ANEXO 1

```
#define en 8      /*pin activar motores*/
#define xdir 5   /*pin direccion eje x*/
#define xstep 2  /*pin pasos eje x*/
#define ydir 6   /*pin direccion eje y*/
#define ystep 3  /*pin pasos eje y*/

#define leds 4  /* pin de iluminacion */
#define ledir 7 /*pin de led infrarojo*/

#define xendstop 9   /*pin fin de carrera eje x*/

#define PASOS (200)

int retardo = 1500;

uint32_t dim;
uint32_t dimenpasos;
int dofenpasos;
float valdiafragma = 5.6;
int numpasosmotory = 20;
int numerofotos;
```

```
//uint32_t dim;
//uint32_t dim;

int intervaloMedidas = 1000;
int auxMillis = 0;

#include "Nextion.h" /*agregar libreria nextion*/

/*referenciar los botones de nextion*/
NexButton b21cambdiaf = NexButton(2, 9, "b21cambdiaf");
NexButton b21cambdism = NexButton(2, 10, "b21cambdism");
NexButton b21cambfactm = NexButton(2, 11, "b21cambfactm");
NexButton b3setdim = NexButton(3, 3, "b3setdim");
NexButton b31iniciar = NexButton(4, 2, "b31iniciar");
NexButton b31emergencia = NexButton(4, 3, "b31emergencia");
NexButton b311izq_ = NexButton(5, 1, "b311izq_");
NexButton b311izq = NexButton(5, 4, "b311izq");
NexButton b311der = NexButton(5, 3, "b311der");
NexButton b311der_ = NexButton(5, 2, "b311der_");

NexDSButton luz = NexDSButton(3 , 5 , "bt3onoff");

/*agregar los botones a la lista de componentes*/
```

```
NexTouch *botones[] =
{
  &b21cambdiaf,
  &b21cambdism,
  &b21cambfactm,
  &b3setdim,
  &b31iniciar,
  &b31emergencia,
  &b311izq_,
  &b311izq,
  &b311der,
  &b311der_,
  &luz,
//  &b0config,//borrar
  &invisible,//borrar
  NULL
};

// FUNCIONES DE BOTONES DE NEXTION

void mostrardatos()
{
  //para mostrar datos de diafragma fact magn y dist min de enfoque
```

```
  diafragma.setText("valdiafragma");
  x2vdiaf.setValue(valdiafragma);
  x21vdiaf.setValue(20);
  n0.setValue(10);
}


void seteardimension()
{
  uint32_t dimingresada;
  n3dimobj.getValue(&dimingresada);
  //dimenpasos=n3dimobj.getValue(&dimingresada)*100;
  dimenpasos = dimingresada * 100;
  dofenpasos = dof() * 100;
  numerofotos = dimenpasos / dofenpasos + 1;
  //  n31tgiro.setValue(numpasosmotory);
  //  n31tfoto.setValue(numerofotos);

  n31tgiro.setValue(numpasosmotory);
}

void moverizq_()
{
```

```
  motorizq(xstep, xdir, en, 100);
}

void moverizq()
{
  motorizq(xstep, xdir, en, 10);
}

void moverder()
{
  motorder(xstep, xdir, en, 10);
}

void moverder_()
{
  motorder(xstep, xdir, en, 100);
}

void encenderluces()
{
  uint32_t estadoluz;
  luz.getValue(&estadoluz);
  if (estadoluz == 1)
```

```
  {
    digitalWrite(leds, 1);
  }
  else
  {
    digitalWrite(leds, 0);
  }
}

void motorder(int paso_, int dire_, int habi_, int pasos)
{
  digitalWrite(habi_, LOW);  // Habilita el Driver
  digitalWrite(dire_, LOW);   // direccion de giro 2
  darpasos (pasos, paso_);
  digitalWrite(habi_, HIGH);   // quita la habilitacion del Driver
  delay(1000);
}
void moovinicial(int paso_, int dire_, int habi_, int pasos)
{
  digitalWrite(habi_, LOW);  // Habilita el Driver
  digitalWrite(dire_, LOW);   // direccion de giro 2
  darpasos (pasos, paso_);
  digitalWrite(habi_, HIGH);   // quita la habilitacion del Driver
```

```
  //delay(1000);
}

void motorizq(int paso_, int dire_, int habi_, int pasos)
{
  digitalWrite(habi_, LOW);  // Habilita el Driver
  digitalWrite(dire_, HIGH);   // direccion de giro 2
  darpasos (pasos, paso_);
  digitalWrite(habi_, HIGH);   // quita la habilitacion del Driver
  delay(1000);
}


//FIN FUNCIONES BOTONES

void darpasos (int numpasos, int pasospin)
{
  for (int i = 0; i < numpasos; i++)
  { // da  pasos
