```
#include <SPI.h>        // needed for Arduino versions later than 0018
#include <Ethernet.h>
#include <EthernetUdp.h>        // UDP library from: bjoern@cs.stanford.edu 12/30/2008
#include <EEPROM.h>
#include <Wire.h>
#include <IRremote.h>

#define DS1307_I2C_ADDRESS 0x68  //Direccion Reloj

/
**********************************************************************************************************************/
/
**********************************************************************************************************************/
/
**********************************************************************************************************************/
/
**********************************************************************************************************************/
//ZONA DE CONFIGURACIONES
//Define numero de entradas salidas
//Configuracion Red

/
**********************************************************************************************************************/

//SETTINGS ZONE
//Defines number of inputs and outputs
//Network configuration
//Set or Restet Daylight saving time o DST
/
**********************************************************************************************************************/
/
**********************************************************************************************************************/
/
**********************************************************************************************************************/
/
**********************************************************************************************************************/
//Activa o desactiva cambio hora automatico invierno verano
//Set or Restet Daylight saving time o DST
//El modo dts esta configurado para europa
//Dts mode is set to europe
//Para otros paises configure funcion CargaHora()
//For other countries configure  function  CargaHora()
const boolean Enable_DaylightSavingTime  = true;
```

```
//Numero de Entradas con conmutador
//Number of swicth Inputs
const int Number_SwicthInput = 2;
//Numero de Entradas
//Number of Inputs
const int Number_Input = 10;
//Numero de Salidas
//Number of Outputs
const int Number_Output = 16;
//Numero de Persianas
//Number of blind
const int NumeroPersianas = 2;
//Pin para recepción infrarrojo = 19
//Infrared reception Pin = 19
//Puedes cambiar ese pin
//You can change this pin


//CONFIGURACION DE RED
//Direccion IP ES MUY PROBABLE QUE TENGAS QUE AJUSTARLO A TU RED LOCAL
//Cambia la ip por una dentro de tu rango, tambien debes actualizarla en tu aplicacion android
//Tambien puedes cambiar el puerto, por defecto 5000

// NETWORK CONFIGURATION
// IP Address, ADJUST TO YOUR LOCAL NETWORK
//If you change the IP address will have to adjust in android application
//If you change the Local Port address will have to adjust in android application

byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
IPAddress ip(192, 168, 1, 200);
unsigned int localPort = 5000;      // puerto local para eschucha de paquete

const String Mail ="";
const char* Key ="12345678";          //8 Characters, no more, no less....8 Caracteres, no mas, no menos
const boolean SecureConnection=false;  // ENABLED SECURE CONNECTION = TRUE.... CONEXION SEGURA =
TRUE
byte EspRfrIp = 0;


/
**********************************************************************************************
*******************/
/
**********************************************************************************************
*******************/
/
**********************************************************************************************
*******************/
/
**********************************************************************************************
```

```
******************/
//FIN DE ZONAS DE CONFIGURACIONES
//END SETTINGS ZONE
/
**********************************************************************************************************
******************/
/
**********************************************************************************************************
******************/
/
**********************************************************************************************************
******************/
/
**********************************************************************************************************
******************/
IRrecv irrecv(19);//El 19 corresponde con el pin de arduino, cambiar para utilizar otro
decode_results results;
IRsend irsend;
// buffers para recepcion y envio de datos
char packetBuffer[UDP_TX_PACKET_MAX_SIZE];
EthernetClient client;

// Instanacia Ethernet UDP Para enviar y recibir paqueteP
EthernetUDP Udp;
//Varibles Reloj
byte second, minute, hour, dayOfWeek, dayOfMonth, month, year,minutoMemory,TipoDia;
boolean HoraRetrasa;
unsigned long LastMsg;

//Variables Gestion Entradas Salidas
int PinSwicthInput[Number_SwicthInput];      //Asocia el pin a la entrada correspondiente
unsigned long LastTimeSwicthInput[Number_SwicthInput];  //Ultima vez que la entrada cambio el estado
int SwicthState[Number_SwicthInput];         // current state of the button

int PinInput[Number_Input];      //Asocia el pin a la entrada correspondiente
unsigned long LastTimeInput[Number_Input];  //Ultima vez que la entrada estaba en reposo
byte InState[Number_Input];  //Estado Entrada
int PinOutput[Number_Output];      //Asocia el pin a la entrada correspondiente

//Registros Salidas Circuitos
byte ElectricalCircuitValue[30];
float Temperatura1=22.2;
float Temperatura2=22.4;
float Temperatura3=49;

////////////////////////////////////////////////////////////////////////////////////////////////
//Configuracion Persianas
////////////////////////////////////////////////////////////////////////////////////////////////
byte PosicionPersiana[NumeroPersianas];  //Controla la posicion de la persiana % Subida
unsigned long TiempoMovPersiana[NumeroPersianas];  //Tiempo de mov desde el ultimo refresco

//Memoria de tiempos y posicion respecto a tiempo
```

```
unsigned long TiempoPosPersianaUp[NumeroPersianas];  //Posicion persiana en subida
unsigned long TiempoPosPersianaDown[NumeroPersianas];  //Posicion persiana en Bajada
unsigned long TimUpPersiana[NumeroPersianas];  //Tiempo en MicrosSg subida persiana
unsigned long TimDowPersiana[NumeroPersianas];  //Tiempo en MicrosSg bajada persiana

//Variables para salidas y entradas
boolean OutUpPersiana[NumeroPersianas];  //Boleana para activar salida persiana
boolean OutDowPersiana[NumeroPersianas];  //Boleana para activar salida persiana
boolean InUpPersiana[NumeroPersianas];  //Boleana pulsador subida Persiana
boolean InDowPersiana[NumeroPersianas];  //Boleana pulsador bajada Persiana

boolean Condicionados[10];            //Guarda el estado de los condicionados
byte Consignas[10];

void setup() {
/
**********************************************************************************************
*******************/
/
**********************************************************************************************
*******************/
/
