A Graphics Terminal with TTGO-VGA32

Martin Hepperle, July 2020

The TTGO VGA-32 is a small board which uses a solder-on ESP32 processor module. This processor is a quite powerful member of the ESP processor family and offers many interesting features including interaction with wireless networks. Similar to the ESP8266, its low cost makes it attractive choice for many hobby development boards. A set of libraries for the Arduino environment makes it easy to use many of its features.

1. Hardware

The TTGO VGA-32 is available in several incarnations, currently in versions V1.2 to V1.4. I used a V1.2 board.

It has interfaces to make a small multi-purpose computer system:

- Standard PS/2 keyboard,
- Standard PS/2 Mouse,
- Standard VGA Monitor,
- Power supply,
- Audio output,
- Serial Interface.

1.1. Serial Connector

The V1.2 board comes with a 4-pin pad arrangement, which accepts a pin header with 2 mm pitch. The sequence of the four pins equals that of many common TTL-RS-232C adapter boards.

Label	3.3V	DIO12	DIO02	GND
Purpose	VCC	RXD	TXD	GND

Table 1: Layout of the header pads on the TTGO-VGA32 board.

The four header pads as well as the FabGL examples (which use GPIO 2 and 12 for serial communication) suggest connecting these pads to a serial interface.

In fact everything worked fine - as long the GPIO pin 12 was connected after the system had been powered up.

However, if it is connected to RXD at boot time, the system won't start. It is not really practical to disconnect and connect the RXD line at each system start.

After finally reading the ESP32 datasheet I learned that the pin GPIO 12 is called MTDI at boot time.

It determines the voltage used to drive the Flash memory (unconnected = Low = 3.3V).

If this pin is connected to the RXD line of the idling TTL-RS232C converter, it is driven high and the Flash voltage is set to 1.8V. This is too low for the 3.3V system and prevents booting.

After soldering a wire to the neighboring GPIO 14 pad of the ESP32 module and connecting it (instead of GPIO 12) to RXD of the TTl-RS232C converter, the system started without any problems.

Conclusion: the pin header seems to be designed for other purposes, not as a serial interface. This is contrary to what the examples in the FabGL library and its layout suggest.

In the sketch I used `Serial` for all communication and replaced the initialization in the `setup()` routine like so:

```
// OLD: GPIO 12 blocks boot process
//  Serial.begin(BAUD_RATE, SERIAL_8N1, 12, 2);

// NEW: for TTGO-VGA32 use GPIO 14 (RX) and 2 (TX)
Serial.begin(BAUD_RATE, SERIAL_8N1, 14, 2);
```

Since then, evcerything worked as expected.

1.2. Power Connectors

Two connectors can be used to provide power:

- a Micro USB connector can be used to plug in a common 5V USB power supply,
- a small two pin connector can be used to connect to a 3,6V Lithium-Polymer battery.

There may be a charging circuit on the board, but you should use this only after careful monitoring the charginging case. There seems to exist an earlier production run of these board where a fuse was mounted instead of a diode, which could lead to overcharging a LiPo battery and subsequent fire.

1.3. Audio Connectors

Two audio connectors are mounted side by side and allow connecting the output signal of the 1W mono audio amplifier NS4150:

- a connector with two pins and a white plastic rim for an internal audio output device like a small loudspeaker,
- a jack for plugging in headphones or other audio systems.

