The Examples of TFmini Plus on Arduino （Ⅱ）

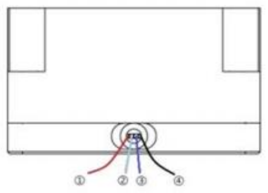
This application takes Uno board of Arduino as an example, writing related instructions of TFmini Plus, reading response data from LIDAR, processing and printing measurement data through Arduino, which helps customers to quickly familiarize themselves with our company's product and reduce development cycle.

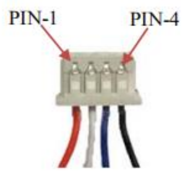
For a detailed introduction of Arduino, please refer to the following website:

official website： www.arduino.cc

Step 1: Hardware connection

As shown in Figure 1, it is the sequence of TFmini Plus lidar. TFmini Plus uses + 5V power supply and can directly connect to 5V and GND of UNO board. It adopts software serial communication mode. The pins of (2,3) of UNO board are defined as soft serial port (RX, TX). The system wiring is shown in Figure 2.





No.	Color	Corresponding PIN	Funciton	Comment
①	Red	PIN-1	+5V	Power supply
②	White	PIN-2	RXD/SDA	Receiving/Data
③	Blue/Green	PIN-3	TXD/SCL/IO	Transmitting/Clock/IO
④	Black	PIN-4	GND	Ground

