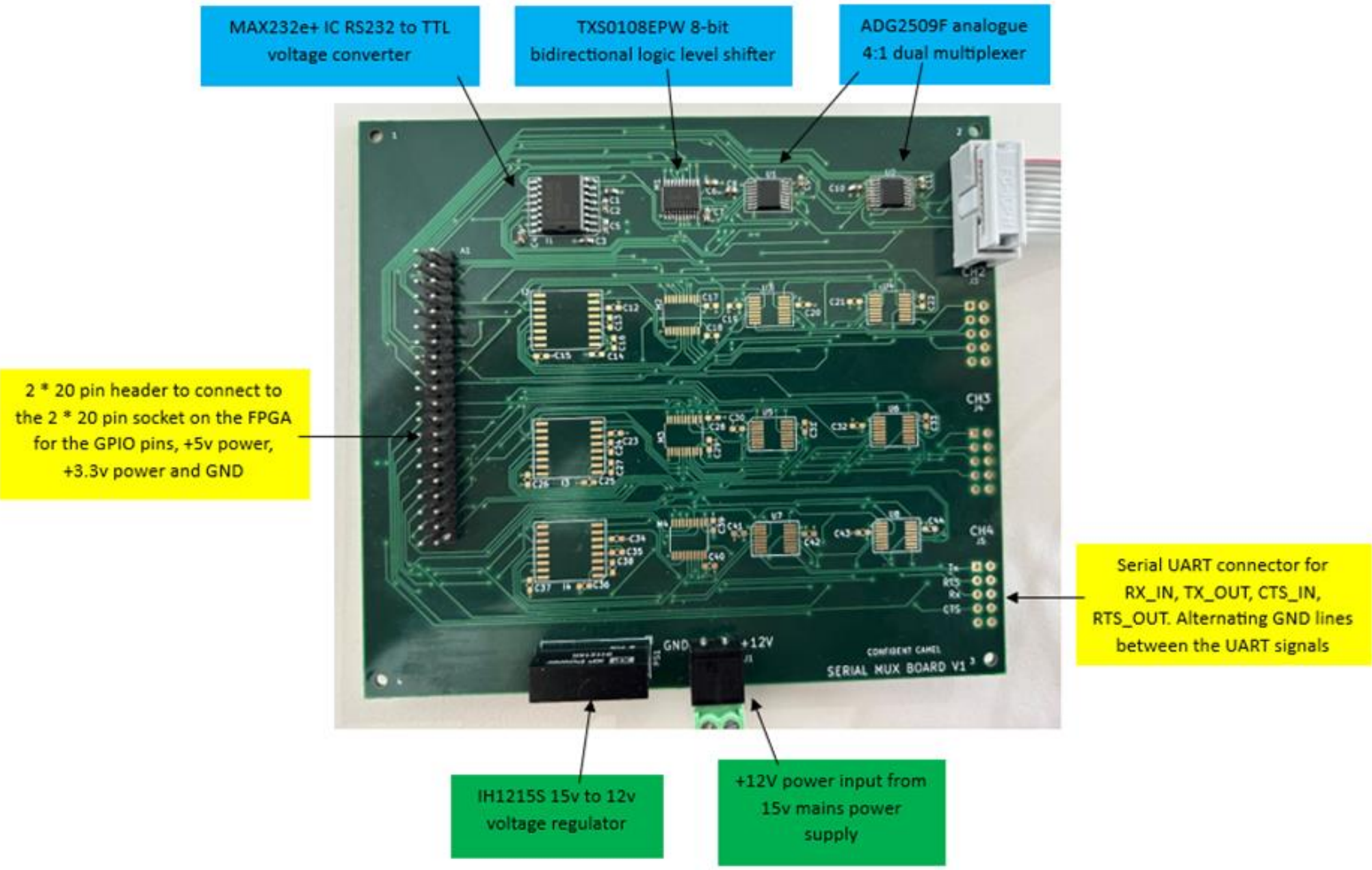


TCP to UART protocol converter for between RS232 serial and 5.0v and 3.3v TTL

Freya Jaleh Vafadari
DSTL Porton Down

Project overview

My final year project is to design, build and test a TCP to serial data protocol converter device for RS – 232 to 0 3.3v/5v TTL. This is to enable a series of devices under test that take serial data as an input in RS 232 form and convert it into TCP data by changing the voltages to either 3.3v or 5v TTL or RS 232/12v. This is a requirement of a wider project in the organisation which needs to concurrently connect multiple serial devices, which don't have Ethernet connections, to an IP network so serial data can be generated and received across the network through a TCP connection.



Manufactured PCB hardware with major electronic components and off board connectors labelled.

Successes

I successfully developed an embedded Linux device tree and kernel for the UART channel interface using Debian specifically the Bookworm version as I had only previously done embedded Linux development using Yocto project.

Challenges

It was challenging to develop the embedded Linux device tree and kernel initially because I used the Bullseye Debian release however, the release wasn't compatible with the latest release of the Linux kernel and Python and so it was unable to build the kernel until the Debian version was switched to Bookworm. It was also a learning curve to understand how the Python 'select' library works and how to implement it to monitor the availability of all serial and TCP ports and when data can be received or transmitted across the bridge.

Future improvements

Currently, the tcp_address parameter is bound to all interfaces with a fixed TCP address of 0.0.0.0, however in the future this could be modified if required so the user can configure the TCP address themselves.

Another potential improvement that could be made is there is an additional option to send and receive data in UDP form not just TCP data across the bridge between the TCP clients on the server side and the serial ports. UDP is generally quicker than TCP because there's no requirement to establish and maintain a connection because UDP is a 'send