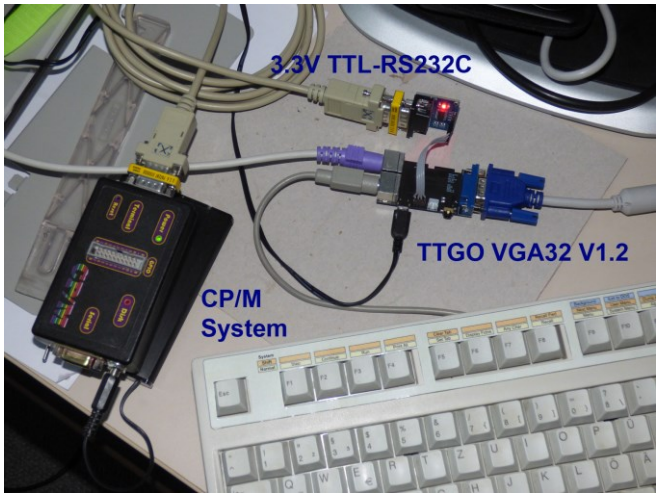

Figure 1 Connecting the power supply, keyboard, mouse, monitor and serial interface is easy.

2. Software

A very interesting piece of software for the ESP32 is the FabGL library which offers many useful functions for video and audio output. This library can be used with the VGA-32 board. One of the examples included with the FabGL library is a Terminal emulator which supports ANSI escape sequences.

I have taken this example and extended it to add Tektronix control sequences for monochrome vector graphics capability.

First I added an audible BELL because the FabGL library already has audio capabilities. In order to make error beeps audible I added code to recognize the ^G (BELL, 0x07) control character.

Secondly, I added an interpreter for Tektronix control sequences for monochrome vector graphics capability. For this purpose the Terminal class was sub-classed into a TekTerminal class and a handful of the private methods of the Terminal class had to be declared virtual to allow overriding them in the TekTerminal class. The result is a mixed emulation of an ANSI terminal which can be switched into Tektronix emulation mode.

To switch into Tektronix emulation the escape sequence

```
ESC [ ? 38 h
```

has to be used. This is a sequence also use by xterm to open its Tektronix window.

Switching back from Tektronix mode to ANSI mode is accomplished by the sequence

```
ESC ETX
```

Notes:

- Different line types (1…5) are translated into different intensity levels, e.g. 5 produces a faint green line.
- Graphical input GIN mode is not supported.
- The available resolution of 640 x 350 pixels is considerably coarser than that of a real Tektronix terminal. Therefore fine details will be lost.

2.1. Using a German Keyboard Layout

For using my German keyboard I had to switch to the desired layout by inserting a line after the PS2Controller has been started:

```
[...]
  PS2Controller.begin(PS2Preset::KeyboardPort0);
  /* switch layout to German */
#ifdef GERMAN
  PS2Controller.keyboard()->setLayout(&fabgl::GermanLayout);
#endif
[...]
```

There are still some smaller inconsistencies in this layout, e.g. with the numeric keypad and some extra keys, but it is very useable.

