A low-cost, universal, open-source, multi-device data acquisition system for RS485, RS232, GPIB, and analog devices.

K.W. Trantham and K.J. Ahrendsen

Abstract

A simple, and low cost computerized data acquisition system is presented. It is based on the Raspberry Pi and can interface to a variety of new and legacy laboratory instruments. For compatibility with the most common connection standards used by laboratory instruments (GPIB, RS232, or analog output), it uses a network of communication bridges based on the RS485 serial bus.

Introduction and motivation

Automated computer or microprocessor data acquisition (DAQ) systems have been in use for small scale, "personal" experiments, since the 1970's [1,2]. There are a number of data acquisition systems, including both hardware and software, available primarily for Windows-based Personal Computers (PCs). Including the cost of the computer itself, these systems can be quite expensive. Furthermore, such third-party products are generally proprietary, and the details of their DAQ systems, such as source-code or electrical schematics, are not available to the researcher. Over the last ten years, there has been an emergence of very low cost, single-board computers that are broadly open-source, both in software and hardware.

Originally developed as a means to teach coding, the Raspberry Pi (RasPi) computer has many general-purpose input/output connections that make it ideal for laboratory automation tasks. The RasPi uses an open-source Linux-based operating system. Furthermore, the vast majority of software and hardware developed for the RasPi is also open-source. However, given that the RasPi is relatively new platform, there are few solutions available to connect it to stand-alone laboratory instruments such as an ammeter or lock-in amplifier. Often, one has a collection of operational legacy equipment, with much or all of it being unusable in a RasPi DAQ system given the variety of connection standards. There are many examples [3-7] where the RasPi has found specific use in the research environment, often dedicated to a specific task. Herein we present a method for generally connecting the RasPi to any number of new and legacy instruments, based on a universal, addressable data bus.

One of the most common laboratory instrumentation interfaces is the General Purpose Interface Bus (GPIB), which conforms to the IEEE 488.1 standard.[8] This is an eight-bit parallel bus, with data management and handshake signals. The bus permits addressing up to 30 different instruments interconnected to a host controller. This bus was first developed by Hewlett-Packard (HP) in the 1960's and has continued to be the *de facto* standard in laboratory instrumentation connectivity. To be used with a host computer (such as a PC) an interface card must be installed. Data acquisition software can be written in native compiled languages, or in graphical development environments such as LabView[9] or TracerDAQ[10]. Unfortunately, a commercial solution does not exist to connect a Raspi to the GPIB. A new circuit directly interfacing the Raspi to the GPIB was considered in the development of our data acquisition system, but abandoned after considering other instrument connectivity issues.

Newer laboratory equipment may include a GPIB connection, but are more likely to include a Universal Serial Bus (USB), or an RS232 serial connection. Developing software to communicate over the USB to third party devices is problematic as special, proprietary drivers are often required and may not be generally available. This became apparent in our group when trying to interface to a HighFinesse Model WS-6 wavemeter. The RS232 serial bus lends itself to development of locally-authored software as manufacturers will generally publish command codes needed to operate their instrument. From an electronics point of view, interfacing the Raspi to a RS232 bus is straightforward. However the RS232 bus is not addressable and permits only one device to be connected to the DAQ computer at a time.

Given the diminishing use of GPIB [11] and the limitations of the RS232 bus, we consider the RS485 serial communication bus. The RS485 bus, standardized in 1998 [12], is also a serial bus like the RS232. However, the RS485 bus allows multiple devices to be on the same network. Many modern industrial controls employ the RS485 bus, and its use is becoming more popular in research settings[13-17]. Herein we show how the RS485 bus can be used as an alternative, universal instrumentation connection scheme for laboratory apparatuses. We show how to connect multiple legacy RS232 devices or GPIB devices with a communication bridge circuit. In developing this bus, it was realized that other, simple interface devices could also be constructed to expand the measurement and control capability of the laboratory data acquisition computer.

The RS485 Bus as a Universal Data Bus

The RS485 communication standard defines potentials to represent logic states differently than do the RS232 systems. Instead of dedicated "transmit" and "receive" potentials measured relative to ground (three wires) as used in RS232, a version of the RS485 standard uses only two wires with floating potentials not necessarily referenced to ground. Convention identifies the two wires and associated potentials as A and B, and V_A and V_B respectively. Logic states are determined by the potential difference between wire A and wire B; i.e the state equals the difference V_A - V_B. For example, if the difference is greater than zero this represents a logic 0, and if it is less than zero it represents a logic 1. The advantage this scheme has over RS232 communications is that it is less susceptible to induced noise and completely immune to ground loops. As a result, RS485 connecting cables can be longer than those for RS232.

To manage transmit and receive functions on the RS485 bus with only two wires, one must adopt a half-duplex communication approach. With this, all devices on the bus are by default "listening" (receiving) with a very large Thevenin impedance seen at the A and B terminals. Only one device on the bus can transmit at a time. The data conversation is initiated by a "host" device such as the data acquisition computer. All other devices are then identified as "guest" devices. In transmit mode, the host device has a very small equivalent Thevenin resistance and is able to impose the appropriate potentials. The communication information must include an address so that the appropriate guest device knows to respond when the host device has completed transmission. After sending a message to the bus, the host enters a high impedance state (receive mode) and awaits a response from the addressed guest device. The addressed guest device, in turn, responds by transmitting data on the bus. Thus, the RS485 is an addressable serial bus, in which there can be many guest devices attached to one host device.

