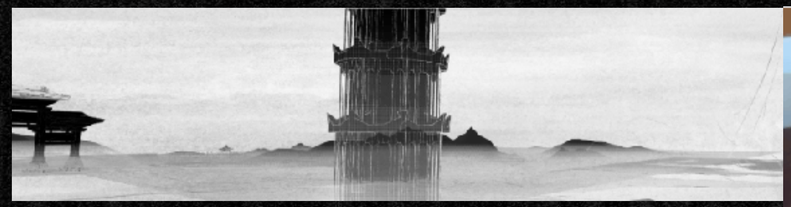
TOUCHDESIGNER EVENT TOKYO 2025

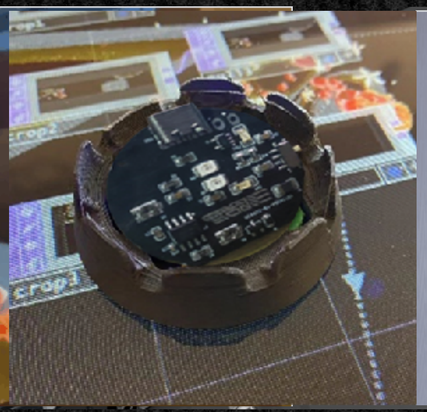
DECODING SENSORS N DATASHEETS

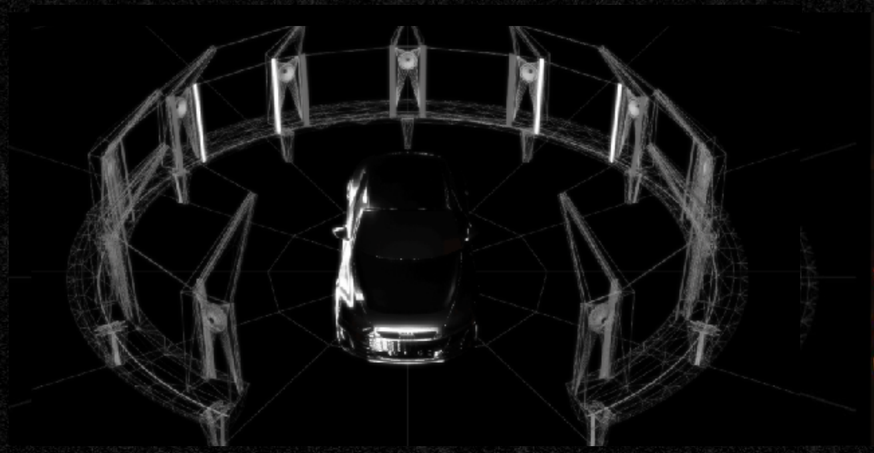
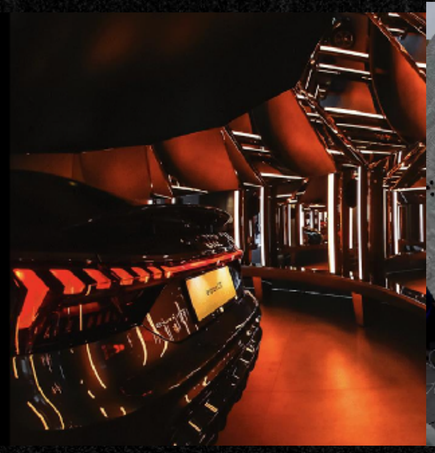
Retrieval & Integrating Sensors with TouchDesigner