**********************************************************************************************
*******************/
/
**********************************************************************************************
*******************/
  //ZONA DE CONFIGURACIONES
  //Configura pines de entrada
  //Configura pines salida

/**********************************************************************************************
***************************/

  //SETTINGS ZONE
  //inputs PIN, number in arduino board
  //Output Pin, number in arduino board
/
**********************************************************************************************
*******************/
/
**********************************************************************************************
*******************/
/
**********************************************************************************************
*******************/
/
**********************************************************************************************
*******************/
```

```
 //Pines de entrada
 PinInput[0]=40;
 PinInput[1]=41;
 PinInput[2]=42;
 PinInput[3]=43;
 PinInput[4]=44;
 PinInput[5]=45;
 PinInput[6]=46;
 PinInput[7]=47;
 PinInput[8]=48;
 PinInput[9]=49;
   //Pines entrada con Conmutador
 PinSwicthInput[0]=38;
 PinSwicthInput[1]=39;
 //Pines de salida

 PinOutput[0]=22;
 PinOutput[1]=23;
 PinOutput[2]=24;
 PinOutput[3]=25;
 PinOutput[4]=26;
 PinOutput[5]=27;
 PinOutput[6]=28;
 PinOutput[7]=29;
 PinOutput[8]=30;
 PinOutput[9]=31;
 PinOutput[10]=32;
 PinOutput[11]=33;
 PinOutput[12]=34;
 PinOutput[13]=35;
 PinOutput[14]=36;
 PinOutput[15]=37;

/**********************************************************************************************************/
/
**********************************************************************************************************/
//FIN DE ZONAS DE CONFIGURACIONES
/
**********************************************************************************************************/
/
**********************************************************************************************************/
/
**********************************************************************************************************/
/
**********************************************************************************************************/
```

```
 Serial.begin(9600);
 // Inico Ethernet y UDP

 Ethernet.begin(mac,ip);
 Udp.begin(localPort);

 //Inicio control Horarios
 Wire.begin();
 CargaHora();

 //Iniciamos Entradas Salidas
 for (int i =0; i<30; i++){ElectricalCircuitValue[i]=0;}

 //Fijamos pines entrada salida
 unsigned long currenMillis = millis();
 for (int i=0; i<Number_Input;i++){pinMode(PinInput[i], INPUT);LastTimeInput[i]=currenMillis;InState[i]=0;}
 for (int i=0; i<Number_Output;i++){pinMode(PinOutput[i], OUTPUT);}
 for (int i=0; i<10;i++){Consignas[i]=EEPROM.read(940 + i);}
 int reading;
 for (int i=0; i<Number_SwicthInput;i++){pinMode(PinSwicthInput[i], INPUT);reading =
digitalRead(PinSwicthInput[i]);LastTimeSwicthInput[i]=millis();SwicthState[i]=reading;}

 //Fijamos Temp.  Consigna en Inicio
 ElectricalCircuitValue[20]=20;
 ElectricalCircuitValue[21]=20;
 ElectricalCircuitValue[22]=20;



 //Fijamo valores y posicion inicio persianas
 //Fijamos el tiempo de subida bajada Persianas
 //Se encuentran apartir de la direccion 3880
 //
 ReiniciarTiempoPersianas();

 //Subimos las persianas a iniciar para ajustar posicion
 for (int per=0; per<NumeroPersianas; per++){
   ReiniciarPosicionPersiana(per);
   InDowPersiana[per]=false;
   InUpPersiana[per]=false;
 }
  irrecv.enableIRIn(); // Start the receiver
}

/**********************************************************************************************
*********************/
/
**********************************************************************************************
*******************/
//EVENTOS CONTROL ENTRADAS SALIDAS
//INPUT OUTPUT CONTROL EVENTS
```

```
//CUATRO EVENTOS PARA ENTRADAS DIGITALES
//CONMUTADOR CAMBIA VALOR
//PULSACION CORTA
//PULSACION LARGA, MAYOR DE 0.5 SEGUNDOS
//FINAL PULSACION LARGA

// FOUR EVENTS FOR DIGITAL INPUTS
//VALUE CHANGE SWITCH
// PRESS SHORT
// PRESS LONG, OVER 0.5 SECONDS
// LONG PRESS END.

/
**********************************************************************************************
*******************/
/
**********************************************************************************************
*******************/
/
**********************************************************************************************
*******************/
/
**********************************************************************************************
*******************/
void SwicthStateChange(int NumberInput){
  //Serial.println(NumberInput);
    /*************************************************************/
  //Este evento se produce cuando un conmutador cambia posicion
  // This event occurs with swicth change state.
  /*************************************************************/
    //Ejemplo de uso
  // Example of use


  //Ado Pasillo
   if (NumberInput==0){if (ElectricalCircuitValue[2]==1)
{ElectricalCircuitValue[2]=0;}else{ElectricalCircuitValue[2]=1;}}
   //Ado con Interruptor
   if (NumberInput==1){if (ElectricalCircuitValue[3]==1)
{ElectricalCircuitValue[3]=0;}else{ElectricalCircuitValue[3]=1;}}
}
void ShortInput(int NumberInput){
  /*************************************************************/
  //Este evento se produce con una pulsación corta.
  // This event occurs with a short press.
  /*************************************************************/
  //Ejemplo de uso
  // Example of use


  //Iluminacion 1
  if (NumberInput==0){
```

```
       switch (ElectricalCircuitValue[0]) {
        case 0:
          ElectricalCircuitValue[0]=3; break;
        case 1:
          ElectricalCircuitValue[0]=3; break;
        case 2:
           ElectricalCircuitValue[0]=1; break;
        case 3:
           ElectricalCircuitValue[0]=2; break;
      }
 }
 //Iluminacion 2
 if (NumberInput==2){
    switch (ElectricalCircuitValue[1]) {
        case 0:
          ElectricalCircuitValue[1]=3; break;
        case 1:
          ElectricalCircuitValue[1]=3; break;
        case 2:
           ElectricalCircuitValue[1]=1; break;
        case 3:
           ElectricalCircuitValue[1]=2; break;
      }
 }
 //Enchufe 1
  if (NumberInput==7){if (ElectricalCircuitValue[6]==1)
{ElectricalCircuitValue[6]=0;}else{ElectricalCircuitValue[6]=1;}}

  //Enchfue 2
  if (NumberInput==6){if (ElectricalCircuitValue[7]==1)
{ElectricalCircuitValue[7]=0;}else{ElectricalCircuitValue[7]=1;}}

  //Persianas
 //blind
 //Ejemplo de uso
 // Example of use

  //Persiana 1
  if (NumberInput==1){ElectricalCircuitValue[23]=100;}
  if (NumberInput==5){ElectricalCircuitValue[23]=0;}
 //Persiana 2
  if (NumberInput==9){ElectricalCircuitValue[24]=100;}
  if (NumberInput==4){ElectricalCircuitValue[24]=0;}

}
void LongInputEnd(int NumberInput){
 /*****************************************************************/
 //FINAL DE PULSACION LARGA
 //LONG PRESS END, EVENT
 // This event occurs with end a long press.
