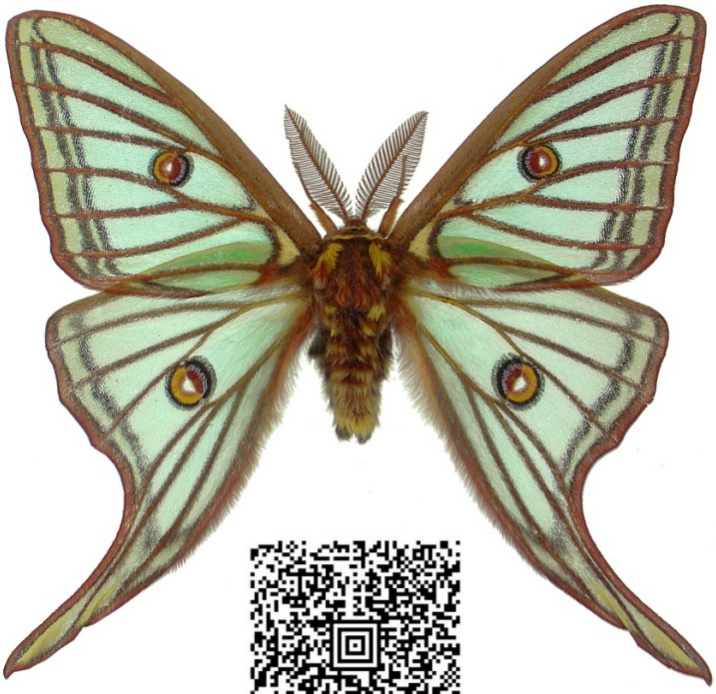
Monografías electrónicas S.E.A
ISSN: 2386-5318
nº 6



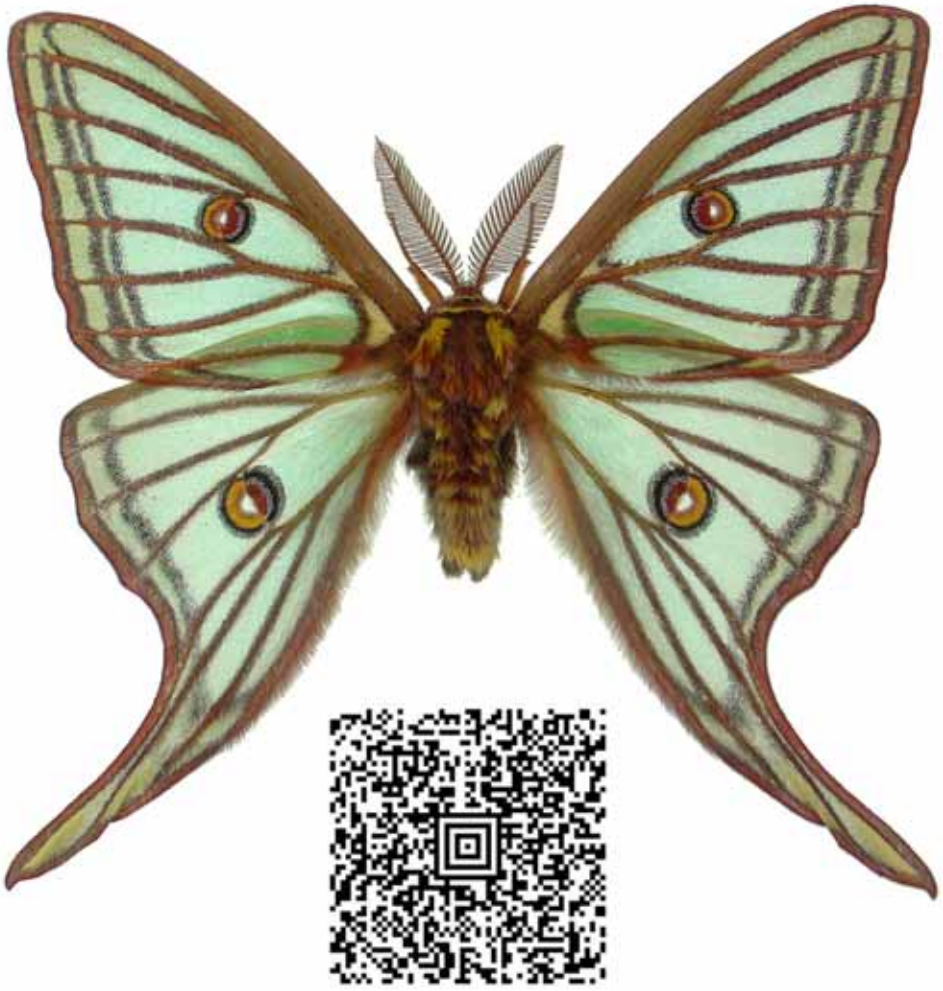
Los códigos gráficos y su procesado dinámico aplicado a la entomología.

Graphics codes and applied entomology dynamic processing.

Rafael Magro

Sociedad Entomológica Aragonesa
www.sea-entomologia.org
Zaragoza, 30-06-2014

Monografías electrónicas S.E.A. nº 6

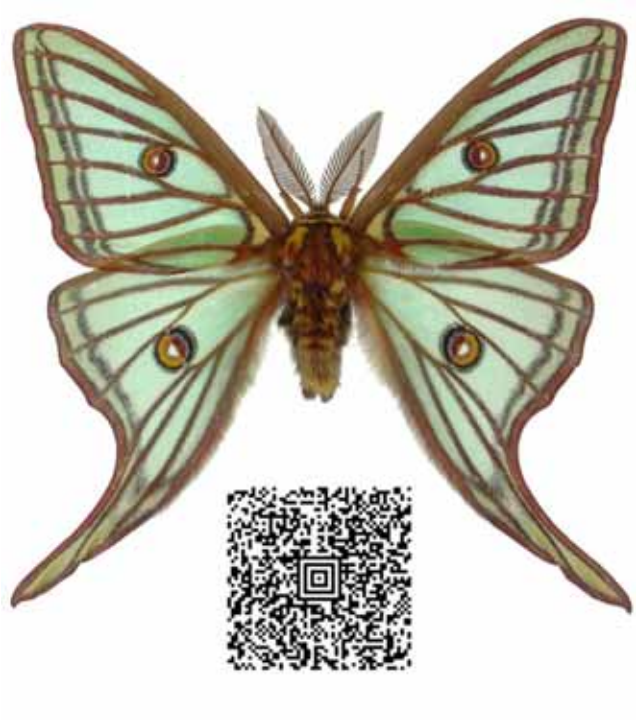


Los códigos gráficos y su procesado dinámico aplicado a la entomología.

Graphics codes and applied entomology dynamic processing.

Rafael Magro

Sociedad **E**ntomológica **A**ragonesa
www.sea-entomologia.org
Zaragoza, 30-06-2014



Los códigos gráficos y su procesado dinámico aplicado a la entomología.

Rafael Magro

correolaboratorio@yahoo.es

Resumen: En este trabajo se enumeran los métodos más usuales para el procesado de códigos 1 D y 2D. Se incluye una breve descripción de los tipos y la tecnología necesaria para la lectura óptica inalámbrica y sus posibles usos en biología y entomología. Exponemos las condiciones ideales para la correcta adquisición visual del valor de los símbolos impresos en micro-etiquetas y detallamos los errores más frecuentes de impresión, color e imagen. Se registra el tiempo de codificación y decodificación en relación con el contingente de caracteres numéricos y alfa-numéricos encriptados en el vector y su tamaño. Explicamos de qué manera los datos obtenidos se tratan de forma dinámica y vinculada a la algoritmia de los aplicativos con acceso a tablas de datos, listas y otros procedimientos. Se adjunta código fuente y se pormenorizan las partes más importantes.

Palabras clave: Códigos 1D-2D, algoritmos, entomología.

Graphics codes and applied entomology dynamic processing.

Abstract: In this essay we list the most common methods for the processing of 1D and 2D codes, including a brief description of the types and the necessary technology for wireless optical reading and its possible uses in biology and entomology. We also quote the ideal conditions for the correct visual acquisition of the value of the printed symbols in micro-tags and the most frequent errors in printing, color and image details. The time of encoding and decoding is recorded in relation to the contingent of encrypted numeric and alphanumeric characters in the vector and its size. We explain how the data are treated dynamically and linked to the algorithmic of software with access to data tables, lists and other procedures. Attached source code and the most important parts are detailed.

Key words: 1D-2D codes, algorithms, entomology.

Ilustraciones, gráficos y fotografías: R. Magro.

Lepidóptero de la portada y de las páginas 1 y 2:
Graellsia isabelae (Graells, 1849).

Texto encriptado en los códigos AZTEC de la portada y las páginas 1 y 2:

“Al augusto nombre de Su Majestad la Reina Doña Isabel II, dedico esta magnífica *Saturnia*, único representante en Europa de la sección a que pertenecen la *Diana*, *Luna*, *Selene*, *Isis* y otras divinidades menos positivas que la nuestra". Mariano de La Paz Graells, 1849.

Índice de apartados

Convenciones 8
Introducción 9
Material y método 10
Breve historia de los códigos 1D y 2D 11
1932 11
1948 11
1961 12
1970-79 12
1980-89 12
1990-99 12
Siglo XXI 13
Tipos de códigos 13
3DI 13
ARRAYTAG 13
AZTEC 14
BEETAGG 14
BIDI 14
BLOTCODE 14
CODABAR 15
CODACBLOCK 15
CODE11 15
CODE16K 15
CODE39 15
CODE49 15
CODE9 16
CODE93 16
CODE128 16
CODEONE 16
COLORCODE 16
CPCODE 16
COOL–DATAMATRIX 17
DATAGLYPHS 17
DATAMATRIX 17
DATASTRIP 17
DOTCODE 18
EAN8, EAN13 y EAN128 18
INTERLEAVED 18
HCCB 19
ISBN 19
ISSN 19
ITF 19
ITF14 20
MAXICODE 20
MCODE 20
MICROQR 20

MINICODE 21
PDF417 21
POSNETCODE y PLNETCODE 21
QRCODE 21
QUICKMARK 22
SHOTCODE 22
SNOWFLAKECODE 22
TRILLCODE 22
UCC/EAN128 23
UPCE 32
UPCA 23
ULTRACODE 23
Introducción a los usos de los códigos 1Dy 2D en la entomología 24
Introducción a los rudimentos para trabajar con códigos 24
Codificación 25
Transmutación 25
Adquisición 26
Decodificación 26
Lanzamiento pasivo 26
Lanzamiento dinámico 26
Doble decodificación 27
La codificación en profundidad 28
Código EAN8 28
Método para calcular el dígito verificador EAN8 28
Algoritmia para el cálculo del dígito verificador EAN8 29
¿Utilizar fuentes EAN o crear gráficos vectoriales? 29
Uso de códigos en color 29
Posicionamiento 31
Prácticas desaconsejadas 32
Truncamiento 32
Supresión de datos 32
No incluir numeración 32
No calcular el dígito de control 33
Uso de colores incompatibles 33
Hechura de las fuentes 33
Intercalación de espacios 33
Código EAN13 33
AZTEC 34
Algoritmia para el cálculo del código AZTEC 34
Uso de códigos AZTEC en color 34
Prácticas desaconsejadas 34
Truncamiento 34
Supresión de datos 34
No incluir numeración 34
Uso de logos 34
Uso de marcas de agua 35
DATAMATRIX 35
Algoritmia para el cálculo del código DATAMATRIX 36

Prácticas desaconsejadas 36
Supresión de datos 36
No incluir numeración 36
QRCODE 37
Algoritmia para el cálculo del QRCODE 38
Uso de códigos en color QRCODE 38
Prácticas desaconsejadas 38
Truncamiento 38
Supresión de datos 38
No incluir la numeración 38
Uso de logos 39
Uso de marcas de agua 39
La transmutación en profundidad 40
Código EAN8 y EAN13 40
AZTEC, DATAMATRIX y QRCODE 41
La adquisición en profundidad 41
Adquisición óptica 41
Pistola, lector laser, tipos y usos 42
Limitación de la anchura del haz de barrido en… 44
Configuración de lectores 44
Cámaras independientes, cámaras en teléfonos móviles y… 45
Cámaras de vídeo y Web 45
Configuración de cámaras 45
Escáner 45
La lectura de códigos 46
Códigos EAN8 y EAN13 46
AZTEC, DATAMATRIX y QRCODE 47
Adquisición desde aplicativos 47
Tolerancia de lectura 47
Envejecimiento y desgaste de material impreso en EAN8… 47
Envejecimiento y desgaste de material impreso códigos 2D 47
Aberración, ruido, contraste, color y tonalidad en los códigos 2D 47
El lanzamiento dinámico en profundidad 52
Algoritmia 52
Automatismo y bucles 52
Lanzamiento dinámico con intervención del usuario 52
Lanzamiento dinámico transparente en cascada 53
Impresión de códigos 59
Tipos de impresoras adecuadas para los códigos 59
Tipos de soportes para impresión de códigos 59
Resultados 60
Posibles aplicaciones de los códigos 1D y 2D en entomología… 61
Agradecimiento 71
Bibliografía 71
Glosario 73
Anexo I 80
Tablas 80
Tabla I 80

