follows:
  - *TopLayer* (signals + 3.3V copper flood)
  - *Inner1* (GND plane)
  - *Inner2* (3.3V plane)
  - *BottomLayer* (signals + GND copper flood)
- Agon light has tiny Vias: **0.4mm** diameter with **0.205mm** drill holes, so choose a compatible process with your manufacturer
- I recommend total PCB thickness of **0.8mm**, so to improve signal integrity

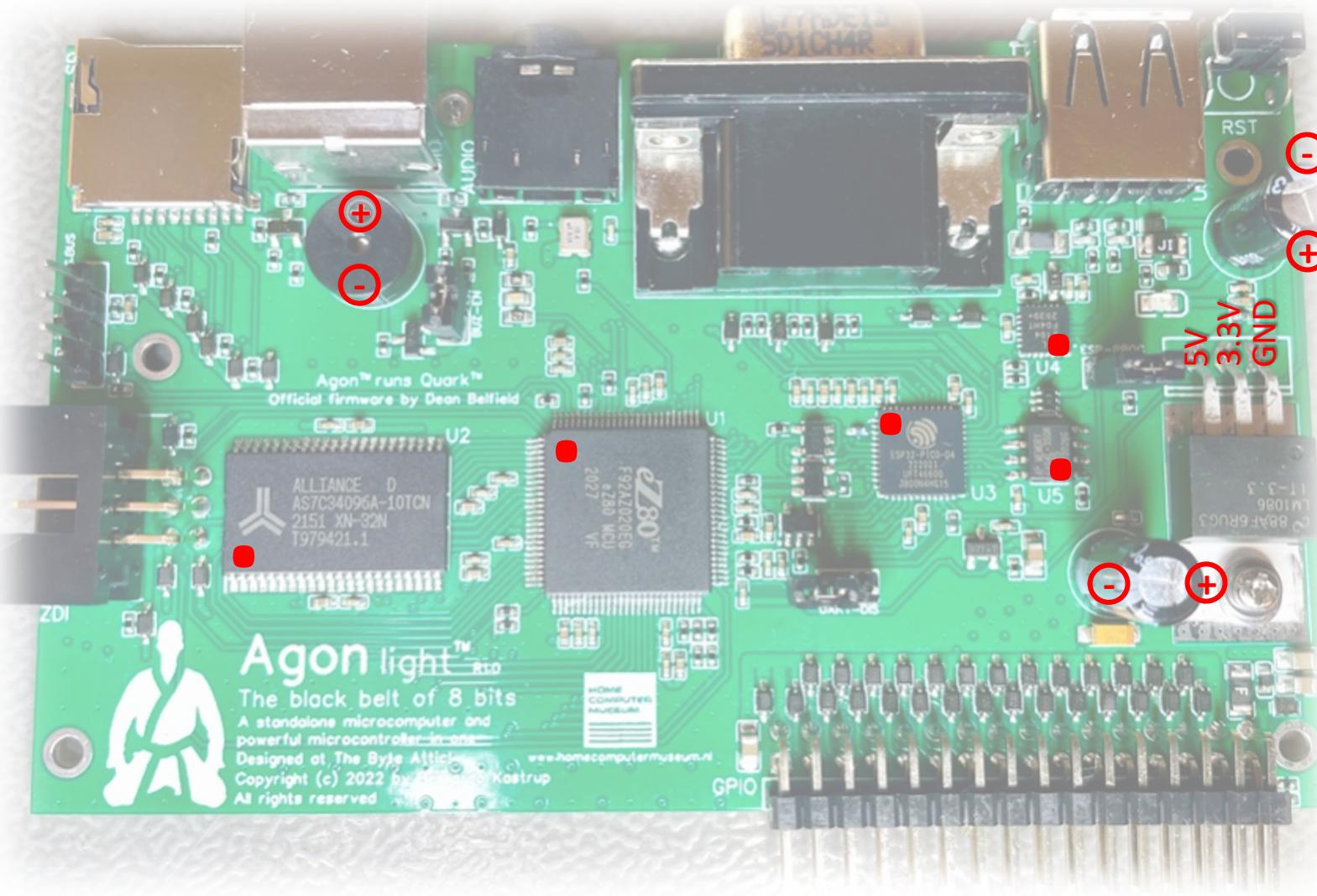


# Mounting the LDO regulator

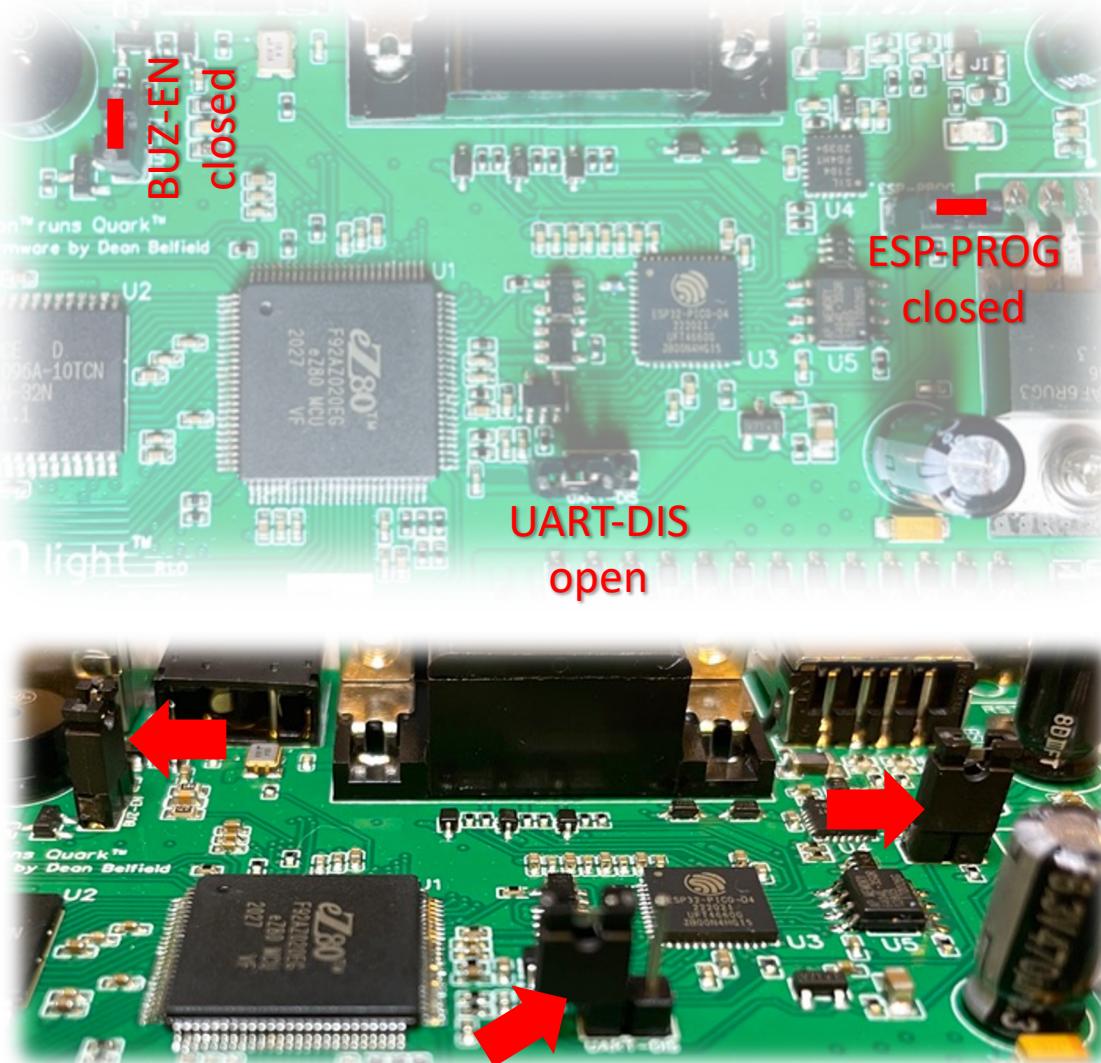
- Agon light's 3.3V  $V_{cc}$  rail is provided by a Low-DropOut (LDO) linear regulator
- The regulator must be mounted flush against the corresponding exposed metal area on the top of the PCB (see top-right photo)
- The regulator's tab (chassis) is at 3.3V, as is the exposed copper area on which it is to be mounted
- Use *no thermal paste or insulating spacers*; simply clean the tab and the exposed metal area with IPA before mounting
- Affix the regulator with a 2mm-diameter bolt, a regular and a lock washer on the top, and a nylon (or other dielectric material) washer and nut at the back (see bottom-right photo)
- The nylon washer is important to insulate the back of the board (which is copper-flooded with GND) from the bolt-nut combination, which will be at 3.3V
  - Using a metal washer on the back side increases the risk of a short-circuit if the solder mask fails



