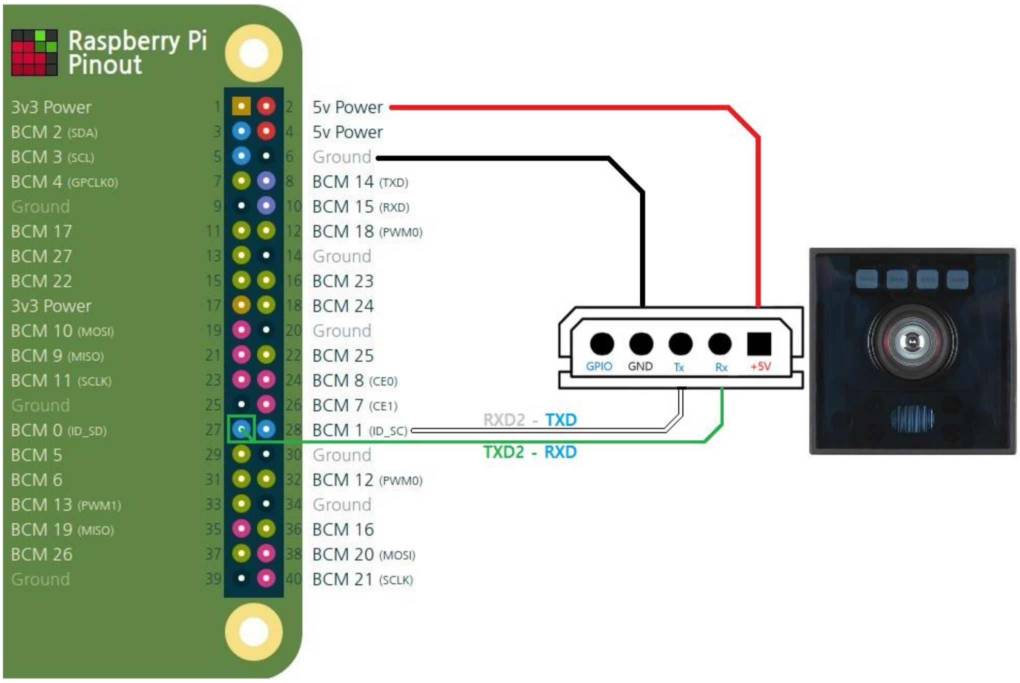
CygLidar Raspberry Pi Guide (EN)

Update: 23-03-07

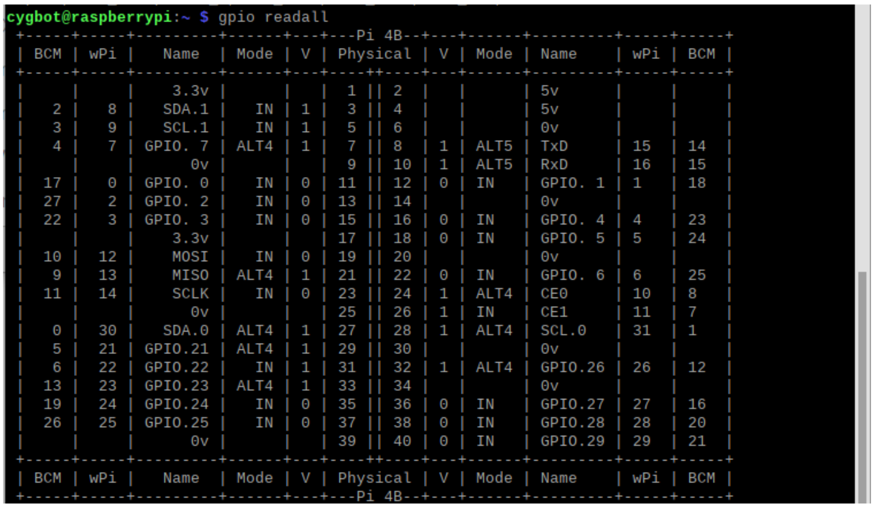
How to connect CygLidar to RaspberryPi (based on Uart2 (ttyAMA1) on RaspberryPi 4B)



Enable GPIO (CMD)

Installation command: sudo apt install raspi-gpio

Confirm installation and Raspberry Pi pin map: gpio readall



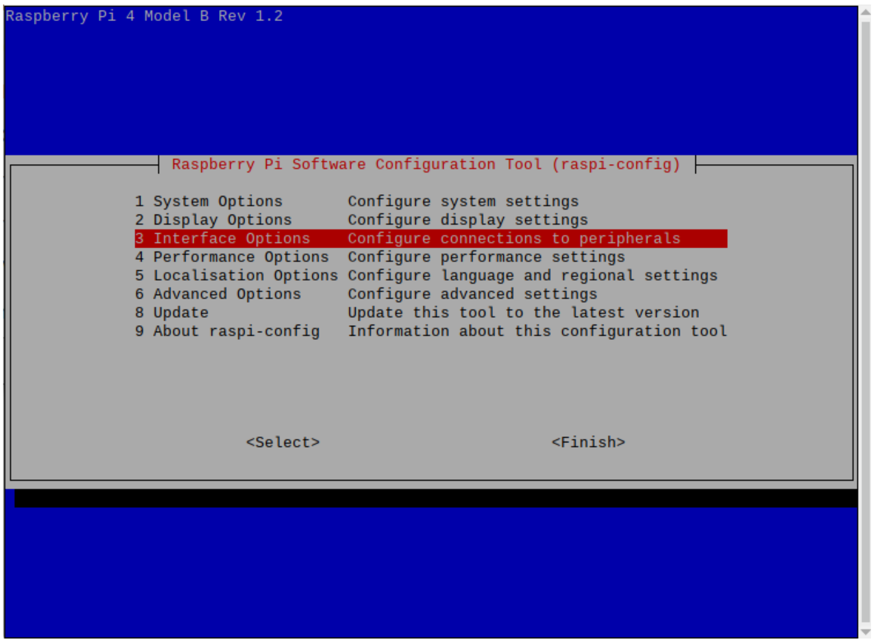
```
cygbot@raspberrypi:~ $ gpio readall
 +-----+-----+---------+------+---+---Pi 4B--+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 |     |     |    3.3v |      |   |  1 || 2  |   |      | 5v      |     |     |
 |   2 |   8 |   SDA.1 |   IN | 1 |  3 || 4  |   |      | 5v      |     |     |
 |   3 |   9 |   SCL.1 |   IN | 1 |  5 || 6  |   |      | 0v      |     |     |
 |   4 |   7 | GPIO. 7 | ALT4 | 1 |  7 || 8  | 1 | ALT5 | TxD     | 15  | 14  |
 |     |     |      0v |      |   |  9 || 10 | 1 | ALT5 | RxD     | 16  | 15  |
 |  17 |   0 | GPIO. 0 |   IN | 0 | 11 || 12 | 0 | IN   | GPIO. 1 | 1   | 18  |
 |  27 |   2 | GPIO. 2 |   IN | 0 | 13 || 14 |   |      | 0v      |     |     |
 |  22 |   3 | GPIO. 3 |   IN | 0 | 15 || 16 | 0 | IN   | GPIO. 4 | 4   | 23  |
 |     |     |    3.3v |      |   | 17 || 18 | 0 | IN   | GPIO. 5 | 5   | 24  |
 |  10 |  12 |    MOSI |   IN | 0 | 19 || 20 |   |      | 0v      |     |     |
 |   9 |  13 |    MISO | ALT4 | 1 | 21 || 22 | 0 | IN   | GPIO. 6 | 6   | 25  |
 |  11 |  14 |    SCLK |   IN | 0 | 23 || 24 | 1 | ALT4 | CE0     | 10  | 8   |
 |     |     |      0v |      |   | 25 || 26 | 1 | IN   | CE1     | 11  | 7   |
 |   0 |  30 |   SDA.0 | ALT4 | 1 | 27 || 28 | 1 | ALT4 | SCL.0   | 31  | 1   |
 |   5 |  21 | GPIO.21 | ALT4 | 1 | 29 || 30 |   |      | 0v      |     |     |
 |   6 |  22 | GPIO.22 |   IN | 1 | 31 || 32 | 1 | ALT4 | GPIO.26 | 26  | 12  |
 |  13 |  23 | GPIO.23 | ALT4 | 1 | 33 || 34 |   |      | 0v      |     |     |
 |  19 |  24 | GPIO.24 |   IN | 0 | 35 || 36 | 0 | IN   | GPIO.27 | 27  | 16  |
 |  26 |  25 | GPIO.25 |   IN | 0 | 37 || 38 | 0 | IN   | GPIO.28 | 28  | 20  |
 |     |     |      0v |      |   | 39 || 40 | 0 | IN   | GPIO.29 | 29  | 21  |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+---Pi 4B--+---+------+---------+-----+-----+
```

