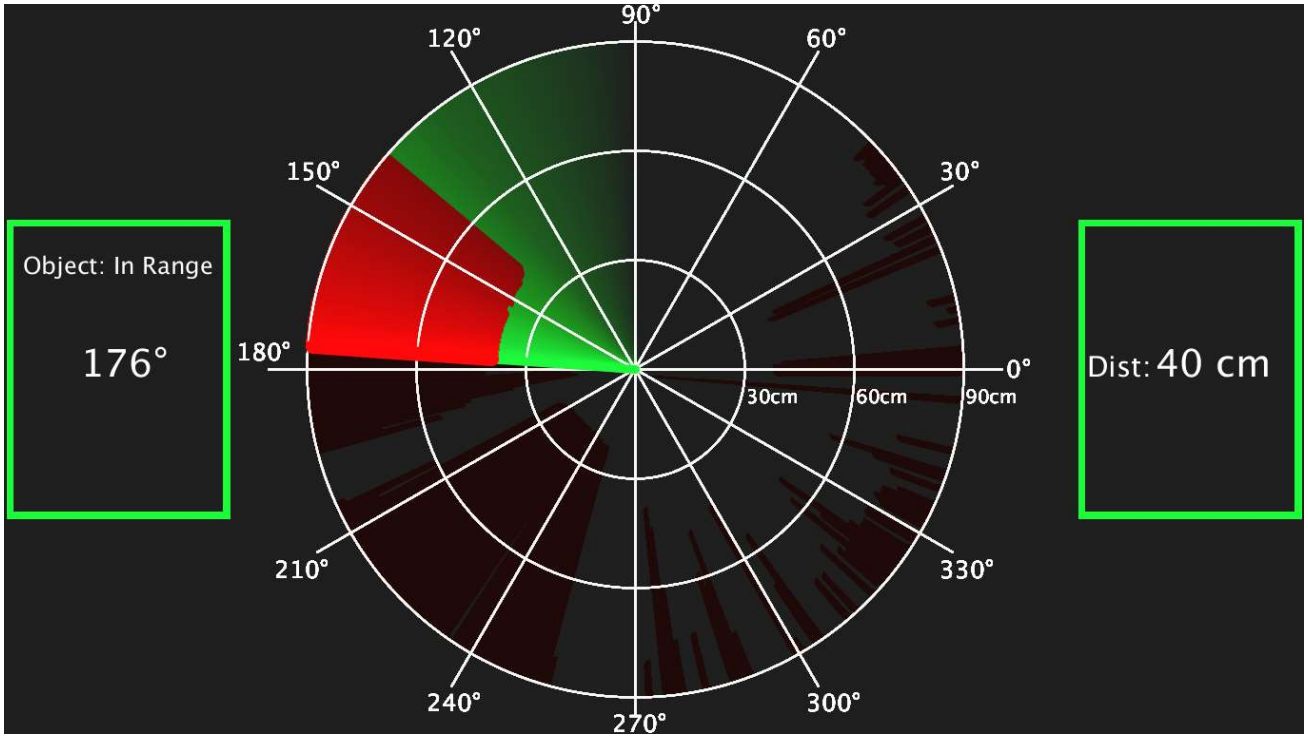
SONAR COM ARDUINO

USANDO IDE PROCESSING E RADIOFREQUÊNCIA



Sumário

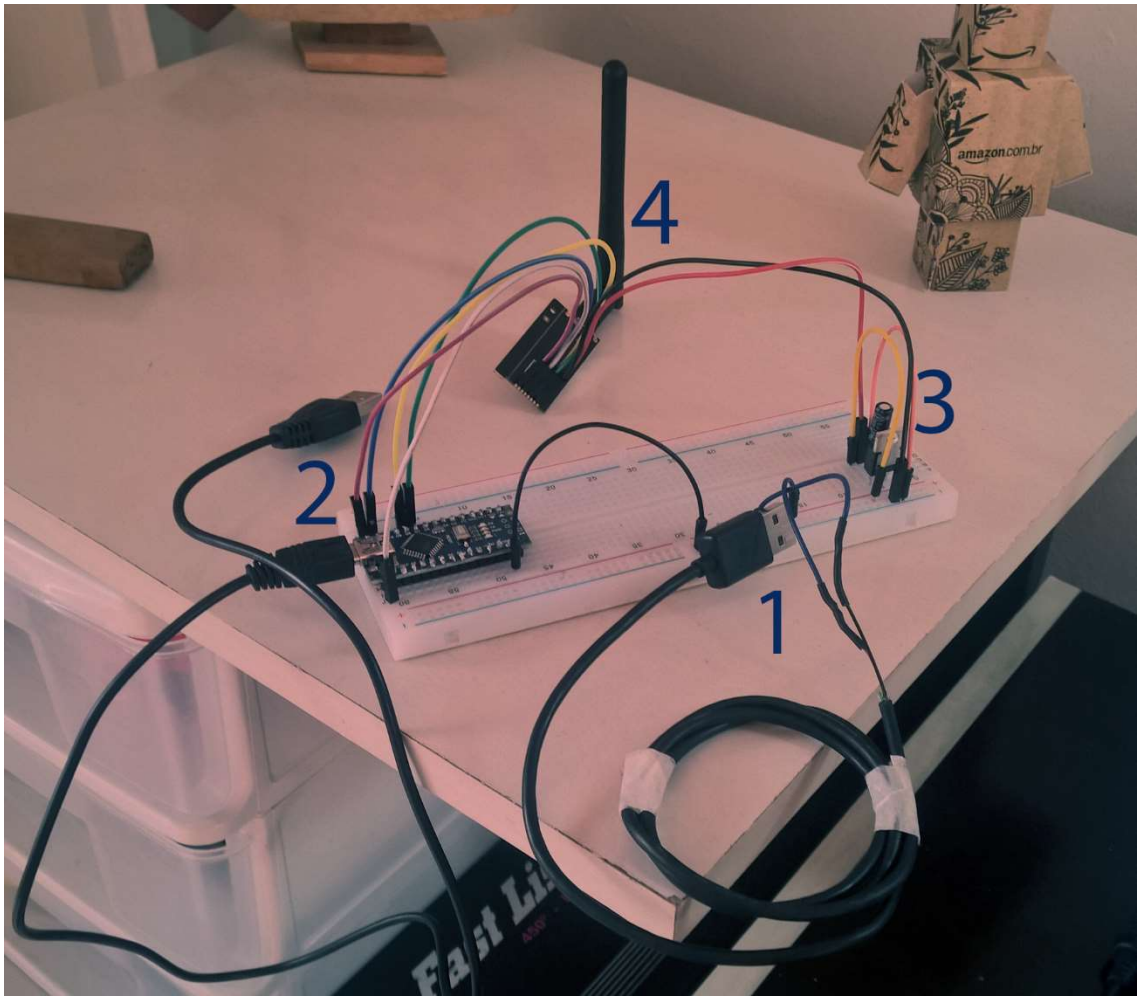
Introdução

O projeto consiste em girar o sensor ultrassônico usando um servo no ângulo de 0 a 180 graus então trocar para o outro servo e girar o outro sensor ultrassônico de 180 a 360 graus totalizando uma volta completa. O ultrassônico tem um alcance de 2 cm a 400 cm mais no processing foi programado para só exibir no "mapa" de 2 cm à 90 cm o motivo dessa escolha foi para não ficar em uma escala muito grande priorizando assim a riqueza de detalhes.

Após o arduino (TX ou seja o transmissor) mover o servo e calcular a distância pelo ultrassônico então chega na melhor parte. A transmissão sem fio! Usando NRF24L01 com antena externa (importante lembrar que este módulo não segue o protocolo IEEE 801.11 então não vai se comunicar com WIFI do roteador ou do celular apesar de trabalhar na mesma frequência). O TX envia a informação (ângulo e distância) o RX (receptor) recebe e envia para o computador através da porta serial. Então os dados são interpretados pelo IDE processing que move a barra verde e desenha a barra vermelha de acordo com o ângulo e a distância recebida. Link para o vídeo mostrando o projeto: https://www.youtube.com/watch?v=B4Q-fz4psLs