Tabla II 80
Tabla III 81
Tabla IV 82
Tabla V 83
Tabla VI 83
Tabla VII 84
Tabla VIII 85
Tabla IX 86
Tabla X 87
Tabla XI 88
Tabla XII 89
Tabla XIII 90
Tabla XIV 91
Tabla XV 92
Tabla XVI 94
Tabla XVII 95
Anexo II 95
Fórmulas 95
Fórmula I 95
Fórmula II 95
Anexo III 96
Algoritmia 96
Código I 96
Código II 97
Código III 98
Código IV 99
Código V 100
Código VI 101
Código VII 101
Código VIII 102
Código IX 104
Código X 137
Código XI 138
Código XII 138
Anexo IV 140
Listas 140
Lista I 141
Lista II 141
Anexo V 143
Referencias 143
Lectores 143
Fuentes de códigos de barras 143
Aplicativos de adquisición 143
Aplicativos decodificadores 144
Librerías 145
Fotografía, aplicativos, control remoto 146
Índice de la monografía 148
Resumen de códigos 154

Convenciones

A lo largo de la monografía en los encabezados se utiliza un código de color que indica su nivel de anidamiento según el siguiente esquema básico:

1
- **1.1**
- **1.2**
  - **1.2.1**
  - **1.2.2**
    - **1.2.2.1**
    - **1.2.2.2**
      - **1.2.2.2.1**
      - **1.2.2.2.2**

- Las tablas se presentan en colores anaranjados.
- Las fórmulas en coloración verde-azulada.
- La algoritmia y los códigos C# y XML, según la codificación de color y resaltado internacional (SyntaxHighlighting) para C# y XML de Microsoft.
- Las referencias a los controles en los algoritmos se realizan en el sentido clásico añadiendo el prefijo del nombre abreviado permitido, por ejemplo, textBoxNombre = txtNombre; pictureBoxNombre = pctNombre; labelNombre = lblNombre, etc.
- En la algoritmia se utiliza en la asignación de nombres a procedimientos, funciones etc., el sistema de intercalación de mayúsculas y minúsculas, por ejemplo: “esteParrafoExplicaLaNomenclatura”, comenzando en minúsculas, sin espacios ni acentos ni símbolos especiales.
- El código se muestra con las sangrías recomendadas en las normas internacionales de estilo en programación de Microsoft.
- Los retornos de carro en la algoritmia y en las frases que no caben en una sola línea, se incluyen exclusivamente en las sentencias vitales y en las que el compilador pudiera lanzar una excepción. Se ignoran los retornos de carro, por ejemplo, en las frases de texto de usuario. Se sobreentiende que en la interfaz de programación los textos se gestionarán automáticamente con el sistema subyacente de autoedición de código. Debido a que el formato de este documento tiene márgenes, no como algunos editores de código que carecen de ellos, ciertas sentencias de cadena muy larga pueden aparecer separadas por el punto del objeto que encapsula o sobrecarga, esto es inevitable, por lo que habrá de tenerse en cuenta en caso de pegar el código.
- Los vínculos a Internet se muestran con resaltado azul.

Introducción

El momento tecnológico actual exige eficiencia, eficacia y rapidez, como consecuencia la transcripción y filtrado manual de datos es un proceso obsoleto. Como respuesta hacia la máxima eficiencia en lo referente a tiempo, economía y seguridad, a las necesidades industriales y los procesos de comercialización de productos, surgieron hacia la segunda mitad del pasado siglo sistemas y dispositivos de lectura automática. Dichos medios, que partieron de primitivas fichas horadadas, rápidamente evolucionaron hacia la codificación-decodificación de códigos de combinaciones de círculos concéntricos de líneas de diferentes grosores y espacios blancos. Estos gráficos representaban una serie numérica o una información sobre el tipo de producto, fechas, trazabilidad, etc. La implantación del escáner con luz láser como decodificador y los símbolos de líneas con diferentes grosores y espacios, en sustitución de los círculos, consolidó el uso de los populares códigos de barras 1D. La creciente miniaturización de los símbolos y las cada vez mayores necesidades de capacidad de almacenamiento de datos, han provocado la búsqueda de nuevos sistemas. Entre ellos los códigos 2D muy utilizados en la actualidad porque pueden ser decodificados con las cámaras y las App para teléfonos móviles. Los códigos 2D mejoran la decodificación con respeto a los 1D incluyendo datos redundantes de varios niveles a modo de copia de seguridad integrada en el propio símbolo por medio de los algoritmos Reed-Solomon. Se ofrece pues, la capacidad de reparar posibles errores de lectura y consolidar la intervención bajo entornos de pérdida de datos por factores como el deterioro del contingente impreso en las etiquetas. También existen otras tecnologías inalámbricas no ópticas, como por ejemplo, la captación de ondas de radio con lectores, de emisiones provocadas o reflejadas por el chip RFID (siglas de Radio Frequency Identification, conocido con diversos nombres: MIFARE, HITANG, ICODE, UCODE, ATMELRFID, HTKRFID. La diferencia con los códigos de 1D-2D estriba en que estos sistemas posibilitan la programación externa del código, es decir, permiten lectura y escritura de forma combinada y asincrónica. El sistema facilita la captación múltiple de códigos cercanos por discriminación de la señal tomada de los mismos con ecuaciones de árboles de singulación. Poco a poco estos sistemas se van implantando pero todavía en la actualidad resultan caros. En lo que atañe a la biología, precisamente, los chips más pequeños y encapsulados, que son los que pudieran resultar de mayor utilidad, todavía tienen elevados precios. Igualmente, los que permiten la escritura, aún no proporcionan suficiente miniaturización para su uso con artrópodos (fig.1). Los chips insertos en grandes etiquetas con transpondedor y antena son baratos pero su gran tamaño y la distancia necesaria entre los mismos limitan el uso en entomología. Por otra parte, los encapsulados que admiten mayor cantidad de caracteres necesitan una pequeña batería (se conocen como 'activos' o ARAT) que con el tiempo se agota. Los encapsulados sin pila (conocidos como 'pasivos' o PRAT) admiten una muy pequeña cantidad de datos, superados con holgura por algunos códigos de barras y muy ampliamente por los códigos 2D. En lo que concierne a su uso con insectos, la falta de funcionalidad, inconvenientes descritos y precio, provoca que en este artículo no insistamos más sobre la cuestión. Para más información sobre el tema en relación a su utilidad con artrópodos y contenidos relacionados pueden consultarse, por ejemplo, Kutsch *et al*., 1993; Kuwana *et al*., 1999; Bock *et al*., 2004; Wikelski *et al*., 2006; Harrison *et al*., 2010, Sato *et al.*, 2009; Satoi & Peeri *et all.,* 2009 y Sato, & Maharbiz, 2010. Otras tecnologías están emergiendo en la actualidad y se encuentran en fase embrionaria y aunque, *a priori,* pudieran parecer buenos sistemas, su establecimiento y supervivencia dentro de las encarnizadas leyes de la procelosa mercadotecnia está por perfilarse. Es el caso del sistema BOKODE (Mohan *et.al,* 2009) que consiste en micro-etiquetas prácticamente imperceptibles

fundadas en la interacción óptica a distancia de una base emisora de luz que se decodifica por medio de una cámara fotográfica con sistema SLR. También necesitan una pequeña fuente de alimentación. El sistema está siendo muy discutido, es caro y totalmente experimental y quizá sobreviva o se desestime, como ha sucedido con tantas tecnologías (para más información pueden consultarse Zhang & Curless *et al.*, 2002; Zhang & Fronz *et al.*, 2002; Matsushita *et al.,* 2003; Raskar *et al.,* 2004; Fernald, 2006; Tremblay *et al.,* 2007 y Zhang *et al.,* 2008). Hasta que no se implante esta tecnología no nos pronunciaremos aquí, por lo prematuro, sobre sus posibles aplicaciones entomológicas. Por lo tanto, en este trabajo profundizaremos sobre la manipulación de los códigos 1D y 2D; generación, codificación y decodificación, algoritmia, etc. aplicado a la biología y entomología. La toma de los datos puede limitarse exclusivamente a la lectura del contenido del código, por ejemplo, obtener un número con el que realizar una búsqueda sobre una lista, pero es el uso dinámico de la información el que posee mayor potencialidad. La decodificación de una etiqueta entomológica, por ejemplo, adherida en un porta con una preparación microscópica y su tratamiento dinámico en conjunción con la ejecución de la algoritmia de un aplicativo puede provocar funcionalidades expandidas muy diferentes variadas y ricas. La captura de uno o varios caracteres de entrada o finalización transmitidos por el aparato lector, por ejemplo, el código de la tecla (INTRO, TAB, etc.) puede provocar eventos en cascada de alta funcionalidad. Con el escueto gesto de batir el adminículo y provocar la transferencia de un solo código, es viable consumar cuantiosas instrucciones. La adquisición dinámica es por tanto tremendamente útil. En esta monografía se desarrollan estos sistemas, explicando detalladamente las diferentes maneras de proceder con la captura del elemento codificado, su interpretación, la escritura y uso de código, su algoritmia y la interrelación con los aplicativos. Como bibliografía básica para la codificación-decodificación de códigos 1D y 2D pueden consultarse: Peterson & Weldon, 1972; Clark *et al.*, 1981; Blahut, 1983; Constantine, 1987; Chuan. & Khee, 1992; Wicker & Bhargava, 1994; Steen, 1999; Hankerson *et al.* 2000; Hecht, 2001 a y b; Parikh & Janke, 2008; Siomg *et al.*, 2008; Bulan *et al.*, 2009 y Grillo *et al.*, 2010.

Figura 1: insecto preparado en acetato que tiene debajo un chip RFID encapsulado.

Material y método

Con el fin de realizar pruebas con diferentes aparatos y plataformas se han usado para este trabajo cuatro computadores diferentes (con S.O. W-XP, W-VISTA, W-7 y LINUX) dos impresoras, una de calidad fotográfica y otra láser, tres escáner, dos lectores láser de códigos, uno con capacidad para 1D y otro para 2D, dos cámaras fotográficas digitales, una cámara Web, una de Vídeo, dos cámaras CCD y tres teléfonos móviles.

Como aplicativos para el computador se han manejado (independientemente de los utilizados para la confección de esta monografía: editores de texto y de fotografías, programas de dibujo, etc.), MICROSOTF OFFICE (EXCEL, ACCESS y WORD, todos en modo programador), MICROSOFT VISUAL STUDIO, SQL SERVER, SOFTWEDGE, METROSET, GOOGLE EARTH y CRYSTAL REPORT. Aparte, se han construido especialmente para esta monografía varios módulos para ANIMALIA-SISTEMA EPHESIA V.4.0, 64 bits (2014), en concreto: el CODIFICADOR-DECODIFICADOR DE CÓDIGOS 2D (carpetas DATAMATRIX, QRCODE, AZTEC, DECODIFICADOR UNIVERSAL y DECODIFICADOR POR CÁMARA WEB), GENERADOR DE CÓDIGOS EAN, LABORATORIO DE IMÁGENES, EDITOR DE MICROETIQUETAS, MICROETIQUETAS EAN, PREPARACIONES MICROSCÓ-PICAS, GESTOR DE BIBLIOGRAFÍA, GESTOR DE CAJAS ENTOMOLÓGICAS y LUNAS ALGORÍTMICAS. Todos los módulos funcionando bajo el sistema operativo W7 Ultimate, 64 bits en modo no AERO. La captura de pantallas bajo las mismas condiciones y con el programa RECORTES de WINDOWS más PHOTOSHOP.

Los cálculos finales mostrados en esta monografía fueron realizados sobre el computador más potente, con procesador de 2.20 GHz de cuatro núcleos primarios + cuatro núcleos lógicos, 8 GB memoria RAM, 6 TB de memoria virtual, tarjeta video 540 MHz, 4,95 GB + 3 GB de memoria compartida. El tiempo agotado en el empleo de los algoritmos y las generaciones de códigos (sobre un exe ON RELESE) se calculó con una clase específica para EPHESIA diseñada a estos efectos, llamada CRONÓMETRO DE SISTEMA sin interfaz gráfica y ejecutada en un hilo indepen-diente del proceso principal (threads).

Para comprobar la decodificación del lector en los diferentes tipos de soportes físicos de impresión de los códigos se han utilizado cierta variedad de papeles fotográficos y hojas de transparencias adhesivas. Las condiciones de iluminación impuestas para las lecturas de los códigos son tres: luz día dominante roja, focos con leds iluminación fría blanca dominante azul y tubos fluorescentes de emisión fría blanca dominante verde.

Breve historia de los códigos 1D y 2D

1932

En esta fecha, un ambicioso proyecto fue realizado por un grupo de estudiantes en la Universidad de Harvard encabezado por Flint. En el mismo se proponía que los clientes seleccionaran la mercancía desde un catálogo mediante tarjetas perforadas. El sistema transportaba automáticamente el producto desde el almacén a la caja. Nunca fue aplicado.

1948

En este año Silver, Woodland & Ligth, estudiantes en Drexel Institute of Technology en Philadelphia desarrollaron un código. El sistema usaba tinta que brillaba bajo luz ultravioleta. El método tuvo problemas con la inestabilidad de la tinta y su elevado precio. Después idearon un código con círculos que se patentó (1949 -1952).

1961

En este año se fabricó del primer escáner de códigos de barras que leía barras de colores rojo, azul, blanco y negro identificando vagones de ferrocarriles y usando luz de gas de helio-neón.

