

Proposal Freya Vafadari

by Freya Vafadari

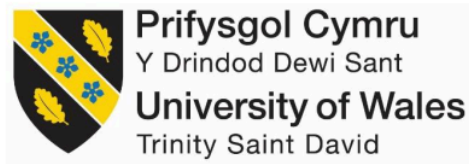
Submission date: 28-Oct-2024 01:45PM (UTC+0000)

Submission ID: 241181018

File name: 32483_Freya_Vafadari_Proposal_Freya_Vafadari_596981_1416163225.docx (916.69K)

Word count: 2062

Character count: 10276



IP to UART protocol converter for between RS – 232 and 5.0v and 3.3v TTL

**For BEng in Embedded electronic systems
engineering**

By Freya Jaleh Vafadari

2024/2025 Academic Year

Contents

List of figures	2
Project description	2
Project breakdown	3
Proposed time scales of work	4
Background information	5
IP and UART protocol	5
RS -232 and 5.0 and 3.3V TTL	7
MAX 232 IC voltage converter	7
DE 10 NANO FPGA	8
Resources.....	8
References	9

List of figures

Figure 1 System block diagram	3
Figure 2 Project breakdown	4
Figure 3 Gantt chart	5
Figure 4 TCP/IP framework.....	Error! Bookmark not defined.
Figure 5 UART communication.....	7
Figure 6 UART timing.....	7
Figure 7 RS – 232 lines and voltages and TTL voltages	8

Project description

My project proposal is to design and build an IP to UART protocol converter for between the RS – 232 and 5.0v and 3.3v TTL voltages to meet the requirements of an existing project within my organisation. The wider project in my organisation and has a requirement 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. Figure 1 shows the system block diagram at a high level for my project.

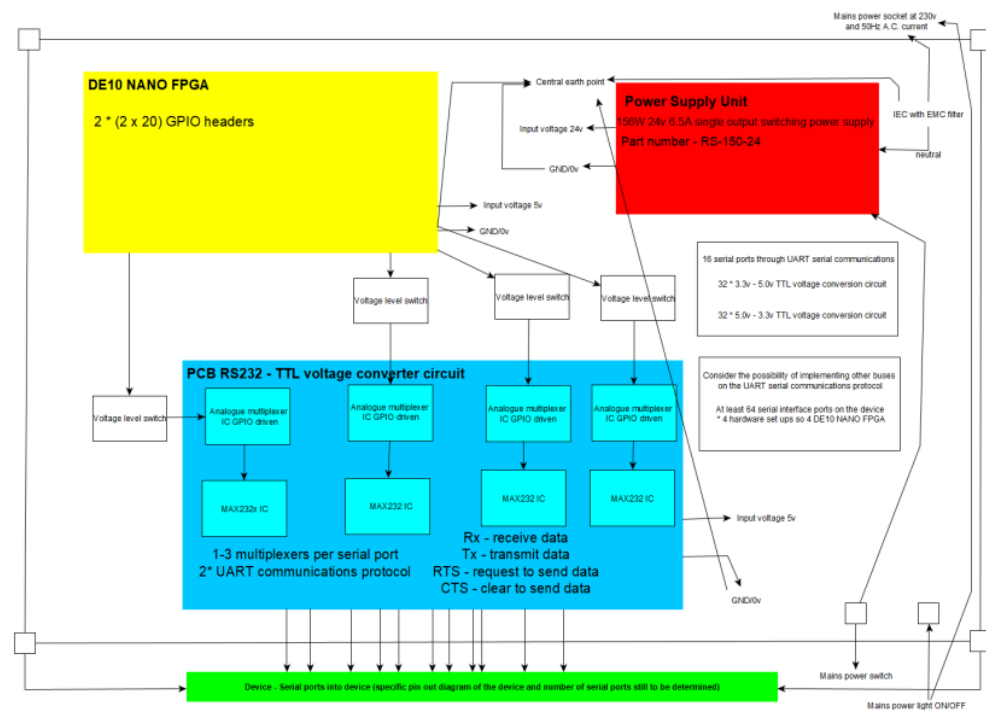


Figure 1 System block diagram

Project breakdown

The diagram below in Figure 2 shows the proposed breakdown of my project and the order which software and hardware development will be completed in.

