

# How to install Benewake TF SERIES on PixHawk

## ( Take TF01 for example )

Install TF SERIES module on your drone vertically to the ground so you can get the absolute altitude of your drone. Here is how to install our module on PixHawk. We provided two methods below.

### 1. How does TF SERIES work on PixHawk

[Lidar's are used in flight modes which have height control, such as Altitude Hold, Loiter and PosHold Mode. The data from the sensor will be used until you exceed RNGFND\\_MAX\\_CM, after that it switches to the barometer. Currently Lidar is not supported in Auto Mode.](#)

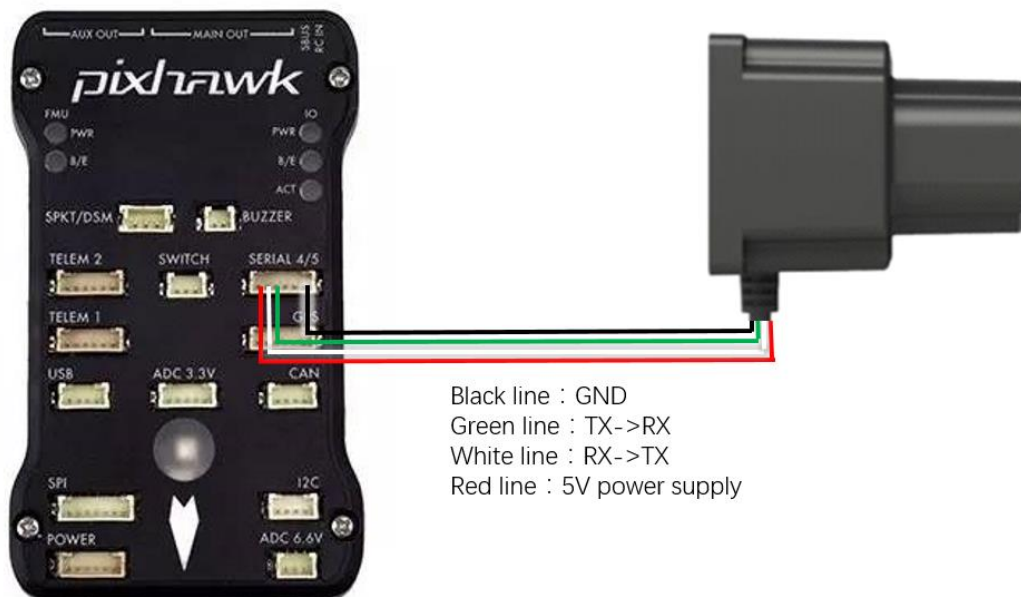
PixHawk interface for rangefinder are I2C , Analog, Serial. Details are on <http://ardupilot.org/copter/docs/common-rangefinder-landingpage.html>

### 2. Serial Mode

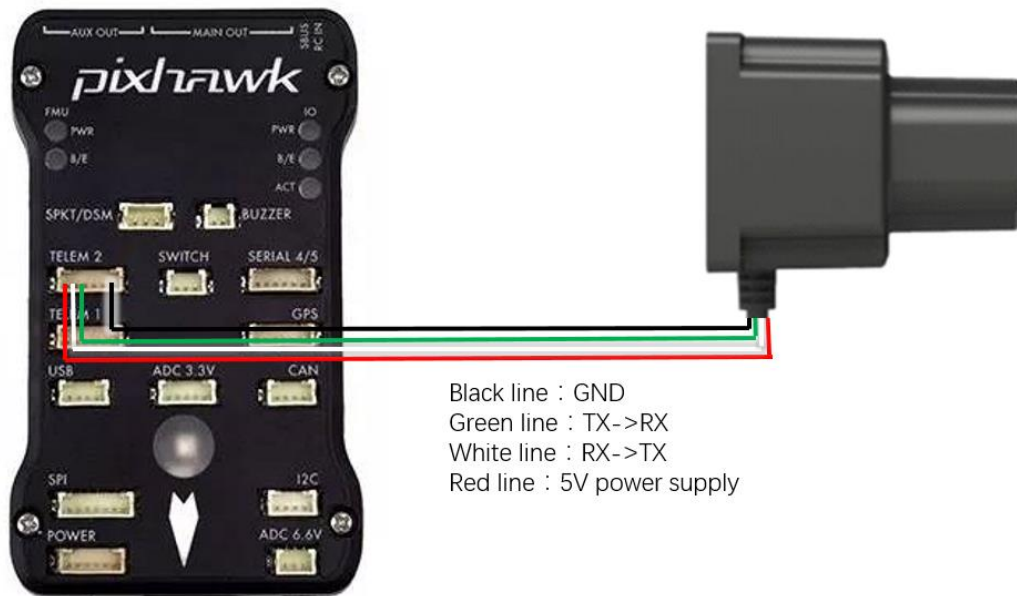
We recommend using **Serial** to send altitude signal to PixHawk.