The general architecture of the RS485 serial bus for this application is outlined in Figure 1. Since the guest devices also require power to operate, +12V and ground are provided along with the RS485 signals A and B. The electrical-network connection between devices is simple parallel, meaning that the network topology can be linear or forked. The RS485 standard does not specify a physical connector. An RJ-45 connector is used here, allowing up to eight independent connections, however only two are used for communication and four more for power (two wires for +12V and two wires for power ground). Since RJ-45 is the physical connection standard for Ethernet, a unique physical connector for this application would prevent accidental connection of this bus to the Ethernet. However, this connection standard was selected given the general availability of short "Ethernet cables" and the relatively low cost of printed-circuit-board (PCB) mounted RJ-45 connectors. The RS485 bus presented here should not be plugged into an Ethernet connection.

The RS485 standard also does not specify how data is encoded. In this application, data is encoded using a modified form of the Modbus protocol[18]. This specification outlines the format of transmitting data, which includes a device address. Most importantly, the protocol specifies a "cycle-redundancy-check" (CRC) to be appended at the end of each transmission. The CRC is two bytes of data calculated based on the original message and is checked by the receiver to ensure that there was not an error in transmission. When a guest device responds, in addition to appending an appropriate CRC, its address is also transmitted so that the host device can confirm that the correct device responded. Details of the data transmitted over the RS485 bus are discussed below.

In the present application, there are two basic types of guest devices used. The first type is a "bridge" device and there are two different versions. The first version is a RS485 to RS232 bridge. This circuit is different from commercially available RS232/RS485 converters, which typically only modify voltage levels. The device presented here permits addressable communication with an otherwise non-addressable RS232 device. In this way, multiple RS232 enabled devices can be controlled with one host device. The second bridge device is a RS485 to GPIB bridge. This permits a communication interface between the serial RS485 bus and parallel

GPIB bus. The second type of guest device presented is a variety of "general purpose" units intended to perform simple measurement or manipulation tasks. All of these guest devices and the host device, which attaches to a RasPi, is described in detail below.

Circuit Descriptions

All guest devices share required common elements in their circuitry. A schematic of these elements is shown in Figure 2. Each device has two RJ-45 connectors so that the bus may be extended to more devices. As shown, pin connections 1 and 2 of the RJ-45 carry the RS485 signals B and A respectively. Pins 7 and 8 deliver +12V power, with power ground on pins 3 and 4. Pins 5 and 6 are unused, and are tied together. A NCP117ST05T3G voltage regulator (U2) provides regulated +5V for the guest device. The µMAX485 integrated circuit (U4) is a data transceiver and provides the necessary electrical translation from RS485 differential potentials A and B to the +5V, ground referenced transistor-transistor-logic (TTL) signals and vice-versa. Digital connection RE/DE of U4 determines the mode of communication. When RE/DE is true, the device is in transmit mode, translating digital TTL level signals at input UART_TX1 to the appropriate differential signals at A and B. When RE/DE is logical false, the device is in receive mode and TTL data is provided to digital connection UART_RX1. Momentary switch SW1 and resistor R2 provide a means to reprogram the device's address. This will be discussed further below.

All of the guest devices presented use a Microchip microcontroller (a "PIC"). The development environment "MPLAB" is available at no cost from the manufacturer. The PIC is programmed using a PICKIT4 programmer.[19] This low-cost device connects to the guest board through the In Circuit Serial Programming (ICSP) connection P1 as shown in Figure 2. Once the PIC is programmed, the programmer is removed and the code remains on the microprocessor indefinitely.

All guest devices share optional circuitry for displaying information locally, as shown in Figure 3. The RS232 and GPIB bridge devices use a 128x128 TFT display while all other general purpose guest devices use a 128x32 OLED display. The displays chief purpose is to show the user their unique RS485 address when turned on. Optionally, they can provide debugging information, such as the messages being carried from one bus to another as in the case for the bridge devices. Only one connector and its associated display is used for a given device, but both are shown in Figure 3. Both displays operate with +3.3V digital signals, and the circuitry shown translates between this and the +5V logic used by the microcontroller of each device. Dual transistor package Q101 contains two BSS138 field effect transistors. The operation of this circuit as a logic level translation circuit is described in reference [20].

The RS485 to RS232 bridge circuit is shown in Figure 4. The indicated ports in Figure 4 connect to the ports of the same name shown in Figures 2 and 3. The microcontroller U1 is a PIC 18F46K22 whose programming determines the overall behavior of the device. The code used on all devices is freely available [21]. The 18F46K22 has two Universal Asynchronous Receive and Transmit (UART) modules. The first UART module is connected to U4 (µMAX485) of Figure 2 and communicates with the RS485 bus. The second UART module is connected to U3 (MAX232) and communicates with the RS232 bus. As the female DB9 is commonly used on the instruments that support RS232 communication, connector P3 in Figure 4 is a male DB9 intended to plug into the back of the instrument. Essentially, this bridge device listens on the RS485 bus for data specifically addressed for it. When properly addressed through the RS485 bus, it strips out RS485 and Modbus specific data and transmits the intended message (usually ASCII text) to the connected instrument via the RS232 bus. The bridge waits for a certain period and transmits any data returned from the instrument back through the RS485 bus with appropriate Modbus encoding data. Printed circuit Gerber files are also available [21].

The RS485 to GPIB bridge circuit is shown in Figure 5 and is similar in function to the RS232 bridge. It accepts properly addressed and encoded messages intended for a GPIB instrument, passes along the instrument-specific instructions, and returns any data passed back from the instrument. This circuit only uses one UART of the microcontroller for the RS485 bus. The GPIB communication is handled manually through the digital data Port A for reading and writing data to the GPIB bus, while digital port D handles GPIB bus management signals per IEEE 488.1. A full description of how the GPIB bus operates can be found in reference [22]. Briefly, the digital signals in the GPIB bus are “active low” meaning that a given signal is not true until the signal is near ground level. With many possible instruments on the bus, the signal line will not be sensed low until all instruments have pulled the line low. The resistor networks shown in Figure 5 permit the microcontroller to both sense the level of a given signal and transmit data when needed.