Serial activation settings (CMD or Raspberry pi configuration)

1. Setting using CMD

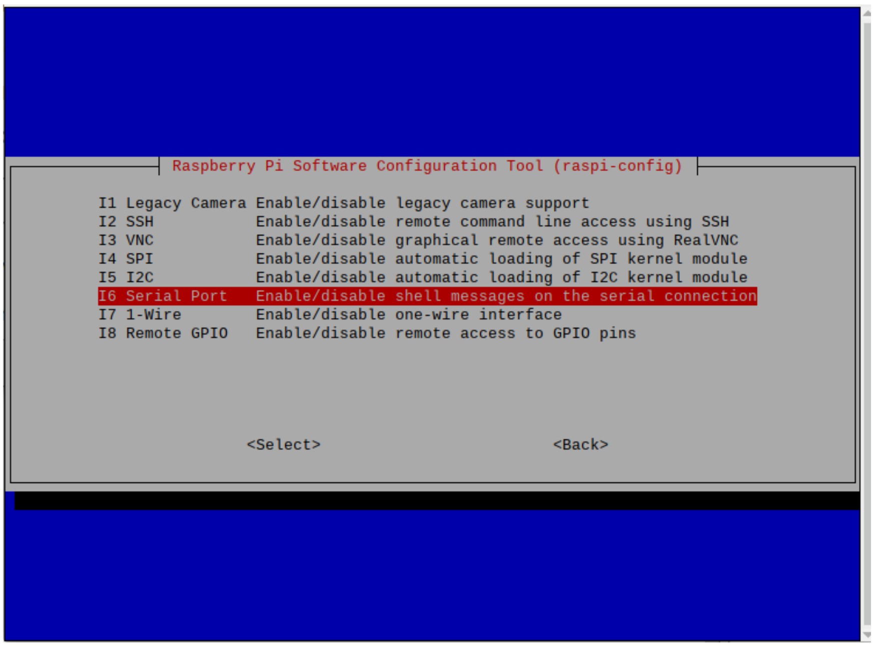
   A. Enter the sudo raspi-config

   B. Interface option Select or enter the



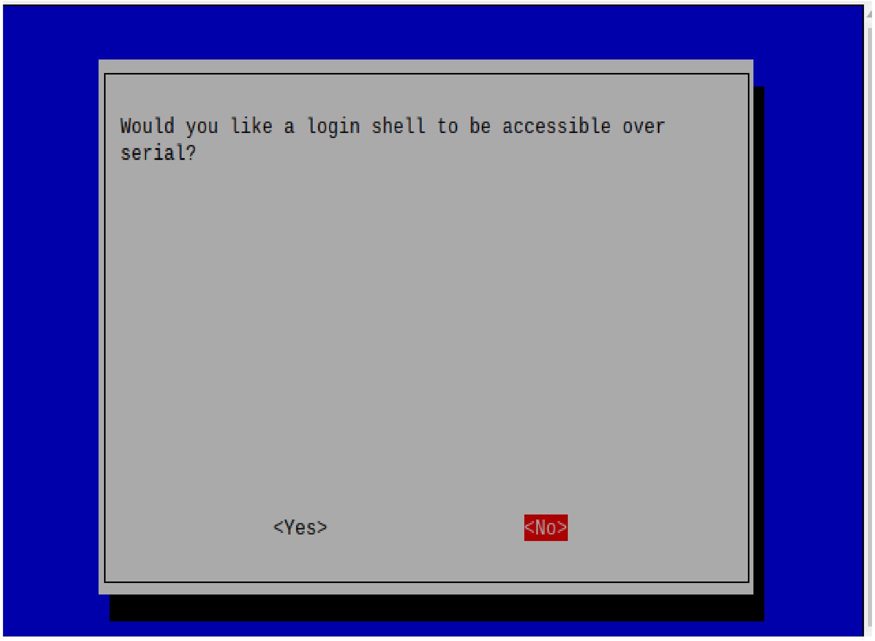
```
Raspberry Pi 4 Model B Rev 1.2

              Raspberry Pi Software Configuration Tool (raspi-config)

      1 System Options       Configure system settings
      2 Display Options      Configure display settings
      3 Interface Options    Configure connections to peripherals
      4 Performance Options  Configure performance settings
      5 Localisation Options Configure language and regional settings
      6 Advanced Options     Configure advanced settings
      8 Update               Update this tool to the latest version
      9 About raspi-config   Information about this configuration tool

                 <Select>                         <Finish>
```

