Arduino running RPLIDAR C1

Arduino running RPLIDAR C1

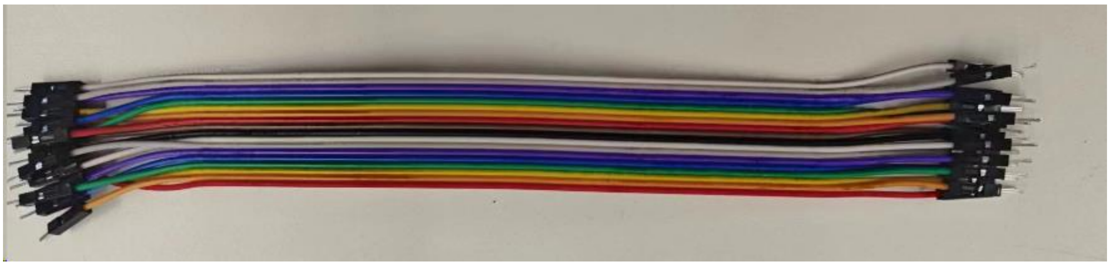
1. Introduction

RPLIDAR C1, as a cost-effective LiDAR, provides a low-cost outdoor ranging and scanning solution for makers. The traditional RPLIDAR SDK has strong performance and can support the entire RPLIDAR series of products, but it uses more upper computer hardware resources. Here, we provide a low hardware requirement RPLIDAR driver solution, which manually parses data packets from RPLIDAR according to the serial port protocol in the datasheet to achieve the same effect. Even Arduino UNO using an 8-bit microcontroller can easily drive the latest RPLIDAR products. In order to display lidar data with additional serial ports, Seeed XIAO SAMD21 with multiple serial ports is used as a demonstration.

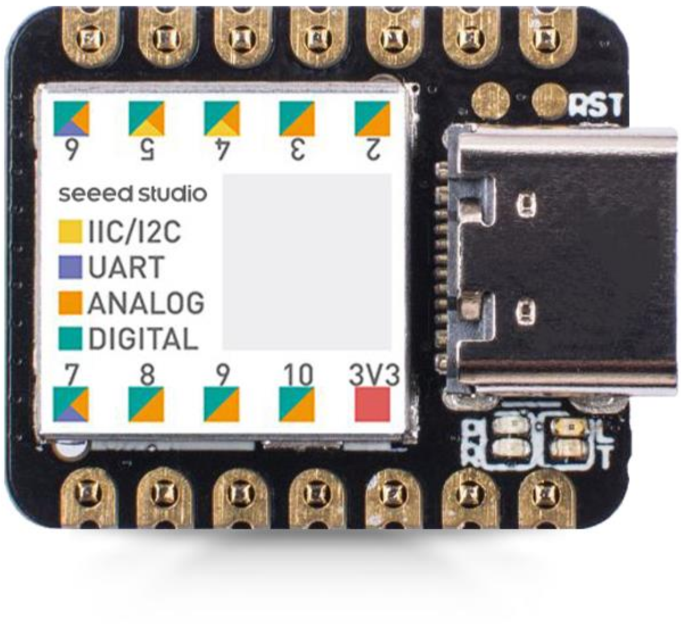
2. Preparations

- One RPLIDAR C1

- male-to-male 7pc DuPont cables (cable length should not be too long, < 15cm)