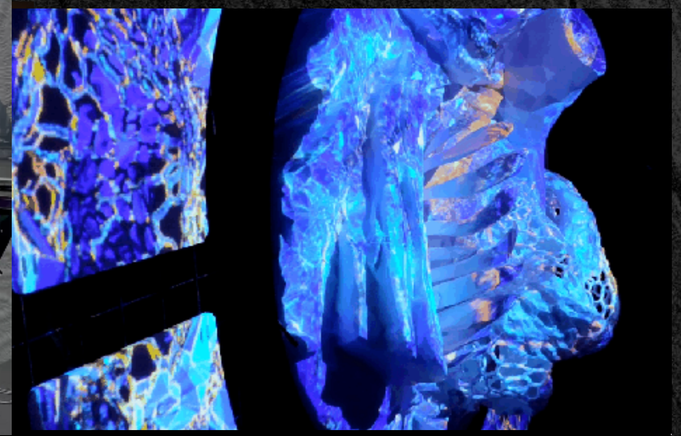
YANGLEI

WORKS



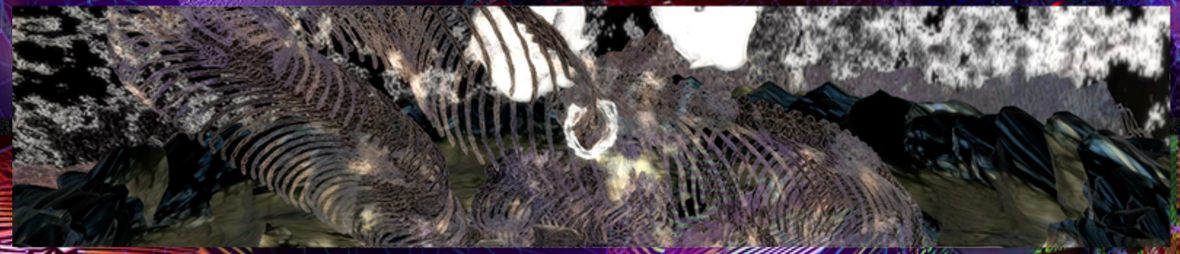



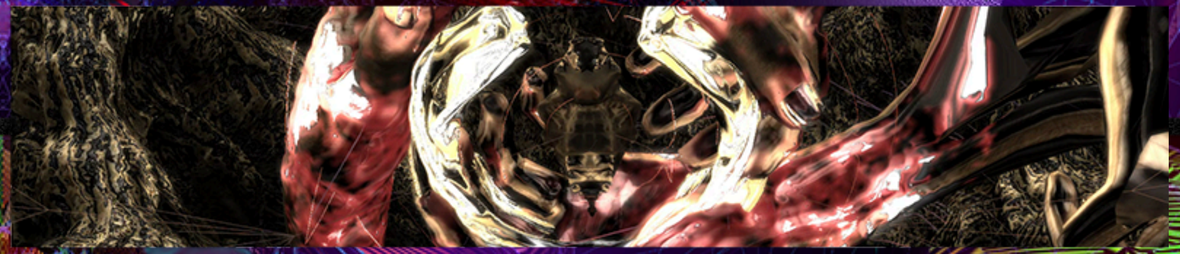






PUBLIC SPACE
INTERACTIVE INSTALLATION

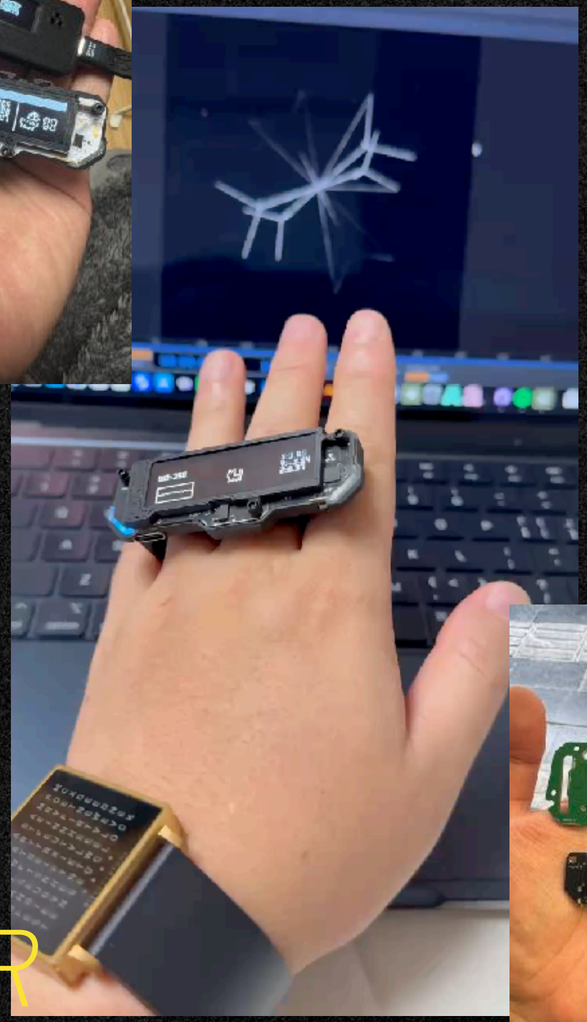
WORKS

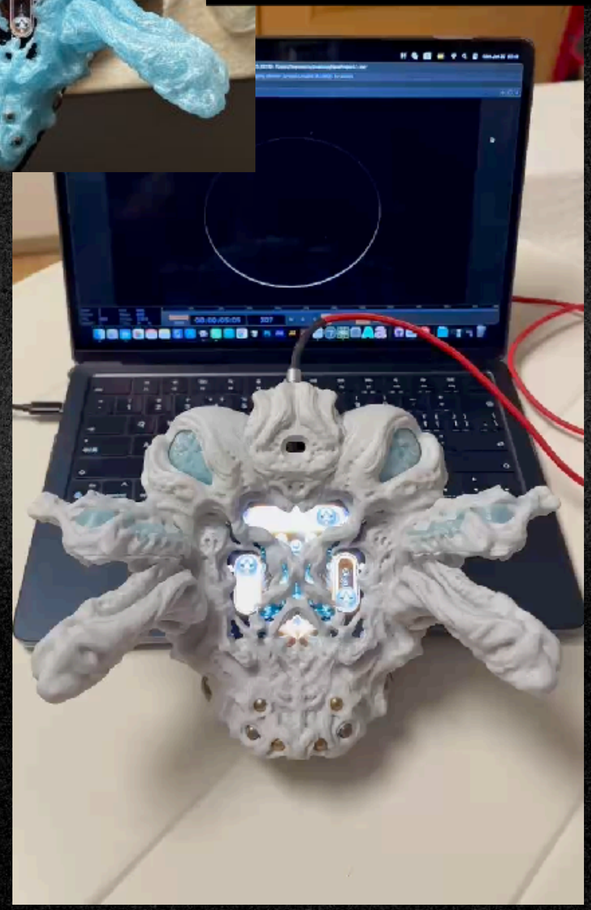




LIVE VISUALS

WORKS





CONTROLLOR DEVELOPMENT

2025 TOUCHDESIGNER TOKYO EVENT

*TRUE TITLE

BEFORE BUYING CHEAP & EASY-TO-USE SENSORS ON ALIBABA

What You Need to Know to Avoid Surprises ** and Make Them Work with Touchdesigner

YANGLEI

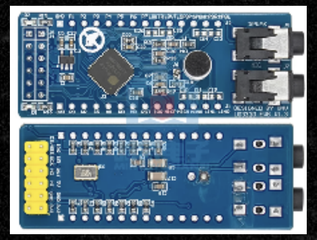
SERIAL PORT!

Most common & accessible interface

the simplest way I think for people without a hardware background to begin with ,