# Part orientations



# Default settings for the jumpers

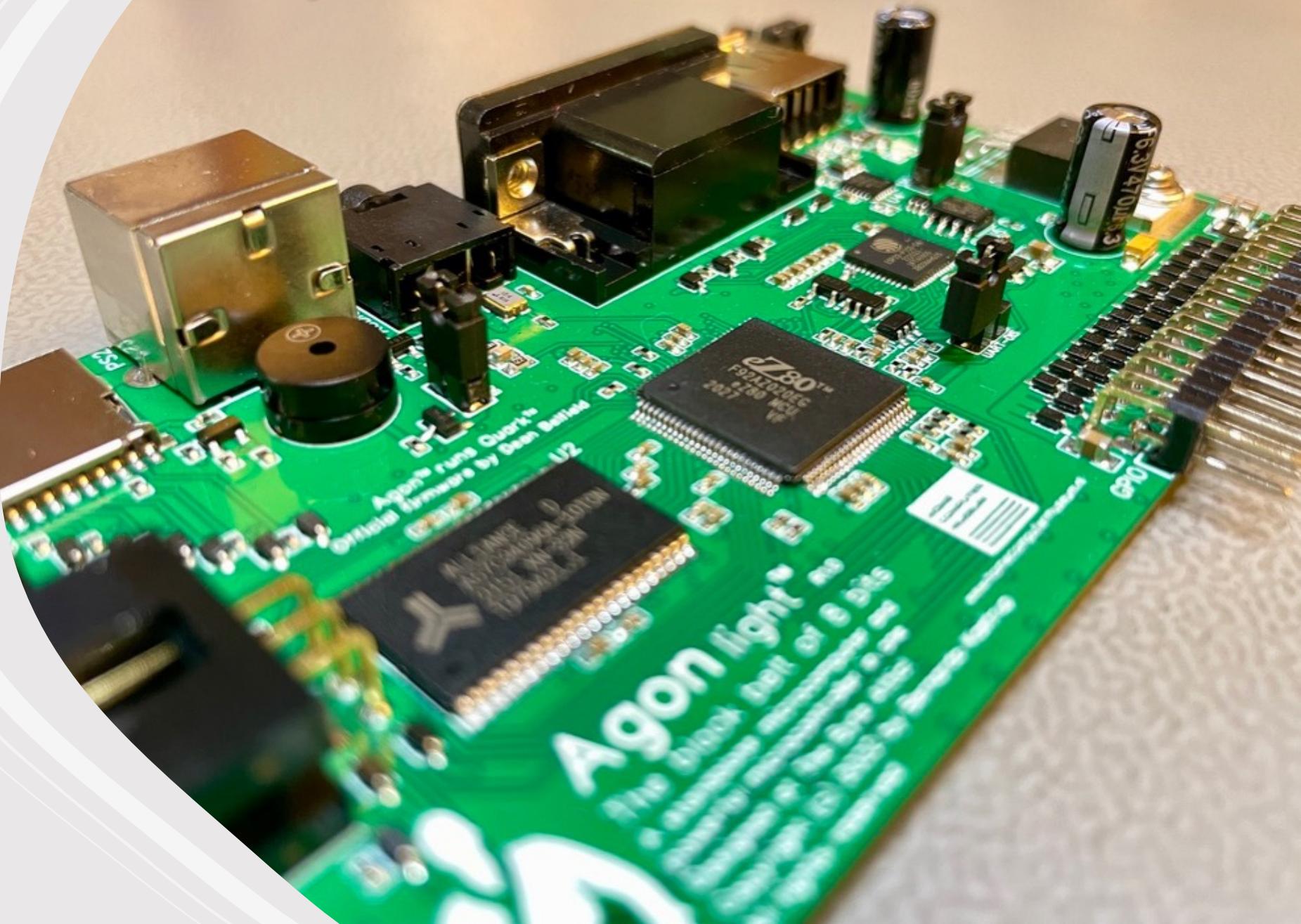


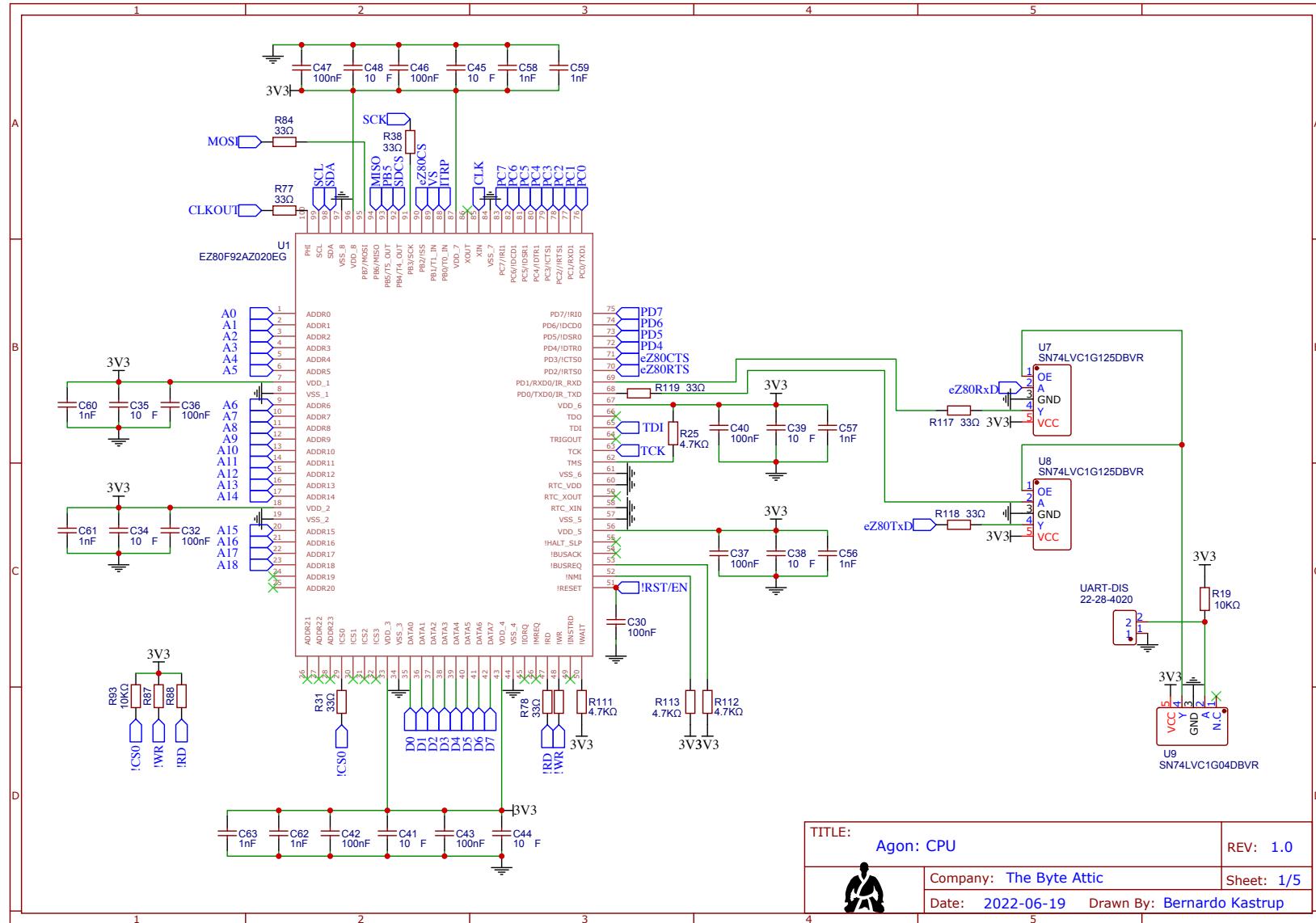
# When to change jumper settings

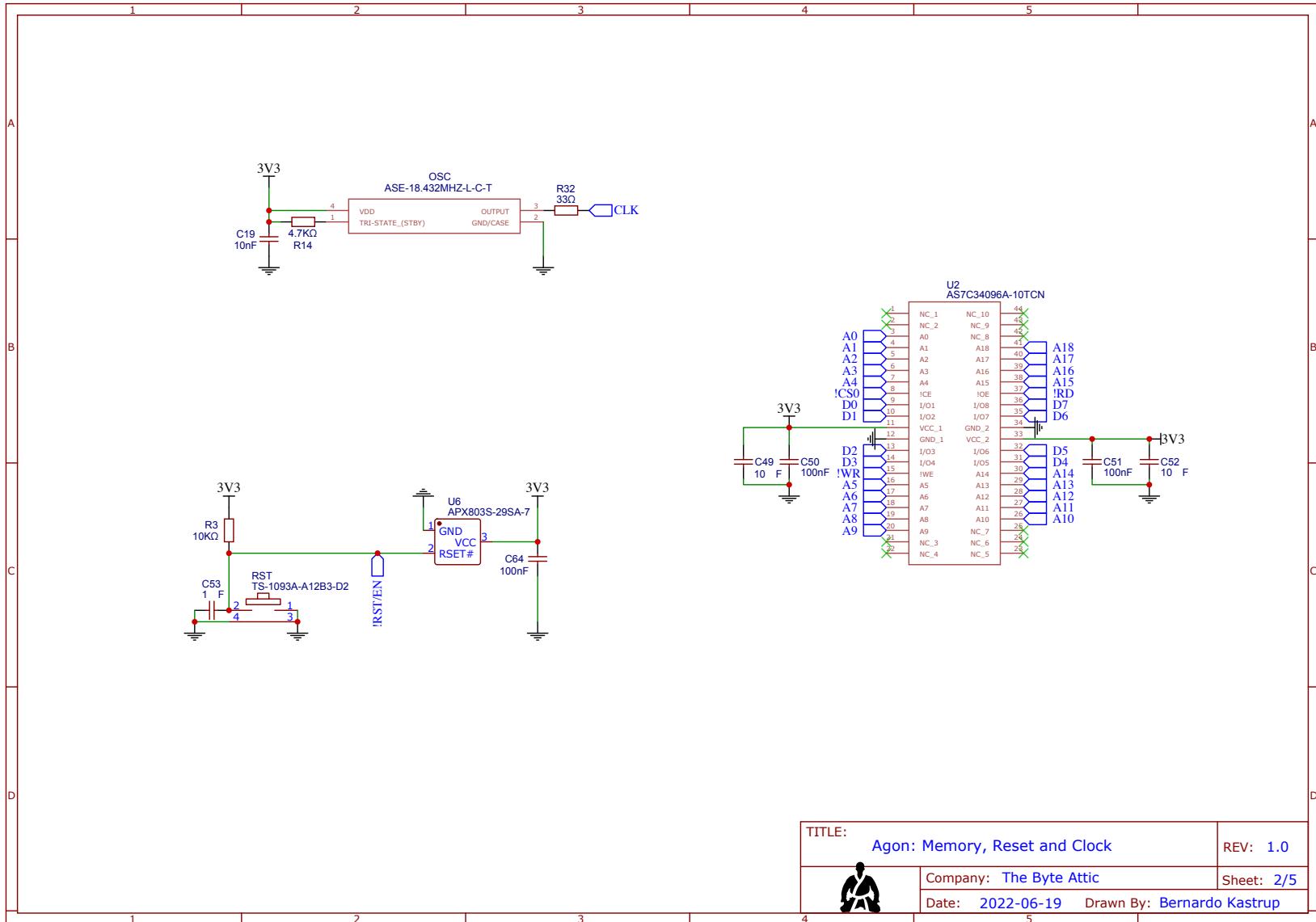
- Agon light should operate normally during both firmware programming and application execution with the default jumper settings (see previous page), but the ESP32 is known to be a sensitive device
  - Therefore, Agon light has built-in resources to deal with that sensitivity
- If the ESP32 goes into programming mode during execution, remove the jumper ‘ESP-PROG’ after programming (remember to place it back before reprogramming the ESP32)
- If you fail to program the ESP32, place the jumper ‘UART-DIS’ during programming (remember to remove it after programming, or Agon light will not operate properly)
- The buzzer produces sounds if speakers are not connected. If those sounds bother you, you can disable the buzzer by removing the jumper ‘BUZ-EN’

