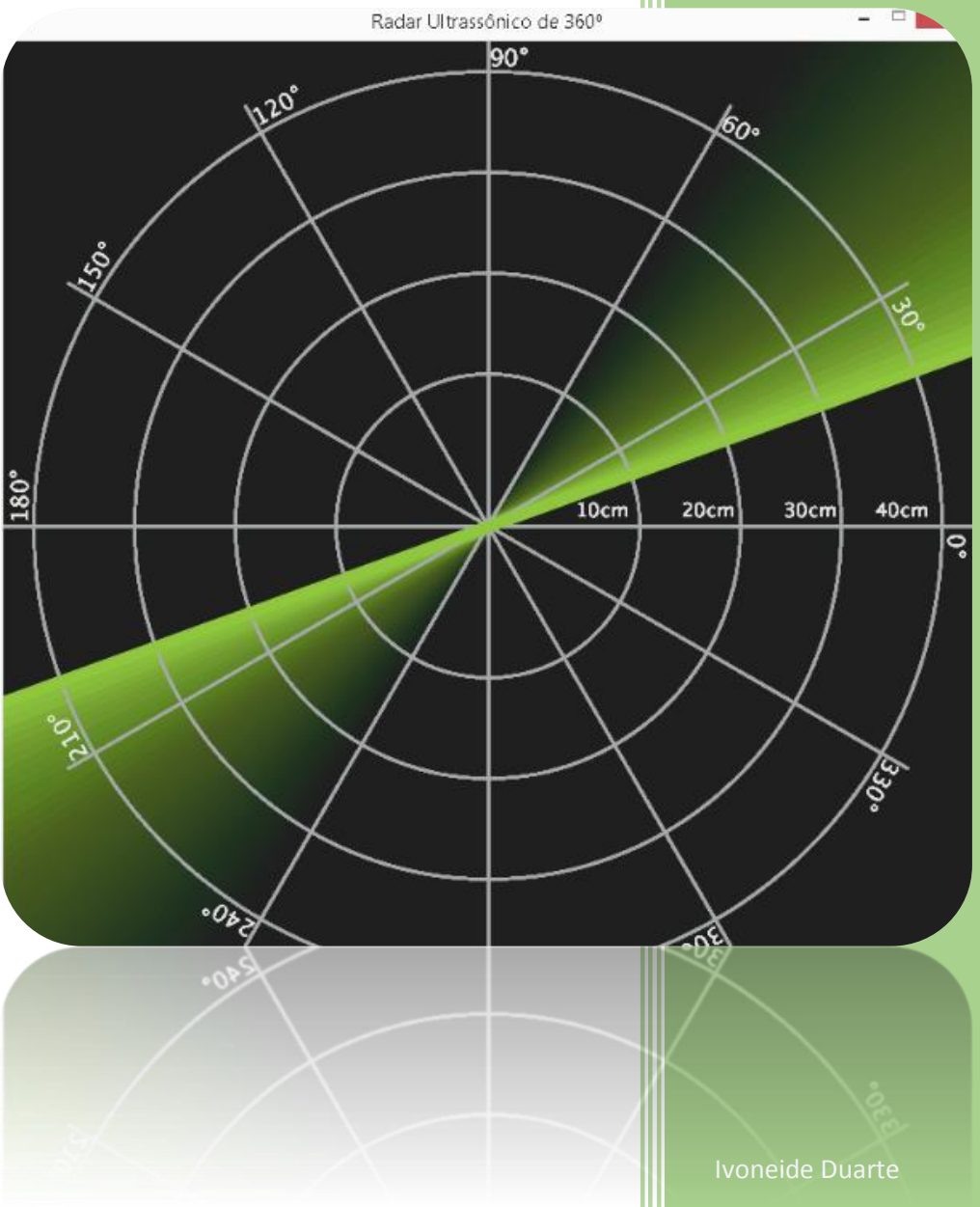
2020

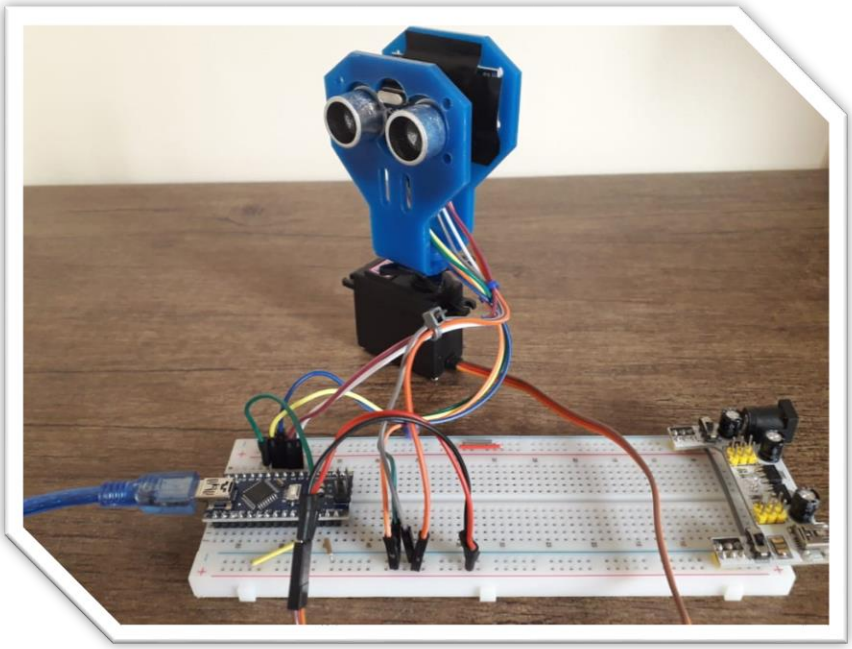
RADAR ULTRASSÔNICO DE 360º

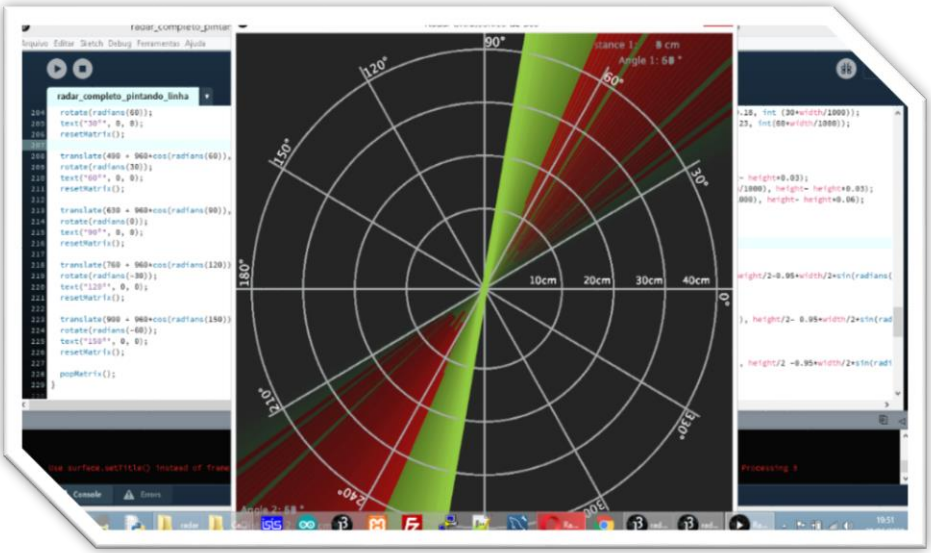


Ivoneide Duarte

29/06/2020

Estou usando neste projeto dois sensores ultrassônicos que serão responsáveis pela captação de obstáculos a sua volta, por meio da distância calculada pelo mesmo e um servo motor que servirá de apoio para fixação dos sensores e fará uma rotação angular de 180º , ajudando assim os sensores na sua tarefa. Contudo, cada sensor irá trabalhar de forma individual e fará uma leitura de 360º à sua volta e mostrará a interatividade com o ambiente através de radar desenvolvido em uma plataforma gráfica.