   C. Serial port Select or enter the



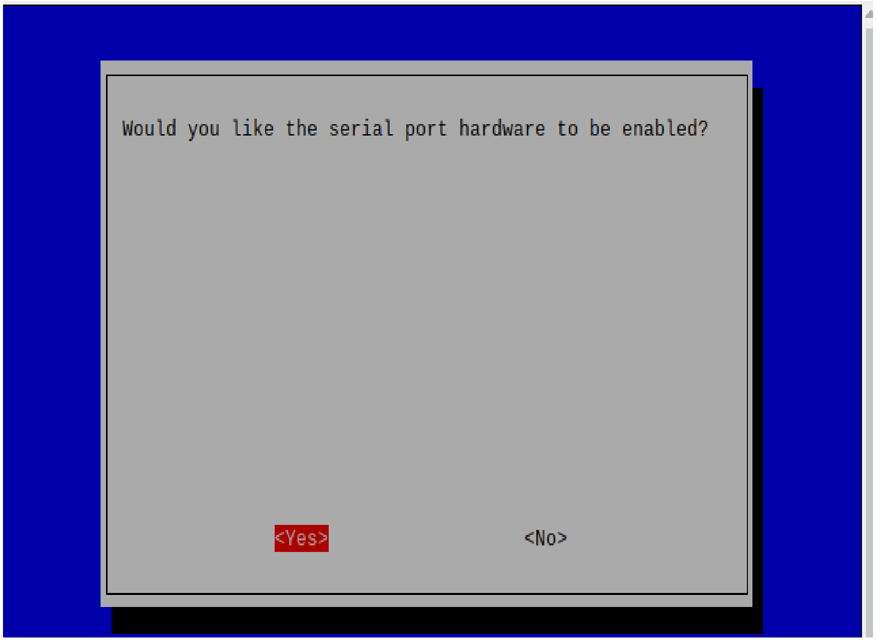
```
              Raspberry Pi Software Configuration Tool (raspi-config)

    I1 Legacy Camera Enable/disable legacy camera support
    I2 SSH           Enable/disable remote command line access using SSH
    I3 VNC           Enable/disable graphical remote access using RealVNC
    I4 SPI           Enable/disable automatic loading of SPI kernel module
    I5 I2C           Enable/disable automatic loading of I2C kernel module
    I6 Serial Port   Enable/disable shell messages on the serial connection
    I7 1-Wire        Enable/disable one-wire interface
    I8 Remote GPIO   Enable/disable remote access to GPIO pins

                 <Select>                         <Back>
```

D. serial console no check



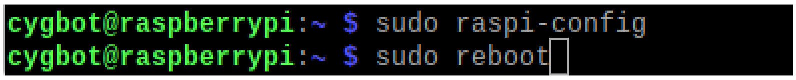
Would you like a login shell to be accessible over serial?

E. Serial port yes check



Would you like the serial port hardware to be enabled?

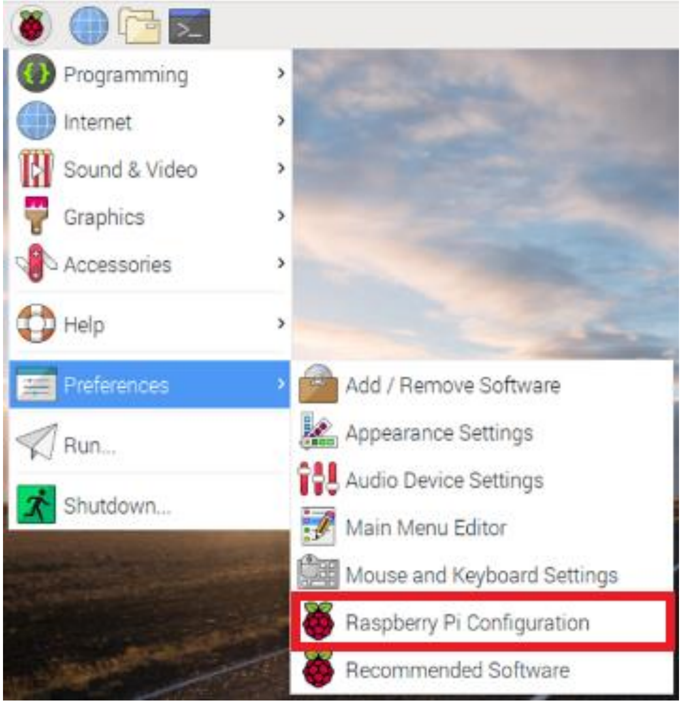
F. CMD – enter the sudo reboot



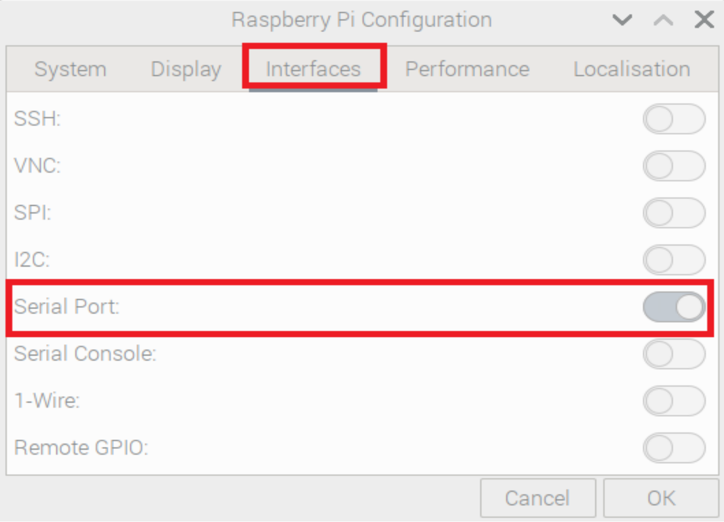
```
cygbot@raspberrypi:~ $ sudo raspi-config
cygbot@raspberrypi:~ $ sudo reboot
```

2. Setting using Raspberry Pi Configuration

A. Raspbian icon – Preferences – Raspberry Pi Configuration



B. Interfaces – Serial port on



SSH:

VNC:

SPI:

I2C:

Serial Port:

Serial Console:

1-Wire:

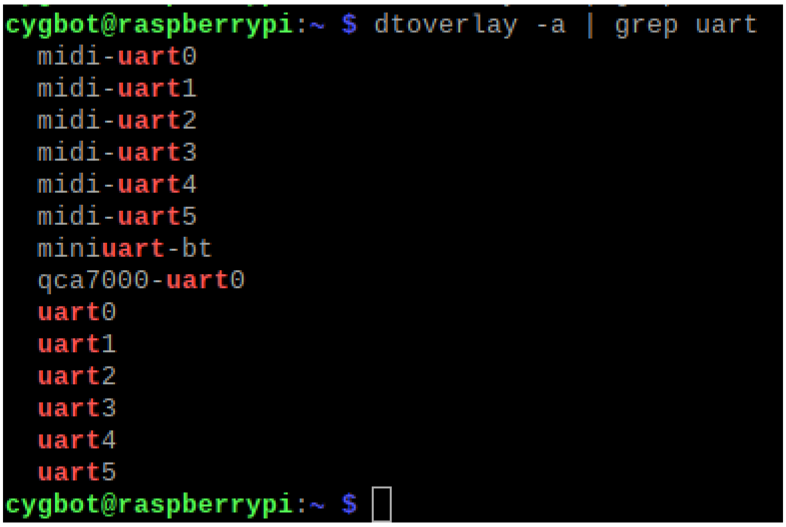
Remote GPIO:

C. Logout - Reboot

Check and set Uart

1. Uart port Check(CMD)

A. Enter the dtoverlay –a | grep uart



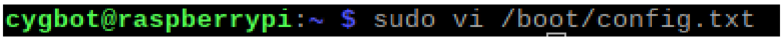
```
cygbot@raspberrypi:~ $ dtoverlay -a | grep uart
  midi-uart0
  midi-uart1
  midi-uart2
  midi-uart3
  midi-uart4
  midi-uart5
  miniuart-bt
  qca7000-uart0
  uart0
  uart1
  uart2
  uart3
  uart4
  uart5
cygbot@raspberrypi:~ $ 
```

Only Uart2 ~ 5 can be used among all Uart (Uart 1 is for hardware only, Uart 2 is a mini Uart connected with bluetooth, so it cannot be used)

2. Uart port activation settings

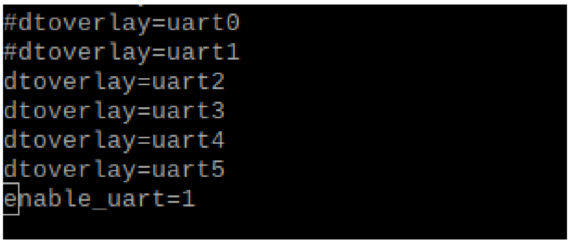
A. Enter the sudo vi /boot/config.txt

Put the editor name you want in the vi part ex) nano etc.

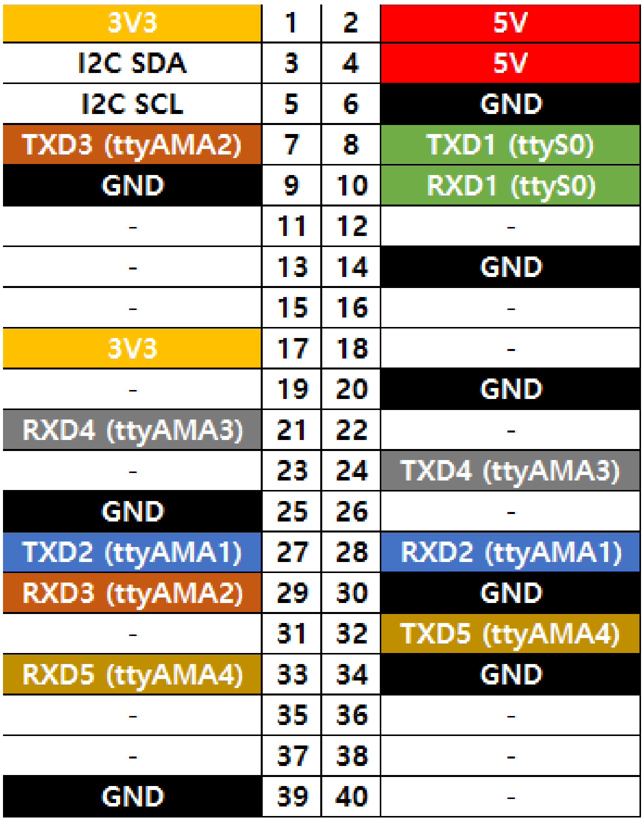


```
cygbot@raspberrypi:~ $ sudo vi /boot/config.txt
```

B. Write the code below the last [all] line by checking the available Uart ports as shown in the picture



```
#dtoverlay=uart0
#dtoverlay=uart1
dtoverlay=uart2
dtoverlay=uart3
dtoverlay=uart4
dtoverlay=uart5
enable_uart=1
```



3V3	1	2	5V
I2C SDA	3	4	5V
I2C SCL	5	6	GND
TXD3 (ttyAMA2)	7	8	TXD1 (ttyS0)
GND	9	10	RXD1 (ttyS0)
-	11	12	-
-	13	14	GND
-	15	16	-
3V3	17	18	-
-	19	20	GND
RXD4 (ttyAMA3)	21	22	-
-	23	24	TXD4 (ttyAMA3)
GND	25	26	-
TXD2 (ttyAMA1)	27	28	RXD2 (ttyAMA1)
RXD3 (ttyAMA2)	29	30	GND
-	31	32	TXD5 (ttyAMA4)
RXD5 (ttyAMA4)	33	34	GND
-	35	36	-
-	37	38	-
GND	39	40	-

Uart0 = Debug console is connected by default. (It is set as hardware Uart, so use is prohibited)

Uart1 = ttyAMA0/ttyS0 = Bluetooth connection by default. (It is set as mini Uart, so use is prohibited)

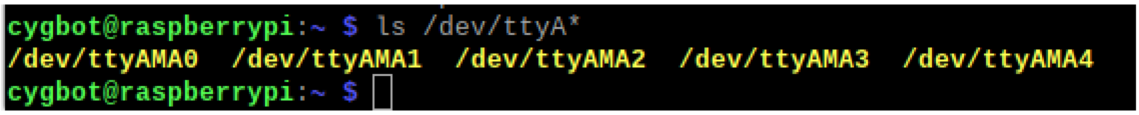
Uart2 = ttyAMA1 (Available)

Uart3 = ttyAMA2 (Available)

Uart4 = ttyAMA3 (Available)

Uart5 = ttyAMA4 (Available)

