```
//defines de tempo
#define SLEEP_ONLINE 10L
//defines de posicao
#define SENSOR_PORT1 6
#define SENSOR_PORT2 5

#define ID_MOTOR_E 1
#define ID_MOTOR_D 2

//definicao de dados
#define DELTA_DADOS 50
#define DELTA_CALIB 10
#define DELTA_SENSOR 3
#define DELTA_SENSOR2 8

//motors
#define POTENCIA_RETA_ME 50
#define POTENCIA_RETA_MD 50

#define POTENCIA_D_ME 80
#define POTENCIA_D_MD 60

#define POTENCIA_E_ME 30
#define POTENCIA_E_MD 90

//mover
#define MOVE_D(power) (motor(ID_MOTOR_D,power))
#define MOVE_E(power) (motor(ID_MOTOR_E,power))
#define MOVE(esquerda,direita) {MOVE_E(esquerda);MOVE_D(direita);}


//GLOBAS
int dados[DELTA_DADOS];
persistent int preto;
persistent int branco;
persistent int midline;

persistent int bloco_00;
persistent int bloco_01;
persistent int bloco_02;
persistent int bloco_03;

char nome_cor_00[] = "Preto";
char nome_cor_01[] = "Vermelho";
char nome_cor_02[] = "Amarelo";
char nome_cor_03[] = "Azul";


//FUNCOES

/***************
funcao que faz a identificao dos blocos baseado na cor calibrada
***************/
void identify_bloco()
{
    int sensor_value;
    int temp[DELTA_SENSOR];
    int diferencas[4];
```

```
    int i;
    int menor;

    //medindo sensor
    get_dados(SENSOR_PORT2, temp, DELTA_SENSOR, 10L);
    quicksort(temp, DELTA_SENSOR);
    sensor_value = temp[DELTA_SENSOR/2];

    diferencas[0] = sensor_value - bloco_00;
    diferencas[1] = sensor_value - bloco_01;
    diferencas[2] = sensor_value - bloco_02;
    diferencas[3] = sensor_value - bloco_03;

    for (i=0; i<4; i++){
        if ( diferencas[i] < 0 )
          diferencas[i] = -diferencas[i];
    }

    menor = 0;
    for (i=1; i<4; i++){
        if (diferencas[i] < diferencas[menor])
          menor = i;
    }

    switch(menor){
        case 0:
          printf("%s\n", nome_cor_00);
        break;
        case 1:
          printf("%s\n", nome_cor_01);
        break;
        case 2:
          printf("%s\n", nome_cor_02);
        break;
        case 3:
          printf("%s\n", nome_cor_03);
        break;
    }

    while(!start_button());
    while(start_button());

    return;
}

/***************
funcao que faz a calibracao das cores dos blocos
***************/

void calibrate_blocos()
{
    _calibrate_bloco(&bloco_00, nome_cor_00);
    _calibrate_bloco(&bloco_01, nome_cor_01);
    _calibrate_bloco(&bloco_02, nome_cor_02);
    _calibrate_bloco(&bloco_03, nome_cor_03);

    return;
}
```

```
void _calibrate_bloco(int* bloco, char cor[])
{
    int temp[DELTA_CALIB];
    printf("Start => calibrar bloco %s.\n", cor);

    while(!start_button());
    while(start_button());

    get_dados(SENSOR_PORT2, temp, DELTA_CALIB, 100L);
    quicksort(temp, DELTA_CALIB);
    *bloco = temp[(DELTA_CALIB/2)];

    return;
}


/***************
funcao que faz a leitura do sensor em tempo real
***************/
void sensor_real_time()
{
    while(1)
      {
        printf("Sensor 1: %d - Sensor 2: %d\n", analog(SENSOR_PORT1),
analog(SENSOR_PORT2));
        if(stop_button() || start_button())
          break;
        sleep(0.2);
    }
    while(!stop_button() && !start_button());

    return;
}

/***************
funcao que faz o robo andar sobre a linha 1
***************/
void run_online(int algoritmo)
{
    int pid;
    printf("Andando sobre a linha (%d)\n", algoritmo);

    if(algoritmo == 1)
      pid = start_process(running_online());
    else
      pid = start_process(running_ondif());
    sleep(8.0);
    kill_process(pid);
    ao();
    return;
}

/***************