1970-79

En 1970 aparece el primer terminal portátil. En 1971 en el Reino Unido surge el código PLESSEY para control de archivos en organismos militares. Su aplicación se difundió para la inspección de documentos en bibliotecas. A la par aparece el CODABAR que encuentra su mayor aplicación en los bancos de sangre. En 1972 se crea el ITF. En el año 1973 se anuncia el código UPC (Universal Product Code) que se convertiría en el estándar de identificación de productos. Se dice que aunque varias empresas tenían sistemas de rastreo por UPC para los almacenes, el primer artículo escaneado por UPC para su compra al por menor, fue vendido a las 8:01h de la mañana del 26 de junio de 1974, en una filial de los supermercados Marsh en Troy, Ohio, y era un paquete de 10 chicles de Wrigley's Juicy Fruit. La golosina está en exhibición en el Museo Nacional de Historia Americana del Instituto Smithsoniano en Washington DC, se desconoce si vacía o con el contenido momificado. En Europa se presenta una versión del UPC, el código EAN (European Article Number) que no se consolidará hasta años más tarde con enorme éxito, sobre todo los denominados EAN8 y EAN13. Fue tal el triunfo de esta tecnología, que en principio estaba concebida para una organización solamente europea, que surgió la necesidad de convertirlo en EAN INTERNATIONAL CODE (EANIC, EANUCC, GS1128 y EANEDI). En 1974 apareció el primero de tipo alfanumérico, CODE39 ideado por Allais & Stevens. El primer sistema patentado de verificación de códigos de barras por medio de láser aparece en el mercado en 1978.

1980-89

En 1980 nace el POSNET para el Servicio Postal de los EEUU. La tecnología de CCD (Charge Coupled Device) es aplicada en un escáner en 1981. Actualmente, este tipo de tecnología tiene bastante difusión en el mercado asiático mientras que el láser domina en el mundo occidental. En ese año también aparece el CODE128, de tipo alfanumérico. Asimismo nace la norma ANSI MH10.8M que especifica las características técnicas de los CODE39, CODABAR, e ITF (Interleaved Two of Five). En 1987, Allais desarrolla el primer código bidimensional, el CODE49. En 1988 le sigue Williams con el CODE16K.

1990-99

En 1990 se publica la especificación ANSIX3.182, que regula la calidad de impresión de códigos de barras lineales. En ese mismo año, Symbol Technologies presenta el código bidimensional PDF417. A partir de entonces surgen numerosos tipos de códigos 2D. El QRCODE (Quick Response o respuesta rápida) es un código creado por Damm & Retes en 1994 con el objetivo de que pudiera ser decodificado a alta velocidad. En 1999 emerge el SHOTCODE en la Universidad de Cambridge, mientras se investigaba en un sistema de bajo coste basado en visión artificial para seguimiento de localizaciones y que acabó dando lugar a TRIPCODE y al SPOTCODE. Su uso no ha trascendido mucho y se emplea poco en la actualidad.

Siglo XXI

En 2005 diversas aerolíneas implementan el QRCODE como tarjeta de embarque. En el año 2007 los teléfonos móviles incluyen tecnología que permite leer los QRCODE. Se pueden utilizar incluso para usarlos como entradas virtuales en conciertos, eventos, internet, etc. También aparecen los códigos MICROQR, MCODE, AZTEC, DATAMATRIX, COOLDATAMATRIX, TRILLCODE, QUICKMARK, SHOTCODE y BEETAGG. En el 2008 en España, se creó el código BIDI de uso privado para Telefónica. En 2009, Microsoft creó los códigos en color de alta capacidad MICROSOFT-TAG, cuya distribución comenzó siendo gratuita para luego ceder su comercialización con el nombre de HCCB a otra marca propietaria. Lo que pareció en un principio la evolución natural de los códigos en blanco y negro no parece tener en la actualidad demasiado calado.

Tipos de códigos

Este apartado no pretende ser una guía exhaustiva de los códigos existentes. Se describen minuciosamente las características de aquellos cuyos usos creemos que pudieran ser más ventajosos en biología y/o entomología. Al contrario, se incluyen descripciones muy someras para los que encontramos menor provecho, sin prejuicio de que el avezado lector pueda encontrar en ellos utilidades en la que nuestra imaginación no nos ha permitido indagar. A continuación detallaremos los códigos más corrientes ordenados alfabéticamente.

3DI

Figura 2: es un código propietario que utiliza pequeños símbolos circulares. Se ha utilizado para la codificación de instrumentos quirúrgicos. Lamentablemente en la actualidad en desuso por temas de patentes. Podría ser un símbolo muy útil en entomología, para etiquetar tapas de frascos, dado que es de los pocos circulares de pequeñas dimensiones.

ARRAYTAG

Figura 3: el símbolo se compone de elementos hexagonales que se imprimen ya sea solos o en grupos secuenciados. El anidamiento de los hexágonos crece de manera proporcional a la cantidad de datos incluidos. Es un formato propietario muy poco utilizado.

AZTEC

Figura 4: este tipo de código puede contener, de 1 a 3748 dígitos, y 1-2999 caracteres (anexo I, tabla I). Es cuadrangular y como otros códigos 2D el número de caracteres depende del tamaño. Los valores 0 a 127 se interpretan como en el conjunto de caracteres ASCII, mientras que los valores de 128 – 255 se descifran como la norma ISO 8859-1, alfabeto latino N ° 1. Es libre y de dominio público. Para usos entomológicos es similar a los códigos DATAMATRIX, QRCODE o MICROQR. De los tres, es el que con el menor tamaño puede contener mayor cantidad de información y se lee mejor. Es posible generar códigos de 10 mm^2 con un nivel de errores nulo o muy pequeño en su decodificación. Nosotros lo utilizamos para la sistematización de tapas de pequeños frascos de preparaciones, por ejemplo, vesicas evaginadas con la técnica del silopreno o material guardado en alcohol.

BEETAGG

Figura 5: un código 2D denominado de “Etiqueta de abeja” poco utilizado pero de todos los existentes es el que soporta una mayor cantidad de pérdida de información. Los datos pueden ser encriptados dentro del propio símbolo, es decir, es factible no almacenar las cadenas originales sino encriptarlas y desencriptarlas en el momento de la generación de la codificación y decodificación. Es uno de los códigos más fiables y seguros pero lamentablemente no está estandarizado, siendo en la actualidad pequeño su impacto en el mercado. Admite logos.

BIDI

Figura 6: parecido al QRCODE, pero utilizado por telefónica. Privado y sujeto a patente.

BLOTCODE

Figura 7: es un código propietario de muy poco uso. Diseñado para teléfonos.

CODABAR

Figura 8: puede contener números y símbolos especiales (-$:/.+). Requiere caracteres para inicio (A, B, C, y D) y final de secuencia (T, N, *, E). Es útil para cadenas matemáticas.

CODACBLOCK

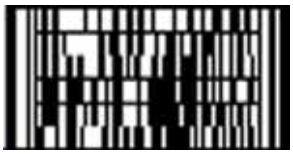


Figura 9: parecido al CODE49.

CODE11

Figura 10: es un código numérico con un carácter especial.

CODE16K

Figura 11: código bidimensional que admite una información máxima de 16K.

CODE39

Figura 12: es de ancho variable y puede tolerar cualquier número de caracteres alfanuméricos. Es popular debido a que puede contener texto y números (A ► Z, 0 ► 9, +, -), puede ser decodificado por casi cualquier lector al ser uno de los más antiguos.

CODE49

Figura 13: como el CODABLOCK pero de menor anchura. Expresamente creado para las aplicaciones médicas.

CODE93

Figura 14: es continuo. Usa código ASCII con los caracteres construidos por tres barras y tres espacios.

CODE128

Figura 15: cuando la longitud de la etiqueta es un punto importante, el CODE128 constituye una buena alternativa ya que es muy compacto y genera un símbolo denso. Para etiquetas cortas es demasiado largo.

CODEONE

Figura 16: el primer código 2D, desbancado por DATAMATRIX.

COLORCODE

Figura 17: permite un reconocimiento de códigos indexados que son a su vez vinculados a los datos externos. La matriz de bloques y los datos analógicos correspondientes al número de colores son digitalizados y luego procesados por un servidor dedicado usando direcciones registradas en los códigos.

CPCODE

Figura 18: similar en el aspecto a DATAMATRIX pero de uso privado.

COOL–DATAMATRIX

Figura 19: parecido a DATAMATRIX pero poco utilizado.

DATAGLYPHS

Figura 20: es un código propietario. Se compone de un patrón de fondo gris y datos binarios de codificación, incluyendo los patrones de sincronización y corrección de errores. Limitado uso.

DATAMATRIX

Figura 21: es una simbología 2D bidimensional de matriz con módulos acomodados en un patrón cuadrado o rectangular. Posee patrones de redundancia para la recuperación de errores. En función al nivel puede almacenar mayor o menor número de caracteres. Al mismo tiempo, la longitud de la información codificada depende de la dimensión del símbolo utilizado. Para labores entomológicas puede ser útil cuando es necesario un código de pequeño tamaño y cuadrado. Léanse los comentarios en el apartado correspondiente sobre la utilidad entomológica del MICROQR puesto que son igualmente aplicables al código DATAMATRIX, pero en comparación con el código MICROQR, en el mismo tamaño y nivel de recuperación, DATAMATRIX puede contener mayor cantidad de información (anexo I, tablas, II, III, IV y V).

DATASTRIP

Figura 22: es el más viejo de los símbolos de dos dimensiones. Es un código propietario. Consiste en un patrón de matriz. Los marcadores de la parte inferior del lado y a través de la parte superior de la tira contienen la información de la alineación para los lectores de código y aseguran integridad de datos. Es necesario un lector especial.

DOTCODE

Figura 23: consiste en un arsenal de puntos en una matriz cuadrangular que se extienden sobre un tablero 6 x 6, a 12 x 12, que va creciendo hasta codificar más de 1 billón de dígitos. Se ha utilizado para identificación de la cristalería de laboratorio. Es propiedad de Philips.

EAN8

Figura 24: puede contener 8 dígitos. Es uno de los códigos más universales. Es escalable y obliga a mantener su proporción y se puede imprimir muy pequeño. Es decodificado por todos los lectores. Necesita patrón de parada (00000000>) pero para los lectores de hoy en día no son ineludibles, pudiendo decodificar las barras sin el más mínimo problema en ausencia de este símbolo. En nuestra opinión, es el más adecuado para micro-etiquetas entomológicas, en el caso de requerir una serie de números no superior al billón. Existen versiones aún más pequeñas: EAN2 y EAN5 que admiten los dígitos que su número indica (anexo I, tabla VI-VII).

EAN13

Figura 25: formado por 13 dígitos. Su tamaño puede ser relativamente pequeño. Es decodificado por todos los lectores. Es escalable y obliga a mantener su proporción. Es uno de los códigos más universales. Necesita patrón de parada (00000000>), pero los lectores actuales lo decodifican sin problema, en ausencia de éste. Por su compatibilidad consideramos que es útil en entomología cuando no se necesitan más de 13 guarismos (anexo I, tabla VI-VII).

EAN128

Figura 26: como los anteriores pero de mayor tamaño. Derivado del UPC128.

GS1128

Figura 27: similar al EAN128, pero tienen una gran variedad de tamaños enumerados que, por lo general, se especifican por medio del ancho y no por los valores de magnificación.

HCCB

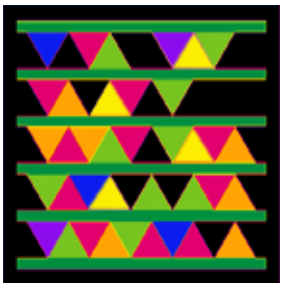


Figura 28: se trata de un código con muy alta capacidad que se logra mediante la combinación de colores. Hubiera sido una buena opción para entomología dado que se pueden generar códigos de 6 x 6 mm pero hay pocos aplicativos compatibles. Antes lo ofrecía Microsoft de manera gratuita, ahora es un código propietario, tanto para codificar como para decodificar y además caro.

INTERLEAVED

Figura 29: también conocido como 2-de-5, ITF y I-2/5, entrelazado. Necesita patrones de inicio y parada.

ISBN

Figura 30: "International Standard Book Number" (Número Internacional Normalizador de Libros), es un sistema para numerar de manera única los títulos de la producción editorial de cada país o región con el objeto de identificar al editor, título y autor, sin posibilidad de repetir el número asignado. Su decodificación resulta útil en entomología si se desean introducir las reseñas de los libros científicos de la biblioteca, en una tabla de datos principal relacionada con otras de editores, títulos, etc.

ISSN

Figura 31: "International Standard Serial Number" (Número Internacional Normalizador para Publicaciones Seriadas) es un código único sin ambigüedad que permite la identificación de cualquier publicación vigente y/o que dejó de comercializarse sin importar su lugar de origen, idioma o contenido. Al igual que el ISBN, resulta útil en entomología si se desean introducir las reseñas de los libros científicos en una tabla de datos principal relacionada con otras de editores, títulos, etc.

ITF

Figura 32: es conocido también como "2 de 5". Es exclusivamente numérico, su figura es ligeramente más larga que el código barras UPCA, admite 10 dígitos pero tienen que ser pares. Si el número es impar se colocan ceros al principio hasta completar la cadena.

ITF14

Figura 33: como el GS1128, tiene gran una variedad de tamaños especificados que, por lo general, se diferencian por medio del ancho y no por los valores de magnificación.

MAXICODE

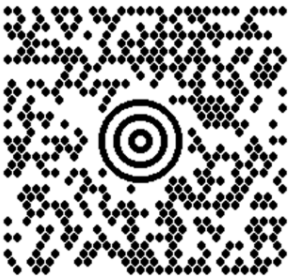


Figura 34: este símbolo cuadrado que consiste en series de código hexagonal alrededor de un círculo prediseñado, no se utiliza mucho.

MCODE

Figura 35: es un código 2D poco utilizado.