give more energy to the creative work but not the tools, and its also where I began with my hardware works

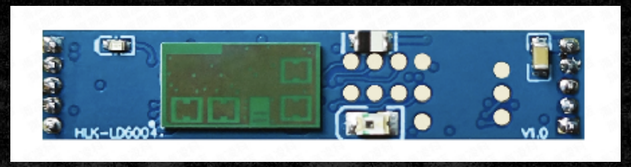
SERIAL PORT SENSORs

There are countless serial port sensors available for you to pick

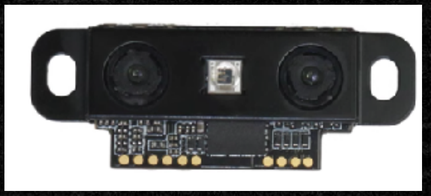


Speech recognition module

Fingerprint recognition module



Millimeter-wave human positioning module

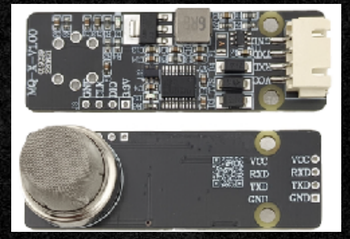


Facial recognition module

Gesture recognition module

Posture/IMU module

ToF distance-measuring module



environment sensor module

And a whole range of super stable industrial sensors

and so on.....