Fig. 1 The line sequence of TFmini Plus

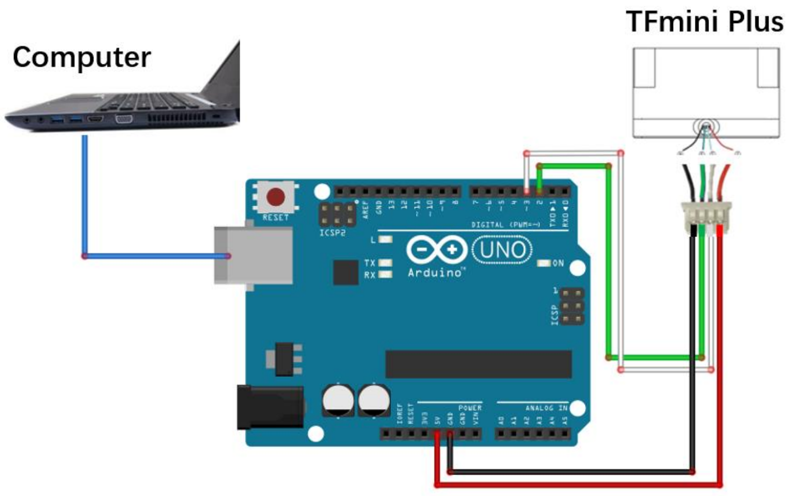


Fig.2 Wiring of TFmini Plus and UNO board

Step 2: Programming

The implementation of this example requires at least two serial ports of Arduino, one is for receiving lidar data and the other is for displaying data to the computer. The following code can be copied and pasted into the IDE command window.

```
#include<SoftwareSerial.h>//header file of software serial port

SoftwareSerial Serial1(2,3); //define software serial port name as Serial1 and define pin2 as RX and pin3 as TX


//TFmini Plus related configuration instructions.There are no instructions on display here, you can refer to the TFmini Plus manual to add by yourself. Note: This routine uses serial port to communicate.

char getversion[4]={0x5a,0x04,0x01,0x5f};//Get firmware version

char reset[4]={0x5a,0x04,0x02,0x60};// System Reset

char enable[5]={0x5a,0x05,0x07,0x00,0x66};//Disable data output

char disable[5]={0x5a,0x05,0x07,0x01,0x67};//Enable data output

char UART[5]={0x5a,0x05,0x0a,0x00,0x69};//Modify communication mode to UART
char I2C[5]={0x5a,0x05,0x0a,0x01,0x6a};//Modify communication mode to I2C

char samplerate_01[6]={0x5a,0x06,0x03,0x01,0x00,0x64};//Set the frame rate  to 1Hz

char samplerate_10[6]={0x5a,0x06,0x03,0x0a,0x00,0x6d};//Set the frame rate  to 10Hz

char samplerate_100[6]={0x5a,0x06,0x03,0x64,0x00,0xc7};//Set the frame rate  to 100Hz

char factoryreset[4]={0x5a,0x04,0x10,0x6e};//Restore factory settings

char save[4]={0x5a,0x04,0x11,0x6f};//save


//The response of TFmini Plus. Note: Output frame rate, output enable switch, return command;
Modify communication mode, no response, execute directly

//The response after getting firmware versions

int return_version[7]={0};

//The response after setting samplerate

int return_samplerate[6]={0};

//The response after setting enable switch

int return_switch[5]={0};

//The response after resetting

int return_reset[5]={0};

char reset_success[5]={0x5a,0x05,0x02,0x00,0x60};

char reset_fail[5]={0x5a,0x05,0x02,0x01,0x61};

//The response after restoring factory settings

int return_factoryreset[5]={0};

char factoryreset_success[5]={0x5a,0x05,0x10,0x00,0x6e};

char factoryreset_fail[5]={0x5a,0x05,0x10,0x01,0x6f};

//The response after saving

int return_save[5]={0};

char save_success[5]={0x5a,0x05,0x11,0x00,0x70};
char save_fail[5]={0x5a,0x05,0x11,0x01,0x71};

//prompt information

String info_getversion="get version ok";

String info_reset="reset ok";

String info_enable="enable ok";

String info_disable="get version ok";

String info_UART="UART ok";

String info_I2C="I2C ok";

String info_samplerate_01="samplerate_01 ok";

String info_samplerate_10="samplerate_10 ok";

String info_samplerate_100="samplerate_100 ok";

String info_factoryreset="factoryreset ok";

String info_save="save ok";
```

//The parameter configuration function, down is the instruction to be written,n1 is the number of downs,buff receives response data,up receives theoretical response data in communication protocols,n2 is the number of buff data,info is the prompt information of corresponding instructions

```
void configure(char down[],int n1,int buff[],char up[],int n2,String info);
```

//used to verify whether the response is consistent with the protocol, mainly transmit the following instructions 1、reset 2、factory reset3、save configuration 4、sampling rate set 5、data output enable/disable

```
void configure(char down[],int n1,int buff[],int n2,String info);
```

//used for the instruction which doesn’ t need the comparison of response and protocol

