Radar wall following: straight line

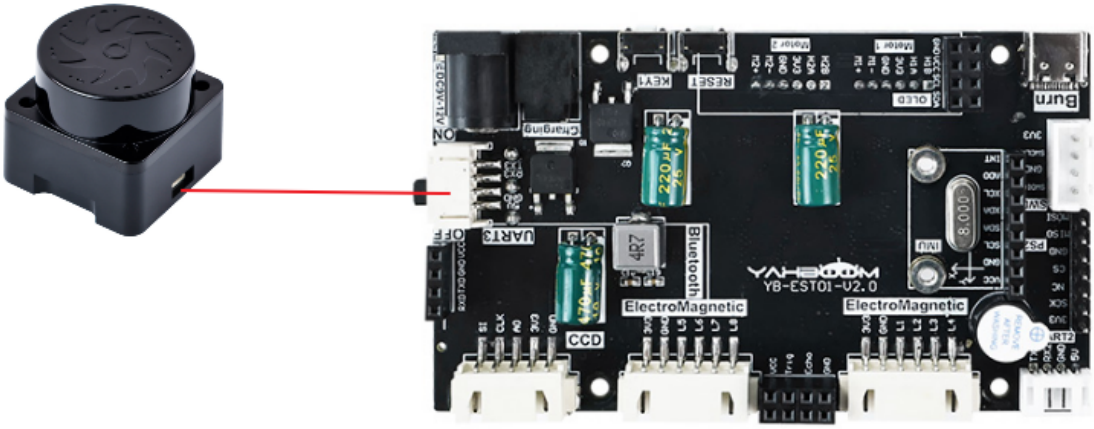
The tutorial mainly demonstrates the function of the balance car combined with the Tmini-Plus radar to follow the wall and walk in a straight line.

```
The tutorial only introduces the standard library project code
```

Hardware connection

Since we have configured a special connection line, we only need to install it to the corresponding interface.

Peripherals	Development Board
Tmini-Plus Radar: VCC	5V
Tmini-Plus Radar: TXD	PC10
Tmini-Plus Radar: RXD	PC11
Tmini-Plus Radar: GND	GND

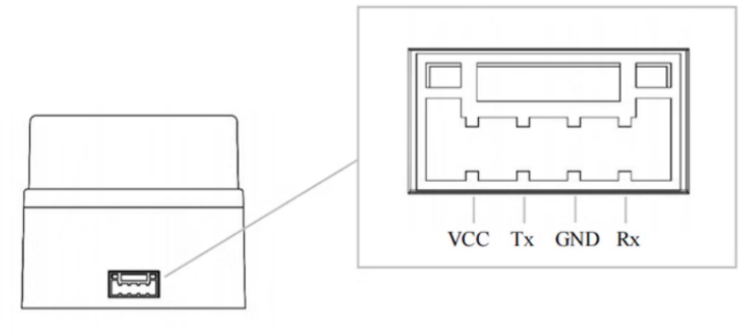


Control Principle

The radar data is parsed by the program, and the distance from the wall and the distance to be maintained for patrolling the wall are set according to the distance collected by the program initialization (radar 72° distance). The 0° distance determines whether there is an obstacle in front.

```
The right side of the radar needs to be close to the wall, and the angle of
distance measurement is 72°
```