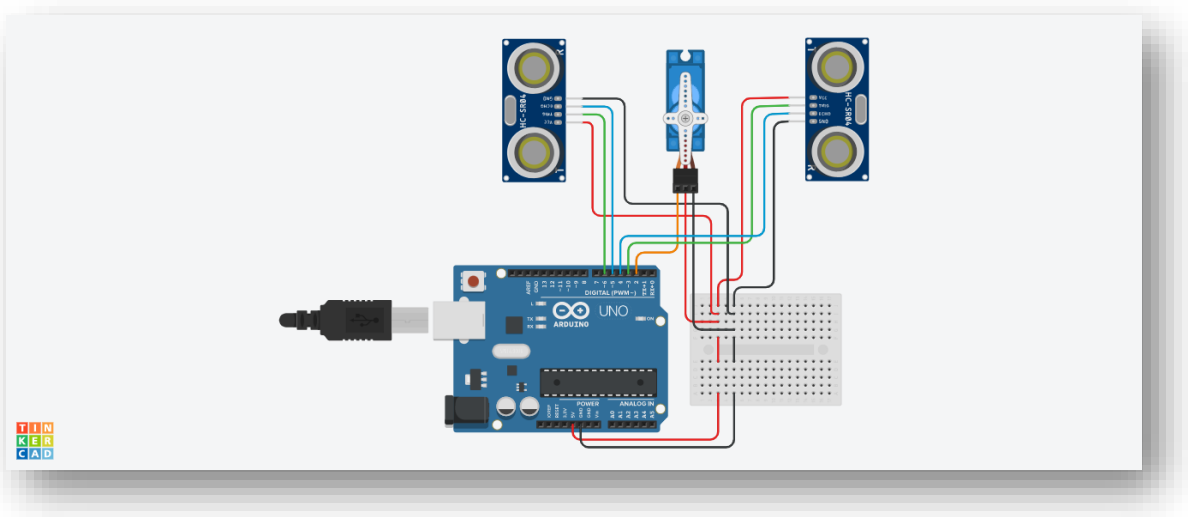




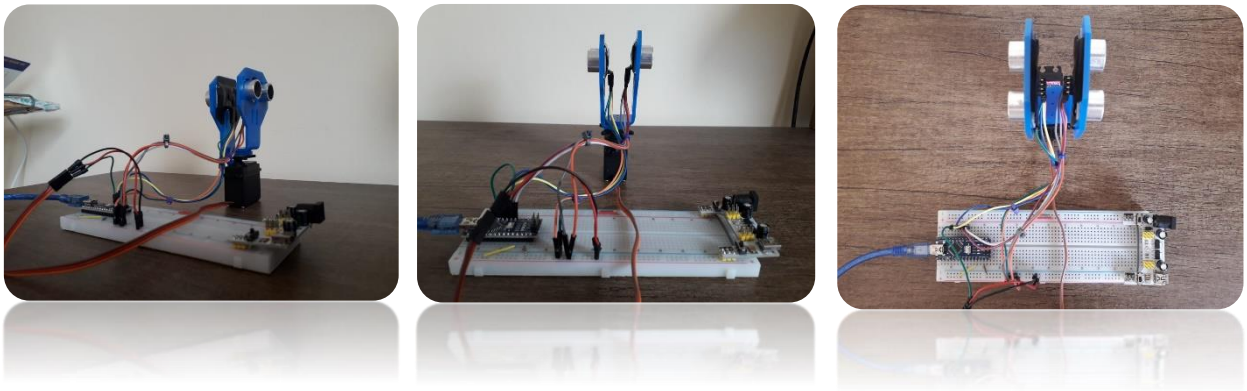
Materiais necessários:

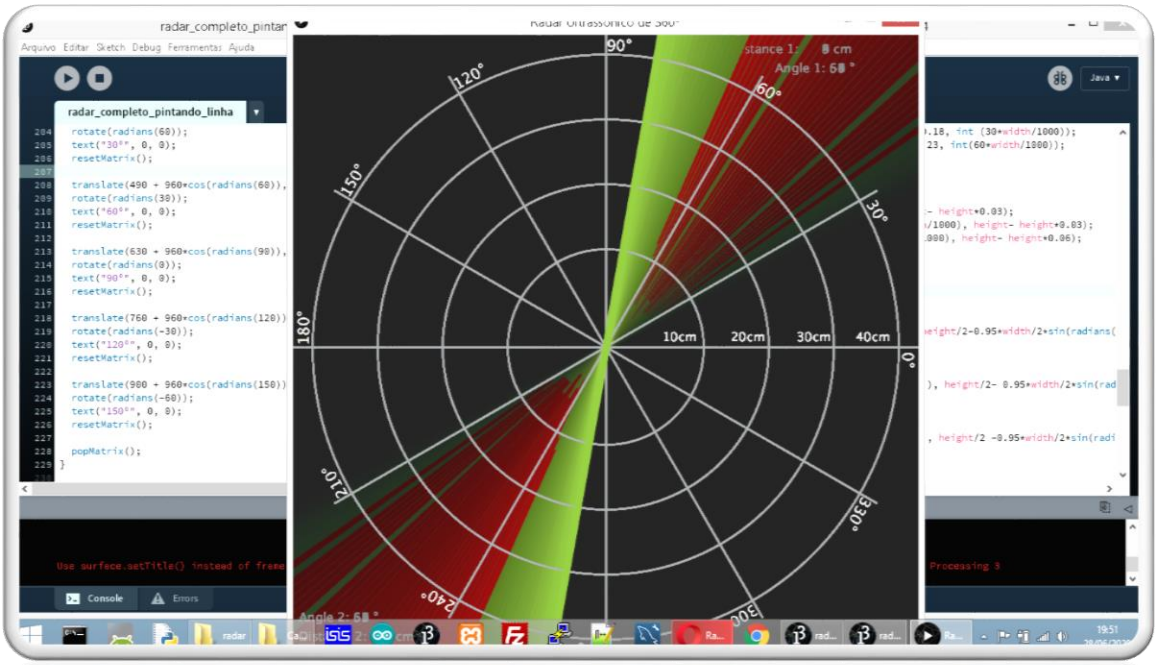
Componentes	Quantidade
Arduino Nano com cabo USB	1 unid.
Sensor HC-SR04	2 unid.
Suporte do sensor HC-SR04	2 unid.
Micro Servo Motor SG90	1 unid.
Mini Protoboard de 170 pontos	1 unid.
Fios jumper M-F / M-M	11 unid.

Montagem do Circuito:



Apresentação do projeto:





Código na IDE Arduino:

```
#include <Servo.h> //Biblioteca do Servo Motor

//Conexões para o primeiro sensor de distância
const int pinTrigger_1 = 6;
const int pinEcho_1 = 5;

//Conexões para o segundo sensor de distância
const int pinTrigger_2 = 4;
const int pinEcho_2 = 3;
#define pinServo 2
long tempo_1, tempo_2;
int dist_1, dist_2; //Distâncias
```

```
Servo servo;
void setup()
{
  pinMode(pinTrigger_1, OUTPUT); //Pino de trigger será saída digital
  pinMode(pinEcho_1, INPUT);     //Pino de echo será entrada digital
  pinMode(pinTrigger_2, OUTPUT);
  pinMode(pinEcho_2, INPUT);

  servo.attach(pinServo);        //Anexar o servo
  Serial.begin(9600);            //Inicia comunicação serial
}
void loop()
{
  for (byte i = 0; i <= 180; i++)
  { // O servo irá girar de 0°  até 180°
    servo.write(i);
    delay(30);

    //Mostra a distância do primeiro sensor
    dist_1 = calculoDistancia_1();
    Serial.print(i); // Mostra na serial qual o angulo de giro o servo está naquele momento
    Serial.print(","); // Aqui vamos separar o angulo da distancia calculada com uma vírgula
    Serial.print(dist_1); // Mostra a distancia que o sensor está calculando
    Serial.print("."); // Coloca um '.' depois da distancia para indexar no processing a distancia
```

```
  }

  for (byte i = 180; i > 0; i--)
  { // O servo irá girar de 180°  até 0°
    servo.write(i);
    delay(30);

