Librerías:

#include <SoftwareSerial.h>

SoftwareSerial ssrfid(11, 10);

Se incluye la librería SoftwareSerial para la comunicación serial a través de pines de software. Se crea un objeto ssrfid de la clase SoftwareSerial para la interacción con el lector RFID.

Variables Globales:

const int BUFFER\_SIZE = 14;

const int DATA\_SIZE = 10;

const int DATA\_VERSION\_SIZE = 2;

const int DATA\_TAG\_SIZE = 8;

const int CHECKSUM\_SIZE = 2;

SoftwareSerial ssrfid(11, 10);

uint8\_t buffer[BUFFER\_SIZE];

int buffer\_index = 0;

const long TagsAlmacenados[] = {3861665, 11463764, 6251477, 11482160, 6250364, 6289464, 6236167, 3233189, 3859906, 3856153, 3928577, 6494497, 6469736, 11256247, 6244441};

long tagsDetectados[10];

int tagsDetectadosIndex = 0;

int SalidaLedAzul = 12;

int SalidaLedRojo = 13;

int SalidaRele = 4;

long tag;

Se definen constantes para los tamaños del búfer y se inicializan las variables globales, incluyendo el objeto ssrfid y arreglos para almacenar etiquetas RFID.

Función setup():

void setup() {

Serial.begin(9600);

ssrfid.begin(9600);

ssrfid.listen();

pinMode(SalidaRele, OUTPUT);

pinMode(SalidaLedAzul, OUTPUT);

pinMode(SalidaLedRojo, OUTPUT);

Serial.println("INICIALIZACIÓN COMPLETADA");

digitalWrite(SalidaRele, HIGH);

}

Se configura la comunicación serial, se inicializan pines de salida y se imprime un mensaje de inicialización. Se establece el estado inicial del relé en alto.

Función loop():

void loop() {

if (ssrfid.available() > 0) {

bool call\_extract\_tag = false;

int ssvalue = ssrfid.read();

if (ssvalue == -1) {

return;

}

if (ssvalue == 2) {

buffer\_index = 0;

} else if (ssvalue == 3) {

call\_extract\_tag = true;

}

if (buffer\_index >= BUFFER\_SIZE) {

Serial.println("Error: ¡Desbordamiento de búfer detectado!");

return;

}

buffer[buffer\_index++] = ssvalue;

if (call\_extract\_tag) {

if (buffer\_index == BUFFER\_SIZE) {

tag = extract\_tag();

if (!tag\_detected(tag)) {

tagsDetectados[tagsDetectadosIndex++] = tag;

for (int i = 0; i < sizeof(TagsAlmacenados) / sizeof(TagsAlmacenados[0]); i++) {

if (tag == TagsAlmacenados[i]) {

activar\_salida();

break;

}

}

} else {

Serial.println("Error: Tag duplicado detectado.");

forceReset();

}

buffer\_index = 0;

} else {

buffer\_index = 0;

return;

}

}

}

}

El bucle principal verifica la disponibilidad de datos del lector RFID. Procesa los datos, verifica condiciones y realiza acciones correspondientes.

Función extract\_tag():

long extract\_tag() {

uint8\_t \*msg\_data = buffer + 1;

uint8\_t \*msg\_data\_tag = msg\_data + 2;

long tag = hexstr\_to\_value(msg\_data\_tag, DATA\_TAG\_SIZE);

Serial.print("Etiqueta extraída: ");

Serial.println(tag);

return tag;

}

Extrae la etiqueta RFID del búfer y la convierte a un valor numérico. Imprime la etiqueta extraída en el puerto serie.

Función hexstr\_to\_value():

long hexstr\_to\_value(uint8\_t \*str, unsigned int length) {

char copy[length + 1];

for (int i = 0; i < length; i++) {

copy[i] = (char)str[i];

}

copy[length] = '\0';

long value = strtol(copy, NULL, 16);

return value;

}

Convierte una cadena hexadecimal a un valor numérico utilizando la función strtol.

Función tag\_detected():

bool tag\_detected(long current\_tag) {

for (int i = 0; i < tagsDetectadosIndex; i++) {

if (current\_tag == tagsDetectados[i]) {

return true;

}

}

return false;

}

Verifica si una etiqueta RFID ya ha sido detectada anteriormente.

Función activar\_salida():

void activar\_salida() {

digitalWrite(SalidaRele, LOW);

digitalWrite(SalidaLedAzul, HIGH);

Serial.println("Abriendo Puerta");

delay(5000);

digitalWrite(SalidaRele, HIGH);

digitalWrite(SalidaLedAzul, LOW);

Serial.println("Cerrando Puerta");

}

Activa la salida, en este caso, controla un relé y un LED azul. Imprime mensajes en el puerto serie indicando la acción realizada.

Función forceReset():

void forceReset() {

asm volatile (" jmp 0");

}

Utiliza la instrucción de salto incondicional (jmp 0) para forzar un reinicio del programa Arduino.