Since the PIC microcontrollers have software-controllable digital input/output ports as well as analog input ports, we realized that RS485 devices could be constructed for simple remote data acquisition and control tasks. Placing data acquisition and control tasks in a remote device frees the main computer for other tasks. An example task that is better removed from the host computer is the provision of regular step pulses to control a stepper motor. Because the RaspPi runs a multitasking Linux operating system, like Windows on a PC, microprocessor time dedicated to user code can be variable. This makes providing regularly timed pulses difficult. A PIC microcontroller, on the other hand, runs dedicated code and timing functions can be more precisely controlled with on-chip counter-timer circuits.

The electrical schematic for a general purpose RS485 digital guest device is shown in Figure 6. This device can be programmed to control two servomotors, control one stepper motor, or simply be a general-purpose digital input/output device. Depending upon how the PIC is programmed, output ports RA5 and RA4 of the PIC provide digital signals to control two servomotors. Alternatively, these pins can output the pulse and clock signals for one stepper-motor, or, along with RC3 and RB4, can be used for general digital input/output (I/O) tasks. The circuit is based on the PIC16F690 microcontroller. Like the PIC18F18K22, this microcontroller is inexpensive and readily available. It has three 8-bit general purpose digital I/O ports, eight 10-bit analog to digital converters (ADCs), two timers, two comparators, and one UART. Many of these functions are tied to the same pin. For example, pin 12 of the PIC16F690 is internally connected to digital port B bit 5, analog to digital converter channel 11, *and* the receive connection for the UART. Therefore, the use of the pin must be configured in software with the intended function. This means that other possible functions tied to a given pin cannot be used at the same time. Connectors J4 and J5 provide power and signals for conventional servo connectors. If this digital guest-device is used as a servo controller, a separate voltage regulator U7 provides dedicated power for servos. Although the servos and microcontroller circuit both require +5V to operate, the higher current drawn from the servos can cause the supply voltage to fluctuate such that the microcontroller will reset. This generally leads to erratic behavior.

Another use of the PIC16F690 as a remote RS485 guest device is to measure analog signals. Some legacy equipment will only have an analog output as the remote sensing method. Figure 7 shows the schematic for a four-input analog RS485 analog guest device. The analog input section has a simple resistor and Zener diode network for limiting the input voltage to +5.1V relative to ground as measured by the microcontroller. If the device needs to measure a different voltage range, or bipolar voltages, then additional analog voltage conditioning circuits can be added. The PIC16F690 uses a 10-bit analog to digital converter (ADC). This resolution is sufficient for many applications, however there are a variety of other PIC microcontrollers that have better resolution or pre-processing options such as averaging (see, e.g. the PIC18F27K42).

Finally, we present the circuitry for the host computer in Figure 8. The host computer is a Rasberry Pi model 2 and connects to the circuit via connector P1. The circuitry is similar to the guest devices in that a μMAX485 integrated circuit translates +5V TTL logic signals to RS485 potentials. The networks around Q1 and Q2 provide logic level shifting between the +5V system of the μMAX485 and the +3.3V logic of the RasPi. There are two extra resistors attached to the A and B wires of the RS485 bus. The A and B wires are completely floating and their potentials could possibly drift outside of the 0 to +5V range used to power U4. These resistors provide a minuscule amount of biasing current to prevent this. The main difference between the RasPi host device and the other devices shown in previous figures is the software. The host computer always initiates a data transfer by first addressing the intended device and then awaiting a response. Although only two RJ-45 connectors are shown in Figure 8, the user can add as many connectors as needed. The RasPi computer can also interface to other devices through the unused circuit node tags shown in Figure 8, e.g. CE0, MOSI, and WPi25. These are not relevant to the RS485 host controller application and will not be discussed.

Data Format

The serial data format for all devices follows a modified form of the Modbus protocol. The order of each 8-bit byte has significance. The first byte transmitted by the host is the unique RS485 address of the device which is intended to reply. This is followed by a control byte; examples include 0x03 for reading from a register or 0x06 for writing to a register. The next two bytes indicate which register to read from or write to. In the case of writing, the next two bytes indicate what value to write. The last two bytes are the low and high order CRC bytes. As an example, the byte stream

0xA1 0x03 0xF0 0xF2 <CRC_l> <CRC_h>

transmitted by the host is intended for device number 0xA1 (decimal 161) and is requesting to read data from register 0xF0F2. The host then goes into listen mode and awaits a response. Only guest device 0xA1 will respond. The byte order of the response also has the address first, and the second, control byte is echoed. In the case of writing to a register, the entire message is typically echoed back. In the case of reading from a register, the third byte indicates the number of bytes of data to follow in the stream. The low and high order CRC bytes are transmitted last. Thus, continuing with the above example, the device might respond with five bytes of ASCII encoded data for "HELLO" as,

0xA1 0x03 0x05 0x48 0x45 0x4C 0x4C 0x4F <CRC_l> <CRC_h>.

All devices on the RS485 network employ a means of changing the RS485 address of the guest device. This is necessary since all devices on the network must have a unique address between 0x00 and 0xFF. When the microcontroller is first programmed, the address is indeterminate. All guest devices have an "ADDRESS" port to the microprocessor and associated momentary switch. When the switch is held closed and a properly encoded message is sent at the same time, the guest device will remember the address used in the message as its own and act on the message. When the switch is released, this new address is permanently saved even if power is removed from the device. A second method of changing the address is provided by software.

