PROYECTO Nº 3

Control de fichajes y rondas.

1.- INTRODUCCIÓN.

Conforme con la resolución del 5 de abril de 2011 por la que se convocan ayudas destinadas a realizar proyectos de innovación aplicada a la formación profesional del sistema educativo ( BOE del 27 de abril de 2011), se presenta la solicitud para **proyecto inmótico con plataforma arduino**. Este proyecto se aprueba y nuestro centro IES Tirant lo Blanc accede a él para desarrollar el tercer subproyecto. Este tiene por misión controlar y verificar un control de rondas por medio de tarjetas RFID (Radio Frecuency IDentification).

El arduino es una plataforma basado en microcontroladores AVR, montado en una placa con electrónica asociada y un entorno libre y gratuito de programación.

Los microcontroladores son de la Empresa americana ATMEL. Estos utilizan arquitectura Harvard y sistema de programación RISC ( Reduced Instruction Set Complex).

El software de programación del ARDUINO consiste en un entorno de desarrollo que implementa el [lenguaje de programación](http://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n) Processing/Wiring( lenguaje C) y un [cargador de arranque](http://es.wikipedia.org/wiki/Cargador_de_arranque) (boot loader) que configura la forma de trabajar la placa.

La empresa ICTEL, ubicada en Vigo, ha desarrollado unos equipos que son totalmente compatibles con el arduino. Nuestro propósito es adecuar este entorno para la realización de proyectos domóticos/inmóticos.

Bajo el nombre de **“Diseño de sistema de control de fichajes y control de rondas de vigilancia basado en la tecnología sin contacto 125 Khz”** se encuadra el tercer subproyecto, que entre sus especificaciones técnicas nos provee:

1.- Alimentación a 24 Vcc.

2.- Sistema conectado a un bus estándar y abierto: MODBUS RTU y MODBUS TCP.

3.- Posibilidad de conectar 128 equipos sin repetidor.

4.- Conectable e integrable con el sistema inmótico general.

5.- Software de administración que permita:

Alta y baja y modificación de usuarios.

Alta, baja y modificación de horarios.

Alta, baja y modificación de rondas de vigilancia y frecuencias.

Emisión de informes. Posibilidad de exportación a PDF, EXCEL y OpenOffice.

Las **tarjetas de RFID** poseen un código grabado de fábrica que no se puede modificar. Este código es único en cada tarjeta que la identifica y no hay dos tarjetas con el mismo número (similar a la identificación MAC en dispositivos IP). Están compuestas de 5 Bytes, de los cuales el primero no es significativo, ya que es un código del fabricante. No poseen alimentación interna y se activan por medio de inducción al acercarlas al lector de RFID. Nosotros solo leeremos los 4 Bytes siguientes. Así, tendremos 232 combinaciones (más de 4200 millones de combinaciones) lo que permite la diversidad de números en las tarjetas.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| COD\_FABRIC | Byte 3 | Byte 2 | Byte 1 | Byte 0 |

Unos ejemplos de lectura serán 24 03 AF 6B C0

00 12 67 FE 1A

2.- DESCRIPCIÓN DE LOS EQUIPOS UTILIZADOS.

Para la realización del proyecto hemos utilizados los equipos provistos en la memoria técnica del proyecto, que paso a describir.

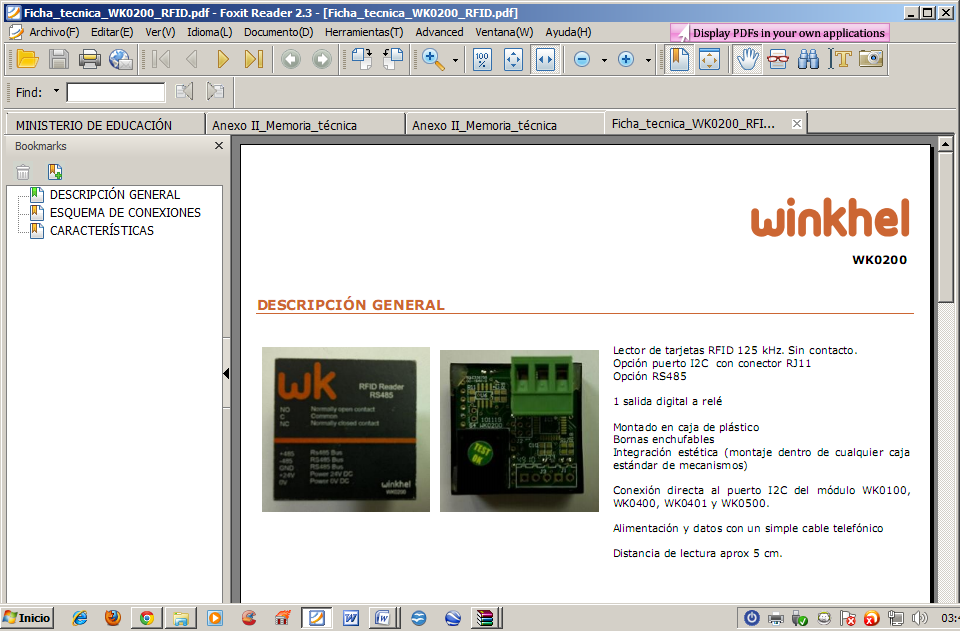
## WK0500: Pasarela



* *Características técnicas:*
* *Puerto Ethernet, puerto USB, puerto serie TTL, puerto I2C con conector RJ11, puerto RS232 y puerto RS485.*
* *Microcontrolador ATMEL Atmega 2560 MCU y con opción de reloj en tiempo real con batería.*
* *Ideal para hacer de pasarela o como maestro de un sistema basado en bus RS485 y con comunicación con PC.*
* *Posibilidad de funcionar como servidor web, por ejemplo para un sistema domótico.*
* *Se alimenta a 24 VDC y se suministra en caja para carril DIN, con unas dimensiones de 100 x 100 x 60 mm.*

## WINKEL 0201

* *Lector de tarjetas RFID. Funciona bajo la tecnología de 125 Khz, posee un puerto serie I2C (RJ11) y RS485.*
* *Salida digital a relé*
* *Integrable en cajas de mecanismos estándar.*
* *Distancia de lectura de 5 cm.*
* *Alimentación y datos por cable telefónico.*



3.- TERMINOLOGIA Y RESTRICIONES DEL PROYECTO.

Antes de definir la estructura que hemos decidido dar a los datos deberíamos conocer la terminología utilizada y la descripción del proyecto.

**Terminal:** es el dispositivo que se comunica con el WK0500 por medio de MODBUS/TCP. Pueden ser un PC, un Tablet, un terminal móvil. Por defecto utilizamos el terminal 0.

**Maestro:** es la pasarela de comunicación del WK0500. Este nombre puede llevar a engaño, ya que se comporta como esclavo para el terminal y como maestro para los remotos (WK200).

**Remotos o esclavos:** son en nuestro proyecto los lectores de RFID WK200.

**Canal:** no tiene sentido en este proyecto, ya que no controlamos habitaciones con varios dispositivos, sino solo la apertura de las puertas. Si trabajamos con el WK400 este tiene opción de controlar 4 dispositivos (canales), por medio de relés.

**Escena:** por medio de una escena queremos agrupar a varios elementos heterogéneos para que realicen una función común preestablecida. Así por ejemplo, podríamos tener la escena ver tv, donde algunas luces se encenderán, otras por medio de un dimmer estarán en un porcentaje de iluminación, etc... Otra escena típica es la de simulación de presencia donde secuencialmente se encenderán algunas luces. En nuestro subproyecto no hay previstas escenas. A continuación mostramos un esquema del conjunto implementado.

C:\Documents and Settings\Administrador\Configuración local\Archivos temporales de Internet\Content.IE5\SC4YS011\MC900237582[1].wmf

TERMINALES

MODBUS TCP

MODBUS RTU

WINKEL 0500

Pasarela de comunicaciones

WINKEL 0200

LECTOR RFID

WINKEL 0200

LECTOR RFID

WINKEL 0200

LECTOR RFID

REMOTOS

Las restricciones impuestas al proyecto son:

– Máximo 16 escenas.

– Máximo 128 elementos (remotos).

– Máximo 50 elementos con valores analógicos.

– Programación horaria con resolución 10 min.

Definición de elemento

– Id tipo+salida: 1 byte (4 bytes para id tipo + 4 bytes para número de salida)

– Id elemento: 2 byte

– Id remoto: 1 byte

– Escenas a las que pertenece: 2 byte. Se pone a 1 el bit correspondiente

– Valores en las escenas:

– Elemento analógico: 16 bytes (uno por valor en cada escena)

– Elemento digital: 2 bytes (1 bit por cada valor en cada escena)

Total analógico: 22 bytes

Total digital: 8 bytes

Definición de un programa horario

– Id programa: 1 byte

– Id elemento/escena/grupo: 12 bits

– elemento/escena/grupo: 2 bit

– habilitado/deshabilitado: 1 bit

– Hora y minuto: 7 bits (24 horas con resolución 10 min.)

– Días de la semana: 7 bits + 1 bit semanal/una sola vez

– Valor: 1 byte

– Si elemento o grupo digital: 0 o 1

– Si elemento o grupo analógico: valor

– Si escena: no se considera

Total programa horario: 6 bytes

Definición de puertas

– Id puerta: 1 byte

– Id remoto: 1 byte

– Id salida: 1 byte + bloqueada

Definición de termostatos

– Id termostato: 1 byte

– Id remoto: 1 byteComandos:

– Activar elemento

4.- PROTOCOLO MODBUS

Se puede considerar a MODBUS como un protocolo de enlace (OSI nivel 2) abierto y gratuito, que se puede implementar con distintas conexiones físicas. La comunicación es por un bus serie asíncrono y acepta distintas velocidades de transmisión. En nuestro caso hemos utilizado 19200 baudios.

El método de acceso al cable es del tipo maestro-esclavo y el número máximo de estaciones serán de 64(1 máster y 63 slaves).

Solo se pueden implementar dos tipos de mensajes:

-**Mensaje punto a punto**, donde el maestro inicia la comunicación y el esclavo escogido será el que responda.

- **Mensajes difundidos**, que es una orden que el maestro envía a todos los esclavos, sin esperar ninguna respuesta de estos. La envía a la dirección 0, ya que todos los remotos tiene esta dirección además de la suya propia. Este paquete de información es llamado también BROADCAST.

Podemos hablar de 3 tipos de codificaciones MODBUS: MODBUS ASCII, MODBUS RTU y MODBUS TCP.

La **MODBUS ASCII** esta en desuso, envía solo caracteres ASCII y duplica el ancho de banda respecto a la codificación RTU.

La codificación **MODBUS RTU** (Remote Transmission Unit) envía los datos codificados en hexadecimal; en cada paquete de 8 bits, dos caracteres hexadecimales. Esta será la forma de comunicación elegida entre el winkel 0500 y los esclavos winkel 0200.

La codificación **MODBUS TCP** transporta la misma información que la RTU, pero va encapsulada dentro del protocolo TCP. Se utilizara para la comunicación entre los terminales y el winkwel 0500.

5.- BUS DE COMUNICACIONES.

5.1 BUS DE COMUNICACIONES RS485.

Para la comunicación por el protocolo MODBUS RTU utilizamos como canal de transporte el **bus de comunicaciones RS-485.** Este necesita de tres hilos, aunque puede funcionar con solo 2. Se nombran de dos formas A, B, tierra o también lo podemos encontrar como A (+), B (-) y tierra. El terminal de tierra es el que se puede omitir. Esto lleva mucha controversia, ya que en instalaciones donde las tierras sean diferentes pueden tener diferencias de potencial elevadas que hagan que los equipos conectados a ellas puedan estropearse.

El bus de comunicaciones RS-485 se considera un bus balanceado, es decir toma la decisión del valor lógico según la diferencia de potencial existente entre los terminales A y B. Esta diferencia de potencial será el valor lógico “1” ó “0”.

Si VA > VB entre 0,2 V y 6 V tendremos un nivel alto “1”

Si VB > VA entre 0,2 V y 6 V tendremos un nivel alto “0”

La gran ventaja del bus RS-458 es su gran inmunidad al ruido. Para mejorar esta característica se debe utilizar cable trenzado de impedancia característica de 120 Ω. El apantallamiento también será una característica positiva a tener en cuenta, pero el trenzado del cable es la más importante. Con las mejores características del cable podemos llegar a cubrir distancias de 1200 m (sin repetidor) a una velocidad de 9600 baudios. A mayor velocidad la distancia disminuye, pero siempre tendremos valores muy elevados respecto a otros buses, como RS-232(aprox. 15 m) y Ethernet (aprox. 90 m). Otra pregunta que nos surge es cuántos equipos se pueden montar en un bus. Este bus admite 32 unidades de carga, pero los equipos winkel 0200 tienen como driver de hardware de red el CI MAX13487 y como estos equipos cargan a la red en ¼ podremos disponer de hasta 128 equipos. Esta es la limitación que impone el acceso al cable: un winkel 0500 podrá controlar 128 remotos.

5.2 BUS DE COMUNICACIONES ETHERNET.

Para la comunicación MODBUS RTU el acceso al cable se realiza con cable Ethernet, de sobra conocido.

6.- LISTA DE COMANDOS Y SU DESCRIPCION.

Los comandos los envía a los remotos el WK0500 por medio de MODBUS/RTU. Este los recibe por medio de los terminales. Los comandos utilizan los holden registers (HR).

Además de la memoria RAM (HR) disponemos de una memoria eeprom.

Las memorias eeprom:

-Sus datos permanecen cuando se corta la alimentación.

-Su lectura es rápida. Pero la escritura es relativamente lenta (ms).

- Los ciclos de escritura en una eeprom son limitados. En el caso de los microcontroladores AVR están limitadas según el fabricante a 100.000 ciclos. Por ello serán utilizadas para guardar datos que no se actualicen en la ejecución de programa, sino cuando ocurra un evento externo.

En la eeprom del WK0200 están las tarjetas dadas de alta en ese remoto además de las tarjetas maestras y supermaestra del sistema.

En la eeprom del WK0500 están los remotos o esclavos dados de alta.

0x20 🡪 Abrir puerta.

Al ejecutar este comando abrimos desde el terminal una puerta. El comando debe ir acompañado del remoto que queremos abrir.

0x21 🡪 Alta de tarjeta normal.

Con este comando damos de alta una tarjeta en un remoto Winkel 0200. En la eeprom de este se guardará el número (4 Bytes) de esta tarjeta.

0x22 🡪 Alta de tarjeta maestra.

Este comando es análogo al anterior, pero se van a dar de alta en todos los remotos que haya en el sistema, es decir dados de alta en la eeprom del maestro.

0x23 🡪 Alta de tarjeta supermaestra.

El formato de este comando es como el anterior. La única diferencia entre una tarjeta maestra y una supermaestra, es que la súper maestra abre puertas bloqueadas, mientras que la maestra no lo hace. La supermaestra abrirá siempre las puertas.

0x24 🡪 Dar de baja una tarjeta.

Al dar de baja una tarjeta lo que haremos es poner a “0” el bit de habilitación de una tarjeta. Sabemos que en la eeprom se reservan 5 bytes por cada tarjeta. Los cuatro primeros son el número de la tarjeta, y el último byte nos da información del tipo de tarjeta y si está o no habilitada. De esta forma podremos discernir de una tarjeta bloqueada a una tarjeta externa que nunca estuvo dada de alta. Cuando una tarjeta no esté dada de alta, recorrerá todos los bloques de 5 bytes y si no la encuentra, entonces devolverá 0xFF .Tiene el formato siguiente:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| X | X | **T1** | **T0** | X | X | X | **H** |

T1 y T0 🡪 00 tarjeta NORMAL.

01 tarjeta MAESTRA.

10 tarjeta SUPERMAESTRA.

H 🡪 0 tarjeta no habilitada (pero dada de alta).

1 tarjeta habilitada.

0x25 🡪 Bloquear una puerta.

Consistirá en poner a 0 el bit de habilitación de una puerta. Sabemos que está en el l último byte de los 5 descritos anteriormente.

0x26 🡪 Desbloquear una puerta.

0x27 🡪 Consultar puerta.

0x28 🡪 Borrado de un esclavo.

Borra la eeprom del winkel 0200. Con ello no tendrá en ese esclavo ninguna tarjeta que la pueda abrir.

0x29 🡪 Dar de alta un esclavo.

0x2A 🡪 Leer un esclavo.

0x2B 🡪 Actualizar hora en el reloj.

0x2C 🡪 Borra los registros MODBUS en remoto.

0x2E 🡪 Leer datos de ultima tarjeta.

0x2F 🡪 Leer terminal ultima tarjeta pasada en remoto.

7.- ESTRUCTURA DE DATOS EN LOS HOLDING REGISTERS.

Utilizaremos un array de unsigned int (16 bits) para poder trabajar con ellos. Cada campo descrito a continuación dispone de 8 bits. **Los dos primeros campos son siempre los mismos**, siendo fijos para todos los comandos. En estos diremos el terminal que hizo una petición u orden, a que remoto está dirigido y el canal (en este subproyecto solo hay un canal, ya que cada remoto controla solo una puerta).

HR [0] 🡪

|  |  |
| --- | --- |
| terminal | comando |

HR [1] 🡪

|  |  |
| --- | --- |
| Número de remoto | canal |

Para el caso de **dar de alta/baja una tarjeta** utilizaremos los HR 2 y 3, donde van los 4 bytes significativos de la tarjeta RFID.

HR [2] 🡪

|  |  |
| --- | --- |
| Byte alto | Byte |

HR [3] 🡪

|  |  |
| --- | --- |
| Byte | Byte bajo |

Para el caso de querer **actualizar o leer la hora** necesitamos 4 HR, que empezaremos en el HR2 hasta el HR5.Al ser comandos distintos no hay problema en machacar los datos del comando anterior.

HR [2] 🡪

|  |  |
| --- | --- |
| Hora | Min |

HR [3] 🡪

|  |  |
| --- | --- |
| Segundos | Día de la semana |

HR [4] 🡪

|  |  |
| --- | --- |
| Día | Mes |

HR [5] 🡪

|  |  |
| --- | --- |
| Ano | Centuria |

El formato de la hora es un valor entero entre 0 y 23.

El formato de los minutos es entre 0 y 59.

El formato de los segundos es de 00 a 59.

El formato de día de la semana está entre 0(domingo) y 6 (sábado).

El formato de día es de 1 a 31.

El formato de mes es de 1 a 12.

El formato de centuria es 🡪0, siglo 21(2000+ ano)

🡪1, siglo 20 (1900 + ano)

Para **devolver el valor de la última tarjeta leída**, utilizaremos los HR 6,7 y 8.

HR[6] 🡪

|  |  |
| --- | --- |
| Byte alto | Byte |

HR[7] 🡪

|  |  |
| --- | --- |
| Byte | Byte bajo |

HR[8] 🡪

|  |  |
| --- | --- |
|  |  |

En este HR 8 está si la puerta está bloqueada, y si la tarjeta era supermaestra, maestra, o normal. Además devolverá FFFF si la tarjeta no está dada de alta.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | x | x | x | x | x | xx | x | **B** | | |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | x | x | **T1** | **T0** | x | x | x | **H** | |

B 🡪 0 puerta no bloqueada.

1 puerta bloqueada.

T1 y T0 🡪 00 tarjeta NORMAL.

01 tarjeta MAESTRA.

10 tarjeta SUPERMAESTRA.

H 🡪 0 tarjeta no habilitada( pero dada de alta).

1 tarjeta habilitada.

Cuando se pase una tarjeta que sea externa al sistema, es decir, que no esté dada de alta, en el HR\_8 se leerá 0x00FF.

También tenemos que tener un **historial de los eventos** que han ocurrido en el sistema; para saber el momento en que ocurrió utilizaremos el comando 0x2E, que leerá en los HR9-13. Esta lectura la hace de forma secuencial según un tiempo aproximado de unas decimas de segundo. Se puede modificar el tiempo si vemos que provoca colisiones en el bus TCP. El formato es el siguiente.

HR[9] 🡪

|  |  |
| --- | --- |
| Hora | Min |

HR[10] 🡪

|  |  |
| --- | --- |
| Segundos | Día de la semana |

HR[11] 🡪

|  |  |
| --- | --- |
| Día | Mes |

HR[12] 🡪

|  |
| --- |
| Año, en formato de 0 hasta año actual( 11 bits->2047) |

HR[13] 🡪

|  |  |
| --- | --- |
|  | Numero de remoto |

Observemos que el formato del año es diferente al de actualizar la hora.

8 .- PROGRAMA EN LOS REMOTOS O ESCLAVOS.

En cada uno de los esclavos (winkel 201) se ha de grabar este programa con la única diferencia del número de esclavo. En este caso vemos que la dirección ModBus de este esclavo es la 60.

//WK0201 -> ATMEGA 328P -> Graba datos tarjeta en EEPROM (1k - 0 a 0x3ff)

#include <SimpleModbusSlave.h>

#include <Wire.h>

#include <EEPROM.h>

#define ADDR\_RFID 0x41 //Direccion I2C del lector RFID en el WK0200

#define BAUD 19200

#define DirMB 60 // Se ha de cambiar para cada remoto

#define EP\_punteroTarjetas 20

#define EP\_puertabloqueada 19

enum

{ COMANDO, ARG1, ARG2, ARG3, ARG4, ARG5, ARG6, ARG7, ARG8, ARG9, ARG10, TOTAL\_ERRORS, TOTAL\_REGS\_SIZE };

unsigned int holdingRegs[TOTAL\_REGS\_SIZE];

boolean rfid=false, abrir=false, puerta\_abierta=false, controlar=false;

unsigned long numero, numero\_tarjeta=0, time, time2;

void setup()

{

Wire.begin(); //Inicia I2C

rfidReleOff(); //Desconectar rele

for (int i=0; i<TOTAL\_REGS\_SIZE; i++) { holdingRegs[i]=0; }

modbus\_configure (BAUD, DirMB,0,TOTAL\_REGS\_SIZE);

if (EEPROM.read(EP\_puertabloqueada)==0xFF)

{

EEPROM.write(EP\_puertabloqueada, 0x00); //inicializar puerta desbloqueada

}

}

void loop()

{

if (puerta\_abierta&&(millis()>(time+2000)))

{

rfidReleOff();

puerta\_abierta=false;

}

holdingRegs[COMANDO]=0xFFFF;

holdingRegs[TOTAL\_ERRORS]=modbus\_update(holdingRegs);

if(holdingRegs[COMANDO]!=0xFFFF)

{

ProcesarComandoMB();

}

CompruebaRFID();

if (rfid==true)

{

boolean encontrada=false;

unsigned char info;

unsigned int punterotarjetas=EP\_punteroTarjetas;

unsigned long n;

rfid=abrir=false;

if ((controlar)&&(millis()<(time2+5000)))

{

anadirtarjetaEEPROM(numero\_tarjeta,0);

controlar=false;

}

else

{

controlar=false;

while((EEPROM.read(punterotarjetas+4)!=0xFF)&&(!encontrada))

{

n=leer\_numero\_tarjeta(punterotarjetas);

info=EEPROM.read(punterotarjetas+4);

if (n==numero\_tarjeta)

{

encontrada=true;

switch(info)

{

case 0x21: //es supermaestra y habilitada

abrir=true; controlar=true; time2=millis();

break;

case 0x11: //es maestra y habilitada

case 0x01: //es normal y habilitada

if (!puerta\_bloqueada()) { abrir=true; }

break;

}

}

punterotarjetas=punterotarjetas+5;

}

if (puerta\_bloqueada())

{

holdingRegs[ARG8]=0x0100; //puerta bloqueada

}

else

{

holdingRegs[ARG8]=0x0000; //puerta desbloqueada

}

if (encontrada)

{

holdingRegs[ARG8]=holdingRegs[ARG8]|info; //tarjeta encontrada -> byte MSB: puerta bloqueada o no

} // -> byte LSB: tipo tarjeta+habilitada o no

else

{

holdingRegs[ARG8]=holdingRegs[ARG8]|0xFF; //tarjeta no encontrada -> byte MSB: puerta bloqueada o no

} // -> byte LSB: 0xFF (no dada de alta)

}

if (abrir)

{

rfidReleOn();

time = millis();

puerta\_abierta=true;

}

}

}

void ProcesarComandoMB(void)

{

unsigned int punterotarjetas=EP\_punteroTarjetas;

numero=(((unsigned long)holdingRegs[ARG1])<<16)|((unsigned long)holdingRegs[ARG2]);

switch(holdingRegs[COMANDO])

{

case 0x01: //Nueva Direccion ModBus

break;

case 0x02: //Nuevos Parametros COM del ModBus

break;

case 0x03: //Validar los nuevos parametros ModBus

break;

case 0x20: //Abrir puerta de forma remota

rfidReleOn();

time = millis();

puerta\_abierta=true;

break;

case 0x21: //Dar de alta tarjeta normal

anadirtarjetaEEPROM(numero,0);

break;

case 0x22: //Dar de alta tarjeta maestra

anadirtarjetaEEPROM(numero,1);

break;

case 0x23: //Dar de alta tarjeta supermaestra

anadirtarjetaEEPROM(numero,2);

break;

case 0x24: //Dar de baja tarjeta (da igual el tipo)

dardebajatarjetaEEPROM(numero);

break;

case 0x25: //Bloquear puerta

EEPROM.write(EP\_puertabloqueada,0x01);

break;

case 0x26: //Desbloquear puerta

EEPROM.write(EP\_puertabloqueada,0x00);

break;

case 0x27: //Consultar puertas

break;

case 0x28: //Borrado total cerradura

for (int i=0; i<TOTAL\_REGS\_SIZE; i++)

{

holdingRegs[i]=0;

}

while (EEPROM.read(punterotarjetas+4)!=0xFF)

{

for (int i=0; i<5; i++)

{

EEPROM.write(punterotarjetas,0xFF);

punterotarjetas++;

}

}

break;

case 0x2c: //Borrar registros RTU con informacion ya leida -> continuacion comando TCP

holdingRegs[ARG6]=0;

holdingRegs[ARG7]=0;

holdingRegs[ARG8]=0;

break;

}

}

void CompruebaRFID (void)

{

int c=0;

byte codigo[5]={0,0,0,0,0};

Wire.requestFrom(ADDR\_RFID, 1);

if (Wire.available())

{

c=Wire.read();

if (c==0x6E) //Tarjeta detectada

{

rfid=true;

Wire.beginTransmission(ADDR\_RFID);

Wire.write(0x52);

Wire.endTransmission();

Wire.requestFrom(ADDR\_RFID,5);

for (int i=0; i<5; i++)

{

if (Wire.available())

{

codigo[i]=Wire.read(); //El primer byte ¿es el de mayor peso?

