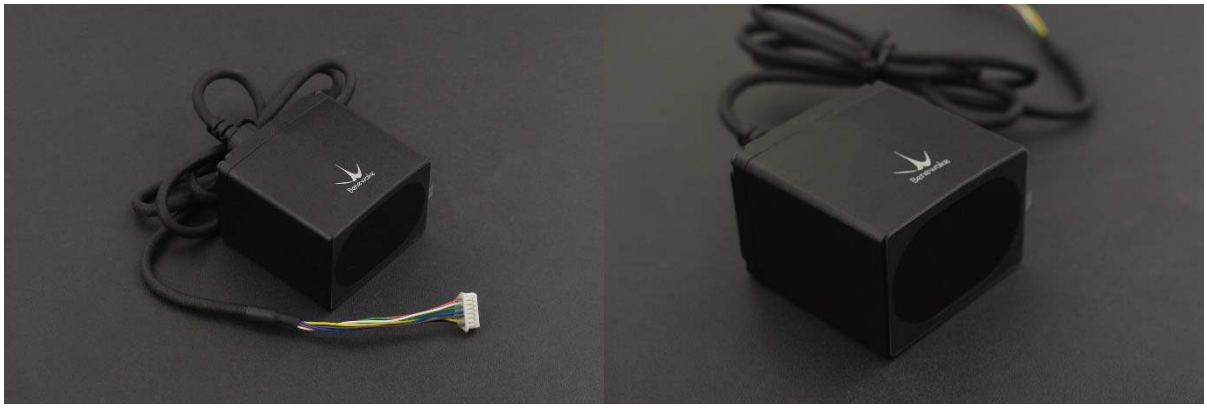


Introduction

As the third generation product of TF series, TF03 inherits the cost-effective and compact-integration advantages from the previous two generations. Meanwhile, TF03 upgrades more than ten key parameters and offers multiple expansion functions to meet the various demands in different application scenarios. We have two detection ranges for you to choose: 100m and 180m. With a small size of 44×42.9×31.8mm, it can be covered by just a palm of one hand. The product is applicable to the terrain following of drones, the collision avoidance of cars, intelligent transportation and industrial safety warning.

TF03 employs the pulsed time-of-flight principle. The unique design of the optical system and the signal processing circuit improved the detecting performance in a compact size. In addition to increasing the range to more than 100 meters, this sensor features 0.1m blind zone, ±10cm accuracy, up to 10KHz frequency and 100Klux ambient light immunity. TF03 also contains a compensation algorithm targeting outdoor highlight environment, so that it can still keep excellent performance in harsh environment.

TF03 sensor uses aluminum-made shell and infrared band-pass glass to improve the overall strength. With IP67 enclosure rate, the sensor can be applied in various extreme environments. It supports multiple interfaces for different applications, such as, UART, CAN, IO. Besides that, multiple parameters of TF03 can be configured by customers, including the measuring frequency, baud rate, trigger mode, over-range assignment, and so on. TF03 is also equipped with BootLoader function, enabling users to upgrade product firmware locally. Power the sensor with 5V. The average power consumption is 0.55W. It is compatible with controllers like Arduino, Raspberry Pi. With Arduino and

Raspberry Pi libraries developed by DFRobot, users can conveniently integrate functions into system to develop their applications.

⚠ **NOTE**:

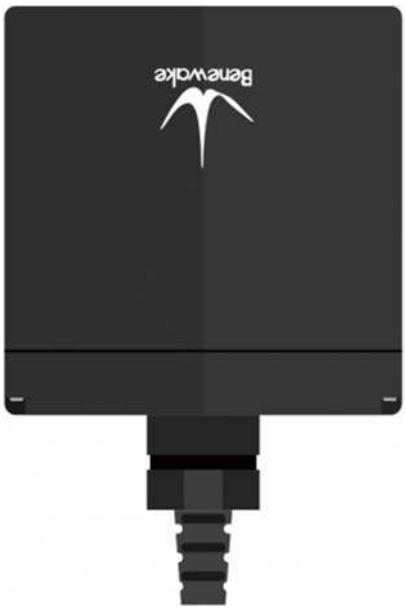
- This product can only be maintained by qualified professionals and only the original spare parts can be used to ensure its performance and safety.
- The product itself has no polarity and over-voltage protection. Please complete wiring and supply power correctly according to the contents of the Manual.
- The working temperature of the product is -20℃~60℃, do not use it beyond this range so as to avoid risks.
- The storage temperature of the product is -40°C~85°C; please do not store it beyond this temperature range, so as to avoid risks.
- Do not open its enclosure for assembly or maintenance beyond this Manual; otherwise, it will affect the product performance.
- When the product transmitter and receiver lens are covered by dirt, there will be a risk of failures. Please keep the lens clean.
- The product will have a risk of failure when immersed completely in water. Do not use it underwater.
- When detecting objects with high reflectivity, such as mirrors and smooth tiles, the product may have a risk of failures.

