ونتیلاتور ریوی COVID-19

علیرضا خشنو

—

مبانی رباتیک

—

استاد: محمد زارع

خرداد ۱۴۰۳

چکیده

ونتیلاتور غیر تهاجمی، یک ماسک تنفسی اضطراری برای مبارزه با بحران بهداشتی کووید ۱۹، در صورت عدم دسترسی به ونتیلاتور و تا زمانی که بیمار تحت بیهوشی یا لوله‌گذاری نباشد میباشد که ساخت آن کم هزینه و آسان است.

تهویه غیرتهاجمی با استفاده از ماسک‌های بینی صورت می‌گیرد که مقدار مشخصی از هوای تحت فشار را به داخل ریه‌ها فشار می‌دهد. هنگامی که بیماری باعث از کار افتادن ریه‌ها شده است، از فرآیند تنفس طبیعی پشتیبانی می‌کند و بدن را قادر می‌سازد تا با عفونت مبارزه کند و بهتر شود.

THE PROCESS

مقدمه

ونتیلاتور غیر تهاجمی، یک ماسک تنفسی اضطراری برای مبارزه با بحران بهداشتی کووید ۱۹، در صورت عدم دسترسی به ونتیلاتور و تا زمانی که بیمار تحت بیهوشی یا لوله‌گذاری نباشد میباشد که ساخت آن کم هزینه و آسان است.

تهویه غیرتهاجمی با استفاده از ماسک‌های بینی صورت می‌گیرد که مقدار مشخصی از هوای تحت فشار را به داخل ریه‌ها فشار می‌دهد. هنگامی که بیماری باعث از کار افتادن ریه‌ها شده است، از فرآیند تنفس طبیعی پشتیبانی می‌کند و بدن را قادر می‌سازد تا با عفونت مبارزه کند و بهتر شود.

تمامی تست ها با موفقیت و با تست عملکردی بیش از ۲۰ روز بدون مشکل و وقفه انجام شده اند.

بر اساس تحقیقات انجام شده، از این فناوری در این پروژه به گونه ای استفاده شده که یک ماسک snorkel Decathlon را به یک دستگاه تنفس مصنوعی اضطراری برای بیماران مبتلا به COVID-19 تبدیل می کند تا به کاهش کمبود دستگاه تنفس مصنوعی در زمان اوج همه گیری ویروس کرونا کمک کند.

این ماسک‌ها به دلیل هزینه کم و سازگاری آسان در مکان‌های مختلف دنیا مورد استفاده آزمایشی قرار گرفته‌اند.

اخطار: این نمونه اولیه هنوز توسط هیچ نهاد رسمی تایید نشده است و من مسئولیت استفاده از آن را رد می کنم.

۱. مواد و قطعات مورد نیاز:

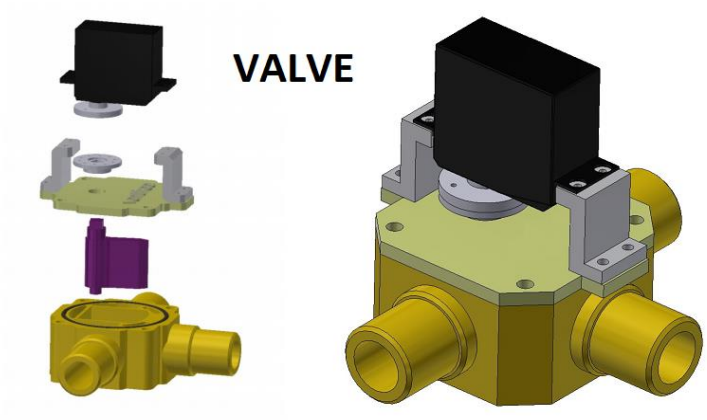
- میکروکنترلر آردوینو UNO
- شیلد آردوینو ٤ رله
- موتور سروی دیجیتال MG995
- دو ماژول **LM2596S**
- پتانسیومتر چند چرخشی خطی k۱۰
- LED ٥ میلی متری : قرمز
- LED ٥ میلی متری : سبز
- Alphanumeric LCD ، ۲۰ × ٤
- دکمه سوئیچ ۲۲۰ ولت
- snorkel تمام صورت
- شیر برقی دو طرفه

۲. نرم افزار ها و ابزار مورد استفاده:

