Anexo II – Circuito do *Driver* da torre de vida elétrica

Devido às dimensões do circuito, a sua colocação nesta página, a imagem perderia resolução e ficaria imperceptivel. Deste modo o esquema do circuito será fornecido no cd que vem incluido no relatório.

Anexo III - Código

CÓDIGO MECHANICAL LIFE TESTER EM TEMPERATURA

```
/*

V9 A ler, comunicar e limpar epprom

*/
#include <TimerOne.h>

#include <EEPROM.h>
#include <TFT.h>
#include <SPI.h>


#define lcd_cs   12
#define dc   13
#define rst  27
//#define sd_cs 50

#define Q1 49  //mosfet 1 - antes era 52, alterado por conflito com SPI
#define Q3 48  //mosfet 3
#define Q5 44  //mosfet 5
#define Q7 40  //mosfet 7
#define Q9 36  //mosfet 9

#define chamber 26 //input
#define PATAMAR 60 //tempo minimo entre ciclos de temperatura

//SimpleTimer timer;


int go = 0;
int mode = 0;
int mode_old = 0;

String Status[4] = {"Running for","Stopped at","Ready to COM","Clearing EEPROM"};
int incomingByte = 0;   // for incoming serial data

// create an instance of the library
TFT TFTscreen = TFT(lcd_cs, dc, rst);
bool lcd_reset = 0;
const int sensor = A0;
int B_start = 22;
int B_comm = 23;

int dut[20] = {14,16,18,20,7,5,3,1,8,10,11,9,0,2,4,6,21,19,17,15};
int fails_sh[20] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
int fails[20] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};


int ticks = 0;
int seconds = 0;
int seconds_old = 0;
int minutes = 0;
int minutes_old = 0;
long hours = 0;
long hours_old = 0;

int temperature = 0;
```

```
int temp_old = 0;

unsigned int failure = 0;
unsigned long int cycles_old = 0;
unsigned long int cycles = 0;
unsigned long int last_ch = 0;
unsigned long int min_est = 0;
unsigned int sec_est = 0;
unsigned int tcycle = 0;
unsigned int tcycle_old = 0;

// char array to print to the screen
char Printcycles[9];
char Printtcycle[9];
char Printtemperature[4];
char Printfailure[2];

void setup() {
  // put your setup code here, to run once:

  //Serial.begin(9600);

  pinMode(B_start, INPUT_PULLUP);
  pinMode(B_comm, INPUT_PULLUP);
  pinMode(chamber, INPUT_PULLUP);

  int i = 0;

  for(i=0;i<20;i++)  //declarar entradas
    pinMode(dut[i], INPUT_PULLUP);

  pinMode(Q3, OUTPUT);
  pinMode(Q5, OUTPUT);
  pinMode(Q7, OUTPUT);
  pinMode(Q9, OUTPUT);


  setup_tft();

  Serial.println("Setup done");

  Timer1.initialize(1000000); //inicializar o timer com 1
  Timer1.attachInterrupt(tick); //função = interrupção de serviço ao timer é a tick()

  check_fails();
}// end setup
////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////
//////////

void loop() {

  //temperature = (5.0 * analogRead(sensor) * 100.0) / 512;      //code for debug
  //getting the voltage reading from the temperature sensor      //code for debug

  read_temp();
  int buttonStart = digitalRead(B_start);
  int buttonComm = digitalRead(B_comm);

  if (Serial.available() > 0)
  {
      // read the incoming byte:

      incomingByte = Serial.read(); //recebendo da Serial um "r", faz um dump da EEPROM
      if(incomingByte == 114)
        read_eeprom();

      if(incomingByte == 99) //recebendo da Serial um "c", faz um clear da EEPROM
```

```
      clear_eeprom();

    if(incomingByte == 122) //recebendo da Serial um "z", coloca a zero os ciclos e o tempo
      {
      cycles=0;
      seconds = 0;
      seconds_old = 0;
      minutes = 0;
      minutes_old = 0;
      hours = 0;
      hours_old = 0;
      Serial.println("Zerado");
      }
}

  if((digitalRead(chamber) == LOW) && ((min_est-last_ch)>PATAMAR))
  {
    tcycle++;
    last_ch = min_est;
    EEPROM.write(4030, tcycle);
  }


  printLCD();

  if (buttonStart == LOW)
  {
    mode = 1;
    LCD_status();
    lcd_reset = 0;
    //t.update();
   }
  else
  {
    if(buttonComm == HIGH)
    {
      mode = 2;
      Serial.end();
      LCD_status();


    }
  }

  if (buttonComm == LOW)
  {
    mode = 3;
    LCD_status();
    lcd_reset = 0;

    Serial.begin(9600);
  }
   else
   {