Specification

- Product Performance
  - SEN0328 Detection Range: 0.1m-100m@90% reflectivity, 0.1m-40m@10% reflectivity, 0.1m-80m@90% reflectivity &100Klux, 0.1m-30m@10%reflectivity &100Klux
  - SEN0329 Detection Range: 0.1m-180m@90% reflectivity, 0.1m-70m@10% reflectivity, 0.1m-130m@90% reflectivity &100Klux, 0.1m-50m@10% reflectivity &100Klux
  - Accuracy: ±10cm (less than 10m), 1%(more than 10m)
  - Distance Resolution: 1cm
  - Frame Rate: 1Hz-1000Hz Adjustable (Default 100Hz)
  - Ambient Light Immunity: 100Klux

    - Repeatability: 1σ：<3cm
    - Operating Temperature: -25℃~60℃
    - Enclosure Rate: IP67
- Optical Parameters
    - Light Source: LD
    - Wavelength of Light Source: 905nm
    - FOV Angle: 0.5°
    - Laser Class: CLASS1（EN60825）
- Electrical Parameters
    - Supply Voltage: 5V±0.5V
    - Average Current: ≤180mA
    - Power Consumption: ≤0.9W
    - Peak Current: ≤180mA
    - Communication Voltage Level: LVTTL（3.3V）
    - Communication Interfaces: λUART/CAN/IO
- Others
    - Dimension: 44*43*32mm/1.73*1.69*1.26″
    - Enclosure Material: aluminum alloy
    - Storage Temperature: -40℃~85℃
    - Weight: 77g±3g
    - Wiring Length: 70cm/27.56″

Board Overview



Num	Label	Description
Red	VCC	Power Supply
White	CAN_L	CAN Bus
Green	CAN_H	CAN Bus
Blue	GPIO	IO Output
Brown	TTL_RXD	Serial Receive
Yellow	TTL_TXD	Serial Transmit
Black	GND	Ground

NOTE

- The interface type of the product is MH1.25-7P, which cannot be directly used on Arduino Uno so we provide you with the 2.54-1P dupont wires and PH-P plastic shell socket. The wires on the sensor and the dupont wires should be connected in this way: black(sensor) to black(dupont), red to red, brown to blue, yellow to green.
- The sensor supports two communication modes: TTL and CAN. The default mode is TTL, and can be changed via command by users. Please note that the two modes cannot be used as output at the same time.

Module Communication Protocol and Data Format

The standard version of TF03 supports two communication modes: TTL and CAN. Users can use command to change the default TTF mode. The two modes cannot be used as output at the same time.

Serial Port Mode

TF03 serial port mode adopts UART-LVTTL interface. The output level is LVTTL level (0~3.3V), and the details are shown below:

Item	Content
Communication Protocols	UART
Band Rate	115200
Data Byte	8
Stop Byte	1
Check Byte	None

Standard Serial Data Format(UART)

The outputs of TF03 are shown as below: all the data are hexadecimal number; there are 9 bytes in data of each frame. The data includes real measured distance(DIST); the other bytes are reserved; the frame tail is checkbyte.

Data Byte	Define	Description
Byte0	Frame Head	0x59
Byte1	Frame Head	0x59
Byte2	DIST_L	DIST low eight bits

Data Byte	Define	Description
Byte3	DIST_H	DIST high eight bits
Byte4	Reserved	/
Byte5	Reserved	/
Byte6	Reserved	/
Byte7	Reserved	/
Byte8	Check	Low eights bytes of Checksum, Checksum=Byte0 + Byte1 + ...+ Byte7

Serial Port Pixhawk Data Format

Pixhawl data format is output in the form of string, unit m. For example, if the measured distance is 1.21m, then output the string 1.21. There is a linefeed behind each distance value. In serial communication, the output mode can be changed to pixhawk output mode via command.

High/Low Level Output

Set a threshold (adjustable), when the measured distance is more /less than the threshold, output high/low.