C. Enter the ls /dev/ttyA* to see which UART is active

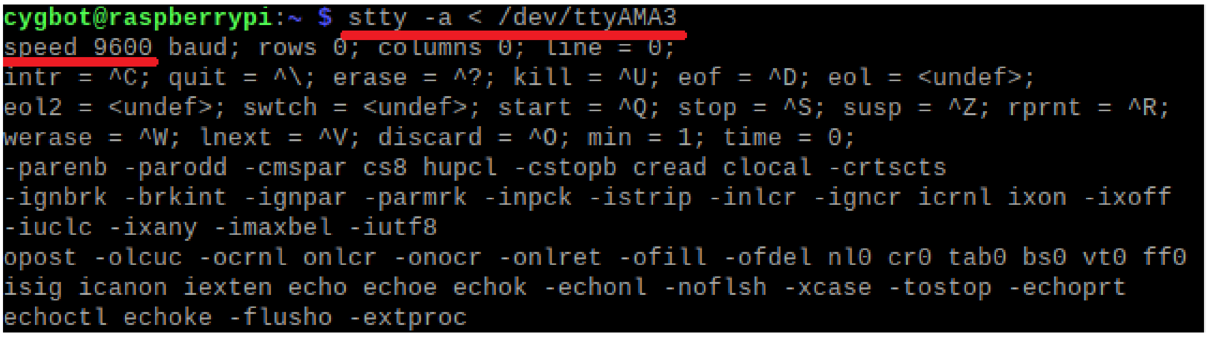


```
cygbot@raspberrypi:~ $ ls /dev/ttyA*
/dev/ttyAMA0  /dev/ttyAMA1  /dev/ttyAMA2  /dev/ttyAMA3  /dev/ttyAMA4
cygbot@raspberrypi:~ $ 
```

Check and set Uart Port speed

1. Uart Port speed Check

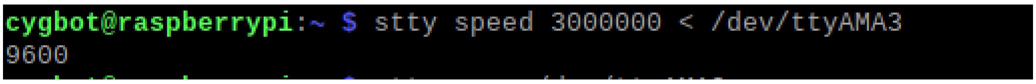
A. CMD – enter the stty –a < /dev/ttyAMA* (Enter the the port number you use for *)



```
cygbot@raspberrypi:~ $ stty -a < /dev/ttyAMA3
speed 9600 baud; rows 0; columns 0; line = 0;
intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D; eol = <undef>;
eol2 = <undef>; swtch = <undef>; start = ^Q; stop = ^S; susp = ^Z; rprnt = ^R;
werase = ^W; lnext = ^V; discard = ^O; min = 1; time = 0;
-parenb -parodd -cmspar cs8 hupcl -cstopb cread clocal -crtscts
-ignbrk -brkint -ignpar -parmrk -inpck -istrip -inlcr -igncr icrnl ixon -ixoff
-iuclc -ixany -imaxbel -iutf8
opost -olcuc -ocrnl onlcr -onocr -onlret -ofill -ofdel nl0 cr0 tab0 bs0 vt0 ff0
isig icanon iexten echo echoe echok -echonl -noflsh -xcase -tostop -echoprt
echoctl echoke -flusho -extproc
```

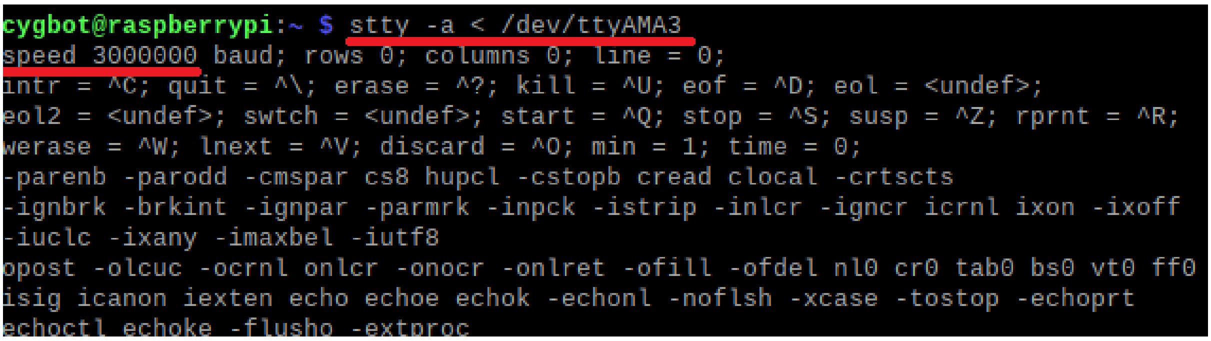
B. After checking the speed enter the stty speed 3000000 < /dev/ttyAMA*

(python recommend 250,000)



```
cygbot@raspberrypi:~ $ stty speed 3000000 < /dev/ttyAMA3
9600
```

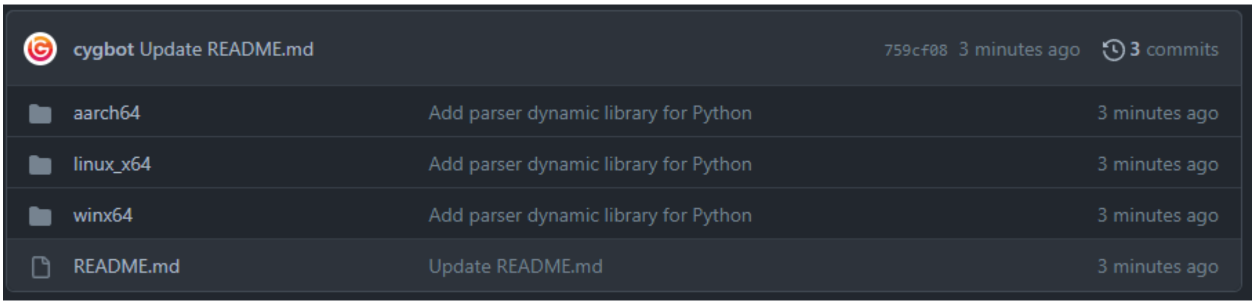
C. Enter the stty –a < /dev/ttyAMA* and check the changed speed



```
cygbot@raspberrypi:~ $ stty -a < /dev/ttyAMA3
speed 3000000 baud; rows 0; columns 0; line = 0;
intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D; eol = <undef>;
eol2 = <undef>; swtch = <undef>; start = ^Q; stop = ^S; susp = ^Z; rprnt = ^R;
werase = ^W; lnext = ^V; discard = ^O; min = 1; time = 0;
-parenb -parodd -cmspar cs8 hupcl -cstopb cread clocal -crtscts
-ignbrk -brkint -ignpar -parmrk -inpck -istrip -inlcr -igncr icrnl ixon -ixoff
-iuclc -ixany -imaxbel -iutf8
opost -olcuc -ocrnl onlcr -onocr -onlret -ofill -ofdel nl0 cr0 tab0 bs0 vt0 ff0
isig icanon iexten echo echoe echok -echonl -noflsh -xcase -tostop -echoprt
echoctl echoke -flusho -extproc
```