    //Mostra a distância do segundo sensor
    dist_2 = calculoDistancia_2();
    Serial.print(i);
    Serial.print(",");
    Serial.print(dist_2);
    Serial.print(".");
  }
}

int calculoDistancia_1()
{
  digitalWrite(pinTrigger_1, LOW); // Aqui vou desligar o trigger para poder realizar os pulsos
  delayMicroseconds(2); //Desativado

  digitalWrite(pinTrigger_1, HIGH); //Pulso de trigger em nível alto
  delayMicroseconds(10); //Pulso ativo  //duração de 10 microsegundos
  digitalWrite(pinTrigger_1, LOW); //Pulso de trigge em nível baixo
  tempo_1 = pulseIn(pinEcho_1, HIGH); //Escuta a porta 10(Echo), tempo de echo: 10 microssegundos e 3min
  dist_1 = tempo_1 * 0.034 / 2; // Por fim, esta é a fórmula que utilizamos para converter o tempo na distancia do objeto até o sensor
```

```
  return dist_1;
}

int calculoDistancia_2()
{
  digitalWrite(pinTrigger_2, LOW); // Aqui vou desligar o trigger para poder realizar os pulsos
  delayMicroseconds(2); //Desativado

  digitalWrite(pinTrigger_2, HIGH); //Pulso de trigger em nível alto
  delayMicroseconds(10);  //Pulso ativo  //duração de 10 microsegundos
  digitalWrite(pinTrigger_2, LOW); //Pulso de trigge em nível baixo
  tempo_2 = pulseIn(pinEcho_2, HIGH); //Escuta a porta 10(Echo), tempo de echo: 10 microssegundos e 3min
  dist_2 = tempo_2 * 0.034 / 2; // Por fim, esta é a fórmula que utilizamos para converter o tempo na distancia do objeto até o sensor

  return dist_2;
}
```

Código na IDE Processing:

```
import processing.serial.*;      // Biblioteca responsável por gerenciar a comunicação serial entre a IDE Arduino e a IDE Processing

import java.awt.event.KeyEvent;  // É uma classe JAVA de um evento que é gerado por um campo de texto quando ele é digitado

import java.io.IOException;      // É uma classe JAVA que sinaliza quando houver alguma exceção de entrada/saída


