```
int inicio = 0;
int pausa = 10;
int tempo_passo = 15;
float sensibilidade = 0;

int angulo_atual_topo = 0;

int vetor_posicao[4] = {0,0,0,0};

int angulo_atual_base = 0;

  int passos_topo = 0;
  int passos_base = 0;
  int amplitude_topo = 0;
  int amplitude_base = 0;

int PUL_TOPO = 10; //define Pulse pin topo
int DIR_TOPO = 9; //define Direction pin topo
int ENA_TOPO = 8; //define Enable Pin topo

int PUL_BASE = 13; //define Pulse pin base
int DIR_BASE = 12; //define Direction pin base
int ENA_BASE = 11; //define Enable Pin base

int sensor_A = 0;
int sensor_B = 0;
int sensor_C = 0;
int sensor_D = 0;

void TOPO_HORARIO();
void TOPO_ANTIHORARIO();

void BASE_HORARIO();
void BASE_ANTIHORARIO();

void LE_SENSORES();

int INICIO(int * vetor);

void setup() {
  //VALOR REFERENCIAL
  analogReference(DEFAULT);
  pinMode(PUL_TOPO, OUTPUT);
  pinMode(DIR_TOPO, OUTPUT);
  pinMode(ENA_TOPO, OUTPUT);

  pinMode(PUL_BASE, OUTPUT);
  pinMode(DIR_BASE, OUTPUT);
  pinMode(ENA_BASE, OUTPUT);

  Serial.begin(9600);
}
```

```
void loop() {

if(inicio ==0){
  INICIO(vetor_posicao);
passos_topo = 0;
passos_base = 0;
}

  inicio = 1;



  sensibilidade = 1 + (analogRead(A6))*0.10;


  LE_SENSORES();

  //-------------------------------------------------------------------------------------------------------------------------------------------------/
  // ------------------------------- MOVIMENTO ALIMENTADO PELA LEITURA DOS SENSORES ---------------------------------------------------------------------------/
  //
-------------------------------------------------------------------------------------------------------------------------------------------------/

  if ((sensor_A - sensor_B > sensibilidade) && (sensor_A - sensor_C > sensibilidade) &&
(sensor_A - sensor_D > sensibilidade)) {

     if(passos_topo <= 0){
       TOPO_ANTIHORARIO();              //         A    X
       passos_topo++;
       BASE_HORARIO();                  //         X    X
       passos_base--;
      }else{
       TOPO_ANTIHORARIO();              //         A    X
       passos_topo++;
       BASE_ANTIHORARIO();              //         X    X
       passos_base++;
      }

  } else if ((sensor_B - sensor_A > sensibilidade) && (sensor_B - sensor_C > sensibilidade) &&
(sensor_B - sensor_D > sensibilidade)) {

     if(passos_topo <= 0){
       TOPO_ANTIHORARIO();              //         X    B
       passos_topo++;
       BASE_ANTIHORARIO();              //         X    X
       passos_base++;
     }else{
       TOPO_ANTIHORARIO();              //         X    B
       passos_topo++;
```

```
    BASE_HORARIO();                 //        X    X
    passos_base--;
   }

  } else if ((sensor_C - sensor_B > sensibilidade) && (sensor_C - sensor_A > sensibilidade) &&
(sensor_C - sensor_D > sensibilidade)) {

   if(passos_topo <= 0){
    TOPO_HORARIO();                 //        X    X
    passos_topo--;
    BASE_HORARIO();                 //        C    X
    passos_base--;
   }else{
    TOPO_ANTIHORARIO();             //        X    X
    passos_topo++;
    BASE_ANTIHORARIO();             //        C    X
    passos_base++;
   }

  } else if ((sensor_D - sensor_B > sensibilidade) && (sensor_D - sensor_C > sensibilidade) &&
(sensor_D - sensor_A > sensibilidade)) {

   if(passos_topo <= 0){
    TOPO_HORARIO();                 //        X    X
    passos_topo--;
    BASE_ANTIHORARIO();             //        X    D
    passos_base++;
   }else{
    TOPO_HORARIO();                 //        X    X
    passos_topo--;
    BASE_HORARIO();                 //        X    D
    passos_base--;
    }

  } else if ((sensor_A - sensor_C > sensibilidade) && (sensor_A - sensor_D > sensibilidade) &&