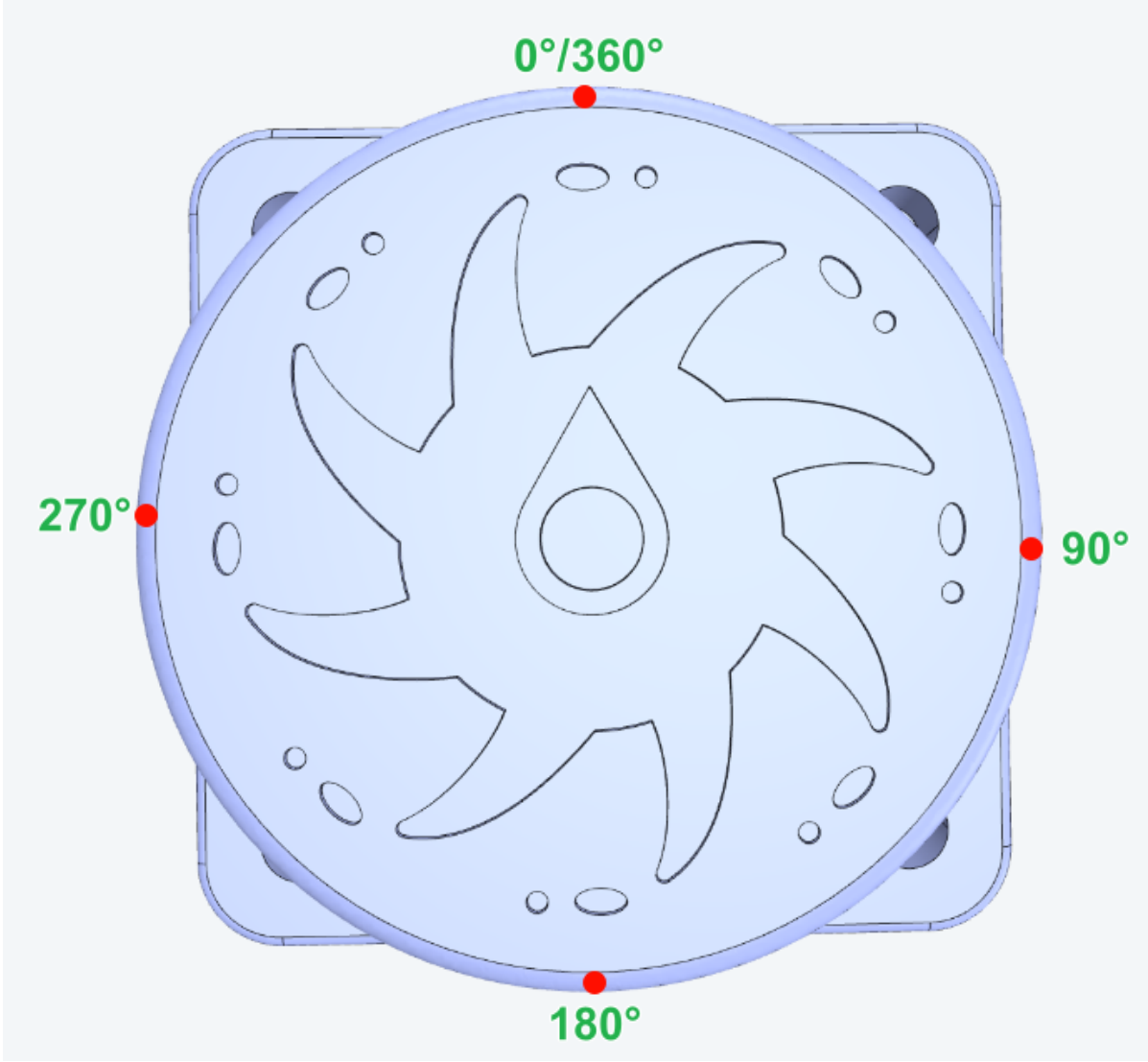
- Tmini-Plus radar



Product name	**Tmini-Plus radar**
Scanning frequency	6-12Hz
Sampling frequency	4000 times/s
Measuring radius	Black object: 12m
Minimum measuring distance	0.05m
Distance measurement principle	TOF distance measurement
Scanning angle	360°
Communication interface	Standard asynchronous serial port (UART) 1. Baud rate: 230400 2. Data bits: 8 3. Check bit: None 4. Stop bit: 1
ROS support	ROS1/ROS2
Windows support	Host computer

> **Radar angle distribution**

The arrow in the center of the radar points to 0°/360°, and the angle increases clockwise.



