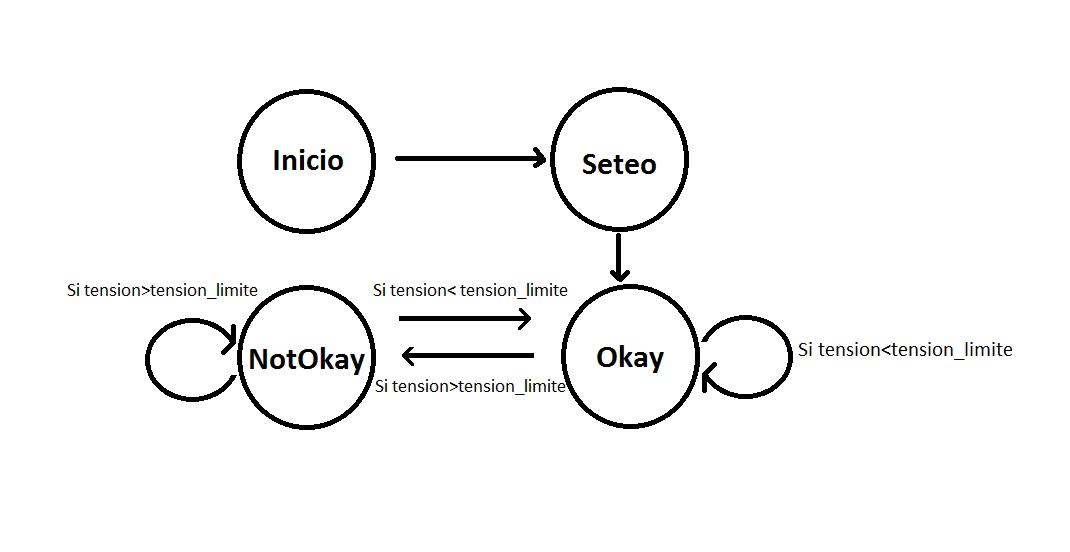
TP Informática Control de protección de sobre tensión

Memoria descriptiva

Funcionará de la siguiente manera: al medir la tensión del cable y en el caso de superar cierto valor se activara la protección e interrumpira la corriente, protegiendo así la instalación eléctrica del lugar, una vez que esto suceda en la instalación, se debe cambiar el DPS manualmente, en nuestra máquina de estado, esto corresponde a un estado donde la máquina no tiene corriente, y cuando se cambia el DPS, la máquina vuelve al estado inicial con corriente y protección inactiva.

Diagrama de estado

Librería

#ifndef LIBRERIA\_h

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

typedef enum {

    okay=0,

    notokay=1

}estados\_t;

typedef enum{

    cables=0,

    tension\_limite=1,

    tension=2,

    avisos=3

}claves\_t;

typedef struct {

 estados\_t cables;

 int tension\_limite;

 int tension;

 char avisos[70];

}datos\_t;

datos\_t f\_inicio(void);

estados\_t f\_espera(datos\_t);

estados\_t f\_desconexion(datos\_t);

int f\_medir\_tension();

estados\_t f\_estados(int, int );

char \*obtenerclave(char \*clave);

#endif

Configuraciones

/\*No se usa este archivo \*/

/\*cargo el estado actual de los cables en 1, que significa que esta bien hasta el momento\*/

cables okay

/\*  se activa la proteccion se establece como 1800 volts \*/

tension\_limite 1800

/\* Inventamos una tension actual medida \*/

tension 60

/\*Avisos dira que esta todo bien \*/

Funciones

#include "...\libreria\LIBRERIA\_h.h"

estados\_t f\_espera(datos\_t config){

    config.tension=f\_medir\_tension();

    config.cables=f\_estados(config.tension, config.tension\_limite);

    return config.cables;

}

int f\_medir\_tension(){

    int tension;

    printf("\nTension: ");

    scanf("%d",&tension);

    return tension;

}

estados\_t f\_estados(int tension, int tension\_limite)