(sensor_B - sensor_C > sensibilidade) && (sensor_B - sensor_D > sensibilidade) &&
(abs(sensor_A - sensor_B) <= sensibilidade)) {

    TOPO_ANTIHORARIO();             //        A    B
    passos_topo++;                  //        X    X

  } else if ((sensor_C - sensor_A > sensibilidade) && (sensor_C - sensor_B > sensibilidade) &&
(sensor_D - sensor_A > sensibilidade) && (sensor_D - sensor_B > sensibilidade) &&
(abs(sensor_C - sensor_D) <= sensibilidade)) {

   TOPO_HORARIO();                  //        X    X
   passos_topo--;
                                    //        C    D
  } else if ((sensor_A - sensor_B > sensibilidade) && (sensor_A - sensor_D > sensibilidade) &&
(sensor_C - sensor_B > sensibilidade) && (sensor_C - sensor_D > sensibilidade) &&
(abs(sensor_A - sensor_C) <= sensibilidade)) {
```

```
     if(passos_topo <= 0){
       BASE_HORARIO();                //        A    X
       passos_base--;                 //        C    X
     }else{
       BASE_ANTIHORARIO();            //        A    X
       passos_base++;                 //        C    X
     }

  } else if ((sensor_B - sensor_A > sensibilidade) && (sensor_B - sensor_C > sensibilidade) &&
(sensor_D - sensor_A > sensibilidade) && (sensor_D - sensor_C > sensibilidade) &&
(abs(sensor_B - sensor_D) <= sensibilidade)) {

    if(passos_topo <= 0){
       BASE_ANTIHORARIO();            //        X    B
       passos_base++;                 //        X    D
    }else{
       BASE_HORARIO();                //        X    B
       passos_base--;                 //        X    D
    }

  } else if ((sensor_A - sensor_C > sensibilidade) && (sensor_B - sensor_C > sensibilidade) &&
(sensor_D - sensor_C > sensibilidade) && (abs(sensor_A - sensor_B) <= sensibilidade) &&
(abs(sensor_A - sensor_D) <= sensibilidade) && (abs(sensor_B - sensor_D) <= sensibilidade)) {

    if(passos_topo <= 0){
       TOPO_ANTIHORARIO();            //        A    B
       passos_topo++;
       BASE_ANTIHORARIO();            //        X    D
       passos_base++;
    }else{
       TOPO_ANTIHORARIO();            //        A    B
       passos_topo++;
       BASE_HORARIO();                //        X    D
       passos_base--;
    }

  } else if ((sensor_B - sensor_A > sensibilidade) && (sensor_C - sensor_A > sensibilidade) &&
(sensor_D - sensor_A > sensibilidade) && (abs(sensor_B - sensor_C) <= sensibilidade) &&
(abs(sensor_B - sensor_D) <= sensibilidade) && (abs(sensor_C - sensor_D) <= sensibilidade)) {

    if(passos_topo <= 0){
       TOPO_HORARIO();                //        X    B
       passos_topo--;
       BASE_ANTIHORARIO();            //        C    D
       passos_base++;
    }else{
       TOPO_HORARIO();                //        X    B
       passos_topo--;
       BASE_HORARIO();                //        C    D
       passos_base--;
     }
```

```
  } else if ((sensor_A - sensor_B > sensibilidade) && (sensor_C - sensor_B > sensibilidade) &&
(sensor_D - sensor_B > sensibilidade) && (abs(sensor_A - sensor_C) <= sensibilidade) &&
(abs(sensor_A - sensor_D) <= sensibilidade) && (abs(sensor_C - sensor_D) <= sensibilidade)) {

     if(passos_topo <= 0){
       TOPO_HORARIO();              //        A    X
       passos_topo--;
       BASE_HORARIO();              //        C    D
       passos_base--;
     }else{
       TOPO_HORARIO();              //        A    X
       passos_topo--;
       BASE_ANTIHORARIO();          //        C    D
       passos_base++;
     }

  } else if ((sensor_A - sensor_D > sensibilidade) && (sensor_B - sensor_D > sensibilidade) &&
(sensor_C - sensor_D > sensibilidade) && (abs(sensor_A - sensor_B) <= sensibilidade) &&