- a) The high/low level can be adjusted. By defaut, high level for short distance, low level for long distance.
- b) Adjustable buffer area(prevent level jumping caused by datashake)
- c) Ajustable delay function of High and low level
    - Delay before triggering. Set a threshold 10m, and if the distance is less than 10m, output high. When the distance is less than 10m, add an adjustable delay(defalut as 0ms). Now if the distance is still less than 10m after the delay, output High.

    - Delay after triggering. Set a threshold 10m, and if the distance is less than 10m, output high. When the output distance is more than 10m, add an adjustable delay(default as 0ms). Now if the distance is still more than 10m after the delay, output low.

For example：

- a) Set the output mode of TF03 as IO high/low level output, command: ---5a 05 05 05 69
- b) Set: When the distance is less than threshold, output high, otherwise, low. Command:---5A 05 61 01 C1
- c) Add a delay of 100ms for the level change when the distance increases or decreases. Commad: ---5A 08 62 64 00 64 00 8C
- d) Set the distance threshold to 500cm, buffer area to 5cm. Command: ---5A 08 63 F4 01 05 00 BF
- e) Save the settings, command: ---5A 04 11 6F
- f) Restore the factory configuration(if necessary), command: ---5A 04 10 6E

CAN Bus Mode

The CAN communication protocol of TF03 can be customized. CAN band rate, ID, and Frame format are adjustable. As seen below:

Item	**Content**
Communication Protocols	CAN
Band Rate	1M
Receive ID	0x3003
Transmit ID	0x3
Check Byte	Transmit Frame is regarded as standard frame by default, and receive frame supports standard frame and expension frame

The data format TF03 in CAN mode is shown below. All the data are hexadecimal number and there are 8 bytes in each data frame. The data includes real measured distance(DIST); the other bytes are reserved.

Data Byte	Define	Description
Byte0	DIST_L	DIST low eight bits
Byte1	DIST_H	DIST high eight bits
Byte2	Reserved	/
Byte3	Reserved	/
Byte4	Reserved	/
Byte5	Reserved	/

Customized Configuration

General Command Description

The product parameters can be changed to meet requirements of various application. Users can send the related command to revise the orignal parameters, such as output data format, frame rate and so on. After the setting completed, input wirte-in configuration command then the parameters will be saved in Flash and users don't need to reset after restarting the device.

Please change the parameters according to the instruction, and do not try sending irrelevant or unstated command in case of casuing unnecessary lost.

Byte	byte0	byte1	byte2	byte3~byteN-2	byteN-1
Description	Head	Len	ID	Payload	Check sum

- Head: fixed 0x5A
- Len: the length of the entire command frame(unit: Byte)
- ID: marking the function of each command
- Payload: parameter, it has different function and legnth in various command
- Check sum: last Len-1 byte data and low 8bit

Function	Downstream	Upstream	Description	Factory Configuration
Get firmware Version	5A 04 01 5F	5A 07 01 V1 V2 V3 SU	Version: V3.V2.V1; SU: checksum	/
System Reset	5A 04 02 60	Succeed: 5A 05 02 00 61; Fail: timeout 1s, no response	/	/
Set Output frame rate	5A 06 03 LL HH SU	Succeed: same as downstream; Fail: timeout 1s, no response	/	100fps
Enable Output	Enable: 5A 05 07 01 67;	Succeed: same as downstream; Fail: timeout	/	Enable

Function	Downstream	Upstream	Description	Factory Configuration
	Disable: 5A 05 07 00 66	1s, no response		
Single Trigger	5A 04 04 62	Data Frame	/	/
Set Output Format	5A 05 05 LL SU	Succeed: same as downstream; Fail: timeout 1s, no response	LL: output format, for instance: 00 : ASCII output(reserved);01: binary output; 02: PIX output; 05: IO output	Binary
Set serial band rate	5A 08 06 H1 H2 H3 H4 SU	Succeed: same as downstream; Fail: timeout 1s, no response	band rate=(H4 << 24)+(H3 << 16)+(H2 << 8)+H1	115200
Check and Enable	Enable:5A 05 08 01 68; Disable: 5A 05 08 00 67	Succeed: same as downstream; Fail: timeout 1s, no response	/	Enable
Restore Factory Configuration	5A 04 10 6E	Succeed: 5A 05 10 00 6F; Fail: 5A 05 10 ER SU	Fails if ER is not equal to 0	/