//Define as variáveis globais

Serial myPort;                   // Objeto da porta serial


String angle="";

String distance="";


String data="";

String noObject;

String val;


float pixsDistance_1, pixsDistance_2;

int iAngle, iDistance_1, iDistance_2;


int index1=0;

int index2=0;


PFont orcFont;
```

```
void setup()
{
 // Criação da janela
 frame.setTitle("Radar Ultrassônico de 360º"); // Define o título que irá aparecer na parte superior da janela
 size (750, 750);
 smooth();
 myPort = new Serial(this, "COM5", 9600);     //Inicia a comunicação serial na COM4 com uma taxa de transmissão de 9600bps
 myPort.bufferUntil('.');
}

void draw()
{
  fill(98,245,31);
  noStroke();
  fill(0,4);
  rect(0, 0, width, height);
  fill(98,245,31);

  //Chama as funções para desenhar o radar
  drawRadar();
  drawLine();
  drawObject();
  drawText();
}
```

```
void serialEvent(Serial myPort) // Começa a ler dados da porta serial
{
  // O ponto serve para separar duas informações
  // Lê os dados da porta serial até o caractere '.' e coloca na variável String "data"
  data = myPort.readStringUntil('.');         // Ler a string até achar o ponto
  data = data.substring(0, data.length()-1);  // Transforma valores recebidos //40.130

  // Encontra o caractere ',' e o coloca na variável "index1"
  index1 = data.indexOf(","); // Achar a posição da virgula //2
  // Lê os dados da posição "0" para a posição da variável index1, ou seja, é o valor do ângulo que a placa Arduino enviou para a porta serial
  angle = data.substring(0, index1); //40

  distance = data.substring(index1+1, data.length()); //130

  // Converte as variáveis String em Inteiro
  iAngle = int(angle); // Mudar para o tipo int
  iDistance_1 = int(distance);
  iDistance_2 = int(distance);
}

void drawRadar()
{ // Draws the radar grid.