(abs(sensor_A - sensor_C) <= sensibilidade) && (abs(sensor_B - sensor_C) <= sensibilidade)) {

     if(passos_topo <= 0){
       TOPO_ANTIHORARIO();          //        A    B
       passos_topo++;
       BASE_HORARIO();              //        C    X
       passos_base--;
     }else{
       TOPO_ANTIHORARIO();          //        A    B
       passos_topo++;
       BASE_ANTIHORARIO();          //        C    X
       passos_base++;
     }

  } else {

    // MOVIMENTO TOPO-----------------------------------------------/
    //--------------------NÃO HÁ-------------------------------------/

    // MOVIMENTO BASE-----------------------------------------------/
    //--------------------NÃO HÁ-------------------------------------/

  }
}
```

```
//---------------------------------------------------------------------
// Funções utilizadas no programa--------------------------------------
```

```
//---------------------------------------------------------------------

void LE_SENSORES() {
  sensor_D = analogRead(A2);
  sensor_B = analogRead(A0);
  sensor_C = analogRead(A3);
  sensor_A = analogRead(A1);
//  if((sensor_A >= sensor_C)&&(sensor_A >= sensor_B)&&(sensor_A >= sensor_D)&&(sensor_A
>= 10)){
//    sensor_B = sensor_B * (1023/sensor_A);
//    sensor_C = sensor_C * (1023/sensor_A);
//    sensor_D = sensor_D * (1023/sensor_A);
//    sensor_A = sensor_A * (1023/sensor_A);
//    }else if((sensor_B >= sensor_A)&&(sensor_B >= sensor_C)&&(sensor_B >=
sensor_D)&&(sensor_B >= 10)){
//    sensor_B = sensor_B * (1023/sensor_B);
//    sensor_C = sensor_C * (1023/sensor_B);
//    sensor_D = sensor_D * (1023/sensor_B);
//    sensor_A = sensor_A * (1023/sensor_B);
//    }else if((sensor_C >= sensor_A)&&(sensor_C >= sensor_B)&&(sensor_C >=
sensor_D)&&(sensor_C >= 10)){
//    sensor_B = sensor_B * (1023/sensor_C);
//    sensor_C = sensor_C * (1023/sensor_C);
//    sensor_D = sensor_D * (1023/sensor_C);
//    sensor_A = sensor_A * (1023/sensor_C);
//    }else if((sensor_D >= sensor_A)&&(sensor_D >= sensor_B)&&(sensor_D >=
sensor_C)&&(sensor_D >= 10)){
//    sensor_B = sensor_B * (1023/sensor_D);
//    sensor_C = sensor_C * (1023/sensor_D);
//    sensor_D = sensor_D * (1023/sensor_D);
//    sensor_A = sensor_A * (1023/sensor_D);
//    }
}


void TOPO_ANTIHORARIO() {

  if ( !digitalRead(2) ) {         //chave_topo_antihorario
    digitalWrite( DIR_TOPO, HIGH); // sentido anti-horario
    digitalWrite( ENA_TOPO, HIGH);
    digitalWrite( PUL_TOPO, HIGH);
    delayMicroseconds(tempo_passo);
    digitalWrite(PUL_TOPO, LOW);
    delay(pausa);
  }
}

//---------------------------------------------------------------------
//---------------------------------------------------------------------
//---------------------------------------------------------------------
```

```
void TOPO_HORARIO() {

  if ( !digitalRead(3) ) {          //chave_topo_horario
    digitalWrite( DIR_TOPO, LOW); // sentido horario
    digitalWrite( ENA_TOPO, HIGH);
    digitalWrite( PUL_TOPO, HIGH);
    delayMicroseconds(tempo_passo);
    digitalWrite(PUL_TOPO, LOW);
    delay(pausa);
  }
}

//---------------------------------------------------------------------
//---------------------------------------------------------------------
//---------------------------------------------------------------------


void BASE_ANTIHORARIO() {

  if ( !digitalRead(5) ) {         // chave base anti-horario
    digitalWrite( DIR_BASE, LOW); // sentido anti-horario
    digitalWrite( ENA_BASE, HIGH);
    digitalWrite( PUL_BASE, HIGH);
    delayMicroseconds(tempo_passo);
    digitalWrite(PUL_BASE, LOW);
    delay(pausa / 2);
  }
}

//---------------------------------------------------------------------
