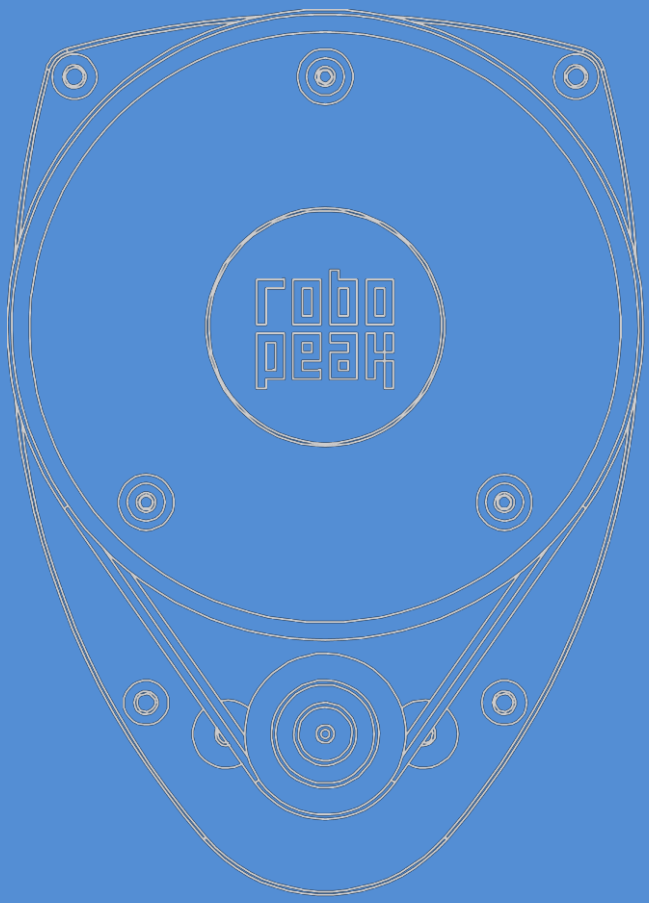
RPLIDAR

Low Cost 360 degree 2D Laser Scanner (LIDAR) System

Introduction and Datasheet

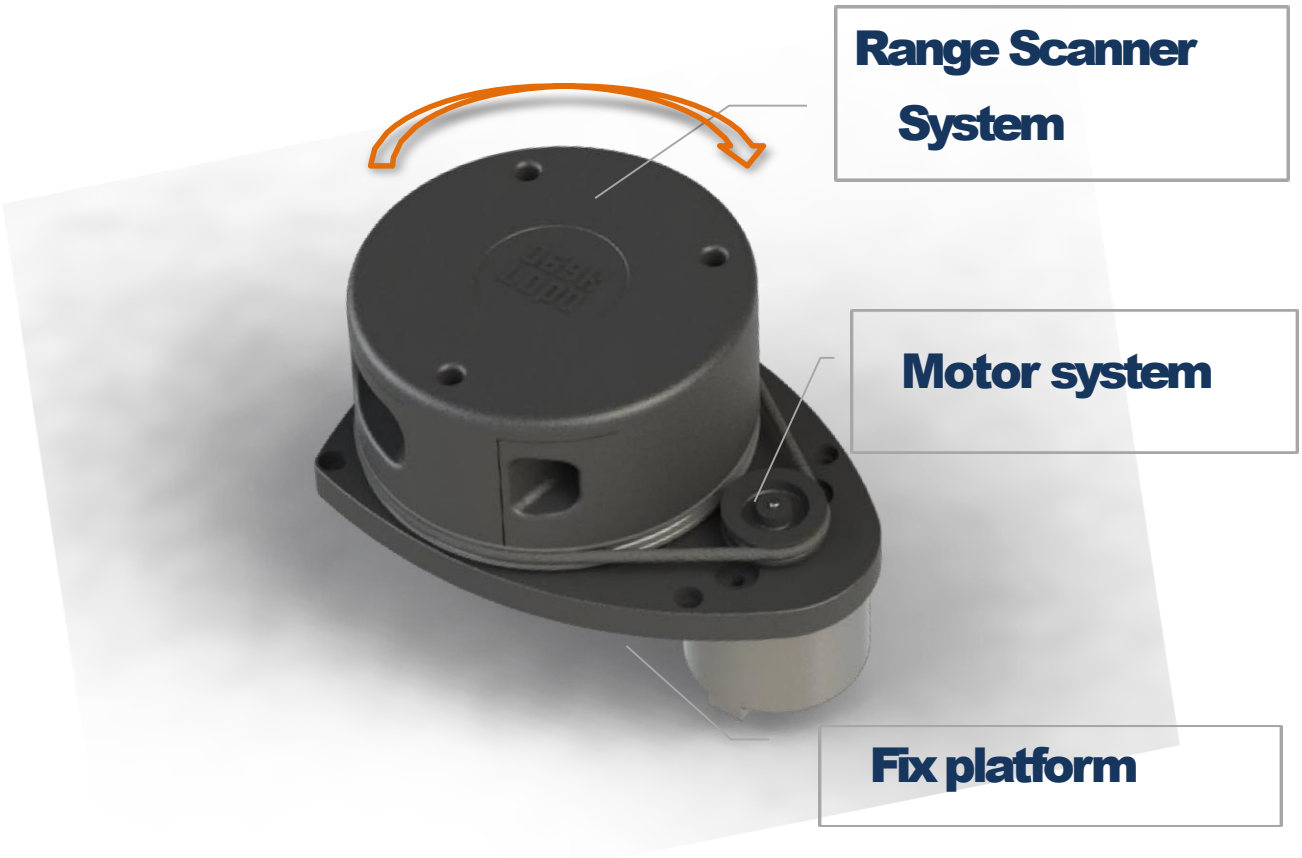
2014-5-6
rev.7

For Model : A1M1