     if(buttonStart == HIGH)
     {
       mode = 2;
       Serial.end();
       LCD_status();
     }
   }

   if (mode != 1){
       coils_off();
       ticks = 0;
     }

    if (mode == 2)
```

```
      {
            if(lcd_reset == 0)
            {
               setup_tft();
               lcd_reset = 1;
           }
      }


}// end loop
////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////
//////////

void tick ()
{
    sec_est++;

    if (sec_est % 60 == 0)
    {
      min_est++;
      sec_est = 0;
    }

  if (mode == 1)
  {
    ticks++;
    seconds++;

    if (seconds % 60 == 0)
    {
      minutes++;
      seconds = 0;
    }
    if ((minutes > 0) && (minutes % 60 == 0))
    {
      hours++;
      minutes = 0;
    }

    check_fails();
  }
    /*Serial.print("Time ");                    //code for debug
    Serial.print(hours);                       //code for debug
    Serial.print(minutes);                     //code for debug
    Serial.println(seconds);                   //code for debug
    */

   if (mode == 1)
   {

    switch(ticks)
    {
      case 1:
        coils_on();
          cycles++;
        break;
      case 2:
        check_on();
        break;
      case 3:
        coils_off();
        break;
      case 4:
        check_off();
        ticks = 0;
        break;
      default:
```

```
        break;
    }

  }

} // end tick()
//////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////////////////////////
//////////

void coils_on ()
{
  digitalWrite(Q1, HIGH);
  digitalWrite(Q3, HIGH);
  digitalWrite(Q5, HIGH);
  digitalWrite(Q7, HIGH);
  digitalWrite(Q9, HIGH);

  //Serial.println("coils ON");                  //code for debug
} // end coils_on()
//////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////////////////////////
//////////

void check_on ()
{
  int i4 = 0;

  //Serial.println("contacts ON?");              //code for debug

  for(i4=0;i4<20;i4++)
  {
    if(digitalRead(dut[i4]) && (fails[i4] < 99))
    {
      //Serial.print("Ton = ");                   //code for debug
      //Serial.println(temperature);              //code for debug
      if(mode == 1)
        write_eeprom(i4+1,cycles,1,temperature);

      //fails_sh[i4]=fails[i4]; //temporaryyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy
      //fails[i4]++; //temporaryyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy
      /*Serial.print("Shadows ON");              //code for debug
      Serial.println(fails_sh[i4]);            //code for debug
      Serial.print("Actual ON");               //code for debug
      Serial.println(fails[i4]);*/             //code for debug
      //Serial.print("Fail to close relay ");    //code for debug
      //Serial.println(i4+1);                    //code for debug
    }
  }
}// end check_on()
//////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////////////////////////
//////////

void coils_off ()
{
  digitalWrite(Q1, LOW);
  digitalWrite(Q3, LOW);
  digitalWrite(Q5, LOW);
  digitalWrite(Q7, LOW);
  digitalWrite(Q9, LOW);

  //Serial.println("coils Off");              //code for debug
} // end coils_off()
//////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////////////////////////
//////////
```

```
void check_off ()
{
  int i4 = 0;
  //Serial.println("contacts OFF?");          //code for debug

  for(i4=0;i4<20;i4++)
  {
    if(!digitalRead(dut[i4]) && (fails[i4] < 99))
    {
      //Serial.print("Toff = ");              //code for debug
      //Serial.println(temperature);          //code for debug
      if(mode==1)
       write_eeprom(i4+1,cycles,2,temperature);

      //fails_sh[i4]=fails[i4]; //temporaryyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy
      //fails[i4]++; //temporaryyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy
     /* Serial.print("Shadows OFF");          //code for debug
      Serial.println(fails_sh[i*j-1]);        //code for debug