> **Communication protocol**

```
For detailed information, please refer to the "T_Mini_Plus Manual"
```

Main code

The tutorial mainly explains the code for the radar to follow the wall in a straight line. For detailed code, please refer to the corresponding project file.

> **LiDar_Straight**

The distance data of the radar at 0° is used to determine whether there is an obstacle in front of the balance car, and the distance of the radar at 72° is used to determine the distance between the balance car and the wall, and PID processing is performed based on this distance.

```
void LiDar_Straight(void)
{
    static u16 target_distance=0;

    u16 current_distance=target_distance;

    static u16 Limit_distance=0;   //Radar's maximum detection range
    static u16 get_timedis = 0;

    for (u8 i = 0;i<5;i++) //The detection distance is 72 degrees
```

```
    {
        if(get_timedis<GET_LidarDIS_Time)
        {
            if(Tminidis[Lidar_Angle]==0 || Tminidis[Lidar_Angle]>400) continue;
//Discard data with a size of 0 or greater than 400 mm

            get_timedis ++;
            target_distance = Tminidis[Lidar_Angle];//Dynamically obtain the
target angle
            Limit_distance=target_distance+200;//200mm larger than the target
distance, mainly to avoid the disappearance of the reference object causing the
car to turn quickly

            if(get_timedis == (GET_LidarDIS_Time-1))
            {
                g_lidar_go_flag = 1;
                get_timedis = GET_LidarDIS_Time;
            }
        }

         if(Tminidis[Lidar_Angle]<(Limit_distance))//Limit the detection range of
the radar
        {
             current_distance=Tminidis[Lidar_Angle];//Determine distance
        }

    }

    if(get_timedis<GET_LidarDIS_Time) //When the distance is not determined, do
not do anything
    {
        return;
    }

    if(Tminidis[0]>0 && Tminidis[0]<400)//Stop due to an obstacle ahead
    {
        Move_X = 0;
        Move_Z = 0;
        return;
    }

    Set_track_speed();
    Move_Z = Distance_Adjust_PID(current_distance,target_distance);
}
```