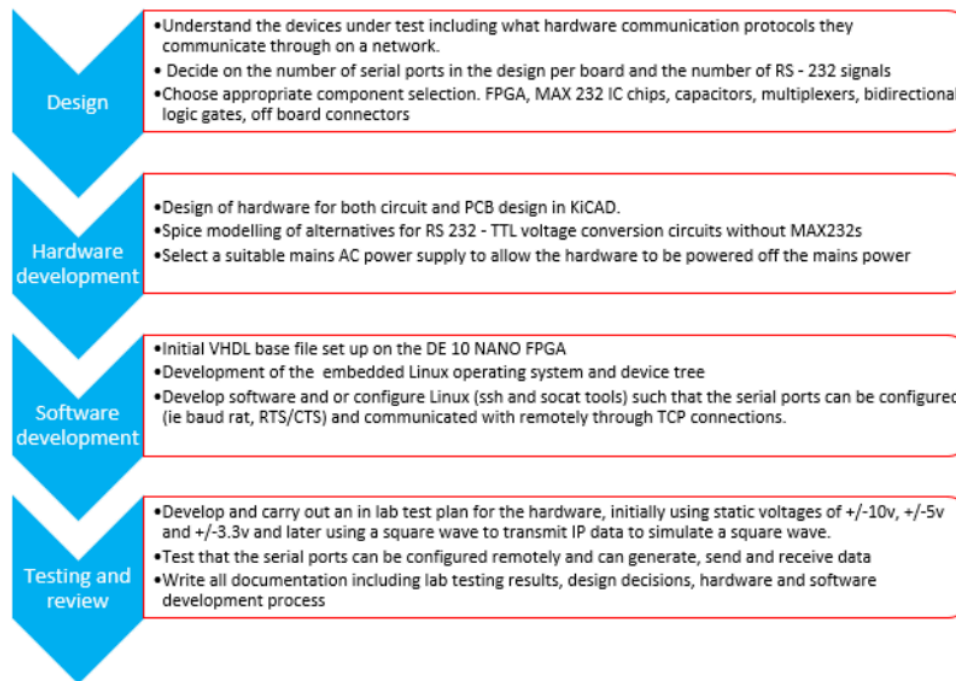


Figure 2 Project breakdown

Proposed time scales of work

Figure 3 below shows the proposed Gantt chart for my project starting from April 2024 to April 2025 and the predicted timelines I hope to achieve each specific task in. The specific tasks on the Gantt chart are explained in more detail above in Figure 2 separated into the four elements of the project cycle – design, hardware development, software development and testing and review.