}

}

if (codigo[4]==0x4E) //seguna parte de la transmision del codigo

{

Wire.beginTransmission(ADDR\_RFID);

Wire.write(0x52);

Wire.endTransmission();

Wire.requestFrom(ADDR\_RFID,1);

if (Wire.available())

{

codigo[4]=Wire.read();

}

}

numero\_tarjeta= (((unsigned long)codigo[1])<<24)|(((unsigned long)codigo[2])<<16)|(((unsigned long)codigo[3])<<8)|((unsigned long)codigo[4]);

holdingRegs[ARG6]=(codigo[1]<<8)|codigo[2];

holdingRegs[ARG7]=(codigo[3]<<8)|codigo[4];

}

}

}

void rfidReleOn()

{

Wire.beginTransmission(ADDR\_RFID);

Wire.write(0x62);

Wire.write(0x01);

Wire.write((byte)0x00);

Wire.write((byte)0x00);

Wire.write((byte)0x00);

Wire.write((byte)0x00);

Wire.write((byte)0x00);

Wire.write(0x63);

Wire.endTransmission();

}

void rfidReleOff()

{

Wire.beginTransmission(ADDR\_RFID);

Wire.write(0x62);

Wire.write((byte)0x00);

Wire.write((byte)0x00);

Wire.write((byte)0x00);

Wire.write((byte)0x00);

Wire.write((byte)0x00);

Wire.write((byte)0x00);

Wire.write(0x62);

Wire.endTransmission();

}

boolean puerta\_bloqueada (void)

{

if (EEPROM.read(EP\_puertabloqueada)==0x01) return true;

else return false;

}

void anadirtarjetaEEPROM(unsigned long n\_leer, int tipo)

{

unsigned char info;

int escrito=0, resto, punterotarjetas=EP\_punteroTarjetas;

unsigned long n;

do

{

if (EEPROM.read(punterotarjetas+4)!=0xFF)

{

n=leer\_numero\_tarjeta(punterotarjetas);

info=EEPROM.read(punterotarjetas+4);

if (n==n\_leer)

{

EEPROM.write(punterotarjetas+4,((tipo<<4)|0x01)); //actualizar info

escrito=1;

}

punterotarjetas=punterotarjetas+5;

}

else

{

punterotarjetas=punterotarjetas+4;

for (int x=0; x<4; x++)

{

resto=n\_leer%256;

n\_leer=n\_leer/256;

punterotarjetas--;

EEPROM.write(punterotarjetas,resto); //escribir numero tarjeta

}

punterotarjetas=punterotarjetas+4;

EEPROM.write(punterotarjetas,((tipo<<4)|0x01)); //escribir info

escrito=1;

}

}

while(escrito==0);

}

void dardebajatarjetaEEPROM (unsigned long n\_leer)

{

unsigned char info;

unsigned int punterotarjetas=EP\_punteroTarjetas;

unsigned long n;

while (EEPROM.read(punterotarjetas+4)!=0xFF)

{

n=leer\_numero\_tarjeta(punterotarjetas);

info=EEPROM.read(punterotarjetas+4);

if (n==n\_leer)

{

EEPROM.write(punterotarjetas+4,(info&0xFE)); //deshabilitar

}

punterotarjetas=punterotarjetas+5;

}

}

unsigned long leer\_numero\_tarjeta (int punt)

{

unsigned long leer=0;

leer=EEPROM.read(punt);

punt++;

leer=EEPROM.read(punt)|(leer<<8);

punt++;

leer=EEPROM.read(punt)|(leer<<8);

punt++;

leer=EEPROM.read(punt)|(leer<<8);

return leer;

}

9 .- PROGRAMA EN EL MAESTRO WINKEL 0500.

//WK0500 -> ATMEGA 2560 -> Graba nº esclavos en EPROM (4k -> 0 a 0x0FFF)

#define debug

#include <SPI.h>

#include <Ethernet.h>

#include "Mudbus.h"

#include <EEPROM.h>

//#include <SimpleModbusMaster.h> //Master ModBuus RTU

#include <Smm\_Juan.h>

#include <Wire.h> //para el I2C del reloj

#include <Rtc\_Pcf8563.h>

#define EP\_punteroesclavos 20

#define intervalotiempo 1000

Mudbus MbTCP;

Packet remotos[1];

packetPointer remoto=&remotos[0];

unsigned int regMbRTU[25];

unsigned int resultadoRTU=0;

boolean cambio=true,comandoRTU=false,lecturaactivada=false;

Rtc\_Pcf8563 rtc;

unsigned char hora, minutos, segundos;

unsigned char dia, dia\_semana, mes, centuria=0, ano;

unsigned int punteroesclavos=EP\_punteroesclavos;

unsigned int indiceesclavos=EP\_punteroesclavos+1, indicealmacen;

unsigned int indicedevolverHR=EP\_punteroesclavos+1, indicealmacen2;

unsigned int almacen [10][8];

unsigned long tiempolectura, nuevotiempo;

int z=0;

void setup ()

{

uint8\_t mac[]= {0x90, 0xa2, 0xda, 0x00, 0x51, 0x06};

uint8\_t ip[]={10, 0, 0, 166};

uint8\_t gateway[]={192,168,1,1};

uint8\_t subnet[]={255,255,0,0};

Ethernet.begin(mac, ip, gateway, subnet);

remoto->register\_array = regMbRTU; //inicializacion ModBus RTU

modbus\_configure(19200,1000,5,1,0,remotos,1);

MbTCP.R[0]=0xFFFF;

if (EEPROM.read(EP\_punteroesclavos)==0xFF) { EEPROM.write(EP\_punteroesclavos,0x00); }

else { punteroesclavos=EEPROM.read(EP\_punteroesclavos)+EP\_punteroesclavos; }

Wire.begin(); //inicia I2C

//rtc.initClock(); //inicializo reloj

tiempolectura=millis();

#ifdef debug

Serial.begin(9600);

#endif

}

void loop ()

{

if (!comandoRTU)

{

MbTCP.Run();

if (MbTCP.R[0]!=0xFFFF)

{

ejecutacomandoTCP();

MbTCP.R[0]=0xFFFF;

}

else

{

nuevotiempo=millis();

if (nuevotiempo > (tiempolectura+intervalotiempo)) //si hay muchos esclavos disminuir el tiempo

{

if (EEPROM.read(EP\_punteroesclavos)!=0)

{

MbTCP.R[1]=(EEPROM.read(indiceesclavos)<<8);

MbTCP.R[0]=0x002F;

ejecutacomandoTCP();

MbTCP.R[0]=0xFFFF;

tiempolectura=nuevotiempo;

}

}

else

{

if (tiempolectura > (nuevotiempo+intervalotiempo)) //caso desborde temporizador

{

tiempolectura=nuevotiempo;

}

}

}

}

/\* --------------------------------------------------------------------

if (regMbRTU[0]!=0xFFFF) //Esta versión funciona... no borrar

{

resultadoRTU=modbus\_update(remotos,true);

if (resultadoRTU!=0) regMbRTU[0]=0xFFFF;

}

--------------------------------------------------------------------- \*/

if (comandoRTU) //tambien funciona -> Usar esta para poder leer tarjeta

{

resultadoRTU=modbus\_update(remotos,true);

if (resultadoRTU!=0) //0->en proceso, 1->OK, 2->error

{

comandoRTU=false;

#ifdef debug

Serial.print("RTU: ");

for (int b=0; b<10; b++)

{

Serial.print(regMbRTU[b],HEX);

Serial.print(" ");

}

Serial.println(" ");

#endif

if ((regMbRTU[6]!=0)&&(regMbRTU[7]!=0)) //hay tarjeta que leer

{

indicealmacen=indiceesclavos-(EP\_punteroesclavos+1);

almacen[indicealmacen][0]=EEPROM.read(indiceesclavos);

almacen[indicealmacen][1]=regMbRTU[6]; //numero tarjeta palabra alta

almacen[indicealmacen][2]=regMbRTU[7]; //numero tarjeta palabra baja

almacen[indicealmacen][3]=regMbRTU[8]; //informacion extra

regMbRTU[6]=regMbRTU[7]=regMbRTU[8]=0;

//Leer hora y fecha

rtc.getTime();

hora=rtc.getHour();

minutos=rtc.getMinute();

segundos=rtc.getSecond();

rtc.getDate();

dia=rtc.getDay();

dia\_semana=rtc.getWeekday();

mes=rtc.getMonth();

ano=rtc.getYear();

almacen[indicealmacen][4]=(hora<<8)|minutos;

almacen[indicealmacen][5]=(segundos<<8)|dia\_semana;

almacen[indicealmacen][6]=(dia<<8)|mes;

if (centuria==0) { almacen[indicealmacen][7]=ano+2000; }

else { almacen[indicealmacen][7]=ano+1900; }

MbTCP.R[0]=0x002c; //borrar registros RTU en esclavo

MbTCP.R[1]=(EEPROM.read(indiceesclavos)<<8); //¿Hace falta?

ejecutacomandoTCP();

MbTCP.R[0]=0xFFFF;

}

if (lecturaactivada)

{

lecturaactivada=false;

indiceesclavos++;

if (indiceesclavos>punteroesclavos){ indiceesclavos=EP\_punteroesclavos+1; }

}

}

}

}

void ejecutacomandoTCP()

{

unsigned char x;

switch(MbTCP.R[0]&0x00FF) //De momento no se maneja el identificador del terminal (PC, tablet, etc.), solo comando TCP

{

case 0x20: //Abrir de forma remota //Comando TCP

regMbRTU[0]=0x20; //Comando RTU en posicion 0

remoto->id=MbTCP.R[1]>>8; //direccion modbus esclavo RTU

remoto->function=PRESET\_MULTIPLE\_REGISTERS;

remoto->address=0;

remoto->no\_of\_registers=5;

comandoRTU=true;

break;

case 0x21: //Dar de alta tarjeta //Comando TCP

regMbRTU[0]=0x21; //Comando RTU en posicion 0

regMbRTU[1]=MbTCP.R[2]; //numero tarjeta -> empieza por byte alto

regMbRTU[2]=MbTCP.R[3];

remoto->id=MbTCP.R[1]>>8; //direccion modbus esclavo RTU

remoto->function=PRESET\_MULTIPLE\_REGISTERS;

remoto->address=0;

remoto->no\_of\_registers=5;

comandoRTU=true;

break;

case 0x22: //Dar de alta tarjeta maestra //Comando TCP

regMbRTU[0]=0x22; //Comando RTU en posicion 0

regMbRTU[1]=MbTCP.R[2]; //numero tarjeta -> empieza por byte alto

regMbRTU[2]=MbTCP.R[3];

remoto->id=MbTCP.R[1]>>8; //direccion modbus esclavo RTU

remoto->function=PRESET\_MULTIPLE\_REGISTERS;

remoto->address=0;

remoto->no\_of\_registers=5;

comandoRTU=true;

break;

case 0x23: //Dar de alta tarjeta super maestra //Comando TCP

regMbRTU[0]=0x23; //Comando RTU en posicion 0

regMbRTU[1]=MbTCP.R[2]; //numero tarjeta -> empieza por byte alto

regMbRTU[2]=MbTCP.R[3];

remoto->id=MbTCP.R[1]>>8; //direccion modbus esclavo RTU

remoto->function=PRESET\_MULTIPLE\_REGISTERS;

remoto->address=0;

remoto->no\_of\_registers=5;

comandoRTU=true;

break;

case 0x24: //Dar de baja tarjeta //Comando TCP

regMbRTU[0]=0x24; //Comando RTU en posicion 0

regMbRTU[1]=MbTCP.R[2]; //numero tarjeta -> empieza por byte alto

regMbRTU[2]=MbTCP.R[3];

remoto->id=MbTCP.R[1]>>8; //direccion modbus esclavo RTU

remoto->function=PRESET\_MULTIPLE\_REGISTERS;

remoto->address=0;

remoto->no\_of\_registers=5;

comandoRTU=true;

break;

case 0x25: //Bloquear puerta //Comando TCP

regMbRTU[0]=0x25; //Comando RTU en posicion 0

remoto->id=MbTCP.R[1]>>8; //direccion modbus esclavo RTU

remoto->function=PRESET\_MULTIPLE\_REGISTERS;

remoto->address=0;

remoto->no\_of\_registers=5;

comandoRTU=true;

break;

case 0x26: //Desbloquear puerta //Comando TCP

regMbRTU[0]=0x26; //Comando RTU en posicion 0

remoto->id=MbTCP.R[1]>>8; //direccion modbus esclavo RTU

remoto->function=PRESET\_MULTIPLE\_REGISTERS;

remoto->address=0;

remoto->no\_of\_registers=5;

comandoRTU=true;

break;

case 0x27: //Consultar puerta //Comando TCP

break;

case 0x28: //Borrar cerradura //Comando TCP

regMbRTU[0]=0x28; //Comando RTU en posicion 0

remoto->id=MbTCP.R[1]>>8; //direccion modbus esclavo RTU

remoto->function=PRESET\_MULTIPLE\_REGISTERS;

remoto->address=0;

remoto->no\_of\_registers=5;

comandoRTU=true;

break;

case 0x29: //Dar de alta esclavo //Comando TCP

punteroesclavos++;

EEPROM.write(punteroesclavos, MbTCP.R[1]>>8);

x=EEPROM.read(EP\_punteroesclavos);

x++;

EEPROM.write(EP\_punteroesclavos,x);

break;

case 0x2A: //Pasar al terminal número remotos total y los que son

for (x=EP\_punteroesclavos; x<=punteroesclavos; x++)

{

MbTCP.R[x]=EEPROM.read(x);

} //La informacion estara a partir del MbTCP.R 20

break;

case 0x2B: //Actualizar hora en el reloj

rtc.initClock(); //inicializo reloj

hora=MbTCP.R[2]>>8;

minutos=MbTCP.R[2]&0x00ff;

segundos=MbTCP.R[3]>>8;

dia\_semana=MbTCP.R[3]&0x00ff;

dia=MbTCP.R[4]>>8;

mes=MbTCP.R[4]&0x00ff;

centuria=MbTCP.R[5]>>8; //0 es 2000, 1 es 1900

ano=MbTCP.R[5]&0x00ff;

rtc.setTime(hora,minutos,segundos);

rtc.setDate(dia,dia\_semana,mes,centuria,ano);

break;

case 0x2C: //Borra registros ModBus en esclavo //Comando TCP

regMbRTU[0]=0x2C;

remoto->id=MbTCP.R[1]>>8; //direccion modbus esclavo RTU

remoto->function=PRESET\_MULTIPLE\_REGISTERS;

remoto->address=0;

remoto->no\_of\_registers=10;

comandoRTU=true;

break;

case 0x2E: //Devolver tarjeta leida al terminal PC -> recoger del array almacen y dejar en HR

if (EEPROM.read(EP\_punteroesclavos)!=0)

{

indicealmacen2=indicedevolverHR-(EP\_punteroesclavos+1);

MbTCP.R[6]=almacen[indicealmacen2][1]; //numero tarjeta

MbTCP.R[7]=almacen[indicealmacen2][2]; //numero tarjeta

MbTCP.R[8]=almacen[indicealmacen2][3]; //informacion extra

MbTCP.R[9]=almacen[indicealmacen2][4]; //informacion hora

MbTCP.R[10]=almacen[indicealmacen2][5]; //informacion hora

MbTCP.R[11]=almacen[indicealmacen2][6]; //informacion fecha

MbTCP.R[12]=almacen[indicealmacen2][7]; //informacion fecha

MbTCP.R[13]=almacen[indicealmacen2][0]; //numero esclavo-remoto

MbTCP.R[14]=indicedevolverHR;

indicedevolverHR++;

if (indicedevolverHR>punteroesclavos)

{

indicedevolverHR=EP\_punteroesclavos+1;

}

}

break;

case 0x2F: //Leer ultima tarjeta pasada en remoto //Comando TCP

remoto->id=MbTCP.R[1]>>8; //direccion modbus esclavo RTU

remoto->function=READ\_HOLDING\_REGISTERS;

remoto->address=0;

remoto->no\_of\_registers=10;

comandoRTU=true;

lecturaactivada=true;

break;

}

#ifdef debug

Serial.print("TCP: ");

for (int b=0; b<15; b++)

{

Serial.print(MbTCP.R[b],HEX);

Serial.print(" ");

}

Serial.println(" ");

#endif

}

10.- PROGRAMA JAVA LECTOR DE TARJETAS 1.0.

**DIAGRAMA DE CLASES PROYECTO ARDUINO**

BtDesbloquearPuertaActionListener



NL : String



BtDesbloquearPuertaActionListener()



BtDesbloquearPuertaActionListener()



actionPerformed()

(from LectorTarje...

Path

(from file)

1



1()



actionPerformed()

(from LectorTarjetas)

BtAbrirPuertaActionListener



NL : String



BtAbrirPuertaActionListener()



BtAbrirPuertaActionListener()



actionPerformed()

(from LectorTarje...

BtActualizarHoraActionListener



BtActualizarHoraActionListener()



BtActualizarHoraActionListener()



actionPerformed()

(from LectorTarjetas)

BtAltaEsclavoActionListener



NL : String



BtAltaEsclavoActionListener()



BtAltaEsclavoActionListener()



actionPerformed()

(from LectorTarje...

BtAltaTarjetaActionListener



NL : String



BtAltaTarjetaActionListener()



BtAltaTarjetaActionListener()



actionPerformed()

(from LectorTarje...

BtBajaTarjetaActionListener



NL : String



BtBajaTarjetaActionListener()



BtBajaTarjetaActionListener()



actionPerformed()

(from LectorTarje...

BtBloquearPuertaActionListener



NL : String



BtBloquearPuertaActionListener()



BtBloquearPuertaActionListener()



actionPerformed()

(from LectorTarje...

BtBorrarCerraduraActionListener



NL : String



BtBorrarCerraduraActionListener()



BtBorrarCerraduraActionListener()



actionPerformed()

(from LectorTarje...

BtBorrarVentanaInfoActionListener



BtBorrarVentanaInfoActionListener()



BtBorrarVentanaInfoActionListener()



actionPerformed()

(from LectorTarjetas)

BtLeerRegistrosActionListener



NL : String



BtLeerRegistrosActionListener()



BtLeerRegistrosActionListener()



actionPerformed()

(from LectorTarje...

MenuAbrirHistorialActionListener



MenuAbrirHistorialActionListener()



MenuAbrirHistorialActionListener()



actionPerformed()

(from LectorTarjetas)

MenuAbrirPuertaActionListener



MenuAbrirPuertaActionListener()



MenuAbrirPuertaActionListener()



actionPerformed()

(from LectorTarjetas)

MenuActualizarHoraActionListener



MenuActualizarHoraActionListener()



MenuActualizarHoraActionListener()



actionPerformed()

(from LectorTarjetas)

MenuAltaTarjetaActionListener



MenuAltaTarjetaActionListener()



MenuAltaTarjetaActionListener()



actionPerformed()

(from LectorTarjetas)

MenuBajaTarjetaActionListener



MenuBajaTarjetaActionListener()



MenuBajaTarjetaActionListener()



actionPerformed()

(from LectorTarjetas)

JTextArea



JTextArea()



JTextArea()



JTextArea()



JTextArea()



JTextArea()



JTextArea()



getUIClassID()



createDefaultModel()



setTabSize()



getTabSize()



setLineWrap()



getLineWrap()



setWrapStyleWord()



getWrapStyleWord()



getLineOfOffset()



getLineCount()



getLineStartOffset()



getLineEndOffset()



insert()



append()



replaceRange()



isManagingFocus()



processKeyEvent()



getRows()



setRows()



getRowHeight()



getColumns()



setColumns()



getColumnWidth()



getPreferredSize()



setFont()



paramString()



getScrollableTracksViewportWidth()



getPreferredScrollableViewportSize()



getScrollableUnitIncrement()



getAccessibleContext()

(from swing)

Comandos



NL : String



puerto : int



solicitudMultRegW : WriteMultipleRegistersRequest



respuestaMultRegW : WriteMultipleRegistersResponse



solicitudMultReg : ReadMultipleRegistersRequest



respuestaMultReg : ReadMultipleRegistersResponse



con : TCPMasterConnection



trans : ModbusTCPTransaction



accionesPuerta()



altaTarjeta()



bajaTarjeta()



inicializarEsclavos()



actualizarHora()



leerRegistros()



capturarEvento()



getRespuestaMultRegW()



setRespuestaMultRegW()

Evento



nRemoto : String



nTarjeta : String



hora : String



minuto : String



segundo : String



diaSemana : String



dia : String



mes : String



anyo : String



altaTarjeta : String



tipoTarjeta : String



habilitacionTarjeta : String



puertaBloqueada : String



Evento()



getAltaTarjeta()



setAltaTarjeta()



getTipoTarjeta()



setTipoTarjeta()



getHabilitacionTarjeta()



setHabilitacionTarjeta()



getPuertaBloqueada()



setPuertaBloqueada()



getnRemoto()



setnRemoto()



getnTarjeta()



setnTarjeta()



getHora()



setHora()



getMinuto()



setMinuto()



getSegundo()



setSegundo()



getDiaSemana()



setDiaSemana()



getDia()



setDia()



getMes()



setMes()



getAnyo()



setAnyo()

Fecha



hora : byte



minuto : byte



segundo : byte



diaSemana : byte



diaMes : byte



mes : byte



anyo : byte



horaTxt : String



minutoTxt : String



segundoTxt : String



diaSemanaTxt : String



diaMesTxt : String



mesTxt : String



anyoTxt : String



calendario : GregorianCalendar



Fecha()



getHora()



getMinuto()



getSegundo()



getDiaSemana()



getDiaMes()



getMes()



getAnyo()



getHoraTxt()



getMinutoTxt()



getSegundoTxt()



getDiaSemanaTxt()



getDiaSemanaTxt()



getDiaMesTxt()



getMesTxt()



getAnyoTxt()

JTextFieldLimit



serialVersionUID : long = 1



limit : int



toUppercase : boolean



JTextFieldLimit()



JTextFieldLimit()



insertString()

LectorTarjetas



NL : String



serialVersionUID : long = 1



temp : String



rutaFichero : String



delay : int



LectorTarjetas()



main()



getVentana()



setVentana()



access$0()



access$1()



access$2()



access$3()



access$4()



access$5()



access$6()



access$7()



access$8()



access$9()



access$10()



access$11()



access$12()



access$13()



access$14()



access$15()

-$ventana

this$0

this$0

this$0

this$0

this$0

this$0

this$0

this$0

this$0

this$0

this$0

this$0

this$0

this$0

this$0

this$0

-taInformacion

Fichero



NL : String



existe : boolean



Fichero()



isExiste()



setExiste()



crearFichero()



insertarDatos()



leerDatos()

-path

En el **anexo 1** se encuentra el código de cada una de las clases realizadas.

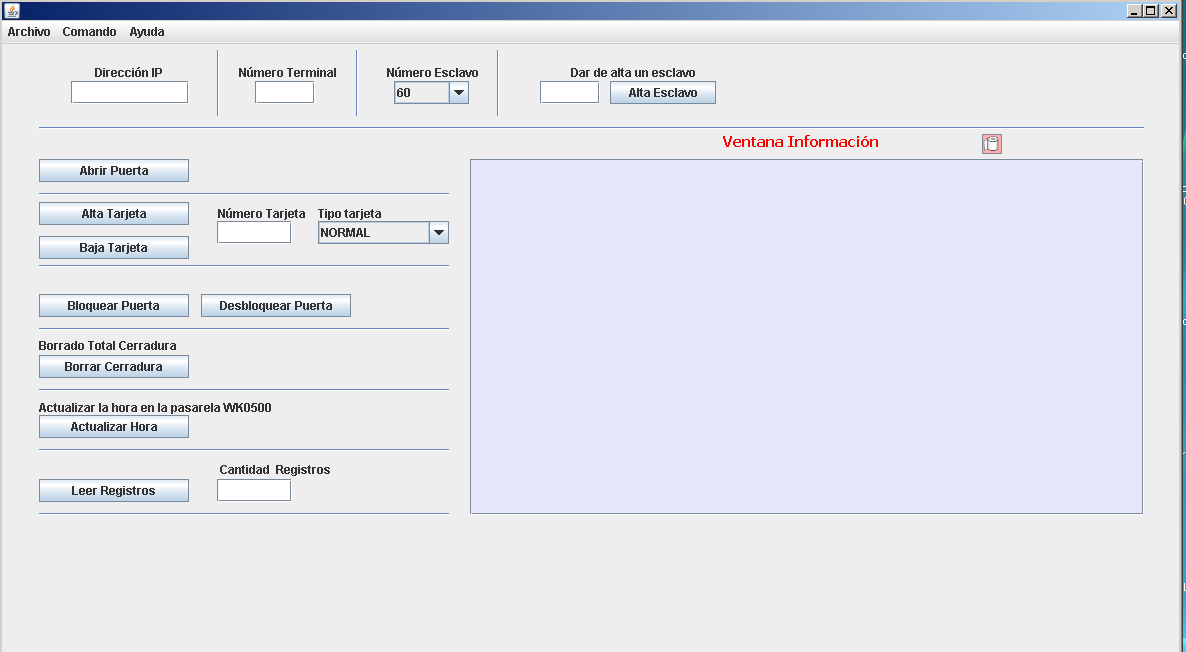
* CLASE LectorTarjetas

**public** **class** LectorTarjetas **extends** JFrame

Es la clase principal donde se encuentra la función MAIN. Desde esta clase se realiza una llamada al constructor de la clase para que se inicialicen todos los componentes que forman la ventana.

El aspecto que tiene la ventana lo podemos ver en la siguiente captura de pantalla:

Interface del programa



* CLASES para capturar los diferentes eventos desde la ventana principal del programa:

**private** **class** BtAbrirPuertaActionListener **implements** ActionListener

**private** **class** BtAltaTarjetaActionListener **implements** ActionListener

**private** **class** BtBajaTarjetaActionListener **implements** ActionListener

**private** **class** BtBloquearPuertaActionListener **implements** ActionListener

**private** **class** BtDesbloquearPuertaActionListener **implements** ActionListener

**private** **class** BtBorrarCerraduraActionListener **implements** ActionListener

**private** **class** BtAltaEsclavoActionListener **implements** ActionListener

**private** **class** BtActualizarHoraActionListener **implements** ActionListener

**private** **class** BtLeerRegistrosActionListener **implements** ActionListener

**private** **class** BtBorrarVentanaInfoActionListener **implements** ActionListener

Para gestionar el evento producido al pulsar sobre un botón implementamos el *listener* en una clase miembro definida dentro de la clase LectorTarjetas. Estas clases realizan las acciones de Abrir una puerta, dar de alta una tarjeta…etc.

* CLASES para capturar los eventos desde el menú Archivo

**private** **class** MenuAbrirHistorialActionListener **implements** ActionListener

En esta clase, se realiza la acción de abrir un cuadro de diálogo con el que podemos buscar el fichero de texto donde se guardan los diferentes eventos. Una vez abierto, podemos ver los datos en la ventana de información.

**private** **class** MenuGuardarHistorialActionListener **implements** ActionListener

En esta clase, también se realiza la acción de abrir un cuadro de diálogo, pero, en este caso, lo que se pretende es indicar la ubicación y el nombre del fichero de texto en el que guardaremos los eventos.

**private** **class** MenuSalirActionListener **implements** ActionListener

Realiza la acción de salir del programa.