 /*****************************************************************/
```

```
 //Persianas
 //blind
 //Ejemplo de uso
 // Example of use

 /*
 if (NumberInput==1){InUpPersiana[0]=false;}
 if (NumberInput==5){InDowPersiana[0]=false;}
 if (NumberInput==9){InUpPersiana[1]=false;}
 if (NumberInput==4){InDowPersiana[1]=false;}
 */
}
void LongInput(int NumberInput){
 /*****************************************************************/
 //EVENTO PRODUCINO AL INICIO DE UNA PULSACION LARGA
 // LONG PRESS START
 // This event occurs with start a long press.
 /*****************************************************************/
 //Ejemplo de uso
 // Example of use

 /*
   //Pulsador Iluminacion 1
  if (NumberInput==0){ElectricalCircuitValue[0]=0;}

  //Pulsador Iluminacion 2
  if (NumberInput==2){ElectricalCircuitValue[1]=0;}

 //Pulsador Enchufe 1
 if (NumberInput==7){ElectricalCircuitValue[6]=1;ElectricalCircuitValue[7]=1;ElectricalCircuitValue[8]=1;}

  //Pulsador Enchfue 2
  if (NumberInput==6){ElectricalCircuitValue[6]=0;ElectricalCircuitValue[7]=0;ElectricalCircuitValue[8]=0;}

 //Persianas
 //blind
 //Ejemplo de uso
 // Example of use

 if (NumberInput==1){InUpPersiana[0]=true;InDowPersiana[0]=false;}
 if (NumberInput==5){InDowPersiana[0]=true;InUpPersiana[0]=false;}
 if (NumberInput==9){InUpPersiana[1]=true;InDowPersiana[1]=false;}
 if (NumberInput==4){InDowPersiana[1]=true;InUpPersiana[1]=false;}
 */
}
void OutControl()
{
  /*****************************************************************/
  //Gestion Salidas
  //Activamos los reles de control.

  // Outputs Management
```

```
 // Activate control relays.
 /*************************************************************/
//Ejemplo de uso
// Example of use


 // Circuito Numero 1
 //Iluminacion Iluminacion con 3 puntos de luz controlados mediante 2 Reles.
 switch (ElectricalCircuitValue[0]) {
  case 0:    // your hand is on the sensor
    digitalWrite(PinOutput[1], LOW);
    digitalWrite(PinOutput[3], LOW);
    break;
  case 1:    // your hand is close to the sensor
    digitalWrite(PinOutput[1], HIGH);
    digitalWrite(PinOutput[3], LOW);
    break;
  case 2:    // your hand is a few inches from the sensor
     digitalWrite(PinOutput[1], LOW);
    digitalWrite(PinOutput[3], HIGH);
    break;
   case 3:
     digitalWrite(PinOutput[1], HIGH);
    digitalWrite(PinOutput[3], HIGH);
    break;
   default:
    ElectricalCircuitValue[0]=0;
    break;
  }

 // Circuito Numero 2
 //Iluminacion Iluminacion con 3 puntos de luz controlados mediante 2 Reles.
   switch (ElectricalCircuitValue[1]) {
  case 0:
    digitalWrite(PinOutput[5], LOW);
    digitalWrite(PinOutput[7], LOW);
    break;
  case 1:
    digitalWrite(PinOutput[5], HIGH);
    digitalWrite(PinOutput[7], LOW);
    break;
  case 2:
     digitalWrite(PinOutput[5], LOW);
    digitalWrite(PinOutput[7], HIGH);
    break;
   case 3:
     digitalWrite(PinOutput[5], HIGH);
    digitalWrite(PinOutput[7], HIGH);
    break;
   default:
    ElectricalCircuitValue[1]=0;
    break;
```

```
}

// Circuito Numero 3
   //Ado Pasillo
 switch (ElectricalCircuitValue[2]) {
  case 0:
    digitalWrite(PinOutput[8], HIGH);
    break;
  case 1:
    digitalWrite(PinOutput[8], LOW);
    break;
  default:
    ElectricalCircuitValue[2]=0;
 }

// Circuito Numero 4
   //Ado Interruptor
 switch (ElectricalCircuitValue[3]) {
  case 0:
    digitalWrite(PinOutput[9], HIGH);
    break;
  case 1:
    digitalWrite(PinOutput[9], LOW);
    break;
  default:
    ElectricalCircuitValue[3]=0;
 }


// Circuito Numero 7
 //Enchufe
   //Enchufes
 switch (ElectricalCircuitValue[6]) {
  case 0:
    digitalWrite(PinOutput[15], HIGH);
    break;
  case 1:
    digitalWrite(PinOutput[15], LOW);
    break;
  default:
    ElectricalCircuitValue[6]=0;
 }
 // Circuito Numero 8
 //Enchufe numero 1
 switch (ElectricalCircuitValue[7]) {
  case 0:
    digitalWrite(PinOutput[13], HIGH);
    break;
  case 1:
    digitalWrite(PinOutput[13], LOW);
    break;
  default:
```

```
    ElectricalCircuitValue[7]=0;
  }
 // Circuito Numero 9
  //Enchufe numero 1
   switch (ElectricalCircuitValue[8]) {
   case 0:
     digitalWrite(PinOutput[11], HIGH);
     break;
   case 1:
     digitalWrite(PinOutput[11], LOW);
     break;
   default:
     ElectricalCircuitValue[8]=0;
  }
  // Circuito Numero 10
  //Enchufe numero 1
  if ((OutDowPersiana[0]==true)||(OutUpPersiana[0]==true))
  {
    if ((OutDowPersiana[0]==true)&&(OutUpPersiana[0]==false)){digitalWrite(PinOutput[2], LOW);
digitalWrite(PinOutput[0], LOW);  }
    if ((OutDowPersiana[0]==false)&&(OutUpPersiana[0]==true)){digitalWrite(PinOutput[2], HIGH);
digitalWrite(PinOutput[0], LOW);  }
  }
  else{
    digitalWrite(PinOutput[0], HIGH);
    digitalWrite(PinOutput[2], HIGH);
  }
  //Persiana 2
  if ((OutDowPersiana[1]==true)||(OutUpPersiana[1]==true))
  {
    if ((OutDowPersiana[1]==true)&&(OutUpPersiana[1]==false)){digitalWrite(PinOutput[6], LOW);
digitalWrite(PinOutput[4], LOW);  }