{

    estados\_t estado;

    if((tension> tension\_limite)){

        estado=notokay;

        return estado;

    }else

        estado=okay;

        return estado;

}

estados\_t f\_desconexion(datos\_t config){

    printf("\nPor seguridad de la instalacion electrica se interrumpe la corriente");

    config.tension=f\_medir\_tension();

    config.cables=f\_estados(config.tension, config.tension\_limite);

    if(config.cables==okay)

    {

        return okay;

    }else{

        return notokay;

    }

}

datos\_t f\_inicio() {

    FILE \*config;

    char cadena[70], \*clave, \*valo;

    char variables[4][50]={"cables","tension\_limite","tension","avisos"}, i;

    datos\_t configuracion;

    claves\_t claves;

    if(!(config=fopen("Config.txt","rb"))){

        printf("\nNo se encontro el archivo\n");

     }

     fgets(cadena,70,config);

     do{

        clave=cadena; /\*  char clave apunta en la direccion de cadena \*/

        if(\*(clave) != '/' && strlen(clave)>=0){ /\* Mientras que la linea del archivo de config sea distinto de '/' y sea mayor que 0 este lee el nombre de la variable en clave \*/

            valo=obtenerclave(clave);

            for(i=0;i<4;i++)

            {

                if(!strcmp(clave, variables[i])){ // Ac� es cuando compara

                    switch(i)

                    {

                        case cables: configuracion.cables=atoi(valo); // Guarda los datos en la estructura

                                break;

                        case tension\_limite: configuracion.tension\_limite=atoi(valo);

                                break;

                        case tension: configuracion.tension=atoi(valo);

                                break;

                        case avisos: strcpy(configuracion.avisos,valo);

                                break;

                     }

                 }

             }

         }

         fgets(cadena,70,config);

     }while(!feof(config));

     fclose(config);

    return configuracion;

}

char \*obtenerclave(char \*clave){

    char i=0;

    while(\*(clave+i)!=' '){

        i++;

    }

    \*(clave+i)=0;

    return clave+i+1;

}

Inicio

#include "C:\Users\Ceit\Downloads\control proteccion de sobre tension.txt"

/\* Puntero a funciones que carga en "clave" a la palabra clave y en "valo" a la posicion de memoria siguiente\*/

char \*obtenerclave(char \*clave);

datos\_t inicio() {

    FILE \*config;

    char cadena[70], \*clave, \*valo;

    char variables[4][30]={"cables","tension\_limite","tension","avisos"}, i;

    datos\_t configuracion;

    if(!(config=fopen("Config.conf","rb"))){

        printf("\nNo se encontro el archivo buscado\n");

     }

     fgets(cadena,70,config);

     do{

        clave=cadena;

        if(\*(clave) != '/' && strlen(clave)>0){

            valo=obtenerclave(clave);

            for(i=0;i<3;i++)

            {

                if(!strcmp(clave, variables[i])){

                    switch(i)

                    {

                        case 0: configuracion.cables=atoi(valo);

                                break;

                        case 1: configuracion.tension\_limite=atoi(valo);

                                break;

                        case 2: configuracion.tension=atoi(valo);

                                break;

                        case 3: strcpy(configuracion.avisos,valo);

                                break;

                     }

                 }

             }

         }

         fgets(cadena,70,config);

     }while(!feof(config));

     fclose(config);

    return configuracion;

}

char \*obtenerclave(char \*clave){

    char i=0;

    while(\*(clave+i)!=' '){

        i++;

    }

    return clave+i+1;

}

C

#include "...\libreria\LIBRERIA\_h.h"

int main()

{

    datos\_t config; // estructura donde va a estar los datos que necesita la maquina

    estados\_t estado = okay; //inicializo la maquina

    estados\_t (\*fsm[])(datos\_t) = {f\_espera, f\_desconexion};/\* Declaro el vector puntero a funciones \*/

    config=f\_inicio(); // Llamo a la funcion inicio

    while(1)

    {

        estado = (\*fsm[estado])(config);

    }

    return 0;

}