* CLASES para capturar los eventos desde el menú Comando

**private** **class** MenuAbrirPuertaActionListener **implements** ActionListener

**private** **class** MenuAltaTarjetaActionListener **implements** ActionListener

**private** **class** MenuBajaTarjetaActionListener **implements** ActionListener

**private** **class** MenuBloquearPuertaActionListener **implements** ActionListener

**private** **class** MenuDesbloquearPuertaActionListener **implements** ActionListener

**private** **class** MenuBorrarCerraduraActionListener **implements** ActionListener

**private** **class** MenuLeerRegistrosActionListener **implements** ActionListener

**private** **class** MenuActualizarHoraActionListener **implements** ActionListener

En estas clases, creamos un objeto para llamar al *listener* que hay en las clases para capturar los diferentes eventos desde la ventana principal del programa.

* CLASES para capturar los eventos desde el menú Ayuda opción: Acerca de…

**private** **class** MntmAcercaDeActionListener **implements** ActionListener

**class** VentanaInfo **extends** JFrame

Aqui, se abre una ventana indicando los autores del proyecto y la versión del programa.

* CLASE que gestiona la creación de los ficheros utilizados para guardar en el disco duro los diferentes eventos producidos.

**public** **class** Fichero

Cuando ejecutamos el programa se crean dos ficheros:

**Fichero copias1.txt**

En este fichero guardamos información para realizar después una tabla informativa de los diferentes eventos.

**Fichero copias{FECHA}\_{HORA}.txt**

En este fichero guardamos todos los eventos.

Cada vez que arranca el programa crea un nuevo fichero con la fecha y la hora actual.

* CLASE que se encarga de ejecutar todos los comandos (Abrir puerta, Alta tarjeta…etc.). Esta clase maneja las librerías *jamod* importadas en el proyecto para comunicarnos con la pasarela WK0500 utilizando el protocolo *ModbusTCP*.

**public** **class** Comandos

Métodos utilizados en la clase:

**public** String accionesPuerta (String direccion, **byte** nTerminal, **byte** comando, **byte** nEsclavo) **throws** Exception

**public** String altaTarjeta(String direccion, **byte** nTerminal, **byte** comando, StringBuffer nRemoto, **int** dato1, **int** dato2, **int** dato3, **int** dato4) **throws** Exception

**public** String bajaTarjeta(String direccion, **byte** nTerminal, **byte** comando, **byte** nEsclavo, **int** dato1, **int** dato2, **int** dato3, **int** dato4) **throws** Exception

**public** StringBuffer inicializarEsclavos (String direccion, **byte** nTerminal, **byte** comando) **throws** Exception

**public** String actualizarHora (String direccion, **byte** nTerminal, **byte** comando) **throws** Exception

**public** String leerRegistros (String direccion, **byte** nTerminal, **byte** comando, **int** cantidad, **byte** nEsclavo) **throws** Exception

**public** String capturarEvento (String direccion, **byte** nTerminal, **byte** comando) **throws** Exception

* CLASE utilizada para que como máximo se introduzcan 8 dígitos en el campo de texto en el cual se debe de teclear el número de la tarjeta.

**public** **class** JTextFieldLimit **extends** PlainDocument

* CLASE para adaptar la hora y la fecha del sistema a nuestras necesidades.

**public** **class** Fecha

* CLASE implementada para capturar los eventos producidos en los remotos.

**public** **class** Evento

Al producirse un evento creamos un objeto de tipo Evento y guardamos la siguiente información:

**private** String nRemoto; //número de remoto

**private** String nTarjeta; //número de tarjeta

**private** String hora; //la hora en la que se produce el evento

**private** String minuto; //los minutos

**private** String segundo; //los segundos

**private** String diaSemana; //el dia de la semana (LUNES, MARTES…)

**private** String dia; //el dia del mes

**private** String mes; //el mes

**private** String anyo; //el año

**private** String altaTarjeta; //Si la tarjeta está dada de alta (o no)

**private** String tipoTarjeta; //Tipo de tarjeta (NORMAL, MAESTRA, SUPERMAESTRA)

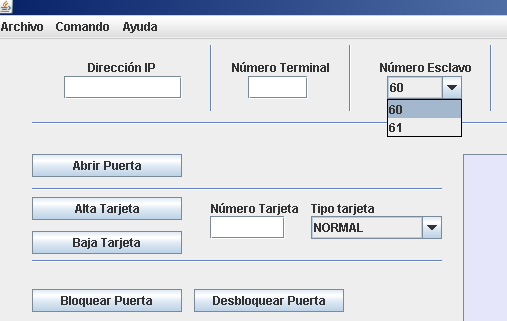
**private** String habilitacionTarjeta; //Si la tarjeta está habilitada (o no)

**private** String puertaBloqueada; ////Si la puerta está bloqueada (o no)

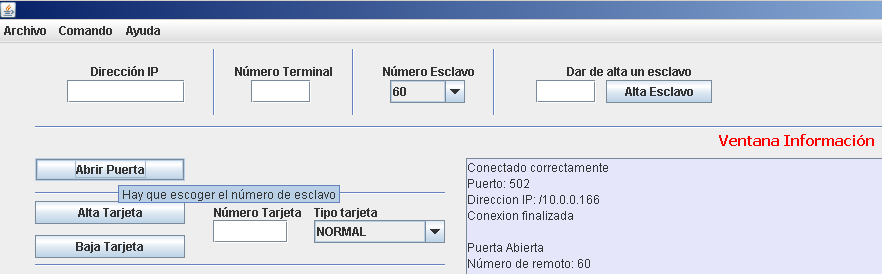
**Veamos ahora algunas capturas de pantalla del funcionamiento del programa:**

**Acción:** Abrir una puerta

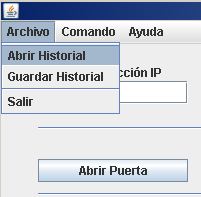
En primer lugar seleccionamos el número de remoto:



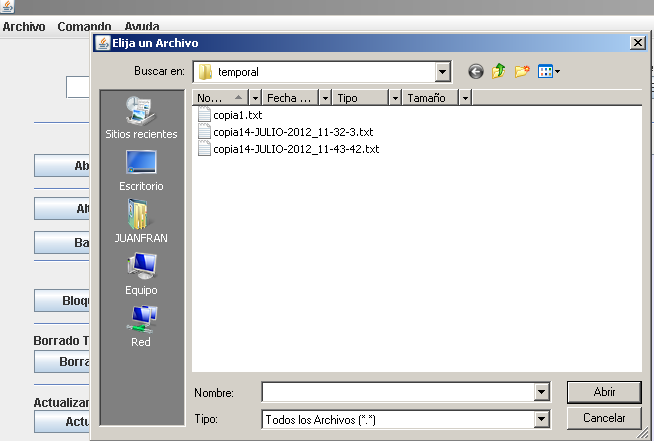
A continuación pulsamos sobre el botón Abrir Puerta:



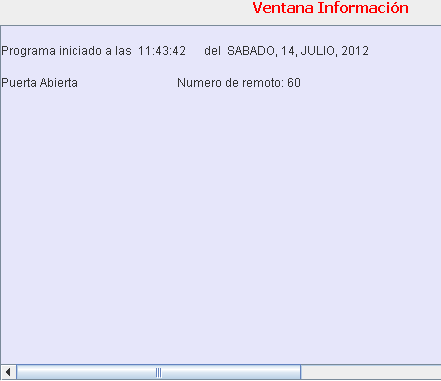
Seguidamente abrimos el fichero para ver si se ha guardado la información:



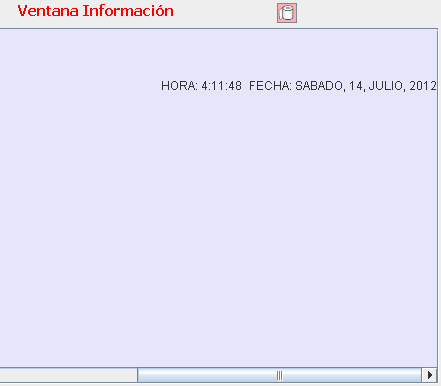
Se abre a continuación el cuadro de diálogo para buscar el fichero:



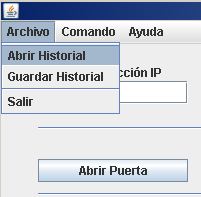
Seleccionamos el fichero y vemos el resultado en la ventana de información

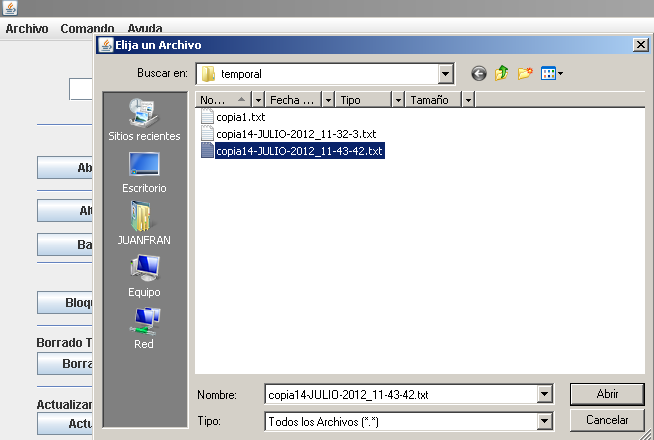


Si nos desplazamos con la barra de desplazamiento horizontal podemos ver la hora y la fecha del evento.



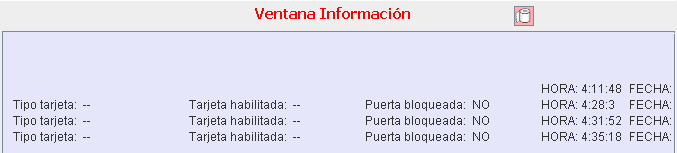
Veamos ahora que sucede si pasamos una tarjeta por un remoto:





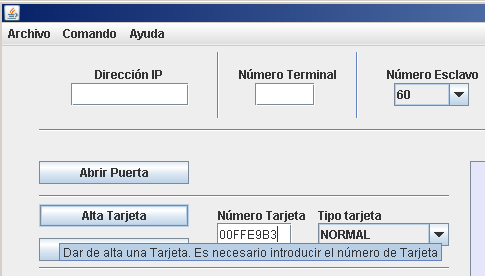
Resultado:



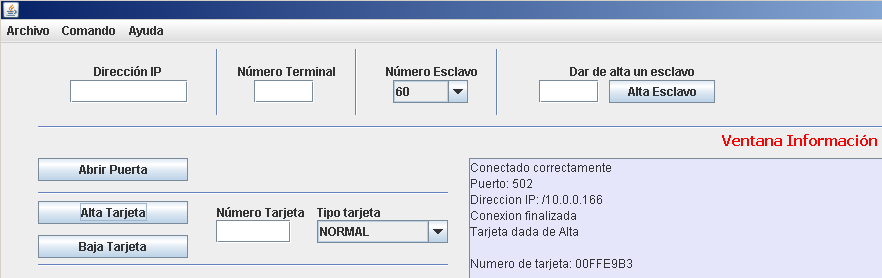


Como la tarjeta no está dada de alta la información de tipo tarjeta y tarjeta habilitada no está disponible.

Vamos a dar de alta una tarjeta en el remoto 60

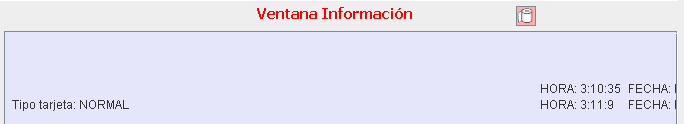


A continuación pulsamos sobre el botón Alta Tarjeta:



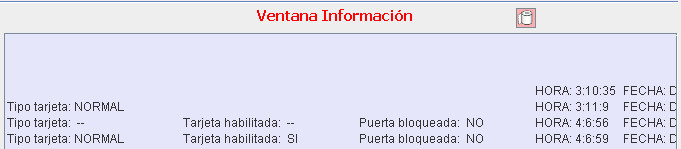
Seguidamente abrimos el fichero para ver si se ha guardado la información:



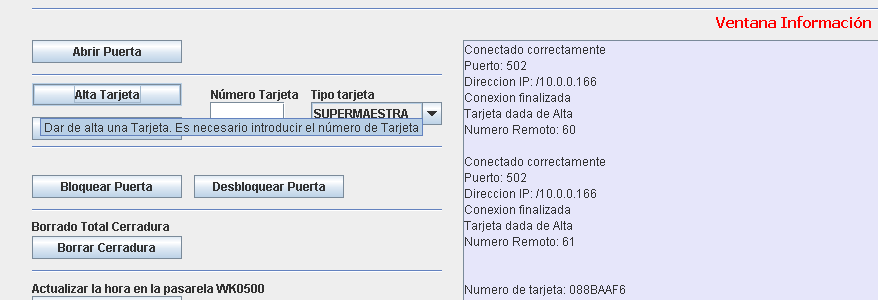


Pasemos seguidamente la tarjeta por los remotos 60 y 61:





Vamos a dar de alta una tarjeta NORMAL en el remoto 61 y a continuación daremos de alta una tarjeta MAESTRA y otra SUPERMAESTRA.



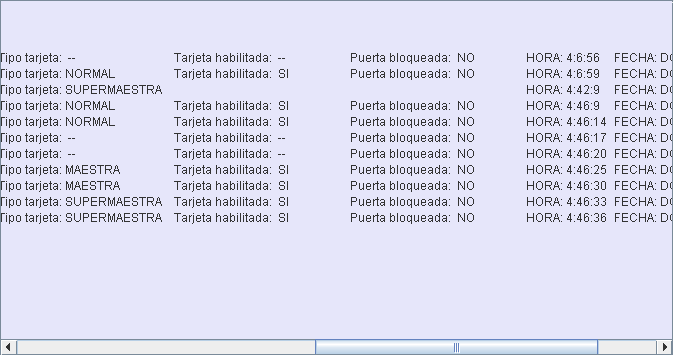
Pasemos las tarjetas por los remotos:

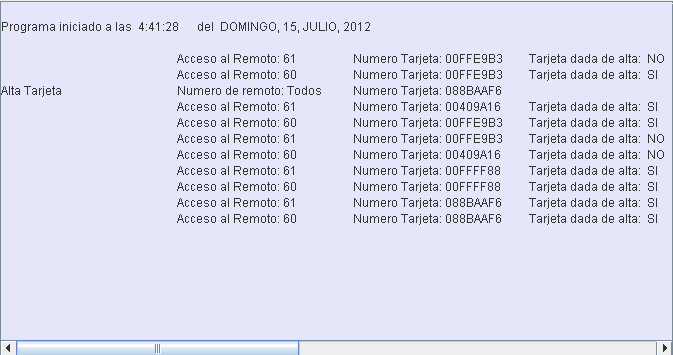
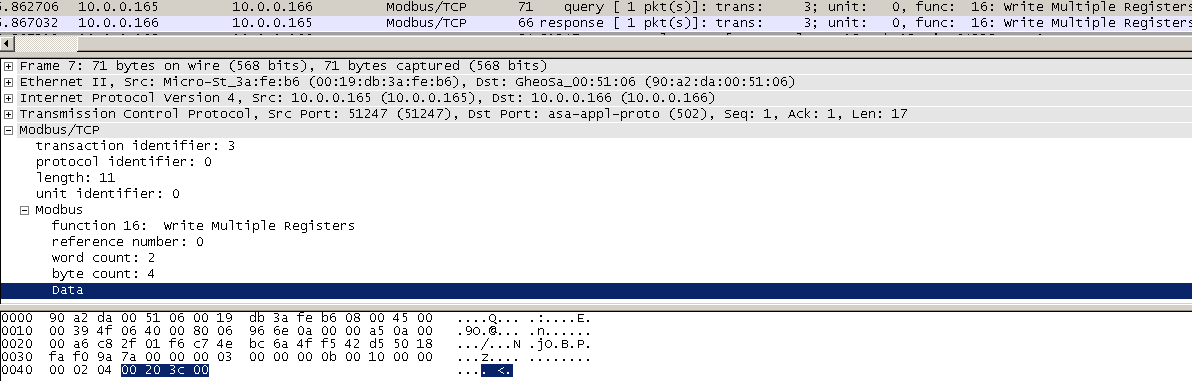
Resultado:

Veamos ahora las tramas que circulan entre el PC y la pasarela al realizar diferentes acciones:

**Acción:** Abrir Puerta:

Número de remoto: 60 (en hexadecimal = 3C)

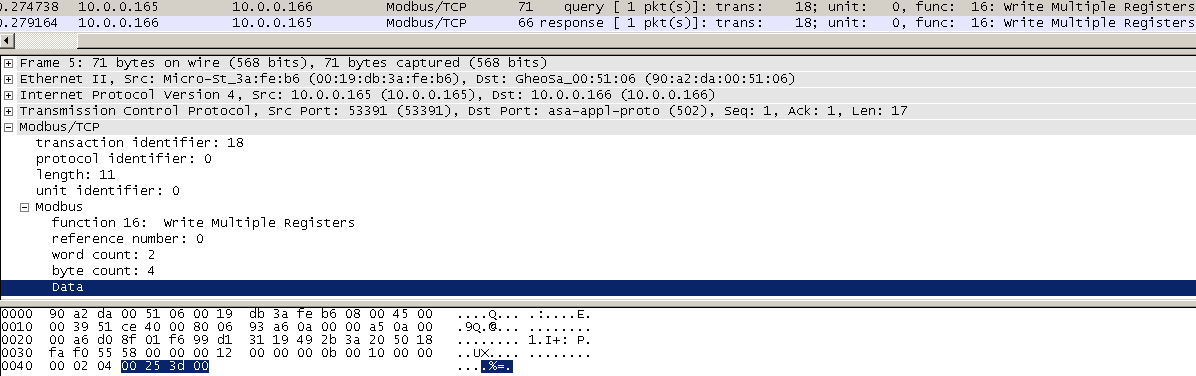
Comando: 20



**Acción:** Bloquear Puerta:

Número de remoto: 61 (en hexadecimal = 3D)

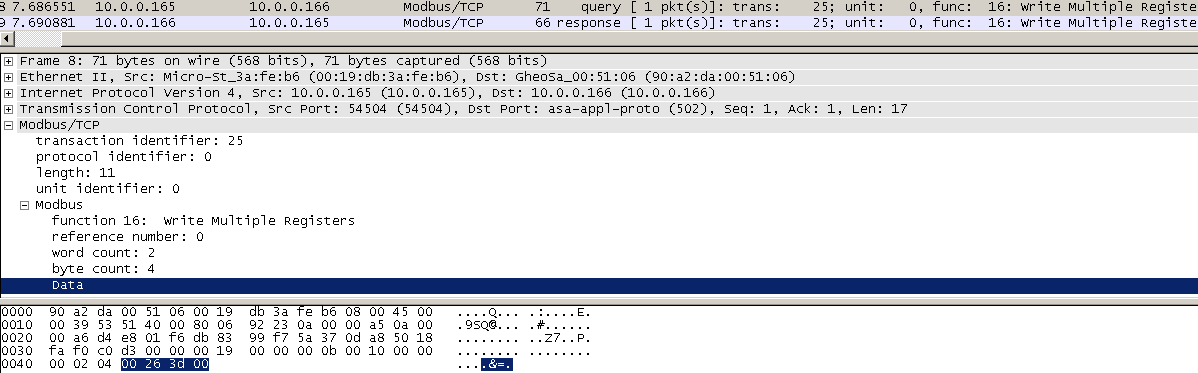
Comando: 25



**Acción:** Desbloquear Puerta:

Número de remoto: 61 (en hexadecimal = 3D)

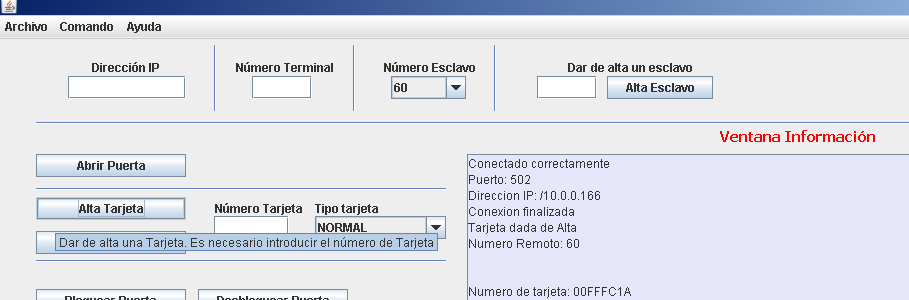
Comando: 26

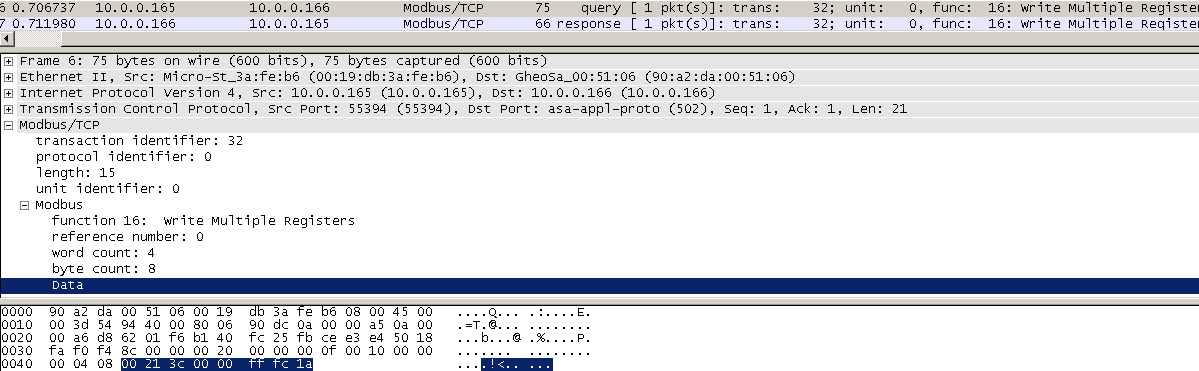


**Acción:** Alta Tarjeta NORMAL:

Número de remoto: 60 (en hexadecimal = 3C)

Comando: 21

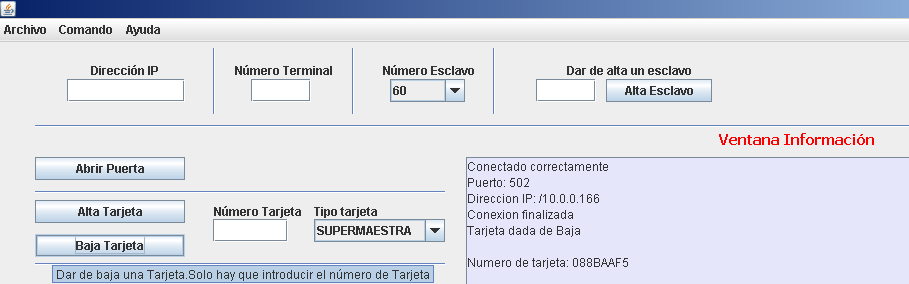


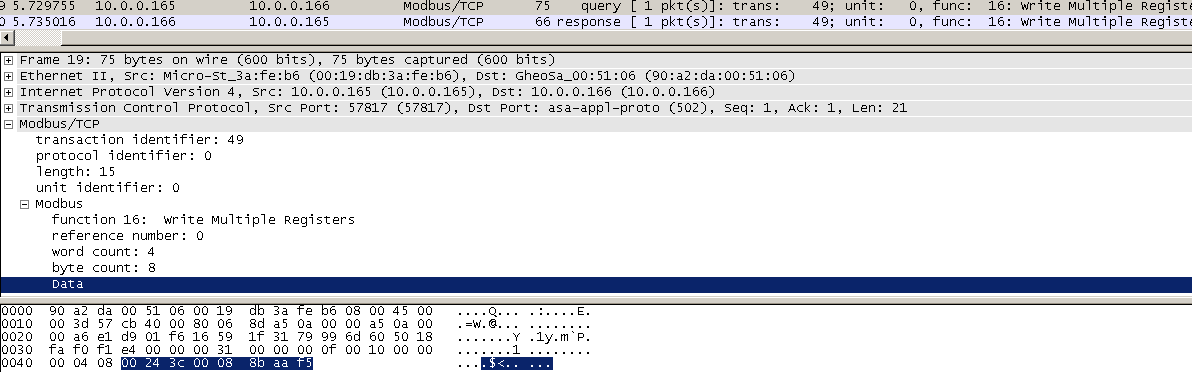


**Acción:** Baja Tarjeta SUPERMAESTRA:

Número de remoto: 60

Comando: 24

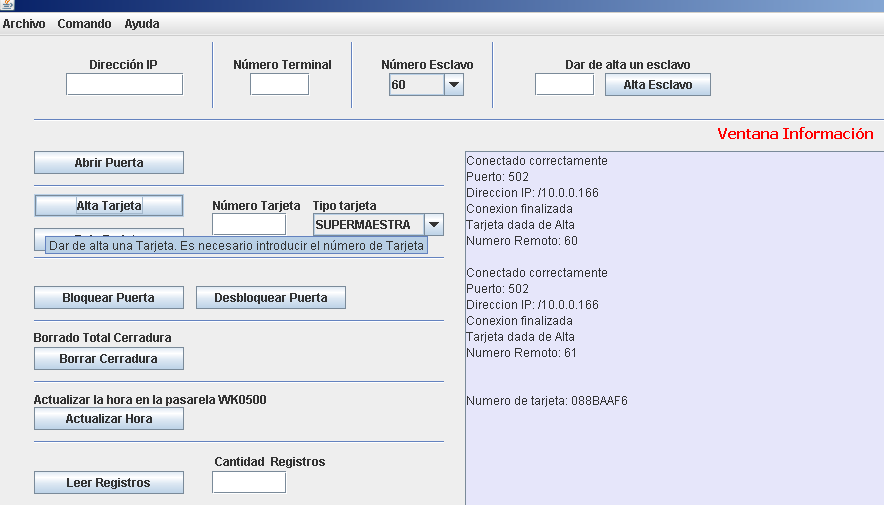


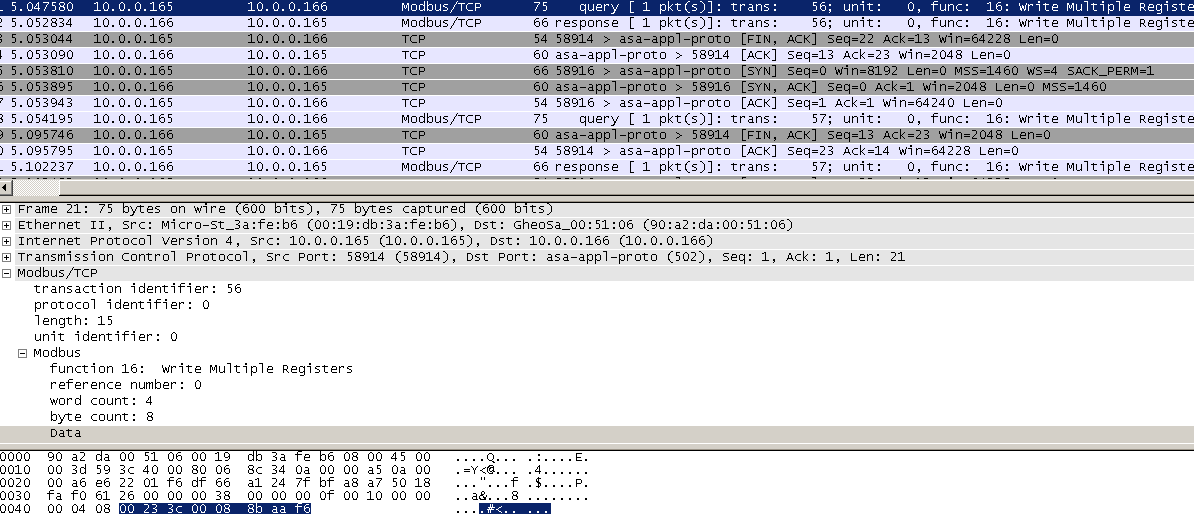
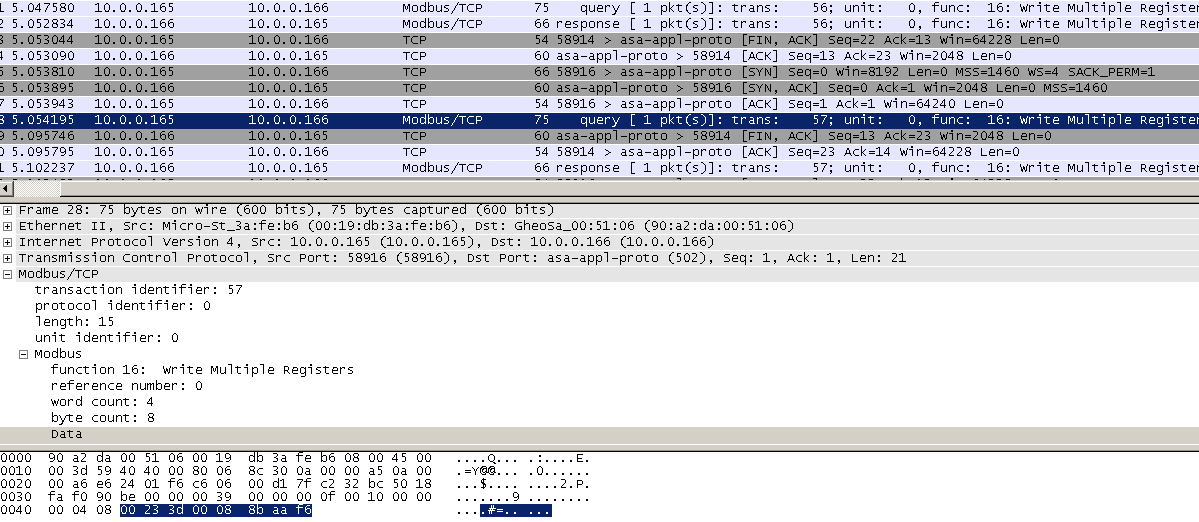