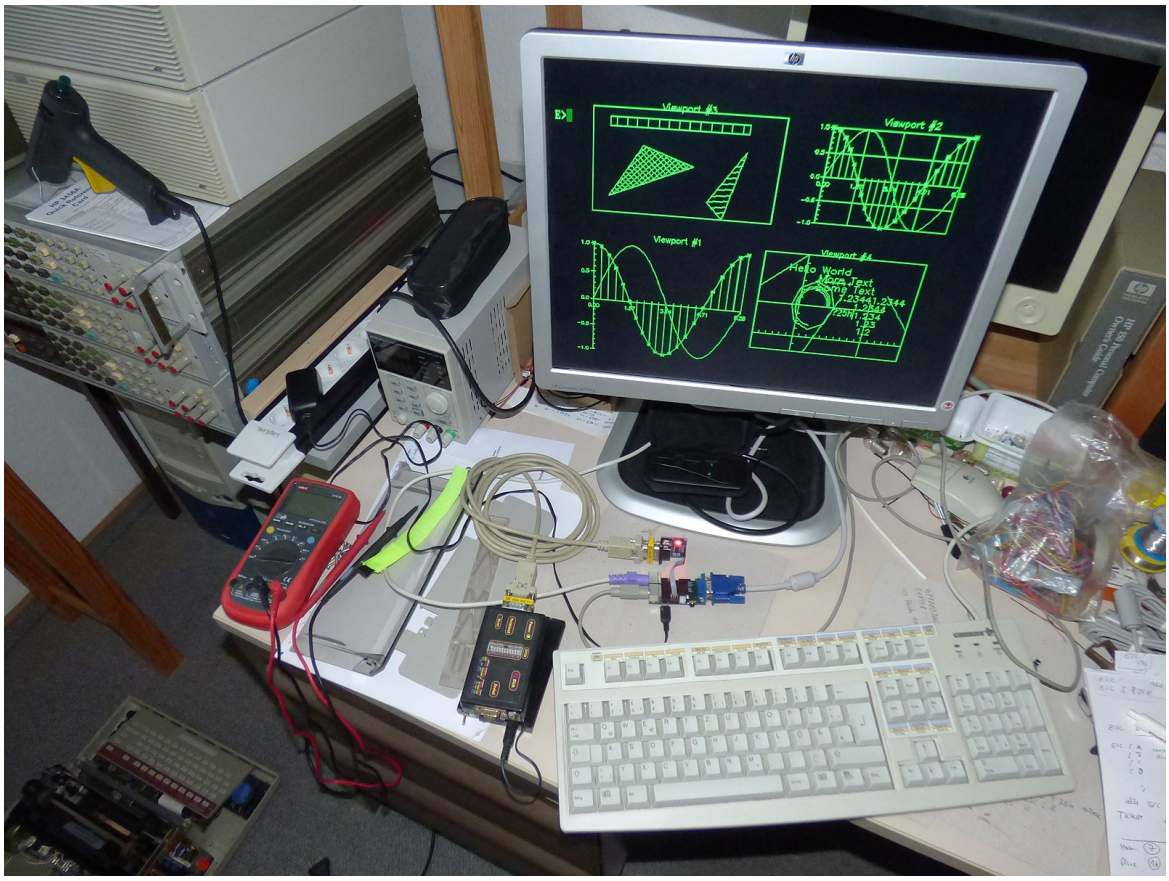

Figure 2 The terminal in action with one of my CP/M-80 GSX test programs.