Cautions: If you use serial, please update your firmware, make sure is newer than V 3.3.3.

#### 2.1 Wire



Pic. 1(a) Connect TF SERIES to PixHawk via Serial



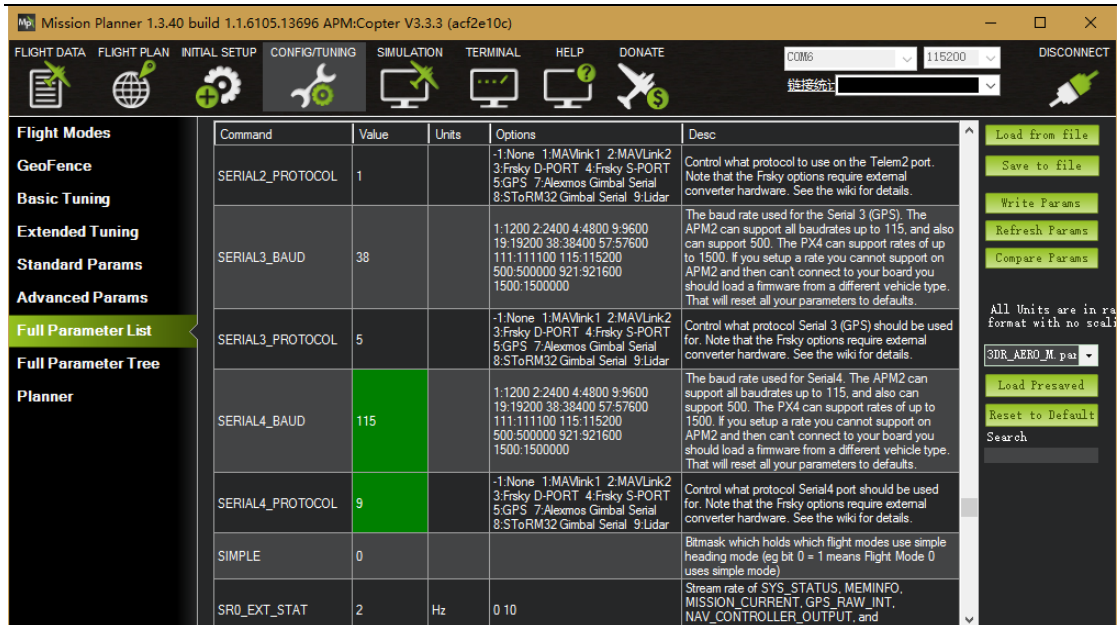
Pic. 1(b) Connect TF SERIES to PixHawk via Serial