      Serial.print("Actual OFF");              //code for debug
      Serial.println(fails[i*j-1]);*/          //code for debug
      //Serial.print("Fail to open relay ");    //code for debug
      //Serial.println(i4+1);                   //code for debug
    }
  }
} // end check_off()
///////////////////////////////////////////////////////////////////////////////////////////////
///////////////////////////////////////////////////////////////////////////////////////////////
//////////

void write_eeprom (int relay, long int cycle, int type, int temp)
{
  int fails = EEPROM.read(4000+relay);

  if(fails < 10)
  {
    int pos = relay*100 + fails*10;
    int b3 = cycle;
    b3 &= 0xFF;
    int b2 = cycle >> 8;
    b2 &= 0xFF;
    int b1 = cycle >> 16;
    b1 &= 0xFF;

    EEPROM.write(pos, b1);
    EEPROM.write(pos+1, b2);
    EEPROM.write(pos+2, b3);
    EEPROM.write(pos+3, type);
    EEPROM.write(pos+4, temp);
    EEPROM.write(pos+5, tcycle);
  }

  EEPROM.write(4000+relay, fails+1);

} // end write_eeprom (int relay, long int cycle, int type, int temp)
///////////////////////////////////////////////////////////////////////////////////////////////
///////////////////////////////////////////////////////////////////////////////////////////////
//////////

void read_eeprom ()
{
  long int fail;
  int i=1;
  int j=0;
  long int b3 = 0;
  long int b2 = 0;
  long int b1 = 0;
  int pos = 0;
  int temp = 0;
```

```
  int type = 0;
  int r_tcycle = 0;

  for(i=0;i<20;i++)
  {
    Serial.print("R# ");
    Serial.println(i+1);
    Serial.print("Failures:");
    int validate = EEPROM.read(4000+i+1);
    Serial.println(validate);

    if (validate > 10)
      validate = 10;

    if(validate!=0)
    for(j=0;j<validate;j++)
    {
      pos = (i+1)*100+j*10;
      b1 = EEPROM.read(pos);
      b2 = EEPROM.read(pos+1);
      b3 = EEPROM.read(pos+2);
      type = EEPROM.read(pos+3);
      temp = EEPROM.read(pos+4);
      r_tcycle = EEPROM.read(pos+5);

      b1 = b1 << 16;
      b2 = b2 << 8;

      fail = b1 + b2 + b3;
      Serial.print(fail);
      Serial.print("\t");
      Serial.print(type);
      Serial.print("\t");
      Serial.print(temp);
      Serial.print("\t");
      Serial.println(r_tcycle);
      }
  }
} // end read_eeprom ()
////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////
//////////

void clear_eeprom ()
{
  long int i;
  float aux=0;
  mode = 4;
  LCD_status();

  for(i=0;i<4095;i++)
  {
      aux=i;
      Serial.println(aux/4095);
      EEPROM.write(i,0);
  }
  Serial.println("EEPROM cleared");

  mode = 3;
  LCD_status();

  check_fails ();
  cycles = 0;
  tcycle = 0;
  seconds = 0;
  seconds_old = 0;
  minutes = 0;
  minutes_old = 0;
```

```
  hours = 0;
  hours_old = 0;
  printLCD ();
} // end clear_eeprom ()
////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////
//////////


void check_fails ()
{
  int i = 0;
  for(i=0;i<20;i++)
  {
    fails_sh[i]=fails[i];
    fails[i]=EEPROM.read(4000+i+1);
      /*Serial.print("Relay # ");        //code for debug
    Serial.print(i+1);                  //code for debug
    Serial.print(" - Falhas:");         //code for debug
    Serial.println(fails[i]);*/        //code for debug
    }
} // end check_fails ()
////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////
//////////


void printLCD (){


  int i = 0;
  int j = 0;

  char aux[4];

  if(temp_old == 0)
      temp_old = temperature;

  //delay(100);

  //TFTscreen.background(0, 0, 0);

  String Val=String(cycles_old);
  Val.toCharArray(Printcycles, 8);

  String Val2= String(tcycle_old);
  Val2.toCharArray(Printtcycle, 4);

  read_temp();

  String Val1= String(temp_old);
  Val1.toCharArray(Printtemperature, 4);

  TFTscreen.stroke(0, 0, 0);  // set the font color
  TFTscreen.setTextSize(1);  //set font size
  TFTscreen.text(Printcycles, 0, 13); //print counter cycles
  TFTscreen.text(Printtcycle, 50, 13); //print counter temperature cycles
  TFTscreen.setTextSize(2);  //set font size