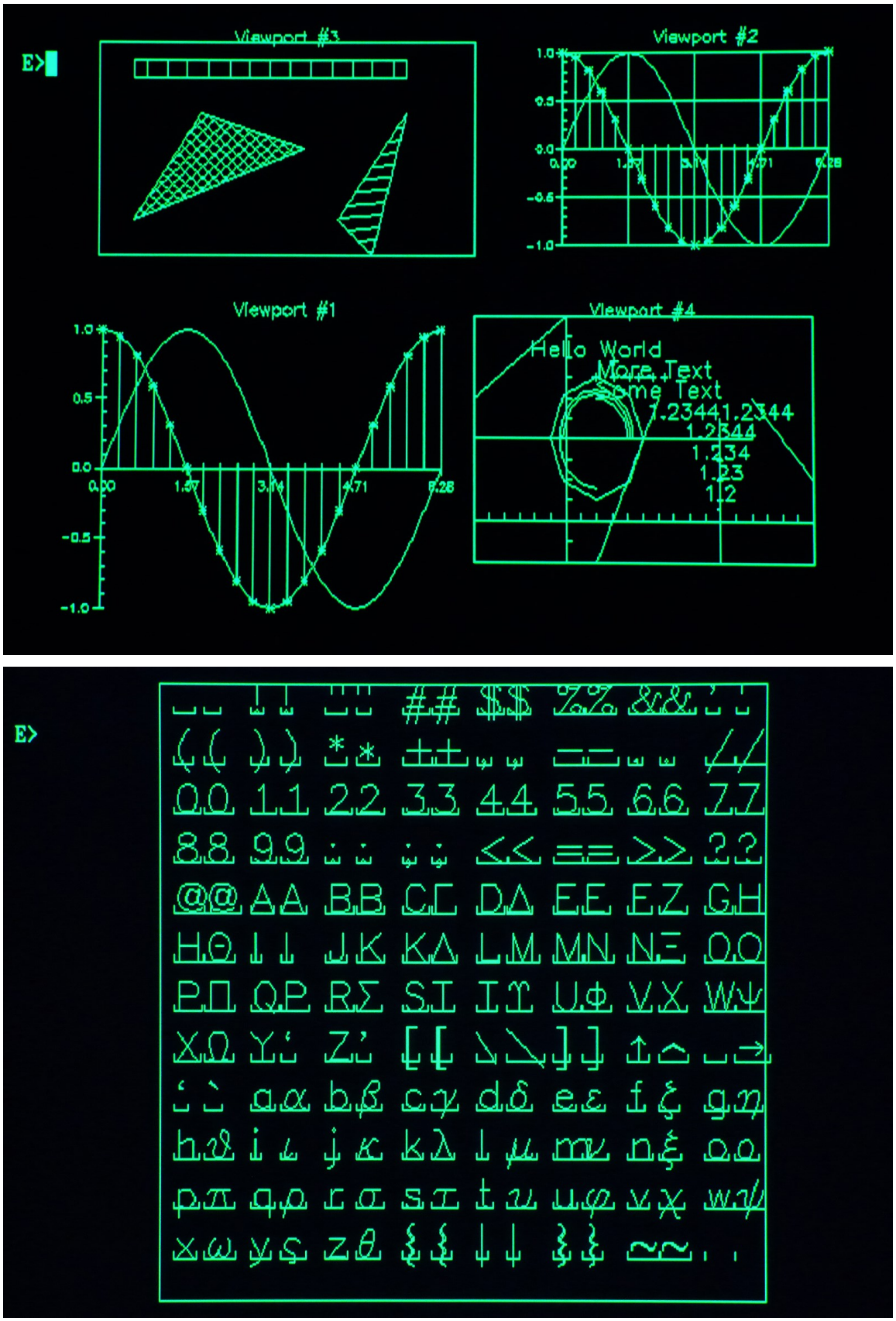

Figure 3 **Close-up view of the output in 640 x 350 VGA mode on a TFT screen.**

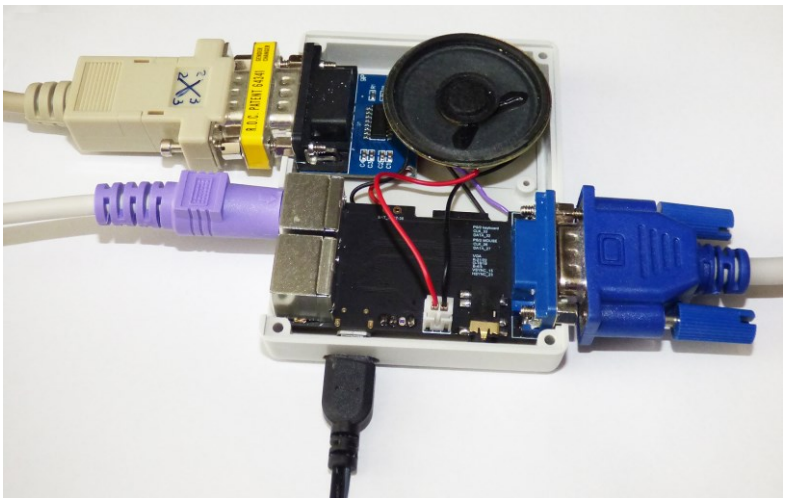

Figure 4 **The complete terminal with VGA32 board, serial interface and loudspeaker.**