## 2.2 Mission Planner

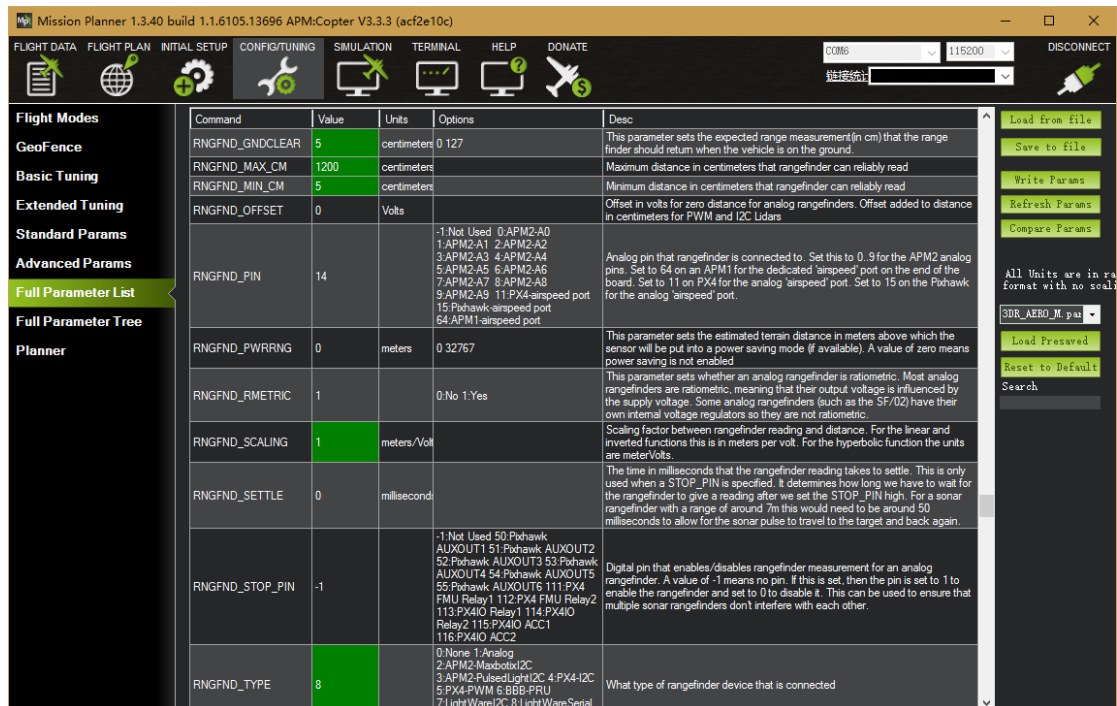
Connect Flight Controller to MP, click **CONFIG/TUNING** then select **Full Parameter List** change parameters as below:

- SERIAL4\_PROTOCOL = 9 (Lidar)
- SERIAL4\_BAUD = 115
- RNGFND\_TYPE = 8 (LightWareSerial)
- RNGFND\_SCALING = 1
- RNGFND\_MIN\_CM = 5
- RNGFND\_MAX\_CM = 1200
- RNGFND\_GNDCLEAR = 5 unit cm, or you can use more specific value, it depends on the height TF SERIES installed.

After all your settings click **Write Params**  
See in Pic.2 and Pic.3



Pic.2 Serial parameter configuration



Pic.3 Serial parameter configuration

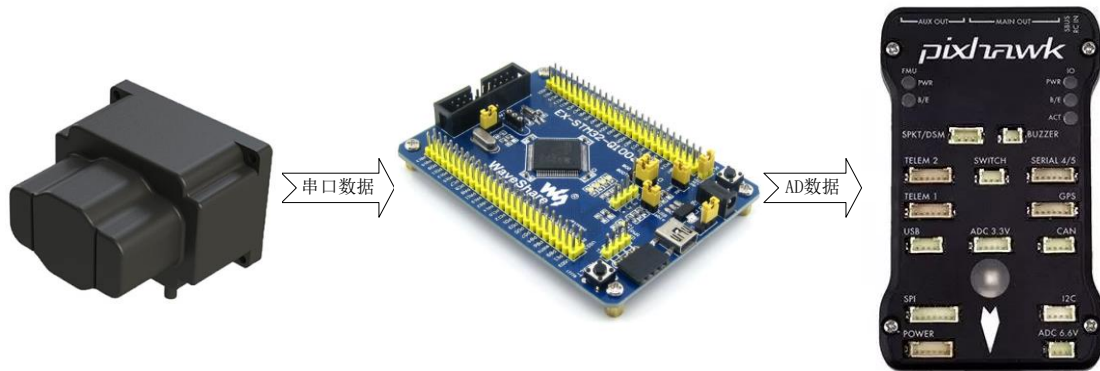
If **Bad Lidar Health** error occurs, please check if TF SERIES Lidar window first, see whether TF SERIES emit red LED light. If there's no red LED light, please check power supply. (Sometimes Pixhawk have Serial 4/5 power supply issue) If there is LED light, please check if you wire the Serial correctly. If you still get this error message and got no reading in sonarrange/sonarvoltage, please connect TF01 to TELEM2 (like Pic.1(b))and change the Parameter for Serial 2 as below.