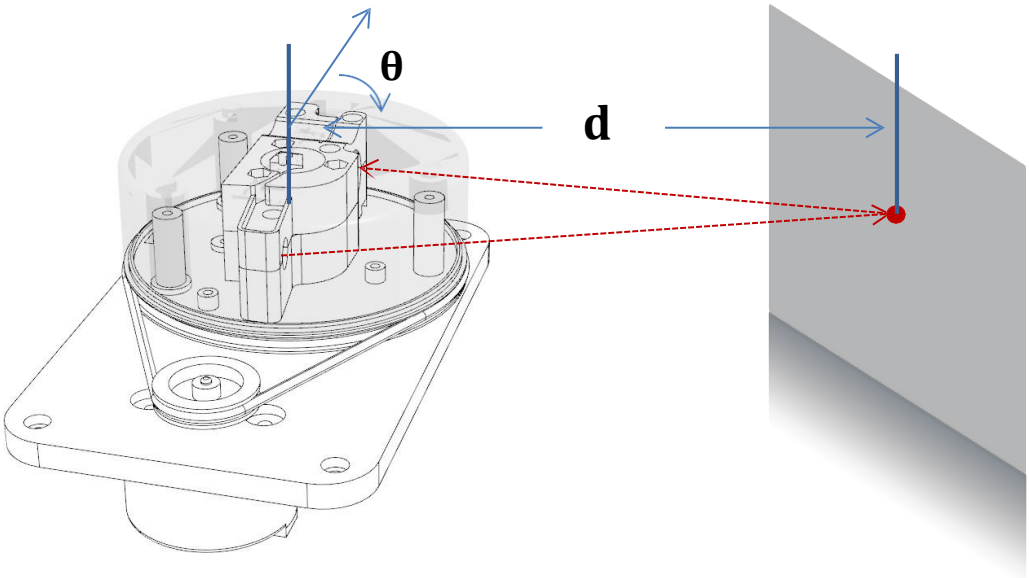


http://www.RoboPeak.com

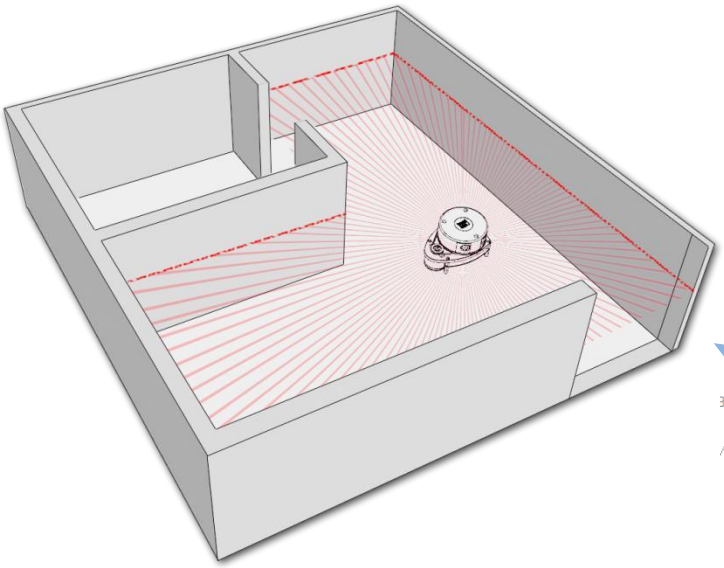
•scanning frequency reached 5.5 hz when sampling 360 points
•can be configured up to 10 hz maximum.