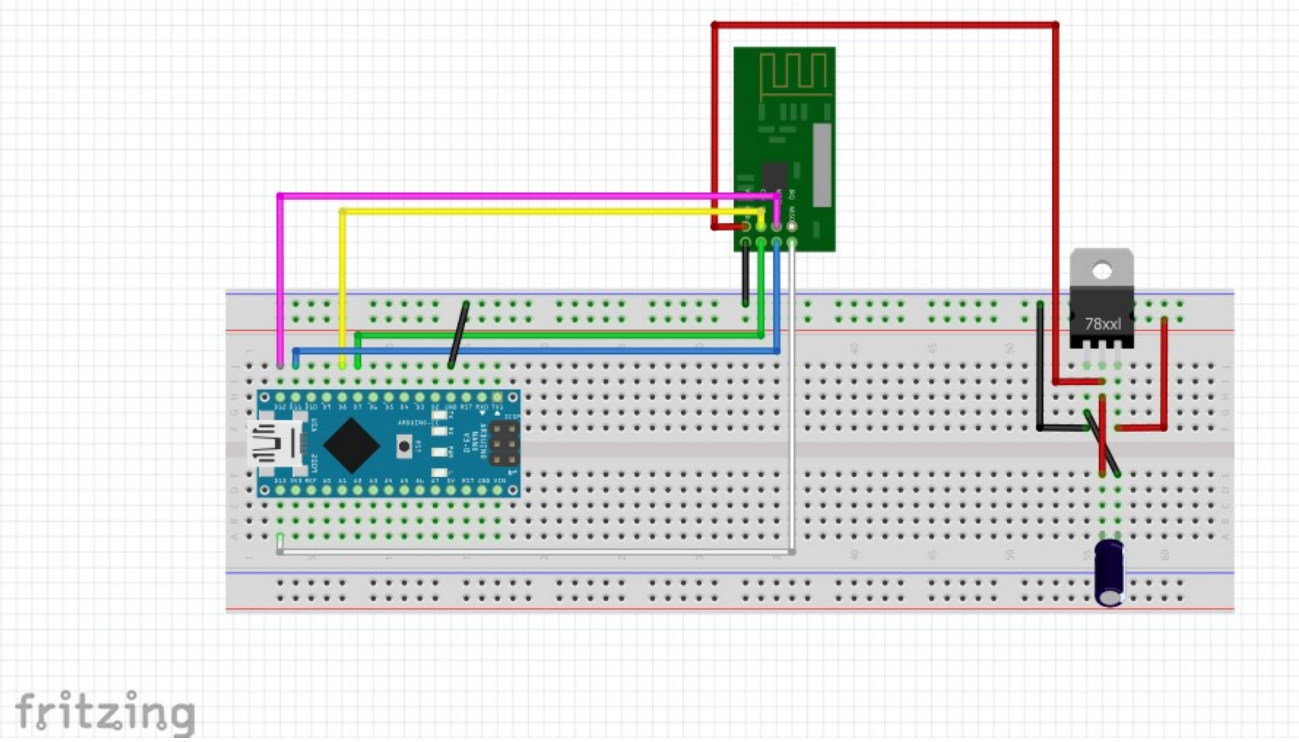
Montagem

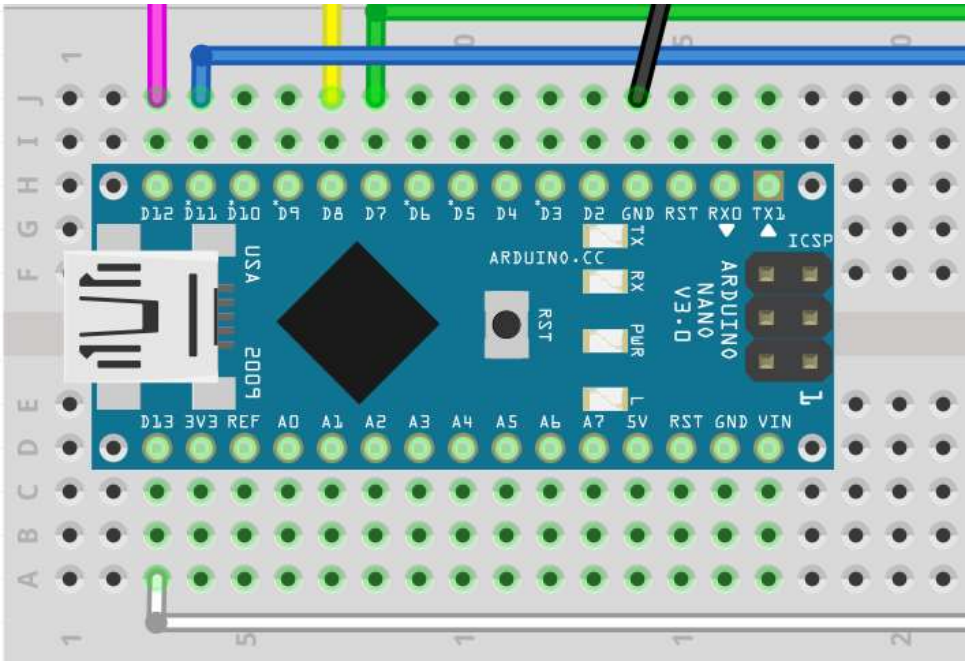
Imagem do RX (receptor):



1. Alimentação externa para antena usando a porta USB do laptop.
2. Arduino do tipo nano e sua conexão de alimentação e dados por USB.
3. Regulador linear de tensão. De 5V para 3V3 (3,3V, não pode ligar o módulo no 5V! se ligar queima) este mesmo regulador também está presente no TX.
4. Módulo NRF24L01 com antena externa.

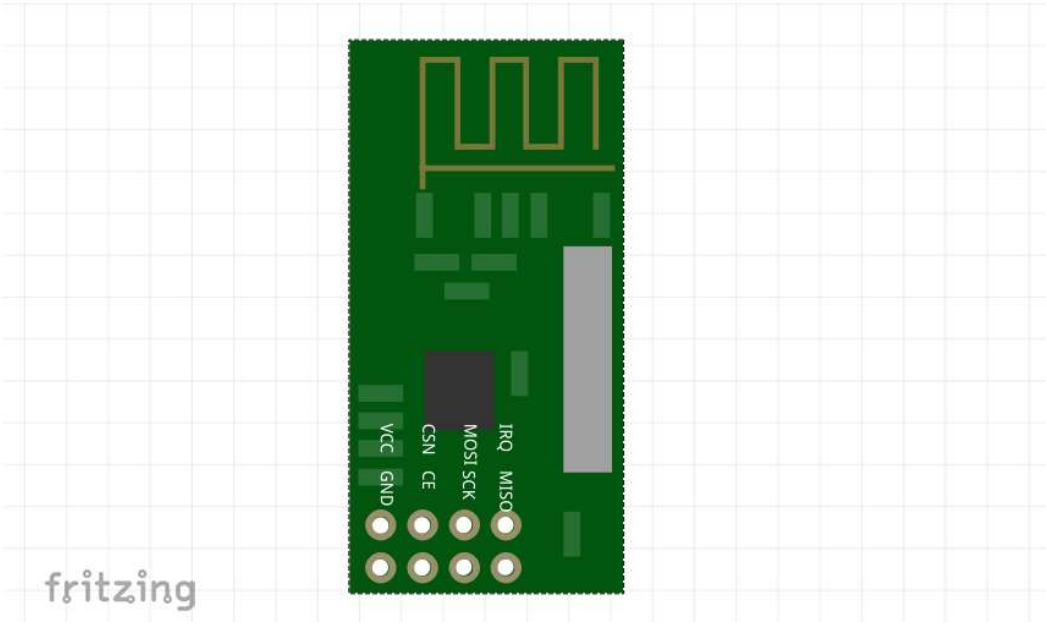
Esquema de ligação usando o fritzing:





Observação:

- Apesar de esta escrito na imagem do regulador de tensão 78xxl na verdade é um LD1117V.
- A foto é uma versão mais antiga do NRF24L01 mas o esquema de ligação é o mesmo. Ficar atento quanto a ligação no módulo (nome do pino).