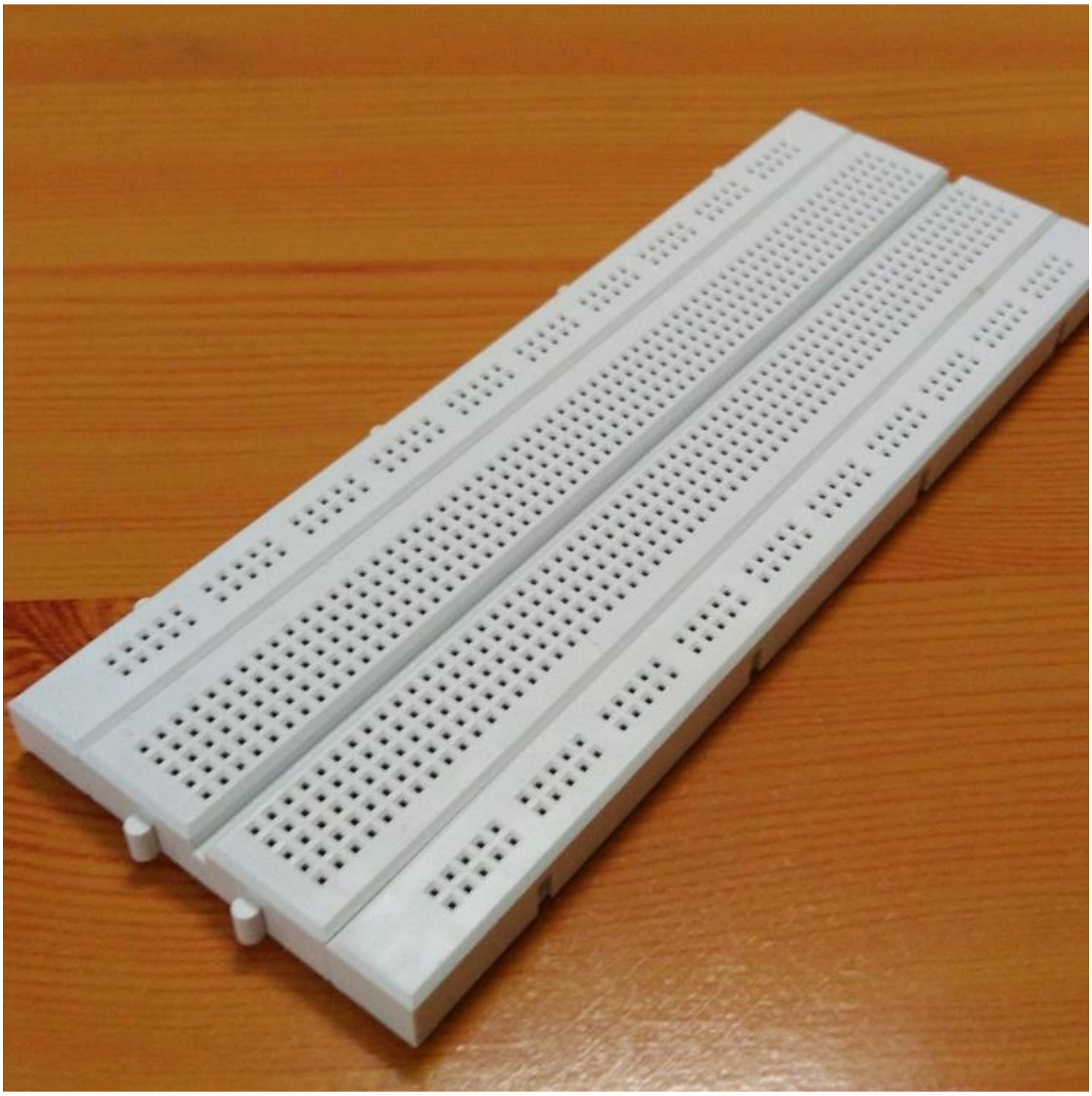


- XIAO SAMD21



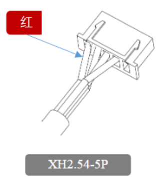
- Passive buzzer

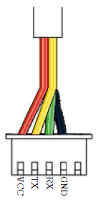
- Breadboard



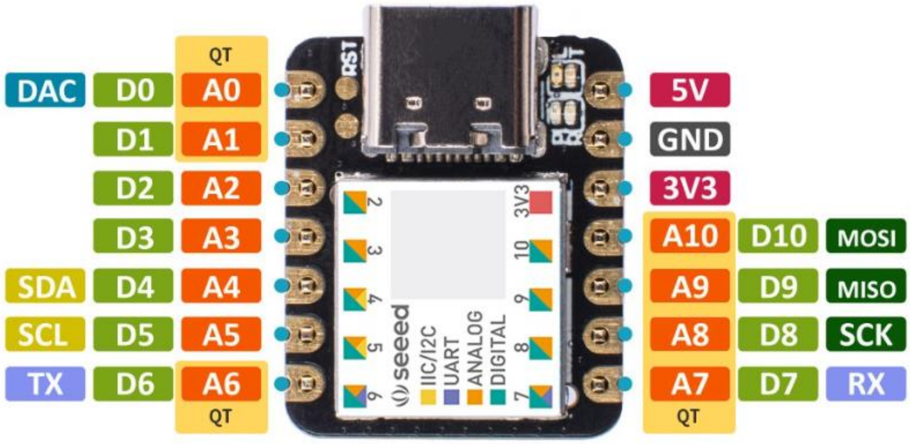
3. XIAO SAMD21 connected to RPLIDAR C1

RPLIDAR C1 is powered by DC 5V power supply, and communicates with XIAO SAMD21 through serial port. The interface definition in the official datasheet of RPLIDAR