1、gain the firmware version

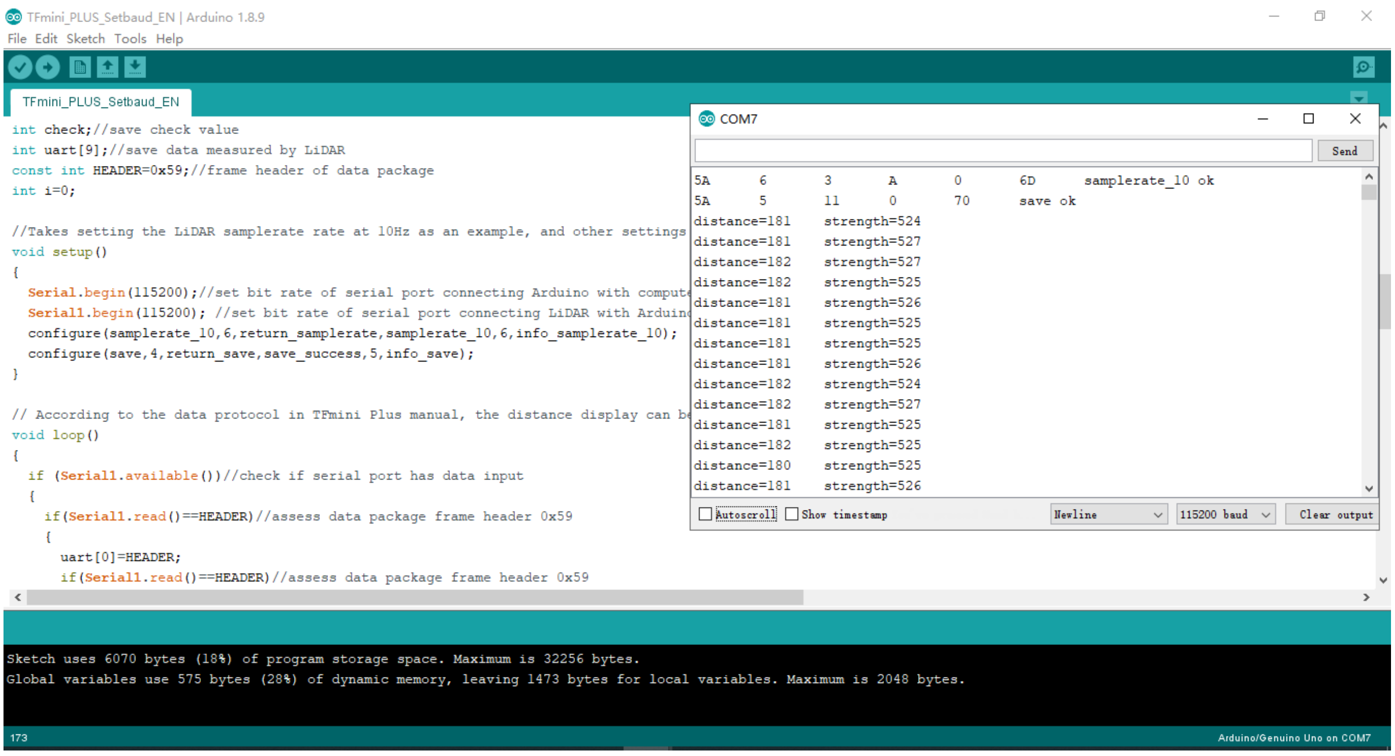
```
void configure(char down[],int n1,String info);
```

```
//used to alter the communication protocol, UART or I2C


// Define the parameters of receiving LiDAR data
int dist;//actual distance measurements of LiDAR
int strength;//signal strength of LiDAR
int check;//save check value
int uart[9];//save data measured by LiDAR
const int HEADER=0x59;//frame header of data package
int i=0;


//Takes setting the LiDAR samplerate rate at 10Hz as an example, and other settings can be modified accordingly.
void setup()
{
  Serial.begin(115200);//set bit rate of serial port connecting Arduino with computer
  Serial1.begin(115200); //set bit rate of serial port connecting LiDAR with Arduino
  configure(samplerate_10,6,return_samplerate,samplerate_10,6,info_samplerate_10);
  configure(save,4,return_save,save_success,5,info_save);
}


// According to the data protocol in TFmini Plus manual, the distance display can be obtained and displayed through serial terminal.
void loop()
{
  if (Serial1.available())//check if serial port has data input
  {
      if(Serial1.read()==HEADER)//assess data package frame header 0x59
      {
        uart[0]=HEADER;
        if(Serial1.read()==HEADER)//assess data package frame header 0x59
        {
          uart[1]=HEADER;
          for(i=2;i<9;i++)//save data in array
          {
            uart[i]=Serial1.read();
          }
          check=uart[0]+uart[1]+uart[2]+uart[3]+uart[4]+uart[5]+uart[6]+uart[7];
          if(uart[8]==(check&0xff))//verify the received data as per protocol
          {
            dist=uart[2]+uart[3]*256;//calculate distance value
            strength=uart[4]+uart[5]*256;//calculate signal strength value
            Serial.print("distance=");
            Serial.print(dist);
            Serial.print('\t');
            Serial.print("strength=");
            Serial.print(strength);
            Serial.print('\n');
          }
        }
      }
    }
```

```
}
void configure(char down[],int n1,int buff[],char up[],int n2,String info)
{
  for(i=0;i<n1;i++)
  {
     Serial1.write(down[i]);
  }
  bool getdata=false;
  int num=0;
  while(!getdata)
  {
    if (Serial1.available())
    {
      if(Serial1.read()==0x5a) //assess communication protocol frame header 0x5a
      {
        buff[0]=0x5a;
        for(i=1;i<n2;i++)
        {
          buff[i]=Serial1.read();
        }
        for(i=0;i<n2;i++)
        {
          if(buff[i]==up[i])
          {
            Serial.print(buff[i],HEX);
            Serial.print('\t');
            num++;
          }
        }
        if(num==n2)
        {
          num=0;
          Serial.print(info);
        }
        getdata=true;
      }
    }
  }
  getdata=false;
  Serial.print('\n');
}
void configure(char down[],int n1,int buff[],int n2,String info)
{
  for(i=0;i<n1;i++)
  {
     Serial1.write(down[i]);
  }
  bool getdata=false;
  while(!getdata)
  {
    if (Serial1.available())
    {
      if(Serial1.read()==0x5a) //assess communication protocol frame header 0x5a
      {
      {
        buff[0]=0x5a;
        for(i=1;i<n2;i++)
        {
          buff[i]=Serial1.read();
        }
        for(i=0;i<n2;i++)
        {
          Serial.print(buff[i],HEX);
          Serial.print('\t');
        }
        Serial.print(info);
        getdata=true;
      }
    }
  }
  getdata=false;
  Serial.print('\n');
}
void configure(char down[],int n1,String info)
{
  for(i=0;i<n1;i++)
  {
      Serial1.write(down[i]);
  }
  Serial.print(info);
  Serial.print('\n');
}
```