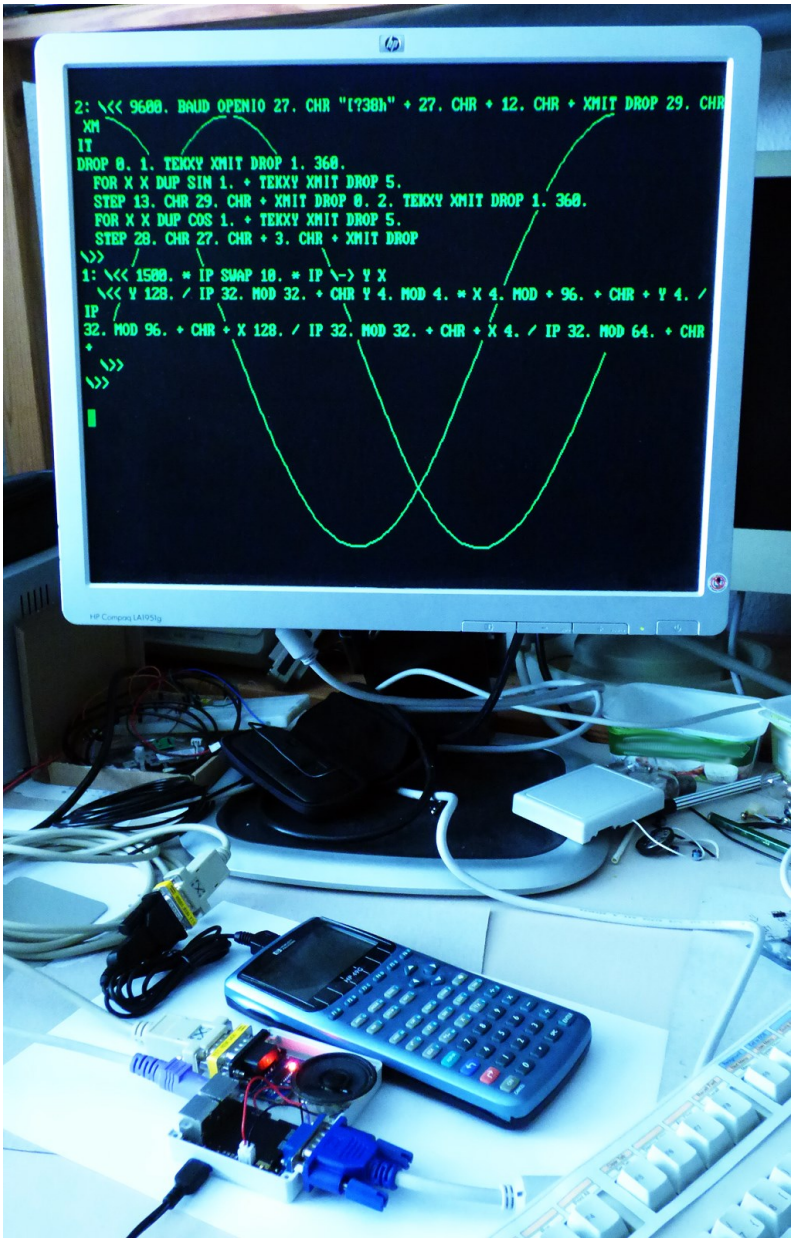

Figure 5 **An alternative test setup with a HP-49G calculator producing graphics output in Tektronix format.**