  TFTscreen.text(Printtemperature, 100, 13); //print temperature

  // wait for a moment
//  delay(100);
  // erase the text you just wrote

  Val1= String(temperature);
  Val1.toCharArray(Printtemperature, 4);
  Val=String(cycles);
  Val.toCharArray(Printcycles, 8);
```

```
  Val2= String(tcycle);
  Val2.toCharArray(Printtcycle, 4);

  TFTscreen.stroke(255, 255, 255);
  TFTscreen.setTextSize(1);  //set font size
  TFTscreen.text(Printcycles, 0, 13);
  TFTscreen.text(Printtcycle, 50, 13); //print counter temperature cycles
  TFTscreen.setTextSize(2);  //set font size
  TFTscreen.text(Printtemperature, 100, 13);

  temp_old = temperature;
  cycles_old = cycles;
  tcycle_old = tcycle;

  TFTscreen.stroke(255, 255, 255);
  TFTscreen.setTextSize(1);
  // write the text to the top left corner of the screen
  TFTscreen.text("___________________________", 0, 24);
  TFTscreen.text(" CH  F  CH  F CH  F  CH  F", 0, 35);
  TFTscreen.text("___________________________", 0, 38);

  TFTscreen.text(" 1      2     3      4   ", 0, 50);
  TFTscreen.text(" 5      6     7      8  ", 0, 65);
  TFTscreen.text(" 9      10    11     12  ", 0, 80);
  TFTscreen.text(" 13     14    15     16  ", 0, 95);
  TFTscreen.text(" 17     18    19     20  ", 0, 110);

  //fails[12] = 0;
//aldrabiceeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee

  for(i=1;i<6;i++) //linhas
  {
    for(j=1;j<5;j++) //colunas
    {
      //Serial.println(-5+i*5+j-1);         //code for debug
      TFTscreen.stroke(0, 0, 0); //escrever em preto os vals anteriores de falhas
      //String aux_string= String(-4+i*4+j);
      String aux_string = String(fails_sh[-4+i*4+j-1]);
      aux_string.toCharArray(Printfailure, 3);
      TFTscreen.text(Printfailure,-15+j*40,35+i*15);

      TFTscreen.stroke(255, 0, 0); //escrever em vermelho os vals actuais de falhas
      //aux_string= String(-4+i*4+j);
      aux_string = String(fails[-4+i*4+j-1]);
      aux_string.toCharArray(Printfailure, 3);
      TFTscreen.text(Printfailure,-15+j*40,35+i*15);

      /*Serial.print("Falhas relé nº ");   //code for debug
      Serial.print(-5+i*5+j-1);            //code for debug
      Serial.print(" ");                   //code for debug
      Serial.print(fails_sh[-5+i*5+j-1]);  //code for debug
      Serial.print(" - ");                 //code for debug
      Serial.println(fails[-5+i*5+j-1]);*/ //code for debug
    }
  }

  LCD_status();

}/// end print_LCD ()
/////////////////////////////////////////////////////////////////////////////////////////////
/////////////////////////////////////////////////////////////////////////////////////////////
//////////


void setup_tft ()
{
    // Put this line at the beginning of every sketch that uses the GLCD:
  TFTscreen.begin();
```

```
  // clear the screen with a black background
  TFTscreen.background(0, 0, 0);

  // write the static text to the screen
  // set the font color to white
  TFTscreen.stroke(255, 255, 255);
  // set the font size
  TFTscreen.setTextSize(1);
  // write the text to the top left corner of the screen
  TFTscreen.text("Ecycle  Tcycle  Temp\n ", 0, 0);

  TFTscreen.setTextSize(2);
  TFTscreen.text(" C", 135, 13);
  // set the font size very large for the loop

  /*
    // set the font size

  TFTscreen.setTextSize(1);
  // write the text to the top left corner of the screen
  TFTscreen.text("_____________________________", 0, 24);
  TFTscreen.text("CH F   CH F   CH F   CH F", 0, 35);
  TFTscreen.text("_____________________________", 0, 38);

  TFTscreen.text("1      6  F   11 F   16 F", 0, 50);
  TFTscreen.text("2      7  F   12 F   17 F", 0, 65);
  TFTscreen.text("3  F   8  F   13 F   18 F", 0, 80);
  TFTscreen.text("4  F   9  F   14 F   19 F", 0, 95);
  TFTscreen.text("5  F   10 F   15 F   20 F", 0, 110);
  */
} // end setup_tft ()
////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////
//////////


void read_temp (void)
{
   int reading = analogRead(sensor);