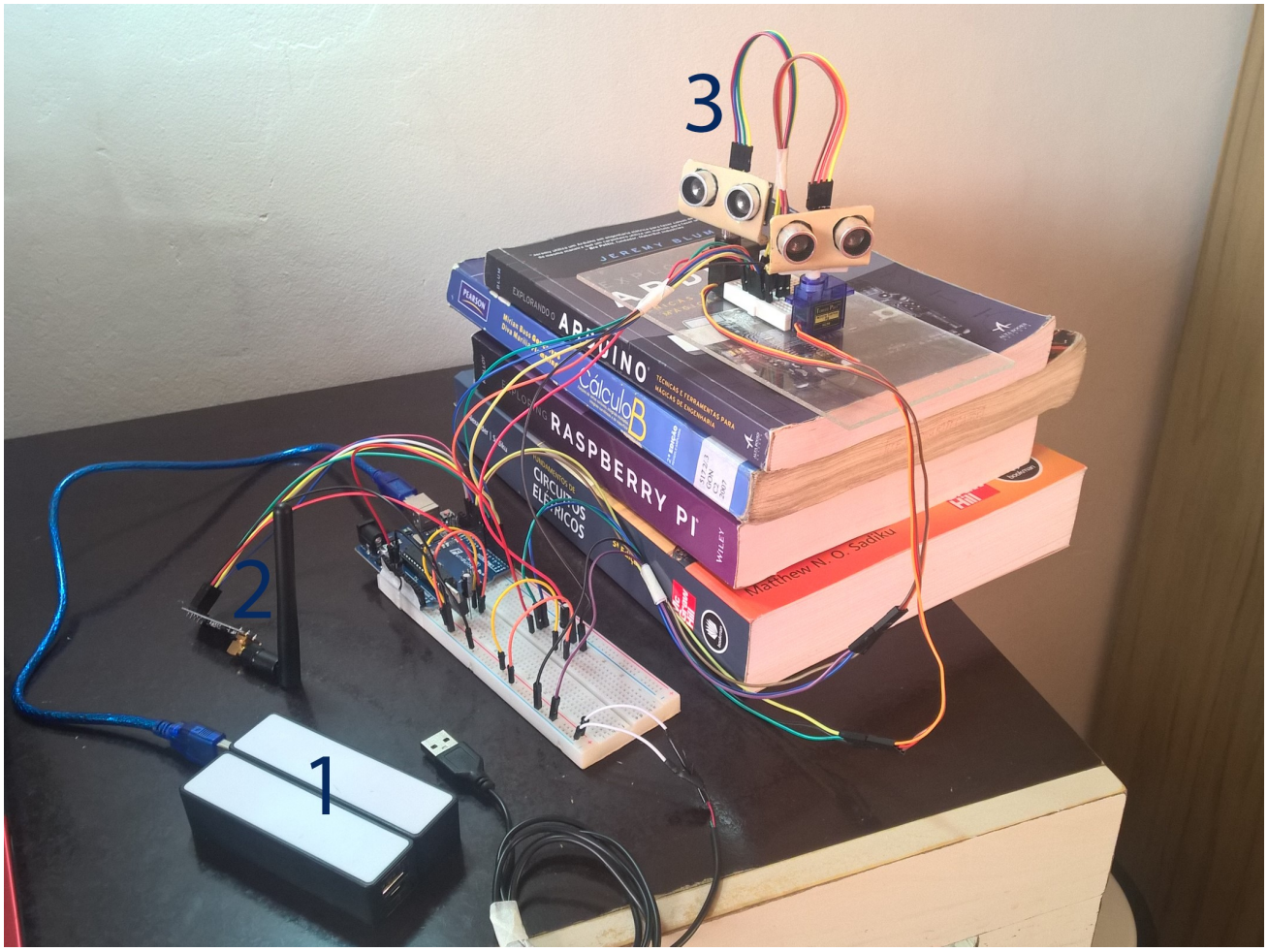


Sobre o módulo NRF24L01.

- VCC é para ligar 3V3. A corrente pode chegar a 115mA no modo transmissão então optei por fazer uma alimentação externa.
- GND é o negativo.
- CE e CSN são pinos de controle. Pode ligar em qualquer pino digital eu escolhi os pinos 7 e 8 para CE e CSN respectivamente.
- SCK, MOSI e MISO são comunicação SPI então deve ser ligado em pinos específicos (11, 12 e 13 respectivamente).
- IRQ não será usado.
- Link para o datasheet do produto em inglês: https://www.nordicsemi.com/chi/content/download/2730/34105/file/nRF24L01_Product_Specification_v2_0.pdf