and forget' protocol which means data is sent without waiting for acknowledgements. The TCP protocol could be implemented initially to configure the serialX_uart channel parameters and the use of control messages for the acknowledgements of sending and receiving the serial or TCP data then switch to sending and receiving data using the UDP protocol when the data is being transmitted continuously.



Hardware and software development

The hardware development involved designing using KiCAD and having the PCBs manufactured by an external company in industry. The circuit shows the voltage and protocol conversion of RS-232 to TTL voltages either 3.3v or 5.0v between serial UART and TCP protocols. The connectors on each end of the circuit are DB9 serial connector on the RS232 side and an eight-pin header on the TTL connector on the TTL side for the serial UART signals: TX_OUT, RX_IN, RTS_OUT, CTS_IN with alternating GND connections. There are four UART channels per PCB each with a single MAX232E+ integrated voltage converter (I1, I2, I3 & I4) where each MAX3232E+ IC connects to an RS232 side and a TTL side to invert and shift RS232 voltages (±12v) to TTL voltages.

The MAX232E+ integrated circuit component is designed to change voltage levels between TTL logic (0 – 5v & 0 – 3.3v) and the RS-232 serial standard. The MAX232E+ integrated circuit is designed to translate voltage levels between TTL logic (0–5v) and the RS-232 standard (typically ±10v to ±12v). However, the IC itself operates from a single 5v power supply, which is insufficient to directly generate the necessary RS-232 voltage levels. To solve this problem, the MAX232E+ integrated circuit component utilises an internal charge pump circuit that uses external capacitors to generate the higher positive and negative voltages needed. The charge pump works in two main phases which are voltage doubling and voltage inversion. Initially, the first one set of capacitors doubles the supply voltage, forming an intermediate voltage of approximately +10v. Then, another set of capacitors inverts this voltage, generating approximately –10v, which is important because RS-232 signalling requires both positive and negative voltages relative to ground. [1]