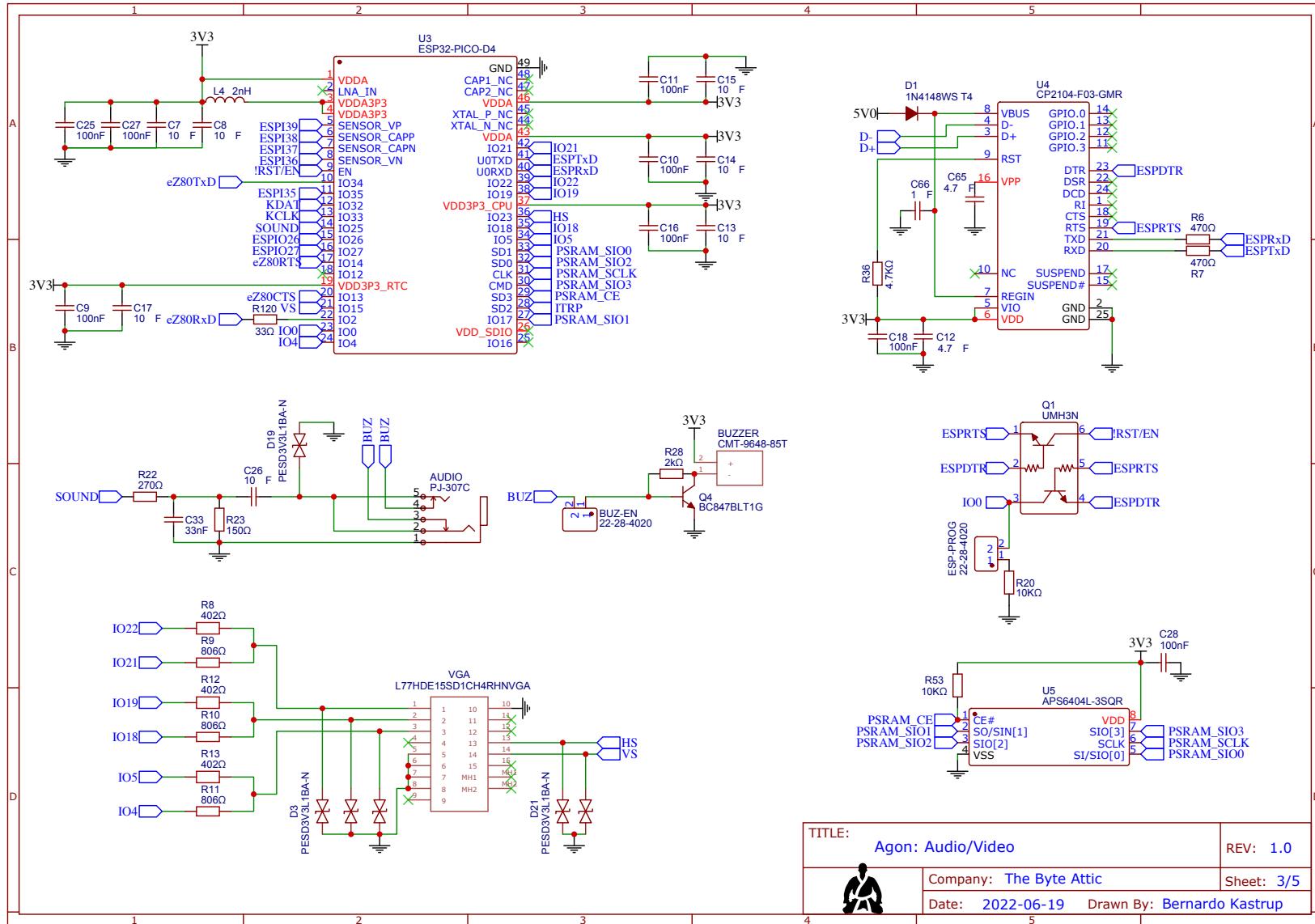


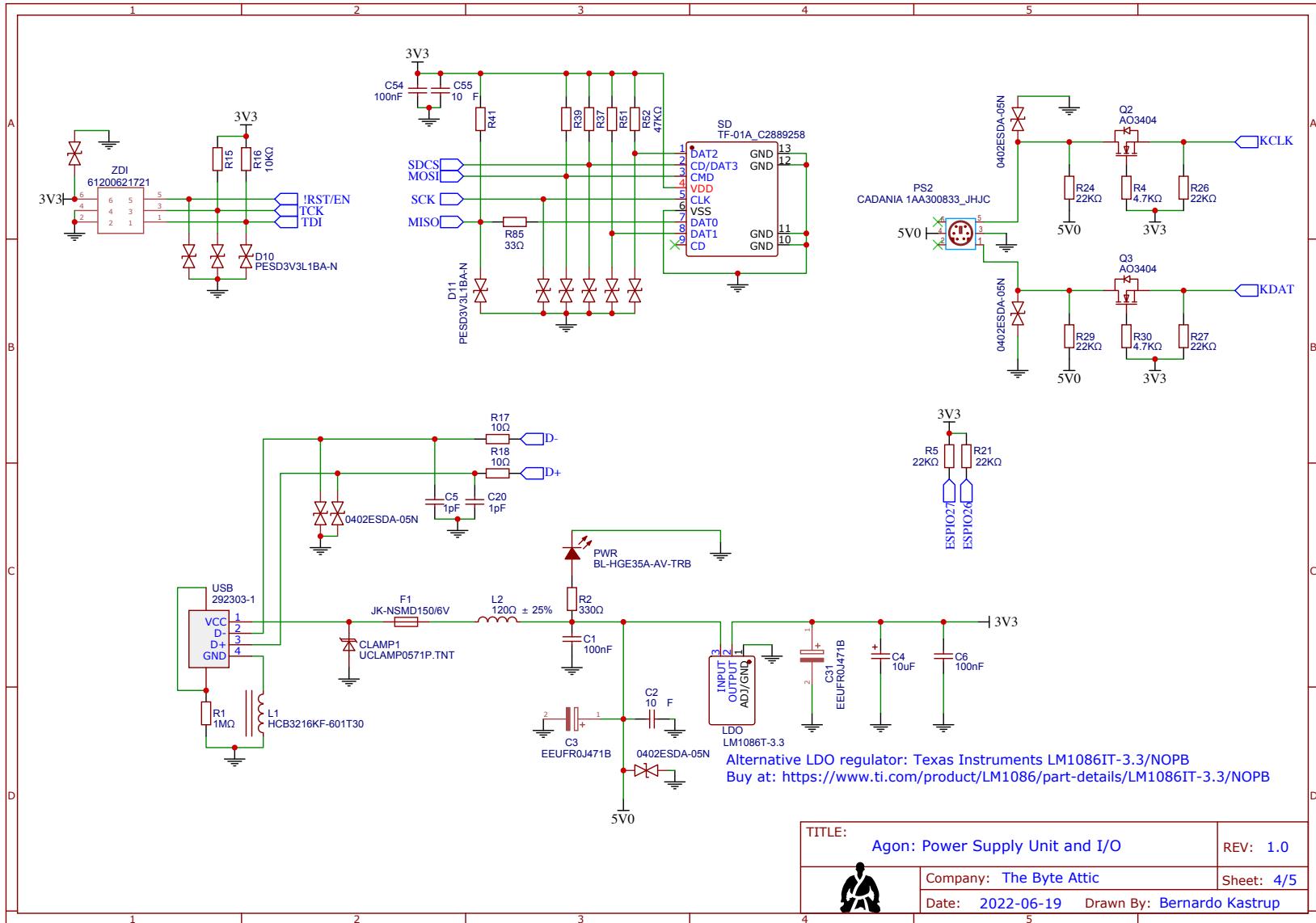
The Byte Attic's  
**Agon**  
light™  
Schematics