Agora para o TX

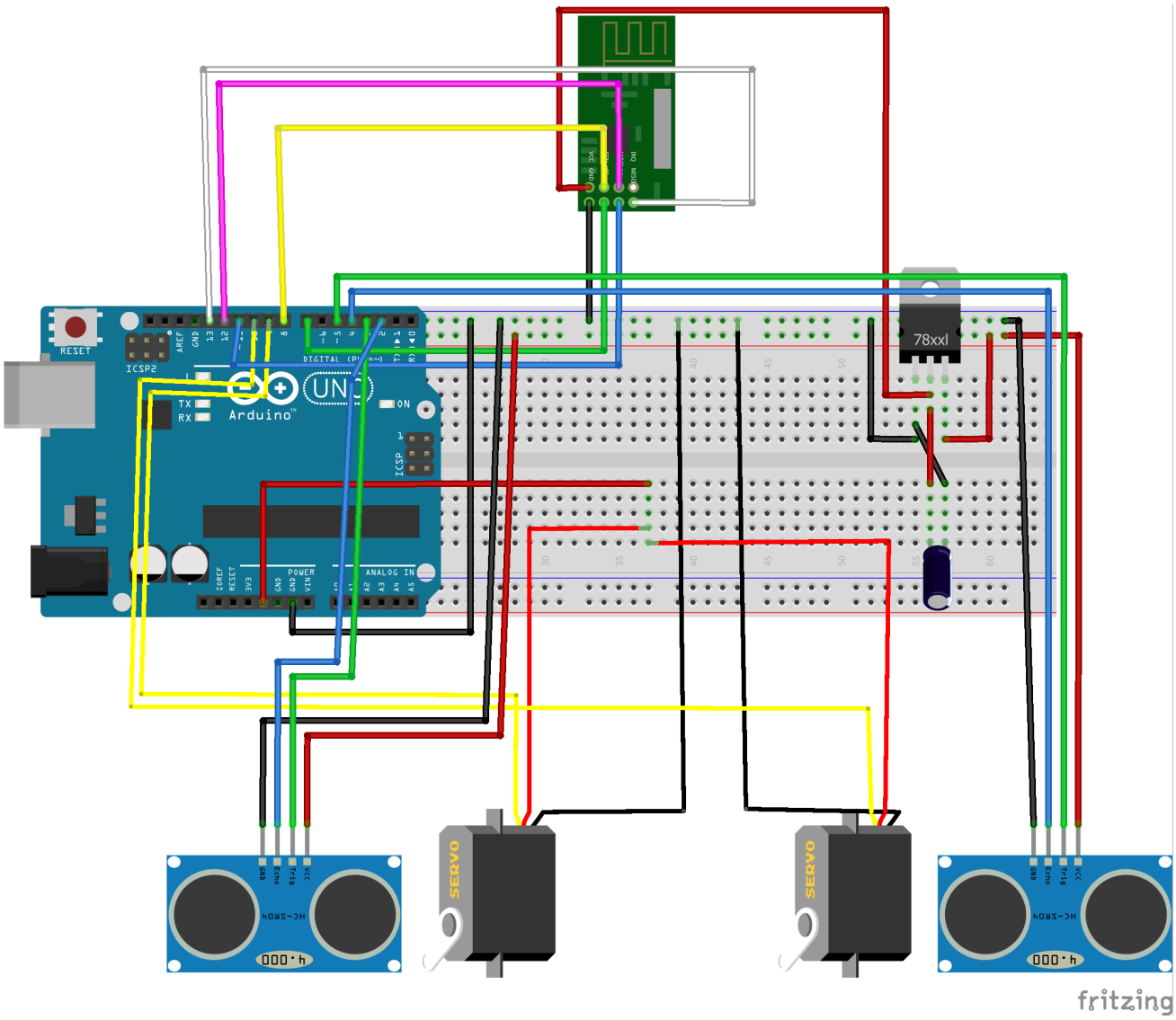
Imagem do TX (transmissor):



1. Dois power bank.
2. Mesmo módulo NRF24L01 do RX.
3. Dois servos e Dois sensores ultrassônicos.

A novidade no TX é o power bank, a troca do arduino nano por um uno, sevos e os ultrassônicos. A montagem do NRF24L01 é exatamente a mesma do RX. A pilha de livros é para os ultrassônicos não detectarem o próprio cabo.

Esquema de ligação usando o fritzing:



Não esqueça que na trilha de energia (positiva e negativa) está ligada em um dos power bank.

datasheet:

Servo https://cdn.instructables.com/ORIG/FA2/O1SS/J7ARLNBW/FA2O1SSJ7ARLNBW.pdf

ultrassônico https://www.mpja.com/download/hc-sr04_ultrasonic_module_user_guidejohn.pdf

Eu recomendo a leitura dos datasheet para um melhor entendimento.

Código

Download da biblioteca nRF24L01.h https://drive.google.com/file/d/0BxP9TuJ18bsFenBjbDRqWkIwZ1E/view

Para o RX https://drive.google.com/open?id=0BxP9TuJ18bsFMnJpbjdmc0JFVG8

```
1  #include <SPI.h>                          //bibliotecas
2  #include "nRF24L01.h"
3  #include "RF24.h"
4
5  int DICE[2];                             //vetor para armazenar os dados recebidos
6
7  RF24 radio(7,8);                          // CE; CSN
8
9  const byte address[6] = "00001";          //canal de comunicação. Tem que ser igual ao do TX
10
11 void setup(void){
12   Serial.begin(9600);                     //comunicação serial
13   radio.begin();                          //liga o modulo RX
14   radio.openReadingPipe(0, address);    //define como leitura o canal
15   radio.setPALevel(RF24_PA_HIGH);       //potência da antena
16   radio.startListening();                 //fica pronto para receber dados
17 }
18
19 void loop(void){
20   if ( radio.available() ){               //verifica se tem dados
21     //teste();                           //para teste do módulo
22
23     radio.read( DICE, sizeof(DICE) );      //faz leitura dos dados recebidos e armazena em DICE
24     Serial.print(DICE[0]);                //imprime na porta serial o ângulo
25     Serial.print(",");
26     Serial.print(DICE[1]);                //imprime na porta serial a distancia
27     Serial.print(".");                    //tirar ln
28
29   }
30 }
31
32 int teste(){
33   radio.read( DICE, sizeof(DICE) );
34
35   Serial.print(DICE[0]);
36   Serial.print(" ");
37   Serial.println(DICE[1]);
38 }
```