**Acción:** Alta Tarjeta SUPERMAESTRA:

Número de remoto: Todos

Comando: 23

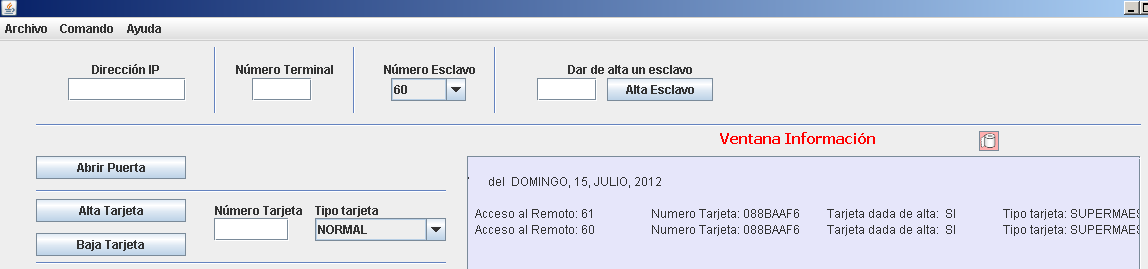


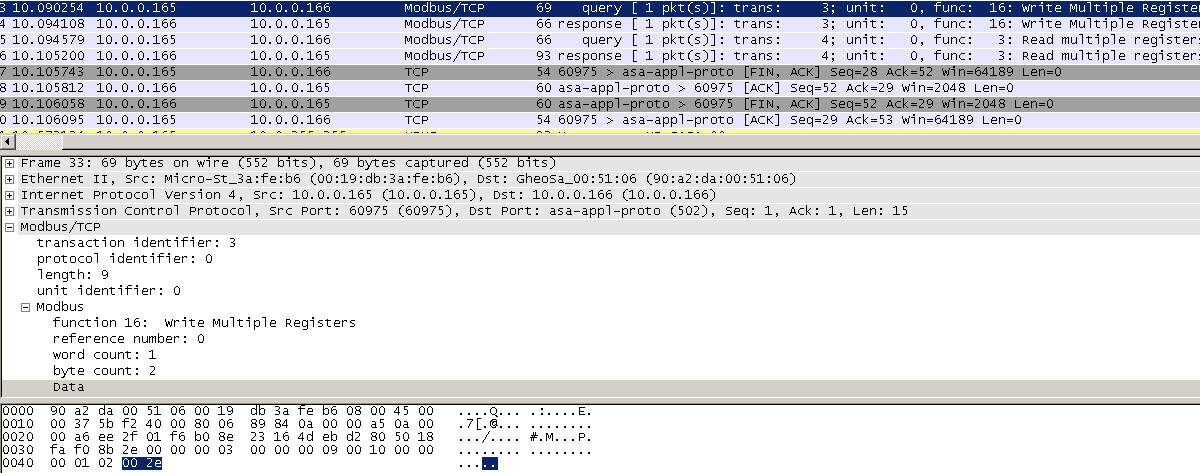


**Acción:** Capturar un evento (Paso de una tarjeta por cualquier remoto):

Número de remoto: Cualquiera

Comando: 2E



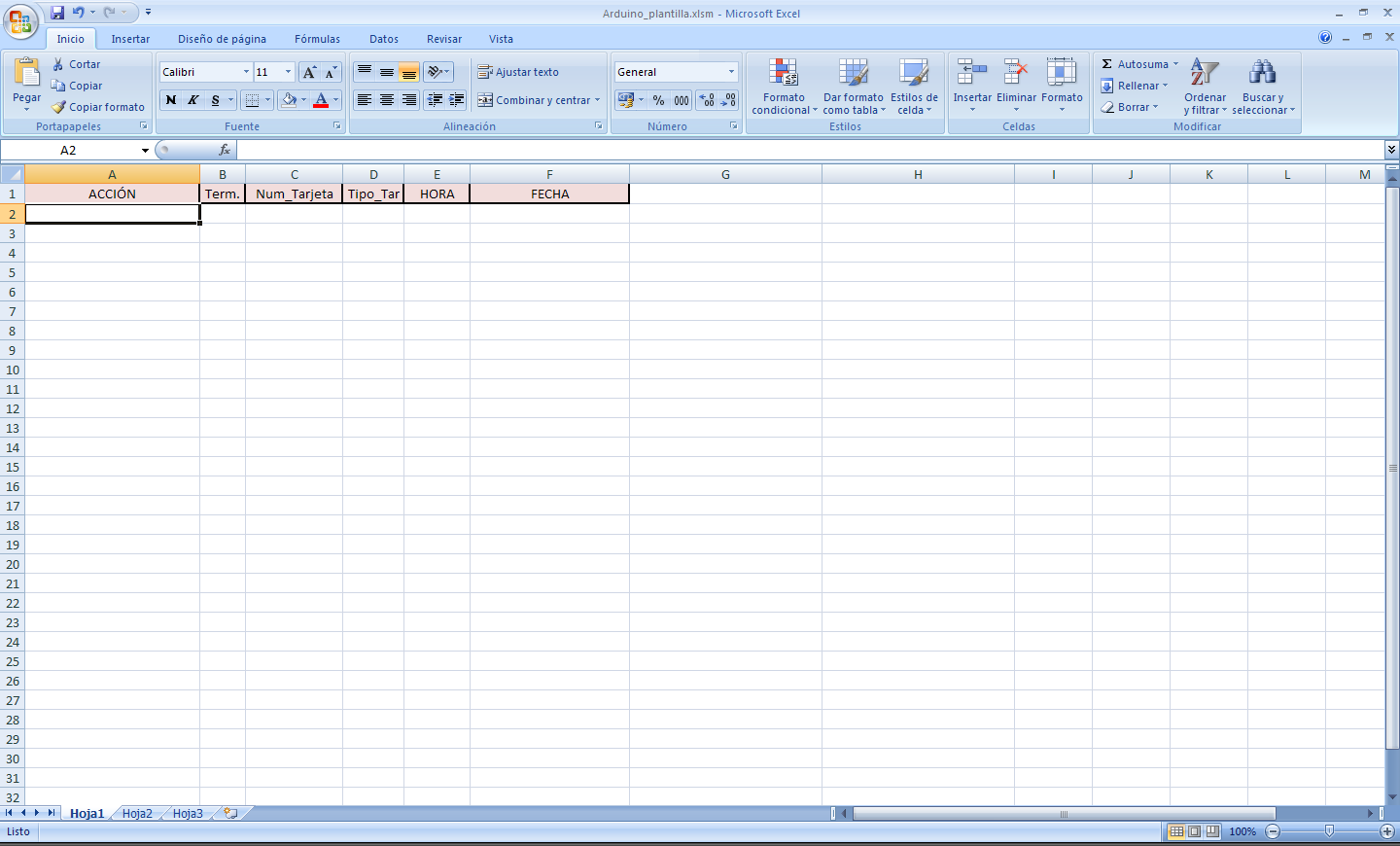


11 .- PRESENTACIÓN DE LOS EVENTOS EN EXCEL.

En las especificaciones del proyecto se pedía que se pudiese hacer un control de las rondas y poder presentarlo en algún formato informático, como Word, Excel o Acces. En la realización del programa **Java Lector de Tarjetas 1.0**, se generan dos ficheros de texto con formato txt. En el primero se utiliza para la presentación en la ventana de información del propio programa. El segundo fichero creado está pensado para poder convertirlo a formato de Excel. Este fichero se llama **copia1.txt**, y los campos se separan por tabulaciones.

Se ha creado una macro para que realice todos los pasos de forma automática y así resulte sencillo a cualquiera poder visualizar o imprimir los eventos ocurridos en un periodo de tiempo desde el Excel.

Lo primero que hemos realizado es un plantilla donde se describen los campos con una anchura fija adecuada.

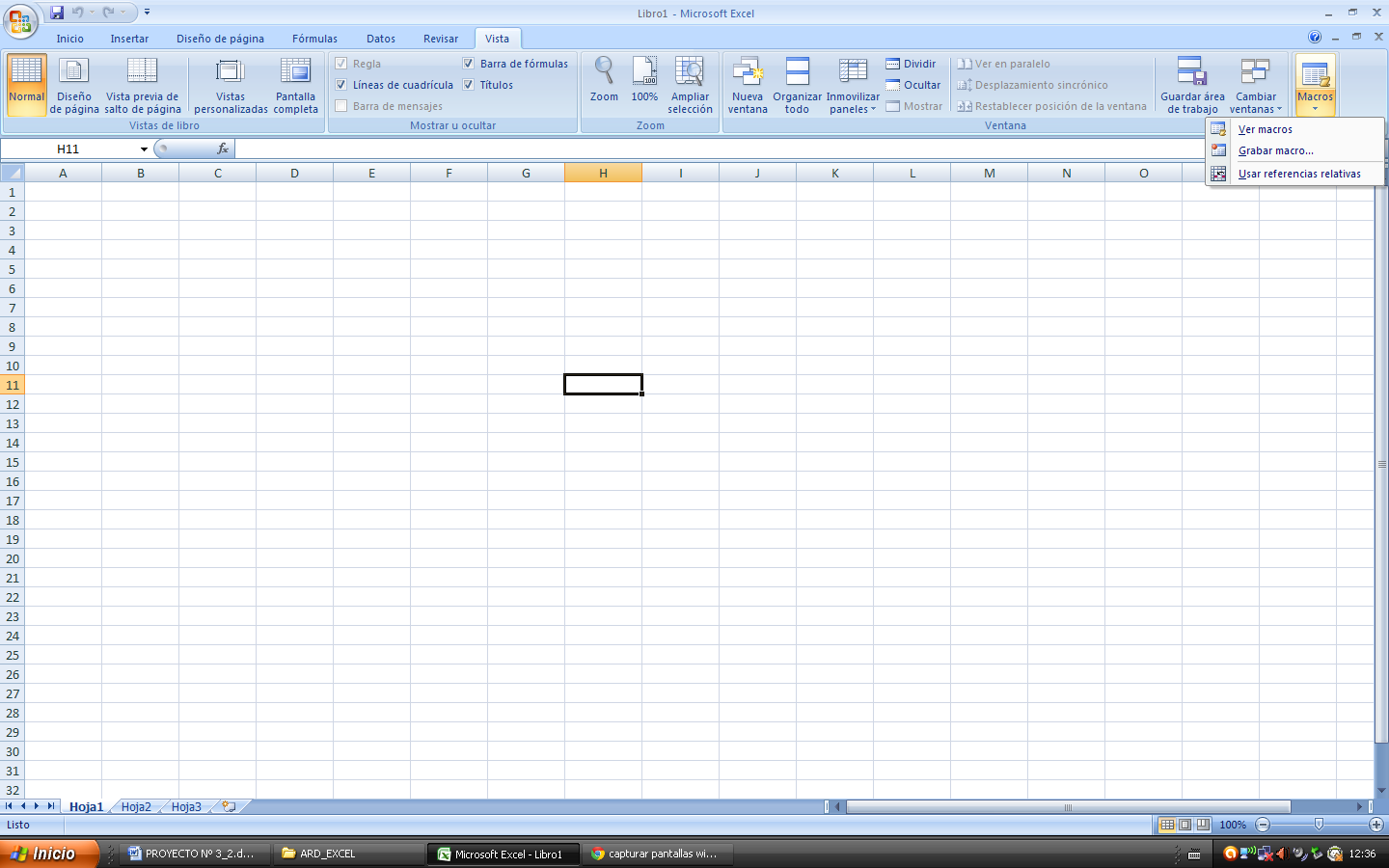


La ubicación del path donde se encuentren los ficheros necesarios ha de ser fija. La macro que crearemos irá a buscar en ese path la plantilla, el fichero copia1.txt. La macro se guardará en el libro1.xlsm que es la que abre por defecto al inicializar el Excel.

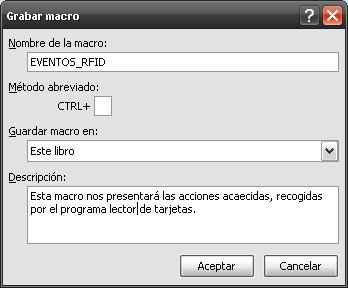
Para crear la macro hemos seguido los siguientes pasos:

1.- Abrir el EXCEL, veremos que nos carga por defecto la plantilla libro1.xlsm.

2.- Iremos a la pestaña **Vista** y dentro de ella a **Macros**. Hacer click en grabar una macro. A partir de ahora, de las acciones que realicemos, se creará un programa en VisualBasic.

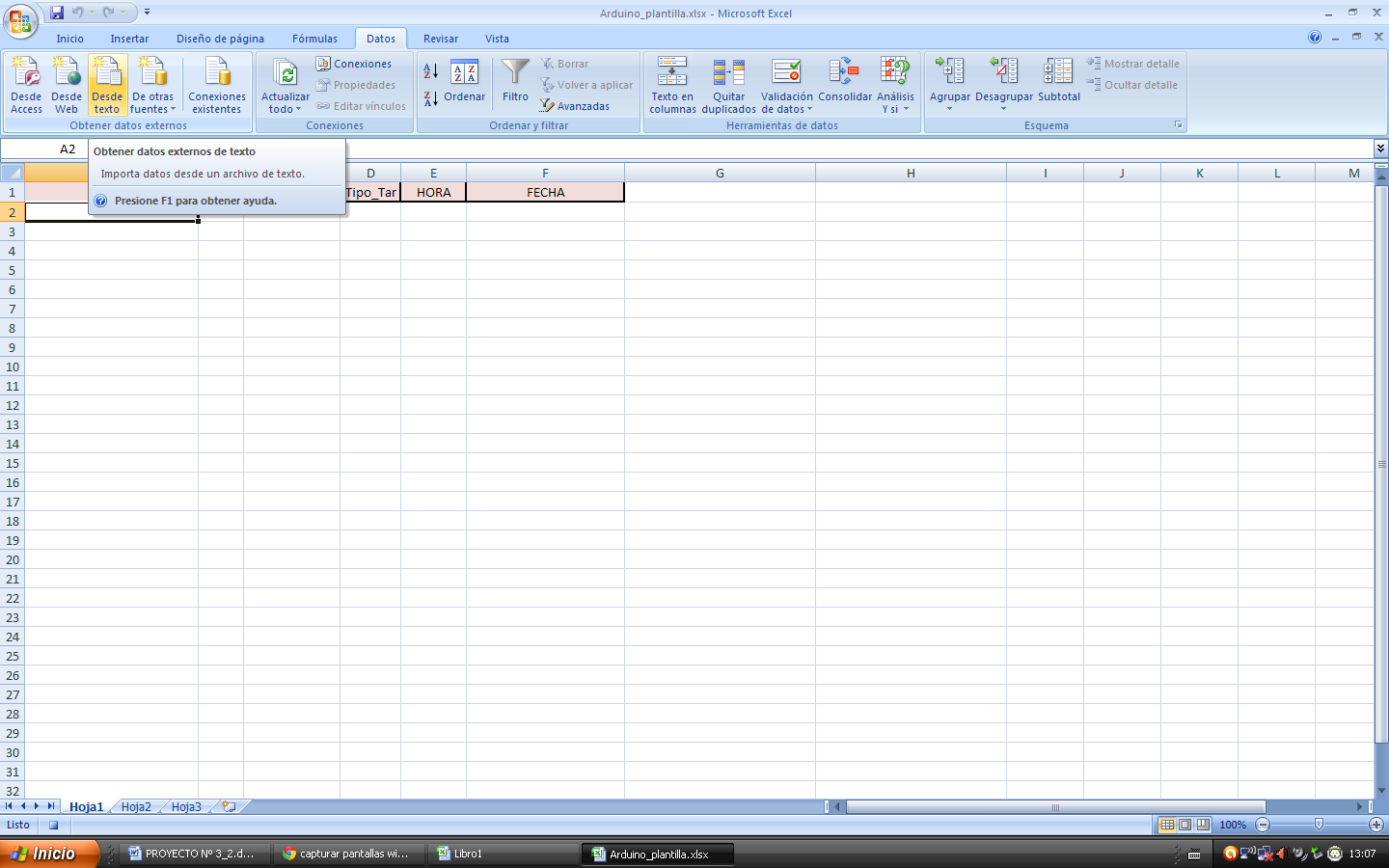


3.- Ahora nos pedirá el nombre con el cual queremos grabarlo y donde queremos se grabe, en nuestro caso en la plantilla que abre al iniciar el EXCEL

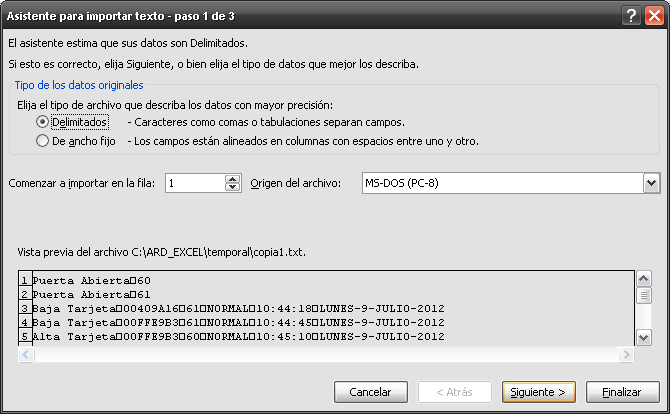


4.-Iremos ahora a **Abrir** y cargaremos el fichero **Arduino\_plantilla.xlsx**, donde hemos creado un simple cabecera en las columnas donde queremos introducir los datos. Estas columnas tiene el tamaño adecuado para imprimirse en formato A4, si fuese necesario. Puede que por seguridad nos muestre un cuadro de dialogo donde nos pida si queremos habilitar las macros, le diremos que sí. Pensemos que una macro en VisualBasic es una forma muy habitual de crear virus informáticos, y es por ello esta precuación.

5.- Ahora iremos a la pestaña **Datos** donde capturaremos los datos del fichero **copia1.txt** en la opción **Desde texto**. Este fichero se machaca cada vez que se abre el programa lector de tarjetas, por lo que sería una buena praxis que, cada vez que se cierra el programa, guardar una copia con los datos ocurridos. Otro criterio podría ser crear un fichero Excel diario.



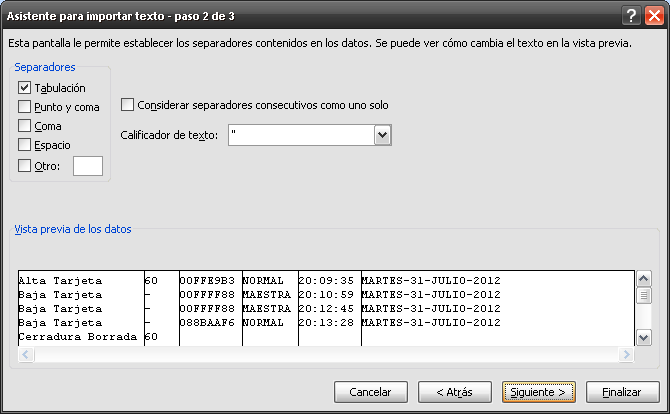
6.- Al cargar los datos nos aparecen varias opciones (que solo aparecerán en la creación de la macro, no en la ejecución)



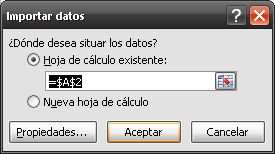
La primera opción dice que los datos están separados por comas o tabulaciones, y esta es la opción que deseamos

También nos pide desde que fila del fichero de texto queremos comenzar la importación, y por defecto esta la primera línea que es lo que deseamos. Clicaremos en **Siguiente**.

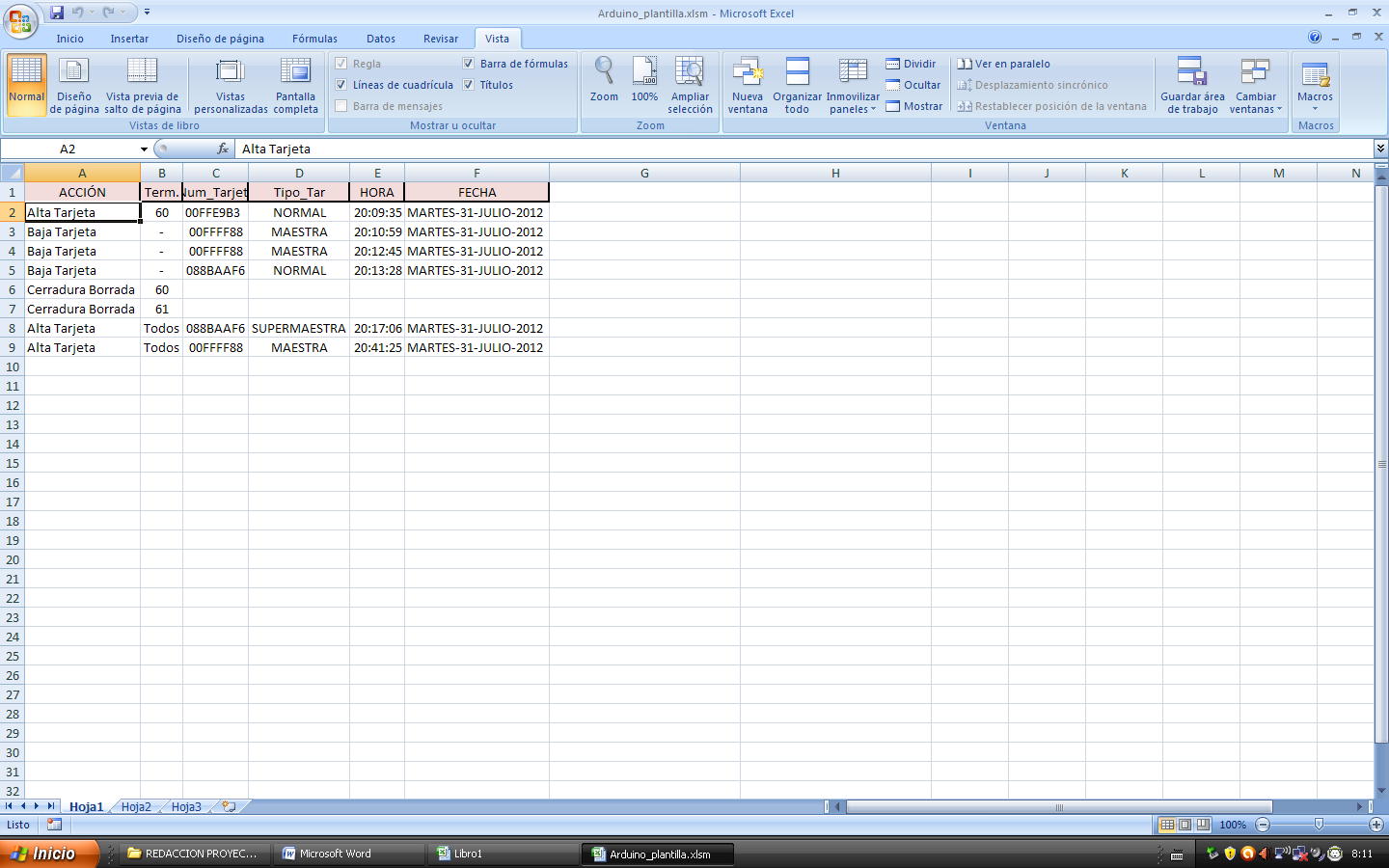
En esta segunda opción nos permite definir cómo van a estar separados nuestros campos. En nuestro caso por tabulaciones. Nos muestra al final una vista preliminar de cómo quedará la presentación de datos. Pulsaremos en finalizar.



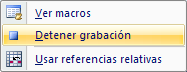
A continuación me pedirá en que celda quiero introducir los datos y le dire a partir de la A2.



A continuación vemos como será el resultado.



7.- Por último iremos a la pestaña **Vista**, la opción **Macros** y pulsaremos en detener macro.



Con todos estos pasos ya nos hemos creado una macro para presentar los datos en Excel. Tendremos que tener presente que los paths para encontrar los ficheros estan predefinidos en la macros, y que otro ordenador la unidad puede cambiar. Esto será detectado en la macro y nos permitirá la opción depurar, donde podremos cambiar dicho path.

Deberíamos ahora permitir los cambios creados en el libro y guardarlo como libro1.xlsm Esta plantilla debemos copiarla en los terminales u ordenadores donde funcionase el programa lector de tarjetas, y así al arrancar el Excel tendríamos a nuestra disposición la macro creada.

ANEXO1.