MICROQR

Figura 36: es una versión más pequeña del estándar del QRCODE. Existen 4 versiones de código MICROQR. La más grande puede contener hasta 35 caracteres y la más pequeña 4. La utilidad del MICROQR consiste en la capacidad de almacenar mayores cantidades de cadenas pero al mismo tiempo su tamaño va creciendo. Su uso entomológico depende de los contenidos, nivel de corrección de error, la miniaturización necesaria del símbolo, cantidad y tipo de caracteres Se puede generar símbolos de hasta 10 mm^2 pero su decodificación es comprometida. Si el usuario pretende desencriptar exclusivamente pequeñas cadenas de números (por ejemplo para filtrar un ID en una tabla de datos entomológica) y generar un código de tamaño minúsculo, su versatilidad puede ser igual o inferior al código EAN8. Para este uso el MICROQR es menos universal y más lento en su decodificación y es preferible decantarse por el EAN8. Puede ser útil en otras labores entomológicas como por ejemplo para pegar en una tapa de diámetro reducido de un frasco, para codificar tubos de ensayo, identificar series de procesos químicos, etc. o cuando sea estrictamente necesario un código de pequeño tamaño y cuadrado

MINICODE

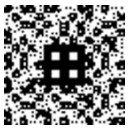


Figura 37: es propietario. Se compone de los símbolos de matriz cuadrada. Es uno de los más pequeños. Poco extendido.

PDF417

Figura 38: es bidimensional. Se trata de una simbología de alta densidad, no lineal. La diferencia con otros tipos de códigos de barras mencionados anteriormente reside en que el tipo PDF417 es un archivo de datos portátil (PDF). Utilizado en medicina.

POSNETCODE

Figura 39: "Postal Numeric Encoding Technique" es utilizado por el servicio postal de los EEUU.

QRCODE

Figura 40: Quick Response Barcode es un método 2D que nos permite almacenar información en una matriz de puntos. Tiene tres cuadrados que se encuentran en las esquinas y que permiten la detección del código cuadrangular y escalable por el lector. Un pequeño cuadrado que se van multiplicando a medida que el símbolo contiene mayor cantidad de datos. Logran leerse prácticamente con cualquier aparato, teléfonos, tabletas, escáner, cámaras, etc. Desde el computador se puede decodificar con multitud de programas. Su código es abierto y su patente libre. En la actualidad tiene mucho auge y gran expansión, está totalmente consolidado en Asia. El tamaño depende de la cantidad de información y la cantidad de caracteres del sistema de restauración (codificación leve, mediana o alta). A mayor redundancia, menor número de caracteres y/o mayor tamaño, a menor cantidad de datos duplicados al contrario. Requiere un gráfico relativamente grande. Por debajo de los 20 mm^2 es difícil de leer y su decodificación queda sometida a la calidad de impresión y a la resolución del escáner o cámara. Existen en la actualidad 40 versiones, a mayor número de versión mayor es el símbolo y cantidad de datos codificables. Por su universalidad en lo referente a la lectura por multitud de aparatos, puede ser útil para labores entomológicas, para químicos, cajas, exposiciones, museística, etc. siempre y cuando sea necesario codificar una larga cadena de texto y el tamaño no sea crucial (anexo I, tablas VIII, IX, X, XI, XII y XIII).

QUICKMARK

Figura 41: incorrectamente es el nombre que se aplica en ocasiones al QRCODE. El tema se complica al existir en el mercado un aplicativo codificador y decodificador muy conocido con el mismo nombre.

SHOTCODE

Figura 42: este es un tipo circular de código muy particular. En realidad no almacena la información en el símbolo sino que encripta una clave de 49 bits y una dirección URL. El dispositivo, tras decodificar el código, se conectará al servidor, obtendrá la URL en cuestión y a continuación leerá de allí los datos relacionados con el SHOTCODE. El servidor puede establecerse en nuestro propio computador al igual que la URL en un disco duro. Un programa diseñado por nosotros o un archivo BAT puede abrir la información almacenada en una tabla de datos a partir de la decodificación. Por lo tanto, es un sistema dinámico de adquisición de datos externos a la propia etiqueta. La información resultante sólo estaría limitada por la memoria del sistema, por ejemplo, con un solo código podría accederse a una biblioteca entera, a una fonoteca, a un álbum de fotografías o a cualquier archivo multimedia. En realidad esto mismo se podría realizar con el universal y viejo código EAN8 o incluso con códigos más simples con mayor sencillez, rapidez y estandarización. Es una pena porque un símbolo en forma de círculo resultaría útil en entomología. Es irónico que el primer código de barras que se inventó fuera con circunferencias y en la actualidad no exista ningún símbolo con ese formato que almacene la información en su propio cuerpo.

SNOWFLAKECODE

Figura 43: código propietario. Consiste en una matriz de puntos que puede contener más de 100 dígitos en un espacio de sólo 5 x 5 mm. La redundancia permite las recuperación de hasta el 40% del código dañado. Se utiliza en la industria farmacéutica.

TRILLCODE

Figura 44: este es un código 2D muy poco utilizado, diseñado para teléfonos móviles. Permite la incrustación de logos y otras imágenes en su base.

UCC/EAN128

Figura 45: tienen una longitud fija. Permiten incluir los 12 dígitos en un espacio razonablemente compacto.

UPCA

Figura 46: es el sistema de códigos de barras 1D más utilizado en Norteamérica desde el año de 1972. Está formado por 12 dígitos. Es similar al EAN13 pero con una menor distribución universal. El primero adquiriere características cosmopolitas en su distribución. Tiende a sustituirse por EAN13.

UPCE

Figura 47: utilizado en artículos muy pequeños. Formado por 8 dígitos. Se trata de un código UPCA reducido por medio de un sistema llamado "supresión de ceros". Para micro-etiquetas y entomología pudiera ser muy útil por su tamaño. La diferencia entre EAN8 y UPCE es que el primero obliga a separar la cadena numérica mientras el segundo puede mantenerse en una única cifra. El único inconveniente del UPCE reside en que no es tan universal y con el tiempo ha tendido a sustituirse por EAN8, estando menos difundido en la actualidad dado que es un código muy antiguo.

ULTRACODE

Figura 48: es de dominio público. El símbolo se compone de una tira de longitud variable de columnas de píxeles. La simbología utiliza pares de columnas verticales de cualquiera de las siete células blanco y negro y 8 multicolores, normalmente blanco, rojo, verde y azul o cian, magenta, amarillo y negro. Es un híbrido entre 1D y 2D, aunque utiliza un sistema de corrección redundante con muchos niveles basado en el algoritmo de Reed Solomon. Se trata de una nueva generación de códigos de alta capacidad. No están teniendo el éxito esperado por lo que están poco extendidos.

Introducción al uso de los códigos 1D y 2D en la Entomología

El lector tendrá que decidir en base a las características aportadas, cuál de los símbolos considera más adecuado a sus necesidades entomológicas y el desembolso que desea efectuar en caso de trabajar con un código propietario. Como ya se ha perfilado sucintamente en el anterior apartado y a modo de resumen, los códigos en los que nosotros encontramos mayores utilidades en entomología son los siguientes: 1D (EAN13, EAN8, UPCA y UPCE) y 2D (AZTEC, DATAMATRIX, MICROQR y QRCODE). Todos los enumerados son gratuitos y de libre acceso. Con respecto a los 1D EAN y UPC, en principio fueron ideados para Europa y Estados Unidos, respectivamente. En este trabajo vamos a centrarnos en los dos tipos de EAN porque su universalidad hoy en día es absoluta. Todos los ejemplos, algoritmos, códigos y comentarios que desarrollemos a partir de aquí pueden ser aplicables también a UPC. Igualmente, con respeto a los códigos 2D. Más adelante se desarrollaran los sistemas, algoritmia, etc., para su codificación-decodificación.

Introducción a los rudimentos para trabajar con códigos

Se pueden distinguir cuatro fases con dos bifurcaciones para el trabajo con códigos 1D y 2D, desde la codificación hasta su uso final (fig. 49).

Figura 49: esquema básico para el trabajo con códigos.

Codificación

Es la parte en la que el gráfico o la fuente del código se generan a partir de la información introducida en un cuadro de texto, de imagen, etc. Se calcula el número de control (Check-Sum) que necesitan algunos códigos. Se establece su tamaño, la paridad, los niveles de redundancia, la versión, los márgenes, el tamaño del píxel, el logo, etc. Esta información se transporta por medio de procedimientos impulsados por rutinas que contienen algoritmos especiales para la generación de los distintos tipos. El resultado se almacena en una variable interna a la espera de la pauta de conversión necesaria.

Transmutación

El código ya ha sido generado, pero el procedimiento de su conformación y su destino puede haber sido muy diferente. Por ejemplo, se puede haber generado con un aplicativo en la Web y almacenado como PDF, o es posible que se generase en el computador personal y se guardase como fuente codificada (TTF, TT, OT, etc.), o quizá se generó con el App de un teléfono móvil y se almacenó como una imagen bitmap. También se puede haber generado en el PC y pintado en la pantalla del monitor. Igualmente, el resultado final puede haber sido la impresión física en muy diferentes soportes, hojas de papel fotográfico, micro-etiquetas, transparencias, polivinilo, etc. o en un archivo binario. El proceso se resume en el siguiente diagrama radial simplificado (fig. 50).

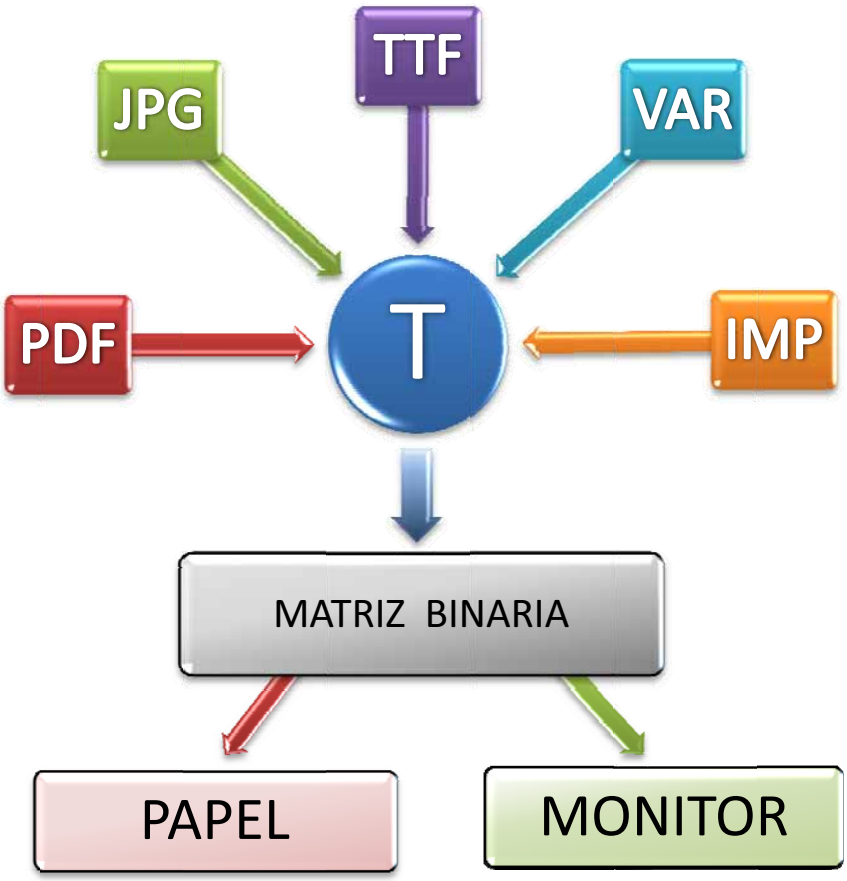


Figura 50: diagrama del proceso de transmutación.

Es en esta fase donde el código se normaliza para prepararlo en caso de una eventual decodificación. Dicha normalización afecta al formato, resolución, opciones establecidas por el operario, etc., con el fin de que la información final se plasme en un soporte virtual o físico adecuado, por ejemplo, en una micro-etiqueta entomológica. En ciertos casos, la lectura del código lo único que contiene es una ruta de sistema donde está ubicada la imagen, incluso una dirección Web donde está alojada la matriz comprimida y para mayor seguridad encriptada. De donde proceda la información es indiferente. El aplicativo decodificador obliga a realizar las conversiones pertinentes porque sólo entiende lenguaje binario, es decir, obligatoriamente los datos de la imagen tienen que ser almacenados en una matriz de puntos sobre la que por recursividad, convolución y bucles anidados pueda ser descifrada (consúltese Niblack, 1986; Magro, 2008: 529 y 541-42 y Magro, 2013: 444).

Adquisición

Ya tenemos el código en el soporte virtual o físico, por ejemplo estampado en una etiqueta o almacenado en una variable de memoria en el computador (con una interfaz gráfica que muestra el gráfico en el monitor). En este paso se decodifica el símbolo impreso por medio de un aparato lector, escáner, teléfono, cámara fotográfica, cámara Web, etc. Si el soporte es virtual, se activa el aplicativo lector de este gráfico en el monitor para mandar desencriptar el contenido del código.

Decodificación

Es la fase en la que el código, que ha sido trasmitido por impulsos eléctricos binarios producidos por el láser, o que ha sido enviado como información binaria en el barrido de la matriz de la imagen, es decodificado por medio de algoritmia. Ésta es muy variada y diferente para cada tipo de código a descifrar y es transparente para el usuario. Es el controlador (driver) del periférico, o en su defecto en los códigos lineales, el del teclado, el que realiza estas operaciones mostrando la información contenida de tipo numérico, alfanumérico, cadenas de texto, imágenes, etc.

Lanzamiento pasivo

El código ya ha sido descifrado con éxito y se ha convertido en una cadena de texto legible para el humano. Se muestra en un cuadro de texto o en cualquier otro formato, acompañado con la trasmisión de la señal de un símbolo invisible (normalmente, ENTER, RETURN, TAB, pero puede ser cualquiera), una o varias veces. Aquí finaliza el proceso, el usuario lee la información y se da por enterado o manipula los datos adquiridos. Por ejemplo, de forma manual: toma el número generado para buscar en una tabla de datos.