By sending the proper information through the RS485 bus to the correct register of the device, the address can be changed as well.

An Example Use

We have adapted the digital host device to control laser paths on an optical table. It is a straight-forward procedure to connect a small flag to a servo. This allows the DAQ system to block, or unblock a laser beam as desired. Further, a servo can be connected to a flip mirror for controlling the laser beam direction. Referring to Figure 9, a small "fork" (item 4) is mounted on a servo motor. An adjustable mirror mount (item2) [Thorlabs KM100] is mounted to a flip mount adapter (1) [Thorlabs FM90]. The servo is mounted so that its axis is concentric with the axis of the flip mount adapter. Finally, a protruding screw (item 3) in the mirror mount fits between the tines of the fork.

Conclusion

We have presented a universal data acquisition bus to network a Linux-based Raspberry Pi to a variety of laboratory equipment. With communication bridges and general purpose RS485 devices, this system has been tested with a variety of devices. We have also developed a novel, remote controlled flip-mirror based on the general-purpose digital device. Table 1 shows a complete list of commercial equipment tested with our system and the mode of interface to the RS485 bus. At any given time, all of this equipment can be connected to the bus and DAQ computer.

Equipment	Mode of connection to RS485 bus
Keithley Model 485 Picoammeter	GPIB bridge
Keithley Model 617 Electrometer	GPIB bridge
Vortex Laser Model 6000	RS232 bridge
Sacher Laser Head Controller PilotPC 500	RS232 bridge
Sacher Tapered Amp Controller PilotPC 4000	RS232 bridge
Omega Model CN7800 Temperature Controller	RS232 bridge
B&K Model 1696 power supply*	RS232 bridge
SRS Model SR830 Digital Lock-in Amplifier	RS232 bridge or GPIB bridge
HighFinesse Model WS-6 Wavemeter*	RS232 bridge
Grandville Phillips Model 307 ion gauge and Convectron gauge controller	Analog input guest device
FLUKE 8840 Multimeter	GPIB bridge
Sorensen 120-0.5 power supply	GPIB bridge

Table 1. A list of commercial laboratory equipment tested and used with the described RS485bus. Items marked with an asterisk have special notes, see text.

Interfacing to the Model WS-6 wavemeter requires a special note. The hardware for this device is controlled through the USB port with proprietary software written for the PC/Windows. The software has a configuration option to continuously broadcast the wavemeter's data through the RS232 serial port of the PC. The above RS485 to RS232 bridge was adapted with a female DB9 connector and wiring the Rx and Tx connections to the appropriate pins. The bridge was reprogrammed to read two sequences of the byte stream from the PC computer since, at any given moment when the data was requested from the DAQ host, the first data sequence could be incomplete. The B&K Model 1696 power supply can also connect directly through RS485. However, its mode of communication is ASCII text only. To preserve the purity of the communication on the RS485 bus and to avoid possible data errors with other devices using the Modbus data format, the RS232 bridge is used with this power supply.

The components used on each device and the host computer are relatively inexpensive. It is straight-forward to connect the RasPi to the RS485 bus since TTL-level serial Rx and Tx connections are readily available. This universal system of connecting multiple devices can be extended to using a PC as the DAQ. In this extension, however, a translation circuit such as a RS232 to RS485 bridge would need to be employed as TTL connections are not available on the PC.

Acknowledgements

This work was supported by the NSF through Awards PHY-1505794 and PHY-1806771.

References

1. D.W. Taylor, Review of Scientific Instruments 49, 1551 (1978).

2. Douglas Whiting, Review of Scientific Instruments 55, 1157 (1984).

3. Alfredo Benítez, Ulises Santiago, John E. Sanchez, and Arturo Ponce, Review of Scientific Instruments 89, 013702 (2018).

4. F. Arnold, I. DeMallie, L. Florence, and D. O. Kashinski, Review of Scientific Instruments 86, 035112 (2015).

5. James Keaveney, Review of Scientific Instruments 89, 035114 (2018).

6. Matthew G. Street, Cristin G. Welle, and Pavel A. Takmakov, Review of Scientific Instruments 89, 094301 (2018).

7. N. B. Clayburn, K. W. Trantham, M. Dunn, and T. J. Gay, Review of Scientific Instruments 87, 124903 (2016).

8. IEEE Standard Digital Interface for Programmable Instrumentation, Institute of Electrical and Electronics Engineers, 1987, ISBN 0-471-62222-2,

9. See www.ni.com/labview for LabVIEW.

10. See https://www.mccdaq.com for TracerDaQ.

11 J Ryland - Autotestcon, 2005. IEEE, 2005 - ieeexplore.ieee.org, DOI: 10.1109/AUTEST.2005.1609228

12. Electronic Industries Alliance standard ANSI/TIA/EIA-485-A-1998.

13. J. Vega, A. Mollinedo, A. López, L. Pacios, and S. Dormido, Review of Scientific Instruments 68, 963 (1997).

14. A. K. Sanyasi, R. Sugandhi, P. K. Srivastava, Prabhakar Srivastav, and L. M. Awasthi, Review of Scientific Instruments 89, 055113 (2018).

15. Hui Gao, Qiuping Wang, Weibin Wang, Qinglin Wu, Wentong Chen, Liusi Sheng, and Yunwu Zhang, Review of Scientific Instruments 72, 3979 (2001)

16. Sanjeev R. Kane, Chander Kant Garg, and R. V. Nandedkar, AIP Conference Proceedings 879, 635 (2007)

17. R. Szplet, P. Kwiatkowski, K. Różyc, Z. Jachna, and T. Sondej, Review of Scientific Instruments 88, 125101 (2017).

18. http://www.ni.com/white-paper/7675/en/ for Introduction to Modbus (white paper).