STEP3: Data review

Upload the sketch to the Arduino board, then open the serial monitor, you will see the data consistent with the instruction you configured from the lidar, configuration status, and to display the measured distance and reflected strength, as shown in figure 3.


TFmini_PLUS_Setbaud_EN | Arduino 1.8.9
File Edit Sketch Tools Help
TFmini_PLUS_Setbaud_EN
COM7
Send
Autoscroll
Show timestamp
Newline
115200 baud
Clear output
173
Arduino/Genuino Uno on COM7

```
int check;//save check value
int uart[9];//save data measured by LiDAR
const int HEADER=0x59;//frame header of data package
int i=0;

//Takes setting the LiDAR samplerate rate at 10Hz as an example, and other settings
void setup()
{
  Serial.begin(115200);//set bit rate of serial port connecting Arduino with compute
  Serial1.begin(115200); //set bit rate of serial port connecting LiDAR with Arduino
  configure(samplerate_10,6,return_samplerate,samplerate_10,6,info_samplerate_10);
  configure(save,4,return_save,save_success,5,info_save);
}

// According to the data protocol in TFmini Plus manual, the distance display can be
void loop()
{
  if (Serial1.available())//check if serial port has data input
  {
    if(Serial1.read()==HEADER)//assess data package frame header 0x59
    {
      uart[0]=HEADER;
      if(Serial1.read()==HEADER)//assess data package frame header 0x59
```

```
5A      6       3       A       0       6D      samplerate_10 ok
5A      5       11      0       70      save ok
distance=181    strength=524
distance=181    strength=527
distance=182    strength=527
distance=182    strength=525
distance=181    strength=526
distance=181    strength=525
distance=181    strength=525
distance=181    strength=526
distance=182    strength=524
distance=182    strength=527
distance=181    strength=525
distance=182    strength=525
distance=180    strength=525
distance=181    strength=526
```

```
Sketch uses 6070 bytes (18%) of program storage space. Maximum is 32256 bytes.
Global variables use 575 bytes (28%) of dynamic memory, leaving 1473 bytes for local variables. Maximum is 2048 bytes.
```

Figure 3 display the lidar data on serial monitor