- SERIAL2\_PROTOCOL = 9 (Lidar)
- SERIAL2\_BAUD = 115

### 3. AD mode to simulate Sonar Sensor Maxbotix

TF SERIES data is first send to a STM32 board and data is translated into AD data then sent to PixHawk.

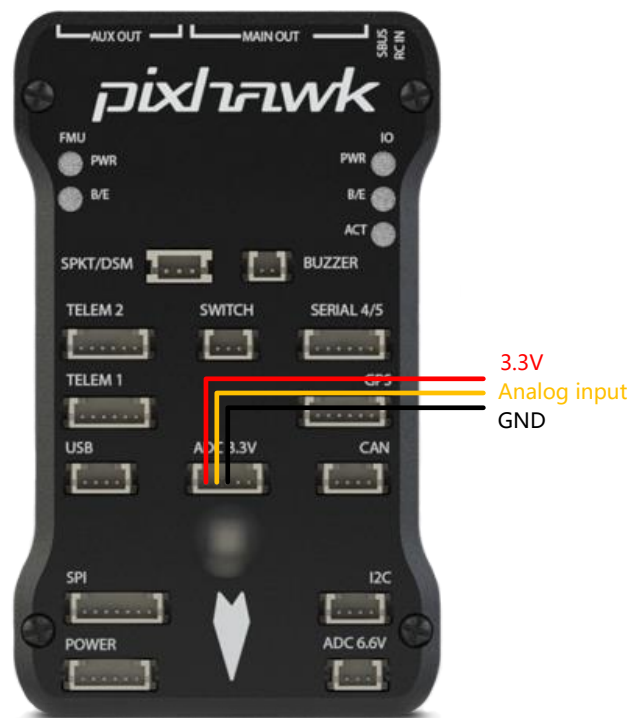
Caution: TF SERIES, STM32 board and PixHawk shall share a common-ground.



Pic. 4 connect TF SERIES to PixHawk

#### 3.1 Wire

Connect TF SERIES to input distance data using analog into PixHawk



Pic. 5 Analog input wire,

- Red: 3.3V
- Orange: AD analog signal
- Black: GND

## 3.2 Mission Planner Configuration

Connect Flight controller to MP, select **Full Parameter List** in **Config/Tuning**, find and change the parameters below:

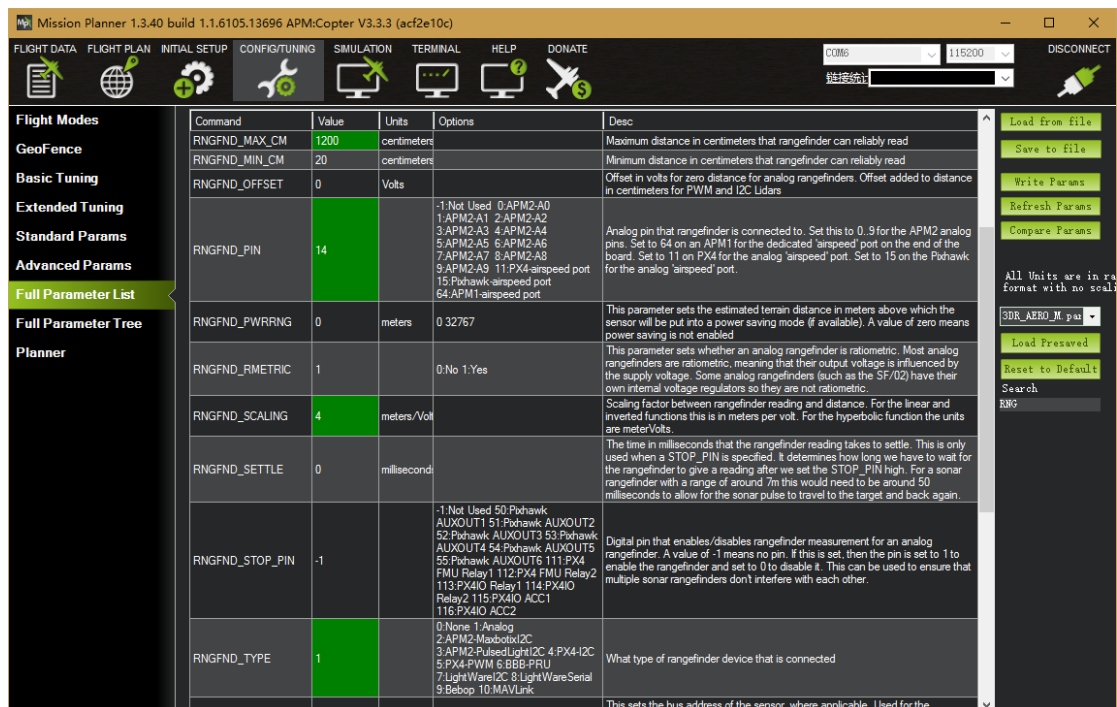
- RNGFND\_PIN = "14" for PixHawk's ADC 3.3v pin #2 **OR** "0" for APM2.x
- RNGFND\_MAX\_CM = "1200" (i.e. 12m max range)
- RNGFND\_SCALING = "4" (i.e. 4m / 1v)
- RNGFND\_TYPE = "1" (Analog)