C1 (can be downloaded in the SLAMTEC resource download center and technical support contact information) is as follows:





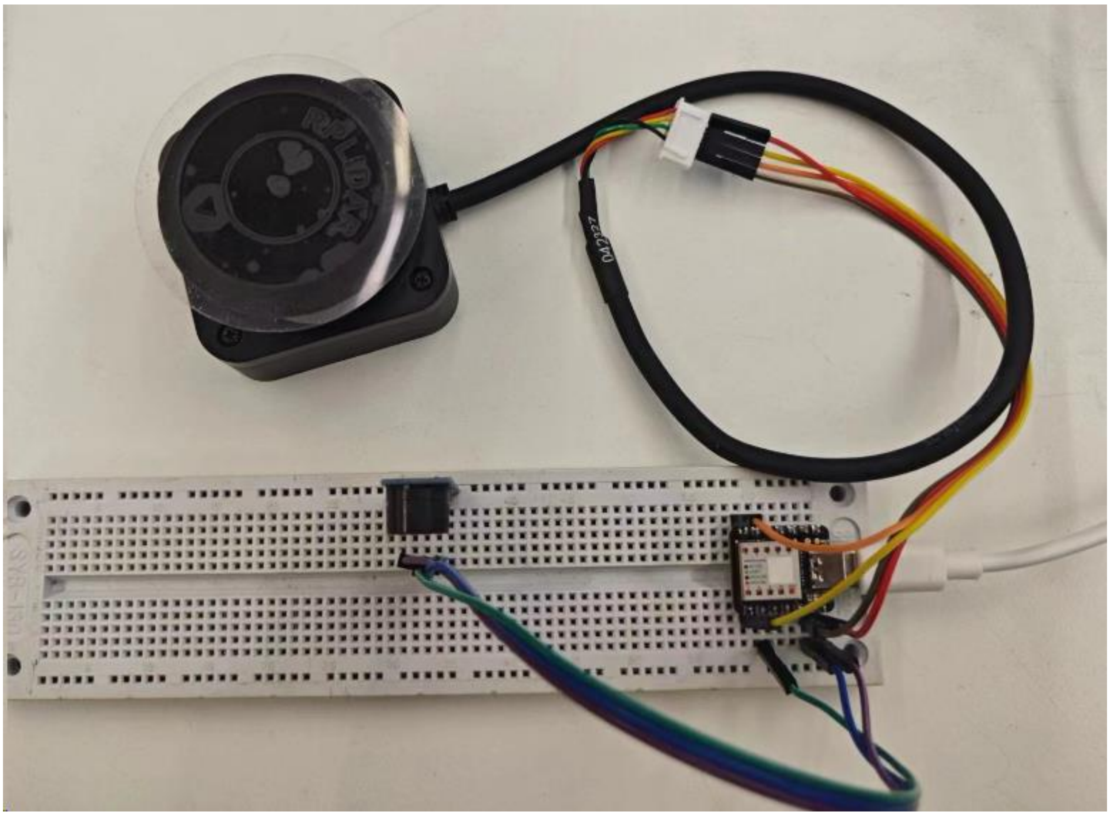
Color	Signal Name	Type	Description	Min	Typical	Max
Red	VCC	Power	Total Power	4.8V	5V	5.2V
Yellow	TX	Output	Serial port output of the scanner core	0V	/	3.5V
Green	RX	Input	Serial port input of the scanner core	0V	/	3.5V
Black	GND	Power	GND	0V	0V	0V