Lanzamiento dinámico

El símbolo se ha descifrado y puede mostrase o no al usuario. Lo más corriente es que la información contenida en el código sea transparente y se almacene en una variable interna en el sistema, un área de memoria, un puntero, un manejador de ventana, etc. Puede suceder que ocurran dos cosas a la vez; una, que parte de la información se almacene como hemos explicado; otra que se discrimine y se muestre

en un control para su lectura por el consumidor. En todo caso, lo importante es lo que ocurre con los impulsos invisibles que han sido enviados por el decodificador. Supongamos que tenemos un código de barras impreso y que sus líneas contienen el número 1. Acercamos el lector y se decodifica el símbolo mostrándolo en un cuadro de texto: efectivamente se podría leer 1. Cuando se ha completado la lectura se manda un impulso de INTRO. Aparentemente no ha sucedido nada, pero lo que ha ocurrido es transcendente: el cursor del ratón ha desaparecido del cuadro de texto y se ha colocado en el siguiente del foco de tabulación. ¿Qué ocurriría si en el control se captura el suceso de escritura del primer carácter? ¿qué ocurriría si a continuación se intercepta el evento pulsación de la tecla INTRO? ¿qué acaecería tras el tratamiento de la de pérdida de foco? Imaginemos que en el primero suceso hemos programado que se escriba la cadena +="23", en el segundo, el texto +="456" y en el tercero +="789". El decodificador lee 1 y desde el cuadro de texto que estaba vacío, saltan los eventos en cascada uno detrás de otro: ● ¡trasmiten una cadena con el número 1! ► ¡han escrito el primer carácter, añado a lo que tengo "23"! ► ¡recibo una pulsación INTRO, completo lo que contengo con "456"! ► ¡estoy perdiendo el foco, termino la cadena con "789"! ■ El resultado escrito en el cuadro de texto, como es evidente, sería: "123456789", sin embargo, lo único que había en el código original era un número 1. Esta manera de proceder se conoce como control de eventos en cascada (anexo III, código I). Si en lugar de programar cadenas de texto creamos un algoritmo del tipo "FORMAT C:/", cuando el cuadro de texto reciba 1 + INTRO habremos acabado con nuestro disco duro con una simple lectura de código de barras. Qué sucedería si escribiéramos lo siguiente (en pseudocódigo): "mientras i > 0; codigoTrasmitido = codigoTrasmitido * codigoTrasmitido; i = codigoTrasmitido++". El número 1 comunicado por el lector se multiplicaría y sumaría de manera infinita colocando al procesador en la tesitura de un bucle perpetuo (1|2|5|26|677| 458330…∞). El evento finalizaría con un error de memoria, bien por haber sobrepasado el número de caracteres en el control (32767, por defecto), bien porque el sistema se ha colapsado por falta de memoria. Así pues, a la par que peligrosa, el uso de la decodificación en cascada con bucles y recursividad proporciona grandes funcionalidades. Valgan estas líneas como preámbulo a la potencialidad del lanzamiento dinámico. La algoritmia, el método y los procedimientos necesarios para realizar estas operaciones, se expondrán en profundidad más adelante en el apartado correspondiente.

Doble decodificación

En el lanzamiento dinámico se puede utilizar una etiqueta donde existan dos o más códigos que llaman a tantos hilos de proceso como símbolos existan (fig. 51).

Figura 51: lepidóptero y etiqueta con doble código y lanzamiento secuencial en dos hilos con procesos diferentes.

Rara vez es necesario acometer lectura múltiple en diferentes hilos pudiéndose utilizar el lanzamiento dinámico simple. En alguna ocasión se podría usar para acelerar procesos con respecto al acceso a tablas de datos cuyos ID no están relacionados. Quizá para realizar varias operaciones repetitivas sin preocuparse de programar un lanzamiento dinámico. En todo caso, nótese que en la figura los códigos están truncados (léanse los comentarios más abajo escritos sobre los inconvenientes que pueden aparecer con el uso de estos cercenamientos).

La codificación en profundidad

Código EAN8

Este código admite 8 números. El último se reserva para un dígito de control "CheckSum" (fig. 52, flecha verde). Tiene importancia a efectos de evitar errores de lectura. El resto de la serie hacia la izquierda puede codificarse como guste si bien es conveniente saber que los dos o tres primeros (fig. 52, flecha negra), cuando se trata de un artículo comercial, se utilizan para la codificación del país. En entomología puede rellenarse la serie del principio como parte integrante del ID numérico o también como código de codificación del orden, por ejemplo en Animalia (véase anexo IV, lista I y fig. 187). En el primer supuesto, el mayor número que se podría utilizar sería 9.999.999. En el segundo, con una codificación de entrada de un dígito: 9.99.999; de dos: 99.999 (fig. 52, flecha roja en EAN8) y de tres: 9.999. Se puede usar la cifra completa, en cuyo caso el mayor número sería 99.999.999. La parte señalada con la flecha roja es la que aumenta o disminuye dependiendo del tipo de EAN. Antiguamente estos códigos de barras llevaban un símbolo de finalización (>) para establecer los espacios en blanco, de manera que la cadena resultante se constituiría de la siguiente manera: 99216467>. Esta forma de codificar se utilizaba para facilitar el barrido por los lectores. Hoy en día no es necesario.

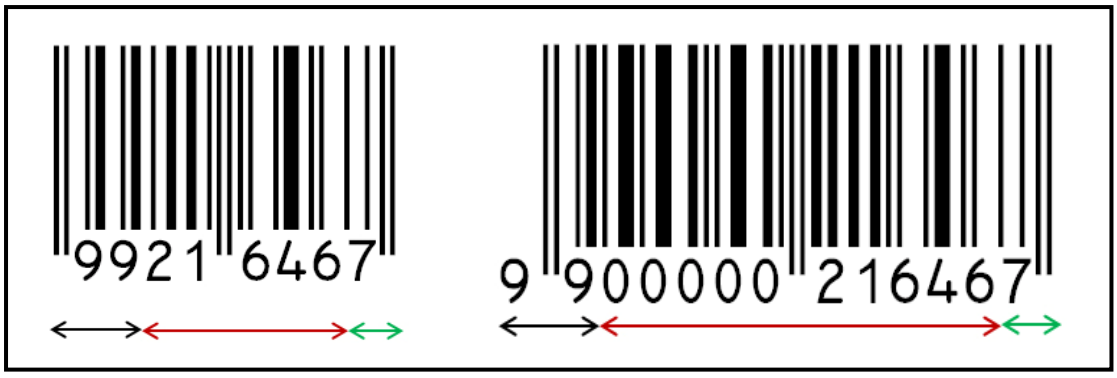


Figura 52: esquema simplificado de las partes de los símbolos EAN.

Método para calcular el dígito verificador EAN8

El cálculo del dígito de control es sencillo y además es fácil encontrar macros para WORD, ACCESS y fórmulas para EXCEL en la Web. Con los números de la etiqueta de la fig. 52, 9921646**?>**, se toman los dígitos de la columnas impares y se multiplican por tres. Los de las columnas pares por uno. El resultado de cada columna se suma. De la decena inmediatamente superior se resta éste (anexo II, fórmula I y II).

Algoritmia para el cálculo del dígito verificador EAN8

Los cálculos necesarios son bastante sencillos. Se puede trabajar con una matriz y realizar los cálculos a partir de las posiciones de los índices o es posible manipular bucles (anexo II, fórmula II; anexo III, código II).

¿Utilizar fuentes EAN o crear gráficos vectoriales?

Para la reproducción de las barras de los códigos se pueden utilizar fuentes de letra o generar las líneas por medio de algoritmos. Cualquiera de los dos sistemas es válido. Sin embargo, el último brinda total independencia de uso al entomólogo dado que muchas fuentes (TTF, fig. 53) no son gratuitas y están sujetas a patente. En un primer momento se había especificado que para los símbolos EAN/UPC/GTIN debía utilizarse la letra OCR-B pero las especificaciones del sistema GS1 ahora permiten utilizar cualquier tipo de letra, siempre y cuando sea claramente legible. Otro tema es que encontremos fuentes libres en la Web, haberlas las hay, pero su uso estará supeditado a las necesidades de calidad y otras características imprescindibles para el usuario, por ejemplo, que sea plenamente funcional. Para entomología conviene una fuente que sea escalable a tamaños muy pequeños.

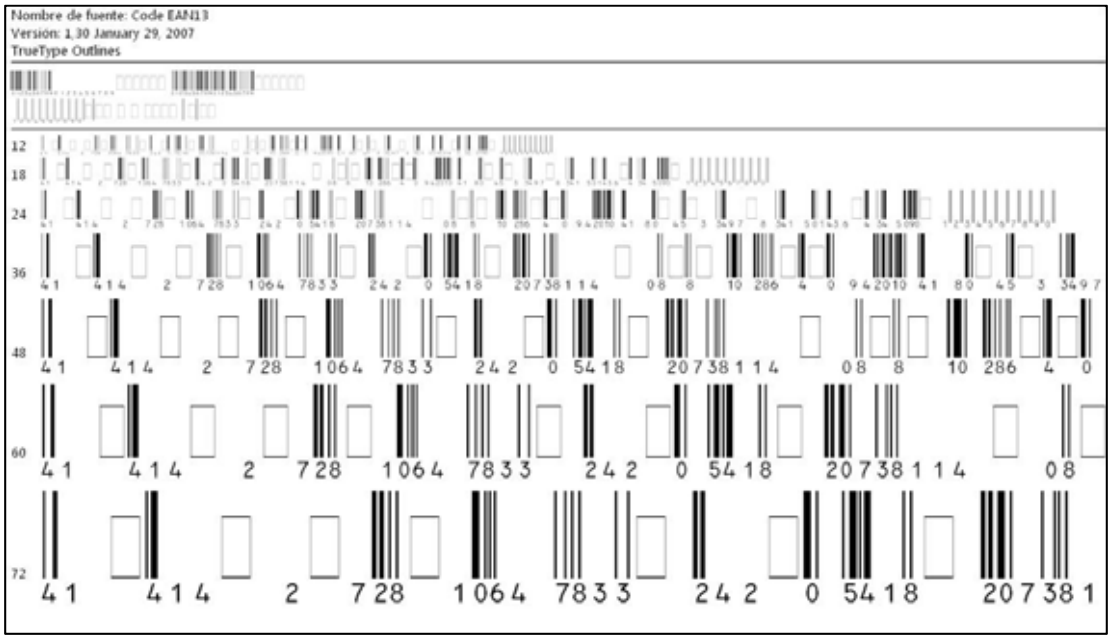
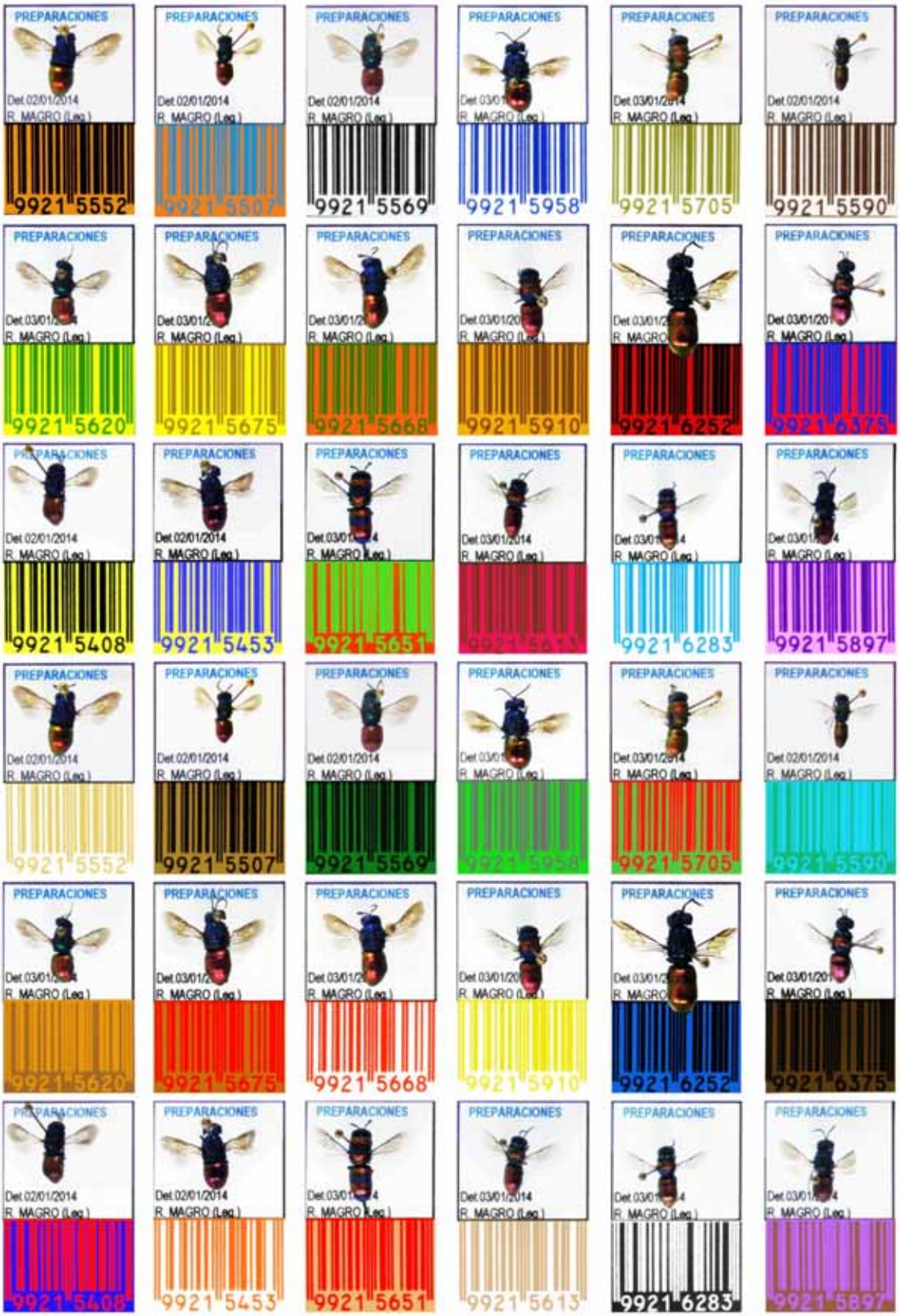


Figura 53: tipos de la fuente escalable TrueType, EAN.