In the RS-232 to TTL level shifting circuit an ADG5209F analogue multiplexer per UART channel is incorporated and several functions. The ADG5209F is a high-voltage, fault-protected, dual 4:1 analogue multiplexer, meaning it internally contains two independent multiplexers, each capable of routing one of four inputs to a single output under digital control. Within this context of RS-232 to TTL conversion, the multiplexer is connected between the UART transceiver (such as the MAX232) and the microcontroller or system's TTL-level UART interface. The primary purpose of using the ADG5209F in this design is to selectively route UART signals based on system conditions. A second important purpose is fault protection because the ADG5209F features overvoltage tolerance up to ±55V on its analogue pins. [2]

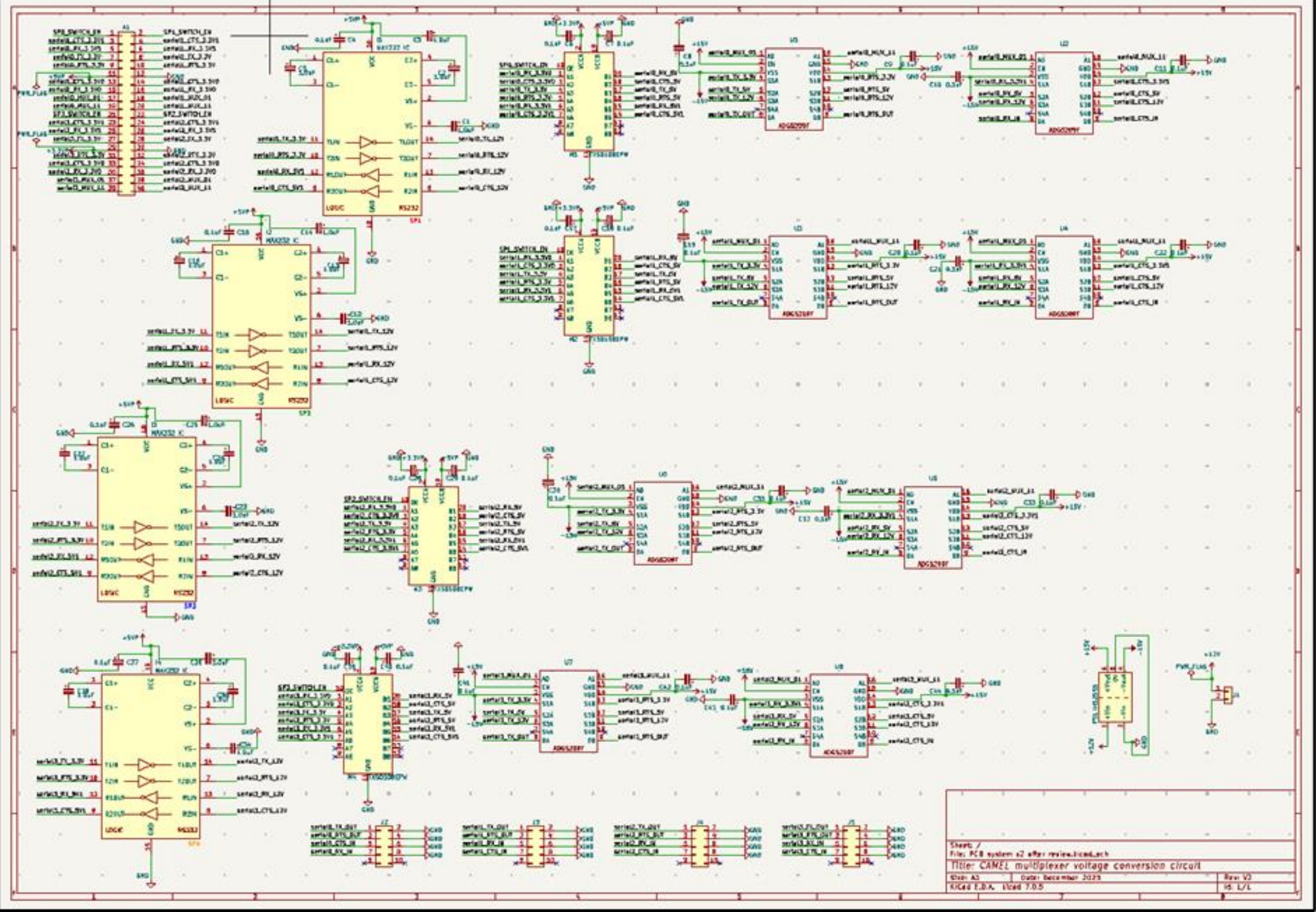
The TX0108EPW in the RS-232 to TTL to TCP/IP interface circuit is used to translate voltage levels bidirectionally between the UART and the TCP module, protect devices from overvoltage, preserve signal integrity at high speeds, manage power sequencing safely. UART interfaces involved in TCP/IP communication often operate at relatively high baud rates like 115200 bps, 921600 bps, or higher, especially when a bridge to a fast Ethernet or Wi-Fi link. The TX0108EPW is designed for high-speed operation, capable of supporting signal transitions up to 60 Mbps (megabits per second) with low propagation delay (~4 ns typical). This ensures that even at high baud rates, the level shifting does not introduce asynchronous timing errors or noise that could corrupt UART frames, especially over longer PCB traces. The circuit requires a voltage regulator in the system because RS-232 driver signalling circuits require significant positive and negative voltage switching ranges typically between +12v and -12v. Modern RS-232 transceivers like the MAX232E+ voltage converter internally generate these voltage switching ranges from a smaller supply (e.g., 5V), using a charge pump that generates voltages of approximately ±7v to ±10v. [3]