19. See https://www.microchip.com for PICkit programmers and MPLAB software.

20. Phillips AN10441 application note https://www.nxp.com/docs/en/application-note/AN10441.pdf

21. Schematics, microcontroller source code, and circuit board Gerber files available at [URL held pending anonymous peer review] github repository.

22. See http://www.ni.com/white-paper/3389/en/ for Introduction to GPIB messages (white paper)

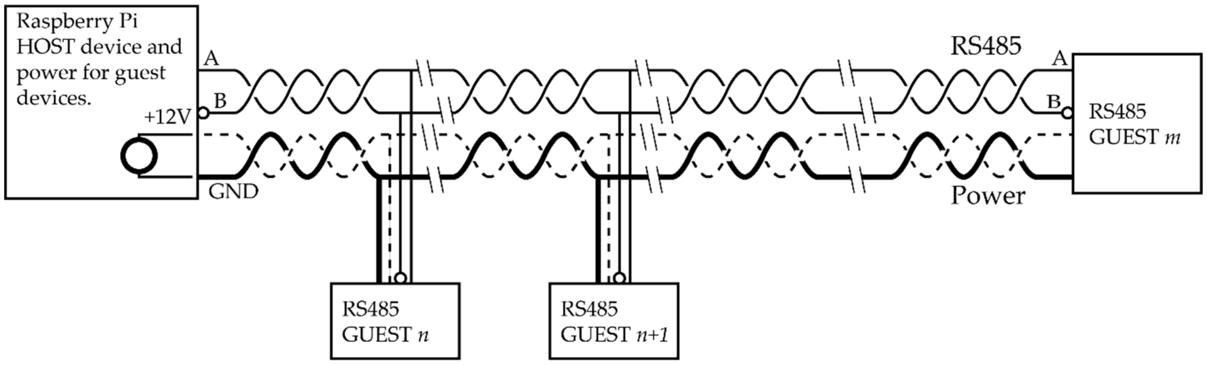

Figure 1. General architecture of the powered RS485 instrumentation bus.

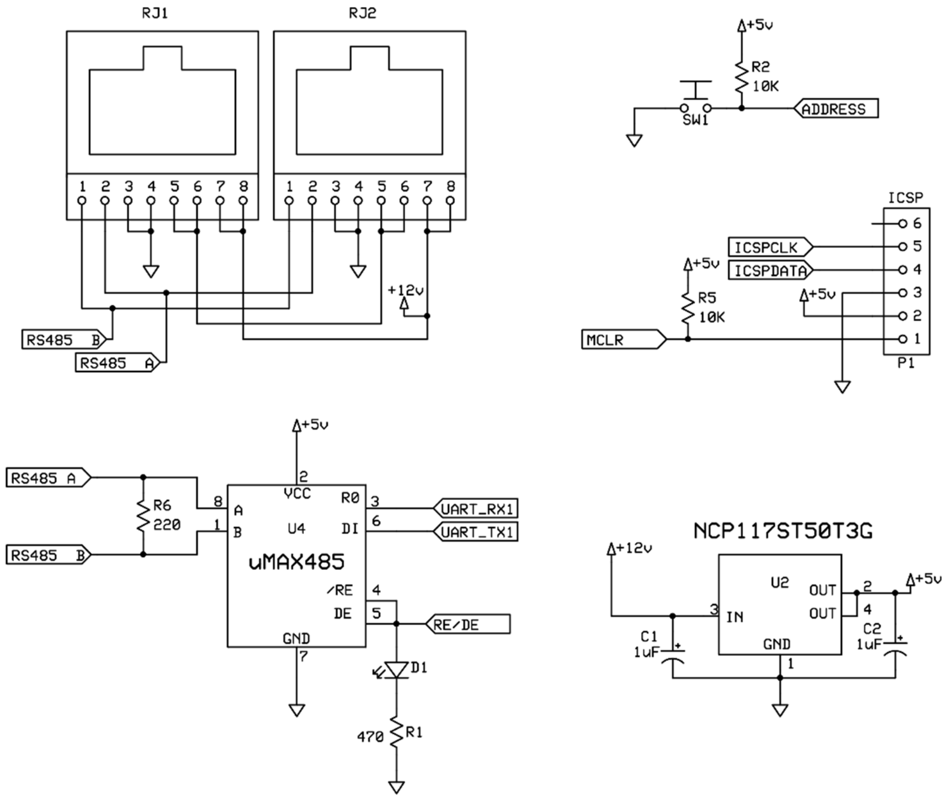

Figure 2. Schematic of the required common elements found in each guest RS485 device.