2.2. Boards and Pins

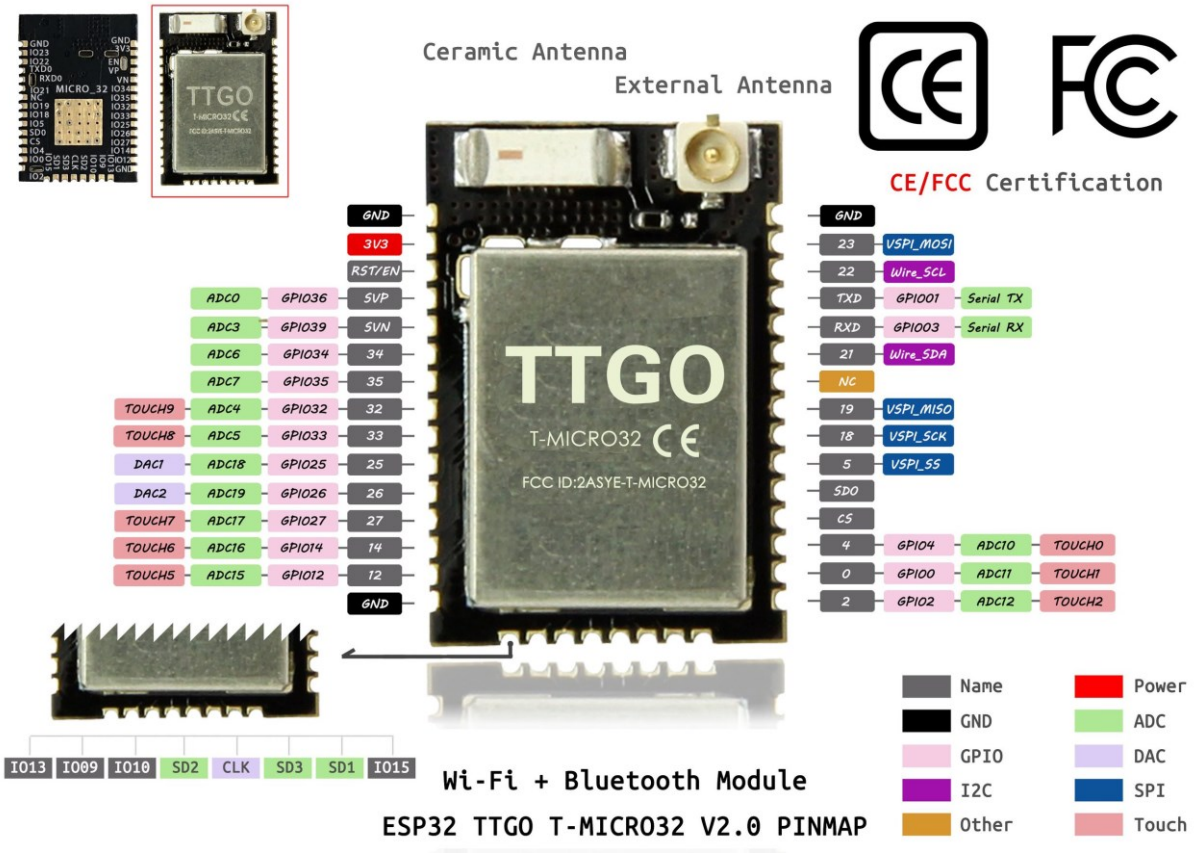

Figure 6 **This ESP32 module is soldered onto the TTGO VGA32 board. I soldered a wire to GPIO14 for RXD of the TTL-RS-232C converter, instead of using the GPIO12 on the 4-pin header.**