This Python script implements a serial-to-TCP bridge system using the Flask web framework, enabling remote configuration and control of up to four UART serial ports, with optional GPIO-based hardware control. It supports dynamic port configuration via a REST API and real-time, bidirectional data forwarding between serial ports and TCP clients. The system consists of four main components, each responsible for a distinct part of the UART-to-TCP communication framework. The first component is a Flask web server, which provides RESTful APIs for configuring UART port parameters. It also serves a basic HTML web interface through the root (/) route using Flask's render_template() function. This interface allows users to interact with the system through a browser.

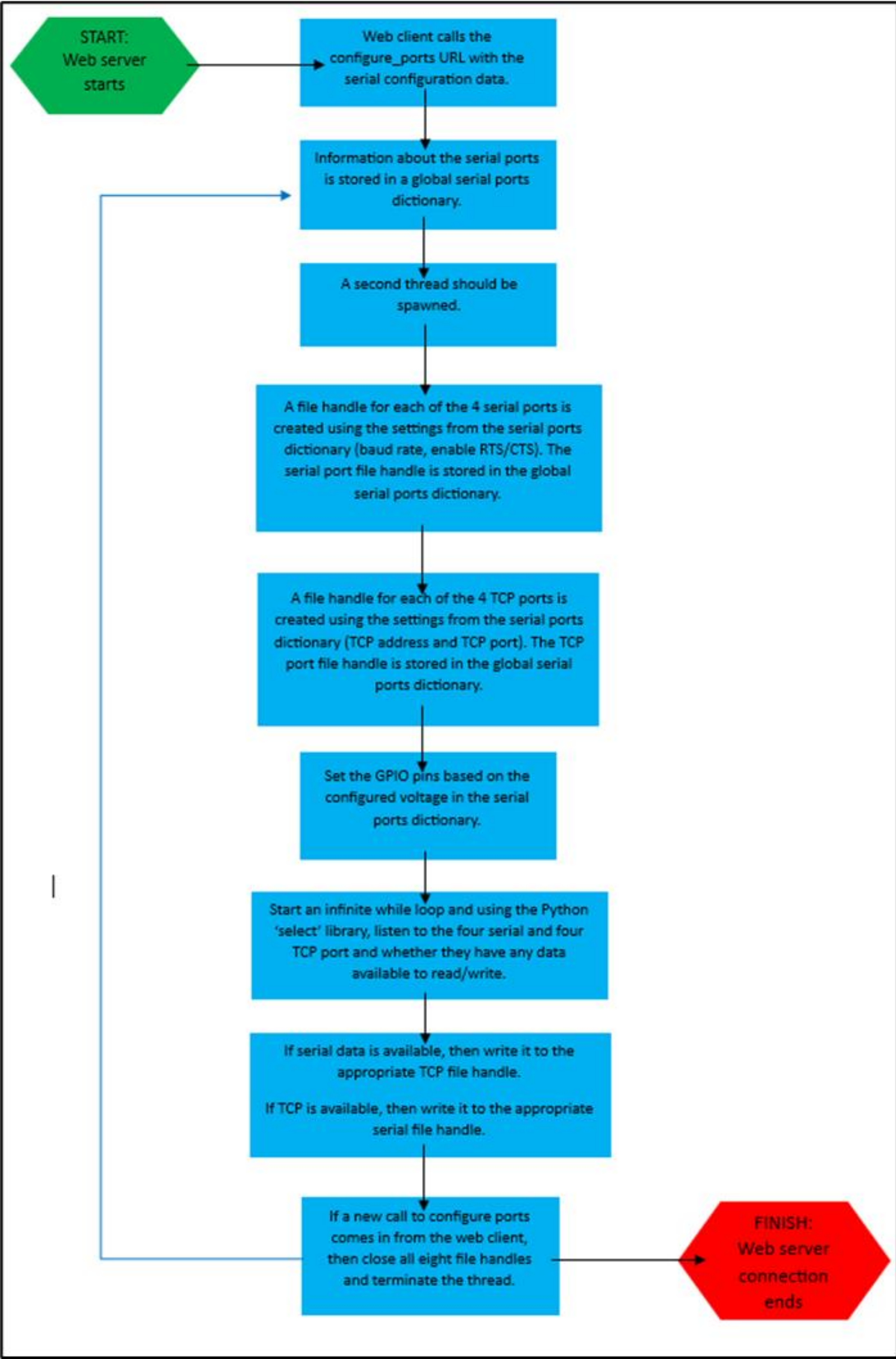
The second component is the Serial Port and TCP Socket Manager, which handles the initialization and management of both hardware and virtual serial devices, such as /dev/ttyAL1. It is responsible for opening and configuring the serial ports and linking each one to a corresponding TCP socket. This module uses the select system call to manage multiple file descriptors efficiently, enabling simultaneous I/O across different ports and clients. The third component is a background thread that functions as an I/O multiplexer. This thread continuously monitors all configured serial and TCP ports and facilitates bi-directional data transfer. It forwards