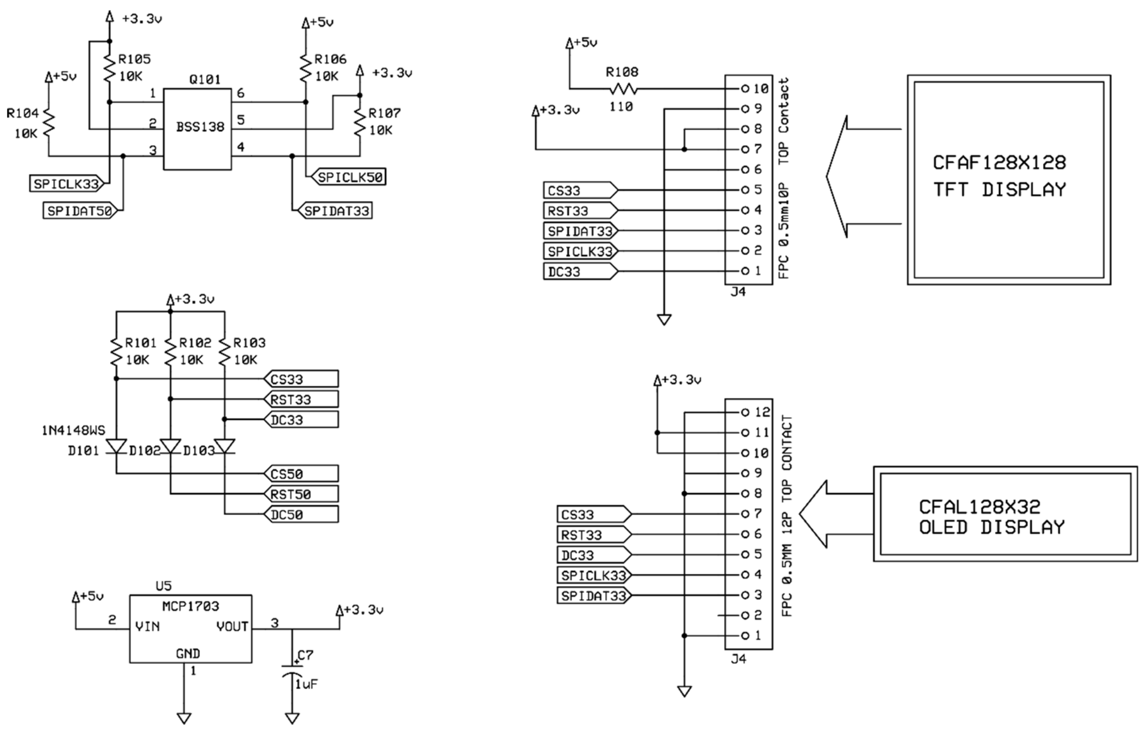

Figure 3. Schematic of optional display circuitry used on each device. The bridge guest devices use the 128x128 TFT display, while all other devices use the 128x32 OLED display.

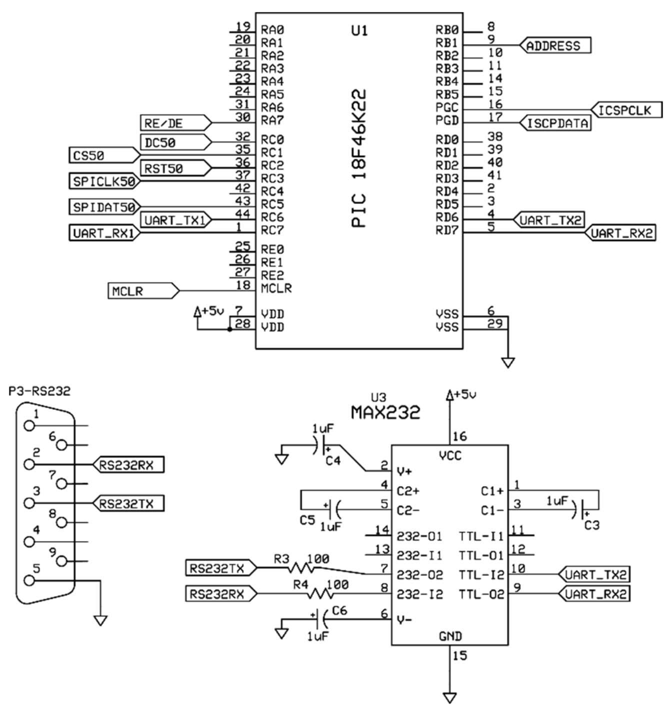

Figure 4. Schematic of RS485 to RS232 bridge guest device.