Function	Downstream	Upstream	Description	Factory Configuration
Save Configuration	5A 04 11 6F	Succeed:5A 05 11 00 70; Fail:5A 05 11 ER SU	Fails if ER is not equal to 0	/
Configure Over Range Value	5A 06 4F LL HH SU	Succeed: 5A 05 4F 00 AE; Fail: timeout 1s, no response	Over Range Value=(HH << 8) + LL, unit: cm	18000
Configure CAN transmit ID	5A 08 50 H1 H2 H3 H4 SU	Succeed:5A 05 50 00 AF; Fail: timeout 1s, no response	ID=(H4 << 24)+(H3 << 16)+(H2 << 8)+H1	0x3
Configure CAN receive ID	5A 08 51 H1 H2 H3 H4 SU	Succeed: 5A 05 51 00 B0; Fail: timeout 1s, no response	ID=(H4 << 24)+(H3 << 16)+(H2 << 8)+H1	0x3003
Configure CAN band rate	5A 08 52 H1 H2 H3 H4 SU	Succeed:5A 05 52 00 B1; Fail: timeout 1s, no response	Baudrate=(H4 << 24)+(H3 << 16)+(H2 << 8)+H1	1000000
Configure CAN transmit frame type	Standard frame:5A 05 5D 00 BC; Expanded	Succeed:5A 05 5D 00 BC; Fail: timeout	/	Standard Frame

Function	Downstream	Upstream	Description	Factory Configuration
	Frame:5A 05 5D 01 BD	1s, no response		
Configure transmit mode	TTL: 5A 05 45 01 A5; CAN: 5A 05 45 02 A6	Succeed: 5A 05 45 00 A4; Fail: timeout 1s, no response	/	TTL
Environment Compensation Algorithm	Enable:5A 05 64 00 C3; Disable: 5A 05 64 01 C4	Succeed: 5A 05 64 00 C3; Fail: timeout 1s, no response	/	Enable
Configure offset	5A 06 69 LL HH SU	Succeed:5A 05 69 00 C8; Fail: timeout 1s, no response	Offset = (HH << 8) + LL, unit: cm	0
Set Trigger level for short distance	5A 05 61 LV SU	Succeed: 5A 05 61 00 C0; Fail: timeout 1s, no response	LV=0, low level trigger; LV=1, high level trigger	Low
Set IO output delay	5A 08 62 L1 H1 L2 H2 SU	Succeed: 5A 05 62 00 C1; Fail: timeout 1s, no response	Delay1 = (H1 << 8) + L1，Delay2 = (H2 << 8) + L2 (unit:ms), respectively represent the	0, 0

Function	Downstream	Upstream	Description	Factory Configuration
			delays of IO level change in short or long distance. Range: 0~65000	
Set distance threshold and buffer distance	5A 08 63 L1 H1 L2 H2 SU	Succeed: 5A 05 63 00 C2; Fail: timeout 1s, no response	Dist= (H1 << 8) + L1, Buff=(H2 << 8) + L2 (unit: cm), respectively represent the distance threshold and buffer distance of IO change. Range: 0~18000.	18000, 0

Description:

- The supported output frames are shown below:
  - 1, 2,...9,
  - 10, 20, ...90,
  - 100, 200, ...900,
  - 1000, 2000, ...9000, 10000;
- In trigger mode, disable the output function first then to use trigger command;
- PIX format "x.yz\r\n", unit: m. For example：output "1.23" when then measured distance is 123cm, followed by "Carriage Return + Line Feed";
- Support CAN band rate: 1M, 500K, 250K and 125K;
- v1.11.3 and above support the second laser distance calibration by users via command "configure offset";
- The CAN ID to be configured muse be legal. Otherwise, unexpected results may occur.
- To use IO trigger function, users have to set the output format to IO output, and configure IO trigger level, output delay, and distance threshold and buffer distance via commands.

Tutorial

To give users a direct impression on the Laser Range Sensor, this tutorial provides the following information:

1. How to use this TF03 Laser Range Sensor on Arduino?
2. How to read the output distance of the sensor on PC?
3. Black and white line detection.