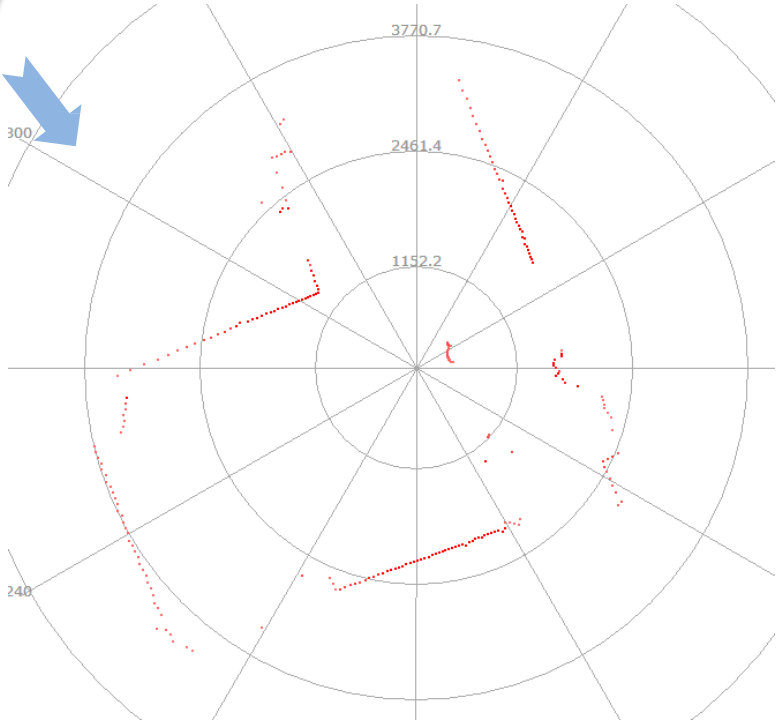
•RPLIDAR is based on laser triangulation ranging principle and uses high-speed vision acquisition and processing hardware developed by RoboPeak. The system measures distance data in more than 2000 times' per second and with high resolution distance output (<1% of the distance).



The high-speed ranging scanner system is mounted on a spinning rotator with a build-in angular encoding system. During rotating, a 360 degree scan of the current environment will be performed.



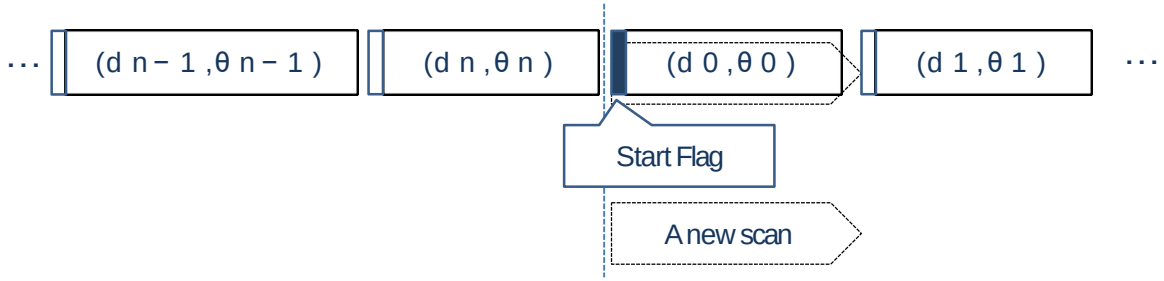
*Note : The LIDAR scan image is not directly relative to the environment showed here. Illustrative purpose only.



RPLIDAR system use a low power (<5mW) infrared laser as its light source, and drives it using modulated pulse. The laser emits in a very short time frame which can make sure its safety to human and pet and reach Class I laser safety standard.