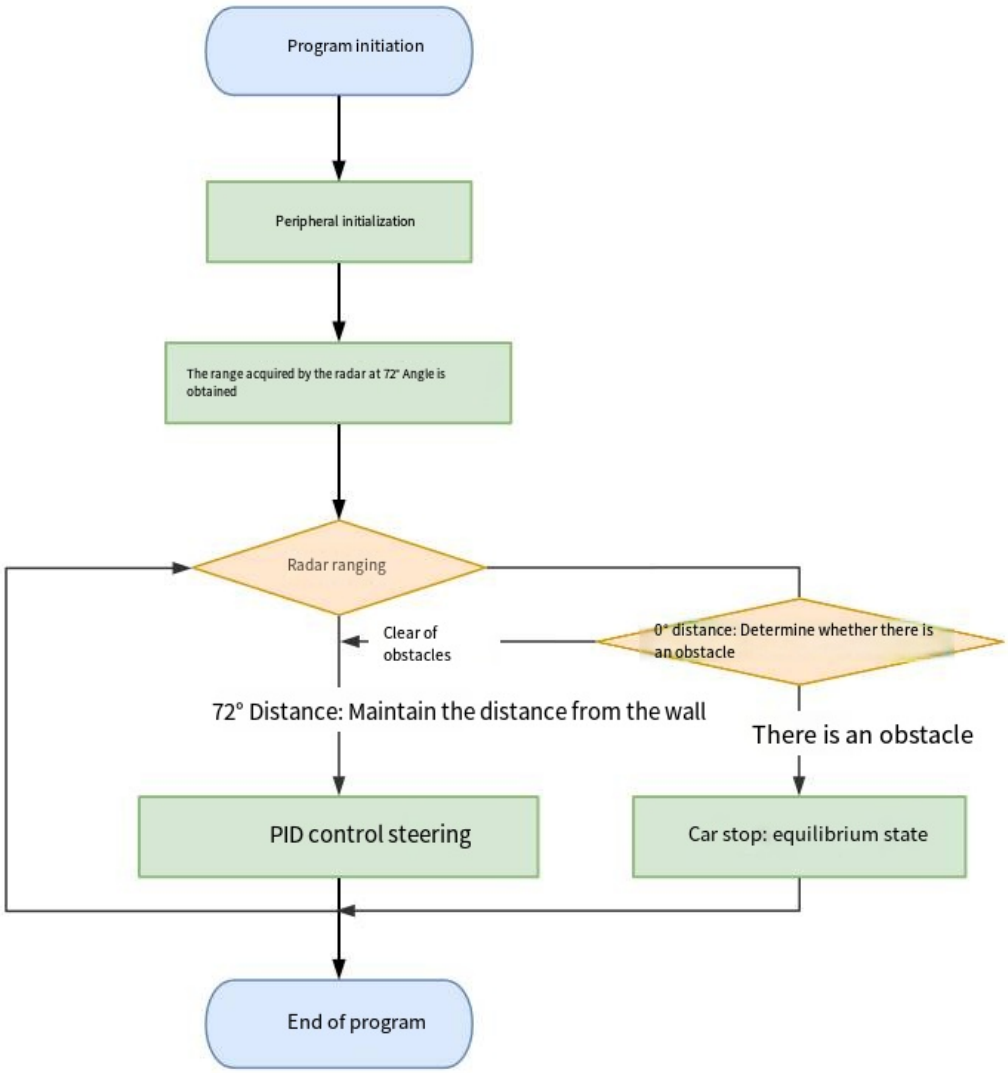
> **Distance_Adjust_PID**

Perform PID processing based on the distance between the radar and the wall. If the balancing car does not patrol the wall in a straight line effectively, modify the PID parameters of the app_lidar_car.c file. It is not recommended to modify the PID parameters of the pid_control.c file (the PID parameters of the pid_control.c file shall be based on the parameters finally confirmed in the balancing car parameter adjustment tutorial).

```
float Distance_Adjust_PID(float Current_Distance,float Target_Distance)//Distance
Adjustment PID
{
    static float error,OutPut,Last_error;
    error=Target_Distance-Current_Distance;
//Calculate Deviation
    OutPut=-Track_Lidar_KP*error-Track_Lidar_KD*(error-Last_error);
//Position PID controller
    Last_error=error;
//Save last deviation
    return OutPut;
}
```

Program flow chart

Briefly introduce the process of function implementation:



Experimental phenomenon

> **Software code**

The Balance_Radar_Line.hex file generated by the project compilation is located in the OBJ folder of the Balance_Radar_Line project. Find the Balance_Radar_Line.hex file corresponding to the project and use the FlyMcu software to download the program into the development board.

```
Product supporting materials source code path: Attachment → Source code summary →
5.Balanced_Car_Extended → 16.Balance_Radar_Line
```

> **Experimental phenomenon**

After the program is started, press KEY1 according to the OLED prompt to start the radar of the balance car to follow the wall in a straight line:

OLED displays wait get dis!: The balance car will take 1 second to obtain the distance that the radar needs to patrol the wall (the right side of the radar needs to be close to the wall, and the source of the patrol distance is the 72° position of the radar); OLED displays start track!: Start the radar wall-following mode.

If an obstacle is encountered during the patrol process, the wall-following mode will be stopped; remove the obstacle to resume the wall-following mode.

```
The program has voltage detection. If the voltage is less than 9.6V, a low
voltage alarm is triggered and the buzzer will sound.
Common situations for triggering voltage alarms:

1. The power switch of the development board is not turned on, and only the Type-
C data cable is connected for power supply

2. The battery pack voltage is lower than 9.6V and needs to be charged in time
```