Uso de códigos en color

Los gráficos de barras pueden ser en color pero hay que tener en cuenta que no todos los colores de la fuente (ForeColor) frente a los colores del fondo (BackColor) son decodificables por los lectores (figs. 54- 90; anexo I, tabla XIV). La conformación más legible es, por supuesto, las líneas en negro puro y el fondo en blanco. En entomología la impresión en color debería de utilizarse con recato, por ejemplo, se podrían imprimir los códigos en rojo para los tipos, o en otros colores para micro-etiquetas provisionales.



Figuras 54-90: en las tres primeras filas, figuras 54 al 71 se muestran combinaciones de color que pueden ser leídas por un lector. En las tres últimas filas, figuras 72 a 90 las combinaciones de color no legibles.

Posicionamiento

El código debe ubicarse de modo que las barras sigan el sentido de la impresión, para reducir las distorsiones inherentes a las reproducciones gráficas. Las etiquetas no deben tener aplastamientos, distorsiones, dobleces o cortes. Se debe posibilitar el acceso paralelo del haz de lectura en la medida de lo posible. Las impresiones se tendrán que realizar sobre papeles adecuados y lisos, nunca rugosos. En el caso de colocar las micro-etiquetas en un ejemplar, si éste es de gran envergadura es necesario ubicar la que contiene el código en el lugar más alejado, es decir, al fondo o la última en el caso de haber varias etiquetas. De esta manera, aunque parte de la anatomía del espécimen tape parcialmente las barras, el lector se podrá inclinar levemente y a la vez tendrá suficiente profundidad de campo como para maniobrar y decodificar los códigos sin problemas (figs. 91 y 92).

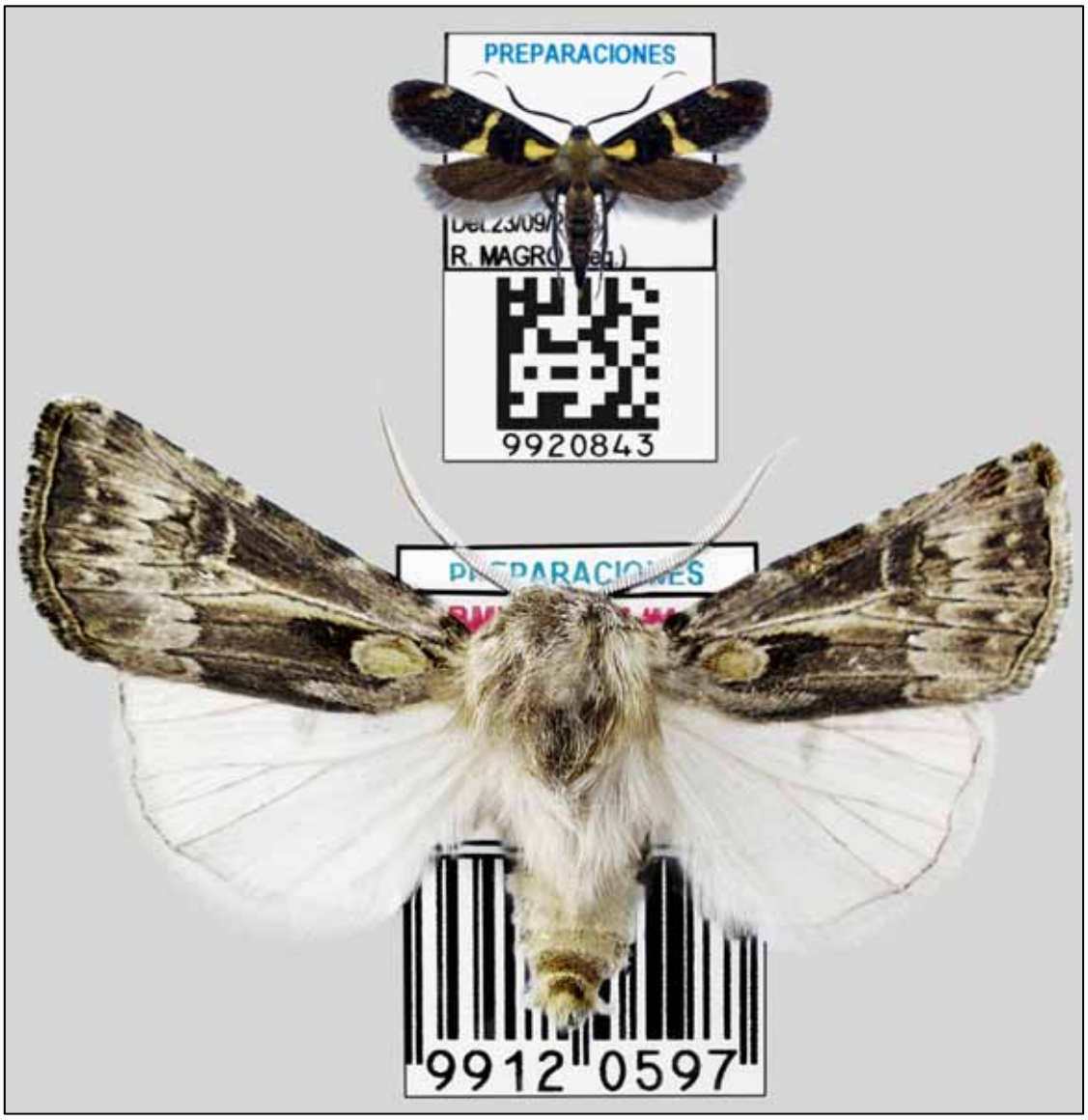


Figura 91-92: en el ejemplar de la parte superior la etiqueta está colocada por encima de las otras porque se trata de un lepidóptero muy pequeño. El abdomen de la falena de la parte inferior tapa parcialmente el código de barras, por lo que se ha colocado en último lugar, facilitando así la lectura con profundidad de campo.

Prácticas desaconsejadas

Truncamiento

Con el fin de reducir el espacio del código en un diseño, es posible especificar la reducción de la altura del símbolo. Este proceso, denominado truncamiento, no está permitido dentro de las especificaciones de la simbología EAN/UPC y debería evitarse debido al impacto negativo que tiene sobre la velocidad de lectura de los escáneres omnidireccionales. Para labores entomológicas, donde no es necesario decodificar códigos a la velocidad del paso por caja de un artículo, se pueden trucar levemente, pero en todo caso no se aconseja reducir la altura más de las tres cuartas partes, (fig. 93).

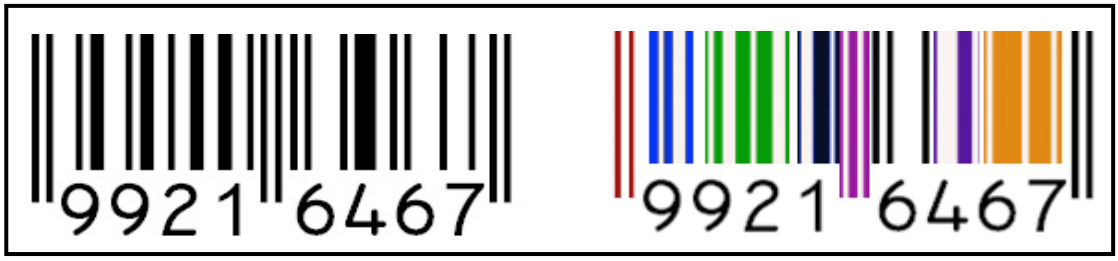


Figura 93: EAN8 en blanco y negro y color truncados hasta el límite recomendable.

Supresión de datos

El código se utiliza para tener mayor agilidad en la aplicación de filtros y operaciones de lanzamiento dinámico. También posibilita la lectura de grandes textos alojados en una base de datos que por su cuantía jamás cabrían en una etiqueta. Nunca se debe utilizar para sustituir los datos de una etiqueta entomológica, para no mostrarlos o suprimirlos. Junto a la etiqueta del código se deberán colocar en el orden adecuado otras etiquetas con la información pertinente: fecha, localidad, altitud, temperatura, humedad, hora, colector, leg., det., nombre de la especie, descriptor, año, etc. Tampoco es recomendable pseudo-encriptar los números con series de sustitución como por ejemplo para 99999, escribir 5-9 (cinco nueves). Hay que cerciorarse de que en el futuro otro entomólogo pueda acceder y decodificar claramente la información. El uso de claves, abreviaturas oscuras y códigos personales puede obstruir el acceso a una información muy valiosa. Un espécimen con datos ininteligibles carece de valor científico, convirtiéndose en un fútil ente decorativo.

No incluir numeración

La única manera de recuperar la información en un código de barras EAN, si la decodificación no es posible, es con la lectura de la secuencia de números, por lo tanto esta práctica se debería desechar. En todo caso no es necesario imprimir la numeración debajo de las barras. Perfectamente se pueden incluir las líneas sin numeración con el único objetivo de no tener que teclear la serie numérica o para el procesado dinámico y escribir los dígitos en otra parte de la micro-etiqueta entomológica, (fig. 94).

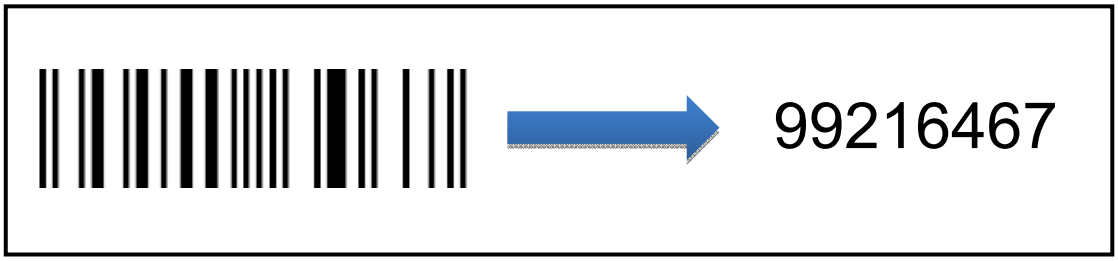


Figura 94: código EAN8, sin numeración. En el caso del deterioro de la etiqueta y la imposibilidad de su lectura, no se podrá recuperar la información porque este tipo de códigos de barras no almacena información redundante. Si no se desea colocar el número debajo resulta imprescindible incluirlo en otro lugar de la etiqueta.

No calcular el dígito de control

Como ya se ha comentado, el número tiene importancia a efectos de evitar errores de lectura. Las barras que codifican el dígito se deben computar, aunque, como en el caso anterior, la cifra es posible colocarla en otro lugar.

Uso de colores incompatibles

Circulan fuentes donde cada barra y número se pueden generar en diferentes colores. Su uso es de tipo decorativo y no asegura que todas las combinaciones de barras y colores sean compatibles con el lector, aunque el fondo del papel sea blanco. Por supuesto, que si además de utilizar fuentes con barras de colores se establecen también los colores del fondo, los errores de lectura del código están asegurados. Su uso es aberrante y está totalmente desaconsejado para labores entomológicas (figs. 54- 90, 93 y anexo I, tabla XIV).

Hechura de las fuentes

Si el código de barras se ha generado con una fuente de letra no se deben utilizar nunca los formatos especiales, itálica, negrita, subrayados, tachados, efectos de sombra, etc. El símbolo no se podrá decodificar.

Intercalación de espacios

Entre las barras no se pueden intercalar más espacios que los que genera la propia fuente o el que se ha estipulado bajo código. Caso contrario el código resultará totalmente ilegible.

Código EAN13

Este tipo de código admite 13 números. El cálculo del dígito de control es igual que en EAN8 pero con 12 guarismos. La algoritmia similar. El resto de las características son idénticas a EAN8 (fig. 52). Por lo tanto, todo lo antedicho con respecto a fuentes, compatibilidad, colores, posicionamiento, prácticas desaconsejadas, etc. es aplicable en éste, por ello aquí no insistiremos más sobre el tema. Para los tamaños de impresión sin truncamiento, el escalado y magnificación en los códigos EAN8 y EAN 13, véase anexo I, tabla VI y VII.

AZTEC

El esquema de codificación es muy similar a un QRCODE simplificado pero sólo con un cuadrado de lectura de posición situado en el centro. Puede almacenar hasta 3748 dígitos. Se puede generar en formatos muy pequeños. Su codificación es rápida y efectiva.

Algoritmia para el cálculo del código AZTEC

La algoritmia es sencilla (anexo III, código III) y similar a la del QRCODE.

Uso de códigos AZTEC en color

Los gráficos AZTEC pueden ser en color pero su uso es mucho menos tolerante a errores en la decodificación que en los códigos de barras.

Prácticas desaconsejadas

Truncamiento

El código AZTEC no permite truncamiento.

Supresión de datos

Todas las advertencias formalizadas y sugeridas para los códigos de barras EAN son adaptables al código AZTEC, consignamos al lector por tanto al apartado oportuno.