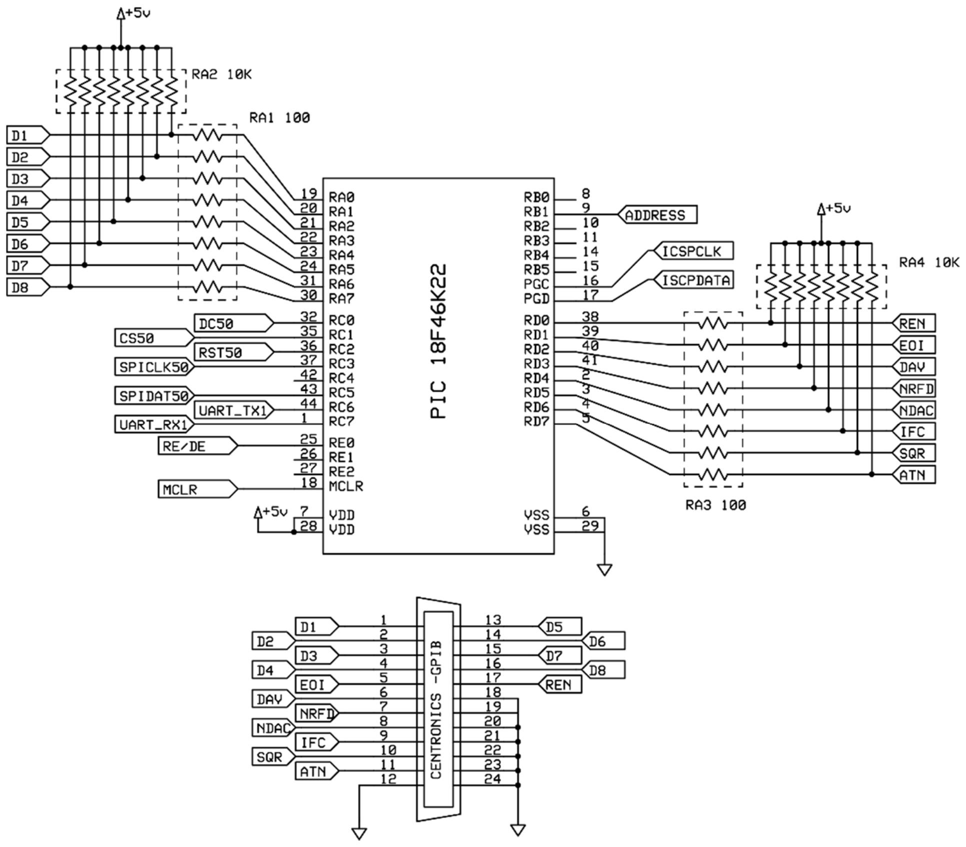

Figure 5. Schematic of RS485 to GPIB bridge guest device.

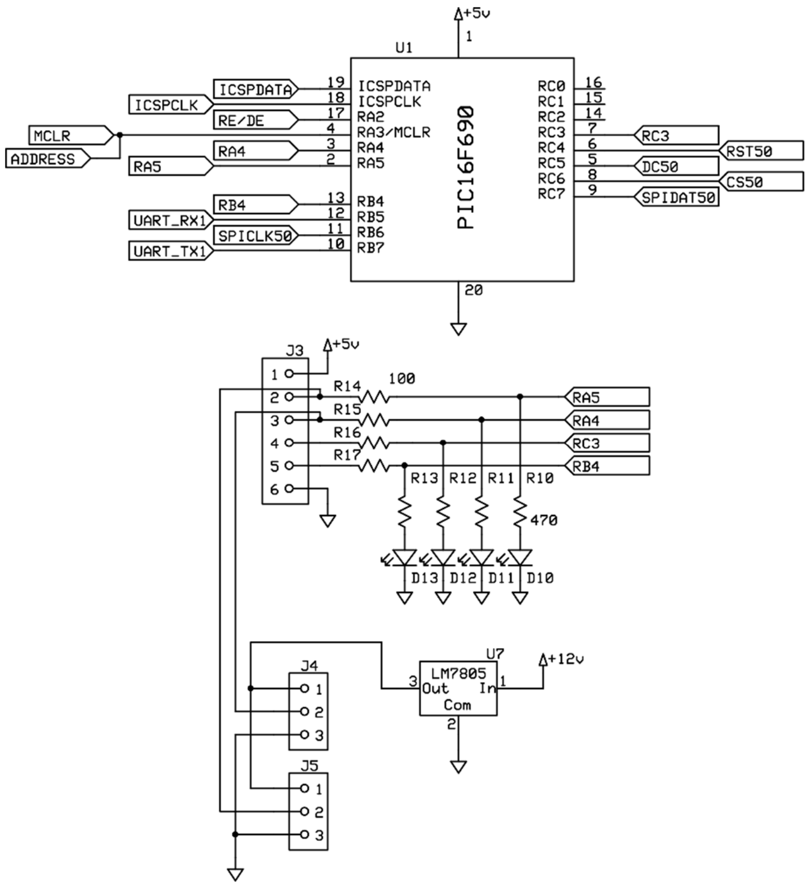

Figure 6. General-purpose digital RS485 guest device. With appropriate software, this circuit is used to control two servomotors, one stepper motor, or provide general digital I/O signals.

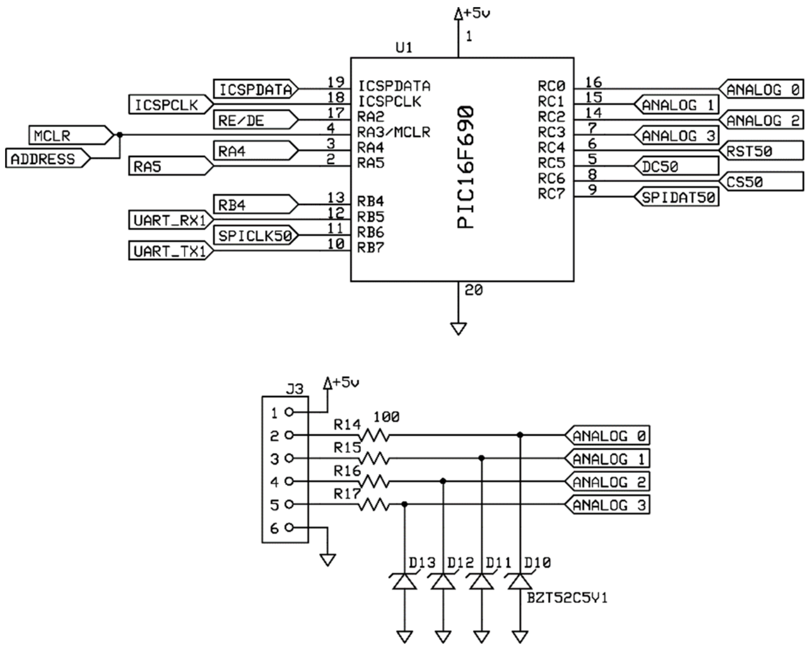

Figure 7. General-purpose analog input RS485 guest device.