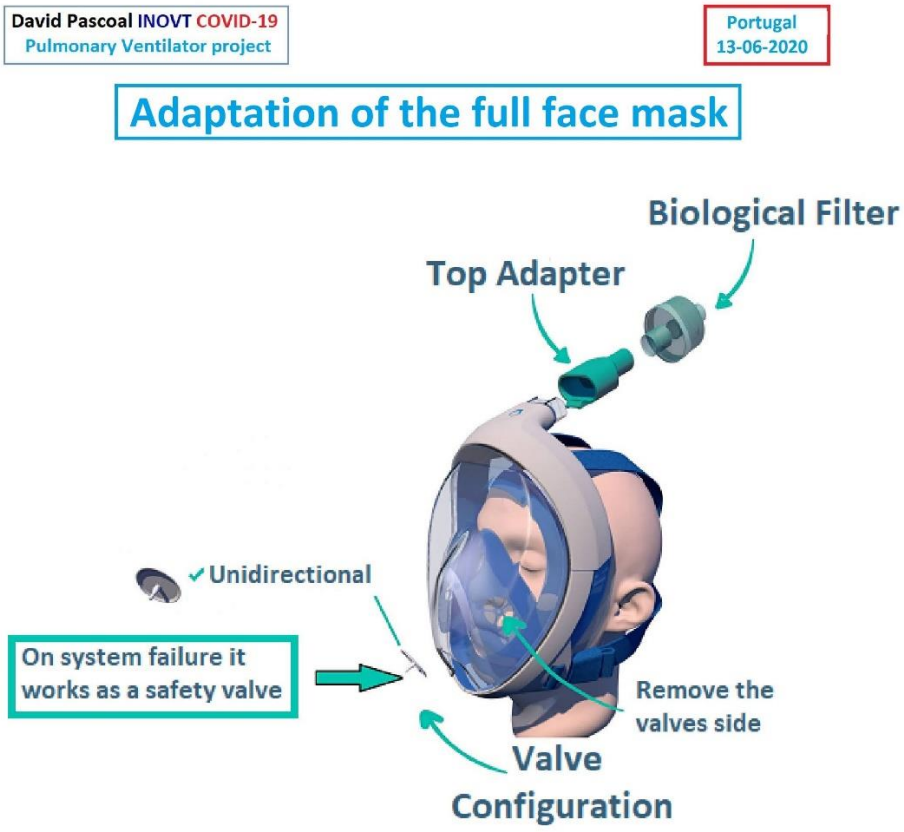
- محیط توسعه آردوینو
- PCB Elegance
- Autodesk Fusion 360
- چاپگر سه‌بعدی A8
- لایت باکس UV، 300-001

۳. مراحل اجرا:

- ابتدا، قطعات و محفظه های سفارشی از جمله دریچه تنفسی باید روی چاپگر سه بعدی چاپ شوند. (فایل سه بعدی CAD برای چاپ در ریپازیتوری موجود است.)



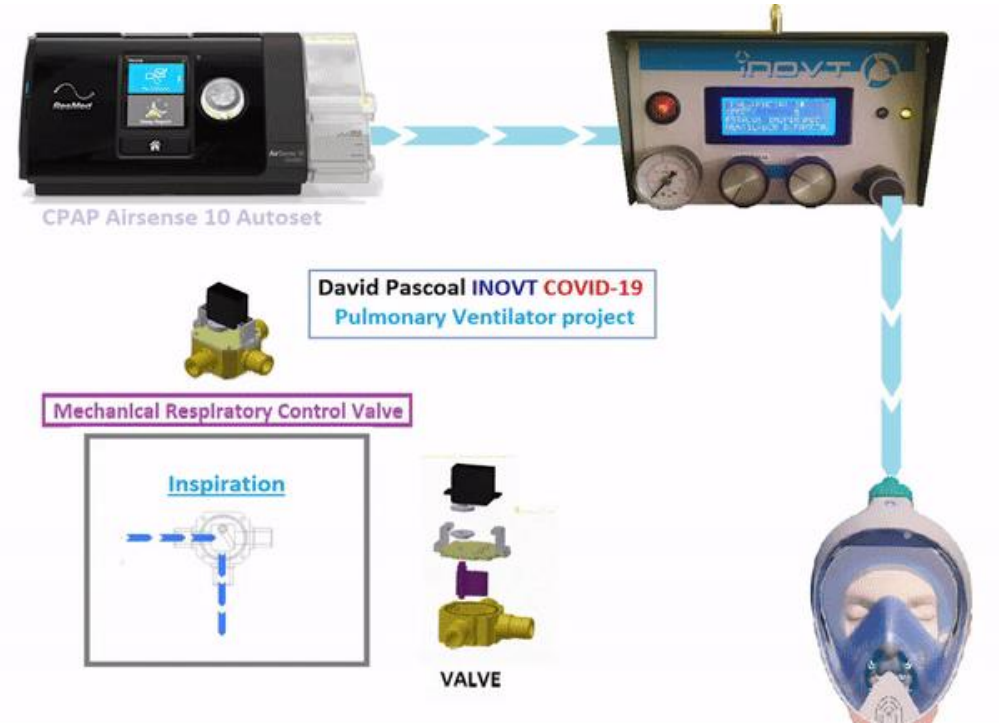
- سپس طبق دستورالعمل گام به گام، ماسک غواصی را تبدیل به ماسک تنفسی تحت فشار کرده.