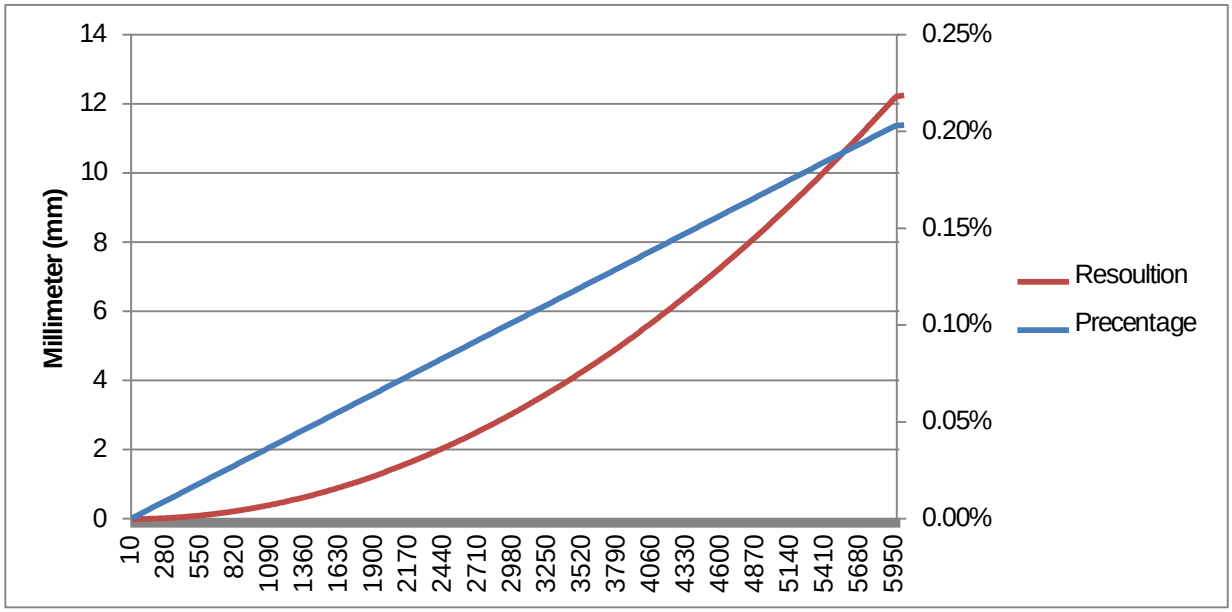
When RPLIDAR is working, sampling data will output to communication interface. Each sample point contains below information. RPLIDAR outputs sampling data continuously. Host systems can configure output format and stop RPLIDAR by sending stop command.

Data Type	Unit	Description
Distance	mm	Current measured distance value
Heading	degree	Current heading angle of the measurement
Quality	level	Quality of the measurement
Start Flag	(Boolean)	Flag of a new scan



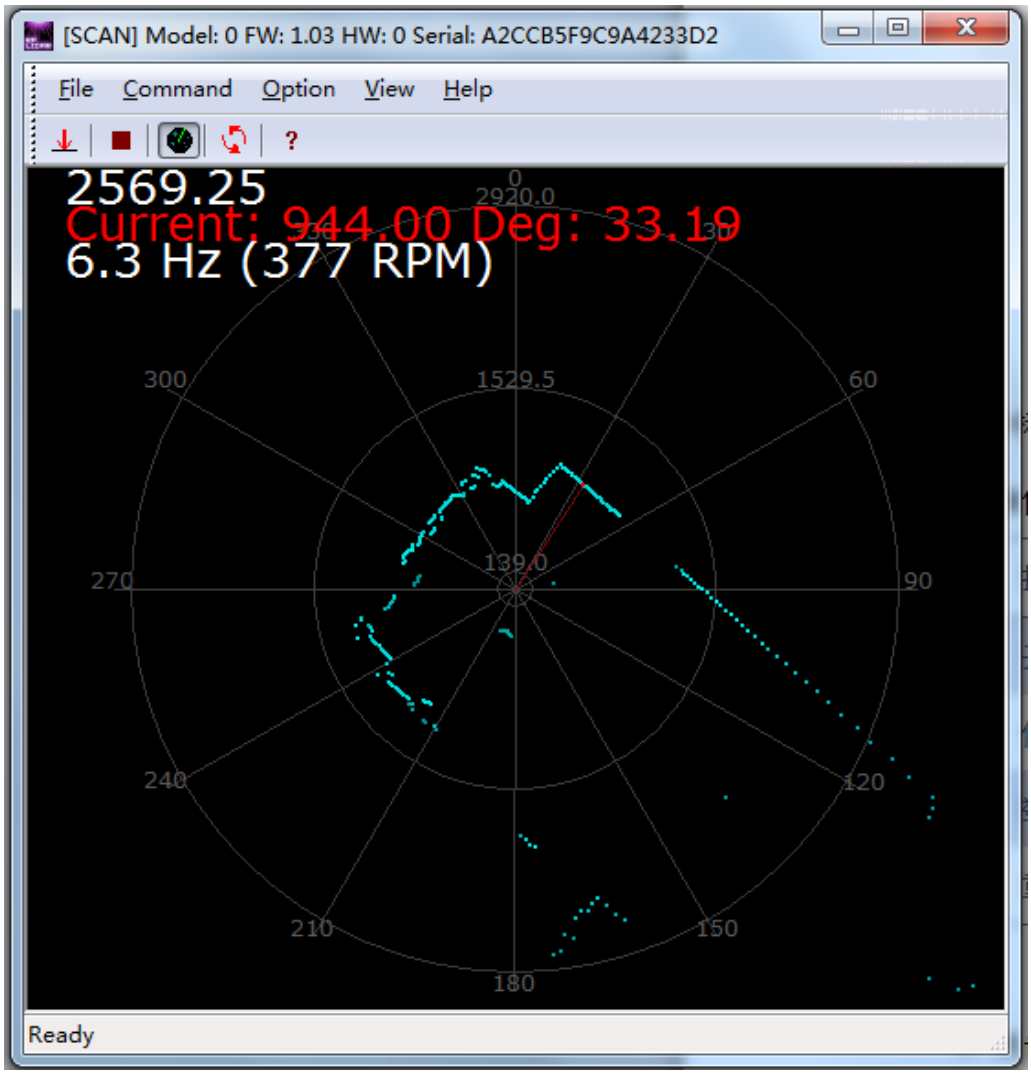
Item	Unit	Min	Typical	Max	Comments
Distance Range	Meter(m)	TBD	0.15 - 6	TBD	White objects
Angular Range	Degree	n/a	0-360	n/a	
Distance Resolution	mm	n/a	<0.5 <1% of the distance	n/a	<1.5 meters All distance range*
Angular Resolution	Degree	n/a	≤1	n/a	5.5Hz scan rate
Sample Duration	Millisecond(ms)	n/a	0.5	n/a	
Sample Frequency	Hz	n/a	≥2000	2010	
Scan Rate	Hz	1	5.5	10	