    if ((OutDowPersiana[1]==false)&&(OutUpPersiana[1]==true)){digitalWrite(PinOutput[6], HIGH);
digitalWrite(PinOutput[4], LOW);  }
  }
  else{
    digitalWrite(PinOutput[4], HIGH);
    digitalWrite(PinOutput[6], HIGH);
  }


  //Reles de reserva
  //Su estado de reposo corresponde a un nivel logico alto
  digitalWrite(PinOutput[10], HIGH);
  digitalWrite(PinOutput[12], HIGH);
  digitalWrite(PinOutput[14], HIGH);

}

char* RunCommand(byte CommandNumber){

   switch (CommandNumber) {
```

```
case 1:
  //APAGAR
  irsend.sendNEC(0x20DF10EF, 32);
  delay(40);

case 2:
  //INPUT
  irsend.sendNEC(0x20DFD02F, 32);
  delay(40);
  break;

case 3:
  //TV-RADIO
  irsend.sendNEC(0x20DF0FF0, 32);
  delay(40);
  break;

case 4:
  //GUIA
  irsend.sendNEC(0x20DFD52A, 32);
  delay(40);
  break;
 case 5:
  //SUBIR CANAL
  irsend.sendNEC(0x20DF00FF, 32);
  delay(40);
  break;

 case 6:
  //BAJAR CANAL
  irsend.sendNEC(0x20DF807F, 32);
  delay(40);
  break;

 case 7:
  //SUBIR VOLUMEN
  irsend.sendNEC(0x20DF40BF, 32);
  delay(40);
  break;

case 8:
  //BAJAR VOLUMEN
  irsend.sendNEC(0x20DFC03F, 32);
  delay(40);
  break;

 case 9:
  //1
  irsend.sendNEC(0x20DF8877, 32);
  delay(40);
  break;
```

```
 case 10:
  //2
  irsend.sendNEC(0x20DF48B7, 32);
  delay(40);
  break;

 case 11:
  //3
  irsend.sendNEC(0x20DFC837, 32);
  delay(40);
  break;

 case 12:
  //4
  irsend.sendNEC(0x20DF28D7, 32);
  delay(40);
  break;

 case 13:
  //5
  irsend.sendNEC(0x20DFA857, 32);
  delay(40);
  break;

case 14:
  //6
  irsend.sendNEC(0x20DF6897, 32);
  delay(40);
  break;

    case 15:
  //7
  irsend.sendNEC(0x20DFE817, 32);
  delay(40);
  break;

case 16:
  //8
  irsend.sendNEC(0x20DF18E7, 32);
  delay(40);
  break;

 case 17:
  //9
  irsend.sendNEC(0x20DF9867, 32);
  delay(40);
  break;

 case 18:
  //0
  irsend.sendNEC(0x20DF08F7, 32);
  delay(40);
```

```
    break;

   case 19:
    //SILENCIO
    irsend.sendNEC(0x20DF906F, 32);
    delay(40);
    break;
   }
 irrecv.enableIRIn(); // Re-enable receiver
 return "COMPLETED";


}
void CommonOrders(byte CommandNumber){
 Serial.print("Command Nª");
 Serial.println(CommandNumber);
}
char* ReadSensor(byte NumeroSensor)
{
  //El parametro numero de sensor representa se corresponde con el sensor configurado en la aplicación android
  //
  if (NumeroSensor==1){return "22,5º";}
  if (NumeroSensor==2){return "60%";}
  return "RESERVA";


}
void RecepcionInfrarrojos(decode_results *results) {


 if (results->decode_type == NEC) {
   //En este caso es nec pero puede ser JVC, PANASONIC...
   //Tienes que adaptarlo a tu mando a distancia
   switch (results->value){
    case  0x40BFA05F://cambia el codigo por el que envio tu mando a distancia
      SelectScene(1);
      break;

    case  0x40BF609F://cambia el codigo por el que envio tu mando a distancia
      SelectScene(2);
      break;

    case  0x40BFE01F://cambia el codigo por el que envio tu mando a distancia
      SelectScene(3);
      break;
    case  0x40BF906F://cambia el codigo por el que envio tu mando a distancia
      SelectScene(4);
      break;

    case  0x40BF50AF://cambia el codigo por el que envio tu mando a distancia
      SelectScene(5);
      break;
```

```
  case  0x40BFD02F://cambia el codigo por el que envio tu mando a distancia
    SelectScene(6);
    break;

  case  0x40BFB04F://cambia el codigo por el que envio tu mando a distancia
    SelectScene(7);
    break;

  case  0x40BF708F://cambia el codigo por el que envio tu mando a distancia
    SelectScene(8);
    break;

  case  0x40BFF00F://cambia el codigo por el que envio tu mando a distancia
    SelectScene(9);
    break;

  case  0x40BF8877://cambia el codigo por el que envio tu mando a distancia
    SelectScene(10);
    break;

 //Subir 20% Persiana 1
  case  0x40BFB847://cambia el codigo por el que envio tu mando a distancia
    if (ElectricalCircuitValue[23]<=80){ElectricalCircuitValue[23]+=20;}else{ElectricalCircuitValue[23]=100;}
    break;
 //bajar 20% Persiana 1
  case  0xC03F807F://cambia el codigo por el que envio tu mando a distancia
    if (ElectricalCircuitValue[23]>=20){ElectricalCircuitValue[23]-=20;}else{ElectricalCircuitValue[23]=0;}
    break;
 //Subir 20% Persiana 2
  case  0x40BF38C7://cambia el codigo por el que envio tu mando a distancia
    if (ElectricalCircuitValue[24]<=80){ElectricalCircuitValue[24]+=20;}else{ElectricalCircuitValue[24]=100;}
    break;
 //bajar 20% Persiana 2
  case  0xC03F00FF://cambia el codigo por el que envio tu mando a distancia
    if (ElectricalCircuitValue[24]>=20){ElectricalCircuitValue[24]-=20;}else{ElectricalCircuitValue[24]=0;}
    break;

  }
 }
}
/
**********************************************************************************************************
*******************/
/
**********************************************************************************************************
*******************/
//FIN DE ZONAS DE CONFIGURACIONES
//NO ES NECESARIO PROGRAMAR NADA POR DEBAJO DE ESTE PUNTO

/