XIAO SAMD21 hardware pinout

XIAO SAMD21	Passive buzzer module	RPLIDAR C1	RPLIDAR logo
5V	5V	Red	VCC
GND	GND	Black	GND
D6		Green	RX
D7		Yellow	TX
D8	I/O		

RPLIDAR C1, XIAO SAMD21, passive buzzer module connection table on breadboard



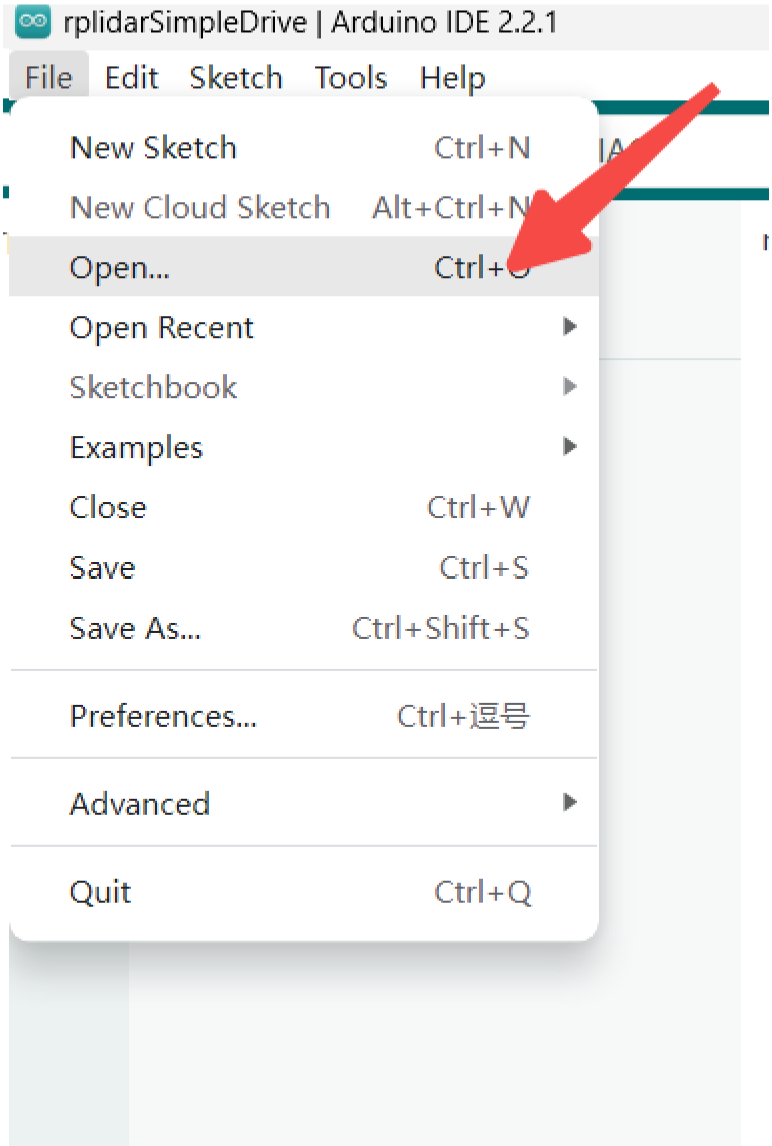
RPLIDAR C1, XIAO SAMD21, passive buzzer connection physical picture

Note: Ensure sufficient power supply, please refer to the Slamtec RPLIDAR C1 datasheet about power supply requirements