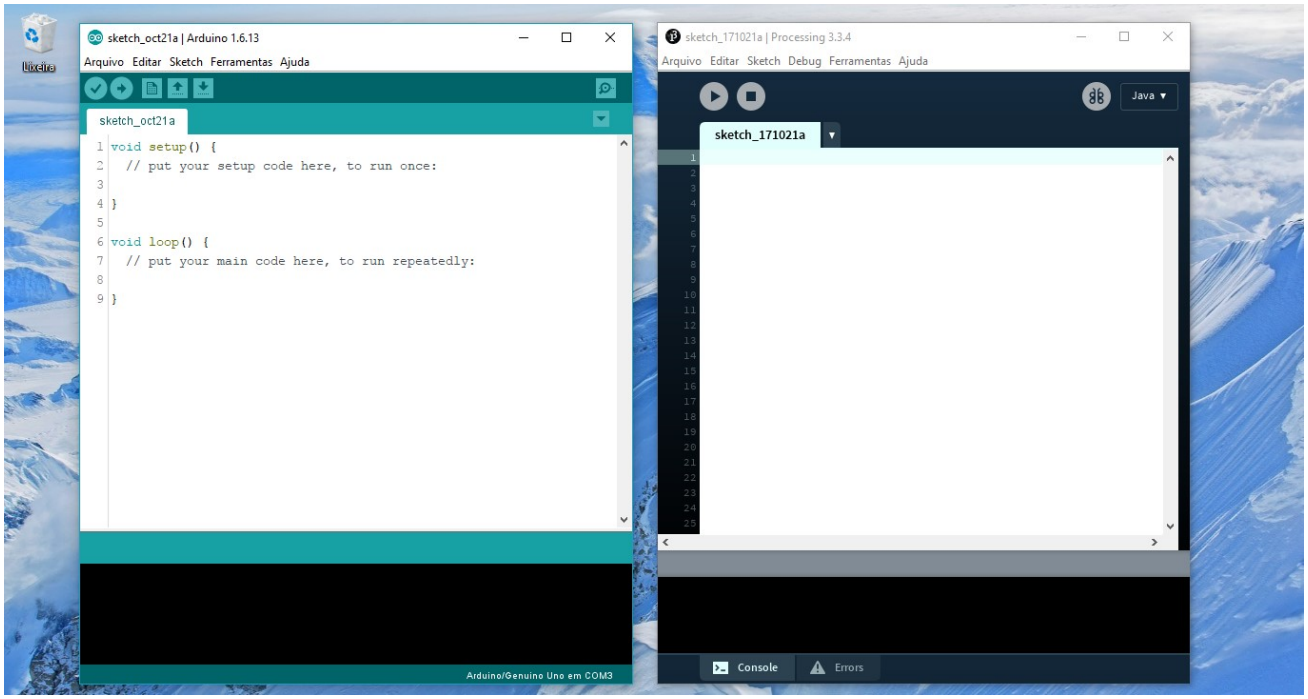
Para o TX https://drive.google.com/open?id=0BxP9TuJ18bsFZm1pYXJJbE5RLXM

```
1  #include <SPI.h>
2  #include "nRF24L01.h"
3  #include "RF24.h"
4  #include <Servo.h>
5  #define ANmax 180  //angulo maximo
6  #define ANmin 0  //angulo minimo
7  Servo Servo1;
8  Servo Servo2;
9
10 const int SERVO1 = 10; //azul
11 const int SERVO2 = 9;  //preto
12
13 const int trigPin = 3;
14 const int echoPin = 2;
15 const int trigPin2 = 5;
16 const int echoPin2 = 4;
17 long duration;
18 float distance;
19
20 int DICE[2];
21 RF24 radio(7,8);
22 const byte address[6] = "00001";
23
24 void setup(void){
25   Servo1.attach(SERVO1);
26   Servo2.attach(SERVO2);
27   pinMode(trigPin, OUTPUT);
28   pinMode(trigPin2, OUTPUT);
29   Serial.begin(9600);
30   radio.begin();
31   radio.openWritingPipe(address);                    //escreve os dados nesse canal
32   radio.setPALevel(RF24_PA_MIN);  //potência se colocar maior a antena desliga. Não sei o motivo
33   radio.stopListening();                             //para transmitir
34 }
35
36 void loop(){
37   //teste();
38   for(int a = 0; a <= 180; a++){                     //move o servo 1 e usar o ultrassônico 1
39     Servo1.write(a);
40     delay(15);
41     DICE[0] = a;
42     DICE[1] = funDIST1();
43     radio.write( DICE, sizeof(DICE) );
44   }
```

```
45
46   for(int a = 1; a <= 180; a++){
48     Servo2.write(a);
49     delay(15);
50     DICE[0] = a + 180;
51     DICE[1] = funDIST2();
52     radio.write( DICE, sizeof(DICE) );
53   }
54 }
55
56 int teste(){
57   DICE[0] = 1;
58   DICE[1] = 4;
59   radio.write( DICE, sizeof(DICE) );
60   delay(50);
61 }
62
63 float funDIST1(){                 //eu sei que existe biblioteca para isso
64   digitalWrite(trigPin, LOW);   //mas tava dando 10 cm a mais do que a distância real
65   delayMicroseconds(2);
66   digitalWrite(trigPin, HIGH);
67   delayMicroseconds(10);
68   digitalWrite(trigPin, LOW);
69   duration = pulseIn(echoPin, HIGH);
70   distance= duration*0.034/2;
71   distance = constrain(distance, 2, 400);
72   return distance;
73 }
74
75 float funDIST2(){
76   digitalWrite(trigPin2, LOW);
77   delayMicroseconds(2);
78   digitalWrite(trigPin2, HIGH);
79   delayMicroseconds(10);
80   digitalWrite(trigPin2, LOW);
81   duration = pulseIn(echoPin2, HIGH);
82   distance= duration*0.034/2;
83   distance = constrain(distance, 2, 400);
84   return distance;
85 }
```

O processing

O processing é uma linguagem gráfica, open source. Assim como quase todas as linguagens de programação essa também tem comunicação serial. Então por que usar processing? Resposta é, Fácil de programar. A IDE (Ambiente de desenvolvimento integrado) é parecida com a IDE do arduino. Aliais o arduino foi inspirado no processing



Notou alguma semelhança?