**********************************************************************************************************
```

```
*******************/
// END CONFIGURATION ZONE
// NO NEED TO SET ANYTHING BELOW THIS POINT
/
*****************************************************************************************************
*******************/
/
*****************************************************************************************************
*******************/
/
*****************************************************************************************************
*******************/
void loop(){

  if(millis() < LastMsg ) {CargaHora();}
  if((millis() - LastMsg) >= 30000) {CargaHora();if (EspRfrIp<1){connectAndRfr();EspRfrIp=120;}else{EspRfrIp--;}}
  if (irrecv.decode(&results)) {
    RecepcionInfrarrojos(&results);
    irrecv.resume(); // Receive the next value
  }
  RecepcionPaqueteUDP();
  InputState();
  CheckSwicth();
  for (int p =0; p<NumeroPersianas;p++){GestionMovPersianas(p);} //Control de movimiento persianas
  OutControl();
}
void CheckSwicth(){
  int reading;
  for (int i=0; i<Number_SwicthInput;i++){
    reading = digitalRead(PinSwicthInput[i]);
    unsigned long InputMillis = millis();
    if (reading==SwicthState[i]){LastTimeSwicthInput[i]=InputMillis;}
    else{
      if(LastTimeSwicthInput[i]>InputMillis){LastTimeSwicthInput[i]=InputMillis;}
      if ((InputMillis-LastTimeSwicthInput[i])>=200){SwicthState[i]=reading;SwicthStateChange(i);}
    }
  }
}
void InputState(){
  for (int i=0; i<Number_Input;i++){
     unsigned long InputMillis = millis()-LastTimeInput[i];
     if ((InState[i]>=4)||(-1 >= InState[i])){InState[i]=0;}
     if (digitalRead(PinInput[i]) == LOW ) {
       if ((InState[i]==0)&&(InputMillis>=60)){LastTimeInput[i]=millis();InState[i]=1;}
       if ((InState[i]==1)&&(InputMillis>=440)){LongInput(i);InState[i]=2;}
       if (InState[i]==2){LastTimeInput[i]=millis();}
       if (InState[i]==3){LastTimeInput[i]=millis();InState[i]=1;}
     }
     else{
```

```
      if (InState[i]==0){LastTimeInput[i]=millis();}
      if (InState[i]==1){LastTimeInput[i]=millis();InState[i]=3;}
      if ((InState[i]==2) &&(InputMillis>=60)){LastTimeInput[i]=millis();InState[i]=0;LongInputEnd(i);}
      if ((InState[i]==3)&&(InputMillis>=60)){LastTimeInput[i]=millis();InState[i]=0;ShortInput(i);}
    }
  }
}


/************************************************************/
  //Gestion Persianas
/************************************************************/
void GestionMovPersianas(int NPersiana){
 if (InUpPersiana[NPersiana] || InDowPersiana[NPersiana])
 {  //Funcionamiento Manual
  if (InUpPersiana[NPersiana] && InDowPersiana[NPersiana])
{OutDowPersiana[NPersiana]=false;OutUpPersiana[NPersiana]=false;}
  else{
    if (InUpPersiana[NPersiana]){SubirPersiana(NPersiana);}
    if (InDowPersiana[NPersiana]){BajarPersiana(NPersiana);}
    ElectricalCircuitValue[23 + NPersiana]=PosicionPersiana[NPersiana];
  }
 }
 else
 {  //Funcionamiento Automatico;
  if (ElectricalCircuitValue[23 + NPersiana]==PosicionPersiana[NPersiana])
{OutDowPersiana[NPersiana]=false;OutUpPersiana[NPersiana]=false;}
  else
  {
    if (ElectricalCircuitValue[23 + NPersiana] > PosicionPersiana[NPersiana]){SubirPersiana(NPersiana);}
    else  {if (ElectricalCircuitValue[23 + NPersiana] < PosicionPersiana[NPersiana])  {BajarPersiana(NPersiana);}}
  }
 }
}
void SubirPersiana(int NPersiana){
 if (OutDowPersiana[NPersiana]==true)
{BajarPersiana(NPersiana);OutDowPersiana[NPersiana]=false;OutControl();delay(200);}
 long TiempoActual = micros();
 if (OutUpPersiana[NPersiana]==false){OutUpPersiana[NPersiana]=true;}
 else{
  long DiferenciaTiempo;
  if (TiempoActual<TiempoMovPersiana[NPersiana]){DiferenciaTiempo=TiempoActual;}else{DiferenciaTiempo =
TiempoActual-TiempoMovPersiana[NPersiana];}
  if ((TiempoPosPersianaUp[NPersiana] + DiferenciaTiempo)<TimUpPersiana[NPersiana])
{TiempoPosPersianaUp[NPersiana]=TiempoPosPersianaUp[NPersiana] + DiferenciaTiempo;}
  else{TiempoPosPersianaUp[NPersiana]=TimUpPersiana[NPersiana];}
  byte porcentajeSubida = TiempoPosPersianaUp[NPersiana] / (TimUpPersiana[NPersiana]/100);
  byte porcentajeBajada=100-porcentajeSubida;
  PosicionPersiana[NPersiana]=porcentajeSubida;
  TiempoPosPersianaDown[NPersiana]=porcentajeBajada*(TimDowPersiana[NPersiana]/100);
 }
```

```
  TiempoMovPersiana[NPersiana]=TiempoActual;
}
void BajarPersiana(int NPersiana){
  if (OutUpPersiana[NPersiana]==true)
{SubirPersiana(NPersiana);OutUpPersiana[NPersiana]=false;OutControl();delay(200);}
  long TiempoActual = micros();
  if (OutDowPersiana[NPersiana]==false){OutDowPersiana[NPersiana]=true;}
  else{
    long DiferenciaTiempo;
    if (TiempoActual<TiempoMovPersiana[NPersiana]){DiferenciaTiempo=TiempoActual;}else{DiferenciaTiempo =
TiempoActual-TiempoMovPersiana[NPersiana];}
    if ((TiempoPosPersianaDown[NPersiana] + DiferenciaTiempo)<TimDowPersiana[NPersiana])
{TiempoPosPersianaDown[NPersiana]=TiempoPosPersianaDown[NPersiana] +