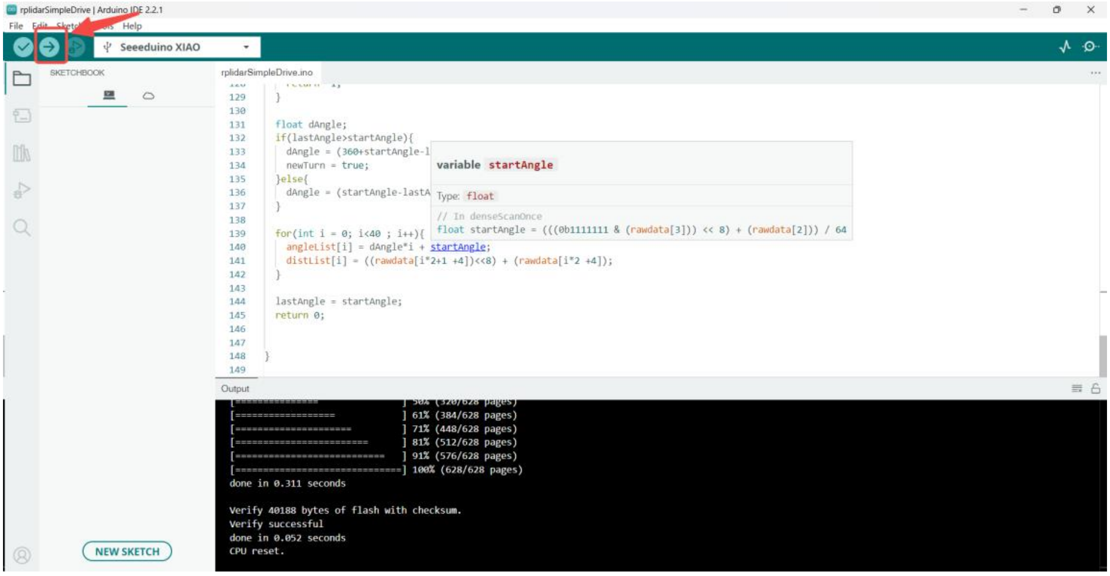
4. Run the sample program on Arduino

Open the sample program rplidarSimpleDrive.ino with Arduino IDE, click Upload to burn the program into the microcontroller. After burning, the microcontroller automatically resets, and the C1 lidar starts running. The microcontroller has lidar data output from the serial port. Place the object in the lidar at an angle of 0-45 degrees and within a range of 300mm. The buzzer will emit different frequencies of sound according to the distance of the object.

4.1 Open the sample program rplidarSimpleDrive.ino



4.2 Click Upload to burn the program to the microcontroller



4.3 Effect Display

23ebe2143f09e9fca32c084c9cfe067c.mp4

5. Arduino sample program analysis

Engineering documents

rplidarSimpleDrive.zip

Here, Serial1 is used to link the radar, and Serial outputs data to the computer through the default serial port

```C
#define RPSERIAL Serial1
#define RPLIDARBAUD 460800

#define MSGSERIAL Serial
#define MSGBAUD 115200
```

The protocol used by RPLIDAR C1 can be seen from the datasheet

```C
#define len_startCmd_dense   9
#define len_endCmd_dense     2
#define len_startReCmd_dense 7
#define measureCmdLen_dense  84
```

```
char startCmd_dense[len_startCmd_dense]    =
{0xA5,0x82,0x05,0x00,0x00,0x00,0x00,0x00,0x22};
char endCmd_dense[len_endCmd_dense]        = {0xA5,0x25};
char startReCmd_dense[len_startReCmd_dense]=
{0xA5,0x5A,0x54,0x00,0x00,0x40,0x85};
char startReCmd_dense_buf[len_startReCmd_dense];
```