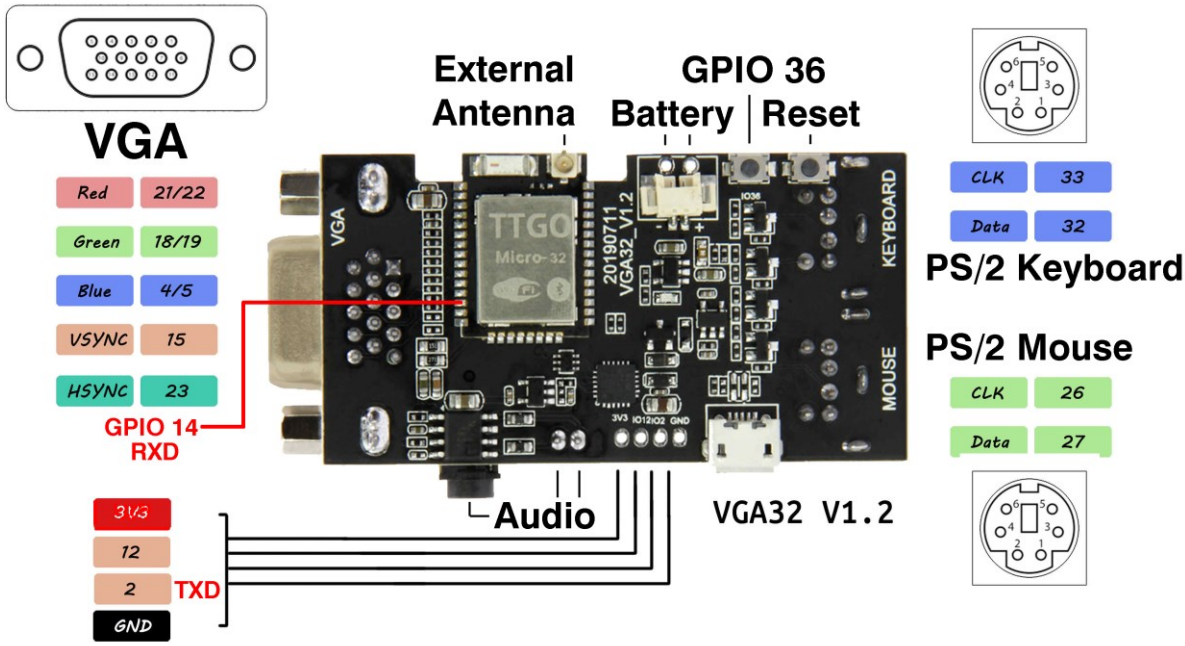

Figure 7 **TTGO VGA32 Version 1.2 board with GPIO 2 and 14 for the serial interface.**