DiferenciaTiempo;}else{TiempoPosPersianaDown[NPersiana]=TimDowPersiana[NPersiana];}

    byte porcentajeBajada = TiempoPosPersianaDown[NPersiana] / (TimDowPersiana[NPersiana]/100);
    byte porcentajeSubida=100-porcentajeBajada;
    PosicionPersiana[NPersiana]=porcentajeSubida;
    TiempoPosPersianaUp[NPersiana]=porcentajeSubida*(TimUpPersiana[NPersiana]/100);
  }
  TiempoMovPersiana[NPersiana]=TiempoActual;
}
void ReiniciarTiempoPersianas(){for ( int c =0; c<NumeroPersianas; c++){TimUpPersiana[c]=(EEPROM.read(3880 + c))*
1000000; TimDowPersiana[c]=(EEPROM.read(3890 + c))* 1000000;}}
void ReiniciarPosicionPersiana(int NumPersiana)
{ TiempoPosPersianaUp[NumPersiana]=0;TiempoPosPersianaDown[NumPersiana]=TimDowPersiana[NumPersiana];Elect
ricalCircuitValue[23+NumPersiana]=100;}

void RecepcionPaqueteUDP()
{

  int packetSize = Udp.parsePacket();

  if(packetSize)
  {
    Udp.read(packetBuffer,UDP_TX_PACKET_MAX_SIZE);

    int LongCadena;
    if  (SecureConnection){
      for(int c=0; c<8;c++){if (Key[7-c]!=packetBuffer[packetSize-(c+1)]){return;}}
      LongCadena=packetSize-7;}
    else{LongCadena=packetSize + 1;}

    char  CadenaEntrada[LongCadena];
    for (int c=0;c<LongCadena;c++){CadenaEntrada[c]=packetBuffer[c];}
    CadenaEntrada[LongCadena-1]='\0';
    String CadenaIn = (String)CadenaEntrada;
    if (CadenaIn.indexOf("COMCOMM")>-1)
{CommonOrders(packetBuffer[7]);EnviarRespuesta("COMCOMOK");return;}
    if (CadenaIn=="CLEARHORARIO"){for (int i = 950; i <= 3879; i++){EEPROM.write(i,
66);}EnviarRespuesta("HORARIOS BORRADOS"); return;}
    if (CadenaIn=="CLEARESPCDAY"){for (int i = 3900; i <= 3999; i++){EEPROM.write(i, 0);}EnviarRespuesta("DIAS
```

```
ESPECIALES BORRADOS");return; }
   if (CadenaIn.indexOf("SETFH")>-1){
     setDateDs1307(packetBuffer[5] ,packetBuffer[6], packetBuffer[7], packetBuffer[8], packetBuffer[9], packetBuffer[10],
packetBuffer[11]);
     CargaHora();
     EnviarRespuesta("SETFHOK");
     return;
   }
    if (CadenaIn.indexOf("GETSENSOR")>-1)
   {
     char* Resultado;
     int LongCad;
     Resultado = ReadSensor(packetBuffer[9]);
     LongCad = strlen(Resultado);
     char Respuesta[LongCad  + 8];

Respuesta[0]='S';Respuesta[1]='E';Respuesta[2]='N';Respuesta[3]='S';Respuesta[4]='O';Respuesta[5]='R';Respuesta[6]=pac
ketBuffer[9];
       int p;
       for (p=0;p<LongCad;p++){ Respuesta[p+7]=Resultado[p];}
       Respuesta[LongCad  + 7]='\0';
       EnviarRespuesta(Respuesta);
       return;
   }
   if (CadenaIn.indexOf("READDAY")>-1){
       char Respuesta[325];
       Respuesta[0]='C';
       Respuesta[1]='F';
       Respuesta[2]='D';
       Respuesta[3]='A';

       int PunteroRegistro;
       if (packetBuffer[7]=='2'){PunteroRegistro=3950;}else{PunteroRegistro=3900;}
       for (int  i = 0; i<50;i++){Respuesta[i+4]=EEPROM.read(PunteroRegistro + i); }
       Respuesta[324]='\0';
       EnviarRespuesta(Respuesta);
       return;
   }
   if (CadenaIn.indexOf("WRIDAYE")>-1){
     int Reg;
     int Pos=8;
     if (packetBuffer[7]==2){Reg=3950;}else{Reg=3900;}
     for (int i = Reg; i<(Reg+50);i++){EEPROM.write(i,packetBuffer[Pos]);Pos++;}
     EnviarRespuesta("COMPLETED");
     return;
   }
   if (CadenaIn.indexOf("RETRIGGER")>-1){
     char Respuesta[53];
     Respuesta[0]='T';
     Respuesta[1]='I';
     Respuesta[2]='G';
     Respuesta[3]='R';
```

```
    int Pos=4;
    for (int i = 950; i<998;i++){Respuesta[Pos]=EEPROM.read(i)+1;Pos++;}
    Respuesta[52]='\0';
    EnviarRespuesta(Respuesta);
    return;
  }
   if (CadenaIn.indexOf("WTGR")>-1){
    int Pos=4;
    for (int i = 950; i<998;i++){EEPROM.write(i,packetBuffer[Pos]);Pos++;}
    EnviarRespuesta("COMPLETED");
    return;
  }


  if (CadenaIn.indexOf("READHOR")>-1){
    char Respuesta[325];
    Respuesta[0]='H';
    Respuesta[1]='O';
    Respuesta[2]='R';
    Respuesta[3]='A';
    int Reg;
    int Pos=4;
    Reg=packetBuffer[7]*320+1000;
    for (int i = Reg; i<(Reg+320);i++){Respuesta[Pos]=EEPROM.read(i)+1;Pos++;}
    Respuesta[324]='\0';
    EnviarRespuesta(Respuesta);
    return;
  }

  if (CadenaIn.indexOf("WRITHOR")>-1){
    int Reg;
    int Pos=8;
    Reg=packetBuffer[7]*320+1000;
    for (int i = Reg; i<(Reg+320);i++){EEPROM.write(i,packetBuffer[Pos]);Pos++;}
    EnviarRespuesta("COMPLETED");
    return;
  }
  if (CadenaIn.indexOf("SVAL")>-1){ElectricalCircuitValue[packetBuffer[4]-1]=packetBuffer[5]-
1;EnvioEstadoActual();return;}

  if (CadenaIn=="VACT"){EnvioEstadoActual();return;}

  if (CadenaIn.indexOf("SSCE")>-1)
  {
   int Dir;
   Dir = (int)packetBuffer[4];
   SelectScene(Dir);
   EnvioEstadoActual();
   return;
  }
  if (CadenaIn.indexOf("WESC")>-1)
  {
```

```
  int Dir;
  Dir = (int)packetBuffer[4];
  Dir = (Dir-1) * 30;
   for (byte  i = 0; i<30;i++){EEPROM.write(i+Dir, packetBuffer[5+i]-1);}
   EnviarRespuesta("COMPLETED");
   return;
}
if (CadenaIn == "ESTADOINST"){ReadDate();return;}
if (CadenaIn.indexOf("RESC")>-1)
{
   int Dir;
   Dir = (int)packetBuffer[4];
   Dir = (Dir-1) * 30;