//---------------------------------------------------------------------
//---------------------------------------------------------------------


void BASE_HORARIO() {

  if ( !digitalRead(4) ) {          // chave base horária
    digitalWrite( DIR_BASE, HIGH); // sentido horario
    digitalWrite( ENA_BASE, HIGH);
    digitalWrite( PUL_BASE, HIGH);
    delayMicroseconds(tempo_passo);
    digitalWrite(PUL_BASE, LOW);
    delay(pausa / 2);
  }

}

//---------------------------------------------------------------------
//---------------------------------------------------------------------
//---------------------------------------------------------------------
```

```
int INICIO(int *vetor) {  // A função INICIO() tem como objetivo posicionar os eixos motores nos respectivos poontos médios de suas amplitudes delimitadas pelas chaves

  //--------------------------- TOPO ----------------------------------/

  if (inicio == 0) {

    int passos_atual_topo = 0;
    int passos_atual_base = 0;
    int amplitude_passos_topo = 0;
    int amplitude_passos_base = 0;
    int valor_medio_topo = 0;
    int valor_medio_base = 0;

    if(digitalRead(3)){
      while ( !digitalRead(2) ) {       //chave_topo_antihorario
      digitalWrite( DIR_TOPO, HIGH); // sentido anti-horario
      digitalWrite( ENA_TOPO, HIGH);
      digitalWrite( PUL_TOPO, HIGH);
      delayMicroseconds(tempo_passo);
      digitalWrite(PUL_TOPO, LOW);
      delay(pausa/2);

      passos_atual_topo++;
      amplitude_passos_topo++;
    }
     }else if(digitalRead(2)){
      while ( !digitalRead(3) ) {      //chave_topo_horario
      digitalWrite( DIR_TOPO, LOW); // sentido horario
      digitalWrite( ENA_TOPO, HIGH);
      digitalWrite( PUL_TOPO, HIGH);
      delayMicroseconds(tempo_passo);
      digitalWrite(PUL_TOPO, LOW);
      delay(pausa/2);
           passos_atual_topo++;
      amplitude_passos_topo++;
    }
       }else{
          while ( !digitalRead(3) ) {      //chave_topo_horario
      digitalWrite( DIR_TOPO, LOW); // sentido horario
      digitalWrite( ENA_TOPO, HIGH);
      digitalWrite( PUL_TOPO, HIGH);
      delayMicroseconds(tempo_passo);
      digitalWrite(PUL_TOPO, LOW);
      delay(pausa/2);
    }
    while ( !digitalRead(2) ) {       //chave_topo_antihorario
      digitalWrite( DIR_TOPO, HIGH); // sentido anti-horario
      digitalWrite( ENA_TOPO, HIGH);
      digitalWrite( PUL_TOPO, HIGH);
      delayMicroseconds(tempo_passo);
      digitalWrite(PUL_TOPO, LOW);
```

```
      delay(pausa/2);

      passos_atual_topo++;
      amplitude_passos_topo++;
    }

        }

   valor_medio_topo = (amplitude_passos_topo) / 2;   // calcula o ponto médio da trajetória

if(digitalRead(2)){
     for (int i = 0; i <= valor_medio_topo; i++) {

    if( !digitalRead(3) ){
     digitalWrite( DIR_TOPO, LOW);                // sentido horario até o ponto médio
     digitalWrite( ENA_TOPO, HIGH);
     digitalWrite( PUL_TOPO, HIGH);
     delayMicroseconds(tempo_passo);
     digitalWrite(PUL_TOPO, LOW);
     delay(pausa/2);

     passos_atual_topo--;
    }
    }
  }else{
        for (int i = 0; i <= valor_medio_topo; i++) {

    if( !digitalRead(2) ){
     digitalWrite( DIR_TOPO, HIGH);                // sentido antihorario até o ponto médio
     digitalWrite( ENA_TOPO, HIGH);
     digitalWrite( PUL_TOPO, HIGH);
     delayMicroseconds(tempo_passo);
     digitalWrite(PUL_TOPO, LOW);
     delay(pausa/2);

     passos_atual_topo++;
    }
    }
    }


   vetor_posicao[0] = passos_atual_topo;
   vetor_posicao[1] = passos_atual_base;
   vetor_posicao[2] = amplitude_passos_topo;
   vetor_posicao[3] = amplitude_passos_base;

   return 1;
  }
}
```