Quantum Computing and Serial Communication Protocols

CHOOSING SENSORs

Comparison of Serial Port Communication Interfaces

Protocol / Interface	Signaling	Max Speed	Max Distance	Nodes Supported	Topology	Voltage Level	Notes / Advantages
UART (TTL)	Single-ended	115.2 kbps (typical), up to 3 Mbps	<1 m (recommended)	2	Point-to-point	3.3V / 5V	On-chip interface, simplest hardware
RS-232	Single-ended	115.2 kbps (common), up to 1 Mbps	~15 m	2	Point-to-point	±3V to ±15V	Mature, PC legacy ports, poor noise immunity
RS-485	Differential	10 Mbps @ 12 m; 100 kbps @ 1200 m	1200 m	32–128 nodes	Multi-drop bus	±1.5V diff	Long distance, high noise immunity, industrial standard
RS-422	Differential	10 Mbps	1200 m	1 TX → 10 RX	Point-to-multipoint	±2V diff	Long range stable, but not true multi-drop
Bluetooth Virtual COM (SPP / BLE UART)	Wireless	SPP: 1 Mbps / BLE UART: 115.2 kbps	10–30 m (typical)	Many devices (depends on profile)	Star (host–peripherals)	2.4 GHz RF	Cable-free, but higher latency (10–30 ms)
USB Virtual COM (CDC)	Differential	USB 2.0: 12 Mbps (FS), 480 Mbps (HS)	5 m (per segment)	1 host + many devices	Tiered star	±400 mV diff	High speed, plug-and-play, also powers the device

when searching sensors on shopping websites, if you see any of the keywords in the first column Especially the first three and the product provides a user manual, most time touchdeisnger can work with it.