No incluir numeración

Aunque en el código AZTEC se almacena información redundante, no se recomienda evadir la inserción del número del ID en las micro-etiquetas entomológicas, sobre todo si se está utilizando el dígito para formalizar operaciones de búsqueda y filtros en una tabla de datos. Igualmente no se debe obviar en operaciones de lanzamiento dinámico.

Uso de logos

El código AZTEC admite la inclusión de un texto, logo, imagen o icono como papel de fondo. La codificación es más laboriosa. La decodificación puede ser compleja y empeorarse hasta el punto de impedir la lectura del símbolo.

Uso de marcas de agua

Para que la lectura se efectiva se puede utilizar una marca de agua suave, incluso en color (fig. 95). Su uso debería ser anecdótico en entomología.

Figura 95: código AZTEC con marca de agua "SEA". Es un ejemplo límite de color de la impronta bajo el símbolo, pero se puede decodificar.

DATAMATRIX

Está hecho por módulos dentro de un patrón descubridor de perímetro. Cada símbolo tiene regiones de datos que contienen un juego de cuadrados nominales en un arreglo regular. Puede codificar 2844 caracteres en modo ASCII. Incluye un sistema redundante de corrección de errores. Se pueden generar códigos de tamaños muy pequeños. En entomología son por tanto bastante útiles. Se decodifican rápidamente y son muy fiables. Admite siete tipos de codificación: ASCII, C40, Text, X12, Edifact, Base256 y ASCIIGS1. Con cada una de ellas se pueden generar un número dispar de caracteres para lo cual el proceso requiere un consumo de tiempo variable (anexo I, tabla II, III, IV y V). El código DATAMATRIX permite truncamiento 8 x 18; 8 x 32; 12 x 26; 12 x 36; 16 x 36 y 16 x 48. Obviamente, la cantidad de datos que se pueden escribir disminuye a la par que la proporción del símbolo.

Algoritmia para el cálculo del código DATAMATRIX

La algoritmia no es compleja (anexo III, código IV). El símbolo DATAMATRIX se codifica y decodifica desde la izquierda en diagonal (fig. 96)

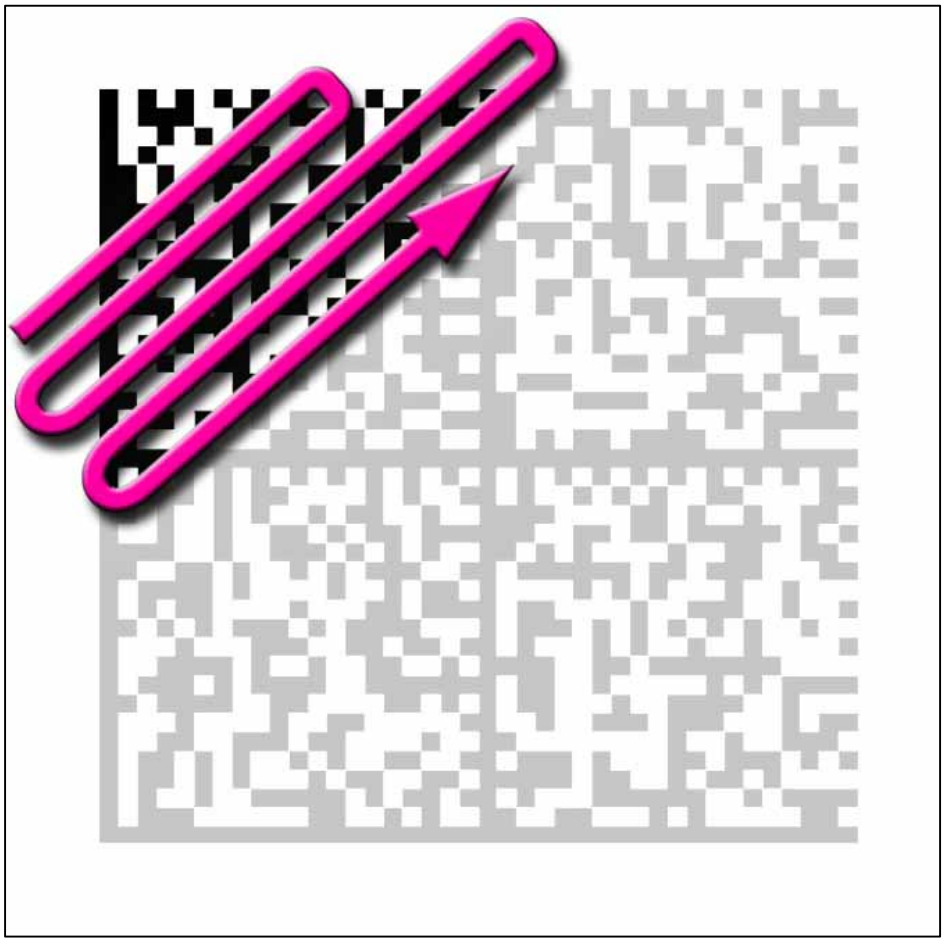


Figura 96: ejemplo de cómo se recorre el código DATAMATRIX para su codificación y decodificación.

Prácticas desaconsejadas

Supresión de datos

Todas las exhortaciones precisadas para los códigos de barras EAN y AZTEC, son aplicables a DATAMATRIX.

No incluir numeración

Léanse las notas y razonamientos al respecto, para el código AZTEC y EAN que también deberían tenerse en cuenta a la hora de generar código DATAMATRIX.

QRCODE

Puede almacenar de 7 a 7089 caracteres, dependiendo del tamaño y el tipo de codificación con respecto al de tratamiento de los datos redundantes (anexo I, tablas VIII, IX, X, XI y XII). Si es leve, el número aumenta y si es grande la cifra disminuye. El uso por encima de los 322 caracteres no es muy útil en entomología puesto que genera códigos demasiado grandes (322 = versión 6, tipo numérico, redundancia leve y tamaño 4). Quizá códigos más grandes pudieran tener aplicaciones museísticas o para material de exposición. Recordemos que el QRCODE se puede decodificar *in situ* con un teléfono móvil. En una caja de exposición, o al lado, se podría incluir un código con información relevante, biología de la especie direcciones Web, etc. La generación por debajo de la dimensión 4, compromete en bastantes casos la decodificación. Más abajo se muestran las zonas más importantes de lectura (fig. 97).

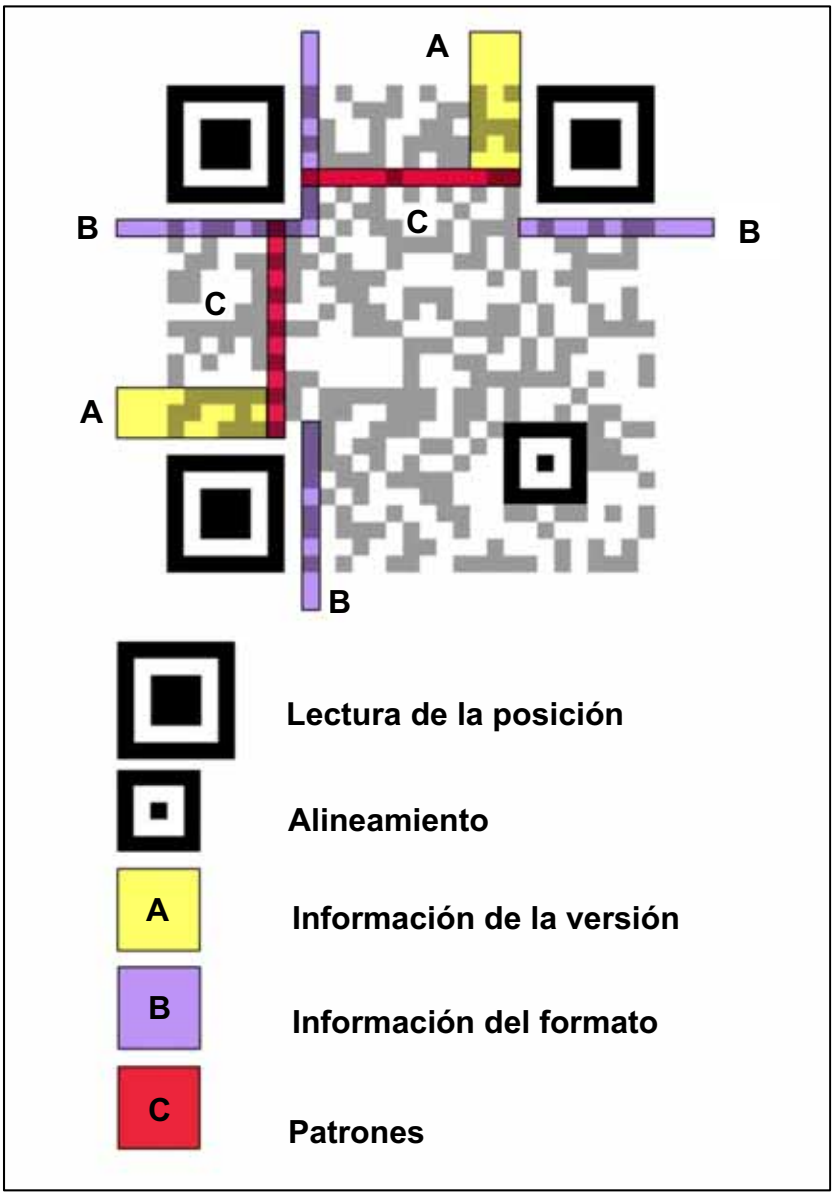


Figura 97: partes sensibles del QRCODE.

Algoritmia para el cálculo del QRCODE

Los cálculos no son complejos, máxime cuando existen varias librerías de código abierto que se pueden utilizar. La codificación se efectúa siempre a través de la lectura de la imagen (anexo III, código V). Es una algoritmia sencilla por lo que no requiere demasiadas explicaciones. Las referencias que se realizan a controles corresponden a sus funcionalidades: txtEncodeData (almacena el texto a codificar); pctEncode (muestra el dibujo codificado en tamaño real); txtCrono (un cuadro de texto que enseña el tiempo de generación); pctMuestra (muestra un ejemplo de la imagen reducida); lblCodigoTamaño (una etiqueta del tamaño que se ha utilizado para la generación). Dentro del procedimiento se realizan llamadas a dos clases estáticas: EphesiaUtilidades y EphesiaError. La primera encapsula todo lo concerniente a funciones, parámetros, constantes y código para utilizar las librerías de generación. La segunda clase todo lo concerniente a la captura y tratamiento de excepciones. Los sub-procedimientos que sobrecargan necesitan el paso de los valores de los controles por medio de parámetros y el tratamiento adecuado de las conversiones de formato. Además de escribir uno mismo el código existen multitud de programas en línea de uso gratuito, esta opción es menos versátil y más incómoda pero evita la escritura de algoritmos. Si se vaticina un uso intensivo, se recomienda la escritura de código, es sencillo y aunque en principio lleva tiempo su implementación rápidamente se sacará provecho de estas rutinas con el ahorro del mismo. Basándose en el código que se ofrece aquí y en la documentación de las librerías que se utilicen, su construcción no conlleva más de una hora. Para Microsoft Office hay complementos de pago compilados como controles ACTIVEX o en DLL'S, éstos se incrustan añadiendo funcionalidad a la plataforma.

Uso de códigos en color QRCODE

Los gráficos QRCODE pueden ser en color pero hay que tener en cuenta que no todos los colores de la fuente frente a los colores del fondo son adecuados. La conformación más legible es en negro puro y el fondo en blanco. En todo caso el uso de color es mucho menos sensible a errores en la decodificación que en los códigos de barras.

Prácticas desaconsejadas

Truncamiento

El QRCODE no permite truncamiento.

Supresión de datos

Todos los comentarios realizados para los códigos de barras EAN son aplicables a QRCODE, remitimos al lector al apartado correspondiente.

No incluir la numeración

Aunque el QRCODE almacena información redundante utilizando cálculos matemáticos Reed-Solomon (anexo III, código VI, Espinosa *et al*., [sin fecha]: 5) y en el caso de errores es difícil que no se puedan leer, no es conveniente obviar la inclusión del número del ID si se está utilizando el código para realizar operaciones en una tabla de datos.

Uso de logos

El QRCODE admite la inclusión de un logo, icono, imagen o un texto en una parte del cuadrado. Esto provoca una drástica reducción en la información que se puede almacenar. Si la codificación no está muy bien realizada la decodificación puede ser dificultosa (figs. 98 y 99).

Figuras 98-99: ejemplo de dos códigos. El de la izquierda se puede descifrar pero ocasiona frecuentemente errores. El de la derecha no se puede decodificar en la mayoría de las ocasiones.

El uso en entomología de pictogramas insertos en códigos QRCODE se debería limitar sólo a cuestiones muy puntuales, siempre y cuando se hubiera comprobado con extrema pulcritud que el código es plenamente decodificable por todos los lectores y algoritmias.

Uso de marcas de agua

Es posible utilizar este tipo de marcas pero los colores o tonalidades oscuras imposibilitan la lectura. Su uso debería limitarse al mínimo en biología y entomología (véanse ejemplo y comentarios al respecto para el código AZTEC perfectamente aplicables al QRCODE).