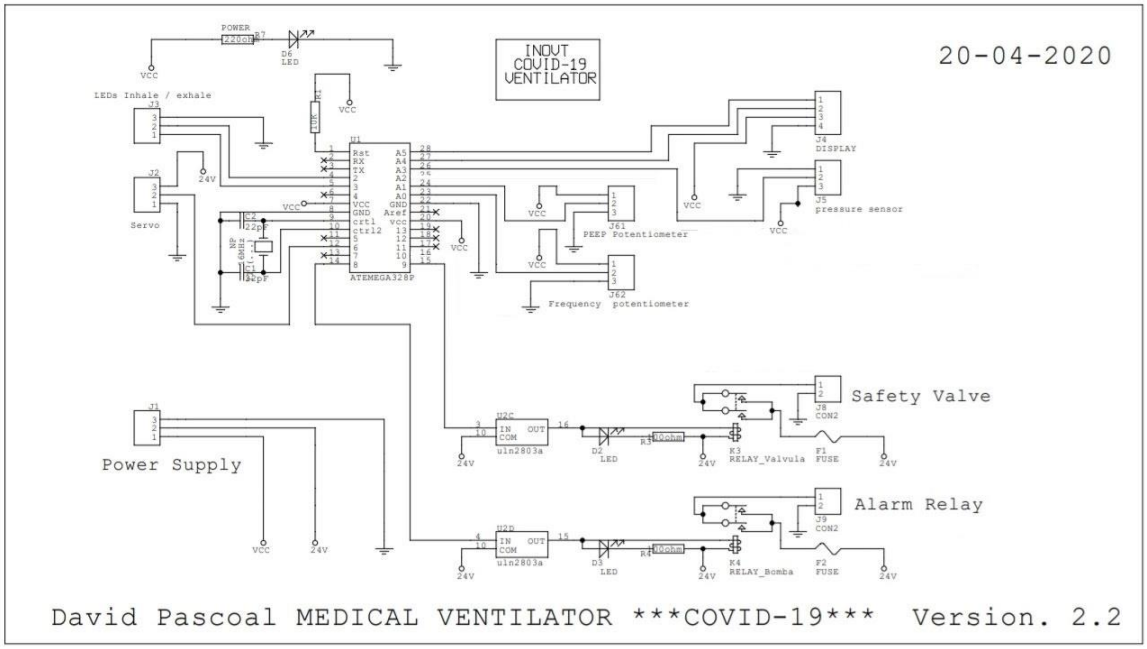
- نمایشگری شبیه به رادار روی کامپیوتر نمایش داده می‌شود که اشیاء را رصد می‌کند.

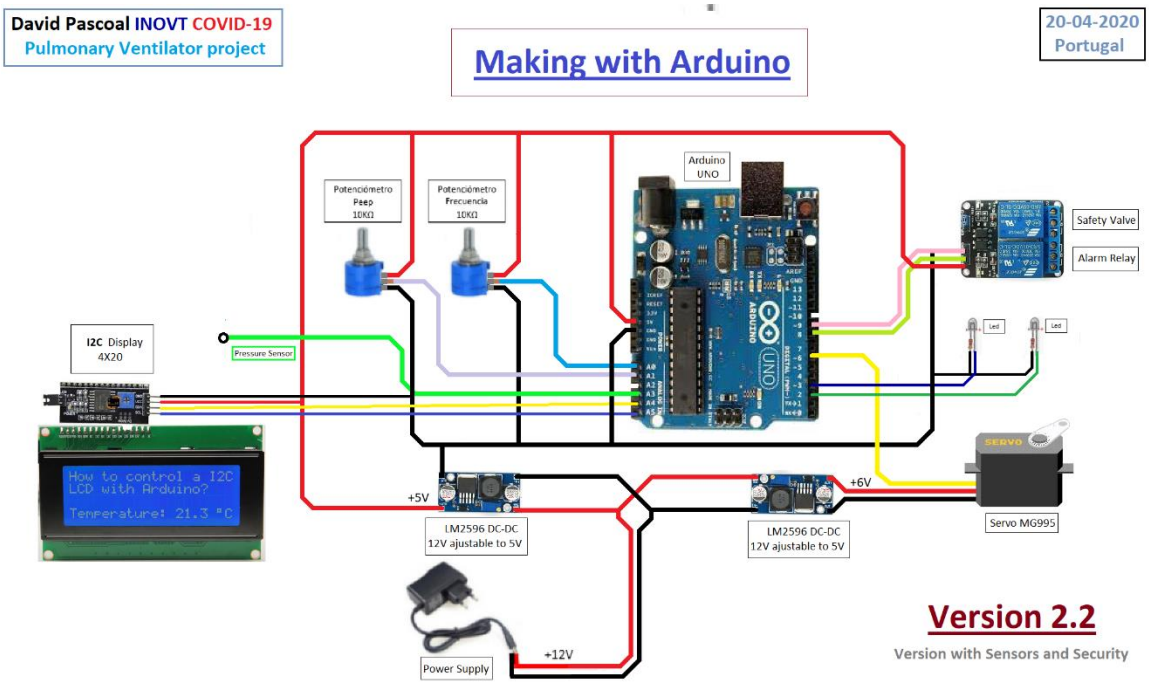
٤. شماتیک ها:

نمودار سیستم: نمودار نسخه خودمختار، بدون نیازی به ستون هوای بیمارستان.



نمودار نسخه پیشرفته INOVT COVID-19 Arduino Diagram V2.2 :





۵. کد آردوینو:

```
/* Program pulmonary ventilator INOVT COVID-19  Versio 2.2
 * Autor: David Pascoal
 * The equipment has been tested and proven,security with pressure sensor,
 * Alarm output to turn on Buzer or beetle.
*/
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <Servo.h>

LiquidCrystal_I2C lcd(0x27,20,4);

#define PIN_SERVO 6
#define PIN_FREQ A1
#define PIN_PEEP A0
#define PIN_LED 2
#define PIN_LED2 3
#define PIN_Alarm_Relay 8
#define PIN_Safety_Valve 9
#define SENSOR_FLOW A3
#define EXPIRANDO  0
#define INSPIRANDO  1
#define MIN_ANGLE 92
#define MAX_ANGLE 139


int frecuencia = 0;
int valInspira = 0;
int valExpira = 0;
int valPeep = 0;
int aux;
int x = 500;
int y = 1000;

unsigned long time;
Servo myServo;
int frecuencia_ant;
int valPeep_ant;

int estado;

void logo()
{


byte a3[8]= { B00011,
              B00111,
```

```
              B01100,
              B11000,
              B10000,
              B00000,
              B00000
};

byte a2[8]= {B00000,
             B00000,
             B00000,
             B00000,
             B10001,
             B11111,
             B01110
};

byte a1[8]={B11000,
            B01100,
            B00110,
            B00011,
            B00001,
            B00000,
            B00000
};

byte a4[9]={ B00000,
             B00000,
             B00000,
             B00001,
             B00011,
             B00110,
             B01100,
             B11000

};

byte a5[9]={ B00000,
             B01110,
             B11111,
             B10001,
             B00000,
             B00000,
             B00000,
             B01110,
             B00000,
};

byte a6[8]={  B00000,
              B00000,
              B00000,
              B10000,
              B11000,
```

```
              B01100,
              B00110,
              B00011
};

byte a7[10]={ B00000,
              B01110,
              B11111,
              B10001,
              B00000,
              B00000,
              B00000,
              B00000,

};

byte a8[8]={B00100,
            B01110,
            B00100,
            B00000,
            B10001,
            B11111,
            B01110
};


  lcd.print("*COVID19*");
  lcd.setCursor(0,1);
  lcd.print("..INOVT..");

  lcd.createChar(0,a1);
  lcd.createChar(1,a2);
  lcd.createChar(2,a3);
  lcd.createChar(7,a8);
  lcd.createChar(3,a4);
  lcd.createChar(4,a5);
  lcd.createChar(5,a6);
  lcd.createChar(6,a7);

  lcd.setCursor(10,0);
  lcd.write(byte(3));
  lcd.write(byte(4));
  lcd.write(byte(5));
  lcd.write(byte(3));
  lcd.write(byte(6));
  lcd.write(byte(5));
  lcd.setCursor(10,1);
  lcd.write(byte(0));
  lcd.write(byte(1));
  lcd.write(byte(2));
  lcd.write(byte(0));
  lcd.write(byte(7));
```

```
  lcd.write(byte(2));

}

void initior() {
byte c11[8] = {
B00000,
B00000,
B11111,
B11111,
B11111,
B11111,
B00000,
B00000
};
lcd.createChar(3, c11);
lcd.setCursor(0, 2);
lcd.write(3);
 delay(x);
lcd.setCursor(1, 2);
lcd.write(3);
 delay(x);
  digitalWrite(PIN_LED,HIGH);
lcd.setCursor(2, 2);
lcd.write(3);
 delay(x);
lcd.setCursor(3, 2);
lcd.write(3);
 delay(x);
 digitalWrite(PIN_LED2,HIGH);
lcd.setCursor(4, 2);
lcd.write(3);
 delay(x);
lcd.setCursor(5, 2);
lcd.write(3);
 delay(x);
  digitalWrite(PIN_LED,LOW);
lcd.setCursor(6, 2);
lcd.write(3);
 delay(x);
lcd.setCursor(7, 2);
lcd.write(3);
 delay(x);
  digitalWrite(PIN_LED2,LOW);
lcd.setCursor(8, 2);
lcd.write(3);
 delay(x);
 digitalWrite(PIN_LED,HIGH);
 digitalWrite(PIN_LED2,HIGH);
lcd.setCursor(9, 2);
lcd.write(3);
 delay(x);
```

```
  myServo.write(100);
 digitalWrite(PIN_LED,LOW);
 digitalWrite(PIN_LED2,LOW);
lcd.setCursor(10, 2);