*Note：triangulation range system resolution changes along with distance change, the below chart showed the theoretical resolution change of RPLIDAR



4. Development SDK and Support

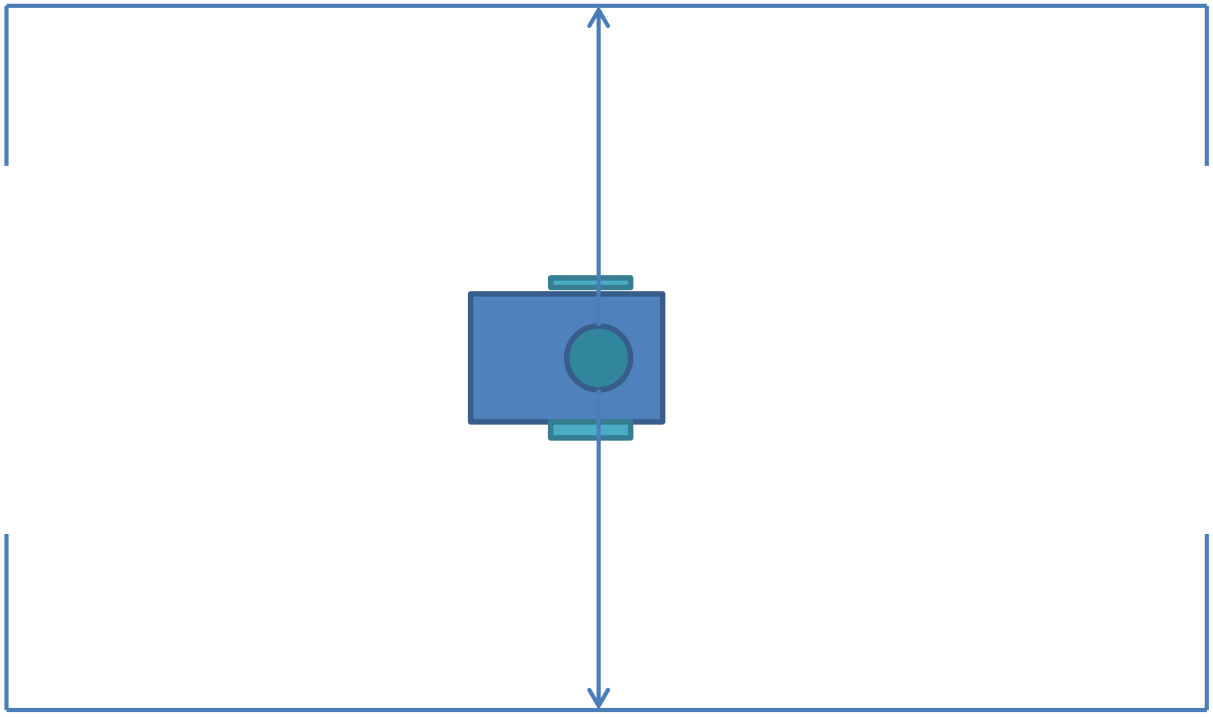
RoboPeak provides debug GUI tool and SDK (available for Windows, x86 Linux and Arm Linux) to speed up product development. Please contact us for detail information.