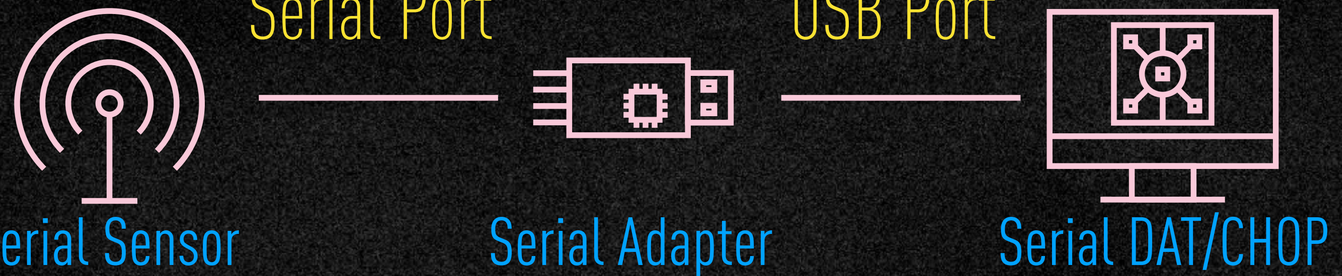
CHOOSING SENSORs

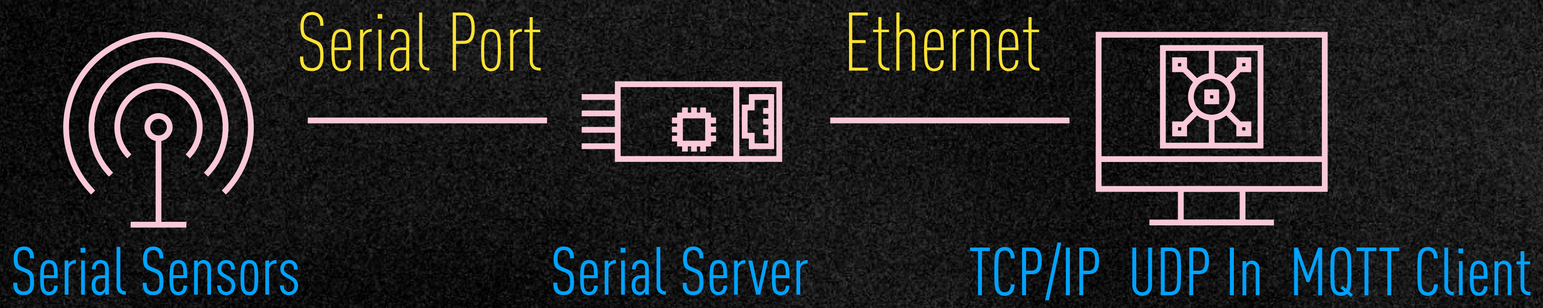
MODBUS

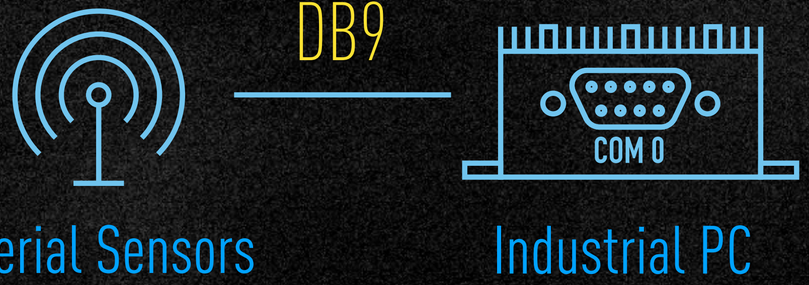
Modbus Type	Transmission Medium	Frame Start/End	Addressing	Error Check	Max PDU Length	Typical Speed / Delay	Typical Use
Modbus RTU	RS-485 / RS-232 / Serial	3.5 char silence defines frame; CRC16 (2 bytes)	1 byte slave address (0 = broadcast)	CRC 16	≤253 bytes; ADU ≤256 bytes (address+CRC included)	Baudrate variable (commonly 9600/19200/115200); obey inter-frame silent interval	Industrial fieldbus, PLC, sensors
Modbus ASCII	RS-485 / RS-232 / Serial	Start : and end CRLF; LRC (1 byte)	1 byte slave address	LRC	Same as RTU	Same as RTU; ASCII each char takes 2 bytes	Older devices, visual debugging
Modbus TCP	Ethernet (TCP/IP)	MBAP header (7 bytes), no CRC	Unit ID (1 byte) in MBAP header	Provided by TCP/IP	≤253 bytes; MBAP + PDU	Depends on network; low latency; supports concurrent connections	Remote monitoring, Ethernet gateways, SCADA/IIoT

This is a widely used industrial application-layer protocol that has been around for many years.and there's a lot of sensors using it, so if you see modbus RTU or ascii， it means we can use it too， but need some extra effort in tochdesigner scripts

CHOOSING ADAPTERs







Built-in serial adapters (most time RS232)

Serial Adapter

Serial Server

CHOOSING ADAPTERs

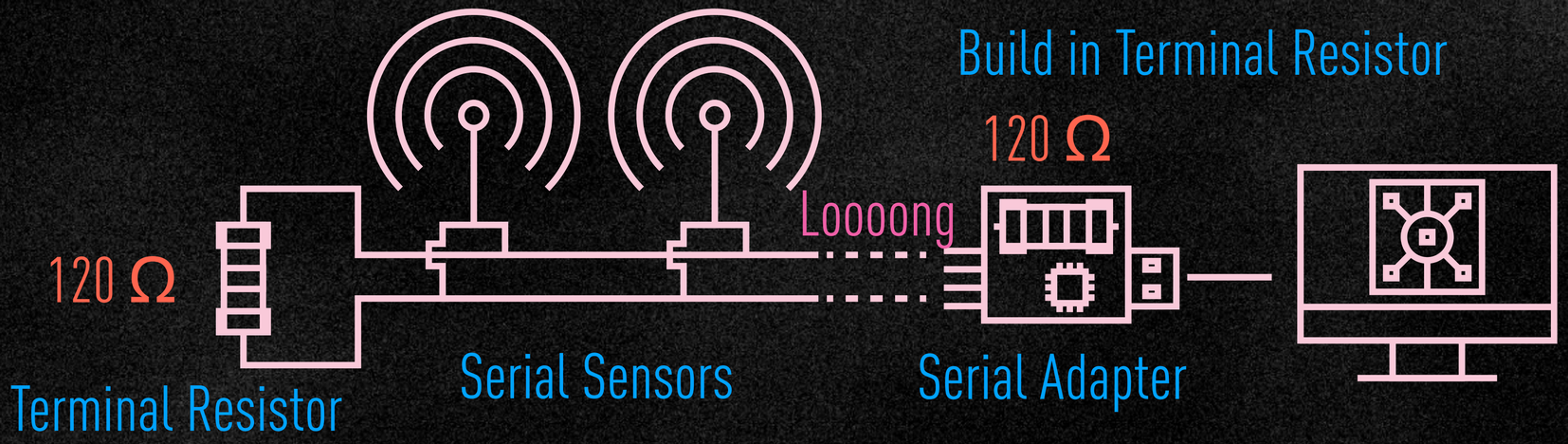
USB to UART(TTL)
Recommendations:

FDTI:
FT232R / FT232RL
Best but expensive

Silicon Labs:
CP210x series
Good

WCH:
Ch340x series
Cheap and good

RS485 Impedance machine



Build in Terminal Resistor

COMMUNICATING VIA USER MANUAL#

Datasheet & user manual reading

PE01

DATASHEET READING - basic parameters

Sensor Performance Specifications

Datasheet Page: 1

Operating range	0.2m~8m(90%reflectivity indoor 0klux)[1] 0.2m~2.5m(10%reflectivity indoor 0klux)[2] 0.2m~8m(90%reflectivity outdoor 90klux) 0.2m~2.5m(10%reflectivity outdoor 90klux)
Accuracy	±6cm@(0.2m-3m)[3] ±2%@(3m-8m)
Distance resolution	1cm
Frame rate	1-250Hz[4]
Ambient light immunity	70Klux