Download do processing pode ser feito nesse link: https://processing.org/download/

Para quem quiser aprender detalhes das funções basta acessar https://processing.org/reference/ e procurar a função.

Observação: Quando eu ia copilar o código (apertar o play) tinha que copilar e colocar a porta USB (que está ligada no arduino RX) quase ao mesmo tempo.

Vamos para o código:

Eu testei em uma tela (1366, 766) em fullscreen. Não sei dizer o que acontece em outras resoluções.

https://drive.google.com/open?id=0BxP9TuJ18bsFRFdlSnFJNE5OckE

```
1  import processing.serial.*;
2  import java.io.IOException;
3  import java.awt.event.KeyEvent;
4
5  Serial myPort;
6  PImage img;
7  String angle="";
8  String distance="";
9  String data="";
10 String noObject;
11 float pixsDistance;
12 int iAngle, iDistance;
13 int index1=0;
14 int index2=0;
15
16 void setup(){
17   //size(1360,760);         //Tamanho da tela
18   fullScreen();             //Tela toda
19   smooth();
20   myPort = new Serial(this, "COM4", 9600);
21   myPort.bufferUntil('.');
22   background(100, 200, 100);
23 }
24
25
26 void draw(){
27    noStroke();
28    fill(0,4);
29    rect(0, 0, width, height-height*0.00001);
30    fill(98,245,31);
31
32    drawRadar();
33    drawLine();
34    //iDistance = 67;     //para teste
35    drawObject();
36    drawText();
37 }
38
39
40 void serialEvent (Serial myPort) {
41   data = myPort.readStringUntil('.');
42   data = data.substring(0,data.length()-1);
43
44   index1 = data.indexOf(",");
45   angle= data.substring(0, index1);
46   distance= data.substring(index1+1, data.length());
```

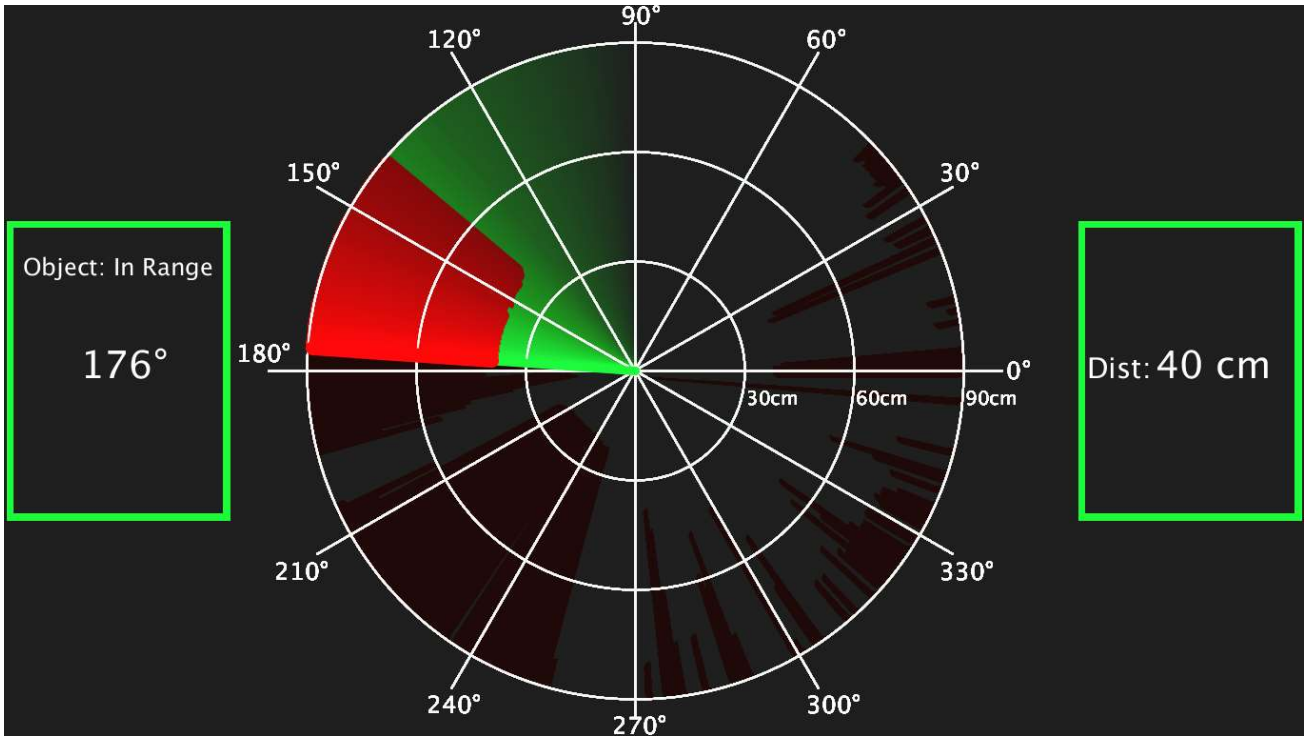
```
47
48   iAngle = int(angle);
49   iDistance = int(distance);
50 }
51
52
53 void drawRadar() {
54   pushMatrix();
55   translate(663, 384);        //centro
56   noFill();
57   strokeWeight(2);
58   stroke(250,250,250);         //Branco
59
60   // draws the arc lines
61   arc(0,0,690,690,PI,2*TWO_PI);
62   arc(0,0,460,460,PI,2*TWO_PI);
63   arc(0,0,230,230,PI,2*TWO_PI);
64
65   stroke(250,250,250);        //Branco
66
67   // draws the angle lines
68   line(-385,0,385,0);
69
70   line(385*cos(radians(30)),385*sin(radians(30)),-385*cos(radians(30)),-385*sin(radians(30)));
71   line(385*cos(radians(60)),385*sin(radians(60)),-385*cos(radians(60)),-385*sin(radians(60)));
72   line(+385*cos(radians(120)),+385*sin(radians(120)),-385*cos(radians(120)),-385*sin(radians(120)));
73   line(+385*cos(radians(150)),+385*sin(radians(150)),-385*cos(radians(150)),-385*sin(radians(150)));
74
75   line(0,-365,0,365);
76   popMatrix();
77 }
78
79 void drawLine(){
80   pushMatrix();
81   strokeWeight(9);