   char Respuesta[36];
   Respuesta[0]='V';
   Respuesta[1]='E';
   Respuesta[2]='S';
   Respuesta[3]='C';
   Respuesta[4]=packetBuffer[4];

   for (int i = 0; i < 30;i++)
   {
    byte V=EEPROM.read(Dir+i);
    if (V<=254){V++;}
    Respuesta[i + 5 ]=V;
   }
   Respuesta[35]='\0';
   EnviarRespuesta(Respuesta);
   return;
}

if (CadenaIn == "ENABLEHOR")
{
  char Respuesta[56];
   Respuesta[0]='E';
   Respuesta[1]='N';
   Respuesta[2]='H';
   Respuesta[3]='O';
   Respuesta[4]='R';

   for (int i = 0; i < 50;i++){Respuesta[i + 5 ]=EEPROM.read(400+i)+1;}
   Respuesta[55]='\0';
   EnviarRespuesta(Respuesta);
   return;
}
if (CadenaIn.indexOf("WHOR")>-1)
{
   for (byte  i = 0; i<50;i++){EEPROM.write(i+400, packetBuffer[4+i]-1);}
   EnviarRespuesta("COMPLETED");
   return;
}
```

```
 if (CadenaIn == "CONENABLE")
{
  char Respuesta[16];
   Respuesta[0]='E';
   Respuesta[1]='N';
   Respuesta[2]='C';
   Respuesta[3]='O';
   Respuesta[4]='N';

   for (int i = 0; i < 10;i++){if (Condicionados[i]==true){Respuesta[i + 5 ]=2;}else{Respuesta[i + 5 ]=1;}}
   Respuesta[15]='\0';
   EnviarRespuesta(Respuesta);
   return;
}
if (CadenaIn.indexOf("WCON")>-1)
{
   for (byte  i = 0; i<10;i++){if ( packetBuffer[4+i]==2){Condicionados[i] = true;}else{Condicionados[i] = false;}}
   EnviarRespuesta("COMPLETED");
   return;
}

if (CadenaIn.indexOf("COMANDO")>-1){EnviarRespuesta(RunCommand(packetBuffer[7]));return;}

if (CadenaIn == "TIMPERSIANA")
{
  char Respuesta[26];
   Respuesta[0]='L';
   Respuesta[1]='E';
   Respuesta[2]='C';
   Respuesta[3]='P';
   Respuesta[4]='E';

   for (int i = 0; i < 20;i++){ Respuesta[i + 5 ]=EEPROM.read(3880+i)+1;}
   Respuesta[25]='\0';
   EnviarRespuesta(Respuesta);
   return;

}
 if (CadenaIn == "SETPOINT")
{
  char Respuesta[16];
   Respuesta[0]='S';
   Respuesta[1]='E';
   Respuesta[2]='P';
   Respuesta[3]='O';
   Respuesta[4]='I';
   for (int i = 0; i < 10;i++){Respuesta[i + 5 ]= Consignas[i]+1;}
   Respuesta[15]='\0';
   EnviarRespuesta(Respuesta);
   return;
```

```
   }


    if (CadenaIn.indexOf("WCOW")>-1)
    {
     char Respuesta[16];
     byte v = packetBuffer[5]-1;
     byte p = packetBuffer[4]-1;

     EEPROM.write(p + 940, v);
     Consignas[p]=v;

     Respuesta[0]='S';
     Respuesta[1]='E';
     Respuesta[2]='P';
     Respuesta[3]='O';
     Respuesta[4]='I';
     for (int i = 0; i < 10;i++){Respuesta[i + 5 ]= Consignas[i]+1;}
     Respuesta[15]='\0';
     EnviarRespuesta(Respuesta);
     return;
    }

    if (CadenaIn.indexOf("WPERS")>-1)
    {
      for (byte  i = 0; i<20;i++){EEPROM.write(i+3880, packetBuffer[5+i]-1);}
      ReiniciarTiempoPersianas();
      EnviarRespuesta("COMPLETED");
      return;
    }
    if (CadenaIn.indexOf("RESTPER")>-1){
       ReiniciarPosicionPersiana(packetBuffer[7]-1);
       EnviarRespuesta("RESETEANDO PERSIANA");
       return;
    }

   EnviarRespuesta("REPETIRMSG");
  }
}

void ReadDate(){
  char Respuesta[26];
  Respuesta[0]='E';
  Respuesta[1]='S';
  Respuesta[2]='T';
  Respuesta[3]='A';
  Respuesta[4]='C';
  Respuesta[5]='T';

  Respuesta[6]=TipoDia + 1;
  Respuesta[7]=hour + 1;
  Respuesta[8]=minute + 1;
```

```
 Respuesta[9]=dayOfMonth + 1;
 Respuesta[10]=month + 1;
 Respuesta[11]=year + 1;

 Respuesta[12]='\0';

 EnviarRespuesta(Respuesta);
}
void EnvioEstadoActual()
{
 char Respuesta[38];
 Respuesta[0]='E';
 Respuesta[1]='V';
 Respuesta[2]='A';
 Respuesta[3]='L';

 for (byte  i = 4; i<34;i++)
 {
   Respuesta[i]=ElectricalCircuitValue[i-4]+1;
 }

 int v;
 if ((Temperatura1 >=1)&&(Temperatura1<=49)){v =(Temperatura1*10)/2; Respuesta[34]=v+1;}else{Respuesta[34]=1;}
 if ((Temperatura2 >=1)&&(Temperatura2<=49)){v =(Temperatura2*10)/2; Respuesta[35]=v+1;}else{Respuesta[35]=1;}
 if ((Temperatura3 >=1)&&(Temperatura3<=49)){v =(Temperatura3*10)/2; Respuesta[36]=v+1;}else{Respuesta[36]=1;}

 Respuesta[37]='\0';
 EnviarRespuesta(Respuesta);

}

void EnviarRespuesta(char  ReplyBuffer[])
{
   // send a reply, to the IP address and port that sent us the packet we received
   Udp.beginPacket(Udp.remoteIP(), Udp.remotePort());
   Udp.write(ReplyBuffer);
   Udp.endPacket();
}

void SelectScene(int Dir){Dir = (Dir-1) * 30;for (byte i =0 ; i<30;i++){byte val =EEPROM.read(Dir+i); if (val<250)
{ElectricalCircuitValue[i]=val;}}}
void CargaHora()
{
 getDateDs1307(&second, &minute, &hour, &dayOfWeek, &dayOfMonth, &month, &year);
 LastMsg=millis();
 if (minute != minutoMemory){ActualizaMinuto();  }
}
void ActualizaMinuto()
{
 if(Enable_DaylightSavingTime==true)
 {
    //Adelanta la hora.Apartir del dia 25 de Marzo, busca el primer domingo
```

```
    //y cuando se han las 2 de la noche adelanta el reloj una hora
    if (month==3)
    {
      if (dayOfMonth >= 26)
      {
       if (dayOfWeek = 7)
       {
        if (hour==2)
        {
          hour = 3;
          setDateDs1307(second, minute, hour, dayOfWeek, dayOfMonth, month, year);
        }
       }

      }
    }
    //Retrasa la hora.Apartir del dia 25 de Octubre, busca el primer domingo