Electrical/Communication Specifications

Datasheet Page: 2

Electrical parameters	Supply voltage	3.7V-5.2V
	Average current	≤70mA
	Power consumption	≤0.35W
	Peak current	150mA
	Communication level	LVTTL(3.3V)
	Communication interface	UART, I²C, I/O

Interface: UART
Signal Level: 3.3v
Need a USB to UART adapter

Power Voltage: around 5v
Peak & average current: no more then 500ma
No need to buy extra power adapter

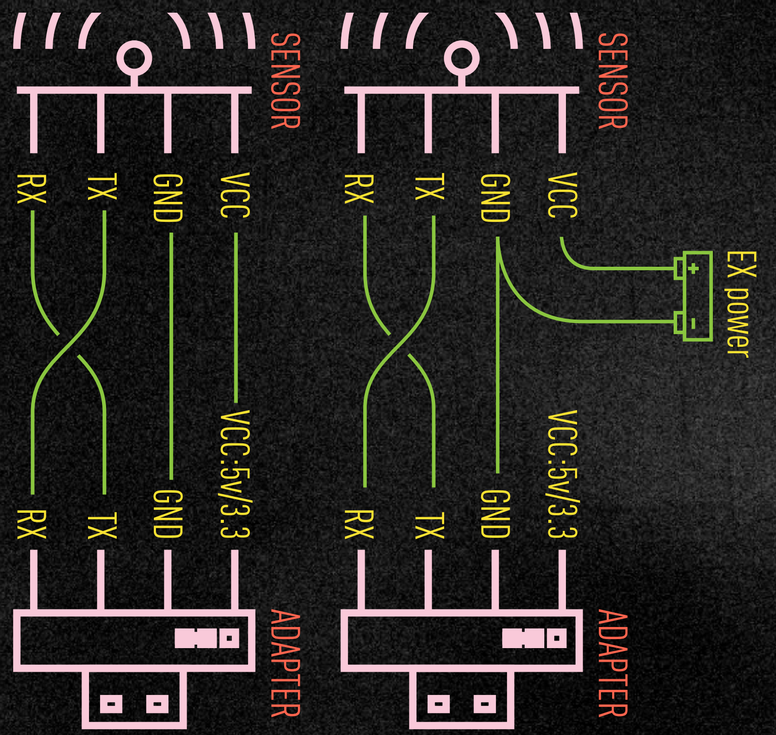
Ask for cable !

DATASHEET READING - pin connection

No.	Function	Description
1	+5V	Power supply
2	RXD/SDA	Receiving/Data
3	TXD/SCL	Transmitting/Clock
4	GND	Ground
5	Configuration Input	Ground: I2C mode Disconnected/3.3V: Serial port Communications mode
6	Multiplexing output	On/off mode: Output I2C mode: Data ready signal

User manual Page: 8

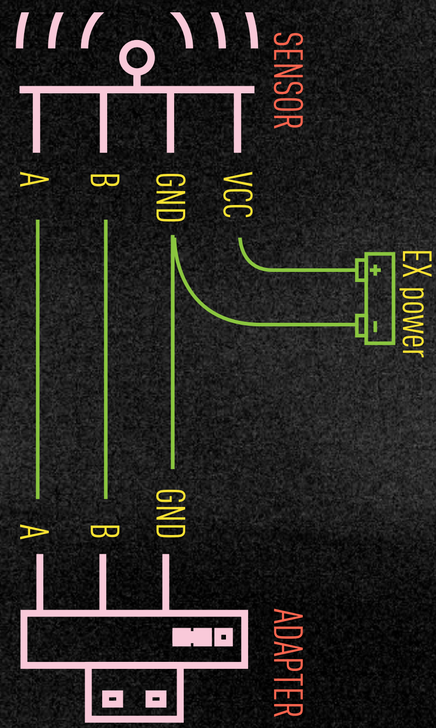
UART/RS232



TX / RX or TXC / RXD

Check Power Level!
some RS232 sensor maybe use 12v

RS485



A / B or D+ / D-

USER MANUAL READING - serial settings

6.2 Serial Port Communication Protocol

Serial port communication starts when pin 5 is disconnected or connected to 3.3v. It will set TF-Luna receiving RXD on pin 2 and sending TXD on pin 3. The serial port communication protocol is defined as follows: 8 data bits, 1 stop bit with no parity check and default baud rate of 115200 bps.