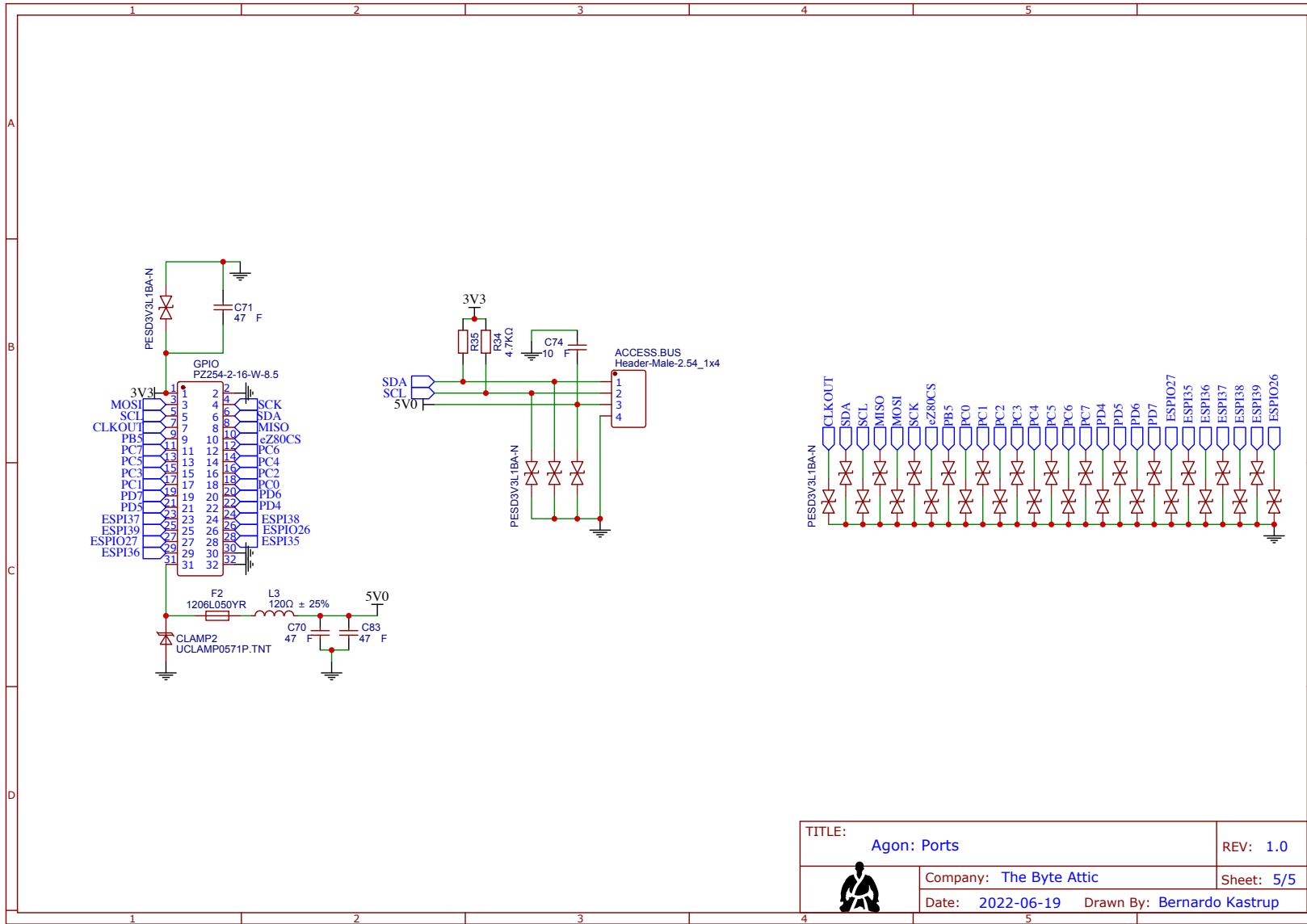










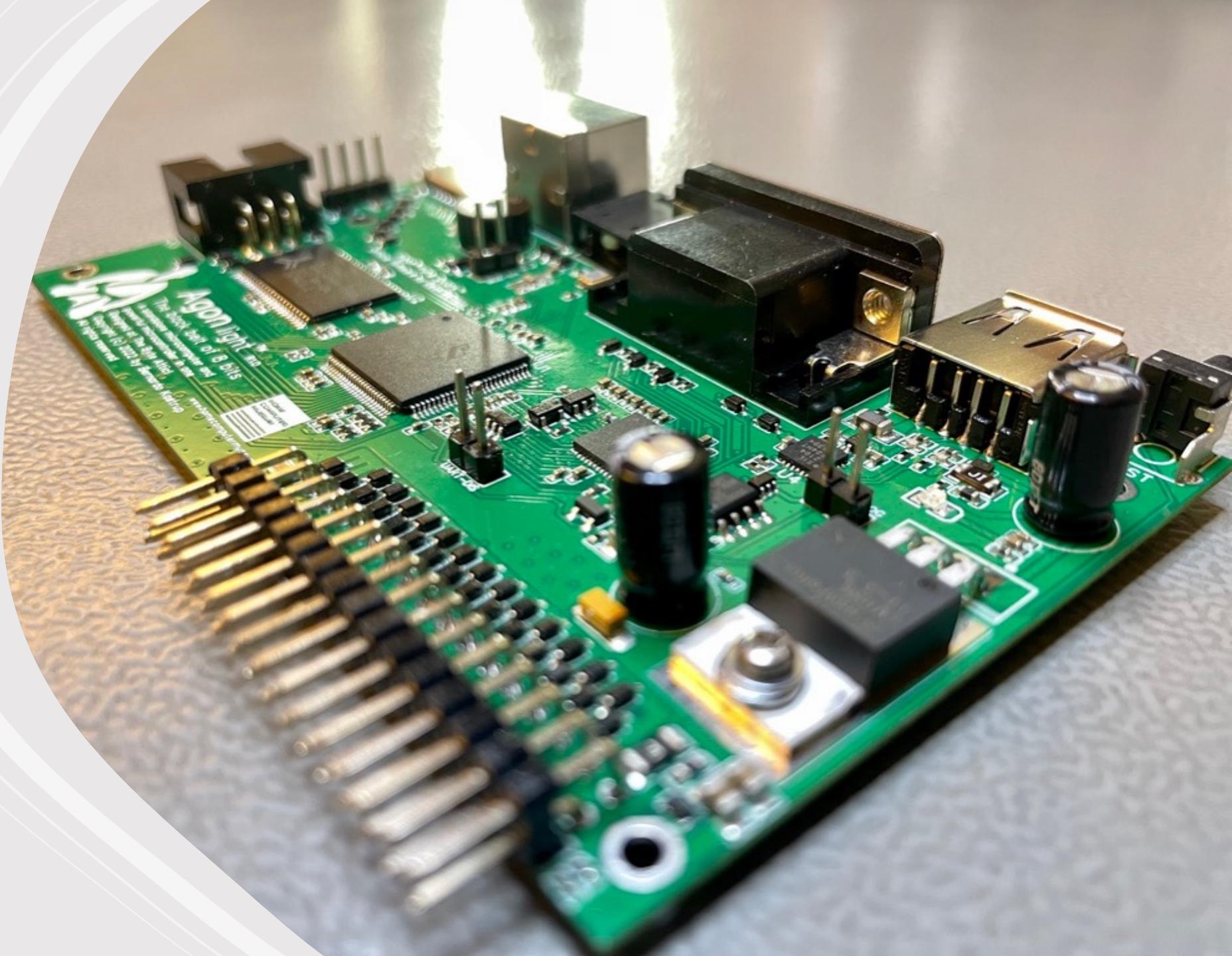




The Byte Attic's

# Agon light™

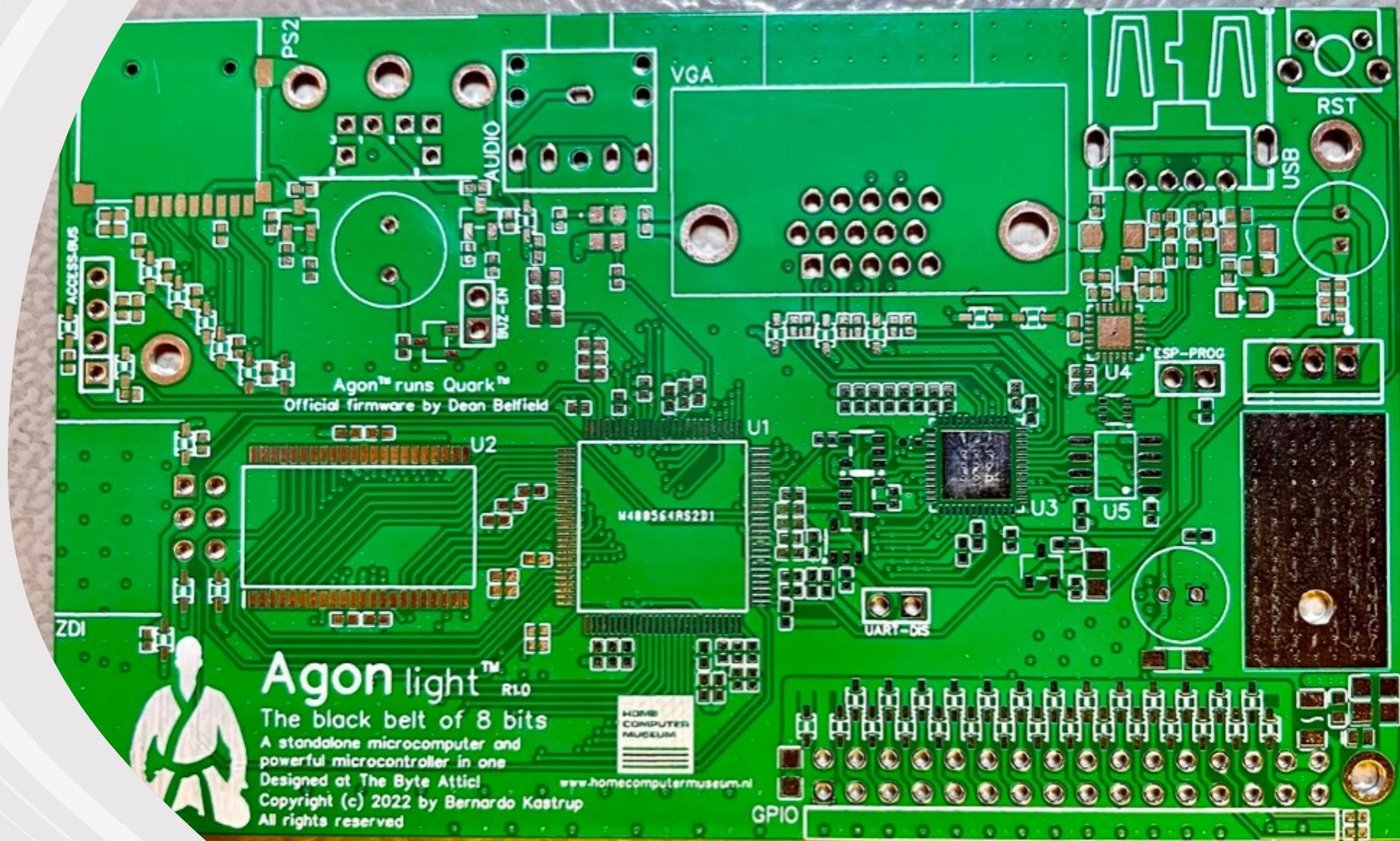
Bill of  
Materials