    //y cuando se han las 2 de la noche retrasa el reloj una hora
    if (month==10)
    {
      if (dayOfMonth >= 26)
      {
       if (dayOfWeek = 7)
       {
        if ((hour==2)&&(HoraRetrasa==false))
        {
          hour = 1;
          HoraRetrasa=true;
          setDateDs1307(second, minute, hour, dayOfWeek, dayOfMonth, month, year);
        }
       }

      }
     }
    }
    if (hour==3){HoraRetrasa=false;}
    minutoMemory=minute;


    TipoDia=dayOfWeek;
    int Reg;

    for (Reg=3900; Reg<=3948;Reg=Reg+2){if (month == EEPROM.read(Reg)){if (dayOfMonth==
EEPROM.read(Reg+1)){TipoDia=8;}}}//Verificacion Dia Especial 1
    for (Reg=3950; Reg<=3998;Reg=Reg+2){if (month == EEPROM.read(Reg)){if (dayOfMonth==
EEPROM.read(Reg+1)){TipoDia=9;}}}//Verificacion Dia Especial 2

    int R= ((TipoDia-1)*320)+1000;
    for (Reg=R; Reg<=(R+316);Reg=Reg+4)     {if ((hour==EEPROM.read(Reg))&&(minute==EEPROM.read(Reg+1)))
{byte ci=EEPROM.read(Reg+2);if ((ci>=0) &&(ci<50) && (EEPROM.read(ci+400)==1)){if (ci<30)
{ElectricalCircuitValue[ci]=EEPROM.read(Reg+3);}else{if (ci<40){SelectScene(ci-29);}else{if
(EEPROM.read(Reg+3)==1){Condicionados[ci-40]=true;}else{Condicionados[ci-40]=false;}}}}}}
```

```
   for (Reg=950; Reg<=994;Reg=Reg+4)    {if ((hour==EEPROM.read(Reg))&&(minute==EEPROM.read(Reg+1)))
{EEPROM.write(Reg, 66); byte ci=EEPROM.read(Reg+2);if ((ci>=0) &&(ci<50) ){if (ci<30)
{ElectricalCircuitValue[ci]=EEPROM.read(Reg+3);}else{if (ci<40){SelectScene(ci-29);}else{if
(EEPROM.read(Reg+3)==1){Condicionados[ci-40]=true;}else{Condicionados[ci-40]=false;}}}}}}

}
/*****************************************************************/
//  FUNCIONES RELOJ
/*****************************************************************/

// Convert normal decimal numbers to binary coded decimal
byte decToBcd(byte val)
{
  return ( (val/10*16) + (val%10) );
}

// Convert binary coded decimal to normal decimal numbers
byte bcdToDec(byte val)
{
  return ( (val/16*10) + (val%16) );
}

// Stops the DS1307, but it has the side effect of setting seconds to 0
// Probably only want to use this for testing
/*void stopDs1307()
{
  Wire.beginTransmission(DS1307_I2C_ADDRESS);
  Wire.send(0);
  Wire.send(0x80);
  Wire.endTransmission();
}*/

// 1) Sets the date and time on the ds1307
// 2) Starts the clock
// 3) Sets hour mode to 24 hour clock
// Assumes you're passing in valid numbers
void setDateDs1307(byte second,        // 0-59
                   byte minute,        // 0-59
                   byte hour,          // 1-23
                   byte dayOfWeek,     // 1-7
                   byte dayOfMonth,    // 1-28/29/30/31
                   byte month,         // 1-12
                   byte year)          // 0-99
{
   Wire.beginTransmission(DS1307_I2C_ADDRESS);
   Wire.write(0);
   Wire.write(decToBcd(second));    // 0 to bit 7 starts the clock
   Wire.write(decToBcd(minute));
   Wire.write(decToBcd(hour));      // If you want 12 hour am/pm you need to set
                                    // bit 6 (also need to change readDateDs1307)
   Wire.write(decToBcd(dayOfWeek));
```

```
  Wire.write(decToBcd(dayOfMonth));
  Wire.write(decToBcd(month));
  Wire.write(decToBcd(year));
  Wire.endTransmission();
}

// Gets the date and time from the ds1307
void getDateDs1307(byte *second,
        byte *minute,
        byte *hour,
        byte *dayOfWeek,
        byte *dayOfMonth,
        byte *month,
        byte *year)
{
 // Reset the register pointer
 Wire.beginTransmission(DS1307_I2C_ADDRESS);
 Wire.write(0);
 Wire.endTransmission();

 Wire.requestFrom(DS1307_I2C_ADDRESS, 7);

 // A few of these need masks because certain bits are control bits
 if (Wire.available()==7){
   *second     = bcdToDec(Wire.read() & 0x7f);
   *minute     = bcdToDec(Wire.read());
   *hour       = bcdToDec(Wire.read() & 0x3f);  // Need to change this if 12 hour am/pm
   *dayOfWeek  = bcdToDec(Wire.read());
   *dayOfMonth = bcdToDec(Wire.read());
   *month      = bcdToDec(Wire.read());
   *year       = bcdToDec(Wire.read());
 }
}

/*****************************************************************/
// REFRESCAR
/*****************************************************************/
void connectAndRfr(){
 if (Mail==""){return;}
 if (client.connect("www.excontrol.es", 80)) {
   client.print("GET http://excontrol.es/Users/IpSet.aspx?Mail=");
   client.print(Mail + "&Key=" + Key);

   client.println(" HTTP/1.0");
   client.println();
   RfIp();

 }
}

void RfIp(){
 int Reintento;
```

```
Reintento=0;

while(true){
  if (client.available()) {
    int c;
    for (c=0;c<5;c++){char c = client.read();}
    client.stop();
    client.flush();
    return;
  }
  else{

    Reintento++;
    RecepcionPaqueteUDP();
    InputState();
    for (int p =0; p<NumeroPersianas;p++){GestionMovPersianas(p);} //Control de movimiento persianas
    OutControl();
    delay(10);
    if (Reintento >= 100 ){
      client.stop();
      client.flush();
      return;
    }
  }
 }
}
```