User manual Page: 8

Baud Rate: 115200

Data Bits: 8

Stop Bits: 1

	0	1
0	message	bytes
1	g.. 2YY.	103 2 8 9 50 89 89 8 0
2	k.. 8YY.	107 2 8 9 56 89 89 6 0

Active	On
Row/Callback Format	One Per Message
Port	/dev/cu.usbserial-0001
Baud Rate	115200
Data Bits	8
Parity	None
Stop Bits	1
DTR	Enable
RTS	Disable

USER MANUAL READING - received massage frame structure

1. 9-byte/cm (Default)

This format is supported for any firmware after Ver. 0.0.5

Byte	0	1	2	3	4	5	6	7	8
Description	0x59	0x59	Dist_L	Dist_H	Amp_L	Amp_H	Temp_L	Temp_H	Check_sum

Dist: cm

Amp: Signal strength indicator. Dist value is unreliable when Amp < 100 or Amp = 65535 (Overexposure)

Temp: Celsius temperature = Temp / 8 - 256℃

User manual Page: 20

Head /// data /// verification

distance*High Byte x 256 +*Low Byte

Byte 0 & 1: 0x59 0x59 - Header bytes. Used to identify the start of a data packet.

Byte 2 & 3 (Dist_L, Dist_H): Distance data. The distance in centimeters is calculated as Dist_L + Dist_H * 256.

Byte 4 & 5 (Amp_L, Amp_H): Signal Amplitude (Strength). Calculated as Amp_L + Amp_H * 256.

Byte 6 & 7 (Temp_L, Temp_H): Chip Temperature data.

Byte 8 (Checksum): The integrity verification byte.

USER MANUAL READING - decode data

0	message	bytes
1	YY	89 89 0 0 254 11 136 9 76
2	YY	89 89 0 0 0 12 136 9 79 89 89 0 0 0 12 136 9 79
3	YY	89 89 0 0 255 11 136 9 77 89 89 0 0 252 11 136 9 74

two 0x59 head bytes. In serial dat bytes are translate from hexadecimal into decimal so it's 89

```
def onReceive(dat, rowIndex, message, bytes):
    # Retrieve the list of raw bytes received
    byte_list = bytes

    # Iterate through the byte list to find the start of a packet (Header: 0x59 0x59)
    for i in range(len(byte_list) - 8):
        # Check for the two header bytes
        if byte_list[i] == 0x59 and byte_list[i+1] == 0x59:

            # Found the start of a 9-byte packet at index i

            # ----------------- Data Extraction -----------------
            dist_L = byte_list[i+2] # Distance Low Byte (Index i+2)
            dist_H = byte_list[i+3] # Distance High Byte (Index i+3)
            amp_L = byte_list[i+4]  # Amplitude Low Byte (Index i+4)
            amp_H = byte_list[i+5]  # Amplitude High Byte (Index i+5)
            # Temperature and Checksum bytes are skipped

            # ----------------- Calculation -----------------
            # Calculate Distance (cm) using Low + High * 256
            distance_cm = dist_L + (dist_H * 256)

            # Calculate Signal Amplitude (strength)
            amplitude = amp_L + (amp_H * 256)

            op('constant1').par.value0 = distance_cm
            op('constant1').par.value1 = amplitude

            return
```

Find the head bytes.

Decode the length bytes

Apply to CHOP's channels

VERIFICATION

Serial data verification methods

```
# ----------------- Checksum Validation -----------------
# The Checksum is the lower 8 bits of the sum of the first 8 bytes.

# Sum the first 8 bytes of the packet
checksum_calc = sum(packet[0:8])

# Take the low 8 bits (equivalent to modulo 256)
checksum_calc = checksum_calc & 0xFF

# The received Checksum is the 9th byte (index 8)
checksum_received = packet[8]

# 2. Validation Check
if checksum_calc == checksum_received:
    # Checksum passed: Data is reliable. Proceed to parsing.
```