Requirements

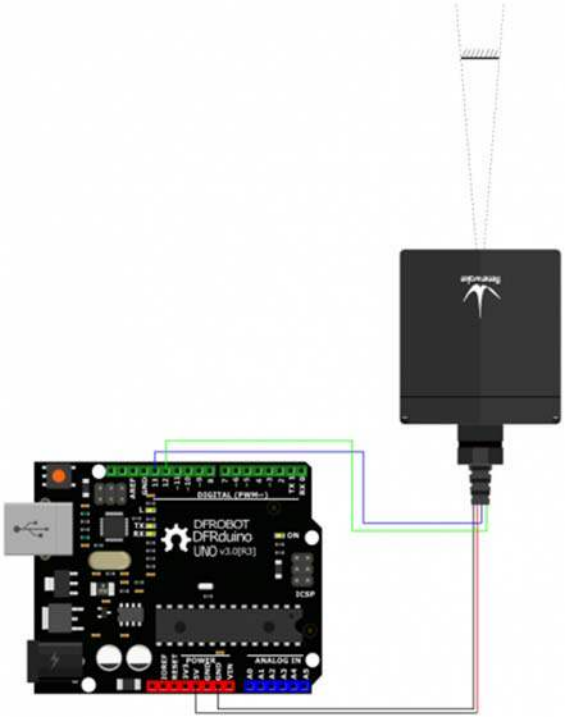
- **Hardware**
    - DFRduino UNO R3 (or similar) x 1
    - IO Sensor Expansion Board V7.1 x 1
    - Gravity: I2C 16x2 Arduino LCD with RGB Backlight Display x 1
    - 7.4V 2500MA Li-ion Battery (With rechargeable protection board) x 1
    - USB to Serial Cable x 1
    - Jumper wires
- **Software**
    - Arduino IDE
    - Click to download DFRobot TF Mini Library
    - Click to download DFRobot LCD library file

Debugging on Arduino (PC serial port)

Since the TF mini is a serial device and the ordinary Arduino has only one hardware serial, we recommand using the sensor together with a software serial. Of course, users can also use device with multi-serial port, such as Arduino Leonardo, Arduino Mega2560 and so on. Here we use the common Arduino Uno as the controller, and define D12 and D13 as software serial port.

Arduino Connection

- Use the serial software to display the measured distance and power the entire system.



Sample Code(Arduino Debugging)

- A PC serial port tool is needed here and the readings will be displayed on the tool's interface.

```
  * @File  : DFRobot_TFmini_test.ino
  * @Brief : This example use TFmini to measure distance
  *         With initialization completed, we can get distance value and signal
strength
  * @Copyright   [DFRobot](http://www.dfrobot.com), 2016
  *             GNU Lesser General Public License
  *
  * @version  V1.0
  * @date  2018-1-10
*/

#include <DFRobot_TFmini.h>

SoftwareSerial mySerial(12, 13); // RX, TX

DFRobot_TFmini  TFmini;
uint16_t distance,strength;

void setup(){
    Serial.begin(115200);
    TFmini.begin(mySerial);
}
```

```
void loop(){
    if(TFmini.measure()){                      //Measure Distance and get signal
strength
        distance = TFmini.getDistance();       //Get distance data
        strength = TFmini.getStrength();       //Get signal strength data
        Serial.print("Distance = ");
        Serial.print(distance);
        Serial.println("cm");
        Serial.print("Strength = ");
        Serial.println(strength);
        delay(500);
    }
    delay(500);
}
```

Copy

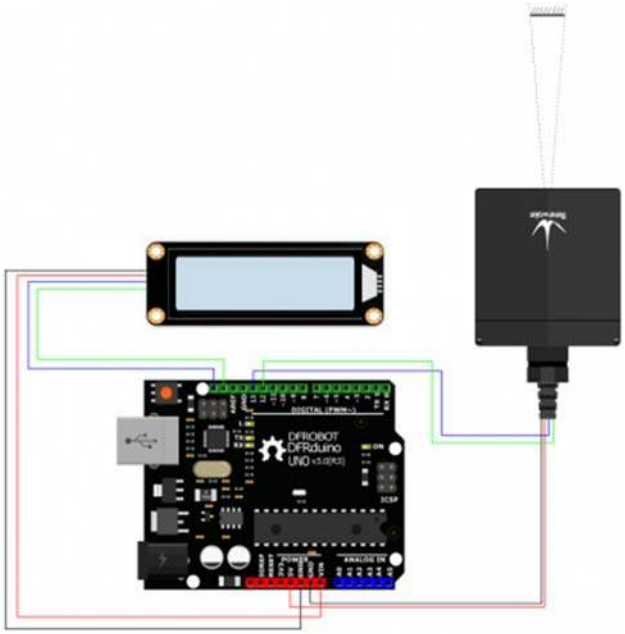
- The following is the data format displayed on serial software.

```
Distance = 1000 mm
```

```
Strength =  688
```

Distance Display on LCD

In actual applications, the sensor may need to be used without a PC. So in this tutorial, the li-ion battery will be power supply and the LCD is used for displaying detected distance.