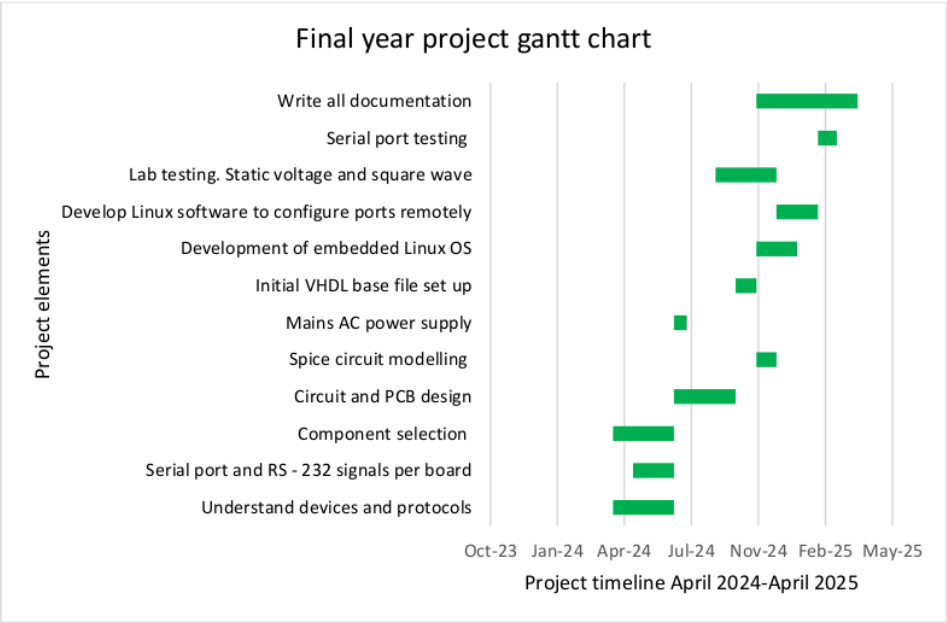


Figure 3 Gantt chart

Background information

IP and UART protocol

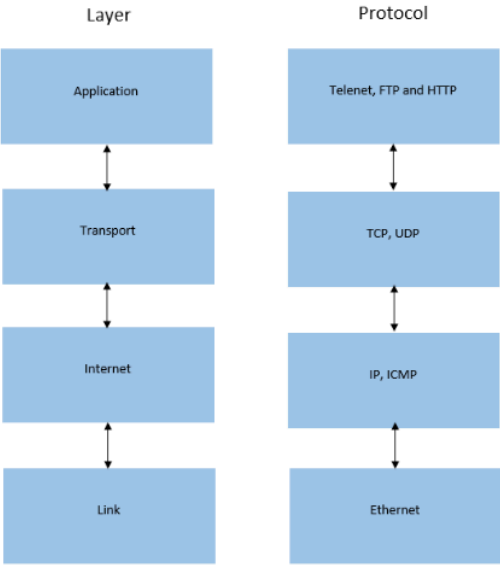


Figure 4 TCP/IP framework

The TCP/IP framework stands for the Transmission Control/Internet Protocol and is a set of standardised regulations that allow computer peripherals to communicate on an internet connected network. TCP and IP are two distinct computer network protocols where IP is the side of that obtains the network address of where the data packets are sent and TCP is responsible for data transmission once the IP address is known. If data was transmitted in one piece, it's likely that it would be corrupted or fragmented and so all the data would have to be transferred again. However, TCP/IP separates each message into packets and the packets are reassembled once the data has been transmitted. The TCP/IP framework separates the different communication tasks into layer and each layer has a specific purpose. Data is transmitted through four individual layers before it is received at the other peripheral where the TCP/IP data is processed through the layers in reverse order to reassemble the data. The layer and protocol structure of TCP/IP are shown in Figure 4.

TCP/IP is a data link protocol that is implemented when transmitting data on the internet. The datalink layer handles the physical aspects of transmitting and receiving data for example through an Ethernet cable, wireless VPN network or a network device driver in the computer. The internet layer controls the packets movements inside the network. The transport layer forms a stable data connections between two peripherals and divides the data into discrete packets and sends and receives acknowledges of recipient of the packets. The application layer is a collection of applications that need network connections like emails and instant messaging.

The UART or Universal Asynchronous Receiver Transmitter protocol is a standard for serial communication between devices and allows devices to send and receive one data bit at a time without the need for synchronisation of external clocks. The two signals of each UART device are the transmitter (Tx) and receiver (Rx) and the function of these signals is to transmit and receive serial data for serial communication. The baud rate on both devices needs to be the same and is the rate of data transfer in the communication channel. If data is being transmitted from a serial port like it is in the case of my project the baud rate is the maximum number of bits per second being transmitted. UART transmits data in asynchronous form and so doesn't use a clock signal and transmit data as packets. A packet consist of a start bit, data frame, parity bit and stop bit. The UART data transmission line is 'high' when not transmitting and the data transfer starts when the state of the line changes from 'high' to 'low' for one clock cycle. When the receiving UART detects a state change, it starts reading the data bits at a frequency equal to the baud rate.

The data frame contains the data being transferred and is 5 – 8 bits longs if a parity bit is present and up to nine bits if there's no parity bit. The parity bit is a check for the UART receiver to determine if any data has changed or been corrupted during transmission. After the receiver reads the UART frame, it counts the number of bits and checks if it's odd or even and if its 0 there's an even parity and if it's 1 it's an odd parity. If the parity bit matches the data, then transmission is free from errors, however if it's 0 and the total is odd or 1 and the result is even then data has changed. At the end of the frame, there's a stop bit sent by the sending UART when the transmission line is driven from 'low' to 'high' for 1-2 bits. Information about UART communication and the diagrams for Figures 5 and 6 is referenced from UART protocol [1].