Verification —— checksum

VERIFICATION

Serial data verification methods

Method	Description	Typical Use
Checksum	Adds all byte values in a message and sends the result. Receiver sums bytes and compares to detect errors. Simple and fast.	Modbus RTU, custom protocols
CRC (Cyclic Redundancy Check)	Calculates a polynomial-based value over the data. Stronger than checksum, can detect burst errors.	Modbus RTU, industrial protocols, CAN
LRC (Longitudinal Redundancy Check)	Sums bytes, then takes two's complement. Similar to checksum but slightly more robust.	Modbus ASCII
None	No error detection; fastest but unreliable if data corruption occurs.	High-speed UART where errors are rare

USER MANUAL READING - commend sending frame structure

byte	0	1	2	3~Len-2	Len-1
Description	Head(0x5A)	Len	ID	Payload	Checksum

User manual Page: 8

Please check **Appendix II Serial communication protocol** for more information.

Note: TF-Luna does not enable checksum check for sending data frames by default, that is, the Checksum at the end of the sending frame can be filled with any value..

User manual Page: 9

*Some of the descriptions may not so clearly, you may have to read official examples for better understanding like this one.

Byte 0 : 0x5A - Header bytes. Used to identify the start of a data packet.

Byte 1 : Message length.

Byte 2 : ID of the parameter you want to setup.

Byte 3 : Value you want to set.

Byte 8 : Verification.

USER MANUAL READING - commend sending using Touchdesigner

*The best way to learn is to follow the official examples

Example 01 — working frequency User manual Page: 23

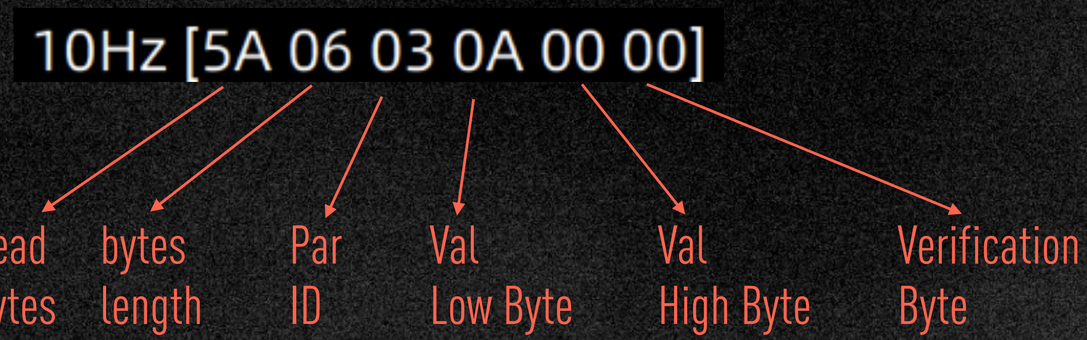
Upward

byte	0	1	2	3~4	Len-1
Description	Head(0x5A)	Len	ID	Freq	Check_sum

Freq: The current working frequency of the LiDAR

Sample instruction:

10Hz [5A 06 03 0A 00 00]



In touchdesigner you can send decimal list

```
n.sendBytes( 90, 6, 3, 10, 0, 0)
```

Or hexadecimal list

```
n.sendBytes( 0x5A, 0x06, 0x03, 0x0A, 0x00, 0x00)
```

Or Byte String literal

```
n.sendBytes(b'\x5A\x06\x03\x0A\x00\x00')
```

USER MANUAL READING - commend sending using Touchdesigner

Example 02 — special functions

Save settings User manual Page: 26

n.sendBytes(0x5A, 0x04, 0x11, 0x00)

Restore default settings User manual Page: 25

n.sendBytes(0x5A, 0x04, 0x11, 0x00)

*Your settings can be saved ,If you forgot your device's baud rate, try change the serial dat's baud rate setting until you see "89 89" as head bytes and if you turn off the data output, just send restore message in different baud rate until you see massage coming in.

Note: Only baud rate in [9600, 921600] are supported.

Example:

9600 [5A 08 06 80 25 00 00 00]

19200 [5A 08 06 00 4B 00 00 00]

38400 [5A 08 06 00 96 00 00 00]

57600 [5A 08 06 00 E1 00 00 00]

115200 [5A 08 06 00 C2 01 00 00]

230400 [5A 08 06 00 84 03 00 00]

460800 [5A 08 06 00 08 07 00 00]

921600 [5A 08 06 00 10 0E 00 00]

User manual Page: 24

Thank YOU

どうもありがとうございます

From 3eyesnuts/YangLei

TOUCHDESIGNER EVENT TOKYO 2025

@3EYESNUTS