Arduino Test Code

```
/*
  * @File  : DFRobot_TFmini_test.ino
  * @Brief : This example use TFmini to measure distance
  *         With initialization completed, we can get distance value and signal
strength
  * @Copyright   [DFRobot](http://www.dfrobot.com), 2016
  *             GNU Lesser General Public License
  *
  * @version  V1.0
  * @date  2018-1-10
*/
#include <Wire.h>
#include <DFRobot_RGBLCD.h>
#include <DFRobot_TFmini.h>            //TF Mini header file

SoftwareSerial mySerial(12, 13);      // RX, TX
DFRobot_TFmini  TFmini;
uint16_t distance,strength;

unsigned int lcd_r = 0, lcd_g = 0, lcd_b = 0;
unsigned long delaytime = 0, lighttime = 0;
DFRobot_RGBLCD lcd(16, 2);
void setup()
{lcd.init();
  delay(5000);
  Serial.begin(115200);
  Serial.println("hello start");

  TFmini.begin(mySerial);
  lighttime = millis();
  lcd.setCursor(0, 0);
  lcd.print("Dis:");
  lcd.setCursor(0, 1);
  lcd.print("Str:");
  lcd.setRGB(255, 255, 000);
}
void loop() {

/*****************LCD******************/
 lcd_r = random(256);
  delayMicroseconds(10);
  lcd_g = random(256);
  delayMicroseconds(10);
  lcd_b = random(256);
  if (millis() - lighttime > 3000)
  {
    lcd.setRGB(lcd_r, lcd_g, lcd_b);
    lighttime = millis();
  }
  //delay(100);
  /**************TF Mini***************/
```

```
 if(TFmini.measure()){                             //Measure Distance and get signal
strength
        distance = TFmini.getDistance();       //Get distance data
        strength = TFmini.getStrength();       //Get signal strength data

    lcd.setCursor(5, 0);                       //LCD display
    lcd.print(  distance / 10000);
    lcd.print(  distance/ 1000 % 10);
    lcd.print('.');
    lcd.print(  distance / 100 % 10);
    lcd.print(  distance / 10 % 10);
    lcd.print(  distance  % 10);
    lcd.print(" m");
    lcd.setCursor(5, 1);
    lcd.print(strength / 10000);
    lcd.print(strength / 1000 % 10);
    lcd.print(strength / 100 % 10);
    lcd.print(strength / 10 % 10);
    lcd.print(strength % 10);
    }
}
```

- The following data format will be displayed ont he LCD screen:

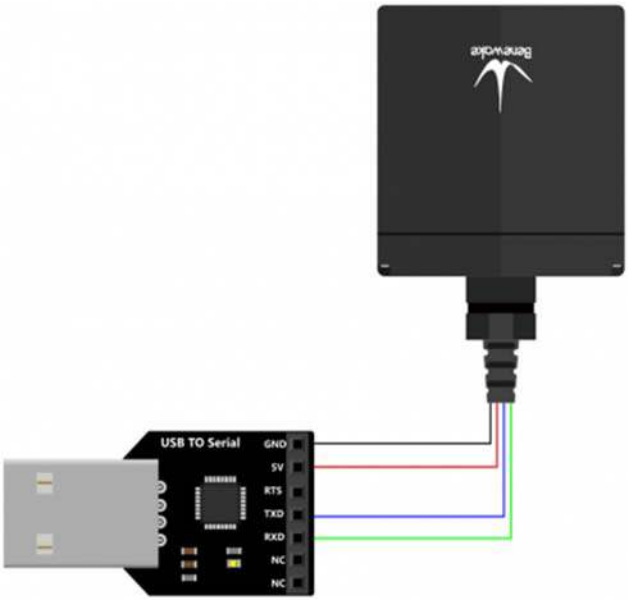
```
- Dis: 05.000 m
```

```
Str: 00600
```