   // converting that reading to voltage, for 3.3v arduino use 3.3
   float voltage = reading * 5.0;
   voltage /= 1024.0;

   temperature = (voltage - 0.5) * 100 ;  //converting from 10 mv per degree wit 500 mV offset
                                          //to degrees ((voltage - 500mV) times 100)

   if (abs(temperature - temp_old) < 1)
     temperature = temp_old;

} // end read_temp ()
////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////
//////////

void LCD_status ()
{



  char aux[20];
  String Val;

  switch(mode)
  {
    case 1:
        TFTscreen.setTextSize(1);
```

```
    TFTscreen.stroke(0, 0, 0);
    Status[mode_old-1].toCharArray(aux, 15);
    TFTscreen.text(aux, 0, 120);
    Val=String(hours_old);
    Val.toCharArray(aux, 5);
    TFTscreen.text(aux, 80, 120);
    TFTscreen.text(":", 105, 120);
    Val=String(minutes_old);
    Val.toCharArray(aux, 3);
    TFTscreen.text(aux, 110, 120);
    TFTscreen.text(":", 120, 120);
    Val=String(seconds_old);
    Val.toCharArray(aux, 3);
    TFTscreen.text(aux, 125, 120);

    TFTscreen.stroke(255, 255, 255);
    TFTscreen.text("Running for ", 0, 120);
    Val=String(hours);
    Val.toCharArray(aux, 5);
    TFTscreen.text(aux, 80, 120);
    TFTscreen.text(":", 105, 120);
    Val=String(minutes);
    Val.toCharArray(aux, 3);
    TFTscreen.text(aux, 110, 120);
    TFTscreen.text(":", 120, 120);
    Val=String(seconds);
    Val.toCharArray(aux, 3);
    TFTscreen.text(aux, 125, 120);

    seconds_old = seconds;
    minutes_old = minutes;
    hours_old = hours;
    break;
case 2:
    TFTscreen.setTextSize(1);
    TFTscreen.stroke(0, 0, 0);
    Status[mode_old-1].toCharArray(aux, 15);
    TFTscreen.text(aux, 0, 120);
    if(mode_old == 1)
    {
      Val=String(hours_old);
      Val.toCharArray(aux, 5);
      TFTscreen.text(aux, 80, 120);
      TFTscreen.text(":", 105, 120);
      Val=String(minutes_old);
      Val.toCharArray(aux, 3);
      TFTscreen.text(aux, 110, 120);
      TFTscreen.text(":", 120, 120);
      Val=String(seconds_old);
      Val.toCharArray(aux, 3);
      TFTscreen.text(aux, 125, 120);
    }

    TFTscreen.stroke(255, 255, 255);
    TFTscreen.text("Stopped at ", 0, 120);
    Val=String(hours);
    Val.toCharArray(aux, 5);
    TFTscreen.text(aux, 80, 120);
    TFTscreen.text(":", 105, 120);
    Val=String(minutes);
    Val.toCharArray(aux, 3);
    TFTscreen.text(aux, 110, 120);
    TFTscreen.text(":", 120, 120);
    Val=String(seconds);
    Val.toCharArray(aux, 3);
    TFTscreen.text(aux, 125, 120);
  break;
case 3:
```

```
      TFTscreen.setTextSize(1);
      TFTscreen.stroke(0, 0, 0);
      Status[mode_old-1].toCharArray(aux, 20);
      TFTscreen.text(aux, 0, 120);
      if(mode_old != 3)
      {
        Val=String(hours_old);
        Val.toCharArray(aux, 5);
        TFTscreen.text(aux, 80, 120);
        TFTscreen.text(":", 105, 120);
        Val=String(minutes_old);
        Val.toCharArray(aux, 3);
        TFTscreen.text(aux, 110, 120);
        TFTscreen.text(":", 120, 120);
        Val=String(seconds_old);
        Val.toCharArray(aux, 3);
        TFTscreen.text(aux, 125, 120);
      }

      TFTscreen.stroke(255, 255, 255);
      TFTscreen.text("Ready to COM     ", 0, 120);
      break;

      case 4:
      TFTscreen.setTextSize(1);
      TFTscreen.stroke(0, 0, 0);
      Status[mode_old-1].toCharArray(aux, 15);
      TFTscreen.text(aux, 0, 120);
      if(mode_old != 3)
      {
        Val=String(hours_old);
        Val.toCharArray(aux, 5);
        TFTscreen.text(aux, 80, 120);
        TFTscreen.text(":", 105, 120);
        Val=String(minutes_old);
        Val.toCharArray(aux, 3);
        TFTscreen.text(aux, 110, 120);
        TFTscreen.text(":", 120, 120);
        Val=String(seconds_old);
        Val.toCharArray(aux, 3);
        TFTscreen.text(aux, 125, 120);
      }

      TFTscreen.stroke(255, 255, 255);
      TFTscreen.text("Clearing EEPROM", 0, 120);
      break;

    default:
      break;
   }

   mode_old = mode;