The Byte Attic's  
**Agon**  
light™  
PCB  
dimensions



# PCB dimensions (diagram *not* to scale)



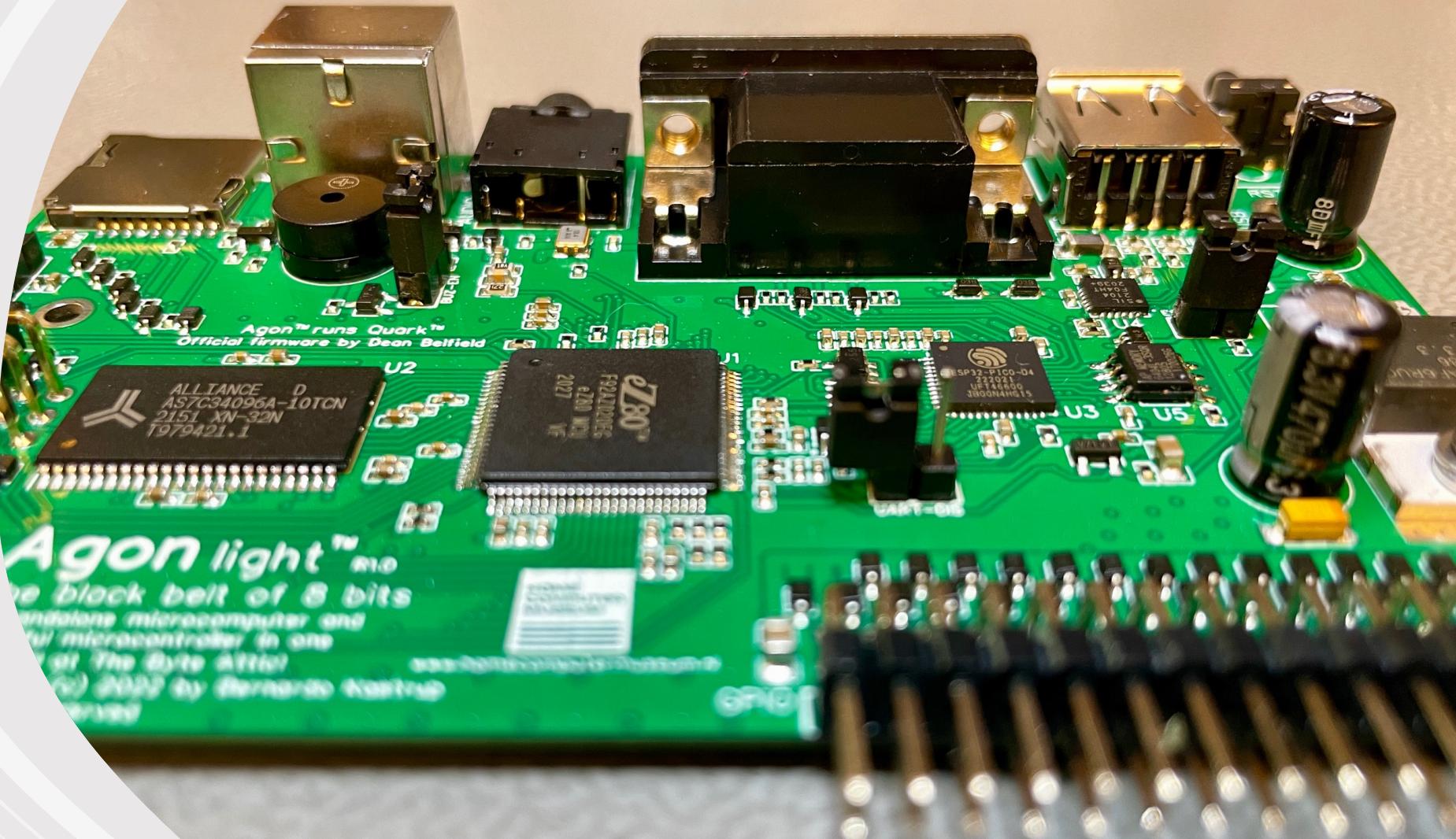
All holes  
are  
2.505mm  
in  
diameter