funcao que faz o robo andar sobre a linha 2
***************/
void running_online()
{
```

```
    int temp[DELTA_SENSOR];
    int sensor_value;
    int state = 1; // 1 = preto, 0= branco
    int direction = 0; // 0 = direita , 1 = esquerda

    while(1)
      {
        get_dados(SENSOR_PORT1, temp, DELTA_SENSOR, 10L);
        quicksort(temp, DELTA_SENSOR);
        sensor_value = temp[DELTA_SENSOR/2];
        MOVE(POTENCIA_D_ME, POTENCIA_D_MD);

        if(state)
          { //estava sobre linha preta
            if(sensor_value > midline)
              {
                printf("00\n");
                msleep(SLEEP_ONLINE);
            }
            else
              {//encontrou branco
                printf("01\n");
                direction = !direction;
                if(direction)
                  MOVE(POTENCIA_E_ME, POTENCIA_E_MD);
                else
                  MOVE(POTENCIA_D_ME, POTENCIA_D_MD);
                sleep(0.5);
            }
        }
        else
          {//linha branca
            if(sensor_value > midline)
              {
                printf("03\n");
                state = 1;
            }
            else
              {
                printf("04\n");
                //do nothing
            }
        }
    }
}

/***************
funcao que faz o robo andar sobre a linha
baseado nas direfencas
***************/
void running_ondif()
{
    int temp[DELTA_SENSOR];
    int sensor_value;
    int state = 1; // 1 = preto, 0= branco
    int direction = 0; // 0 = direita , 1 = esquerda

    while(1)
```

```
    {
        get_dados(SENSOR_PORT1, temp, DELTA_SENSOR, 10L);
        quicksort(temp, DELTA_SENSOR);
        sensor_value = temp[DELTA_SENSOR/2];
        MOVE(POTENCIA_D_ME, POTENCIA_D_MD);

        if(sensor_value > midline)
        {
            printf("00\n");
            MOVE(POTENCIA_E_ME, POTENCIA_E_MD);
        }
        else
        {
            printf("00\n");
            MOVE(POTENCIA_D_ME, POTENCIA_D_MD);
        }
        sleep(0.0);
    }
}
/***************
funcao quick sort de ordenao
***************/
void quicksort(int Vetor[], int size) {
    _quicksort(Vetor, 0, size-1);
}

void _quicksort(int Vetor[], int esq, int dir) {
    int temp;
    int pivo = Vetor[(esq+dir)/2];
    int i = esq;
    int j = dir;

    do{
        while(Vetor[i] < pivo) i++;
        while(Vetor[j] > pivo) j--;

        if (!(i < j)) break;

        temp = Vetor[i]; Vetor[i]=Vetor[j]; Vetor[j]=temp;
        i++; j--;

    } while(10);

    if (j-esq < dir-(j+1)){
        if(  esq <  j ) _quicksort(Vetor, esq, j);
        if((j+1) < dir) _quicksort(Vetor, j+1, dir);
    }
    else {
        if((j+1) < dir) _quicksort(Vetor, j+1, dir);
        if(  esq <  j ) _quicksort(Vetor, esq, j);
    }
}


/***************
***************/
void inicialize_dados()
{
```

```
    int i;
    printf("#Inicializando Dados\n");
    for(i=0; i< DELTA_DADOS; i++)
      dados[i] = -1;
    return;
}

/***************
***************/
void get_dados(int sensor, int Vetor[], int size, long sleep_time)
{
    int count;
    for(count =0; count < size; count++)
      {
        Vetor[count] = analog(sensor);
        msleep(sleep_time);
        //printf("%d\n",count);
    }
    return;
}


/***************
funcao que faz a leitura do sensor 1 por 5s

-mode :
 0 para leitura parada
 1 para leitura em movimento
***************/
void time_sensoring(int mode)
{
    int pid;
    printf("Iniciando a leitura dos sensores\n");

    if(mode)
      MOVE(POTENCIA_RETA_ME, POTENCIA_RETA_MD);