STM32 DA module can output voltage between 0-3.3V. When the distance reaches 12m, the voltage sent to PixHawk is 3V. Therefore set RNGFND\_SCALING to 4.

When all parameters are set, click **Write Param**.

After all parameters are set and written, the TF SERIES data works on Altitude Hold, Loiter and PosHold Mode. The data from the sensor will be used until you exceed RNGFND\_MAX\_CM, (which as our set is 12m) after that it switches to the barometer. Currently Lidar is not supported in Auto Mode.

Details are in Pic.6:



Command	Value	Units	Options	Desc
RNGFND_MAX_CM	1200	centimeters		Maximum distance in centimeters that rangefinder can reliably read
RNGFND_MIN_CM	20	centimeters		Minimum distance in centimeters that rangefinder can reliably read
RNGFND_OFFSET	0	Volts		Offset in volts for zero distance for analog rangefinders. Offset added to distance in centimeters for PWM and I2C Lidars
RNGFND_PIN	14		-1:Not Used 0:APM2-A0 1:APM2-A1 2:APM2-A2 3:APM2-A3 4:APM2-A4 5:APM2-A5 6:APM2-A6 7:APM2-A7 8:APM2-A8 9:APM2-A9 11:PX4-airspeed port 15:Pixhawk-airspeed port 64:APM1-airspeed port	Analog pin that rangefinder is connected to. Set this to 0..9 for the APM2 analog pins. Set to 64 on an APM1 for the dedicated 'airspeed' port on the end of the board. Set to 11 on PX4 for the analog 'airspeed' port. Set to 15 on the Pixhawk for the analog 'airspeed' port.
RNGFND_PWRRNG	0	meters	0 32767	This parameter sets the estimated terrain distance in meters above which the sensor will be put into a power saving mode (if available). A value of zero means power saving is not enabled
RNGFND_RMTRIC	1		0:No 1:Yes	This parameter sets whether an analog rangefinder is ratiometric. Most analog rangefinders are ratiometric, meaning that their output voltage is influenced by the supply voltage. Some analog rangefinders (such as the SF-02) have their own internal voltage regulators so they are not ratiometric.
RNGFND_SCALING	4	meters/Volt		Scaling factor between rangefinder reading and distance. For the linear and inverted functions this is in meters per volt. For the hyperbolic function the units are meterVolts
RNGFND_SETTLE	0	millisecond		The time in milliseconds that the rangefinder reading takes to settle. This is only used when a STOP_PIN is specified. It determines how long we have to wait for the rangefinder to give a reading after we set the STOP_PIN high. For a sonar rangefinder with a range of around 7m this would need to be around 50 milliseconds to allow for the sonar pulse to travel to the target and back again.
RNGFND_STOP_PIN	-1		-1:Not Used 50:Pixhawk AUXOUT1 51:Pixhawk AUXOUT2 52:Pixhawk AUXOUT3 53:Pixhawk AUXOUT4 54:Pixhawk AUXOUT5 55:Pixhawk AUXOUT6 111:PX4 FMU Relay1 112:PX4 FMU Relay2 113:PX4IO Relay1 114:PX4IO Relay2 115:PX4IO ACC1 116:PX4IO ACC2	Digital pin that enables/disables rangefinder measurement for an analog rangefinder. A value of -1 means no pin. If this is set, then the pin is set to 1 to enable the rangefinder and set to 0 to disable it. This can be used to ensure that multiple sonar rangefinders don't interfere with each other.
RNGFND_TYPE	1		0:None 1:Analog 2:APM2-MaxbotixI2C 3:APM2-PulseLightI2C 4:PX4-I2C 5:PX4-PWM 6:BBB-PRU 7:LightWareI2C 8:LightWareSerial 9:BeBop 10:MAVLink	What type of rangefinder device that is connected
				This sets the bus address of the sensor, where applicable. Used for the