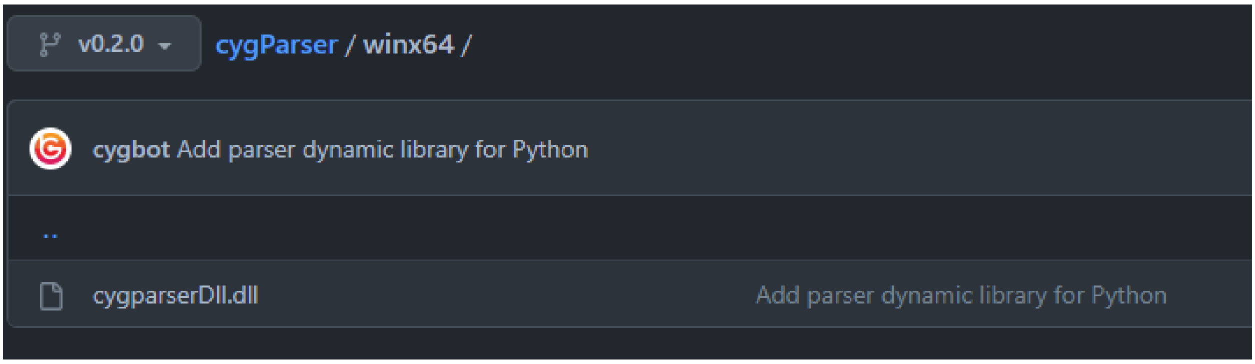
Example code github

Dynamic library Download link : https://github.com/cygbot/cyglidarPython
Git clone : https://github.com/cygbot/cyglidarPython.git



cygbot Update README.md 759cf08 3 minutes ago 3 commits

aarch64	Add parser dynamic library for Python	3 minutes ago
linux_x64	Add parser dynamic library for Python	3 minutes ago
winx64	Add parser dynamic library for Python	3 minutes ago
README.md	Update README.md	3 minutes ago



v0.2.0 cygParser / winx64 /

cygbot Add parser dynamic library for Python

..	
cygparserDll.dll	Add parser dynamic library for Python

Example1

```
import time
import serial

RUN_2D = [0x5A, 0x77, 0xFF, 0x02, 0x00, 0x01, 0x00, 0x03]
RUN_3D = [0x5A, 0x77, 0xFF, 0x02, 0x00, 0x08, 0x00, 0x0A]
RUN_DUAL = [0x5A, 0x77, 0xFF, 0x02, 0x00, 0x07, 0x00, 0x05]
COMMAND_STOP = [0x5A, 0x77, 0xFF, 0x02,0x00,0x02,0x00,0x00]

HEADER1, HEADER2, HEADER3, LENGTH_LSB, LENGTH_MSB, PAYLOAD_HEADER, PAYLOAD_DATA, 
CHECKSUM = 0, 1, 2, 3, 4, 5, 6, 7
POS_CYGBOT_HEADER, POS_DEVICE, POS_ID, POS_LENGTH_1, POS_LENGTH_2, 
POS_PAYLOAD_HEADER = 0, 1, 2, 3, 4, 5
PAYLOAD_POS_HEADER, PAYLOAD_POS_DATA = 0, 1
NORMAL_MODE = 0x5A
PRODUCT_CODE = 0x77
DEFAULT_ID = 0xFF
HEADER_LENGTH_SIZE = 5

buffercounter, CPC, lengthLSB, lengthMSB, data_length = 0, 0, 0, 0, 0
step = HEADER1
receivedData = []



def Parser(data):
    global step, CPC, lengthLSB, lengthMSB, data_length, buffercounter, receivedData
    if step != CHECKSUM:  # CPC is a variable for storing checksum. If it is not a 
checksum part, XOR operation is performed on each data and then stored.
        CPC = CPC ^ data

    if step == HEADER1 and data == NORMAL_MODE:
        step = HEADER2

    elif step == HEADER2 and data == PRODUCT_CODE:
        step = HEADER3

    elif step == HEADER3 and data == DEFAULT_ID:
        step = LENGTH_LSB
        CPC = 0

    elif step == LENGTH_LSB:
        step = LENGTH_MSB
        lengthLSB = data

    elif step == LENGTH_MSB:
        step = PAYLOAD_HEADER
        lengthMSB = data
        data_length = ((lengthMSB << 8) & 0xff00) | (lengthLSB & 0x00ff)

    elif step == PAYLOAD_HEADER:
        step = PAYLOAD_DATA
        if data_length == 1:
            step = CHECKSUM
        buffercounter = 0
        receivedData = []

    elif step == PAYLOAD_DATA:
        receivedData.append(data)
        buffercounter = buffercounter+1
        if buffercounter >= data_length - 1:
            step = CHECKSUM

    elif step == CHECKSUM:
        step = HEADER1

        if CPC == data:
            return True
    else:
        step = HEADER1
        return False

ser = serial.Serial(  # Port settings
    port= '/dev/ttyAMA1',
    baudrate=3000000, # recommend 250,000
    parity=serial.PARITY_NONE,
    stopbits=serial.STOPBITS_ONE,
    bytesize=serial.EIGHTBITS
)
ser.write(RUN_2D) # Mode settings
print("send : ", RUN_2D)
time.sleep(1)

while True:
    try:
        readdata = ser.readline()
        for i in range(len(readdata)):
            if Parser(readdata[i]):
                print(len(receivedData))


    except KeyboardInterrupt:
        ser.write(COMMAND_STOP)
        ser.close()
```