}
```

CÓDIGO DO DRIVER PARA RELÉS AUXILIARES EM TESTES DE VIDA ELÉTRICA

```
#include <DueFlashStorage.h>
DueFlashStorage dueFlashStorage;

#include <TFT.h>  // Arduino LCD library
#include <SPI.h>
//#include <EEPROM.h>

#define cs   13 //10
#define dc   12 //9
#define rst  11 //8

#define wait 200
#define col1 0
#define col2 40
#define col3 80
#define col4 120

#define b_dn 22
#define b_up 24
#define b_ls 26
#define b_pl 28

// create an instance of the library
TFT TFTscreen = TFT(cs, dc, rst);

// char array to print to the screen
char sensorPrintout[4];
int incomingByte = 0;

int mode = 0;
int ch_st = HIGH;
int ch_cyc = HIGH;
int last_change = 0;
int ed = 0;
int cur_val = 0;

unsigned long pisca = 0;
unsigned long last_ch = 0;
int state = LOW;

int in[8] = {2,3,4,5,6,7,8,9};
int out[8] = {14,15,16,17,18,19,20,21};
int st[8] = {0,0,0,0,0,0,0,0};
int st_o[8] = {0,0,0,0,0,0,0,0};
int wu[8] = {0,0,0,0,0,0,0,0};
int wd[8] = {0,0,0,0,0,0,0,0};
unsigned long c[8] = {0,0,0,0,0,0,0,0};
unsigned long c_o[8] = {0,0,0,0,0,0,0,0};
unsigned long t[16] = {500,500,500,500,500,500,500,500,500,500,500,500,500,500,500,500};

unsigned long toff[8] = {0,0,0,0,0,0,0,0};
unsigned long ton[8] = {0,0,0,0,0,0,0,0};


void setup() {

  int i = 0;
  String auxs;
  char auxc[4];
  int line = 40;
```

```
for(i=0;i<8;i++)
{
  pinMode(in[i], INPUT_PULLUP);
}

for(i=0;i<8;i++)
{
  pinMode(out[i], OUTPUT);
  digitalWrite(out[i], LOW);
}

pinMode(b_up, INPUT_PULLUP);
pinMode(b_dn, INPUT_PULLUP);
pinMode(b_pl, INPUT_PULLUP);
pinMode(b_ls, INPUT_PULLUP);
delay(200);

Serial.begin(9600);

attachInterrupt(in[0], reg_ch1, CHANGE);
attachInterrupt(in[1], reg_ch2, CHANGE);
attachInterrupt(in[2], reg_ch3, CHANGE);
attachInterrupt(in[3], reg_ch4, CHANGE);
attachInterrupt(in[4], reg_ch5, CHANGE);
attachInterrupt(in[5], reg_ch6, CHANGE);
attachInterrupt(in[6], reg_ch7, CHANGE);
attachInterrupt(in[7], reg_ch8, CHANGE);

attachInterrupt(b_up, up, FALLING);
attachInterrupt(b_dn, down, FALLING);
attachInterrupt(b_pl, plus, FALLING);
attachInterrupt(b_ls, less, FALLING);

//pinMode(52, OUTPUT);
//digitalWrite(52, LOW);

// Put this line at the beginning of every sketch that uses the GLCD:
TFTscreen.begin();

// clear the screen with a white background
TFTscreen.background(255, 255, 255);

Serial.println("MANUAL");
while(mode == 0)
{
  if (Serial.available() > 0)
  {
    Serial.println("Clear ask");
    incomingByte = Serial.read();

    if(incomingByte == 99) //recebendo da Serial um "c", faz um clear da EEPROM
      clear_eeprom();
  }

  manual_screen();
}

TFTscreen.background(255, 255, 255);

TFTscreen.stroke(0,0,255);
for(i=1;i<9;i++)
{
  auxs = String(i);
  auxs.toCharArray(auxc, 5);
  TFTscreen.text(auxc, 0, line);
  line += 10;
}
```

```
  delay_screen();
  cycles_show();
  set_delays();

  detachInterrupt(b_up);
  detachInterrupt(b_dn);
  detachInterrupt(b_pl);
  detachInterrupt(b_ls);

  Serial.println("SETUP DONE");

  TFTscreen.stroke(0,255,0);
  TFTscreen.text("RUNNING", 50, 120);
}

void set_delays ()
{
  String auxs;
  char auxc[4];
  int i = 0;
  int col = 0;
  int line = 40;
  char auxi[2];
  int b = 0;

  Serial.println("SETTING DELAYS");

  //for(b=0;b<8;b++)
    //c[b] = dueFlashStorage.read(b)*10000;

 while(ed < 16)
  {
    if (Serial.available() > 0)
    {
      Serial.println("Clear ask");
      incomingByte = Serial.read();

      if(incomingByte == 99) //recebendo da Serial um "c", faz um clear da EEPROM
        clear_eeprom();
    }

    if (ed & 1)
      col = col3;
    else
      col = col2;

    TFTscreen.stroke(255,0,0);
    auxs = String(t[ed]);
    auxs.toCharArray(auxc, 5);
    TFTscreen.text(auxc, col, 40+(ed/2)*10);

    delay(500);
    }

}

void manual_screen ()
{
  String auxs;
  char auxc[4];
  int i = 0;
  int line_m = 40;
  char auxi[2];

  TFTscreen.stroke(0,0,255);
  // set the font size
  TFTscreen.setTextSize(1);
```

```
  TFTscreen.text("TE Aux Relay Driver \n", 25, 2);