Pic. 6 Details for parameter setting

## 3.3 STM32 Trans Board Code

Receive and analyze the distance data from TF SERIES

```
// Global Variables
u16 distance = 0;
// Variables used by serial
static u8 Usart1buf[USART1_BUF_SIZE];
```

```
static u8 pointer = 0;

// Serial Port 1 Function Initialize
void USART1_Init(void)
{
    USART_InitTypeDef USART1_InitStructure;
    GPIO_InitTypeDef GPIO_InitStructure;
    NVIC_InitTypeDef NVIC_InitStructure;

    // GPIOA Clock
    RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA, ENABLE);
    // A9 -> TX , A10 -> RX
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9 | GPIO_Pin_10;
    GPIO_InitStructure.GPIO_Speed = GPIO_High_Speed;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;
    GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
    GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
    GPIO_Init(GPIOA,&GPIO_InitStructure);

    // Alternative Function Configuration
    GPIO_PinAFConfig(GPIOA,GPIO_PinSource9,GPIO_AF_USART1);
    GPIO_PinAFConfig(GPIOA,GPIO_PinSource10,GPIO_AF_USART1);
    // USART1 clock
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1, ENABLE);
    // USART1 Interrupt Priority
    NVIC_InitStructure.NVIC_IRQChannel = USART1_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);
    // USART1 Initialize
    USART_DeInit(USART1);
    USART1_InitStructure.USART_BaudRate = 115200;
    USART1_InitStructure.USART_WordLength = USART_WordLength_8b;
    USART1_InitStructure.USART_StopBits = USART_StopBits_1;
    USART1_InitStructure.USART_Parity = USART_Parity_No;
    USART1_InitStructure.USART_Mode = USART_Mode_Rx|USART_Mode_Tx;
    USART1_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
    USART_Init(USART1,&USART1_InitStructure);
    USART_Cmd(USART1,ENABLE);
    USART_ITConfig(USART1,USART_IT_RXNE,ENABLE);
}

// Serial Interrupt Function
void USART1_IRQHandler(void)
{
    if(USART_GetITStatus(USART1, USART_IT_RXNE) != RESET
        || (USART_GetITStatus(USART1, USART_IT_ORE_RX) != RESET))
    {
        USART_ClearITPendingBit(USART1, USART_IT_RXNE);
        Usart1buf[pointer++%USART1_BUF_SIZE] = USART_ReceiveData(USART1);

        // Receive data program
        if((pointer%USART1_BUF_SIZE >= 9))
        {
```

```
        // Check the head and the tail of the packet
        // Better do a checksum, not written here
        if( (Usart1buf[pointer%USART1_BUF_SIZE-3]==0x59)
            &&(Usart1buf[pointer%USART1_BUF_SIZE-4]==0x59))
        {
            //Distance
            distance=((u16)((Usart1buf[pointer%USART1_BUF_SIZE-1]<<8)
                |(u16)(Usart1buf[pointer%USART1_BUF_SIZE-2]));)
        }
    }
}
```

#### DAC Configuration

```
void DAC_Config(void)
{
    GPIO_InitTypeDef GPIO_InitStructure;

    RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA, ENABLE);
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_DAC, ENABLE);

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_4;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AN;
    GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_InitStructure.GPIO_OType = GPIO_OType_OD;
    GPIO_Init(GPIOA, &GPIO_InitStructure);

    DAC_InitTypeDef DAC_InitStructure;
    DAC_DeInit();
    DAC_InitStructure.DAC_Trigger = DAC_Trigger_Software;
    DAC_InitStructure.DAC_WaveGeneration = DAC_WaveGeneration_None;
    DAC_InitStructure.DAC_OutputBuffer = DAC_OutputBuffer_Enable;
    DAC_Init(DAC_Channel_1, &DAC_InitStructure);
    DAC_Cmd(DAC_Channel_1,ENABLE);

    DAC_SetChannel1Data(DAC_Align_12b_R,0x1fff);

    DAC_SoftwareTriggerCmd(DAC_Channel_1,ENABLE);
}
```