Example2(for example, OpenCV 3D data visualize)

```
import serial
import cv2
import numpy as np

RUN_3D       =  [0x5A, 0x77, 0xFF, 0x02, 0x00, 0x08, 0x00, 0x0A]
COMMAND_STOP =  [0x5A, 0x77, 0xFF, 0x02, 0x00, 0x02, 0x00, 0x00]

HEADER1, HEADER2, HEADER3, LENGTH_LSB, LENGTH_MSB, PAYLOAD_HEADER, PAYLOAD_DATA,
CHECKSUM = 0, 1, 2, 3, 4, 5, 6, 7
NORMAL_MODE = 0x5A
PRODUCT_CODE = 0x77
DEFAULT_ID = 0xFF

normalizeDistanceLimit = 4080
dataLength3D = 14400

def ReceivedCompleteData(receivedData):
    global dataLength3D
    print(f'receive complete data : {len(receivedData)}')
    if len(receivedData) == dataLength3D:
        Visualize(receivedData)


def Visualize(receivedData):
    distanceData = Get3DDistanceDataFromReceivedData(receivedData)
    image = DistanceDataToNormalizedNumpyArray(distanceData)
    image = np.array(image, dtype=np.uint8)
    image = image.reshape(60, 160)
    image = cv2.resize(image, dsize=(480, 180), interpolation=cv2.INTER_NEAREST)
    cv2.imshow('test', image)
    cv2.waitKey(1)


def Get3DDistanceDataFromReceivedData(receivedData):
    global dataLength3D,normalizeDistanceLimit
    index = 0
    distanceData = [0 for i in range(int(dataLength3D / 3 * 2))]
    for i in range(0, dataLength3D-2, 3):
        pixelFirst = receivedData[i] << 4 | receivedData[i+1] >> 4
        pixelSecond = (receivedData[i+1] & 0xf) << 8 | receivedData[i+2]

        if pixelFirst > normalizeDistanceLimit:
            pixelFirst = normalizeDistanceLimit
        if pixelSecond > normalizeDistanceLimit:
            pixelSecond = normalizeDistanceLimit

        distanceData[index] = pixelFirst
        index += 1
        distanceData[index] = pixelSecond
        index += 1
    return distanceData

def DistanceDataToNormalizedNumpyArray(distanceData):
    global normalizeDistanceLimit
    result = np.array(distanceData)
    result = result / normalizeDistanceLimit * 255
    return result


#baud = 57600
#baud = 115200
baud = 250000
#baud = 3000000  # recommend baudrate under 3,000,000
ser = serial.Serial(  # port open
    #port="/dev/ttyUSB0",  # <- USB connection
    '/dev/ttyAMA1',# <- GPIO connection
    # "COM14", #<- Windows PC
    baudrate=baud,
    parity=serial.PARITY_NONE,
    stopbits=serial.STOPBITS_ONE,
    bytesize=serial.EIGHTBITS
)
if __name__ == "__main__":
    ser.write(RUN_3D)
    print("send : ", RUN_3D)
    step = HEADER1
    CPC = 0

    bufferCounter = 0
    receivedData = [0 for i in range(dataLength3D)]
    while True:
        try:
            for byte in ser.readline():
                parserPassed = False
                 # Parse Start
                if step != CHECKSUM:
                    CPC = CPC ^ byte
                if step == PAYLOAD_DATA:
                    receivedData[bufferCounter] = byte
                    bufferCounter += 1
                    if bufferCounter >= dataLength :
                        step = CHECKSUM
                elif step == HEADER1 and byte == NORMAL_MODE:
                    step = HEADER2
                elif step == HEADER2 and byte == PRODUCT_CODE:
                    step = HEADER3
                elif step == HEADER3 and byte == DEFAULT_ID:
                    step = LENGTH_LSB
                    CPC = 0
                elif step == LENGTH_LSB:
                    step = LENGTH_MSB
                    lengthLSB = byte
                elif step == LENGTH_MSB:
                    step = PAYLOAD_HEADER
                    lengthMSB = byte
                    dataLength = (lengthMSB << 8)  | lengthLSB  - 1
                elif step == PAYLOAD_HEADER:
                    step = PAYLOAD_DATA
                    if dataLength == 0:
                        step = CHECKSUM
                    bufferCounter = 0
                    receivedData = [0 for i in range(dataLength)]  # clear
                elif step == CHECKSUM:
                    step = HEADER1
                    if CPC == byte:
                        parserPassed = True
                else:
                    step = HEADER1
                    parserPassed = False
                # Parse End

                if parserPassed:
                    ReceivedCompleteData(receivedData)
        except KeyboardInterrupt:
            ser.write(COMMAND_STOP)
            ser.close()
```

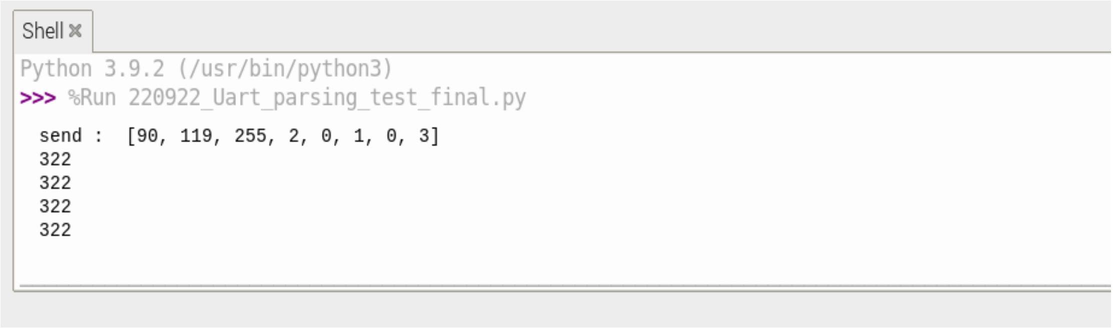
Example 3(for example, use parser library ,OpenCV 3D data visualize)

```
import ctypes

import platform

import serial
import cv2
import numpy as np


RUN_2DMode = [0x5A, 0x77, 0xFF, 0x02, 0x00, 0x01, 0x00, 0x03]
RUN_3DMode = [0x5A, 0x77, 0xFF, 0x02, 0x00, 0x08, 0x00, 0x0A]
RUN_DualMode = [0x5A, 0x77, 0xFF, 0x02, 0x00, 0x07, 0x00, 0x05]
COMMAND_STOP = [0x5A, 0x77, 0xFF, 0x02, 0x00, 0x02, 0x00, 0x00]
dataLength3D = 14400
dataLength2D = 322
normalizeDistanceLimit = 4080


def makeCtypesPacket(pythonPacket):
    return ((ctypes.c_uint8 * len(pythonPacket))(*pythonPacket),len(pythonPacket))

def Visualize(resultlist, dataLength):
    if dataLength == dataLength3D:
        distanceData = Get3DDistanceDataFromReceivedData(resultlist)
        image = DistanceDataToNormalizedNumpyArray(distanceData)
        image = np.array(image, dtype=np.uint8)
        image = image.reshape(60, 160)
        image = cv2.resize(image, dsize=(480, 180), interpolation=cv2.INTER_NEAREST)
        cv2.imshow('test', image)
        cv2.waitKey(1)
    #elif dataLength == dataLength2D:
        # type your code

def Get3DDistanceDataFromReceivedData(receivedData):
    global dataLength3D,normalizeDistanceLimit
    index = 0
    distanceData = [0 for i in range(int(dataLength3D / 3 * 2))]
    for i in range(0, dataLength3D-2, 3):
        pixelFirst = receivedData[i] << 4 | receivedData[i+1] >> 4
        pixelSecond = (receivedData[i+1] & 0xf) << 8 | receivedData[i+2]
        if pixelFirst > normalizeDistanceLimit:
            pixelFirst = normalizeDistanceLimit
        if pixelSecond > normalizeDistanceLimit:
            pixelSecond = normalizeDistanceLimit
        distanceData[index] = pixelFirst
        index += 1
        distanceData[index] = pixelSecond
        index += 1
    return distanceData

def DistanceDataToNormalizedNumpyArray(distanceData):
    global normalizeDistanceLimit
    result = np.array(distanceData)
    result = result / normalizeDistanceLimit * 255
    return result

current_system = platform.system()
if current_system == 'Windows':
    #path = 'C:\\Users\\cygbot\\cygparserDll.dll'
    path=''
    c_module = ctypes.windll.LoadLibrary(path)
elif current_system == 'Linux':
    #path = '/home/cygbot/cygparser.so'
    path = ''
    c_module = ctypes.cdll.LoadLibrary(path)
else:
    raise OSError()




getPayloadSize = c_module.getPayloadSize
getParserPassed = c_module.getParserPassed
Parser = c_module.Parser


Parser.argtypes = (ctypes.POINTER(ctypes.c_uint8),ctypes.c_uint16)
Parser.restype = ctypes.POINTER(ctypes.c_uint8 * 15000)

getPayloadSize.argtypes = None
getPayloadSize.restype = ctypes.c_uint16

getParserPassed.argtypes = None
getParserPassed.restype = ctypes.c_uint8

ser = serial.Serial(  # port open
    #port="/dev/ttyUSB0", # <- USB connection
    #'/dev/ttyAMA1',# <- GPIO connection
     "COM17", #<- Windows PC
    #baudrate=3000000,
    baudrate=250000, # recommend 250,000
    parity=serial.PARITY_NONE,
    stopbits=serial.STOPBITS_ONE,
    bytesize=serial.EIGHTBITS
)
if __name__ == "__main__":
```

```
#runMode = RUN_2DMode
runMode = RUN_3DMode
#runMode = RUN_DualMode
ser.write(runMode)

print("send : ", runMode)
while True:
    try:
        read_lines = ser.readline()
        read_lines_ctypes = makeCtypesPacket([x for x in read_lines])
        parserResult = Parser(read_lines_ctypes[0],read_lines_ctypes[1])

        resultlist=[]

        if getParserPassed() == 1:
            print('parser passed')
            resultlist = [x for x in parserResult.contents]
            Visualize(resultlist,getPayloadSize())
    except KeyboardInterrupt:
        ser.write(COMMAND_STOP)
        ser.close()
```