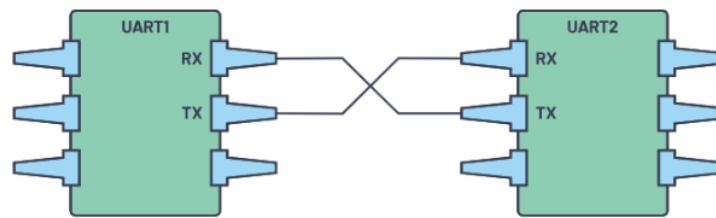


Figure 5 UART communication. UART protocol [1]

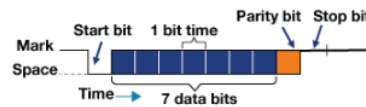


Figure 6 UART timing. UART protocol [1]

RS-232 and 5.0 and 3.3V TTL

RS-232 is a standard protocol for serial communication utilised to connect computers and external peripherals. The RS-232 standard defines the electrical characteristics and timing of signals, behaviour of signals, physical size and pin out diagrams of peripheral connectors. Logic '1' in the standard is in the voltage range -15 to -3v and logic '0' is in the range +3 to +15v. Typically for RS-232, logic '1' is -12v and logic '0' is +12v and all are defined with respect to a common GND. A voltage between -3V and +3V is in an undefined logic state.

In RS232, the data is transmitted serially in one direction over a single data line and at least three wires are needed for two way communications (Rx, Tx and GND) in addition to control signals and so it uses start and stop bits to tell the receiver when to check for data. The individual signals on a serial port are unidirectional and if connecting between two devices, the output of one device must be connected to the input of the other device. The devices are separated into two types of data terminal equipment (DTE) like microcontrollers and data circuit terminating equipment (DCE) like an LCD display. In my project the DTE is the DE 10 NANO and the DCE is the connected device. In RS-232 the transmitter or the DTE sends a start bit to the receiver to inform it that data transmission is starting from the next bit and the 5-9 bits are data bits. If there is a parity bit, then a maximum of 8 bits can be transmitted otherwise 9 bits are transmitted and after this a stop bit is transmitted. Information about RS-232 standards is summarised from RS-232 standards [2].

TTL or transistor-transistor logic is a collection of digital logic circuits that utilise bipolar junction transistors to implement different logic functions. The fundamental block is the transistor which is a switch and can be formed of two different types of logic gates: NAND or NOR gates and are formed by connecting several transistors in a specific arrangement. Each logic gate in a TTL circuit will switch between two states: high (1) at 5v and low (0) at 0v representing binary data. TTL serial is a type of single ended serial communication using transistor voltage levels "high" for 1 and "low" for 0. One example of driver-receiver interface between TTL and long range serial communication is the MAX232 implementation which converts between TTL and RS-232. Information about the TTL standard is referenced from TTL standard [3].

MAX 232 IC voltage converter

I've chosen to use a MAX 232 integrated voltage converter for each channel because it's a common component to implement in circuits which require serial signals from a microcontroller at TTL logic

levels to be converted to signals that are compatible with the RS – 232 serial communication standards. The MAX232 is a dual transmitter and receiver that is commonly implemented in circuits to convert Rx, Tx, CTS and RTS signals between RS – 232 and TTL voltages. In my project, a single channel comprises of the four separate signals Rx, Tx, CTS and RTS to form the UART protocol and each FPGA converts signals for a maximum of eight channels. The MAX 232 series IC chips have two receivers that change from RS-232 to TTL voltages and two drivers that translate from TTL to RS-232 voltages and, therefore only two out of four RS-232 signals can be converted in each direction. The first receiver/driver pair of the MAX232 used for Tx and Rx signals and the second pair for CTS and RTS signals. Information about the MAX232 IC is summarised from MAX 232 IC [4] and Figure 7 is a table RS – 232 line types, voltages and TTL voltages.

RS – 232 line type and logic level	RS – 232 voltages	TTL voltages to/from MAX 232
Data transmission (Rx/Tx) logic 1	-3V to -15v	5.0v
Data transmission (Rx/Tx) logic 0	+3V to +15v	0v
Control signals (RTS/CTS) logic 1	-3V to -15v	5.0v
Control signals (RTS/CTS) logic 0	+3V to +15v	0v

Figure 7 RS – 232 lines and voltages and TTL voltages. MAX 232 IC [4]

DE 10 NANO FPGA

I've chosen to use a DE 10 NANO to send and receive the serial signals in the system featuring an Intel Cyclone SoC FPGA which has an ARM processor. I've chosen to use the DE 10 NANO because there are 80 GPIO pins separated into 2 * 40 pin headers which allows me to have eight IP to UART protocols lines for each FPGA because each channel requires exactly 9 GPIO pins in addition to a 5.0v power line, 3.3v power line and two GND lines for each 40-pin header. This means that for every FPGA, there is a maximum of eight IP to UART conversion channels between RS – 232 and TTL logic equally split between two 40 pin headers. The FPGA has 1 * 10 pin analog input header which is also compatible with the Arduino analog input lines.