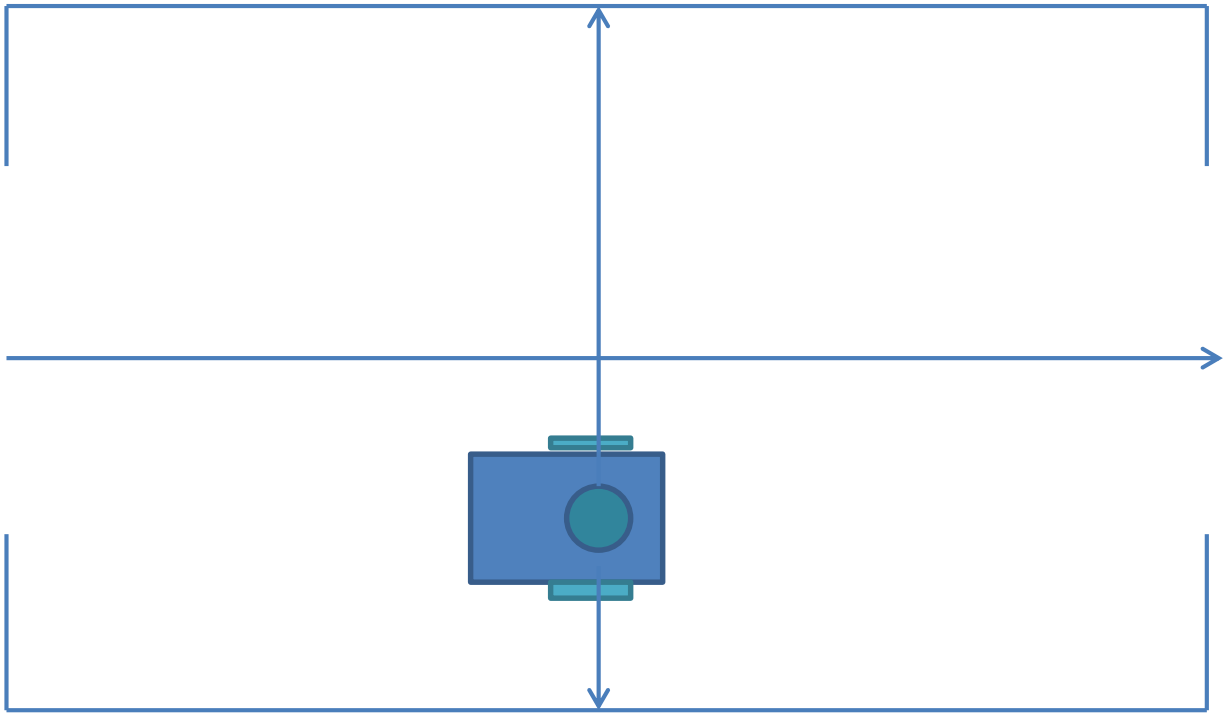


Get Width

```
float localGetWidth(uint16 baseangle)
{
int16 Angle1,Angle2;
int16 Dist1,Dist2;


        Angle1 = baseangle;
        Angle2 = ConverBiDirHeading((float) baseangle + 180);

        Dist1 = LIDARDat[Angle1];//dist in mm
        Dist2 = LIDARDat[Angle2];//dist in mm

        return (Dist1 + Dist2)/25.4;

}
```