    digitalWrite(pasospin, HIGH);
    delayMicroseconds(retardo);
    digitalWrite(pasospin, LOW);
    delayMicroseconds(retardo);
```

```
  }
}

//FUNCIONES PARA CLIC AUTOMATICO DE FOTO

void mandarpulsos(int anchodepulso)
{
  int reps = anchodepulso / 23.6;
  for (int i = 0; i <= reps; i++)
  {
    digitalWrite(ledir, HIGH);
    delayMicroseconds(11);
    digitalWrite(ledir, LOW);
    delayMicroseconds(5);
  }
}

void enviarsecuencia()
{
  for (int i = 0; i < 1; i++)
  {
    mandarpulsos(2000);
    delay(27);
```

```
    delayMicroseconds(800);
    mandarpulsos(500);
    delayMicroseconds(1500);
    mandarpulsos(500);
    delayMicroseconds(3500);
    mandarpulsos(500);
    if (i < 1)
    {
      delay(63);
    }
  }
}

void hacerfoto(int reps, int timeInterval)
{
  for (int i = 0; i <= reps; i++)
  {
    enviarsecuencia();
    delay(timeInterval * 1000);
  }
}
// FIN FUNCIONES DE TOMAR FOTO
```

```
//CALCULO DE APILADO
void apilado()
{
  n31tfoto.setValue(numerofotos);
  n31tgiro.setValue(numpasosmotory);
  for (int i = 1; i <= numpasosmotory; i++)
  {
    n31cgiro.setValue(i);
    for (int foto = 1; foto <= numerofotos; foto++)
    {
      n31cfoto.setValue(foto);
      recorridocortox(xstep, xdir, en, dofenpasos);
      //CAMBIAR PASOS
      delay(750);
      hacerfoto(1, 0);
      delay(750);
    }
    regresox(xstep, xdir, en, dimenpasos);
    giroy(ystep, ydir, en, PASOS / numpasosmotory);
    //CAMBIAR PASOS
  }
  delay(500);
}
```

```
//profundidad de campo
double dof()
{
  double f = 8;
  int m = 1;
  double coc = 0.020;
  double dof = 2 * coc * f * ((m + 1) / (m * m));
  double consolapamiento = dof * 0.8;
  //dofenpasos = consolapamiento * 100;
  return (consolapamiento);
}

void recorridocortox(int paso_, int dire_, int habi_, int pasos)
{
  digitalWrite(habi_, LOW);  // Habilita el Driver
  digitalWrite(dire_, LOW);   // direccion de giro 1
  darpasos (pasos, paso_);
  digitalWrite(habi_, HIGH);   // quita la habilitacion del Driver
  delay(1000);
}
void regresox(int paso_, int dire_, int habi_, int pasos)
{
  digitalWrite(habi_, LOW);  // Habilita el Driver
```

```
  digitalWrite(dire_, HIGH);   // direccion de giro 2
  darpasos (pasos, paso_);
  digitalWrite(habi_, HIGH);   // quita la habilitacion del Driver
  delay(1000);
}
void giroy(int paso_, int dire_, int habi_, int pasos)
{
  digitalWrite(habi_, LOW);  // Habilita el Driver
  digitalWrite(dire_, LOW);   // direccion de giro 1
  //darpasos (pasos, paso_);
  for (int i = 0; i < pasos; i++)
  { // da  pasos
    digitalWrite(paso_, HIGH);
    delayMicroseconds(10000);
    digitalWrite(paso_, LOW);
    delayMicroseconds(10000);
  }
  digitalWrite(habi_, HIGH);   // quita la habilitacion del Driver
  delay(1000);
}

void posicionarplato()
{
```

```
  while (digitalRead(xendstop) == HIGH)
  {
    moovinicial(xstep, xdir, en, 100);
  }

  motorizq(xstep, xdir, en, 10400);
}


void setup()
{
  Serial.begin(9600);
  nexInit();
  b21cambdiaf.attachPop(cambiardiafragma);
  b21cambdism.attachPop(cambiardisminima);
  b21cambfactm.attachPop(cambiarfactmagnificacion);
  b3setdim.attachPop(seteardimension);
  b31iniciar.attachPop(apilado);
  b311izq_.attachPop(moverizq_);
  b311izq.attachPop(moverizq);
  b311der.attachPop(moverder);
  b311der_.attachPop(moverder_);
  luz.attachPop(encenderluces);
```

```
  pinMode(en, OUTPUT);
  pinMode(xdir, OUTPUT); pinMode(xstep, OUTPUT);
  pinMode(ydir, OUTPUT); pinMode(ystep, OUTPUT);
  pinMode(xendstop, INPUT_PULLUP);
  pinMode(leds, OUTPUT);
  pinMode(ledir, OUTPUT);
}

void loop()
{
  nexLoop(botones);
}
```