Resources

The circuit and PCB designs for my project are designed using the KiCAD 8.0 circuit and PCB development software. I've decided to design the circuit and 4-layer PCB using KiCAD because the software is open source and has extensive component libraries. One advantage to designing hardware in KiCAD is that there is an extensive set of component libraries with symbol and PCB footprints. Another advantage of KiCAD is that there is a 3D viewer so while the PCB is being developed it can be viewed virtually in 3D.

KiCAD has an in-built circuit modelling package called Spice which I've decided to use in the design phase to model analogue examples of RS – 232 to TTL voltage conversion circuits for example using NPN transistors and NOT gates as analogue alternatives to the MAX232x IC chip of in level shifting circuits.

The embedded Linux and VHDL software is going to be designed in the Intel Quartus Prime FPGA software inside the Intel Quartus Prime Lite. There's a platform designer which is a tool that eliminates the manual system integration and VHDL coding by automatically generating the digital logic including clock, reset and power lines to form the VHDL testbench.

References

1. UART protocol information and images "An Introduction to the UART, SPI, and I2C Communications Protocols," *element14 Community*, Jan. 06, 2020.
<https://community.element14.com/learn/learning-center/the-tech-connection/w/documents/4664/an-introduction-to-the-uart-spi-and-i2c-communications-protocols> (accessed Oct. 25, 2024)
2. RS – 232 standard "Fundamentals of RS-232 Serial Communications | Analog Devices," *Analog.com*, 2024. <https://www.analog.com/en/resources/technical-articles/fundamentals-of-rs232-serial-communications.html> (accessed Oct. 26, 2024)
3. TTL standard "TTL -Transistor-Transistor Logic Families History and Applications," *ElProCus - Electronic Projects for Engineering Students*, Apr. 21, 2016.
<https://www.elprocus.com/transistor-transistor-logic-ttl/> (accessed Oct. 25, 2024)
4. MAX 232 IC information Wikipedia Contributors, "MAX232," *Wikipedia*, May 12, 2023.
https://en.wikipedia.org/wiki/MAX232#Voltage_levels (accessed Oct. 27, 2024)

Proposal Freya Vafadari

GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

74/100

Intro / Aims & Objectives		
The purpose of the project is clearly identified, focusing on the challenge of connecting devices without Ethernet connections to an IP network. However, the section could benefit from a more detailed problem statement and clearly defined aims. The application of the IP-to-UART protocol converter in the project could be elaborated further. The project breakdown is well-structured and clearly explained. Comprehensive and focused with a clear project scope!	20	14
Background to the Project / Literature Review		
Provides a good explanation of the TCP/IP framework and UART protocols, supported by appropriate diagrams. However, it would be improved by including more references, particularly from journal papers. Additionally, diagrams could be better positioned within the text to align with discussions on bit frames, enhancing readability and clarity. Detailed explanation with clear diagrams; improved references needed.	20	14
Project Design		
Demonstrates commendable technical knowledge. Detailed explanations of the protocols and system block diagrams provide a clear understanding of the proposed work's functionality. Placing the system block diagram under the "Project Design" title would improve the organization of the content.	30	26

Strong technical insights: improved diagram placement suggested.		
Resources / Planning		
The project is well-planned and organized, as evidenced by the detailed time scales, tasks, and durations provided. The project breakdown is clear and well-structured but could be more appropriately placed under the planning section. Ethics form submission is noted. Well-organized plan with clear structure; minor adjustments needed.	15	12
Document Presentation		
The document is neatly presented with minimal errors. However, the writing could be improved by organizing content more effectively under distinct topics, such as Introduction, Literature Review, Project Design, and Planning, for better structure and readability. Neat presentation: improved topic organization recommended.	15	8
TOTAL	100	74

PAGE 1
PAGE 2
PAGE 3
PAGE 4
PAGE 5
PAGE 6
PAGE 7