Find Center

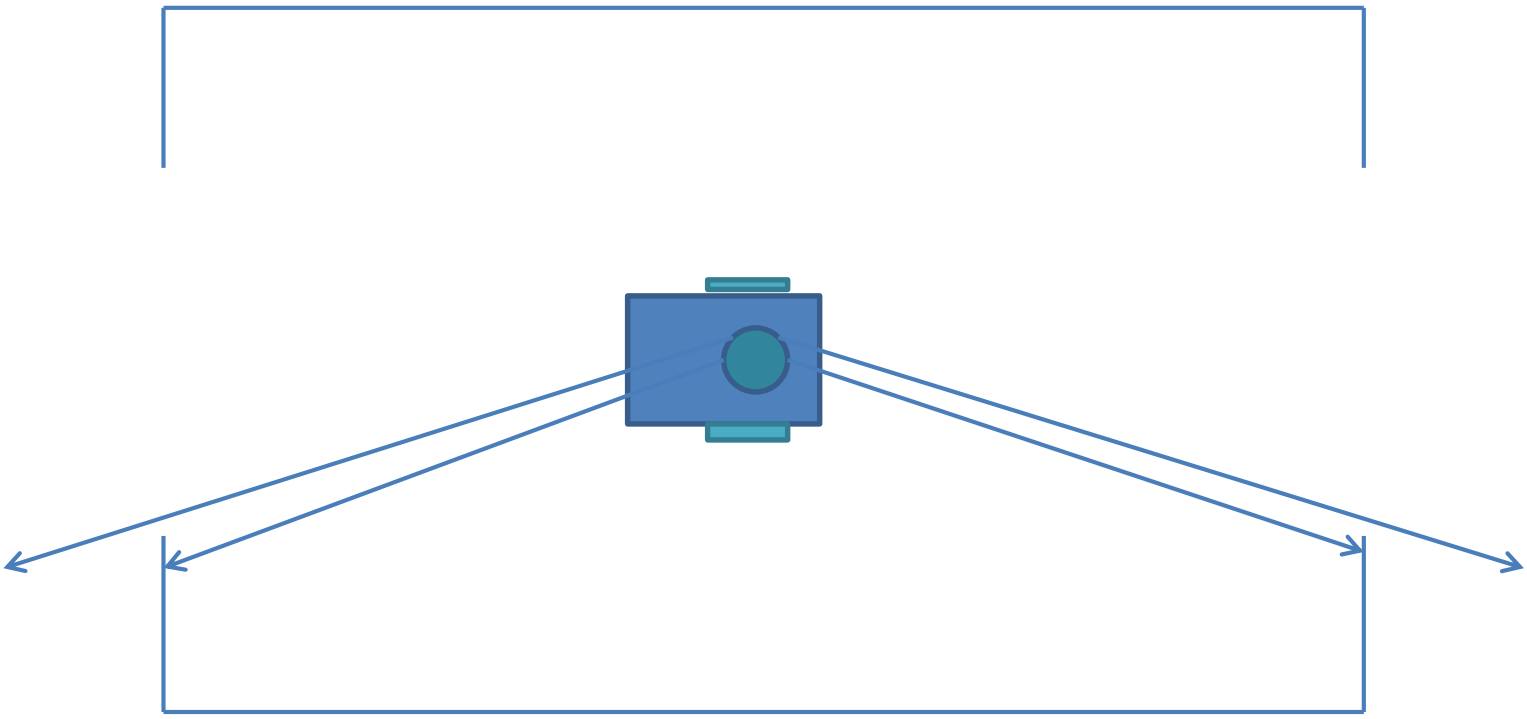




Find Center

```
/*****************************************************************
Returns distance to center baseangle in 0->359
*****************************************************************/
float localFindCenter(uint16 baseangle)
{
int16 Angle1,Angle2;
float RetVal;
int16 Dist1,Dist2;

    //convert to bidirectional heading
    Angle1 = baseangle;//this angle should be 90deg from CCW from our current heading
    Angle2 = ConverBiDirHeading(baseangle + 180);

    Dist1 = LIDARDat[Angle1];//dist in mm to the left
    Dist2 = LIDARDat[Angle2];//dist in mm to the right

     // We're almost in the center
    if(KindaEqual(Dist1 , Dist2,30))
        RetVal = 0;//we are good …..zero out any error
    else
    {//we need to calculate our x offset
        RetVal = (float) ((Dist1 - Dist2)/2);//find center of the difference
        RetVal = RetVal /25.4;//convert to inches
    }

    return RetVal;//this is the distance from our current X position to the center of the course

}
```