lcd.write(3);
 delay(x);
 digitalWrite(PIN_LED,HIGH);
 digitalWrite(PIN_LED2,HIGH);
lcd.setCursor(11, 2);
lcd.write(3);
 delay(x);
  myServo.write(110);
 digitalWrite(PIN_LED,LOW);
 digitalWrite(PIN_LED2,LOW);
lcd.setCursor(12, 2);
lcd.write(3);
 delay(x);
 digitalWrite(PIN_LED,HIGH);
 digitalWrite(PIN_LED2,HIGH);
 lcd.setCursor(13, 2);
lcd.write(3);
 delay(x);
  myServo.write(115);
 digitalWrite(PIN_LED,LOW);
 digitalWrite(PIN_LED2,LOW);
lcd.setCursor(14, 2);
lcd.write(3);
 delay(x);
 digitalWrite(PIN_LED,HIGH);
 digitalWrite(PIN_LED2,HIGH);
lcd.setCursor(15, 2);
lcd.write(3);
 delay(x);
  myServo.write(120);
 digitalWrite(PIN_LED,LOW);
 digitalWrite(PIN_LED2,LOW);
 lcd.setCursor(16, 2);
lcd.write(3);
 delay(x);
 digitalWrite(PIN_LED,HIGH);
 digitalWrite(PIN_LED2,HIGH);
 lcd.setCursor(17, 2);
lcd.write(3);
 delay(x);
myServo.write(130);
 digitalWrite(PIN_LED,LOW);
 digitalWrite(PIN_LED2,LOW);
lcd.setCursor(18, 2);
lcd.write(3);
 delay(x);
 lcd.setCursor(19, 2);
lcd.write(3);
```

```
 delay(x);

}
void maobc() {
        thumbdownA();
        delay(x);
        lcd.clear();
        thumbsup();
        delay(x);
        thumbdownA();
        delay(x);
        lcd.clear();
        thumbsup();
        delay(x);
        thumbdownA();
        delay(x);
        lcd.clear();
        thumbsup();
        delay(x);
        thumbdownA();
        delay(x);
        lcd.clear();
        thumbsup();
        delay(1000);
        lcd.clear();
}

void thumbdownA() {
  byte thumb1[8] =
{B00001,B00010,B00011,B00100,B00011,B00100,B00011,B00100};
  byte thumb2[8] =
{B00011,B00000,B00000,B00000,B00000,B00000,B00000,B00000};
  byte thumb3[8] =
{B11110,B00001,B00000,B00000,B00000,B00000,B00000,B00000};
  byte thumb4[8] =
{B00000,B11110,B01000,B10001,B10010,B10010,B01100,B00000};
  byte thumb5[8] =
{B00000,B10000,B01110,B00010,B00010,B00010,B00010,B00010};
  byte thumb6[8] =
{B00110,B01000,B10000,B00000,B00000,B00000,B00000,B00000};
  lcd.createChar(6, thumb1);
  lcd.createChar(1, thumb2);
  lcd.createChar(2, thumb3);
  lcd.createChar(3, thumb4);
  lcd.createChar(4, thumb5);
  lcd.createChar(5, thumb6);
  lcd.setCursor(7,0);
  lcd.write(6);
  lcd.setCursor(7,1);
  lcd.write(1);
  lcd.setCursor(8,0);
  lcd.write(2);
```

```
  lcd.setCursor(8,1);
  lcd.write(3);
  lcd.setCursor(9,0);
  lcd.write(4);
  lcd.setCursor(9,1);
  lcd.write(5);
}

void thumbsup() {
 byte thumb1[8] =
{B00100,B00011,B00100,B00011,B00100,B00011,B00010,B00001};
 byte thumb2[8] =
{B00000,B00000,B00000,B00000,B00000,B00000,B00000,B00011};
 byte thumb3[8] =
{B00000,B00000,B00000,B00000,B00000,B00000,B00001,B11110};
 byte thumb4[8] =
{B00000,B01100,B10010,B10010,B10001,B01000,B11110,B00000};
 byte thumb5[8] =
{B00010,B00010,B00010,B00010,B00010,B01110,B10000,B00000};
 byte thumb6[8] =
{B00000,B00000,B00000,B00000,B00000,B10000,B01000,B00110};
 lcd.createChar(6, thumb1);
 lcd.createChar(1, thumb2);
 lcd.createChar(2, thumb3);
 lcd.createChar(3, thumb4);
 lcd.createChar(4, thumb5);
 lcd.createChar(5, thumb6);
 lcd.setCursor(7,1);
 lcd.write(6);
 lcd.setCursor(7,0);
 lcd.write(1);
 lcd.setCursor(8,1);
 lcd.write(2);
 lcd.setCursor(8,0);
 lcd.write(3);
 lcd.setCursor(9,1);
 lcd.write(4);
 lcd.setCursor(9,0);
 lcd.write(5);

}

void setServo()
{
  if ( millis() > time )
  {
    if ( estado == EXPIRANDO )
    {
      // Pasar a inspirandO
      digitalWrite(PIN_LED2,LOW);
      digitalWrite(PIN_LED,HIGH);
      myServo.write(MIN_ANGLE);
```

```
      time = millis() + (valInspira * 100);
      estado = INSPIRANDO;
      lcd.setCursor(8, 2);

       lcd.print("EXPIRANDO > ");
    }
    else if ( estado == INSPIRANDO )
    {
      // P0asar a expirando
      digitalWrite(PIN_LED2,HIGH);
      digitalWrite(PIN_LED,LOW);
      myServo.write(MAX_ANGLE - valPeep);