82   stroke(30,250,60);          //verde
83   translate(663, 384);        //centro
84   line(0,0,(height-height*0.558)*cos(radians(iAngle)),-(height-height*0.558)*sin(radians(iAngle)));
85   popMatrix();
86 }
87
88 void drawObject() {
89   pushMatrix();
90   translate(663, 384);  // moves the starting coordinats to new location
91   strokeWeight(15);
92   stroke(255,10,10); // red color
```

```
93
94  pixsDistance = iDistance*((height-height*0.558)/90);       // O max que pixsDistance pode ir é 50
95                                          //X = (height-height*0.558) / 90
96
97  if(iDistance <= 90){
98  //line(500*cos(radians(iAngle)),-500*sin(radians(iAngle)),pixsDistance*cos(radians(iAngle)),-pi
99  line(pixsDistance*cos(radians(iAngle)),-pixsDistance*sin(radians(iAngle)), (height-height*0.558
100 //line(x1,y1,x2,y2);
101 }
102 popMatrix();
103 }
104
105 void drawText() {
106  stroke(30,250,60);
107  pushMatrix();
108  fill(98,245,31);
109  rect(10,234,220,300);
110  rect(1136,234,220,300);
111  noStroke();
112  fill(31,31,31);
113  rect(10,234,220,300);
114  rect(1136,234,220,300);
115
116  if(iDistance > 90) {
117   noObject = "> 90cm";
118  }
119  else {
120   noObject = "In Range";
121  }
122
123  fill(255,255,255); //color white
124  textSize(25);
125  text("Object: " + noObject,20,284);
126  textSize(40);
127  text("" + iAngle +"°", 80,392);
128
129  textSize(30);
130  text("Dist: ", 1138,392);
131  textSize(40);
132  text("" + iDistance +" cm", 1210, 392);
133
134  textSize(20);
135  text("30cm",780,420);
136  text("60cm",895,420);
137  text("90cm",1010,420);
```

```
138
139 //fill(98,245,60); //verde
140 fill(255,255,255); //branco
141 textSize(25);
142
143 translate(663, 384);
144 text("0°",390,9);
145 resetMatrix();
146
147 translate(663, 384);
148 text("180°",-420,-10);
149 resetMatrix();
150
151 translate(663, 384);
152 text("90°",-16,-365);
153 resetMatrix();
154
155 translate(663, 384);
156 text("270°",-25,382);
157 resetMatrix();
158
159 translate(663, 384);
160 text("30°",320,-200);
161 resetMatrix();
162
163 translate(663, 384);
164 text("150°",-368,-200);
165 resetMatrix();
166
167 translate(663, 384);
168 text("210°",-380,220);
169 resetMatrix();
170
171 translate(663, 384);
172 text("330°",320,220);
173 resetMatrix();
174
175 translate(663, 384);
176 text("60°",180,-340);
177 resetMatrix();
178
179 translate(663, 384);
180 text("120°",-220,-340);
181 resetMatrix();
182
```

```
183  translate(663, 384);
184  text("240°",-220,360);
185  resetMatrix();
186
187  translate(663, 384);
188  text("300°",180,360);
189  resetMatrix();
190
191  popMatrix();
192 }
193                          //UFA!!
```