**CLASE COMANDOS**

//importaciones de todos los paquetes jamod:

import java.net.InetAddress;

import net.wimpi.modbus.io.ModbusTCPTransaction;

import net.wimpi.modbus.msg.ReadMultipleRegistersRequest;

import net.wimpi.modbus.msg.ReadMultipleRegistersResponse;

import net.wimpi.modbus.msg.WriteMultipleRegistersRequest;

import net.wimpi.modbus.msg.WriteMultipleRegistersResponse;

import net.wimpi.modbus.net.TCPMasterConnection;

import net.wimpi.modbus.procimg.SimpleRegister;

/\*\*

\* @author Juan Francisco Grau

\* @author Arturo Aparicio

\* @author Santiago Carbo

\*

\* PROYECTO ARDUINO

\*

\* CONTROL FICHAJES RONDAS

\*

\* Departamento Electricidad/Electronica

\* IES TIRANT LO BLANC

\* GANDIA

\* VALENCIA (ESPANYA)

\*/

public class Comandos {

private int puerto = 502;

private String NL = System.getProperty("line.separator");

private WriteMultipleRegistersRequest solicitudMultRegW = null;

private WriteMultipleRegistersResponse respuestaMultRegW = null;

private ReadMultipleRegistersRequest solicitudMultReg = null; //la solicitud

private ReadMultipleRegistersResponse respuestaMultReg = null; //la respuesta

private TCPMasterConnection con = null;

private ModbusTCPTransaction trans = null;

/\*\*

\* Metodo que realiza las acciones de: Abrir una puerta

\* bloquear una puerta

\* desbloquear una puerta

\* Borrado Total de una cerradura

\* Alta de un esclavo o remoto

\*

\* @param direccion direccion IP de la pasarela (WK0500) que conecta con los remotos

\* @param nTerminal numero de terminal del PC

\* @param comando comandos recibidos: comando = 0x20 Abrir una puerta

\* comando = 0x25 Bloquear una puerta

\* comando = 0x26 Desbloquear una puerta

\* comando = 0x28 Borrado Total de una cerradura

\* comando = 0x29 Dar de alta un esclavo

\* @param nEsclavo numero del remoto

\* @throws Exception

\*/

public String accionesPuerta (String direccion, byte nTerminal, byte comando, byte nEsclavo) throws Exception{

String ret = "";

int ref = 0; //el numero de referencia del registro a leer

int count = 2; //el numero de palabras leidas.

//1. Abrir la conexion

con = new TCPMasterConnection(InetAddress.getByName(direccion));

con.setPort(puerto);

con.connect();

ret = "Conectado correctamente " + NL + "Puerto: " + con.getPort() + NL + "Direccion IP: " + con.getAddress() + NL;

//ESCRIBIR MULTIPLES REGISTROS

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

SimpleRegister[] reg = new SimpleRegister[count];

reg[0] = new SimpleRegister(nTerminal, comando);

reg[1] = new SimpleRegister(nEsclavo, (byte)0x00);

//2. Preparar la solicitud registros

solicitudMultRegW = new WriteMultipleRegistersRequest(ref, reg);

solicitudMultRegW.setRegisters(reg);

//3. Preparar la transccion

trans = new ModbusTCPTransaction(con);

trans.setRequest(solicitudMultRegW);

//4. Ejecutar la transaccion

trans.execute();

setRespuestaMultRegW((WriteMultipleRegistersResponse) trans.getResponse());

//5. Cerrar la conexion

con.close();

ret += "Conexion finalizada";

return ret;

}

/\*

\*

\* Metodo que da de alta una tarjeta

\*

\* Si la tarjeta es MAESTRA o SUPERMAESTRA se envia a todos los esclavos presentes en el sistema.

\*

\* @param direccion direccion IP de la pasarela (WK0500) que conecta con los remotos

\* @param nTerminal numero de terminal del PC

\* @param comando comandos recibidos:

\* tenemos tres tipos de comando en funcion del tipo de tarjeta

\* tipo NORMAL comando = 0x21

\* tipo MAESTRA comando = 0x22

\* tipo SUPERMAESTRA comando = 0x23

\* @param nRemoto numero del remoto.

\* @param dato1 datos tarjeta

\* @param dato2

\* @param dato3

\* @param dato4

\* @return ret String con los resultados

\* @throws Exception

\*/

public String altaTarjeta(String direccion, byte nTerminal, byte comando, StringBuffer nRemoto, int dato1, int dato2, int dato3, int dato4) throws Exception {

String ret = "";

String remoto = ""; //numero del remoto

int ref = 0; //el numero de referencia del registro a leer

int count = 4; //el numero de palabras leidas

int nEsclavoTemp = 0;

byte nEsclavo = 0;

for (int x = 0; x < (nRemoto.length())/2; x++) {

remoto = (String)nRemoto.subSequence(x+x, x+2+x);

nEsclavoTemp = Integer.parseInt(remoto);

nEsclavoTemp = (nEsclavoTemp) & 0xff;

nEsclavo = (byte) nEsclavoTemp;

//1. Abrir la conexion

con = new TCPMasterConnection(InetAddress.getByName(direccion));

con.setPort(puerto);

con.connect();

ret += "Conectado correctamente " + NL + "Puerto: " + con.getPort() + NL + "Direccion IP: " + con.getAddress();

//ESCRIBIR MULTIPLES REGISTROS

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

SimpleRegister[] reg = new SimpleRegister[count];

reg[0] = new SimpleRegister(nTerminal, comando);

reg[1] = new SimpleRegister(nEsclavo, (byte)0x00);

reg[2] = new SimpleRegister((byte)dato1, (byte)dato2);

reg[3] = new SimpleRegister((byte)dato3, (byte)dato4);

//2. Preparar la solicitud registros

solicitudMultRegW = new WriteMultipleRegistersRequest(ref, reg);

solicitudMultRegW.setRegisters(reg);

//3. Preparar la transccion

trans = new ModbusTCPTransaction(con);

trans.setRequest(solicitudMultRegW);

//4. Ejecutar la transaccion

trans.execute();

setRespuestaMultRegW((WriteMultipleRegistersResponse) trans.getResponse());

//5. Cerrar la conexion

con.close();

ret += NL + "Conexion finalizada" + NL + "Tarjeta dada de Alta" + NL + "Numero Remoto: " + remoto + NL + NL;

}

return ret;

}

/\*\*

\* Metodo que da de baja una tarjeta

\*

\* @param direccion direccion IP de la pasarela (WK0500) que conecta con los remotos

\* @param nTerminal numero de terminal del PC

\* @param comando comando recibido: 0x24

\* @param nEsclavo numero del remoto

\* @param dato1 datos tarjeta

\* @param dato2

\* @param dato3

\* @param dato4

\* @return ret String con los resultados

\* @throws Exception

\*/

public String bajaTarjeta(String direccion, byte nTerminal, byte comando, byte nEsclavo, int dato1, int dato2, int dato3, int dato4) throws Exception {

String ret = "";

int ref = 0; //el numero de referencia del registro a leer

int count = 4; //el numero de palabras leidas

//1. Abrir la conexion

con = new TCPMasterConnection(InetAddress.getByName(direccion));

con.setPort(puerto);

con.connect();

ret = "Conectado correctamente " + NL + "Puerto: " + con.getPort() + NL + "Direccion IP: " + con.getAddress() + NL;

//ESCRIBIR MULTIPLES REGISTROS

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

SimpleRegister[] reg = new SimpleRegister[count];

reg[0] = new SimpleRegister(nTerminal, comando);

reg[1] = new SimpleRegister(nEsclavo, (byte)0x00);

reg[2] = new SimpleRegister((byte)dato1, (byte)dato2);

reg[3] = new SimpleRegister((byte)dato3, (byte)dato4);

//2. Preparar la solicitud registros

solicitudMultRegW = new WriteMultipleRegistersRequest(ref, reg);

solicitudMultRegW.setRegisters(reg);

//3. Preparar la transccion

trans = new ModbusTCPTransaction(con);

trans.setRequest(solicitudMultRegW);

//4. Ejecutar la transaccion

trans.execute();

setRespuestaMultRegW((WriteMultipleRegistersResponse) trans.getResponse());

//5. Cerrar la conexion

con.close();

ret += "Conexion finalizada" + NL + "Tarjeta dada de Baja" + NL;

return ret;

}

/\*\*

\* Metodo que realiza la accion de inicializar los remotos

\*

\* @param direccion direccion IP de la pasarela (WK0500) que conecta con los remotos

\* @param nTerminal numero de terminal del PC

\* @param comando comando recibido: comando = 0x2A;

\* @throws Exception

\*/

public StringBuffer inicializarEsclavos (String direccion, byte nTerminal, byte comando) throws Exception{

int ref = 0; //el numero de referencia del registro a leer

int count = 30; //el numero de palabras leidas.

int cantidadReg = 0;

int posicion = 20;

StringBuffer ret = new StringBuffer();

//2. Abrir la conexion

con = new TCPMasterConnection(InetAddress.getByName(direccion));

con.setPort(puerto);

con.connect();

//ESCRIBIR MULTIPLES REGISTROS

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

SimpleRegister[] reg = new SimpleRegister[1];

reg[0] = new SimpleRegister(nTerminal, comando);

//3. Preparar la solicitud registros

solicitudMultRegW = new WriteMultipleRegistersRequest(ref, reg);

solicitudMultRegW.setRegisters(reg);

//4. Preparar la transccion

trans = new ModbusTCPTransaction(con);

trans.setRequest(solicitudMultRegW);

//5. Ejecutar la transaccion

trans.execute();

setRespuestaMultRegW((WriteMultipleRegistersResponse) trans.getResponse());

//LEER MULTIPLES REGISTROS

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//3. Preparar la solicitud registros

solicitudMultReg = new ReadMultipleRegistersRequest(ref, count);

//4. Preparar la transccion

trans.setRequest(solicitudMultReg);

//5. Ejecutar la transaccion

trans.execute();

respuestaMultReg = (ReadMultipleRegistersResponse) trans.getResponse();

cantidadReg = respuestaMultReg.getRegisterValue(posicion);

for (int k = 0; k < cantidadReg; k++) {

ret.append(respuestaMultReg.getRegisterValue(posicion + (k + 1)));

}

//6. Cerrar la conexion

con.close();

return ret;

}

/\*\*

\* Comando que realiza la accion de Actualizar la hora en la pasarela WK0500

\*

\* @param direccion direccion IP de la pasarela (WK0500) que conecta con los remotos

\* @param comando comando recibido: comando = 0x2B

\* @throws Exception

\*/

public String actualizarHora (String direccion, byte nTerminal, byte comando) throws Exception{

String ret = "";

int ref = 0; //el numero de referencia del registro a leer

int count = 6; //el numero de palabras leidas.

//2. Abrir la conexion

con = new TCPMasterConnection(InetAddress.getByName(direccion));

con.setPort(puerto);

con.connect();

ret = "Conectado correctamente " + NL + "Puerto: " + con.getPort() + NL + "Direccion IP: " + con.getAddress() + NL;

//ESCRIBIR MULTIPLES REGISTROS

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

SimpleRegister[] reg = new SimpleRegister[count];

Fecha fecha = new Fecha();

ret += "HORA: " + String.valueOf(fecha.getHora()) + NL + "MINUTOS: " + String.valueOf(fecha.getMinuto()) + NL;

ret += "SEGUNDOS: " + String.valueOf(fecha.getSegundo()) + NL + "DIA DE LA SEMANA: " + String.valueOf(fecha.getDiaSemana()) + " " + fecha.getDiaSemanaTxt() + NL;

ret += "DIA DEL MES: " + String.valueOf(fecha.getDiaMes()) + NL + "MES: " + String.valueOf(fecha.getMes()) + " " + fecha.getMesTxt() + NL;

ret += "ANYO: " + String.valueOf(fecha.getAnyo()) + " " + fecha.getAnyoTxt() + NL;

reg[0] = new SimpleRegister(nTerminal, comando);

reg[1] = new SimpleRegister((byte)0x00, (byte)0x00);

reg[2] = new SimpleRegister(fecha.getHora(), fecha.getMinuto());

reg[3] = new SimpleRegister(fecha.getSegundo(), fecha.getDiaSemana());

reg[4] = new SimpleRegister(fecha.getDiaMes(), fecha.getMes());

reg[5] = new SimpleRegister((byte)0x00, fecha.getAnyo());

//3. Preparar la solicitud registros

solicitudMultRegW = new WriteMultipleRegistersRequest(ref, reg);

solicitudMultRegW.setRegisters(reg);

//4. Preparar la transccion

trans = new ModbusTCPTransaction(con);

trans.setRequest(solicitudMultRegW);

//5. Ejecutar la transaccion

trans.execute();

setRespuestaMultRegW((WriteMultipleRegistersResponse) trans.getResponse());

//6. Cerrar la conexion

con.close();

ret += NL + "Conexion finalizada";

return ret;

}

/\*\*

\* @param direccion direccion IP de la pasarela (WK0500)

\* @param cantidad el numero de palabras a leer.

\* @param comando comando recibido: comando = 0x2E

\* @return ret con la informacion de los registros leidos

\* @throws Exception

\*/

public String leerRegistros (String direccion, byte nTerminal, byte comando, int cantidad, byte nEsclavo) throws Exception{

String ret = "";

int ref = 0; //el numero de referencia del registro a leer

int count = 1; //el numero de palabras leidas

//2. Abrir la conexion

con = new TCPMasterConnection(InetAddress.getByName(direccion));

con.setPort(puerto);

con.connect();

ret = "Conectado correctamente " + NL + "Puerto: " + con.getPort() + NL + "Direccion IP: " + con.getAddress() + NL;

//ESCRIBIR MULTIPLES REGISTROS SIN ENVIAR LA SOLICITUD (para poder asi realizar su lectura)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

SimpleRegister[] reg = new SimpleRegister[count];

reg[0] = new SimpleRegister(nTerminal, comando);

//reg[1] = new SimpleRegister(nEsclavo, (byte)0x00);

//3. Preparar la solicitud registros

solicitudMultRegW = new WriteMultipleRegistersRequest(ref, reg);

solicitudMultRegW.setRegisters(reg);

//4. Preparar la transccion

trans = new ModbusTCPTransaction(con);

trans.setRequest(solicitudMultRegW);

//5. Ejecutar la transaccion

trans.execute();

setRespuestaMultRegW((WriteMultipleRegistersResponse) trans.getResponse());

//LEER MULTIPLES REGISTROS

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//3. Preparar la solicitud registros

solicitudMultReg = new ReadMultipleRegistersRequest(ref, cantidad);

//4. Preparar la transccion

trans.setRequest(solicitudMultReg);

//5. Ejecutar la transaccion

trans.execute();

respuestaMultReg = (ReadMultipleRegistersResponse) trans.getResponse();

for (int k = 0; k < cantidad; k++) {

ret += "Estado del registro " + String.valueOf(k)+ " = " + String.valueOf((respuestaMultReg.getRegisterValue(k))) + NL;

}

//6. Cerrar la conexion

con.close();

ret += NL + "Conexion finalizada";

return ret;

}

/\*\*

\* @param direccion direccion IP de la pasarela (WK0500)

\* @param comando comando recibido: comando = 0x2E

\* @return ret con toda la informacion del evento producido (numero de tarjeta, hora, minutos, segundos, dia de la semana, dia, mes anyo, numero remoto....)

\* @throws Exception

\*/

public String capturarEvento (String direccion, byte nTerminal, byte comando) throws Exception{

String ret = "";

String valor = "";

int valorEntero = 0;

int ref = 0; //el numero de referencia del registro a leer

int count = 1; //el numero de palabras leidas

int cantidad = 15; //el numero de palabras a leer

int posicion = 0;

int dato\_0 = 0;

int dato\_1 = 0;

//2. Abrir la conexion

con = new TCPMasterConnection(InetAddress.getByName(direccion));

con.setPort(puerto);

con.connect();

//ESCRIBIR MULTIPLES REGISTROS SIN ENVIAR LA SOLICITUD (para poder asi realizar su lectura)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

SimpleRegister[] reg = new SimpleRegister[count];

reg[0] = new SimpleRegister(nTerminal, comando);

//reg[1] = new SimpleRegister(nEsclavo, (byte)0x00);

//3. Preparar la solicitud registros

solicitudMultRegW = new WriteMultipleRegistersRequest(ref, reg);

solicitudMultRegW.setRegisters(reg);

//4. Preparar la transccion

trans = new ModbusTCPTransaction(con);

trans.setRequest(solicitudMultRegW);

//5. Ejecutar la transaccion

trans.execute();

setRespuestaMultRegW((WriteMultipleRegistersResponse) trans.getResponse());

//LEER MULTIPLES REGISTROS

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//3. Preparar la solicitud registros

solicitudMultReg = new ReadMultipleRegistersRequest(ref, cantidad);

//4. Preparar la transccion

trans.setRequest(solicitudMultReg);

//5. Ejecutar la transaccion

trans.execute();

respuestaMultReg = (ReadMultipleRegistersResponse) trans.getResponse();

//array para guardar la parte alta(MSB) y la parte baja(LSB) de una palabra

byte[] codeBytes = new byte[2];

//Capturamos el numero de tarjeta.

posicion = 6; //mayor peso

valor = Integer.toHexString(respuestaMultReg.getRegisterValue(posicion));

if (valor.length() < 4) {

if (valor.length() < 3) {

if (valor.length() < 2) {

if (valor.length() < 1) {

ret += "0000";

}else {

ret += "000" + valor.toUpperCase();

}

}else {

ret += "00" + valor.toUpperCase();

}

}else {

ret += "0" + valor.toUpperCase();

}

}else {

ret += valor.toUpperCase();

}

posicion = 7; //menor peso

valor = Integer.toHexString(respuestaMultReg.getRegisterValue(posicion));

if (valor.length() < 4) {

if (valor.length() < 3) {

if (valor.length() < 2) {

if (valor.length() < 1) {

ret += "0000";

}else {

ret += "000" + valor.toUpperCase();

}

}else {

ret += "00" + valor.toUpperCase();

}

}else {

ret += "0" + valor.toUpperCase();

}

}else {

ret += valor.toUpperCase();

}

if (ret.equalsIgnoreCase("00000000")) {

ret = "";

}else {

ret += "$";

//Capturamos la HORA(MSB) y los MINUTOS(LSB).

posicion = 9;

valorEntero = Integer.valueOf(respuestaMultReg.getRegisterValue(posicion));

codeBytes[0] = (byte)(valorEntero & 0xFF); // los MINUTOS

codeBytes[1] = (byte)((valorEntero & 0xFF00)>>8); // la HORA

if (codeBytes[1] < 10) {

ret += "0" + (String.valueOf(codeBytes[1])) + "$"; // la HORA

}else {

ret += (String.valueOf(codeBytes[1])) + "$"; // la HORA

}

if (codeBytes[0] < 10) {

ret += "0" + (String.valueOf(codeBytes[0])) + "$"; // los MINUTOS

}else {

ret += (String.valueOf(codeBytes[0])) + "$"; // los MINUTOS

}

//Capturamos los SEGUNDOS(MSB) y el DIA DE LA SEMANA(LSB) 0 = Domingo,…. 6 = Sabado.

posicion = 10;

valorEntero = Integer.valueOf(respuestaMultReg.getRegisterValue(posicion));

codeBytes[0] = (byte)(valorEntero & 0xFF); // el DIA DE LA SEMANA

codeBytes[1] = (byte)((valorEntero & 0xFF00)>>8); // los SEGUNDOS

if ((codeBytes[1]) < 10) {

ret += "0" + (String.valueOf(codeBytes[1])) + "$"; // los SEGUNDOS

}else {

ret += (String.valueOf(codeBytes[1])) + "$"; // los SEGUNDOS

}

Fecha fecha = new Fecha();

ret += fecha.getDiaSemanaTxt(codeBytes[0]) + "$"; // el DIA DE LA SEMANA

//Capturamos el DIA(MSB) y el MES(LSB).

posicion = 11;

valorEntero = Integer.valueOf(respuestaMultReg.getRegisterValue(posicion));

codeBytes[0] = (byte)(valorEntero & 0xFF); // el MES

codeBytes[1] = (byte)((valorEntero & 0xFF00)>>8); // el DIA

ret += (String.valueOf(codeBytes[1])) + "$"; // el DIA

ret += (String.valueOf(codeBytes[0])) + "$"; // el MES

//Capturamos el ANYO.

posicion = 12;

ret += (String.valueOf(respuestaMultReg.getRegisterValue(posicion))) + "$"; // el ANYO

//Capturamos el numero de remoto.

posicion = 13;

ret += (String.valueOf(respuestaMultReg.getRegisterValue(posicion))) + "$"; // el numero de REMOTO

//Capturamos si la tarjeta esta dada de alta (o no), el tipo de tarjeta, si la puerta esta bloqueada (o no) y si la tarjeta habilitada (o no).

posicion = 8;

//Tarjeta dada de alta

valorEntero = Integer.valueOf(respuestaMultReg.getRegisterValue(posicion));

codeBytes[0] = (byte)(valorEntero & 0xFF); // LSB

codeBytes[1] = (byte)((valorEntero & 0x0100)>>8); // MSB

//Quitamos el signo

dato\_0 = codeBytes[0];

dato\_0 = dato\_0 & 0xff;

if (dato\_0 == 255) {

ret += " NO" + "$";

//Tipo tarjeta

ret += " --\t" + "$";

//Habilitacion tarjeta

ret += " --" + "$";

}else {

ret += " SI" + "$";

//Tipo tarjeta

valorEntero = Integer.valueOf(respuestaMultReg.getRegisterValue(posicion));

codeBytes[0] = (byte)((valorEntero & 0x30)>>4);

//Quitamos el signo

dato\_0 = codeBytes[0];

dato\_0 = dato\_0 & 0xff;

if (dato\_0 == 0) {

ret += "NORMAL" + "$";

}else if (dato\_0 == 1) {

ret += "MAESTRA" + "$";

}else {

ret += "SUPERMAESTRA" + "$";

}

//Habilitacion tarjeta

valorEntero = Integer.valueOf(respuestaMultReg.getRegisterValue(posicion));

codeBytes[0] = (byte)(valorEntero & 0x01); // LSB

//Quitamos el signo

dato\_0 = codeBytes[0];

dato\_0 = dato\_0 & 0xff;

if (dato\_0 == 0) {

ret += " NO" + "$";

}else {

ret += " SI" + "$";

}

}

//Quitamos el signo

dato\_1 = codeBytes[1];

dato\_1 = dato\_1 & 0xff;

//Bloqueo puerta

if (dato\_1 == 0) {

ret += " NO" + "$";

}else {

ret += " SI" + "$";

}

}

//6. Cerrar la conexion

con.close();

return ret;

}

/\*\*

\* Este metodo no es necesario.

\* @return respuestaMultRegW

\*/

public WriteMultipleRegistersResponse getRespuestaMultRegW() {

return respuestaMultRegW;

}

/\*\*

\* Este metodo no es necesario.

\* @param respuestaMultRegW

\*/

public void setRespuestaMultRegW(WriteMultipleRegistersResponse respuestaMultRegW) {

this.respuestaMultRegW = respuestaMultRegW;

}

}

**CLASE EVENTO**

/\*\*

\* @author Juan Francisco Grau

\* @author Arturo Aparicio

\* @author Santiago Carbo

\*

\* PROYECTO ARDUINO

\*

\* CONTROL FICHAJES RONDAS

\*

\* Departamento Electricidad/Electronica

\* IES TIRANT LO BLANC

\* GANDIA

\* VALENCIA (ESPANYA)

\*/

public class Evento {

private String nRemoto;

private String nTarjeta;

private String hora;

private String minuto;

private String segundo;

private String diaSemana;

private String dia;

private String mes;

private String anyo;

private String altaTarjeta;

private String tipoTarjeta;

private String habilitacionTarjeta;

private String puertaBloqueada;

public String getAltaTarjeta() {

return altaTarjeta;

}

public void setAltaTarjeta(String altaTarjeta) {

this.altaTarjeta = altaTarjeta;

}

public String getTipoTarjeta() {

return tipoTarjeta;

}

public void setTipoTarjeta(String tipoTarjeta) {

this.tipoTarjeta = tipoTarjeta;

}

public String getHabilitacionTarjeta() {

return habilitacionTarjeta;

}

public void setHabilitacionTarjeta(String habilitacionTarjeta) {

this.habilitacionTarjeta = habilitacionTarjeta;

}

public String getPuertaBloqueada() {

return puertaBloqueada;

}

public void setPuertaBloqueada(String puertaBloqueada) {

this.puertaBloqueada = puertaBloqueada;

}

public String getnRemoto() {

return nRemoto;

}

public void setnRemoto(String nRemoto) {

this.nRemoto = nRemoto;

}

public String getnTarjeta() {

return nTarjeta;

}

public void setnTarjeta(String nTarjeta) {

this.nTarjeta = nTarjeta;

}

public String getHora() {

return hora;

}

public void setHora(String hora) {

this.hora = hora;

}

public String getMinuto() {

return minuto;

}

public void setMinuto(String minuto) {

this.minuto = minuto;

}

public String getSegundo() {

return segundo;

}

public void setSegundo(String segundo) {

this.segundo = segundo;

}

public String getDiaSemana() {

return diaSemana;

}

public void setDiaSemana(String diaSemana) {

this.diaSemana = diaSemana;

}

public String getDia() {

return dia;

}

public void setDia(String dia) {

this.dia = dia;

}

public String getMes() {

return mes;

}

public void setMes(String mes) {

this.mes = mes;

}

public String getAnyo() {

return anyo;

}

public void setAnyo(String anyo) {

this.anyo = anyo;

}

}

**CLASE FECHA**

import java.util.GregorianCalendar;

/\*\*

\* @author Juan Francisco Grau

\* @author Arturo Aparicio

\* @author Santiago Carbo

\*

\* PROYECTO ARDUINO

\*

\* CONTROL FICHAJES RONDAS

\*

\* Departamento Electricidad/Electronica

\* IES TIRANT LO BLANC

\* GANDIA

\* VALENCIA (ESPANYA)

\*/

public class Fecha {

private byte hora;

private byte minuto;

private byte segundo;

private byte diaSemana;

private byte diaMes;

private byte mes;

private byte anyo;

private String horaTxt;

private String minutoTxt;

private String segundoTxt;

private String diaSemanaTxt;

private String diaMesTxt;

private String mesTxt;

private String anyoTxt;

GregorianCalendar calendario = new GregorianCalendar();

/\*\*

\* @return byte de la hora actual

\*/

public byte getHora() {

hora = (byte)calendario.get(GregorianCalendar.HOUR\_OF\_DAY);

return hora;

}

/\*\*

\* @return byte de los minutos transcurridos en el momento actual

\*/

public byte getMinuto() {

minuto = (byte)calendario.get(GregorianCalendar.MINUTE);

return minuto;

}

/\*\*

\* @return byte de los segundos transcurridos en el momento actual

\*/

public byte getSegundo() {

segundo = (byte)calendario.get(GregorianCalendar.SECOND);

return segundo;