pushMatrix();
```

```
  translate(width/2,height/2); // Deslocando o centro
  noFill();                    // Elimina o fundo do semi circulo
  strokeWeight(2);             // Espessura da linha do circulo
  stroke(168, 171, 174);           // Mostrar o contorno
  //Desenha as linhas da elipse
  ellipse(0, 0,(width-width*0.0625), (width-width*0.0625));
  ellipse(0, 0,(width-width*0.27), (width-width*0.27));
  ellipse(0, 0,(width-width*0.479), (width-width*0.479));
  ellipse(0, 0,(width-width*0.687), (width-width*0.687));
  //Desenha as linhas do arco
  line(-width/2,0,width/2,0);
  line((-width/2)*cos(radians(210)),(-width/2)*sin(radians(210)),(-
width/2)*cos(radians(30)),(-width/2)*sin(radians(30)));
  line((-width/2)*cos(radians(240)),(-width/2)*sin(radians(240)),(-
width/2)*cos(radians(60)),(-width/2)*sin(radians(60)));
  line((-width/2)*cos(radians(270)),(-width/2)*sin(radians(270)),(-
width/2)*cos(radians(90)),(-width/2)*sin(radians(90)));
  line((-width/2)*cos(radians(300)),(-width/2)*sin(radians(300)),(-
width/2)*cos(radians(120)),(-width/2)*sin(radians(120)));
  line((-width/2)*cos(radians(330)),(-width/2)*sin(radians(330)),(-
width/2)*cos(radians(150)),(-width/2)*sin(radians(150)));

popMatrix();
}

void drawObject()
{ //Desenha as linhas do objeto
pushMatrix();

  translate(width/2,height/2);
```

```
  strokeWeight(9);
  stroke(255, 10, 10);

  if (iDistance_1 == 0) {iDistance_1 = 60; }    //Pulso testado de 60
cm
  if (iDistance_2 == 0) {iDistance_2 = 60; }    //Pulso testado de 60
cm

  pixsDistance_1 = iDistance_1*((height-height*0.1666)*0.025); //
Converte a distância do sensor de cm para pixels
  pixsDistance_2 = iDistance_2*((height-height*0.1666)*0.025); //
Converte a distância do sensor de cm para pixels

  if(iDistance_1 < 40)
  {  // Limitando o alcance para 40 cms
     // Desenha o objeto de acordo com o ângulo e a distância
     line(pixsDistance_1*cos(radians(iAngle)),-
pixsDistance_1*sin(radians(iAngle)),(width-
width*0.12)*cos(radians(iAngle)),-(width-
width*0.12)*sin(radians(iAngle))); //0.505
  }

  if(iDistance_2 < 40)
  { // Limitando o alcance para 40 cms
    // Desenha o objeto de acordo com o ângulo e a distância
    line(pixsDistance_2*cos(radians(iAngle+180)),-
pixsDistance_2*sin(radians(iAngle+180)),(width-
width*0.12)*cos(radians(iAngle+180)),-(width-
width*0.12)*sin(radians(iAngle+180)));
  }
```

```
popMatrix();


}


void drawLine()
{ //Desenha as linhas transversais verdes
pushMatrix();


  strokeWeight(9);
  stroke(143, 199, 62);
  translate(width/2,height/2);
  line(0,0,(height-height*0.12)*cos(radians(iAngle)),-(height-
height*0.12)*sin(radians(iAngle)));          // Desenha a linha de
acordo com o ângulo
  line(0,0,(height-height*0.12)*cos(radians(iAngle+180)),-(height-
height*0.12)*sin(radians(iAngle+180))); // Desenha a linha de acordo
com o ângulo


popMatrix();
}



void drawText()
{ // Desenha os textos


pushMatrix();