Após copilar vai exibir uma janela parecida com a da imagem abaixo.



- “Object: in Range” quer dizer tem algo dentro do raio de 90 cm.
- 176º é o ângulo em que se encontra o servo.
- “Dist: 40 cm” é que tem algum objeto a 40 cm do sensor ultrassônico

Conclusão

O sensor ultrassônico tem que emitir o som (é trabalho do trigger), Desliga e então escutar (é trabalho do echo). O tempo em que o som viaja e volta faz com que o código fique mais lendo ou mais rápido. Mudando a velocidade angular média. Uma solução boa para manter a velocidade do algoritmo constante seria substituir o ultrassônico por um sensor sharp IR. Porém teremos novos problemas.

Primeiramente, O preço. Sensores sharp custam em média 10 dólares. O alcance é outro problema, um sensor sharp de 20 dólares tem alcance de 20 a 150 cm enquanto o ultrassônico que encontra em lojas brasileiras com preço entorno de R$8 a R$13 tem alcance de 2cm a 400cm.

Última coisa que queria dizer é a montagem pode até parecer simples, mas muitas vezes tudo travava ai era aquela dificuldade em saber onde estava o erro. As vezes o erro estava no código as vezes na montagem mais na maioria dos casos eu não sabia onde estava.

No projeto inicial era de colocar esse projeto em um carrinho. Porém, estava dando muito erro ao tentar ligar os motores DC e acabei desistindo. Espero que esse trabalho inspire vocês a montarem e modificarem seus projetos. Caso alguém queira entrar em contato meu email é “wesley.cantarino@engenharia.ufjf.br”.

Inspirações:

http://howtomechatronics.com/projects/arduino-radar-project/ acesso: 21/10/2017

http://howtomechatronics.com/tutorials/arduino/arduino-wireless-communication-nrf24l01-tutorial/ acesso: 21/10/2017