The lidar can be started by outputting the start command through the serial port, but in order to ensure that the lidar works in the desired state, the lidar state is reset here

```C
void endScan(){
  RPSERIAL.write(endCmd_dense,len_endCmd_dense);
}

void startScan(){
  endScan();
  RPSERIAL.flush();
  delay(1000);
  RPSERIAL.write(startCmd_dense,len_startCmd_dense);
  MSGSERIAL.println("Starting");
  RPSERIAL.readBytes(startReCmd_dense_buf,len_startReCmd_dense);

if(!chArraryCmp(startReCmd_dense_buf,startReCmd_dense,len_startReCmd_den
se)){
    MSGSERIAL.println("ERROR, Restarting!");
    delay(1000);
    startScan();
  }else{
    MSGSERIAL.println("Init finished");
  }
}
```

Protocol analysis can refer to the specific instructions on data packets in the manual. Each packet of data in C1 contains distance information for 40 points and the starting angle of 40 points. The actual angle of each point needs to be converted based on the angle difference between the two packets.

```C
int denseScanOnce(){
  RPSERIAL.readBytes(rawdata,measureCmdLen_dense);
  char sync1 = rawdata[0]>>4;
```

```
  char sync2 = rawdata[1]>>4;
  char ChkSum = (0b1111 & (rawdata[0])) + ((0b1111 & (rawdata[1]))<<4);
  char S                  = (rawdata[3])>>7;
  float startAngle        = (((0b1111111 & (rawdata[3]))<<8) + (rawdata[2]))/64;

  if(S == 1){
    MSGSERIAL.println("reset database");
  }

  if(((sync1<<4)+sync2) != 0xA5){
    MSGSERIAL.println("INVALID DATA!! RESTARTING");
    startScan();
    return -1;
  }
  float dAngle;
  if(lastAngle>startAngle){
    dAngle = (360+startAngle-lastAngle)/40;
    newTurn = true;
  }else{
    dAngle = (startAngle-lastAngle)/40;
  }

  for(int i = 0; i<40 ; i++){
    angleList[i] = dAnglei + startAngle;
*    distList[i] = ((rawdata[i*2+1 +4])<<8) + (rawdata[i*2 +4]);
  }

  lastAngle = startAngle;
  return 0;


}
```

To start the lidar, run the trigScan function at startup. Here, we also use the LED on the board and a pin to control the buzzer.

```C
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(8, OUTPUT);

  MSGSERIAL.begin(MSGBAUD);
  RPSERIAL.begin(RPLIDARBAUD);
  startScan();
```

```
}
```

The loop contains three parts. The denseScanOnce () function obtains a data packet from RPLIDAR and parses it. Arduino controls the buzzer sound based on the obtained lidar data and outputs the data from the lidar to the serial port. To reduce the amount of displayed data, only the first point of each packet is output here.

```C
void loop()
{

  // Process a packet from RPLIDAR
  // Angle and distance data will be stored
  // into arrary angleList[40] and distList[40]

  // If the packet include angle of a new round, the
  // bool value newTurn will be set to true. Use it or not
  // depends on you. You might need to reset it manually.

  denseScanOnce();



  // Here's a demo of how to use these data.

  if(newTurn){
    newTurn = false;
    buzz = false;
    mindist = 1000;
  }

  for(int i=0;i<40;i++){
    if((angleList[i]>0 && angleList[i] < 45 )\
      &&(distList[i]<300 &&distList[i] >0)){
        buzz = true;
        if(distList[i] < mindist){
          mindist = distList[i];
        }
      }
  }

  if(buzz){
    analogWrite(LED_BUILTIN, 128);
```

```
    tone(8, (uint32_t)(mindist*10));
  }else{
    analogWrite(LED_BUILTIN, 0);
    tone(8, 0);
  }


  MSGSERIAL.print(angleList[0]);
  MSGSERIAL.print(",");
  MSGSERIAL.print(distList[0]);
  MSGSERIAL.print(",");
  MSGSERIAL.println(buzz);

}
```