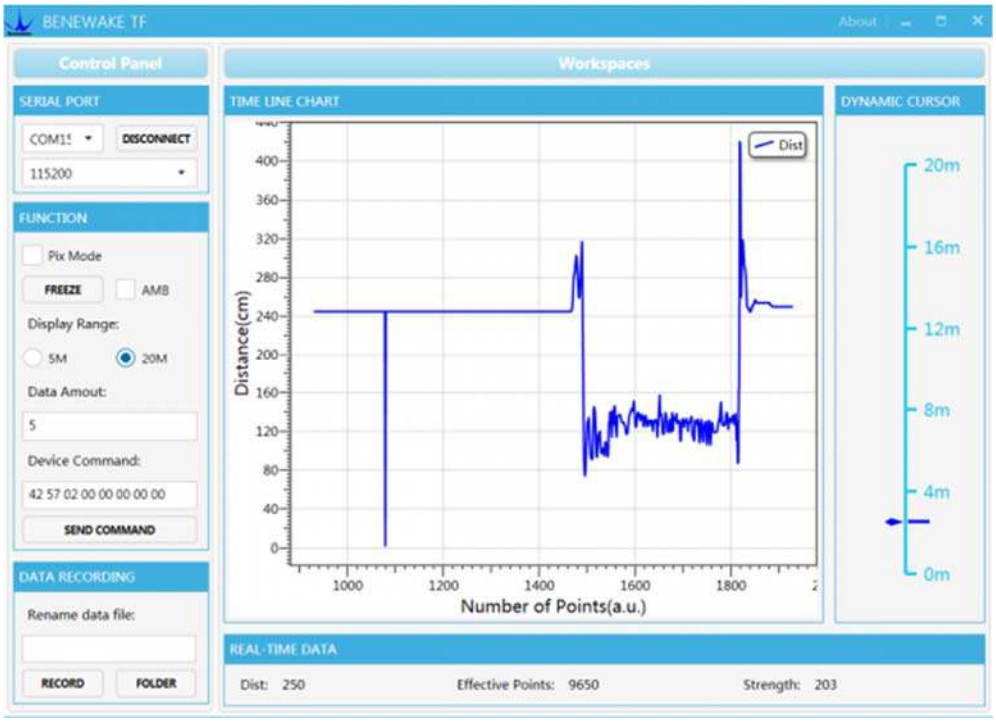
Distance Display on Upper PC

In addition to reading data through a single chip, we can also use a PC software to read the detected distance.

Connection: connect the TF mini to a computer via a USB-to-TTL module, and read data through the upper PC.



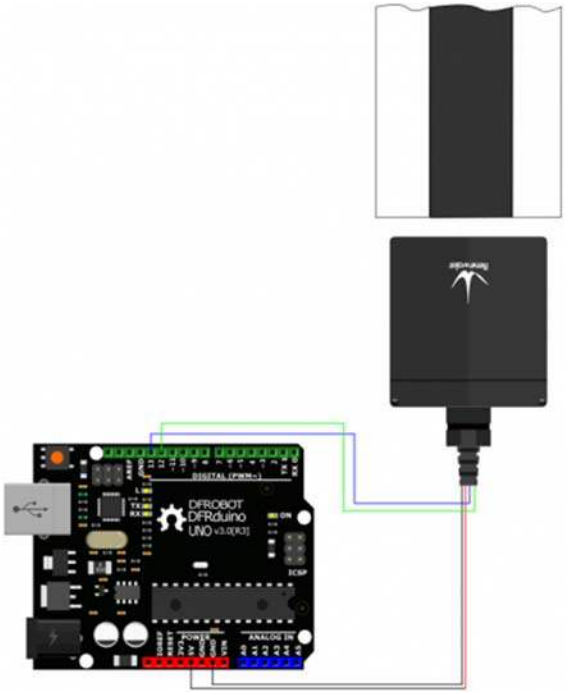
Result:



Detection of Black and White Line

- Since TF Mini Laser Ranging Sensor is an optical sensor and features high sensitivity to light, we can use it to achieve close-range black and white line detection (White has the highest reflectivity while black has the lowest reflectivity).
- The signal "strength" will be used to distinguish the two colors.