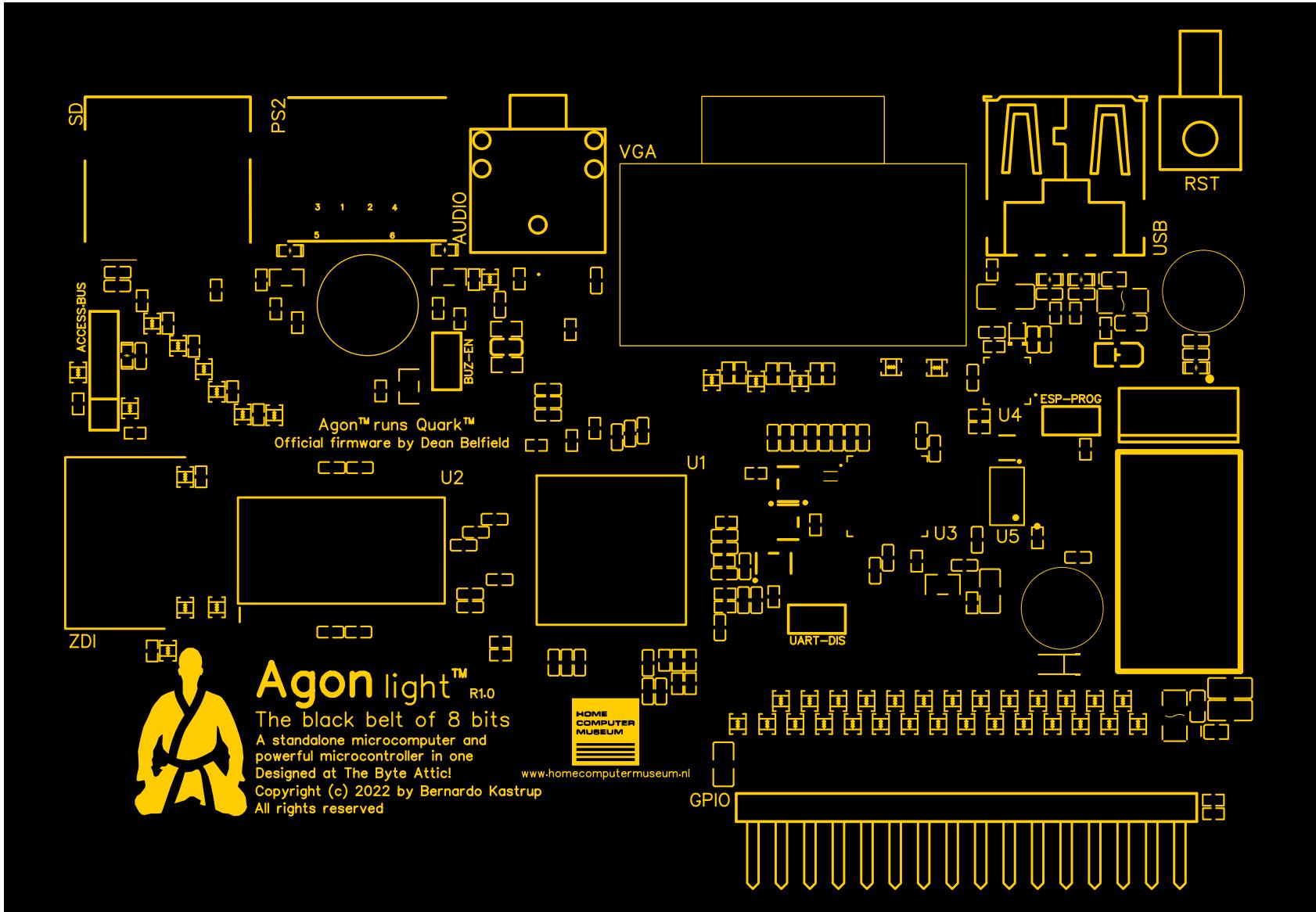
The Byte Attic's

# Agon light™

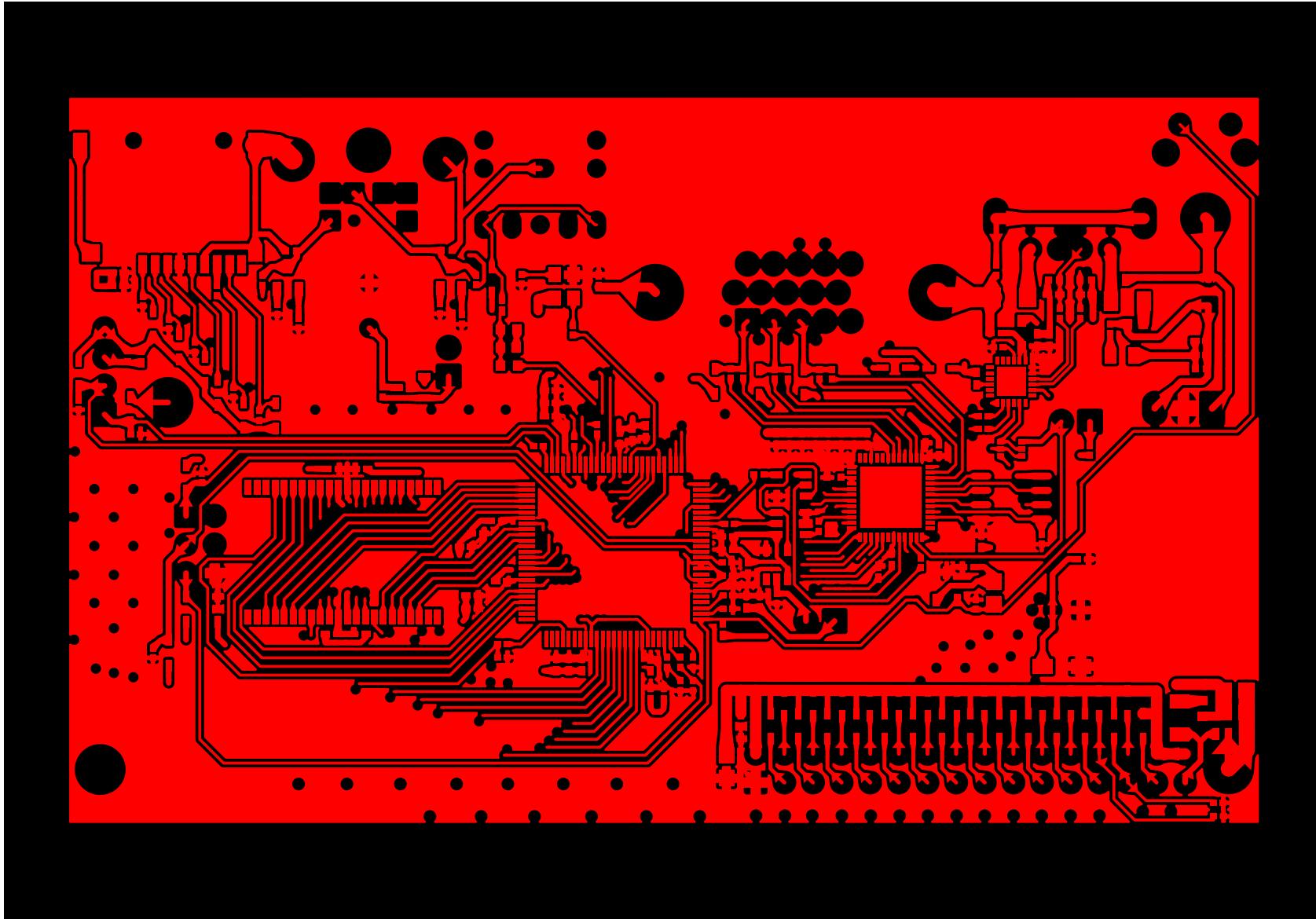
PCB layers  
design



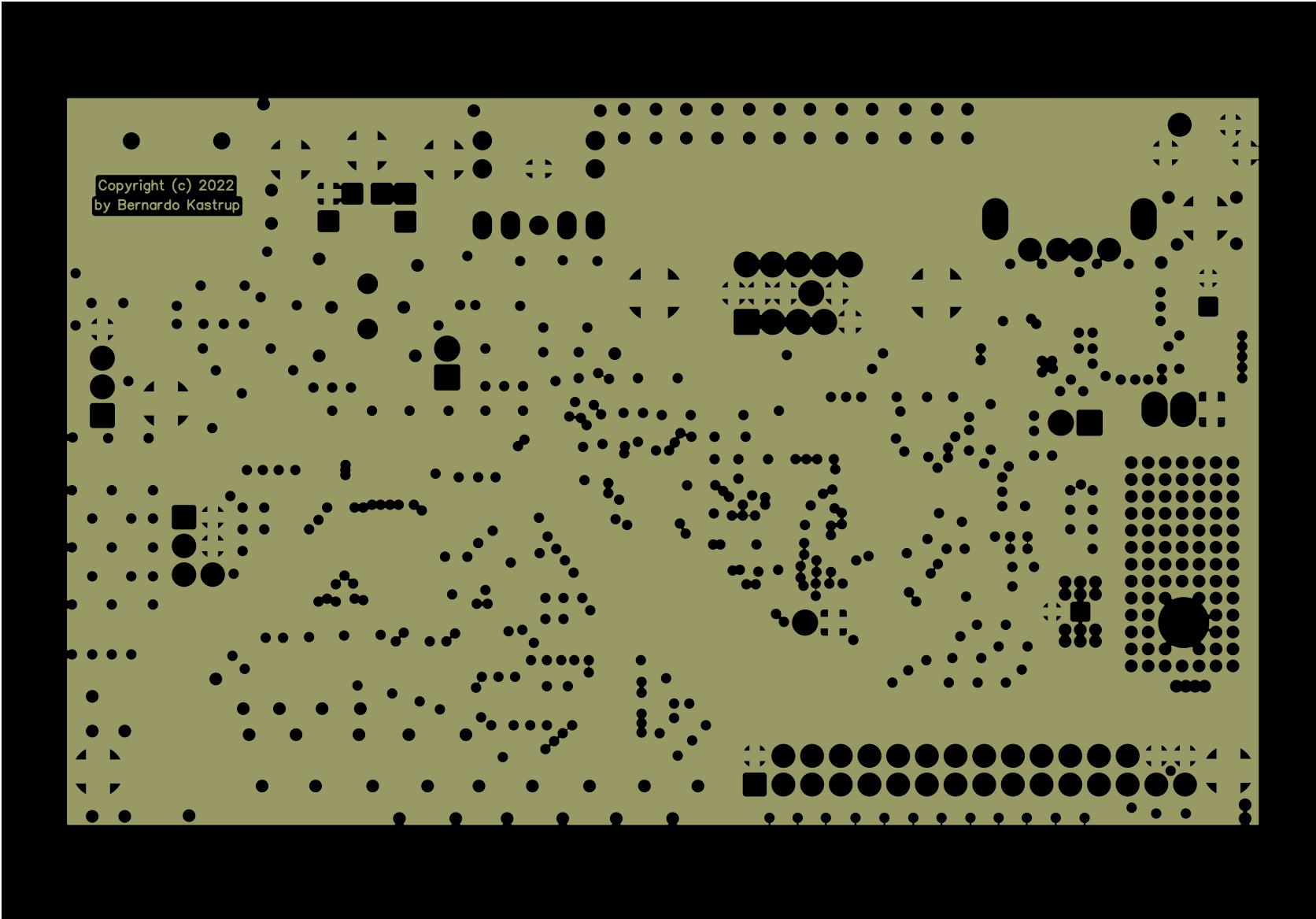