}

/\*\*

\* @return byte con el dia de la semana

\*

\* GregorianCalendar nos devuelve: 1 - DOMINGO Nosotros necesitamos un 0,

\* 2 - LUNES Nosotros necesitamos un 1,

\* 3 - MARTES Nosotros necesitamos un 2,

\* 4 - MIERCOLES Nosotros necesitamos un 3,

\* 5 - JUEVES Nosotros necesitamos un 4,

\* 6 - VIERNES Nosotros necesitamos un 5,

\* 7 - SABADO Nosotros necesitamos un 6,

\*/

public byte getDiaSemana() {

int numDia = calendario.get(GregorianCalendar.DAY\_OF\_WEEK);

switch (numDia) {

case 1:

numDia = numDia - 1;

diaSemana = (byte)numDia;

break;

case 2:

numDia = numDia - 1;

diaSemana = (byte)numDia;

break;

case 3:

numDia = numDia - 1;

diaSemana = (byte)numDia;

break;

case 4:

numDia = numDia - 1;

diaSemana = (byte)numDia;

break;

case 5:

numDia = numDia - 1;

diaSemana = (byte)numDia;

break;

case 6:

numDia = numDia - 1;

diaSemana = (byte)numDia;

break;

case 7:

numDia = numDia - 1;

diaSemana = (byte)numDia;

break;

}

return diaSemana;

}

/\*\*

\* @return byte con el dia del mes (1 a 31)

\*/

public byte getDiaMes() {

diaMes = (byte)calendario.get(GregorianCalendar.DAY\_OF\_MONTH);

return diaMes;

}

/\*\*

\* @return byte con el mes actual (0 "Enero",.... 11 "Diciembre")

\*/

public byte getMes() {

int m = calendario.get(GregorianCalendar.MONTH) + 1;

mes = (byte)m;

return mes;

}

/\*\*

\* @return byte con los dos digitos de menor peso del anyo actual

\*/

public byte getAnyo() {

int year = calendario.get(GregorianCalendar.YEAR);

year = year - 2000;

anyo = (byte)year;

return anyo;

}

/\*\*

\* @return String con la hora actual

\*/

public String getHoraTxt() {

if (calendario.get(GregorianCalendar.HOUR\_OF\_DAY) < 10) {

horaTxt = "0" + String.valueOf(calendario.get(GregorianCalendar.HOUR\_OF\_DAY));

}else {

horaTxt = String.valueOf(calendario.get(GregorianCalendar.HOUR\_OF\_DAY));

}

return horaTxt;

}

/\*\*

\* @return String de los minutos transcurridos en el momento actual

\*/

public String getMinutoTxt() {

if (calendario.get(GregorianCalendar.MINUTE) < 10) {

minutoTxt = "0" + String.valueOf(calendario.get(GregorianCalendar.MINUTE));

}else {

minutoTxt = String.valueOf(calendario.get(GregorianCalendar.MINUTE));

}

return minutoTxt;

}

/\*\*

\* @return String de los segundos transcurridos en el momento actual

\*/

public String getSegundoTxt() {

if (calendario.get(GregorianCalendar.SECOND) < 10) {

segundoTxt = "0" + String.valueOf(calendario.get(GregorianCalendar.SECOND));

}else {

segundoTxt = String.valueOf(calendario.get(GregorianCalendar.SECOND));

}

return segundoTxt;

}

/\*\*

\* @return String con el dia de la semana (LUNES,....,DOMINGO)

\*/

public String getDiaSemanaTxt(int numDia) {

switch (numDia) {

case 0:

diaSemanaTxt = "DOMINGO";

break;

case 1:

diaSemanaTxt = "LUNES";

break;

case 2:

diaSemanaTxt = "MARTES";

break;

case 3:

diaSemanaTxt = "MIERCOLES";

break;

case 4:

diaSemanaTxt = "JUEVES";

break;

case 5:

diaSemanaTxt = "VIERNES";

break;

case 6:

diaSemanaTxt = "SABADO";

break;

}

return diaSemanaTxt;

}

/\*\*

\* @return String con el dia de la semana (LUNES,....,DOMINGO)

\*/

public String getDiaSemanaTxt() {

int numDia = calendario.get(GregorianCalendar.DAY\_OF\_WEEK);

switch (numDia) {

case 1:

diaSemanaTxt = "DOMINGO";

break;

case 2:

diaSemanaTxt = "LUNES";

break;

case 3:

diaSemanaTxt = "MARTES";

break;

case 4:

diaSemanaTxt = "MIERCOLES";

break;

case 5:

diaSemanaTxt = "JUEVES";

break;

case 6:

diaSemanaTxt = "VIERNES";

break;

case 7:

diaSemanaTxt = "SABADO";

break;

}

return diaSemanaTxt;

}

/\*\*

\* @return String con el dia del mes actual

\*/

public String getDiaMesTxt() {

diaMesTxt = String.valueOf(calendario.get(GregorianCalendar.DAY\_OF\_MONTH));

return diaMesTxt;

}

/\*\*

\* @return String con el mes actual (ENERO,....,DICIEMBRE)

\*/

public String getMesTxt() {

int m = calendario.get(GregorianCalendar.MONTH) + 1;

switch (m) {

case 1:

mesTxt = "ENERO";

break;

case 2:

mesTxt = "FEBRERO";

break;

case 3:

mesTxt = "MARZO";

break;

case 4:

mesTxt = "ABRIL";

break;

case 5:

mesTxt = "MAYO";

break;

case 6:

mesTxt = "JUNIO";

break;

case 7:

mesTxt = "JULIO";

break;

case 8:

mesTxt = "AGOSTO";

break;

case 9:

mesTxt = "SEPTIEMBRE";

break;

case 10:

mesTxt = "OCTUBRE";

break;

case 11:

mesTxt = "NOVIEMBRE";

break;

case 12:

mesTxt = "DICIEMBRE";

break;

}

return mesTxt;

}

/\*\*

\* @return String con el anyo actual

\*/

public String getAnyoTxt() {

anyoTxt = String.valueOf(calendario.get(GregorianCalendar.YEAR));

return anyoTxt;

}

}

**CLASE FICHERO**

import java.io.BufferedReader;

import java.io.File;

import java.io.FileWriter;

import java.io.IOException;

import java.nio.charset.Charset;

import java.nio.file.Path;

import java.nio.file.Paths;

import java.nio.file.Files;

/\*\*

\* @author Juan Francisco Grau

\* @author Arturo Aparicio

\* @author Santiago Carbo

\*

\* PROYECTO ARDUINO

\*

\*

\* Departamento Electricidad/Electronica

\* IES TIRANT LO BLANC

\* GANDIA

\* VALENCIA (ESPANYA)

\*/

public class Fichero {

private String NL = System.getProperty("line.separator");

private Path path;

private boolean existe;

public void crearFichero (String rutaFichero) throws IOException {

path = Paths.get(rutaFichero);

setExiste(false);

if (!(setExiste(Files.exists(path)))) {

Files.createFile(path);

}

}

public void insertarDatos (String texto, String rutaFichero) throws IOException {

path = Paths.get(rutaFichero);

File TextFile = new File(rutaFichero);

FileWriter TextOut = new FileWriter(TextFile, true);

TextOut.append(texto);

TextOut.close();

}

public String leerDatos(String rutaNombreFichero) throws IOException {

String ret = "";

String line = null;

path = Paths.get(rutaNombreFichero);

Charset charset = Charset.forName("UTF-8");

BufferedReader datos = Files.newBufferedReader(path, charset);

Files.readAllLines(path, charset);

while ((line = datos.readLine()) != null) {

ret += line + NL;

}

return ret;

}

public boolean isExiste() {

return existe;

}

public boolean setExiste(boolean existe) {

this.existe = existe;

return existe;

}

}

**CLASE JTEXTFIELDLIMIT**

import javax.swing.JOptionPane;

import javax.swing.text.\*;

/\*\*

\* @author Juan Francisco Grau

\* @author Arturo Aparicio

\* @author Santiago Carbo

\*

\* PROYECTO ARDUINO

\*

\* CONTROL FICHAJES RONDAS

\*

\* Departamento Electricidad/Electronica

\* IES TIRANT LO BLANC

\* GANDIA

\* VALENCIA (ESPANYA)

\*/

public class JTextFieldLimit extends PlainDocument {

/\*\*

\*

\*/

private static final long serialVersionUID = 1L;

private int limit;

// optional uppercase conversion

private boolean toUppercase = false;

JTextFieldLimit(int limit) {

super();

this.limit = limit;

}

JTextFieldLimit(int limit, boolean upper) {

super();

this.limit = limit;

toUppercase = upper;

}

@Override

public void insertString (int offset, String str, AttributeSet attr) {

if (str == null) return;

if ((getLength() + str.length()) <= limit) {

if (toUppercase) str = str.toUpperCase();

try {

super.insertString(offset, str, attr);

} catch (BadLocationException e) {

JOptionPane.showMessageDialog(null, "inserte 8 caracteres en hexadecimal", "ERROR", JOptionPane.ERROR\_MESSAGE);

}

}

}

}

**CLASE LECTORTARJETAS (main)**

import javax.swing.JFormattedTextField;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JMenuItem;

import javax.swing.JOptionPane;

import javax.swing.JTextField;

import javax.swing.JButton;

import javax.swing.JTextArea;

import java.awt.event.ActionListener;

import java.awt.event.ActionEvent;

import javax.swing.JMenuBar;

import javax.swing.JMenu;

import java.text.ParseException;

import javax.swing.JSeparator;

import java.awt.Color;

import java.awt.FileDialog;

import javax.swing.ImageIcon;

import javax.swing.JScrollPane;

import javax.swing.text.MaskFormatter;

import javax.swing.JComboBox;

import java.awt.Font;

import java.io.File;

import java.io.IOException;

import javax.swing.SwingConstants;

import java.util.ArrayList;

import java.util.List;

import java.util.Iterator;

import javax.swing.Timer;

import javax.swing.UIManager;

/\*\*

\* @author Juan Francisco Grau

\* @author Arturo Aparicio

\* @author Santiago Carbo

\*

\* PROYECTO ARDUINO

\*

\* CONTROL FICHAJES RONDAS

\*

\* Departamento Electricidad/Electronica

\* IES TIRANT LO BLANC

\* GANDIA

\* VALENCIA (ESPANYA)

\*/

public class LectorTarjetas extends JFrame{

private String NL = System.getProperty("line.separator");

private static final long serialVersionUID = 1L;

private static LectorTarjetas ventana;

private JTextArea taHistorial;

private JTextArea taInformacion;

private JTextField tfNumeroTarjeta;

private JTextField tfdirIP;

private JTextField tfnEsclavo;

private JTextField tfNumTerminal;

private JTextField tfCantidadReg;

private JComboBox<String> cBoxTipoTarjeta;

private JComboBox<String> cBoxEsclavos;

private String temp = "";

private String rutaFichero;

private int delay = 1000;

/\*\*

\* Lista de eventos que se han producido

\*/

private List<Evento> evento;

//Constructor

/\*\*

\* Interfaz grafico

\*

\* @throws ParseException

\* @throws IOException

\*/

public LectorTarjetas() {

super();

setTitle("Proyecto ARDUINO Control Fichajes Rondas");

String direccion = "10.0.0.166";

String rutaFichero1 = "C:\\temporal\\copia1.txt";

String retornoInicial = "";

StringBuffer ret = new StringBuffer();

String remoto = " ";

byte numTerminal = 0; //mi terminal

byte comando = 0x2A; //2A Inicializar Remotos (en hexadecimal)

getContentPane().setLayout(null);

/\*\*

\* Creamos un directorio y un fichero para guardar la informacion de los diferentes eventos.

\*/

File directorio = new File("C:\\temporal"); //Creacion directorio

directorio.mkdir();

final Fichero fichero = new Fichero(); //Creacion fichero

Fecha fecha = new Fecha();

this.rutaFichero = "C:\\temporal\\copia" + fecha.getDiaMesTxt() + "-" + fecha.getMesTxt() + "-" + fecha.getAnyoTxt() + "\_" + fecha.getHoraTxt() + "-" + fecha.getMinutoTxt() + "-" + fecha.getSegundoTxt() + ".txt";

try {

fichero.crearFichero(rutaFichero);

} catch (IOException e2) {

JOptionPane.showMessageDialog(null, "La creaci\u00F3n del fichero en C:\\temporal\\copia.txt ha fallado", "ERROR", JOptionPane.ERROR\_MESSAGE);

}

try {

fichero.crearFichero(rutaFichero1);

} catch (IOException e2) {

JOptionPane.showMessageDialog(null, "La creaci\u00F3n del fichero en C:\\temporal\\copia1.txt ha fallado", "ERROR", JOptionPane.ERROR\_MESSAGE);

}

/\*\*

\* Texto para informar en que zona se debe introducir la direccion IP

\*/

JLabel jLabDirIP = new JLabel("Direcci\u00F3n IP ");

jLabDirIP.setHorizontalAlignment(SwingConstants.CENTER);

jLabDirIP.setToolTipText("(por ejemplo: 10.0.0.166)");

jLabDirIP.setBounds(69, 21, 118, 14);

getContentPane().add(jLabDirIP);

/\*\*

\* Campo para introducir la direccion IP

\*/

tfdirIP = new JTextField();

tfdirIP.setDocument(new JTextFieldLimit(15));

tfdirIP.setColumns(15);

tfdirIP.setBounds(69, 37, 118, 23);

getContentPane().add(tfdirIP);

// Si no introducimos la direccion IP nos quedamos con la direccion por defecto 10.0.0.166

temp = tfdirIP.getText(); //Capturamos lo que hay en la ventana direccion IP.

tfdirIP.setText(""); //Borramos la ventana direccion IP.

if (!temp.equalsIgnoreCase("")) { //Si hemos introducido la direccion IP en la ventana direccion IP, la cogemos.

direccion = temp;

}

/\*\*

\* Campo desplegable para seleccionar el numero de esclavo (remoto).

\* Este desplegable se inicializa al arrancar el programa

\* Incluimos un Proceso para inicializar el desplegable con los numeros de esclavo dados de alta.

\*/

cBoxEsclavos = new JComboBox<String>();

Comandos EnviarComando = new Comandos();

try {

ret = EnviarComando.inicializarEsclavos(direccion, numTerminal, comando);

for (int k = 0; k < (ret.length())/2; k++) {

remoto = (String)ret.subSequence(k+k, k+2+k);

cBoxEsclavos.addItem(remoto);

}

} catch (Exception e) {

JOptionPane.showMessageDialog(null, "La inicializaci\u00F3n de los remotos ha fallado", "ERROR", JOptionPane.ERROR\_MESSAGE);

}

retornoInicial = NL + "Programa iniciado a las " + fecha.getHoraTxt() + ":" + fecha.getMinutoTxt() + ":" + fecha.getSegundoTxt() + " del " + fecha.getDiaSemanaTxt() + ", " + fecha.getDiaMesTxt() + ", " + fecha.getMesTxt() + ", " + fecha.getAnyoTxt() + NL + "\r\n";

try {

fichero.insertarDatos(retornoInicial, rutaFichero);

} catch (IOException e2) {

JOptionPane.showMessageDialog(null, "ERROR al escribir en el fichero", "ERROR", JOptionPane.ERROR\_MESSAGE);

}

cBoxEsclavos.setBounds(392, 37, 75, 23);

getContentPane().add(cBoxEsclavos);

/\*\*

\* Temporizador para ir rastreando los posibles eventos que se pueden producir en los remotos.

\* De esta forma los podremos ir guardando en el fichero "copia.txt" los nuevos eventos que se produzcan.

\* Para cada nuevo evento se crea un objeto en el que guardaremos los datos siguientes:

\* numero de remoto

\* numero de tarjeta

\* Hora actual

\* Fecha actual

\*/

evento = new ArrayList<Evento>();

Timer timer = new Timer(delay, new ActionListener() {

public void actionPerformed(ActionEvent e) {

String retEvento = "";

String direccion = "10.0.0.166";

String retFichero = "";

boolean found = false;

Evento a = null;

byte numTerminal = 0; //mi terminal

byte comando = 0x2E; //2E leer registros (en hexadecimal)

int x = 0;

Comandos EnviarComando = new Comandos();

try {

retEvento = EnviarComando.capturarEvento(direccion, numTerminal, comando);

} catch (Exception e1) {

JOptionPane.showMessageDialog(null, "No se ha podido establecer conexi\u00F3n con la pasarela WK0500", "ERROR", JOptionPane.ERROR\_MESSAGE);

}

if (!(retEvento.equalsIgnoreCase(""))) {

Evento nuevoEvento = new Evento();

String[] eventos = retEvento.split("\\$");

/\*\*

\* Si todavia no se ha producido ningun evento lo guardamos en un array de eventos y en el fichero "copia.txt"

\* En caso contrario: miramos si el nuevo evento ya lo tenemos guardado en el array de eventos.

\* Si no lo tenemos lo guardamos y enviamos la informacion al fichero.

\*/

for (String eventoActual:eventos) {

switch (x) {

case 0:

nuevoEvento.setnTarjeta(eventoActual); //guardamos el numero de la tarjeta

x++;

break;

case 1:

nuevoEvento.setHora(eventoActual); //guardamos la hora

x++;

break;

case 2:

nuevoEvento.setMinuto(eventoActual); //guardamos los minutos

x++;

break;

case 3:

nuevoEvento.setSegundo(eventoActual); //guardamos los segundos

x++;

break;

case 4:

nuevoEvento.setDiaSemana(eventoActual); //guardamos el dia de la semana

x++;

break;

case 5:

nuevoEvento.setDia(eventoActual); //guardamos el dia

x++;

break;

case 6:

nuevoEvento.setMes(eventoActual); //guardamos el mes

x++;

break;

case 7:

nuevoEvento.setAnyo(eventoActual); //guardamos el anyo

x++;

break;

case 8:

nuevoEvento.setnRemoto(eventoActual); //guardamos el numero del remoto

x++;

break;

case 9:

nuevoEvento.setAltaTarjeta(eventoActual); //tarjeta dada de alta (o no)

x++;

break;

case 10:

nuevoEvento.setTipoTarjeta(eventoActual); //tipo tarjeta

x++;

break;

case 11:

nuevoEvento.setHabilitacionTarjeta(eventoActual); //Habilitacion tarjeta

x++;

break;

case 12:

nuevoEvento.setPuertaBloqueada(eventoActual); //Bloqueo puerta

x++;

break;

}

}

Fichero fichero = new Fichero();

if (evento.size() == 0) {

evento.add(nuevoEvento);

retFichero = "\t" + "\t" +"Acceso al Remoto: " + nuevoEvento.getnRemoto() + "\t" + "Numero Tarjeta: " + nuevoEvento.getnTarjeta() + "\t" + "Tarjeta dada de alta: " + nuevoEvento.getAltaTarjeta() + "\t" + "Tipo tarjeta: " + nuevoEvento.getTipoTarjeta() + "\t" + "Tarjeta habilitada: " + nuevoEvento.getHabilitacionTarjeta() + "\t" + "Puerta bloqueada: " + nuevoEvento.getPuertaBloqueada() + "\t" + "HORA: " + nuevoEvento.getHora() + ":" + nuevoEvento.getMinuto() + ":" + nuevoEvento.getSegundo() + "\t" + "FECHA: " + nuevoEvento.getDiaSemana() + ", " + nuevoEvento.getDia() + ", " + nuevoEvento.getMes() + ", " + nuevoEvento.getAnyo() + "\r\n";

try {

fichero.insertarDatos(retFichero, rutaFichero);

} catch (IOException e1) {

JOptionPane.showMessageDialog(null, "ERROR al escribir en el fichero", "ERROR", JOptionPane.ERROR\_MESSAGE);

}

}else {

Iterator<Evento> it = evento.iterator();

while (it.hasNext() && !found){

a = (Evento) it.next();

if ((a.getDiaSemana().equalsIgnoreCase(nuevoEvento.getDiaSemana())) && (a.getnRemoto().equalsIgnoreCase(nuevoEvento.getnRemoto())) && (a.getnTarjeta().equalsIgnoreCase(nuevoEvento.getnTarjeta())) && (a.getHora().equalsIgnoreCase(nuevoEvento.getHora())) && (a.getMinuto().equalsIgnoreCase(nuevoEvento.getMinuto())) && (a.getSegundo().equalsIgnoreCase(nuevoEvento.getSegundo())) && (a.getDia().equalsIgnoreCase(nuevoEvento.getDia())) && (a.getMes().equalsIgnoreCase(nuevoEvento.getMes())) && (a.getAnyo().equalsIgnoreCase(nuevoEvento.getAnyo())))

found = true;

}

if (!found){

evento.add(nuevoEvento);

retFichero = "\t" + "\t" +"Acceso al Remoto: " + nuevoEvento.getnRemoto() + "\t" + "Numero Tarjeta: " + nuevoEvento.getnTarjeta() + "\t" + "Tarjeta dada de alta: " + nuevoEvento.getAltaTarjeta() + "\t" + "Tipo tarjeta: " + nuevoEvento.getTipoTarjeta() + "\t" + "Tarjeta habilitada: " + nuevoEvento.getHabilitacionTarjeta() + "\t" + "Puerta bloqueada: " + nuevoEvento.getPuertaBloqueada() + "\t" + "HORA: " + nuevoEvento.getHora() + ":" + nuevoEvento.getMinuto() + ":" + nuevoEvento.getSegundo() + "\t" + "FECHA: " + nuevoEvento.getDiaSemana() + ", " + nuevoEvento.getDia() + ", " + nuevoEvento.getMes() + ", " + nuevoEvento.getAnyo() + "\r\n";

try {

fichero.insertarDatos(retFichero, rutaFichero);

} catch (IOException e1) {

JOptionPane.showMessageDialog(null, "ERROR al escribir en el fichero", "ERROR", JOptionPane.ERROR\_MESSAGE);

}

}

}

}

}

});

timer.start();

/\*\*

\* Texto para informar en que zona se debe introducir el numero de Terminal

\*/

JLabel jLabNumTerminal = new JLabel("N\u00FAmero Terminal");

jLabNumTerminal.setToolTipText("El n\u00FAmero asignado a nuestro PC (por defecto el 0)");

jLabNumTerminal.setHorizontalAlignment(SwingConstants.CENTER);

jLabNumTerminal.setBounds(233, 21, 106, 14);

getContentPane().add(jLabNumTerminal);

/\*\*

\* Campo para introducir el numero de Terminal

\*/

tfNumTerminal = new JTextField();

tfNumTerminal.setColumns(10);

tfNumTerminal.setBounds(253, 37, 60, 23);

getContentPane().add(tfNumTerminal);

/\*\*

\* Texto para informar en que zona se debe introducir el numero de esclavos

\*/

JLabel jLabEsclavosAlta = new JLabel("N\u00FAmero Esclavo");

jLabEsclavosAlta.setToolTipText("Desplegable para escoger el remoto");

jLabEsclavosAlta.setHorizontalAlignment(SwingConstants.CENTER);

jLabEsclavosAlta.setBounds(372, 21, 119, 14);

getContentPane().add(jLabEsclavosAlta);

/\*\*

\* Boton Abrir Puerta

\*/

JButton btAbrirPuerta = new JButton("Abrir Puerta");

btAbrirPuerta.setToolTipText("Hay que escoger el n\u00FAmero de esclavo");

btAbrirPuerta.addActionListener(new BtAbrirPuertaActionListener());

btAbrirPuerta.setBounds(37, 115, 150, 23);

getContentPane().add(btAbrirPuerta);

/\*\*

\* Texto para informar en que zona esta la Ventana de informacion

\*/

JLabel label = new JLabel("Ventana Informaci\u00F3n");

label.setForeground(Color.RED);

label.setHorizontalAlignment(SwingConstants.CENTER);

label.setFont(new Font("Tahoma", Font.BOLD, 15));

label.setBounds(712, 90, 174, 14);

getContentPane().add(label);

/\*\*

\* Ventana de informacion con desplazamientos Horizontal y Vertical

\*/

JScrollPane scrollPane = new JScrollPane();

scrollPane.setBounds(468, 115, 674, 356);

getContentPane().add(scrollPane);

taHistorial = new JTextArea();

scrollPane.setViewportView(taHistorial);

taHistorial.setBackground(new Color(230, 230, 250));

/\*\*

\* Boton Alta Tarjeta

\*/

JButton btAltaTarjeta = new JButton("Alta Tarjeta");

btAltaTarjeta.setToolTipText("Dar de alta una Tarjeta. Es necesario introducir el n\u00FAmero de Tarjeta");

btAltaTarjeta.addActionListener(new BtAltaTarjetaActionListener());

btAltaTarjeta.setBounds(37, 158, 150, 23);

getContentPane().add(btAltaTarjeta);

/\*\*

\* Boton Baja Tarjeta

\*/

JButton btBajaTarjeta = new JButton("Baja Tarjeta");

btBajaTarjeta.addActionListener(new BtBajaTarjetaActionListener());

btBajaTarjeta.setToolTipText("Dar de baja una Tarjeta.Solo hay que introducir el n\u00FAmero de Tarjeta");

btBajaTarjeta.setBounds(37, 192, 150, 23);

getContentPane().add(btBajaTarjeta);

/\*\*

\* Texto para informar en que zona esta el cuadro de texto para introducir el numero de tarjeta

\*/

JLabel jLabnumTarjeta = new JLabel("N\u00FAmero Tarjeta");

jLabnumTarjeta.setToolTipText(" (8 d\u00EDgitos m\u00E1ximo en Hexadecimal)");

jLabnumTarjeta.setBounds(216, 162, 93, 14);

getContentPane().add(jLabnumTarjeta);

/\*\*

\* Campo para introducir el codigo de la tarjeta en hexadecimal

\*/

MaskFormatter mascaraHex = null;

try {

mascaraHex = new MaskFormatter("HHHHHHHH");

} catch (ParseException e1) {

JOptionPane.showMessageDialog(null, "Introduzca un n\u00FAmero en Hexadecimal", "ERROR", JOptionPane.ERROR\_MESSAGE);

}

tfNumeroTarjeta = new JFormattedTextField(mascaraHex);

tfNumeroTarjeta.setDocument(new JTextFieldLimit(8));

tfNumeroTarjeta.setBounds(215, 177, 75, 23);

getContentPane().add(tfNumeroTarjeta);

/\*\*

\* Boton para borrar la ventana de informacion

\*/

JButton btBorrarVentanaInfo = new JButton("Borrar");

btBorrarVentanaInfo.setToolTipText("Borrar Ventana Informaci\u00F3n");

btBorrarVentanaInfo.addActionListener(new BtBorrarVentanaInfoActionListener());

btBorrarVentanaInfo.setBackground(UIManager.getColor("Button.background"));

btBorrarVentanaInfo.setBounds(907, 90, 83, 20);

getContentPane().add(btBorrarVentanaInfo);

/\*\*

\* Texto para informar en que zona esta el campo desplegable para seleccionar el tipo de tarjeta

\*/

JLabel lblTipoTarjeta = new JLabel("Tipo tarjeta");

lblTipoTarjeta.setToolTipText("NORMAL, MAESTRA, SUPERMAESTRA");

lblTipoTarjeta.setBounds(316, 162, 67, 14);

getContentPane().add(lblTipoTarjeta);

/\*\*

\* Campo desplegable para seleccionar el tipo de tarjeta.