Connection



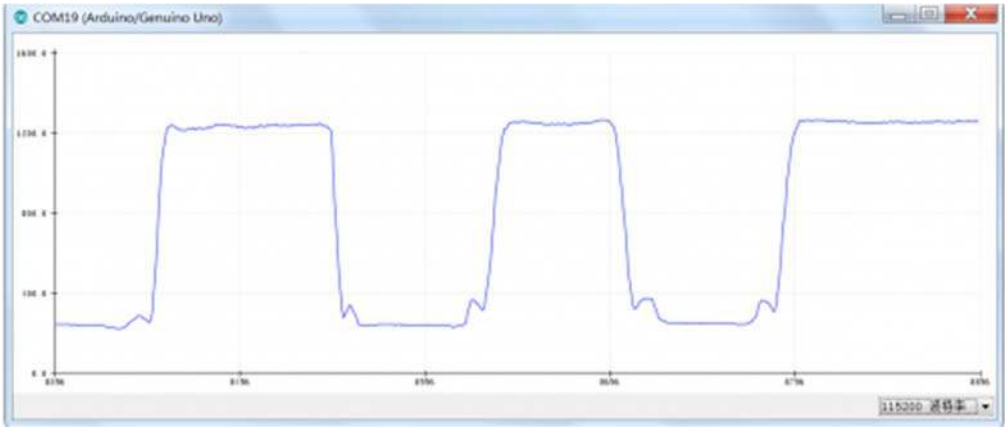
Arduino Code

- Note: the code needs to be used with TF Mini Library.

```
/*
  * @File  : DFRobot_TFmini_test.ino
  * @Brief : This example use TFmini to measure distance
  *         With initialization completed, we can get distance value and signal
strength
  * @Copyright   [DFRobot](http://www.dfrobot.com), 2016
  *             GNU Lesser General Public License
  *
  * @version  V1.0
  * @date  2018-1-10
*/

#include <DFRobot_TFmini.h>

SoftwareSerial mySerial(12, 13); // RX, TX

DFRobot_TFmini  TFmini;
uint16_t distance,strength;

void setup(){
    Serial.begin(115200);
    TFmini.begin(mySerial);
}
```

```
void loop(){
    if(TFmini.measure()){                       //Measure Distance and get signal
strength
        strength = TFmini.getStrength();        //Get signal strength data
        Serial.print("Strength = ");
        Serial.println(strength);
    }
}
```

- Download the program, use the sensor to detect the white and black paper at a same distance, then the signal strength will be showed on the serial drawing tool interface.

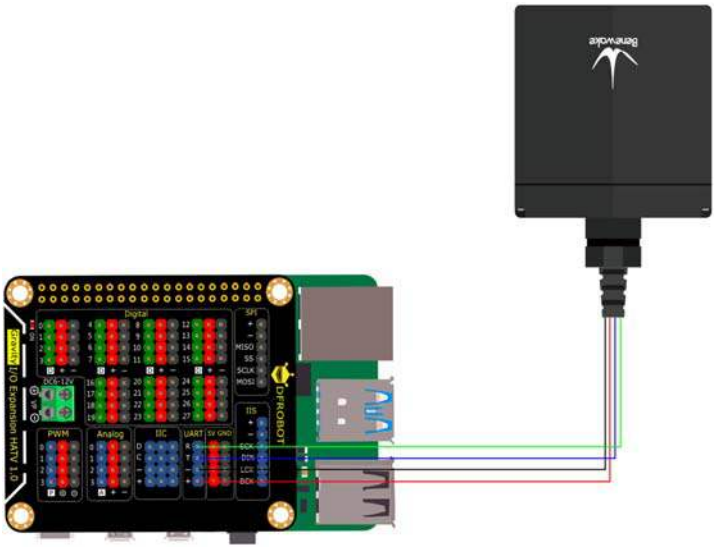


Tutorial on Raspberry Pi

Requirements

- Raspberry Pi 4B
- Raspberry IO Expansion Board
- TF03(TOF) Laser Range Sensor(180m)
- Connector

Connection with Raspberry Pi



Sample Code

```
# -*- coding:utf-8 -*-

'''
  # DFRobot_TFmini.py
  #
  # Connect board with raspberryPi.
  # Run this demo.
  #
  # Connect TFmini to UART
  # get the distance value
  #
  # Copyright   [DFRobot](http://www.dfrobot.com), 2016
  # Copyright   GNU Lesser General Public License
  #
  # version  V1.0
  # date  2019-8-31
'''

import time

from DFRobot_TFmini import TFMINI

mini = TFMINI()

def main():
    while True:
        if mini.measure():
            distance = mini.getDistance()
            strength = mini.getStrength()
            print("Distance = %.d" % distance)
            print("Strength = %.d" % strength)
            time.sleep(0.5)
```

```
        time.sleep(0.5)

if __name__ == "__main__":
    main()
```

FAQ

For any questions, advice or cool ideas to share, please visit the **DFRobot Forum**