# Top silkscreen



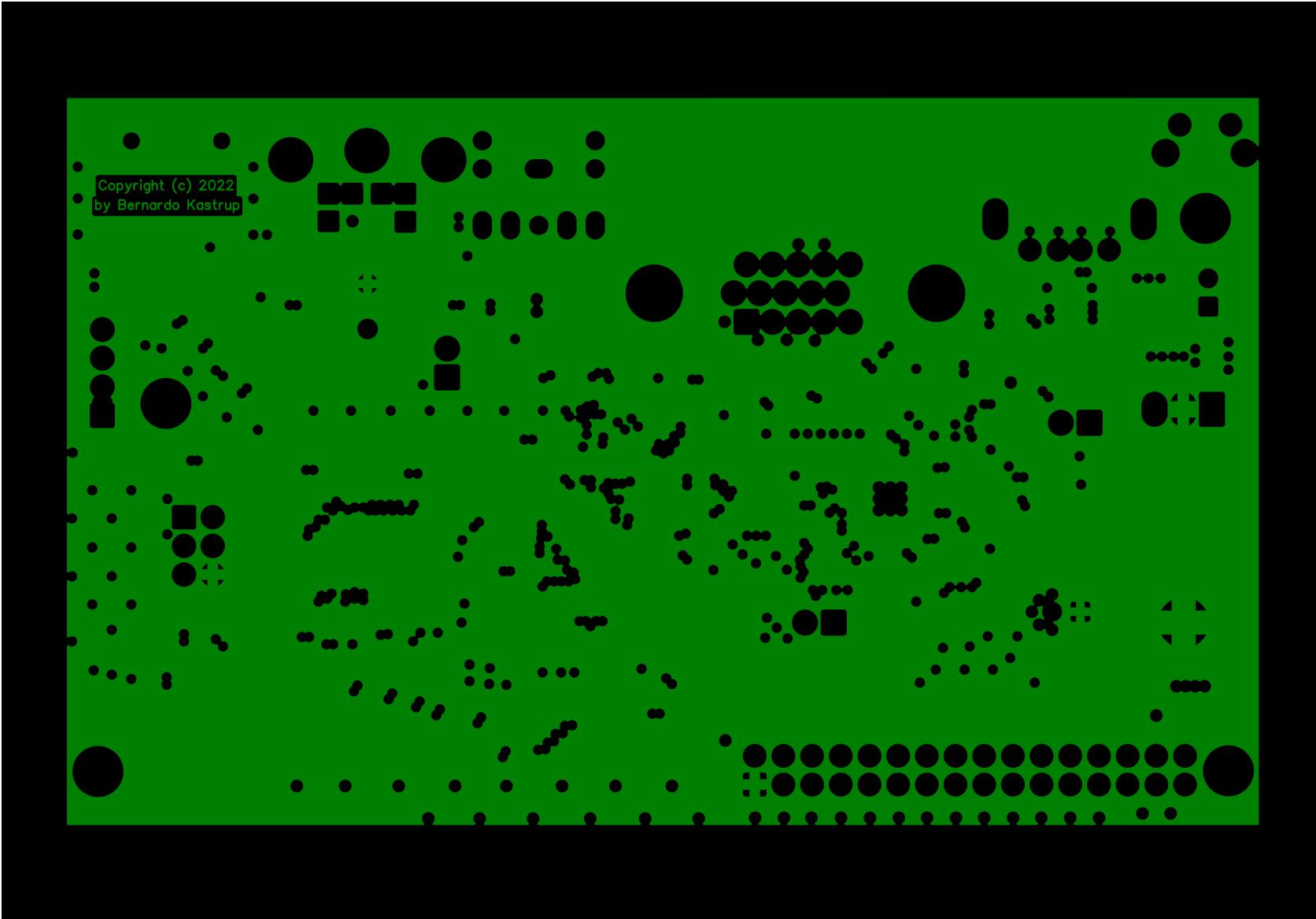
Top metal layer (3.3V filled)



## First inner plane (GND)



## Second inner plane (3.3V)



Bottom metal layer (GND filled)