  TFTscreen.text("+ to ON, - to OFF \n", 0, 20);
  TFTscreen.text("_____________________ \n", 0, 30);

  for(i=1;i<9;i++)
  {
    TFTscreen.stroke(0,0,255);
    auxs = String(i);
    auxs.toCharArray(auxc, 5);
    TFTscreen.text(auxc, 0, line_m);

    if(st[i-1] != st_o[i-1])
    {
      TFTscreen.stroke(255,255,255);
      if(st_o[i-1] == 1)
        TFTscreen.text("ON", 40, line_m);
      else
        TFTscreen.text("OFF", 40, line_m);
    }

    if (cur_val == (i-1))
      TFTscreen.stroke(255,0,0);
    else
      TFTscreen.stroke(0,0,255);

    if(st[i-1] == 1)
      TFTscreen.text("ON", 40, line_m);
    else
      TFTscreen.text("OFF", 40, line_m);

    line_m += 10;
    st_o[i-1]=st[i-1];

    }

}

void delay_screen ()
{
  String auxs;
  char auxc[4];
  int i = 0;
  int line = 40;
  char auxi[2];
  long c_aux = 0;

  TFTscreen.stroke(0,0,255);
  // set the font size
  TFTscreen.setTextSize(1);

  TFTscreen.text("TE Aux Relay Driver \n", 25, 2);
  TFTscreen.text("CH     Doff  Don    Cycles\n", 0, 20);
  TFTscreen.text("_____________________________\n", 0, 30);

  for(i=1;i<9;i++)
  {
    TFTscreen.stroke(0,0,255);

    /*auxs = String(i);
    auxs.toCharArray(auxc, 5);
    TFTscreen.text(auxc, 0, line);*/

    auxs = String(t[2*i-2]);
    auxs.toCharArray(auxc, 4);
    TFTscreen.text(auxc, col2, line);

    auxs = String(t[2*i-1]);
```

```
    auxs.toCharArray(auxc, 4);
    TFTscreen.text(auxc, col3, line);

    line += 10;
  }
}

void cycles_show ()
{

  int i = 0;
  long c_aux = 0;
  String auxs;
  char auxc[4];
  int line = 40;
  int cyc_chars = 0;

  TFTscreen.stroke(0,0,255);

  for(i=1;i<9;i++)
  {

    c_aux = dueFlashStorage.read(i-1);

    if(c_aux != c_o[i-1])
    {
      TFTscreen.stroke(255,255,255);
      auxs = String(c_o[i-1]);
      auxs.toCharArray(auxc, 7);
      cyc_chars = auxs.length();
      TFTscreen.text(auxc, col4 , line);
      TFTscreen.text("0k", col4+cyc_chars*6, line);
    }

    TFTscreen.stroke(0,0,255);
    auxs = String(c_aux);
    auxs.toCharArray(auxc, 7);
    cyc_chars = auxs.length();
    TFTscreen.text(auxc, col4 , line);
    TFTscreen.text("0k", col4+cyc_chars*6, line);

    line += 10;
    c_o[i-1]=c_aux;

  }
}

void loop() {

  mode = 2;

  int line = 40;
  String auxs;
  char auxc[4];
  long c_aux = 0;
  int i = 0;
  unsigned long time = millis();

  if (Serial.available() > 0)
  {
    //Serial.println("COM ASK");

    incomingByte = Serial.read();
    if(incomingByte == 99) //recebendo da Serial um "c", faz um clear da EEPROM
      clear_eeprom();
  }

  /*
```

```
  if(time - pisca > 1000)
    {
      pisca = time;
      if (state == LOW)
        state = HIGH;
      else
        state = LOW;

      c[1] += 1;
      c[2] += 2;

      digitalWrite(52, state);
    }
  */

  for(i=0;i<8;i++)
  {

    if((time > (toff[i] + t[2*i])) && (wu[i] == HIGH))
    {
      st[i] = HIGH;
      wu[i] = LOW;

      c[i]++;
      ch_st = HIGH;

      c_aux = dueFlashStorage.read(i);
      if ((c[i] % 10000) == 0)
      {
        Serial.println("Got one");
        dueFlashStorage.write(i,c_aux+1);
        ch_cyc = HIGH;
      }

      /*Serial.print(i+1);
      Serial.print(" AUX ON: ");
      Serial.println(time);*/
    }

    if((time > (ton[i] + t[2*i+1])) && (wd[i] == HIGH))
    {
      st[i] = LOW;
      wd[i] = LOW;

      ch_st = HIGH;
      /*Serial.print(i+1);
      Serial.print(" AUX OFF: ");
      Serial.println(time);*/
    }

    digitalWrite(out[i], st[i]);

    if(ch_st)
    {
      auxs = String(i+1);
      auxs.toCharArray(auxc, 5);
      if(st[i]==HIGH)
        TFTscreen.stroke(0,255,0);
      else
        TFTscreen.stroke(255,0,0);
      TFTscreen.text(auxc, 0, line+10*i);
      ch_st = LOW;
    }

  }

  if(ch_cyc)
  {
```

```
    cycles_show();
    ch_cyc = LOW;
  }

}

void reg_ch (int n)
{
  n -= 1;

  if(millis() - last_ch > 0)
  {

  Serial.print(n+1);
  Serial.print(" TENTOU @ ");
  Serial.print(millis());
  Serial.print(" E ESTA ");