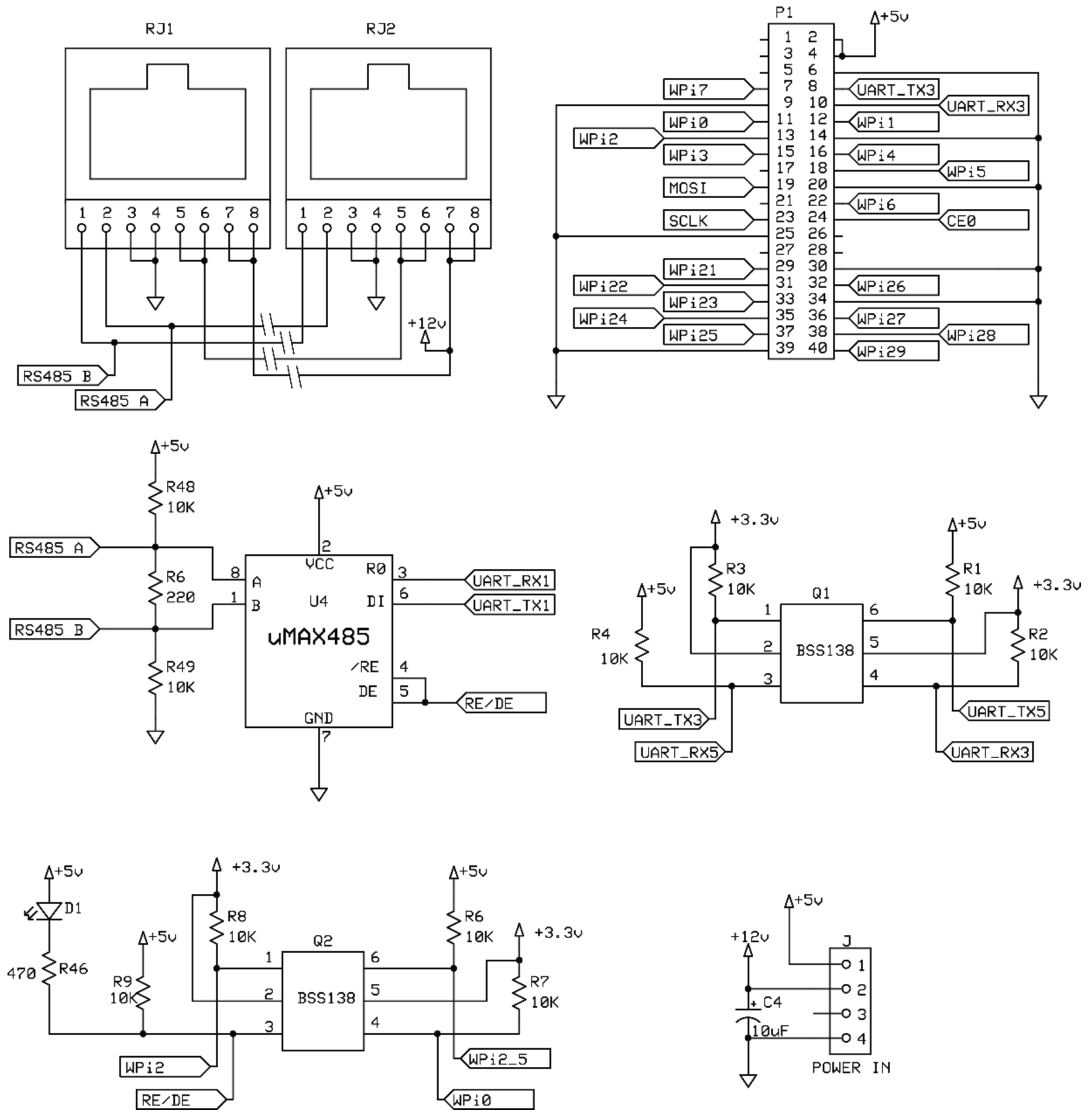

Figure 8. Circuit for the raspberry Pi RS485 host controller.

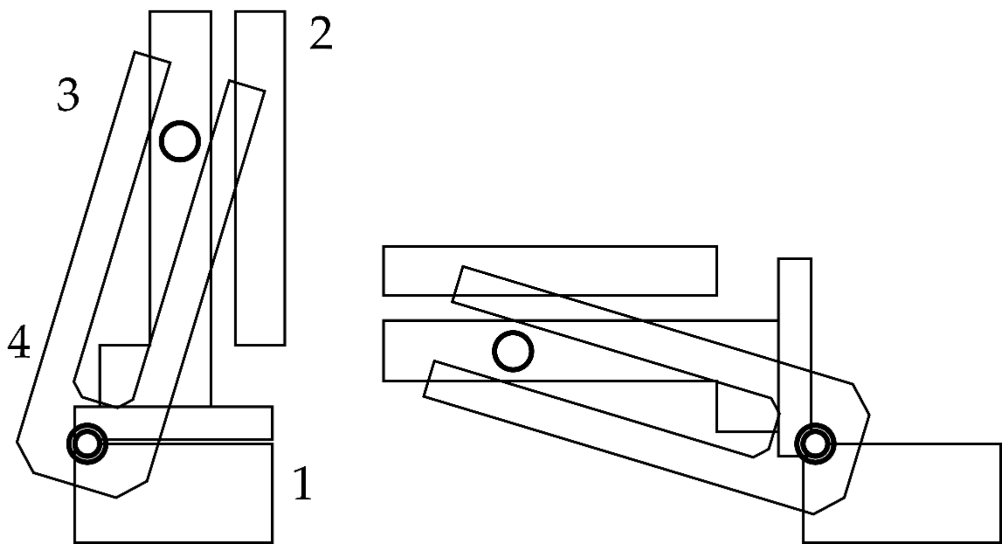

Figure 9. An automatic flip-mirror based on the digital servo guest device. A servo (not shown) is used to change the position of a mirror. See text for numbered item descriptions.