Algorithm of Multiple Targets

The port 6 is grounded, the output is distance + spectral data, and it includes: data header, distance data, spectral data, and data tail. The format is as below:

0xff, 0xff, 0xff, 0x, 0x, **0x##...0x##, 0x00, 0x00, 0x00**

Header distance Spectral tail

The first three Oxff are data headers, then next 0x** is the upper eight bits in the 16 bits distance information, and the second 0x** is the lower eight bits. The 16 bits binary data represents the distance to the target. (unit: cm) Closest, furthest, average?

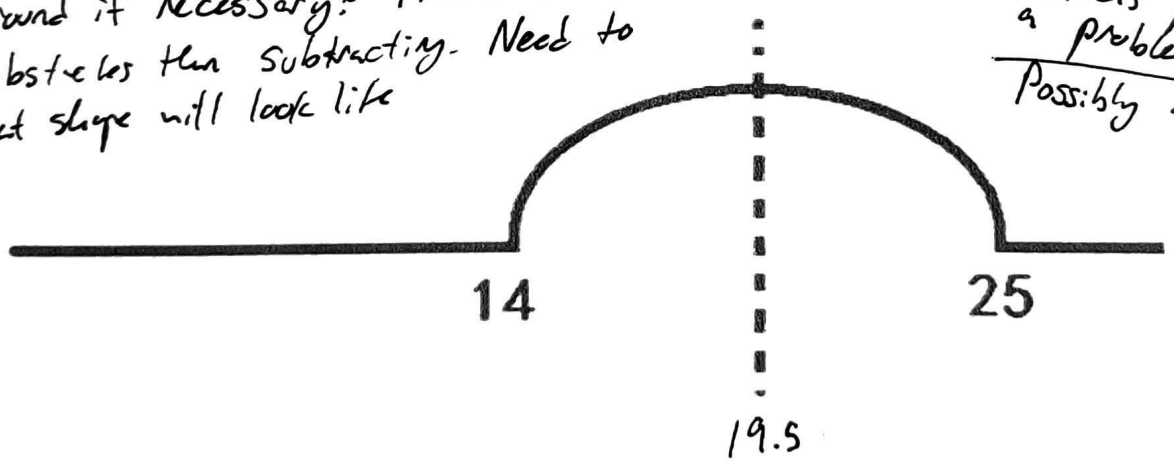
The first 0x## is amplitude of the first spectral line of the distance spectral lines, the next one is that of the second spectral line, and so on. There are 126 spectral lines in total. The amplitude ranges from 1 to 44. After post-processing, users can use these spectral lines to realize mutilple targets detection. The last three 0x00 are data tails, marking the ending of this group of data.

For example: if the amplitude reaches the maximum peak, caculate the central point of the maximum peak. The central point will be the n-th data, then multiply by 0.126 ?

eg. If the spectral line amplitude forms peaks between 14th to 25th: The distance of the point should be: (14+(25-14)/2)*0.126=2.457 (Unit:m)

Will have to look @ how to filter out noise of ground if necessary? Probably w/ measuring w/o obstacles then subtracting. Need to see what shape will look like

will different surfaces pose a problem? Possibly adaptive



So . . .

implies object @ A & B distances

V3-P1

Provided Code from Manufacturer example

Multiple Targets Detection

```
* ***************************************************
* @brief 24GHz Microwave Radar Sensor

 * @copyright    [DFRobot](https://www.dfrobot.com), 2016
 * @copyright    GNU Lesser General Public License

* @author [Xiaoyu](Xiaoyu.zhang@dfrobot.com)
* @version  V1.0
* @date  2019-03-11

* GNU Lesser General Public License.
* All above must be included in any redistribution
* ***************************************************/
#include <SoftwareSerial.h>
char col;// For storing data read from serial port
unsigned char buffer_RTT[134] = {};
int YCTa = 0, YCTb = 0,YCT1 = 0,checka,checkb,Tarnum=1,TargetY1 = 0;
double Tar1a,Tar1b,Distance,Distance1,Distance2,Distance3;
SoftwareSerial mySerial(4,5);
void setup() {
        mySerial.begin(57600);
        Serial.begin(115200);
}

void loop() {
        // Send data only when received data
        if (mySerial.read() == 0xff)
        {
                // Read the incoming byte
           for (int j = 0; j < 134; j++)
           {
                col = mySerial.read();
                buffer_RTT[j] = (char)col;
                delay(2);
           }
                mySerial.flush();
                if(buffer_RTT[1]==0xff){
                if(buffer_RTT[2]==0xff){
                  YCTa = buffer_RTT[3];
                  YCTb = buffer_RTT[4];
                  YCT1 = (YCTa << 8) + YCTb;
                }
                }//Read obstacle distance of the maximum reflection intensity.
               for(int i=6;i<134;i++){
                if(buffer_RTT[i]==buffer_RTT[i-1]){
                  if(buffer_RTT[i-1]>buffer_RTT[i-2]){
                  Tar1a = i-6;
                  checka= buffer_RTT[i-1];
                  } //Check the increase of the peak
                }
                if(buffer_RTT[i]<buffer_RTT[i-1])
```

134 = 3 head + 3 foot + 2 distance + 126 spectral

Jasan device driver
Adapt for STM32F4

should actually start an if going to both

index 6 – 2nd spectral line

if plateaued;
if was increasing:
is a peak

if decreasing: next page

if decreasing:

```
    {
      if(buffer_RTT[i-1]==buffer_RTT[i-2])
      {
        checkb= buffer_RTT[i-1];//Check the decrease of the peak
        if(checka==checkb&&checkb>=10)
        {
        Tar1b = i-6;
        Tar1b=Tar1b-Tar1a;
        Tar1b=Tar1b/2;
        Tar1a=Tar1a+Tar1b;
        Distance=Tar1a*0.126;
        Distance=Distance*100;//Calculate distance
        Serial.print("Distance");
        Serial.print(Tarnum);
        Serial.print(":");
        Serial.println(Distance);//Output the distance of other
obstacles, can read other 3 obstacles at most.
        Serial.print("D: ");
        Serial.println(YCT1);//Output the obstacle distance of the
maximum reflection intensity.
        if(Tarnum==1){
          Distance1=Distance;
        }
        if(Tarnum==2){
          Distance2=Distance;
        }
        if(Tarnum==3){
          Distance3=Distance;
        }
          Tarnum++;
        }
      }
    }
  Tarnum=1;
  }
}
```

if was plateau

maybe allow a bit more leniency than ==

previous might be a peak

if peak from increasing & decreasing sides are the same

Peak halfway between identified peaks

distance = peak location · 0.126 · 100

Save 3 identified peak indexes of of Tarnum

maybe up to 5?

Output Result: output about 5 distance values at most

IMPORTANT: - CAN GIVE MULTIPLE TARGET DISTANCES
- DOES NOT GIVE TARGET DIRECTION.

General Code Structure

Drivers / Device Interaction – Provided by Jason

Processing

Input(s) – Array of spectral lines (current)

- Either input or memory for a number of previous arrays or at least distances

Stage 1

- Filter out road on current sample
- Possibly will need whole previous arrays depending on whether this needs to be adaptive or constants can be used

Stage 2

- Determine current object distances
- Similar to provided algorithm

Stage 3

- Determine relative object speeds (gaining on bike)?

Stage 4

- Possibly combine w/ data from other sensors
- Use logic akin to that from V1-P1 taking speed & distance in to account to determine whether to alert user and vehicle.