  Serial.println(digitalRead(in[n]));

    if(digitalRead(in[n]))
    {
      toff[n] = millis();
      wu[n] = HIGH;
      Serial.print(n+1);
      Serial.print(" FALL: ");
      Serial.println(toff[n]);
      }
    else
    {
      ton[n] = millis();
      wd[n] = HIGH;
      Serial.print(n+1);
      Serial.print(" RISE: ");
      Serial.println(ton[n]);
    }

    last_ch = millis();
  }
}

void reg_ch1 ()
{
  reg_ch(1);
}

void reg_ch2 ()
{
  reg_ch(2);
}

void reg_ch3 ()
{
  reg_ch(3);
}

void reg_ch4 ()
{
  reg_ch(4);
}

void reg_ch5 ()
{
  reg_ch(5);
}

void reg_ch6 ()
{
  reg_ch(6);
```

```
}

void reg_ch7 ()
{
  reg_ch(7);
}

void reg_ch8 ()
{
  reg_ch(8);
}

void down ()
{
  String auxs;
  char auxc[4];
  int col = 0;

  if ((millis()-last_change) > wait)
  {
    if (mode == 0)
    {
      if(cur_val<8)
        cur_val++;
      if (cur_val == 8)
        mode = 1;
      Serial.print("DOWN to ");
      Serial.println(cur_val);

    }
    else
    {
      if (ed< 16)
      {
        if (ed & 1)
          col = col3;
        else
          col = col2;

        TFTscreen.stroke(0,0,255);
        auxs = String(t[ed]);
        auxs.toCharArray(auxc, 5);
        TFTscreen.text(auxc, col, 40+(ed/2)*10);

        ed += 1;
        Serial.print("Editing delay #");
        Serial.println(ed);
      }
    }
    last_change = millis();
  }
}

void up ()
{
  String auxs;
  char auxc[4];
  int col = 0;

  if ((millis()-last_change) > wait)
  {
    if (mode == 0)
    {
      if(cur_val>0)
        cur_val--;
      Serial.print("UP to ");
      Serial.println(cur_val);
    }
```

```
    else
    {
      if(ed > 0)
      {
        if (ed & 1)
          col = col3;
        else
          col = col2;

        TFTscreen.stroke(0,0,255);
        auxs = String(t[ed]);
        auxs.toCharArray(auxc, 5);
        TFTscreen.text(auxc, col, 40+(ed/2)*10);

        ed -= 1;
        Serial.print("Editing delay #");
        Serial.println(ed);
      }
    }
  last_change = millis();
  }
}

void plus ()
{
  String auxs;
  char auxc[4];
  int col = 0;
  int line = 40 + cur_val*10;

  if ((millis()-last_change) > wait)
  {
    if (mode == 0)
    {
      Serial.print("MAN ON ");
      Serial.println(cur_val+1);
      st[cur_val] = HIGH;
      digitalWrite(out[cur_val], st[cur_val]);
    }
    else
    {
      if (ed & 1)
        col = col3;
      else
        col = col2;

      if((ed<16) && (ed>-1))
      {
        TFTscreen.stroke(255,255,255);
        auxs = String(t[ed]);
        auxs.toCharArray(auxc, 5);
        TFTscreen.text(auxc, col, 40+(ed/2)*10);

        Serial.print(ed);
        Serial.println(" PLUS");
        t[ed] += 100;
      }
    }
  last_change = millis();
  }

}

void less ()
{
  String auxs;
  char auxc[4];
  int col = 0;
```

```
  int line = 40 + cur_val*10;

  if ((millis()-last_change) > wait)
  {
    if (mode == 0)
      {
        Serial.print("MAN OFF ");
        Serial.println(cur_val+1);
        st[cur_val] = LOW;
        digitalWrite(out[cur_val], st[cur_val]);
      }
      else
      {
        if (ed & 1)
          col = col3;
        else
          col = col2;

        if((ed<16) && (ed>=0))
        {
          TFTscreen.stroke(255,255,255);
          auxs = String(t[ed]);
          auxs.toCharArray(auxc, 5);
          TFTscreen.text(auxc, col, 40+(ed/2)*10);

          Serial.print(ed);
          Serial.println(" LESS");
          t[ed] -= 100;
        }
    }
    last_change = millis();
  }

}

void clear_eeprom ()
{
  int a;
  int line = 40;
  String auxs;
  char auxc[4];
  int cyc_chars = 0;

  TFTscreen.stroke(255,255,255);

  for(a=0;a<8;a++)
  {
    c_o[a] = dueFlashStorage.read(a);
    dueFlashStorage.write(a,0);

    auxs = String(c_o[a]);
    auxs.toCharArray(auxc, 7);
    cyc_chars = auxs.length();
    TFTscreen.text(auxc, col4 , line);
    TFTscreen.text("0k", col4+cyc_chars*6, line);
    line += 10;
  }

  Serial.println("EEPROM cleared");
}
```