    pid = start_process(get_dados(SENSOR_PORT1,dados, DELTA_DADOS, 100L));
    sleep(5.0);
    kill_process(pid);
    ao();

    return;
}

/***************
***************/
void read_dados()
{
    int i;
    for(i=0; i<DELTA_DADOS; i++)
      {
        printf("%d - %d\n",i, dados[i]);
        while(!stop_button() && !start_button());
        while(stop_button() || start_button());
    }
}
```

```
/***************
funcao que faz a calibragem
dos sensores
***************/
void calibrate_sensores()
{
    int temp[DELTA_CALIB];
    printf("Start => calibrar preto\n");

    while(!start_button());
    while(start_button());

    printf("Calibrando sensor preto.\n");

    get_dados(SENSOR_PORT1, temp, DELTA_CALIB, 100L);
    quicksort(temp, DELTA_CALIB);
    preto = temp[(DELTA_CALIB/2)];

    printf("Start => calibrar branco\n");

    while(!start_button());
    while(start_button());

    printf("Calibrando sensor branco.\n");
    get_dados(SENSOR_PORT1, temp, DELTA_CALIB, 100L);
    quicksort(temp, DELTA_CALIB);
    branco = temp[(DELTA_CALIB/2)];

    midline = (preto + branco) / 2;

    return;
}

/***************
funcao que mostra o resultado
da calibragem dos sensores
***************/
void show_calibration()
{
    printf("Prt: %d Brc:%d MdL:%d\n", preto, branco, midline);
    while(!stop_button());
    return;
}

void menu_principal()
{
    int numero_opcoes = 10; //Numero de opcoes do menu
    int escolha = -1;

    while(1)
      {
        //menu de escolhas
        if(escolha == -1)
          printf("##Menu principal##\n");
        else if(escolha == 0)
          printf("##Calibramento dos blocos\n");
        else if(escolha ==1)
          printf("##Identifica bloco\n");
```

```
        else if(escolha == 2)
          printf("##Calibramento dos sensores p/b\n");
        else if(escolha == 3)
          printf("##Ver resultado calibracao p/b\n");
        else if(escolha ==4)
          printf("##Segue linha 01\n");
        else if(escolha ==5)
          printf("##Segue linha 02\n");
        else if(escolha == 6)
          printf("##Ler dados armazenados\n");
        else if(escolha == 7)
          printf("##Leitura do sensor em movimento\n");
        else if(escolha == 8)
          printf("##Leitura do sensor parado\n");
        else if(escolha == 9)
          printf("##Sensor em tempo real\n");

        //while para travar a impressao do menu
        while(!stop_button() && !start_button());

        //tratamento dos butoes
        if (stop_button())
          {
            escolha = (escolha  + 1) % numero_opcoes;
        }
        else if (start_button())
          {
            while(start_button());

            if(escolha == 0)
              calibrate_blocos();
            else if(escolha == 1)
              identify_bloco();
            else if(escolha == 2)
              calibrate_sensores();
            else if(escolha == 3)
              show_calibration();
            else if(escolha == 4)
              run_online(1);
            else if(escolha == 5)
              run_online(2);
            else if(escolha == 6)
              read_dados();
            else if(escolha == 7)
              time_sensoring(1);
            else if(escolha == 8)
              time_sensoring(0);
            else if(escolha == 9)
              sensor_real_time();


        }

        while(stop_button() || start_button());
    }
}

int main()
```

```
{
    printf("###GRUPO TTF###\n");
    sleep(1.0);
    menu_principal();
    return 0;
}
```