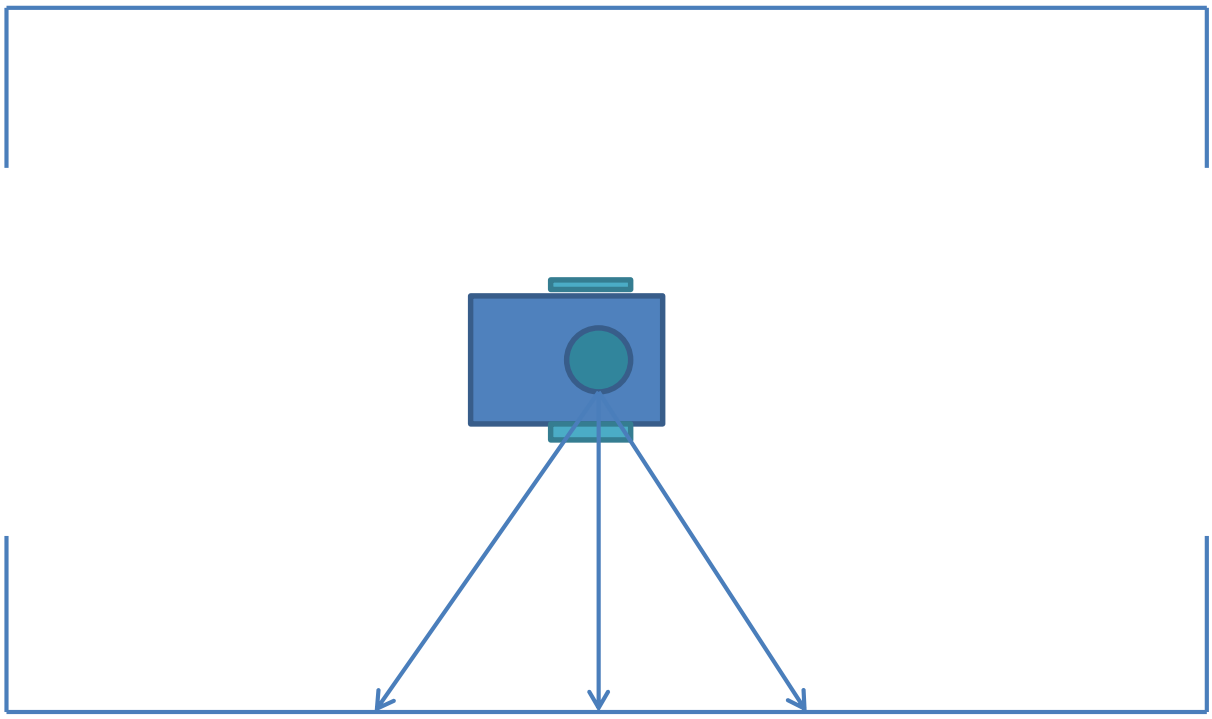
Find Edge



Find Edge

```
void localSetEdges(void)
{
uint32 i,k = 0;

    memset(Edges,0,sizeof(Edges));
    for (i=0; i < LIDAR_DATA_SIZE - 1 ; i++)
    {
        //look for large differnces in consecutive samples
        if(!KindaEqual(LIDARDat[i],LIDARDat[i+1],NVAdj.Adj.EdgeLimit))
        {
            if(LIDARDat[i] < LIDARDat[i+1])//determine what angle the edge is on
                Edges[k] = i;
            else
                Edges[k] = i+1;//load angle into Edges array
            if(k < EDGE_SIZE)
                k++;
            else
              break; //break from loop we are done
        }
    }
}
```

Align to Wall



Align to wall

```
    if( KindaEqual(LIDARDat[75],LIDARDat[105],2))
    {//the length of our angles are the same so we should be aligned
            //.965 = cos(15)
        if(KindaEqual((LIDARDat[105] * .965),LIDARDat[90],NVAdj.Adj.CornerError))
        {//looks good now update heading and X location
            ba[WayPoint].Distance = 0;//stop robot
            if(abs(Location.Heading) > 150)
            {
                Location.Heading = 180;
                Location.Theta = Location.Heading/RADS;
                 //1066 is 1/2 width of CanCan course in mm
                Location.X =  (float)((1066 - LIDARDat[270])/25.4);
             }
            else if(abs(Location.Heading) < 30)
            {
                Location.Heading = 0;
                Location.Theta = Location.Heading/RADS;
                Location.X =  (float)((1066 - LIDARDat[90])/25.4);
            }
            else
                printf("Heading error\r\n");
            State++;
        }
        else
            printf("Corner error\r\n");
    }
    else
    {//we need to align
        if((abs(Location.Heading) < 5) | (abs(Location.Heading) > 175))
        {//we should be close
            if(!KindaEqual(LIDARDat[75],LIDARDat[105],100))
            {//but the lengths are not even close
                Timer = 210;//cancel out
            }
        }
        //rotate to error
        ba[WayPoint].HeadingErr = (LIDARDat[75] - LIDARDat[105]);
        ba[WayPoint].HeadingErr = Clip(ba[WayPoint].HeadingErr,-
            NVAdj.Adj.AlignHeadingError,NVAdj.Adj.AlignHeadingError);
}
```