  fill(0,0,0);
  noStroke();
```

```
  fill(231, 231, 232);
  // Escrita das distâncias
  textSize(int(20*width/1000));
  text("10cm",width-width*0.410,height- height*0.51);
  text("20cm",width-width*0.300,height- height*0.51);
  text("30cm",width-width*0.195,height- height*0.51);
  text("40cm",width-width*0.100,height- height*0.51);

  textSize(int(20*width/1000));
  if(iDistance_1 < 40)
  {
    text("Distance 1: ", width-width*0.30,int (30*width/1000));
    text("    " + iDistance_1 +" cm", width-width*0.18, int
(30*width/1000));
    text("Angle 1: " + iAngle +" ° ", width-width*0.23,
int(60*width/1000));
  }

  if(iDistance_2 < 40)
  {
    text("Distance 2: ", int(10*width/1000), height- height*0.03);
    text("    " + iDistance_2 +" cm", int(120*width/1000), height-
height*0.03);
    text("Angle 2: " + iAngle +" ° ", int(10*width/1000), height-
height*0.06);
  }

  textSize(int (width*25/1000));
  fill(231, 231, 232);
```

```
  // Metade superior
  translate(width/2+0.95*width/2*cos(radians(-1)),height/2-
0.95*width/2*sin(radians(-1)));
  rotate(radians(90));
  text("0° ",0,0);
  resetMatrix();
  translate(width/2 +0.95*width/2*cos(radians(28.9)), height/2-
0.95*width/2*sin(radians(28.9)));
  rotate(-radians(-60));
  text("30° ",0,0);
  resetMatrix();
  translate(width/2+0.95*width/2*cos(radians(59.9)), height/2 -
0.95*width/2*sin(radians(59.9)));
  rotate(-radians(-30));
  text("60° ",0,0);
  resetMatrix();
  translate(width/2 +0.95*width/2*cos(radians(89.9)), height/2-
0.95*width/2*sin(radians(89.9)));
  rotate(radians(0));
  text("90° ",0,0);
  resetMatrix();
  translate(width/2+0.95*width/2*cos(radians(119.9)),height/2-
0.95*width/2*sin(radians(119.9)));
  rotate(radians(-30));
  text("120° ",0,0);
  resetMatrix();
  translate(width/2+0.95*width/2*cos(radians(149.9)),height/2-
0.95*width/2*sin(radians(149.9)));
  rotate(radians(-60));
```

```
  text("150° ",0,0);
  resetMatrix();


  //lower half
  translate(width/2+0.95*width/2*cos(radians(179.7)),height/2-
0.95*width/2*sin(radians(179.7)));
  rotate(radians(-90));
  text("180° ",0,0);
  resetMatrix();
  translate(width/2 +0.95*width/2*cos(radians(209.8)), height/2-
0.95*width/2*sin(radians(209.8)));
  rotate(-radians(120));
  text("210° ",0,0);
  resetMatrix();
  translate(width/2+0.95*width/2*cos(radians(239.8)), height/2 -
0.95*width/2*sin(radians(239.8)));
  rotate(-radians(150));
  text("240° ",0,0);
  resetMatrix();
  translate(width/2 +0.95*width/2*cos(radians(269.8)), height/2-
0.95*width/2*sin(radians(269.8)));
  rotate(radians(180));
  text("270° ",0,0);
  resetMatrix();
  translate(width/2+0.95*width/2*cos(radians(299.8)),height/2-
0.95*width/2*sin(radians(299.8)));
  rotate(radians(150));
  text("300° ",0,0);
  resetMatrix();
```

```
  translate(width/2+0.95*width/2*cos(radians(329.8)),height/2-
0.95*width/2*sin(radians(329.8)));
  rotate(radians(120));
  text("330° ",0,0);
  resetMatrix();


popMatrix();


}
```