The following is the completed code

```C
#define RPSERIAL Serial1
#define RPLIDARBAUD 460800

#define MSGSERIAL Serial
#define MSGBAUD 115200

#define len_startCmd_dense   9
#define len_endCmd_dense     2
#define len_startReCmd_dense 7
#define measureCmdLen_dense  84

char startCmd_dense[len_startCmd_dense]    =
{0xA5,0x82,0x05,0x00,0x00,0x00,0x00,0x00,0x22};
char endCmd_dense[len_endCmd_dense]        = {0xA5,0x25};
char startReCmd_dense[len_startReCmd_dense]=
{0xA5,0x5A,0x54,0x00,0x00,0x40,0x85};
char startReCmd_dense_buf[len_startReCmd_dense];


float lastAngle;
float angleList[40];
float distList[40];
char rawdata[measureCmdLen_dense];
```

```
bool newTurn = false;
bool buzz = false;
float mindist = 1000;

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(8, OUTPUT);

  MSGSERIAL.begin(MSGBAUD);
  RPSERIAL.begin(RPLIDARBAUD);
  startScan();

}

void loop()
{

  // Process a packet from RPLIDAR
  // Angle and distance data will be stored
  // into arrary angleList[40] and distList[40]

  // If the packet include angle of a new round, the
  // bool value newTurn will be set to true. Use it or not
  // depends on you. You might need to reset it manually.

  denseScanOnce();



  // Here's a demo of how to use these data.

  if(newTurn){
    newTurn = false;
    buzz = false;
    mindist = 1000;
  }

  for(int i=0;i<40;i++){
    if((angleList[i]>0 && angleList[i] < 45 )\
      &&(distList[i]<300 &&distList[i] >0)){
        buzz = true;
        if(distList[i] < mindist){
          mindist = distList[i];
        }
      }
```

```
  }

  if(buzz){
    analogWrite(LED_BUILTIN, 128);
    tone(8, (uint32_t)(mindist*10));
  }else{
    analogWrite(LED_BUILTIN, 0);
    tone(8, 0);
  }


  MSGSERIAL.print(angleList[0]);
  MSGSERIAL.print(",");
  MSGSERIAL.print(distList[0]);
  MSGSERIAL.print(",");
  MSGSERIAL.println(buzz);

}


bool chArraryCmp(char* arrary1, char* arrary2, int len){
  bool eq = true;
  for(int i = 0; i< len; i++){
    if(arrary1[i] != arrary2[i]){
      eq = false;
    }
  }
  return eq;

}

void endScan(){
  RPSERIAL.write(endCmd_dense,len_endCmd_dense);
}

void startScan(){
  endScan();
  RPSERIAL.flush();
  delay(1000);
  RPSERIAL.write(startCmd_dense,len_startCmd_dense);
  MSGSERIAL.println("Starting");
  RPSERIAL.readBytes(startReCmd_dense_buf,len_startReCmd_dense);

if(!chArraryCmp(startReCmd_dense_buf,startReCmd_dense,len_startReCmd_den
se)){
```

```
    MSGSERIAL.println("ERROR, Restarting!");
    delay(1000);
    startScan();
  }else{
    MSGSERIAL.println("Init finished");
  }
}





int denseScanOnce(){
  RPSERIAL.readBytes(rawdata,measureCmdLen_dense);
  char sync1 = rawdata[0]>>4;
  char sync2 = rawdata[1]>>4;
  char ChkSum = (0b1111 & (rawdata[0])) + ((0b1111 & (rawdata[1]))<<4);
  char S              = (rawdata[3])>>7;
  float startAngle       = (((0b1111111 & (rawdata[3]))<<8) + (rawdata[2]))/64;

  if(S == 1){
    MSGSERIAL.println("reset database");
  }

  if(((sync1<<4)+sync2) != 0xA5){
    MSGSERIAL.println("INVALID DATA!! RESTARTING");
    startScan();
    return -1;
  }
  float dAngle;
  if(lastAngle>startAngle){
    dAngle = (360+startAngle-lastAngle)/40;
    newTurn = true;
  }else{
    dAngle = (startAngle-lastAngle)/40;
  }

  for(int i = 0; i<40 ; i++){
    angleList[i] = dAnglei + startAngle;
*    distList[i] = ((rawdata[i*2+1 +4])<<8) + (rawdata[i*2 +4]);
  }

  lastAngle = startAngle;
  return 0;
```

```
}
```