      time = millis() + (valExpira * 100);
      estado = EXPIRANDO;
      lcd.setCursor(8, 2);
      lcd.print("INSPIRANDO  < ");
    }
  }
}

void setup()
{

  myServo.attach(PIN_SERVO);
  myServo.write(92);
  //digitalWrite(PIN_LED,LOW);
  pinMode(PIN_FREQ,INPUT_PULLUP);
  pinMode(PIN_PEEP,INPUT_PULLUP);
  pinMode(SENSOR_FLOW,INPUT);
  pinMode(PIN_LED,OUTPUT);
  pinMode(PIN_LED2,OUTPUT);
  pinMode(PIN_Alarm_Relay,OUTPUT);
  pinMode(PIN_Safety_Valve,OUTPUT);
  lcd.init();
  lcd.backlight();
  lcd.clear();
  lcd.setCursor(0,0);
  logo();
  lcd.setCursor(4,2);
  lcd.print("OPEN-SOURCE");
  lcd.setCursor(0,3);
  lcd.print("Pulmonary Ventilator");
  delay(4000);
  lcd.clear();
  lcd.print("....Initializing....");
  initior();
  delay(500);
  lcd.clear();
  maobc();
  lcd.clear();
        if(digitalRead(SENSOR_FLOW) ==  LOW)
                    {
```

```
                    thumbsup();
                    lcd.setCursor(0,0);
                    lcd.print("SYSTEM");
                    lcd.setCursor(2,1);
                    lcd.print("OK");
                    delay(5000);
                    lcd.clear();
                    }
                                else
                    {
                    myServo.write(139);
                    thumbdownA();
                    lcd.setCursor(0,0);
                    lcd.print("SYSTEM");
                    lcd.setCursor(1,1);
                    lcd.print("FAIL");
                    delay(5000);
                    lcd.clear();
                    Serial.begin(9600);
                    time = millis();
                    frecuencia_ant=-1;
                    valPeep_ant=-1;                 }
                    delay(500);
                    estado=EXPIRANDO;

}

void loop()
{
digitalRead(SENSOR_FLOW);


  aux = analogRead(PIN_FREQ);
  frecuencia = map(aux,0,1023, 10, 60);
  valInspira = ((1.0 / frecuencia )* 600.0 ) * (1.0/3.0);
  valExpira = ((1.0 / frecuencia ) * 600.0 ) * (2.0/3.0);


  aux = analogRead(PIN_PEEP);
  valPeep = map(aux,0,1023, 0, 10);

  if ( (frecuencia_ant != frecuencia) || (valPeep_ant != valPeep))
  {
    //Pode monitorizar os valores de Frequencia e PEEP sem display usando
o Monitor serie do Arduino IDE.

    Serial.print("Frecuencia: ");
    Serial.println(frecuencia);
    Serial.print("Inspira: ");
    Serial.println(valInspira);
    Serial.print("Expira: ");
    Serial.println(valExpira);
```

```
   Serial.print("Peep: ");
   Serial.println(valPeep);

   lcd.setCursor(1, 0);
   lcd.print("FREQUENCIA: ");
   lcd.setCursor(13, 0);
   lcd.print(frecuencia);
   lcd.setCursor(1, 1);
   lcd.print("PEEP:  ");
   lcd.setCursor(13, 1);
   lcd.print(valPeep);
  // delay(500);
   lcd.setCursor(0, 2);
   lcd.print("ESTADO: ");
   lcd.setCursor(0, 3);
   lcd.print("VENTILADOR D.PASCOAL ");
   frecuencia_ant = frecuencia;
   valPeep_ant = valPeep;
 }
   if(digitalRead(SENSOR_FLOW) ==   HIGH)

 {
   myServo.write(139);
   digitalWrite(PIN_Alarm_Relay,HIGH);
   digitalWrite( PIN_Safety_Valve,HIGH);
   lcd.clear();
   thumbdownA();
   lcd.setCursor(0,0);
   lcd.print("SYSTEM");
   lcd.setCursor(1,1);
   lcd.print("FAIL");
   lcd.setCursor(1,2);
   lcd.print("******************");
   lcd.setCursor(3,3);
   lcd.print("Check AR flow");
   lcd.setCursor(12,1);
   lcd.print("ALARM");
   digitalWrite(PIN_LED,HIGH);
   digitalWrite(PIN_LED2,HIGH);
   delay(500);
   lcd.setCursor(12,1);
   lcd.print("        ");
   delay(500);
   lcd.setCursor(12,1);
   lcd.print("ALARM");
   digitalWrite(PIN_LED,LOW);
   digitalWrite(PIN_LED2,LOW);
   delay(1000);
   lcd.clear();
}
  else
{
```

```
  digitalWrite(PIN_Alarm_Relay,LOW);
  digitalWrite( PIN_Safety_Valve,LOW);
  setServo();

  delay(10);
}
}
```