data from the serial port to the TCP client and vice versa, ensuring real-time communication. Additionally, it dynamically manages TCP client connections and updates file descriptor sets to reflect changes in connection status. The fourth component is an optional GPIO control module, which is designed for System-on-Chip (SoC) platforms like the DE10 NANO FPGA. Although this part of the system may be commented out in certain configurations, it provides functionality to control external multiplexers or switches using GPIO pins. Each UART port can be associated with specific GPIO pins for activation or switching purposes.

References

- [1] Analog Devices, MAX232E+ IC Voltage Converter Datasheet, MAX202E–MAX241E Series, Rev. 4, Aug. 2016. [Online]. Available: https://www.mouser.co.uk/datasheet/2/609/MAX202E_MAX241E-3127932.pdf
- [2] Analog Devices, ADG5208F/ADG5209F Fault-Protected Analog Multiplexers Datasheet, Rev. 0, Oct. 2016. [Online]. Available: https://www.analog.com/media/en/technical-documentation/data-sheets/adg5208f_5209f.pdf
- [3] 3 Texas Instruments, TXS0108E 8-Bit Bi-Directional, Level-Shifting, Voltage Translator for Open-Drain and Push-Pull Applications, Rev. L, Nov. 2024. [Online]. Available: <https://www.ti.com/lit/ds/symlink/txs0108e.pdf>



KiCAD circuit diagram for TCP to UART protocol converter for between RS232 serial and 5.0v and 3.3v TTL



Python software flowchart for the Serial to TCP bridge and configuration