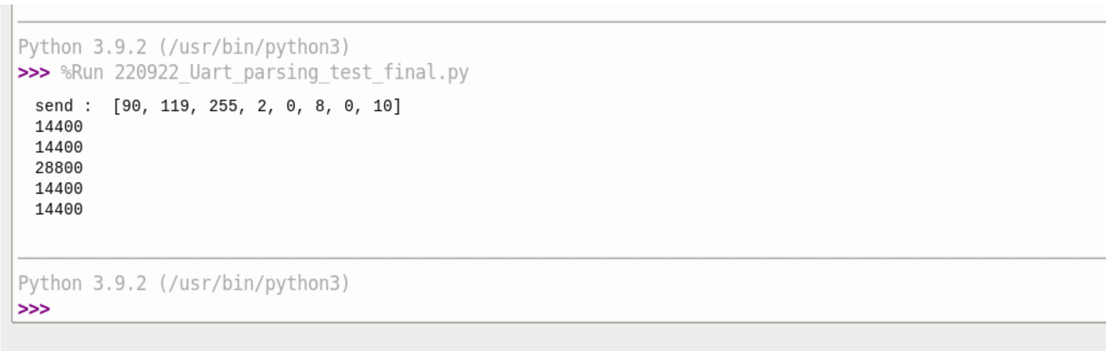
Result - Test code 1

1. RUN_2D Mode



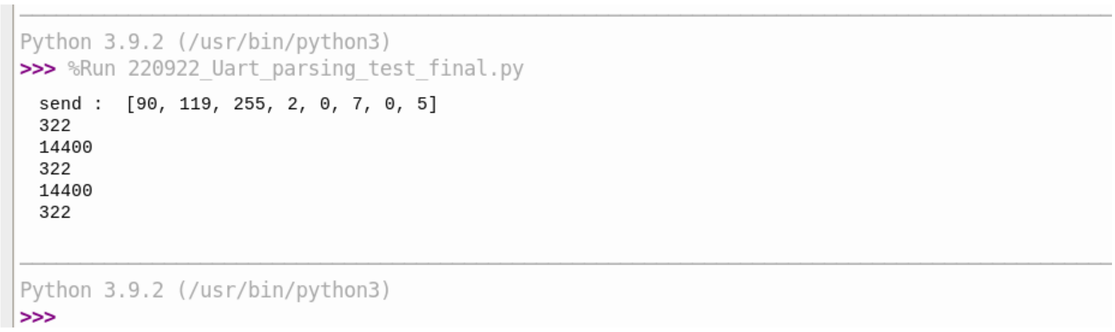
```
Python 3.9.2 (/usr/bin/python3)
>>> %Run 220922_Uart_parsing_test_final.py

  send :  [90, 119, 255, 2, 0, 1, 0, 3]
  322
  322
  322
  322
```

2. RUN_3D Mode



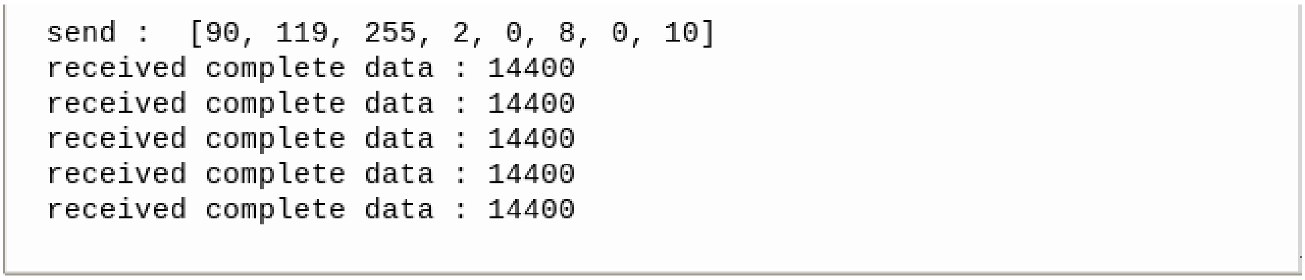
```
Python 3.9.2 (/usr/bin/python3)
>>> %Run 220922_Uart_parsing_test_final.py

  send :  [90, 119, 255, 2, 0, 8, 0, 10]
  14400
  14400
  28800
  14400
  14400

Python 3.9.2 (/usr/bin/python3)
>>>
```

3. RUN_Dual Mode



```
Python 3.9.2 (/usr/bin/python3)
>>> %Run 220922_Uart_parsing_test_final.py

  send :  [90, 119, 255, 2, 0, 7, 0, 5]
  322
  14400
  322
  14400
  322

Python 3.9.2 (/usr/bin/python3)
>>>
```

Result - Test code 2



```
send :  [90, 119, 255, 2, 0, 8, 0, 10]
received complete data : 14400
received complete data : 14400
received complete data : 14400
received complete data : 14400
received complete data : 14400
```