٦- توضیحات توابع:

- setup () :
  - این تابع برای تنظیم اولیه یک سیستم تهویه ریوی (Pulmonary Ventilator) طراحی شده است. در این تابع:
  - تنظیمات اولیه: موتور سروو به پین مشخص شده متصل می‌شود و به موقعیت اولیه خود حرکت می‌کند.
  - پیکربندی پین‌ها: پین‌های مختلف به عنوان ورودی یا خروجی تنظیم می‌شوند.
  - تنظیمات LCD: نمایشگر LCD راه‌اندازی و روشن می‌شود، سپس پیام‌های اولیه و لوگو نمایش داده می‌شوند.
  - بررسی سنسور جریان: وضعیت سنسور جریان بررسی می‌شود. اگر سنسور جریان LOW باشد، تابع thumbsup فراخوانی می‌شود و پیام "SYSTEM OK" نمایش داده می‌شود. در غیر این صورت، موتور سروو به موقعیت دیگری حرکت می‌کند، تابع thumbdownA فراخوانی می‌شود و پیام "SYSTEM FAIL" نمایش داده می‌شود.
  - تنظیمات سریال: ارتباط سریال با نرخ ۹۶۰۰ بیت بر ثانیه آغاز می‌شود.
  - تنظیم متغیرها: متغیرهای time، frecuencia_ant و valPeep_ant مقداردهی اولیه می‌شوند.
  - تأخیر و تغییر حالت: پس از یک تأخیر کوتاه، حالت سیستم به EXPIRANDO تغییر می‌کند.

- این تابع به طور کلی برای آماده‌سازی و بررسی اولیه سیستم تهویه ریوی استفاده می‌شود.

- Initior() :
  - این تابع برای راه‌اندازی اولیه استفاده می‌شود .
  - ایجاد کاراکتر سفارشی: با استفاده از تابع lcd.createChar() یک کاراکتر سفارشی برای نمایشگر LCD ایجاد می‌کند.
  - به‌روزرسانی نمایشگر LCD : به صورت متوالی نمایشگر LCD را با نوشتن کاراکتر سفارشی در موقعیت‌های مختلف نشانگر به‌روزرسانی می‌کند.
  - کنترلLED : دو LED (PIN_LED) و (PIN_LED2) را با روشن و خاموش کردن در فواصل زمانی مشخص کنترل می‌کند.
    کنترل موتور سروو: موقعیت یک موتور سروو (myServo) را در زوایای مختلف (۱۰۰، ۱۱۰، ۱۱۵، ۱۲۰ و ۱۳۰ درجه) در مراحل مختلف تنظیم می‌کند.
- maobc () : این تابع به صورت متوالی توابع thumbdownA() و thumbsup()را اجرا می‌کند.
- thumbdownA() : این تابع برای ایجاد و نمایش یک شکل خاص (انگشت شست پایین) بر روی نمایشگر LCD طراحی شده است.
- thumbsup () : این تابع برای ایجاد و نمایش یک شکل خاص (انگشت شست بالا) بر روی نمایشگر LCD طراحی شده است.