\* Tipos disponibles:

\* NORMAL

\* MAESTRA

\* SUPERMAESTRA

\*/

cBoxTipoTarjeta = new JComboBox<String>();

cBoxTipoTarjeta.addItem("NORMAL");

cBoxTipoTarjeta.addItem("MAESTRA");

cBoxTipoTarjeta.addItem("SUPERMAESTRA");

cBoxTipoTarjeta.setSelectedIndex(0); //Seleccionamos una opcion por defecto

getContentPane().add(cBoxTipoTarjeta);

cBoxTipoTarjeta.setBounds(316, 177, 131, 23);

getContentPane().add(cBoxTipoTarjeta);

/\*\*

\* Boton para bloquear una puerta

\*/

JButton btBloquearPuerta = new JButton("Bloquear Puerta");

btBloquearPuerta.addActionListener(new BtBloquearPuertaActionListener());

btBloquearPuerta.setToolTipText("Bloquear una puerta. Hay que escoger el n\u00FAmero del Esclavo");

btBloquearPuerta.setBounds(37, 250, 150, 23);

getContentPane().add(btBloquearPuerta);

/\*\*

\* Boton para desbloquear una puerta

\*/

JButton btDesbloquearPuerta = new JButton("Desbloquear Puerta");

btDesbloquearPuerta.addActionListener(new BtDesbloquearPuertaActionListener());

btDesbloquearPuerta.setToolTipText("Desbloquear una puerta. Hay que escoger el n\u00FAmero del Esclavo");

btDesbloquearPuerta.setBounds(199, 250, 150, 23);

getContentPane().add(btDesbloquearPuerta);

/\*\*

\* Texto informativo Borrado total cerradura

\*/

JLabel jLabBorrarCerradura = new JLabel("Borrado Total Cerradura");

jLabBorrarCerradura.setToolTipText("Borrado cerradura (indicar n\u00FAmero esclavo)");

jLabBorrarCerradura.setBounds(37, 294, 205, 14);

getContentPane().add(jLabBorrarCerradura);

/\*\*

\* Boton para realizar el borrado total de una cerradura

\*/

JButton btBorrarCerradura = new JButton("Borrar Cerradura");

btBorrarCerradura.addActionListener(new BtBorrarCerraduraActionListener());

btBorrarCerradura.setToolTipText("Borrado Total de una cerradura. Hay que escoger el n\u00FAmero del Esclavo");

btBorrarCerradura.setBounds(37, 311, 150, 23);

getContentPane().add(btBorrarCerradura);

/\*\*

\* Texto informativo Actualizar la hora en la pasarela WK0500

\*/

JLabel jLabActualizarHora = new JLabel("Actualizar la hora en la pasarela WK0500");

jLabActualizarHora.setToolTipText("Borrado cerradura (indicar n\u00FAmero esclavo)");

jLabActualizarHora.setBounds(37, 356, 253, 14);

getContentPane().add(jLabActualizarHora);

/\*\*

\* Boton para realizar la actualizacion de la hora en el WK0500

\*/

JButton btActualizarHora = new JButton("Actualizar Hora");

btActualizarHora.addActionListener(new BtActualizarHoraActionListener());

btActualizarHora.setToolTipText("Actualiza la hora del WK0500");

btActualizarHora.setBounds(37, 371, 150, 23);

getContentPane().add(btActualizarHora);

/\*\*

\* Texto informativo para indicar lo que tendremos que introducir en el campo cantidad de registros.

\*/

JLabel jLabCantRegistros = new JLabel("Cantidad Registros");

jLabCantRegistros.setToolTipText("m\u00E1ximo 100 registros");

jLabCantRegistros.setBounds(218, 418, 131, 14);

getContentPane().add(jLabCantRegistros);

/\*\*

\* Campo para introducir la cantidad de registros a leer.

\*/

tfCantidadReg = new JTextField();

tfCantidadReg.setColumns(8);

tfCantidadReg.setBounds(215, 435, 75, 23);

getContentPane().add(tfCantidadReg);

/\*\*

\* Boton para realizar la lectura de una cierta cantidad de HOLDING REGISTERS

\*/

JButton btLeerRegistros = new JButton("Leer Registros");

btLeerRegistros.addActionListener(new BtLeerRegistrosActionListener());

btLeerRegistros.setToolTipText("Hay que introducir la cantidad de registros a leer");

btLeerRegistros.setBounds(37, 435, 150, 23);

getContentPane().add(btLeerRegistros);

/\*\*

\* Texto informativo para indicar lo que tendremos que introducir en el campo Dar de alta un esclavo.

\*/

JLabel jLabNumEsclavo = new JLabel("Dar de alta un esclavo");

jLabNumEsclavo.setToolTipText("Dar de alta un esclavo en el sistema");

jLabNumEsclavo.setHorizontalAlignment(SwingConstants.CENTER);

jLabNumEsclavo.setBounds(544, 21, 174, 14);

getContentPane().add(jLabNumEsclavo);

/\*\*

\* Campo para introducir el numero del esclavo y asi ser actualizado en el sistema.

\*/

tfnEsclavo = new JTextField();

tfnEsclavo.setBounds(538, 37, 60, 23);

getContentPane().add(tfnEsclavo);

tfnEsclavo.setColumns(10);

/\*\*

\* Boton para dar de alta un remoto en el sistema.

\*/

JButton btAltaEsclavo = new JButton("Alta Esclavo");

btAltaEsclavo.addActionListener(new BtAltaEsclavoActionListener());

btAltaEsclavo.setToolTipText("Dar de alta un esclavo en el sistema");

btAltaEsclavo.setBounds(608, 37, 106, 23);

getContentPane().add(btAltaEsclavo);

/\*\*

\* Lineas separadoras horizontales y verticales

\*/

JSeparator separator = new JSeparator();

separator.setBounds(37, 83, 1105, 2);

getContentPane().add(separator);

JSeparator separator\_1 = new JSeparator();

separator\_1.setBounds(37, 149, 410, 2);

getContentPane().add(separator\_1);

JSeparator separator\_2 = new JSeparator();

separator\_2.setBounds(37, 221, 410, 2);

getContentPane().add(separator\_2);

JSeparator separator\_3 = new JSeparator();

separator\_3.setBounds(37, 284, 410, 2);

getContentPane().add(separator\_3);

JSeparator separator\_4 = new JSeparator();

separator\_4.setBounds(37, 345, 410, 2);

getContentPane().add(separator\_4);

JSeparator separator\_5 = new JSeparator();

separator\_5.setBounds(37, 405, 410, 2);

getContentPane().add(separator\_5);

JSeparator separator\_6 = new JSeparator();

separator\_6.setBounds(37, 469, 410, 2);

getContentPane().add(separator\_6);

JSeparator separator\_7 = new JSeparator();

separator\_7.setOrientation(SwingConstants.VERTICAL);

separator\_7.setBounds(495, 6, 8, 66);

getContentPane().add(separator\_7);

JSeparator separator\_8 = new JSeparator();

separator\_8.setOrientation(SwingConstants.VERTICAL);

separator\_8.setBounds(354, 6, 8, 66);

getContentPane().add(separator\_8);

JSeparator separator\_9 = new JSeparator();

separator\_9.setOrientation(SwingConstants.VERTICAL);

separator\_9.setBounds(215, 6, 8, 66);

getContentPane().add(separator\_9);

//Hacemos la ventana visible y definimos el tamanyo

setSize(1185,659);

/\*\*

\* BARRA DE MENUS

\*/

JMenuBar menuBar = new JMenuBar();

setJMenuBar(menuBar);

/\*\*

\* Menu Archivo de la barra de menus

\*/

JMenu mnArchivo = new JMenu("Archivo");

menuBar.add(mnArchivo);

//OPCIONES MENU ARCHIVO

JMenuItem menuGuardarHistorial = new JMenuItem("Guardar Historial");

menuGuardarHistorial.addActionListener(new MenuGuardarHistorialActionListener());

JMenuItem menuAbrirHistorial = new JMenuItem( "Abrir Historial");

menuAbrirHistorial.addActionListener(new MenuAbrirHistorialActionListener());

mnArchivo.add(menuAbrirHistorial);

mnArchivo.add(menuGuardarHistorial);

mnArchivo.addSeparator();

JMenuItem menuSalir = new JMenuItem( "Salir");

menuSalir.addActionListener(new MenuSalirActionListener());

mnArchivo.add(menuSalir);

/\*\*

\* Menu Comando de la barra de menus

\*/

JMenu mnComando = new JMenu("Comando");

menuBar.add(mnComando);

//OPCIONES MENU COMANDO

JMenuItem menuAbrirPuerta = new JMenuItem("Abrir Puerta");

menuAbrirPuerta.addActionListener(new MenuAbrirPuertaActionListener());

mnComando.add(menuAbrirPuerta);

JMenuItem menuAltaTarjeta = new JMenuItem( "Alta Tarjeta");

menuAltaTarjeta.addActionListener(new MenuAltaTarjetaActionListener());

mnComando.add(menuAltaTarjeta);

JMenuItem menuBajaTarjeta = new JMenuItem("Baja Tarjeta");

menuBajaTarjeta.addActionListener(new MenuBajaTarjetaActionListener());

mnComando.add( menuBajaTarjeta );

JMenuItem menuBloquearPuerta = new JMenuItem("Bloquear Puerta");

menuBloquearPuerta.addActionListener(new MenuBloquearPuertaActionListener());

mnComando.add(menuBloquearPuerta);

JMenuItem menuDesbloquearPuerta = new JMenuItem("Desbloquear Puerta");

menuDesbloquearPuerta.addActionListener(new MenuDesbloquearPuertaActionListener());

mnComando.add(menuDesbloquearPuerta);

JMenuItem menuBorrarCerradura = new JMenuItem("Borrar Cerradura");

menuBorrarCerradura.addActionListener(new MenuBorrarCerraduraActionListener());

mnComando.add(menuBorrarCerradura);

JMenuItem menuActualizarHora = new JMenuItem("Actualizar Hora");

menuActualizarHora.addActionListener(new MenuActualizarHoraActionListener());

mnComando.add(menuActualizarHora);

JMenuItem menuLeerRegistros = new JMenuItem("Leer Registros");

menuLeerRegistros.addActionListener(new MenuLeerRegistrosActionListener());

mnComando.add(menuLeerRegistros);

/\*\*

\* Menu Ayuda de la barra de menus

\*/

JMenu mnAyuda = new JMenu("Ayuda");

menuBar.add(mnAyuda);

//OPCIONES MENU AYUDA

JMenuItem mntmAcercaDe = new JMenuItem("Acerca de...");

mntmAcercaDe.addActionListener(new MntmAcercaDeActionListener());

mnAyuda.add(mntmAcercaDe);

setVisible(true);

}

/\*\*

\* Clase que realiza la accion de Abrir una puerta

\*

\* COMANDO 0x20

\*/

private class BtAbrirPuertaActionListener implements ActionListener {

private String NL = System.getProperty("line.separator");

public void actionPerformed(ActionEvent arg0) {

String direccion = "10.0.0.166";

String rutaFichero1 = "C:\\temporal\\copia1.txt";

String retFichero = "";

String retFichero1 = "";

String ret = "";

byte numEsclavo = 0; //numero del remoto

byte numTerminal = 0; //mi terminal

byte comando = 0x20; //20 orden Abrir puerta (en hexadecimal)

int miTerminal = 0;

int nEsclavo = 0;

if (cBoxEsclavos.getItemCount() != 0) {

// Capturamos el numero de esclavo

temp = cBoxEsclavos.getSelectedItem().toString();

nEsclavo = Integer.parseInt(temp);

nEsclavo = (nEsclavo) & 0xff;

numEsclavo = (byte) nEsclavo;

// Si no introducimos la direccion IP nos quedamos con la direccion por defecto 10.0.0.166

temp = tfdirIP.getText(); //Capturamos lo que hay en la ventana direccion IP. tfdirIP.setText(""); //Borramos la ventana direccion IP.

if (!temp.equalsIgnoreCase("")) { //Si hemos introducido la direccion IP en la ventana direccion IP, la cogemos.

direccion = temp;

}

// Si no introducimos el numero de terminal nos quedamos con el introducido por defecto

temp = tfNumTerminal.getText(); //Capturamos lo que hay en la ventana Numero Terminal

tfNumTerminal.setText(""); //Borramos la ventana

if (!temp.equalsIgnoreCase("")) { //Si hemos introducido en numero de terminal lo cogemos

miTerminal = Integer.parseInt(temp);

miTerminal = (miTerminal) & 0xff;

numTerminal = (byte)miTerminal;

}

Fichero fichero = new Fichero();

Fecha fecha = new Fecha();

retFichero = "Puerta Abierta" + "\t" + "\t" + "Numero de remoto: " + cBoxEsclavos.getSelectedItem().toString() + "\t" + "\t" + "\t" + "\t" + "\t" + "\t" + "\t" + "\t" + "\t" + "\t" + "\t" + "HORA: " + fecha.getHoraTxt() + ":" + fecha.getMinutoTxt() + ":" + fecha.getSegundoTxt() + "\t" + "FECHA: " + fecha.getDiaSemanaTxt() + ", " + fecha.getDiaMesTxt() + ", " + fecha.getMesTxt() + ", " + fecha.getAnyoTxt() + "\r\n";

retFichero1 = "Puerta Abierta" + "\t" + cBoxEsclavos.getSelectedItem().toString() + "\r\n";

Comandos EnviarComando = new Comandos();

try {

ret = EnviarComando.accionesPuerta(direccion, numTerminal, comando, numEsclavo);

} catch (Exception e) {

JOptionPane.showMessageDialog(null, "No se ha podido establecer conexi\u00F3n con la pasarela WK0500", "ERROR", JOptionPane.ERROR\_MESSAGE);

}

try {

fichero.insertarDatos(retFichero, rutaFichero);

fichero.insertarDatos(retFichero1, rutaFichero1);

} catch (IOException e) {

JOptionPane.showMessageDialog(null, "ERROR al escribir en el fichero", "ERROR", JOptionPane.ERROR\_MESSAGE);

}

ret += NL + NL + "Puerta Abierta " + NL + "N\u00FAmero de remoto: " + cBoxEsclavos.getSelectedItem().toString() + NL;

}else {

ret = "No hay remotos dados de alta en el sistema";

}

taHistorial.setText(ret);

}

}

/\*\*

\* Clase que realiza la accion de dar de alta una tarjeta

\* Se introduce en un cuadro de texto el codigo de la tarjeta (8 digitos en Hexadecimal)

\* En un desplegable se escoge el tipo de tarjeta (NORMAL, MAESTRA, SUPERMAESTRA)

\*

\* COMANDOS

\* Tipo NORMAL comando 0x21

\* Tipo MAESTRA comando 0x22

\* Tipo SUPERMAESTRA comando 0x23

\*/

private class BtAltaTarjetaActionListener implements ActionListener {

private String NL = System.getProperty("line.separator");

public void actionPerformed(ActionEvent arg0) {

String direccion = "10.0.0.166";

String rutaFichero1 = "C:\\temporal\\copia1.txt";

String retFichero = "";

String retFichero1 = "";

String ret = "";

String numTarjeta = "";

String dato = "";

String str = "";

StringBuffer remoto = new StringBuffer();

byte comando = 0x21;

byte numTerminal = 0;//mi terminal

int miTerminal = 0;

int datoN1 = 0;

int datoN2 = 0;

int datoN3 = 0;

int datoN4 = 0;

if (cBoxEsclavos.getItemCount() != 0) {

// Si no introducimos la direccion IP nos quedamos con la direccion por defecto 10.0.0.166

temp = tfdirIP.getText(); //Capturamos lo que hay en la ventana direccion IP.

tfdirIP.setText(""); //Borramos la ventana direccion IP.

if (!temp.equalsIgnoreCase("")) { //Si hemos introducido la direccion IP en la ventana direccion IP, la cogemos.

direccion = temp;

}

// Si no introducimos el numero de terminal nos quedamos con el introducido por defecto

temp = tfNumTerminal.getText(); //Capturamos lo que hay en la ventana Numero Terminal

tfNumTerminal.setText(""); //Borramos la ventana

if (!temp.equalsIgnoreCase("")) { //Si hemos introducido en numero de terminal lo cogemos

miTerminal = Integer.parseInt(temp);

miTerminal = (miTerminal) & 0xff;

numTerminal = (byte)miTerminal;

}

numTarjeta =tfNumeroTarjeta.getText();

for (int x = 0; x < 4; x++) {

switch (x) {

case 0:

dato = numTarjeta.substring(x,x+2);

datoN1 = Integer.parseInt(dato, 16);

datoN1 = (datoN1) & 0xff;

break;

case 1:

dato = numTarjeta.substring(x+1,x+3);

datoN2 = Integer.parseInt(dato, 16);

datoN2 = (datoN2) & 0xff;

break;

case 2:

dato = numTarjeta.substring(x+2,x+4);

datoN3 = Integer.parseInt(dato, 16);

datoN3 = (datoN3) & 0xff;

break;

case 3:

dato = numTarjeta.substring(x+3,x+5);

datoN4 = Integer.parseInt(dato, 16);

datoN4 = (datoN4) & 0xff;

break;

}

}

Fichero fichero = new Fichero();

Fecha fecha = new Fecha();

str = cBoxTipoTarjeta.getSelectedItem().toString();

if (str.equalsIgnoreCase("NORMAL")){

comando = 0x21;

retFichero = "Alta Tarjeta" + "\t" + "\t" + "Numero de remoto: " + cBoxEsclavos.getSelectedItem().toString() + "\t" + "Numero Tarjeta: " + tfNumeroTarjeta.getText() + "\t" + "\t" + "\t" + "Tipo tarjeta: " + cBoxTipoTarjeta.getSelectedItem().toString() + "\t" + "\t" + "\t" + "\t" + "\t" + "HORA: " + fecha.getHoraTxt() + ":" + fecha.getMinutoTxt() + ":" + fecha.getSegundoTxt() + "\t" + "FECHA: " + fecha.getDiaSemanaTxt() + ", " + fecha.getDiaMesTxt() + ", " + fecha.getMesTxt() + ", " + fecha.getAnyoTxt() + "\r\n";

retFichero1 = "Alta Tarjeta" + "\t" + cBoxEsclavos.getSelectedItem().toString() + "\t" + tfNumeroTarjeta.getText() + "\t" + cBoxTipoTarjeta.getSelectedItem().toString() + "\t" + fecha.getHoraTxt() + ":" + fecha.getMinutoTxt() + ":" + fecha.getSegundoTxt() + "\t" + fecha.getDiaSemanaTxt() + "-" + fecha.getDiaMesTxt() + "-" + fecha.getMesTxt() + "-" + fecha.getAnyoTxt() + "\r\n";

remoto.append(cBoxEsclavos.getSelectedItem().toString());

}else if (str.equalsIgnoreCase("MAESTRA")) {

comando = 0x22;

for (int k = 0; k < cBoxEsclavos.getComponentCount(); k++) {

remoto.append(cBoxEsclavos.getItemAt(k).toString());

}

retFichero = "Alta Tarjeta" + "\t" + "\t" + "Numero de remoto: " + "Todos" + "\t" + "Numero Tarjeta: " + tfNumeroTarjeta.getText() + "\t" + "\t" + "\t" + "Tipo tarjeta: " + cBoxTipoTarjeta.getSelectedItem().toString() + "\t" + "\t" + "\t" + "\t" + "\t" + "HORA: " + fecha.getHoraTxt() + ":" + fecha.getMinutoTxt() + ":" + fecha.getSegundoTxt() + "\t" +"FECHA: " + fecha.getDiaSemanaTxt() + ", " + fecha.getDiaMesTxt() + ", " + fecha.getMesTxt() + ", " + fecha.getAnyoTxt() + "\r\n";

retFichero1 = "Alta Tarjeta" + "\t" + "Todos" + "\t" + tfNumeroTarjeta.getText() + "\t" + cBoxTipoTarjeta.getSelectedItem().toString() + "\t" + fecha.getHoraTxt() + ":" + fecha.getMinutoTxt() + ":" + fecha.getSegundoTxt() + "\t" + fecha.getDiaSemanaTxt() + "-" + fecha.getDiaMesTxt() + "-" + fecha.getMesTxt() + "-" + fecha.getAnyoTxt() + "\r\n";

}else {

comando = 0x23;

for (int k = 0; k < cBoxEsclavos.getComponentCount(); k++) {

remoto.append(cBoxEsclavos.getItemAt(k).toString());

}

retFichero = "Alta Tarjeta" + "\t" + "\t" + "Numero de remoto: " + "Todos" + "\t" + "Numero Tarjeta: " + tfNumeroTarjeta.getText() + "\t" + "\t" + "\t" + "Tipo tarjeta: " + cBoxTipoTarjeta.getSelectedItem().toString() + "\t" + "\t" + "\t" + "\t" + "\t" + "HORA: " + fecha.getHoraTxt() + ":" + fecha.getMinutoTxt() + ":" + fecha.getSegundoTxt() + "\t" +"FECHA: " + fecha.getDiaSemanaTxt() + ", " + fecha.getDiaMesTxt() + ", " + fecha.getMesTxt() + ", " + fecha.getAnyoTxt() + "\r\n";

retFichero1 = "Alta Tarjeta" + "\t" + "Todos" + "\t" + tfNumeroTarjeta.getText() + "\t" + cBoxTipoTarjeta.getSelectedItem().toString() + "\t" + fecha.getHoraTxt() + ":" + fecha.getMinutoTxt() + ":" + fecha.getSegundoTxt() + "\t" + fecha.getDiaSemanaTxt() + "-" + fecha.getDiaMesTxt() + "-" + fecha.getMesTxt() + "-" + fecha.getAnyoTxt() + "\r\n";

}

Comandos EnviarComando = new Comandos();

try {

ret = EnviarComando.altaTarjeta(direccion, numTerminal, comando, remoto, datoN1, datoN2, datoN3, datoN4);

ret += NL + "Numero de tarjeta: " + tfNumeroTarjeta.getText() + NL + NL;

} catch (Exception e) {

JOptionPane.showMessageDialog(null, "No se ha podido establecer conexi\u00F3n con la pasarela W0500", "ERROR", JOptionPane.ERROR\_MESSAGE);

}

try {

fichero.insertarDatos(retFichero, rutaFichero);

fichero.insertarDatos(retFichero1, rutaFichero1);

} catch (Exception e) {

JOptionPane.showMessageDialog(null, "ERROR al escribir en el fichero", "ERROR", JOptionPane.ERROR\_MESSAGE);

}

}else {

ret = "No hay remotos dados de alta en el sistema";

}

taHistorial.setText(ret);

tfNumeroTarjeta.setText("");

}

}

/\*\*

\* Clase que realiza la accion de dar de baja una tarjeta

\*

\* Solo es necesario introducir el numero de la tarjeta

\*

\* COMANDO 0x24

\*/

private class BtBajaTarjetaActionListener implements ActionListener {

private String NL = System.getProperty("line.separator");

public void actionPerformed(ActionEvent arg0) {

String direccion = "10.0.0.166";

String rutaFichero1 = "C:\\temporal\\copia1.txt";

String retFichero = "";

String retFichero1 = "";

String ret = "";

String numTarjeta = "";

String dato = "";

byte comando = 0x24;

byte numEsclavo = 0;

byte numTerminal = 0;//mi terminal

int nEsclavo = 0;

int miTerminal = 0;

int datoN1 = 0;

int datoN2 = 0;

int datoN3 = 0;

int datoN4 = 0;

if (cBoxEsclavos.getItemCount() != 0) {

// Capturamos el numero de esclavo

temp = cBoxEsclavos.getSelectedItem().toString();

nEsclavo = Integer.parseInt(temp);

nEsclavo = (nEsclavo) & 0xff;

numEsclavo = (byte) nEsclavo;

// Si no introducimos la direccion IP nos quedamos con la direccion por defecto 10.0.0.166

temp = tfdirIP.getText(); //Capturamos lo que hay en la ventana direccion IP.

tfdirIP.setText(""); //Borramos la ventana direccion IP.

if (!temp.equalsIgnoreCase("")) { //Si hemos introducido la direccion IP en la ventana direccion IP, la cogemos.

direccion = temp;

}

// Si no introducimos el numero de terminal nos quedamos con el introducido por defecto

temp = tfNumTerminal.getText(); //Capturamos lo que hay en la ventana Numero Terminal

tfNumTerminal.setText(""); //Borramos la ventana

if (!temp.equalsIgnoreCase("")) { //Si hemos introducido en numero de terminal lo cogemos

miTerminal = Integer.parseInt(temp);

miTerminal = (miTerminal) & 0xff;

numTerminal = (byte)miTerminal;

}

numTarjeta =tfNumeroTarjeta.getText();

for (int x = 0; x < 4; x++) {

switch (x) {

case 0:

dato = numTarjeta.substring(x,x+2);

datoN1 = Integer.parseInt(dato, 16);

datoN1 = (datoN1) & 0xff;

break;

case 1:

dato = numTarjeta.substring(x+1,x+3);

datoN2 = Integer.parseInt(dato, 16);

datoN2 = (datoN2) & 0xff;

break;

case 2:

dato = numTarjeta.substring(x+2,x+4);

datoN3 = Integer.parseInt(dato, 16);

datoN3 = (datoN3) & 0xff;

break;

case 3:

dato = numTarjeta.substring(x+3,x+5);

datoN4 = Integer.parseInt(dato, 16);

datoN4 = (datoN4) & 0xff;

break;

}

}

Fichero fichero = new Fichero();

Fecha fecha = new Fecha();

retFichero = "Baja Tarjeta" + "\t" + "\t" + "Numero de remoto: " + cBoxEsclavos.getSelectedItem().toString() + "\t" + "Numero Tarjeta: " + tfNumeroTarjeta.getText() + "\t" + "\t" + "\t" + "Tipo tarjeta: " + cBoxTipoTarjeta.getSelectedItem().toString() + "\t" + "\t" + "\t" + "\t" + "\t" + "HORA: " + fecha.getHoraTxt() + ":" + fecha.getMinutoTxt() + ":" + fecha.getSegundoTxt() + "\t" + "FECHA: " + fecha.getDiaSemanaTxt() + ", " + fecha.getDiaMesTxt() + ", " + fecha.getMesTxt() + ", " + fecha.getAnyoTxt() + "\r\n";

retFichero1 = "Baja Tarjeta" + "\t" + "-" + "\t" + tfNumeroTarjeta.getText() + "\t" + cBoxTipoTarjeta.getSelectedItem().toString() + "\t" + fecha.getHoraTxt() + ":" + fecha.getMinutoTxt() + ":" + fecha.getSegundoTxt() + "\t" + fecha.getDiaSemanaTxt() + "-" + fecha.getDiaMesTxt() + "-" + fecha.getMesTxt() + "-" + fecha.getAnyoTxt() + "\r\n";

Comandos EnviarComando = new Comandos();

try {

ret = EnviarComando.bajaTarjeta(direccion, numTerminal, comando, numEsclavo, datoN1, datoN2, datoN3, datoN4);

ret += NL + "Numero de tarjeta: " + tfNumeroTarjeta.getText();

} catch (Exception e) {

JOptionPane.showMessageDialog(null, "No se ha podido establecer conexi\u00F3n con la pasarela WK0500", "ERROR", JOptionPane.ERROR\_MESSAGE);