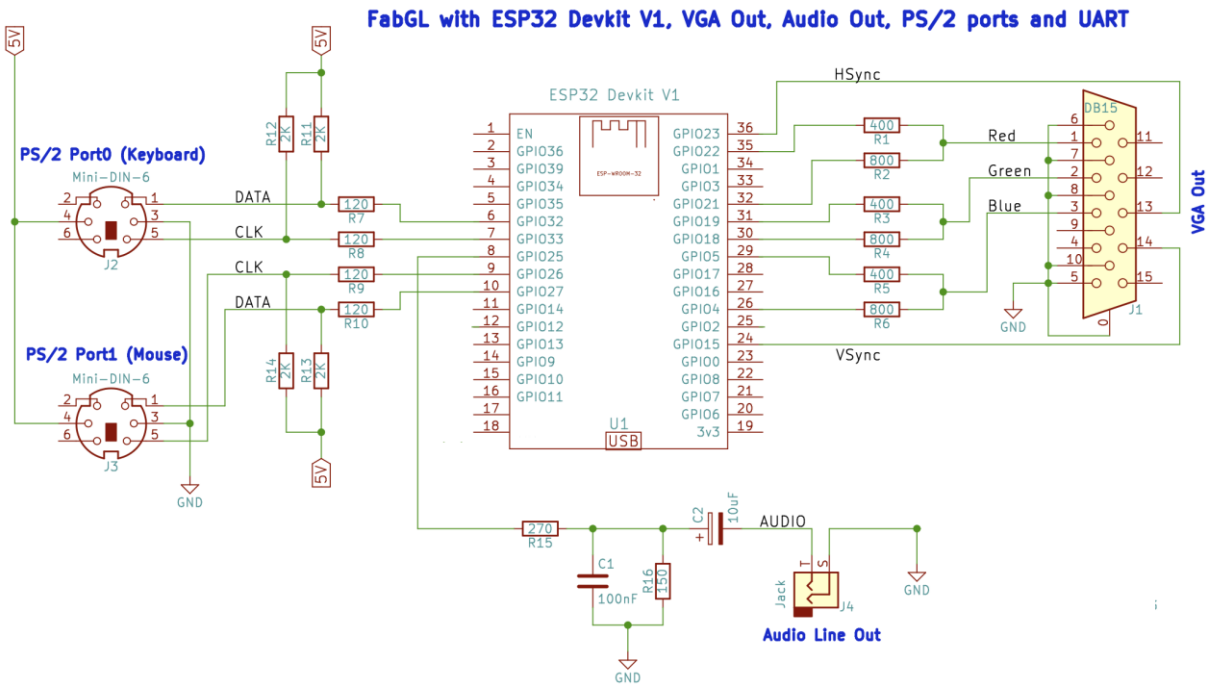

Figure 8 **Wiring as proposed by the author of FabGL. the TTGO VGA32 Version 1.2 board follows this scheme and adds USB, audio amplifier and battery charging circuits.**

GPIO pin	Connection	GPIO pin	Connection
2	4 pin header (TX)	23	VGA Hsync
4	VGA Blue	25	Audio
5	VGA Blue	26	Mouse Data
12	4 pin header	27	Mouse Clock
13	*V 1.4: to header*	32	Keyboard Data
14	*V 1.4: to header (RX)*	33	Keyboard Clock
15	VGA Vsync	*34*	*V 1.4: to header*
18	VGA Green	36	Button 2
19	VGA Green	*39*	*V 1.4: to header*
21	VGA Red		
22	VGA Red		

Table 2: **GPIO pins used on the TTGO-VGA32 boards.**

(The board V1.4 brings four additional GPIO pins to a hole pattern and moves the second button to a different place).

The two USB data lines go to the SIL 2104 F03LL chip.

2.3. Buttons

Button 1 (close to the keyboard connector) on the TTGO VGA32 Board connects GND to RST and can be pressed to reset the system.

Button 2 (close to the battery connector) pulls GPIO 36 to GND and may be used for user code. It could be used to clear the Tektronix screen.

2.4. A simple HP 49G Demo Program

Even with a pocket calculator you can generate output on a Tektronix terminal.

Simple Main Demo Program

Plots a sine and a cosine curve. The sine curve is plotted with additional drop lines to the x-axis.

```
'TEKDEMO'
[-] -> [-]     (no effect)
```

```
<<
 TIME HMS-> 9600. BAUD OPENIO
 27. CHR "[?38h" + 27. CHR + 12. CHR + XMIT DROP
 DEG
 0.1. TEKMOVE
 1. 360. FOR X
   X DUP SIN 1. + DUP2 TEKDRAW
   X 1. TEKDRAW
   TEKDRAW
 5. STEP

 0. 2. TEKMOVE
 1. 360. FOR X
   X DUP COS 1. + TEKDRAW
 5. STEP

 28. CHR 27. CHR + 3. CHR + 7. CHR + XMIT DROP
 TIME HMS-> SWAP - ->HMS
>>
```

Move Subroutine

Moves the pen to the given point. First switches to ALPHA mode, moves the pen to the X Y point and then starts PLOT mode for subsequent calls of the TEKDRAW routine.

```
'TEKMOVE'
[X Y] -> [-]     (leaves empty stack)
```

```
<<
 TEKXY 28. CHR SWAP + 29. CHR + XMIT DROP
>>
```

Draw Subroutine

Draws a line from the current point to the given point.

```
'TEKDRAW'
[X Y] -> [-]     (leaves empty stack)
```

```
<<
 TEKXY XMIT DROP
>>
```

Scaling and Conversion to 5-char Tektronix String Subroutine

Scales data from X=[0…360], Y=[0…2] to the [0…4095] viewport.

```
'TEKXY'
[X Y] -> ["12345"]     (leaves 5 character string for output to Tektronix)
```

```
<<
 1500. * SWAP 10. *
 2. PICK 128. / IP 32. MOD 32. + CHR
 PICK3 4. MOD 4. * PICK3 4. MOD + 96. + CHR +
 PICK3 4. / IP 32. MOD 96. + CHR +
 2. PICK 128. / IP 32. MOD 32. + CHR +
 SWAP 4. / IP 32. MOD 64. + CHR +
 SWAP DROP
>>
```