- Loop() :
  - خواندن سنسورها: مقدار سنسور جریان (SENSOR_FLOW) و دو ورودی آنالوگ (PIN_FREQ و PIN_PEEP) خوانده می‌شوند.
  - محاسبه مقادیر: مقادیر فرکانس (frecuencia)، زمان دم (valInspira) و زمان بازدم (valExpira) و همچنین مقدار PEEP محاسبه می‌شوند.
  - نمایش مقادیر: اگر مقادیر فرکانس یا PEEP تغییر کرده باشند، این مقادیر بر روی نمایشگر LCD و مانیتور نمایش داده می‌شوند.
  - بررسی وضعیت سنسور جریان: اگر سنسور جریان مقدار HIGH داشته باشد، سیستم به حالت خطا می‌رود:
    - موتور سروو به موقعیت خاصی حرکت می‌کند.
    - رله آلارم و سوپاپ ایمنی فعال می‌شوند.
    - پیام‌های خطا و آلارم بر روی LCD نمایش داده می‌شوند.
    - LEDها روشن و خاموش می‌شوند تا آلارم را نشان دهند.
  - حالت عادی: اگر سنسور جریان مقدار LOW داشته باشد، سیستم به حالت عادی بازمی‌گردد:
    - رله آلارم و سوپاپ ایمنی غیرفعال می‌شوند.
    - تابع setServo فراخوانی می‌شود تا موتور سروو تنظیم شود.
  - این تابع به طور کلی برای کنترل و نظارت بر عملکرد سیستم تهویه ریوی و نمایش وضعیت آن بر روی LCD و مانیتور استفاده می‌شود.

نتیجه گیری

پروژه " ونتیلاتور ریوی COVID-19 " برای رفع کمبود ونتیلاتور در طول همه گیری COVID-19 طراحی شده است. ونتیلاتور غیر تهاجمی که از ماسک اسنورکل اصلاح شده برای رساندن هوای تحت فشار به ریه های بیماران استفاده می کند، و از فرآیند تنفس طبیعی آنها پشتیبانی می کند، به ویژه برای بیمارانی که بیهوش یا لوله گذاری نشده اند مفید است.
از مزیت های این پروژه میتوان به موارد زیر اشاره کرد:

۱- خلاقیت

این پروژه نمونه‌ای از کاربرد رباتیک در دنیای واقعی و در شرایط بحرانی است که با استفاده از وسایل در دسترس و ترکیب آنها به حل یک مسئله واقعی می‌پردازد. همچنین با ارائه طراحی به صورت منبع باز، این پروژه به هر کسی اجازه می‌دهد تا به آن دسترسی داشته باشد، آن را تغییر دهد و بهبود بخشد. این امر امکان نوآوری و همکاری سریع را فراهم می‌کند.

۲- قابلیت تطبیق: ونتیلاتور به گونه‌ای طراحی شده است که با نیازهای بالینی و محیط‌های مختلف سازگار باشد، که در دوران همه‌گیری که منابع و نیازها می‌توانند به طور گسترده‌ای متفاوت باشند، بسیار مهم است.

۳- مقرون به صرفه: پروژه بر استفاده از قطعات در دسترس و ارزان تمرکز دارد، که هزینه کلی را کاهش می دهد.

٤- سهولت ساخت: می توان آن را با ابزارها و مواد اولیه پایه ای ساخت که استفاده گسترده از آن را امکان پذیر میکند.

۵- غیر تهاجمی: پشتیبانی تنفسی را بدون نیاز به لوله گذاری فراهم می کند که می تواند برای بیماران راحت تر و مدیریت آن آسان تر باشد.

این پروژه نشان می‌دهد که چگونه راه‌حل‌های نوآورانه و جامعه محور می‌توانند نقشی حیاتی در رسیدگی به چالش‌های بهداشت جهانی ایفا کنند.

٦- کمک به درک آردوینو و رباتیک

پروژه از بردهای آردوینو برای کنترل عملکردهای مختلف ونتیلاتور استفاده می‌کند. این یک مثال عملی از چگونگی استفاده از میکروکنترلرها در کاربردهای دنیای واقعی ارائه می‌دهد.

ساخت ونتیلاتور شامل درک و پیاده‌سازی مفاهیم مختلف رباتیک مانند سنسورها، محرک‌ها و سیستم‌های کنترلی است. این تجربه عملی برای یادگیری نحوه طراحی و کارکرد سیستم‌های رباتیک بسیار ارزشمند است.

این پروژه دانش الکترونیک، برنامه‌نویسی، طراحی مکانیکی و مهندسی زیست‌پزشکی را ترکیب می‌کند. این رویکرد بین‌رشته‌ای به درک چگونگی ترکیب زمینه‌های مختلف برای حل مشکلات پیچیده کمک می‌کند.

منبع

Open-Source COVID-19 Pulmonary Ventilator _ By David Pascoal
Published April 19, 2020 © MIT

https://www.hackster.io/david-pascoal/open-source-covid-19-pulmonary-ventilator-4f4586