}

try {

fichero.insertarDatos(retFichero, rutaFichero);

fichero.insertarDatos(retFichero1, rutaFichero1);

} catch (Exception e) {

JOptionPane.showMessageDialog(null, "ERROR al escribir en el fichero", "ERROR", JOptionPane.ERROR\_MESSAGE);

}

}else {

ret = "No hay remotos dados de alta en el sistema";

}

taHistorial.setText(ret);

tfNumeroTarjeta.setText("");

}

}

/\*\*

\* Clase que realiza la accion de bloquear una puerta

\*

\* Hay que indicar el numero del esclavo

\*

\* COMANDO 0x25

\*/

private class BtBloquearPuertaActionListener implements ActionListener {

private String NL = System.getProperty("line.separator");

public void actionPerformed(ActionEvent arg0) {

String direccion = "10.0.0.166";

String rutaFichero1 = "C:\\temporal\\copia1.txt";

String retFichero = "";

String retFichero1 = "";

String ret = "";

byte numEsclavo = 0; //numero del remoto

byte numTerminal = 0; //mi terminal

byte comando = 0x25; //25 orden Bloquear una puerta (en hexadecimal)

int miTerminal = 0;

int nEsclavo = 0;

if (cBoxEsclavos.getItemCount() != 0) {

// Capturamos el numero de esclavo

temp = cBoxEsclavos.getSelectedItem().toString();

nEsclavo = Integer.parseInt(temp);

nEsclavo = (nEsclavo) & 0xff;

numEsclavo = (byte) nEsclavo;

// Si no introducimos la direccion IP nos quedamos con la direccion por defecto 10.0.0.166

temp = tfdirIP.getText(); //Capturamos lo que hay en la ventana direccion IP.

tfdirIP.setText(""); //Borramos la ventana direccion IP.

if (!temp.equalsIgnoreCase("")) { //Si hemos introducido la direccion IP en la ventana direccion IP, la cogemos.

direccion = temp;

}

// Si no introducimos el numero de terminal nos quedamos con el introducido por defecto

temp = tfNumTerminal.getText(); //Capturamos lo que hay en la ventana Numero Terminal

tfNumTerminal.setText(""); //Borramos la ventana

if (!temp.equalsIgnoreCase("")) { //Si hemos introducido en numero de terminal lo cogemos

miTerminal = Integer.parseInt(temp);

miTerminal = (miTerminal) & 0xff;

numTerminal = (byte)miTerminal;

}

Fichero fichero = new Fichero();

Fecha fecha = new Fecha();

retFichero = "Puerta Bloqueada" + "\t" + "Numero de remoto: " + cBoxEsclavos.getSelectedItem().toString() + "\t" + "\t" + "\t" + "\t" + "\t" + "\t" + "\t" + "\t" + "\t" + "\t" + "\t" + "HORA: " + fecha.getHoraTxt() + ":" + fecha.getMinutoTxt() + ":" + fecha.getSegundoTxt() + "\t" + "FECHA: " + fecha.getDiaSemanaTxt() + ", " + fecha.getDiaMesTxt() + ", " + fecha.getMesTxt() + ", " + fecha.getAnyoTxt() + "\r\n";

retFichero1 = "Puerta Bloqueada" + "\t" + cBoxEsclavos.getSelectedItem().toString() + "\r\n";

Comandos EnviarComando = new Comandos();

try {

ret = EnviarComando.accionesPuerta(direccion, numTerminal, comando, numEsclavo);

} catch (Exception e) {

JOptionPane.showMessageDialog(null, "No se ha podido establecer conexi\u00F3n con la pasarela WK0500", "ERROR", JOptionPane.ERROR\_MESSAGE);

}

try {

fichero.insertarDatos(retFichero, rutaFichero);

fichero.insertarDatos(retFichero1, rutaFichero1);

} catch (Exception e) {

JOptionPane.showMessageDialog(null, "ERROR al escribir en el fichero", "ERROR", JOptionPane.ERROR\_MESSAGE);

}

ret += NL + NL + "Puerta Bloqueada " + NL + "N\u00FAmero de remoto: " + cBoxEsclavos.getSelectedItem().toString() + NL;

}else {

ret = "No hay remotos dados de alta en el sistema";

}

taHistorial.setText(ret);

}

}

/\*\*

\* Clase que realiza la accion de desbloquear una puerta

\*

\* Hay que indicar el numero del esclavo

\*

\* COMANDO 0x26

\*/

private class BtDesbloquearPuertaActionListener implements ActionListener {

private String NL = System.getProperty("line.separator");

public void actionPerformed(ActionEvent arg0) {

String direccion = "10.0.0.166";

String rutaFichero1 = "C:\\temporal\\copia1.txt";

String retFichero = "";

String retFichero1 = "";

String ret = "";

byte numEsclavo = 0; //numero del remoto

byte numTerminal = 0; //mi terminal

byte comando = 0x26; //20 orden desbloquear una puerta (en hexadecimal)

int miTerminal = 0;

int nEsclavo = 0;

if (cBoxEsclavos.getItemCount() != 0) {

// Capturamos el numero de esclavo

temp = cBoxEsclavos.getSelectedItem().toString();

nEsclavo = Integer.parseInt(temp);

nEsclavo = (nEsclavo) & 0xff;

numEsclavo = (byte) nEsclavo;

// Si no introducimos la direccion IP nos quedamos con la direccion por defecto 10.0.0.166

temp = tfdirIP.getText(); //Capturamos lo que hay en la ventana direccion IP.

tfdirIP.setText(""); //Borramos la ventana direccion IP.

if (!temp.equalsIgnoreCase("")) { //Si hemos introducido la direccion IP en la ventana direccion IP, la cogemos.

direccion = temp;

}

// Si no introducimos el numero de terminal nos quedamos con el introducido por defecto

temp = tfNumTerminal.getText(); //Capturamos lo que hay en la ventana Numero Terminal

tfNumTerminal.setText(""); //Borramos la ventana

if (!temp.equalsIgnoreCase("")) { //Si hemos introducido en numero de terminal lo cogemos

miTerminal = Integer.parseInt(temp);

miTerminal = (miTerminal) & 0xff;

numTerminal = (byte)miTerminal;

}

Fichero fichero = new Fichero();

Fecha fecha = new Fecha();

retFichero = "Puerta Desbloqueada" + "\t" + "Numero de remoto: " + cBoxEsclavos.getSelectedItem().toString() + "\t" + "\t" + "\t" + "\t" + "\t" + "\t" + "\t" + "\t" + "\t" + "\t" + "\t" + "HORA: " + fecha.getHoraTxt() + ":" + fecha.getMinutoTxt() + ":" + fecha.getSegundoTxt() + "\t" + "FECHA: " + fecha.getDiaSemanaTxt() + ", " + fecha.getDiaMesTxt() + ", " + fecha.getMesTxt() + ", " + fecha.getAnyoTxt() + "\r\n";

retFichero1 = "Puerta Desbloqueada" + "\t" + cBoxEsclavos.getSelectedItem().toString() + "\r\n";

Comandos EnviarComando = new Comandos();

try {

ret = EnviarComando.accionesPuerta(direccion, numTerminal, comando, numEsclavo);

} catch (Exception e) {

JOptionPane.showMessageDialog(null, "No se ha podido establecer conexi\u00F3n con la pasarela WK0500", "ERROR", JOptionPane.ERROR\_MESSAGE);

}

try {

fichero.insertarDatos(retFichero, rutaFichero);

fichero.insertarDatos(retFichero1, rutaFichero1);

} catch (Exception e) {

JOptionPane.showMessageDialog(null, "ERROR al escribir en el fichero", "ERROR", JOptionPane.ERROR\_MESSAGE);

}

ret += NL + NL + "Puerta Desbloqueada " + NL + "N\u00FAmero de remoto: " + cBoxEsclavos.getSelectedItem().toString() + NL;

}else {

ret = "No hay remotos dados de alta en el sistema";

}

taHistorial.setText(ret);

}

}

/\*\*

\* Clase que realiza el borrado Total de una cerradura

\*

\* Hay que indicar el numero del esclavo

\*

\* COMANDO 0x28

\*/

private class BtBorrarCerraduraActionListener implements ActionListener {

private String NL = System.getProperty("line.separator");

public void actionPerformed(ActionEvent arg0) {

String direccion = "10.0.0.166";

String rutaFichero1 = "C:\\temporal\\copia1.txt";

String retFichero = "";

String retFichero1 = "";

String ret = "";

byte numEsclavo = 0; //numero del remoto

byte numTerminal = 0; //mi terminal

byte comando = 0x28; //28 orden Borrar cerradura (en hexadecimal)

int miTerminal = 0;

int nEsclavo = 0;

if (cBoxEsclavos.getItemCount() != 0) {

// Capturamos el numero de esclavo

temp = cBoxEsclavos.getSelectedItem().toString();

nEsclavo = Integer.parseInt(temp);

nEsclavo = (nEsclavo) & 0xff;

numEsclavo = (byte) nEsclavo;

// Si no introducimos la direccion IP nos quedamos con la direccion por defecto 10.0.0.166

temp = tfdirIP.getText(); //Capturamos lo que hay en la ventana direccion IP.

tfdirIP.setText(""); //Borramos la ventana direccion IP.

if (!temp.equalsIgnoreCase("")) { //Si hemos introducido la direccion IP en la ventana direccion IP, la cogemos.

direccion = temp;

}

// Si no introducimos el numero de terminal nos quedamos con el introducido por defecto

temp = tfNumTerminal.getText(); //Capturamos lo que hay en la ventana Numero Terminal

tfNumTerminal.setText(""); //Borramos la ventana

if (!temp.equalsIgnoreCase("")) { //Si hemos introducido en numero de terminal lo cogemos

miTerminal = Integer.parseInt(temp);

miTerminal = (miTerminal) & 0xff;

numTerminal = (byte)miTerminal;

}

Fichero fichero = new Fichero();

Fecha fecha = new Fecha();

retFichero = "Cerradura Borrada" + "\t" + "Numero de remoto: " + cBoxEsclavos.getSelectedItem().toString() + "\t" + "\t" + "\t" + "\t" + "\t" + "\t" + "\t" + "\t" + "\t" + "\t" + "\t" + "HORA: " + fecha.getHoraTxt() + ":" + fecha.getMinutoTxt() + ":" + fecha.getSegundoTxt() + "\t" + "FECHA: " + fecha.getDiaSemanaTxt() + ", " + fecha.getDiaMesTxt() + ", " + fecha.getMesTxt() + ", " + fecha.getAnyoTxt() + "\r\n";

retFichero1 = "Cerradura Borrada" + "\t" + cBoxEsclavos.getSelectedItem().toString() + "\r\n";

Comandos EnviarComando = new Comandos();

try {

ret = EnviarComando.accionesPuerta(direccion, numTerminal, comando, numEsclavo);

} catch (Exception e) {

JOptionPane.showMessageDialog(null, "No se ha podido establecer conexi\u00F3n con la pasarela WK0500", "ERROR", JOptionPane.ERROR\_MESSAGE);

}

try {

fichero.insertarDatos(retFichero, rutaFichero);

fichero.insertarDatos(retFichero1, rutaFichero1);

} catch (Exception e) {

JOptionPane.showMessageDialog(null, "ERROR al escribir en el fichero", "ERROR", JOptionPane.ERROR\_MESSAGE);

}

ret += NL + "Borrado Total Cerradura " + NL + "N\u00FAmero de remoto: " + cBoxEsclavos.getSelectedItem().toString() + NL;

}else {

ret = "No hay remotos dados de alta en el sistema";

}

taHistorial.setText(ret);

}

}

/\*\*

\* Boton para dar de alta un esclavo en el sistema

\*

\* COMANDO 0x29

\*/

private class BtAltaEsclavoActionListener implements ActionListener {

private String NL = System.getProperty("line.separator");

public void actionPerformed(ActionEvent arg0) {

String direccion = "10.0.0.166";

String rutaFichero1 = "C:\\temporal\\copia1.txt";

String retFichero = "";

String retFichero1 = "";

String ret = "";

String remoto = "";

StringBuffer retEsclavos = new StringBuffer();

String esclavo = " ";

byte numEsclavo = 0; //numero del remoto

byte numTerminal = 0; //mi terminal

byte comando = 0x29; //29 orden Alta Esclavo (en hexadecimal)

int miTerminal = 0;

int nEsclavo = 0;

boolean trobat = false;

temp = tfnEsclavo.getText();

if (temp.equalsIgnoreCase("")) {

temp = "0";

}

nEsclavo = Integer.parseInt(temp);

nEsclavo = (nEsclavo) & 0xff;

numEsclavo = (byte) nEsclavo;

if ((nEsclavo != 0)) {

remoto = tfnEsclavo.getText();

Comandos EnviarComando = new Comandos();

try {

comando = 0x2A;

retEsclavos = EnviarComando.inicializarEsclavos(direccion, numTerminal, comando);

for (int k = 0; k < (retEsclavos.length())/2; k++) {

esclavo = (String)retEsclavos.subSequence(k+k, k+2+k);

if (esclavo.equalsIgnoreCase(remoto)) {

trobat = true;

}

}

} catch (Exception e) {

JOptionPane.showMessageDialog(null, "Puntero fuera de rango", "ERROR", JOptionPane.ERROR\_MESSAGE);

}

if (!trobat) {

// Si no introducimos la direccion IP nos quedamos con la direccion por defecto 10.0.0.166

temp = tfdirIP.getText(); //Capturamos lo que hay en la ventana direccion IP.

tfdirIP.setText(""); //Borramos la ventana direccion IP.

if (!temp.equalsIgnoreCase("")) { //Si hemos introducido la direccion IP en la ventana direccion IP, la cogemos.

direccion = temp;

}

// Si no introducimos el numero de terminal nos quedamos con el introducido por defecto

temp = tfNumTerminal.getText(); //Capturamos lo que hay en la ventana Numero Terminal

tfNumTerminal.setText(""); //Borramos la ventana

if (!temp.equalsIgnoreCase("")) { //Si hemos introducido en numero de terminal lo cogemos

miTerminal = Integer.parseInt(temp);

miTerminal = (miTerminal) & 0xff;

numTerminal = (byte)miTerminal;

}

try {

comando = 0x29;

ret = EnviarComando.accionesPuerta(direccion, numTerminal, comando, numEsclavo);

} catch (Exception e) {

JOptionPane.showMessageDialog(null, "No se ha podido establecer conexi\u00F3n con la pasarela WK0500", "ERROR", JOptionPane.ERROR\_MESSAGE);

}

cBoxEsclavos.addItem(String.valueOf(numEsclavo));

Fichero fichero = new Fichero();

Fecha fecha = new Fecha();

retFichero = "Alta remoto" + "\t" + "Numero de remoto: " + cBoxEsclavos.getSelectedItem().toString()+ "\t" + "\t" + "\t" + "\t" + "\t" + "\t" + "\t" + "\t" + "\t" + "\t" + "\t" + "HORA: " + fecha.getHoraTxt() + ":" + fecha.getMinutoTxt() + ":" + fecha.getSegundoTxt() + "\t" + "FECHA: " + fecha.getDiaSemanaTxt() + ", " + fecha.getDiaMesTxt() + ", " + fecha.getMesTxt() + ", " + fecha.getAnyoTxt() + "\r\n";

retFichero1 = "Alta remoto" + "\t" + cBoxEsclavos.getSelectedItem().toString() + "\r\n";

try {

fichero.insertarDatos(retFichero, rutaFichero);

fichero.insertarDatos(retFichero1, rutaFichero1);

} catch (IOException e) {

JOptionPane.showMessageDialog(null, "ERROR al escribir en el fichero", "ERROR", JOptionPane.ERROR\_MESSAGE);

}

ret += NL + "Alta Esclavo " + NL + "N\u00FAmero de remoto: " + String.valueOf(numEsclavo) + NL;

}else {

ret = "El n\u00FAmero de esclavo introducido ya existe.";

}

}else {

ret = "Introduzca un n\u00FAmero de remoto";

}

taHistorial.setText(ret);

tfnEsclavo.setText("");

}

}

/\*\*

\* Boton para actualizar la hora en la pasarela WK0500

\*

\* COMANDO 0x2B

\*/

private class BtActualizarHoraActionListener implements ActionListener {

public void actionPerformed(ActionEvent arg0) {

String ret = "";

String direccion = "10.0.0.166";

String retFichero = "";

byte numTerminal = 0; //mi terminal

byte comando = 0x2B; //2B orden Actualizar Hora (en hexadecimal)

// Si no introducimos la direccion IP nos quedamos con la direccion por defecto 10.0.0.166

temp = tfdirIP.getText(); //Capturamos lo que hay en la ventana direccion IP.

tfdirIP.setText(""); //Borramos la ventana direccion IP.

if (!temp.equalsIgnoreCase("")) { //Si hemos introducido la direccion IP en la ventana direccion IP, la cogemos.

direccion = temp;

}

Comandos EnviarComando = new Comandos();

try {

ret += EnviarComando.actualizarHora(direccion, numTerminal, comando);

} catch (Exception e) {

JOptionPane.showMessageDialog(null, "No se ha podido establecer conexi\u00F3n con la pasarela WK0500", "ERROR", JOptionPane.ERROR\_MESSAGE);

}

taHistorial.setText(ret);

Fichero fichero = new Fichero();

Fecha fecha = new Fecha();

retFichero = "Hora actualizada" + "\t" + "\t" + "\t" + "\t" + "\t" + "\t" + "\t" + "\t" + "\t" + "\t" + "\t" + "\t" + "\t" + "HORA: " + fecha.getHoraTxt() + ":" + fecha.getMinutoTxt() + ":" + fecha.getSegundoTxt() + "\t" + "FECHA: " + fecha.getDiaSemanaTxt() + ", " + fecha.getDiaMesTxt() + ", " + fecha.getMesTxt() + ", " + fecha.getAnyoTxt() + "\r\n";

try {

fichero.insertarDatos(retFichero, rutaFichero);

} catch (IOException e) {

JOptionPane.showMessageDialog(null, "ERROR al escribir en el fichero", "ERROR", JOptionPane.ERROR\_MESSAGE);

}

}

}

/\*\*

\* Boton para leer los registros

\*

\* La cantidad de registros a leer se introducen en un cuadro de texto

\*

\* COMANDO 0x2E

\*/

private class BtLeerRegistrosActionListener implements ActionListener {

private String NL = System.getProperty("line.separator");

public void actionPerformed(ActionEvent arg0) {

String ret = "";

String direccion = "10.0.0.166";

String retFichero = "";

int cantidad = 15; //numero de palabras a leer

int nEsclavo = 0;

byte numEsclavo = 0; //numero del remoto

byte comando = 0x2E; //2E orden Leer Registros (en hexadecimal)

byte numTerminal = 0; //mi terminal

if (cBoxEsclavos.getItemCount() != 0) {

// Capturamos el numero de esclavo

temp = cBoxEsclavos.getSelectedItem().toString();

nEsclavo = Integer.parseInt(temp);

nEsclavo = (nEsclavo) & 0xff;

numEsclavo = (byte) nEsclavo;

// Si no introducimos el numero de terminal nos quedamos con el introducido por defecto

temp = tfCantidadReg.getText(); //Capturamos lo que hay en la ventana Numero Terminal

tfCantidadReg.setText(""); //Borramos la ventana

if (!temp.equalsIgnoreCase("")) { //Si hemos introducido en numero de terminal lo cogemos

cantidad = Integer.parseInt(temp);

if (cantidad > 100) {

cantidad = 0;

ret += "Ha introducido una cantidad demasiado elevada de registros" + NL + "Introduzca como maximo: 100" + NL;

}

}

Comandos EnviarComando = new Comandos();

try {

ret += EnviarComando.leerRegistros(direccion, numTerminal, comando, cantidad, numEsclavo);

} catch (Exception e) {

JOptionPane.showMessageDialog(null, "No se ha podido establecer conexi\u00F3n con la pasarela WK0500", "ERROR", JOptionPane.ERROR\_MESSAGE);

}

Fichero fichero = new Fichero();

Fecha fecha = new Fecha();

retFichero = "Leer registros" + "\t" + "\t" + "Registros leidos: " + String.valueOf(cantidad) + "\t" + "\t" + "\t" + "\t" + "\t" + "\t" + "\t" + "\t" + "\t" + "\t" + "\t" + "HORA: " + fecha.getHoraTxt() + ":" + fecha.getMinutoTxt() + ":" + fecha.getSegundoTxt() + "\t" + "FECHA: " + fecha.getDiaSemanaTxt() + ", " + fecha.getDiaMesTxt() + ", " + fecha.getMesTxt() + ", " + fecha.getAnyoTxt() + "\r\n";

try {

fichero.insertarDatos(retFichero, rutaFichero);

} catch (IOException e) {

JOptionPane.showMessageDialog(null, "ERROR al escribir en el fichero", "ERROR", JOptionPane.ERROR\_MESSAGE);

}

}else {

ret = "No hay remotos dados de alta en el sistema";

}

taHistorial.setText(ret);

}

}

/\*\*

\* Boton para borrar la ventana de informacion

\*/

private class BtBorrarVentanaInfoActionListener implements ActionListener {

public void actionPerformed(ActionEvent e) {

taHistorial.setText("");

}

}

/\*\*

\* Ventana de informacion del menu Ayuda

\*/

class VentanaInfo extends JFrame {

private static final long serialVersionUID = 1L;

private String NL = System.getProperty("line.separator");

public VentanaInfo ()

{

inicializar();

}

private void inicializar() {

JLabel label = new JLabel();

getContentPane().add(label);

this.setTitle("Proyecto Arduino");

taInformacion = new JTextArea();

taInformacion.setSize(300, 300);

getContentPane().add(taInformacion);

taInformacion.setText("Autores:" + NL + "Juan Francisco Grau" + NL + "Arturo Aparicio" + NL + "Santiago Carbo" + NL + NL + "Departamento Electricidad/Electronica" + NL + "IES TIRANT LO BLANC" + NL + "GANDIA" + NL + "VALENCIA (ESPANYA)" + NL + NL + "Control fichajes Rondas" + NL + "Version 2.3");

this.setSize(300, 300);

this.setVisible(true);

}

}

/\*\*

\* Menu Archivo Cerrar el progama

\*/

private class MenuSalirActionListener implements ActionListener {

public void actionPerformed(ActionEvent arg0) {

System.exit(0);

}

}

/\*\*

\* Menu Comando Abrir Puerta

\*/

private class MenuAbrirPuertaActionListener implements ActionListener {

public void actionPerformed(ActionEvent e) {

BtAbrirPuertaActionListener abrirPuerta = new BtAbrirPuertaActionListener();

abrirPuerta.actionPerformed(e);

}

}

/\*\*

\* Menu Comando dar de Alta una Tarjeta

\*

\*/

private class MenuAltaTarjetaActionListener implements ActionListener {

public void actionPerformed(ActionEvent e) {

BtAltaTarjetaActionListener altaTarjeta = new BtAltaTarjetaActionListener();

altaTarjeta.actionPerformed(e);

}

}

/\*\*

\* Menu Comando dar de Baja una Tarjeta

\*

\*/

private class MenuBajaTarjetaActionListener implements ActionListener {

public void actionPerformed(ActionEvent e) {

BtBajaTarjetaActionListener bajaTarjeta = new BtBajaTarjetaActionListener();

bajaTarjeta.actionPerformed(e);

}

}

/\*\*

\* Menu Comando Bloquear una Puerta

\*

\*/

private class MenuBloquearPuertaActionListener implements ActionListener {

public void actionPerformed(ActionEvent e) {

BtBloquearPuertaActionListener bloquearPuerta = new BtBloquearPuertaActionListener();

bloquearPuerta.actionPerformed(e);

}

}

/\*\*

\* Menu Comando Desbloquear una Puerta

\*

\*/

private class MenuDesbloquearPuertaActionListener implements ActionListener {

public void actionPerformed(ActionEvent e) {

BtDesbloquearPuertaActionListener desbloquearPuerta = new BtDesbloquearPuertaActionListener();

desbloquearPuerta.actionPerformed(e);

}

}

/\*\*

\* Menu Comando Borrar Cerradura

\*

\*/

private class MenuBorrarCerraduraActionListener implements ActionListener {

public void actionPerformed(ActionEvent e) {

BtBorrarCerraduraActionListener borrarCerradura = new BtBorrarCerraduraActionListener();

borrarCerradura.actionPerformed(e);

}

}

/\*\*

\* Menu Comando Leer Registros

\*

\*/

private class MenuLeerRegistrosActionListener implements ActionListener {

public void actionPerformed(ActionEvent e) {

BtLeerRegistrosActionListener leerRegistros = new BtLeerRegistrosActionListener();

leerRegistros.actionPerformed(e);

}

}

/\*\*

\* Menu Comando Actualizar Hora

\*

\*/

private class MenuActualizarHoraActionListener implements ActionListener {

public void actionPerformed(ActionEvent e) {

BtActualizarHoraActionListener actualizarHora = new BtActualizarHoraActionListener();

actualizarHora.actionPerformed(e);

}

}

/\*\*

\* Menu Ayuda Acerca de....

\*/

private class MntmAcercaDeActionListener implements ActionListener {

public void actionPerformed(ActionEvent arg0) {

VentanaInfo acercaDe = new VentanaInfo();

acercaDe.inicializar();

}

}

/\*\*

\* Menu Archivo Guardar Historial

\*/

private class MenuGuardarHistorialActionListener implements ActionListener {

public void actionPerformed(ActionEvent arg0) {

//creamos el dialogo, le decimos quien es su padre, titulo y modal.

FileDialog dialogoFichero = new FileDialog(ventana,"Guardar Fichero", FileDialog.SAVE);

//Lo hacemos visible.

dialogoFichero.setVisible(true);

//Si cancelamos la accion no hacemos nada de esta forma evitamos que el puntero se desborde

if (!(dialogoFichero.getFile() == null)) {

File f = new File(dialogoFichero.getDirectory(), dialogoFichero.getFile());

rutaFichero = f.toString();

}

}

}

/\*\*

\* Menu Archivo Abrir Historial

\*/

private class MenuAbrirHistorialActionListener implements ActionListener {

public void actionPerformed(ActionEvent arg0) {

String ret = "";

//creamos el dialogo, le decimos quien es su padre, titulo y modal.

FileDialog dialogoFichero = new FileDialog(ventana, "Elija un Archivo", FileDialog.LOAD);

//Lo hacemos visible.

dialogoFichero.setVisible(true);

//Si cancelamos la accion no hacemos nada de esta forma evitamos que el puntero se desborde

if (!(dialogoFichero.getFile() == null)) {

File f = new File(dialogoFichero.getDirectory(), dialogoFichero.getFile());

Fichero fichero = new Fichero();

try {

ret = fichero.leerDatos(f.toString());

taHistorial.setText(ret);

} catch (IOException e) {

JOptionPane.showMessageDialog(null, "ERROR al leer datos del fichero", "ERROR", JOptionPane.ERROR\_MESSAGE);

}

}

}

}

public static void main (String args[]) throws ParseException, IOException

{

setVentana(new LectorTarjetas());

}

public static LectorTarjetas getVentana() {

return ventana;

}

public static void setVentana(LectorTarjetas ventana) {

LectorTarjetas.ventana = ventana;

}

}