#### Main function:

```
// The test height of TE-01 ,Unit mm
float test_height;
// Height bias, Unit mm
float bias = 180;
// Analog value output
s16 analog=0;

// Main
void main(void)
{
    // Interrupt Group Configuration
    NVIC_Config();
    // Serial Initialize
    USART1_Init();
}
```

```

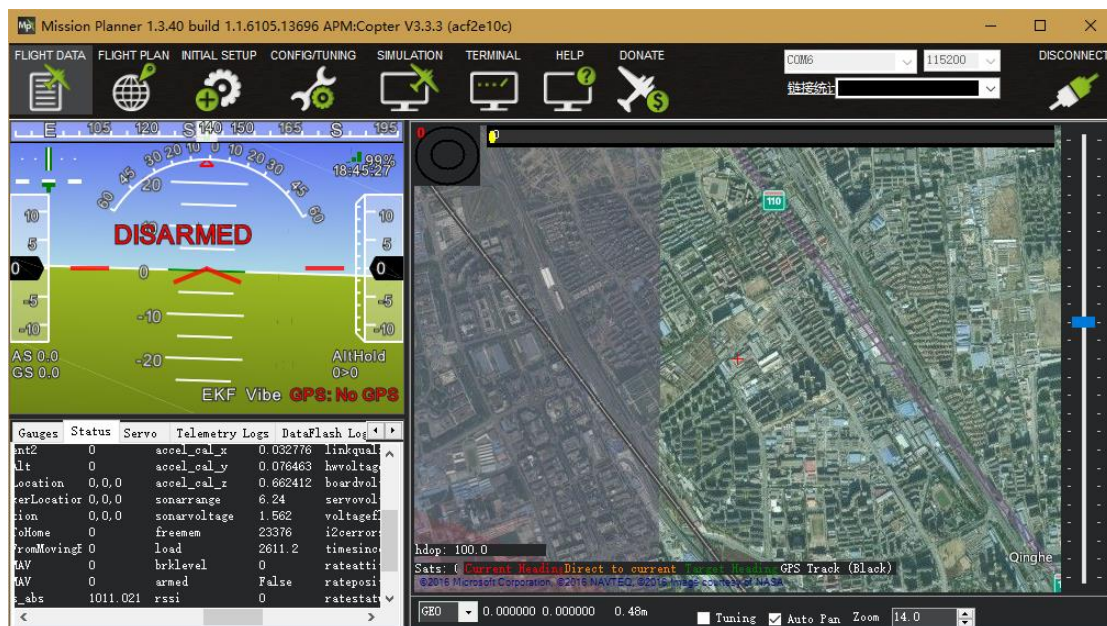
// DAC Config
DAC_Config();

// Main Loop
while(1)
{
    // Remove Bias
    test_height = distance - bias;
    // Data Conversion, 3V corresponding to the height of 12m, therefore
    // test_height * 3.3 * 4096/ (3 * 1200) = test_height * 4096 / 1320
    analog = (s16)(test_height*4096/1320);
    // Range Limit
    analog = analog < 4095 ? analog : 4095;
    analog = analog > 0 ? analog : 0;
    // Voltage Output
    DAC_SetChannel1Data(DAC_Align_12b_R,analog);
    DAC_SoftwareTriggerCmd(DAC_Channel_1,ENABLE);
}
}

```

## 4. Data Test

In Flight Data of Mission Planner, Click **Status** below, find **sonarrange(actual distance)** and **sonarvoltage(analog input voltage)**.



Pic 7 Distance Sensor Test (Test if the sensor gets readings correctly)

## 5. PID configuration

All PID configuration for PixHawk can be done on Mission Planner. See in <http://ardupilot.org/copter/docs/common-tuning.html>.