La transmutación en profundidad

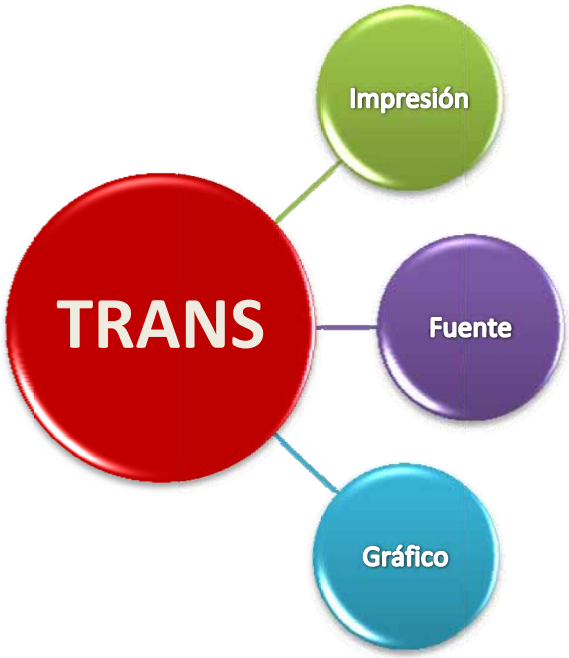


Figura 100: Diagrama radial de los procesos de transmutación.

Código EAN8 y EAN13

Es posible que las barras generadas no tengan el tamaño adecuado para su decodificación en lo concerniente a las proporciones entre la altura y la anchura (consúltense en el anexo I, tabla VI y VII las proporciones apropiadas). El código aquí se normaliza con el fin de disponer de él para su decodificación. Hay tres posibilidades: imprimir en papel, en un archivo o guardar el resultado como fuente y por último, convertir el formato a imagen (fig. 100). La compilación con una impresora virtual en formato PDF es factible, pero ha de tenerse presente que la decodificación se tendrá que realizar con un aplicativo adecuado para escanear el símbolo encapsulado en el PDF o en su defecto extraer la imagen del archivo resultante para su decodificación. Aunque existen aplicativos en la Web que realizan con soltura estas operaciones el proceso es engorroso. Para guardar el código generado con la fuente, lo único que hay que realizar es la conversión del texto del control donde se ha generado y guardar en formato de texto enriquecido con la fuente EAN incrustada. Por lo general esta conversión es transparente al usuario, basta simplemente con usar un cuadro de diálogo y establecer la extensión, rtf, doc, xdoc, etc. y especificar el nombre, la ruta y la carpeta donde se almacenará el código. La conversión a bitMap se realiza bajo el análisis de los píxeles generados, creando una matriz del tamaño adecuado y escribiendo en la misma los datos binarios. En ninguno de los casos se pueden alterar las proporciones. Si el código es en color se toma el formato adecuado (8, 16, bits, etc.) dependiendo del número de colores utilizados. Si se ha hecho un truncamiento se crea una matriz en píxeles correspondiente al tamaño descrito (probablemente en milímetros) con una función de conversión de milímetros a píxeles. El código finalmente se guarda en un archivo extensión, jpg, gif, bmp, png., etc. (anexo III, código VII).

AZTEC, DATAMATRIX y QRCODE

Se pueden almacenar de las tres maneras descritas. En el caso de truncamiento en DATAMATRIX, hay que respetar las proporciones: 8 x 18; 8 x 32; 12 x 26; 12 x 36; 16 x 36 y 16 x 48. En general ser aplicarán las mismas normas que se han descrito para los otros códigos EAN. También hay que tener en cuenta que DATAMATRIX y QRCODE permiten la modificación del tamaño del píxel. Si la imagen generada corresponde a los píxeles de pantalla habrá que adaptar el tamaño de la matriz teniendo en cuenta el tamaño de pixel establecido en el control de entrada.

La adquisición en profundidad

El código ya está impreso en papel o está normalizado en una matriz de imagen y está dispuesto para su lectura por medio de un aplicativo, con un lector láser o con una cámara (fig. 101).

Figura 101: diagrama de los procesos de adquisición.

Adquisición óptica

Figura 102: la lectura se puede realizar de varias maneras.

Pistola, lector laser, tipos y usos

Hay cuatro tipos de lectores de códigos: lápiz; lectores omnidireccionales (aquellos que pueden leer en cualquier dirección) para 1D; híbridos que leen símbolos 1D-2D y CCD (CHARGE COUPLED DEVICE). Los lápices generan una señal de baja frecuencia. Los lectores omnidireccionales componen una señal digital binaria y pura de las barras y los espacios. Los CCD, mediante un arreglo de fotodiodos hacen una fotografía del símbolo y la traducen a una señal similar a la enviada por el láser (HHLC, Hand Held Laser Compatible) o a la del lápiz óptico. La frecuencia HHLC generada por un lector láser es mayor a la que utiliza un lápiz. Obviamente, las señales requieren ser decodificadas para poder ser entendidas por la computadora, para ello existen diferentes interfaces. La más corriente es la KEYBOARD WEDGE. Se conecta por medio de un cable en "Y" realizando un puente entre el teclado y el lector. También se usa RS-232, aquí la información se envía al puerto de serie del PC y es decodificado por un TSR que convierte la señal en información de entrada de pulsación de tecla. De la misma manera se procede cuando la conexión es por medio de una toma USB. Por último, existe la WAND EMULATION que sólo se utiliza para terminales tontos y portátiles. El tipo de lector más adecuado para entomología es el híbrido para decodificación de códigos 1D y 2D, si bien suele ser el más caro. Se desaconsejan los lectores de tipo lápiz, muy económicos pero que requieren gran habilidad para su uso puesto que es necesario barrer a mano horizontalmente el código en distancias muy cortas. En los lectores omnidireccionales, inalámbricos o no, el operador mueve el aparato sobre el código. Para la lectura de las micro-etiquetas entomológicas colocadas en ejemplares de colección, la manera más útil consiste en apoyar la cabeza del lector sobre el cristal de la caja entomológica y bascular el cuerpo con pequeños movimientos radiales hacia arriba y hacia abajo, (fig. 103). Para una correcta lectura tiene que existir una separación mínima en el sentido horizontal entre las etiquetas de los especímenes (fig. 104).

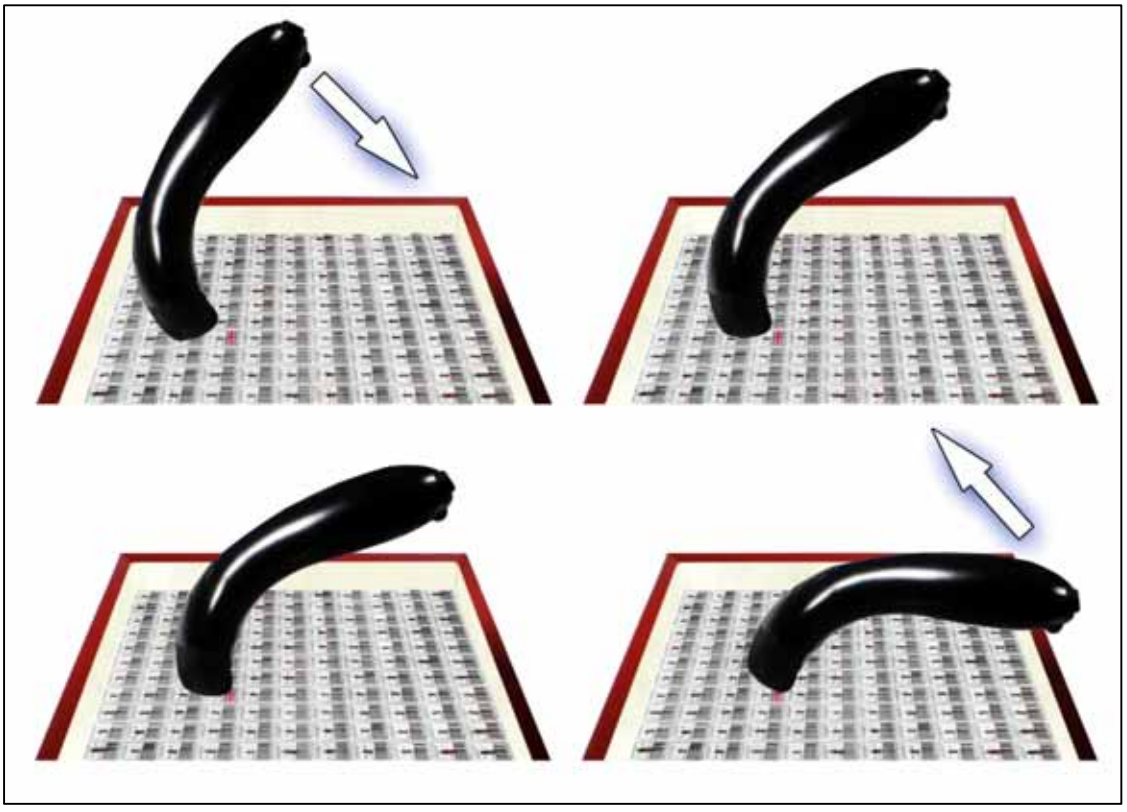


Figura 103: forma correcta de efectuar una operación de lectura.

Figura 104: fragmento de una caja entomológica donde se observa la separación mínima para la lectura de los códigos mediante un lector con haz de barrido modificado.

Limitación de la anchura del haz de barrido en lectores omnidireccionales

El haz generado por el cabezal del lector puede configurarse para tener mayor o menor anchura. Para leer etiquetas de colecciones es necesario que el haz sea lo más estrecho posible tolerado por la pistola. Quince milímetros suele ser el mínimo. Esta anchura se puede modificar leyendo los comandos de configuración que tiene el propio lector. Si se va a utilizar el aparato constantemente con diferentes anchuras, lo más práctico es limitar de forma física del haz en la pantalla del lanzamiento colocando trozos de espuma densa y blanca. Una restricción de la entrada a 15 mm es la más adecuada en entomología y genera, alejando o acercando el cabezal del lector, haces que oscilan entre 15 y 120 mm (fig. 105).

Figura 105: limitación física por medio de sendos pedazos de espuma del haz de barrido en el cabezal del lector de códigos de barras 1D. Esta apertura es la más adecuada para la lectura de micro-etiquetas y para la distancia de barrido sobre el cristal de una caja entomológica de 50 a 70 mm de altura. Para la adquisición y decodificación de códigos más anchos se tienen que extraer los tramos de espuma como se muestra en la fotografía de la parte superior. Nótese en ésta que el haz queda fuera de campo en el gráfico al haberse inclinado ligeramente el cabezal lector de la pistola hacia la parte superior del eje en sentido vertical.

Configuración de lectores

Normalmente se leen los códigos de configuración facilitados por el fabricante con el propio aparato. Éstos son parecidos a códigos de barras con uso dinámico y su lectura provocará el cambio en la configuración del lector. Existen algunos modelos más raros que se configuran con DIP SWITCHES o enviándoles los comandos de programación por el puerto de serie. Los que además tienen CCD se pueden configurar desde el aplicativo que brinda el fabricante o con un controlador estándar.

Cámaras independientes, cámaras en teléfonos móviles y tabletas

Estos aparatos funcionan de la misma manera que los lectores CCD. Algunos modelos no trasmiten datos binarios insertables en un control de imagen, sino que generan directamente el archivo de matriz de imagen, por ejemplo JPG. Se conectan en un puerto USB o bien de forma inalámbrica con un captador WIFI. Algunas trasmiten las fotografías tomadas, por medio de una tarjeta de memoria especial que es capaz de transferir la información de forma remota. Algunas SLR (NIKON, CANON) se ofrecen con aplicativos para el control de la cámara a distancia desde el computador y almacenar directamente las tomas (anexo V). Las imágenes en formato RAW, no son decodificables, hay que revelarlas para almacenarlas en una matriz GIF, JPG, etc.

Cámaras de vídeo y Web

Trasmiten la lectura simultánea de varias fotografías (frames) en un lapso de tiempo, con una velocidad de barrido determinada. Para ver el vídeo basta conectar la cámara pero tiene que tener instalados sus drivers. La película se puede mostrar en cualquier control que admita este formato o en aplicativos específicos de edición de vídeo. Por lo general, las cámaras van acompañadas de un DVD donde se ofrecen diversos entornos para la visualización de las tomas, basta con instalar los aplicativos. El código no se puede decodificar directamente del vídeo, se debe capturar una de las imágenes y guardarla en una matriz de puntos normalizada que sirve para la decodificación del símbolo. La adquisición desde una cámara Web es delicada en cuanto al enfoque de las etiquetas y a la colocación de la cámara. La etiqueta debe iluminarse de manera plana, sin sombras y con una fuente de luz fría y potente. La iluminación tiene que ser uniforme y sin cono por lo es conveniente usar un difusor, por ejemplo una hoja de plástico semitransparente. Es frecuente provocar aberraciones que dificultan la decodificación del código debido al objetivo angular. Un buen sistema consiste en poner la cámara en un pie con cremallera de deslizamiento para ajustar su enfoque, vale cualquier columna de una vieja lupa (fig. 106).

Configuración de cámaras

Todas permiten especificar las imágenes trasmitidas por segundo, la tonalidad, brillo, contraste, etc. Lo más importante es ajustar la resolución al máximo y desactivar la corrección de entornos con poca luz. Es conveniente efectuar un balance de blancos con una cartulina.

Escáner

Los códigos se capturan en forma de imágenes que se almacenan en el computador. La resolución mínima recomendada es de 300 pp. Los símbolos en blanco y negro se tomarán en escala de grises sin opciones de suavizado y los gráficos en color con una profundidad de 24 bits.

Figura 106: soporte improvisado para la lectura de códigos por medio de una cámara Web que se conecta al computador.

La lectura de códigos

Códigos EAN8 y EAN13

En el caso de estos modelos de códigos de barras, el lector láser al activarse por movimiento y/o pulsar el gatillo, manda directamente los impulsos que corresponden con los códigos leídos más un código de pulsación de tecla establecido en la configuración del periférico. Basta colocar el foco de acogida en un control (por ejemplo el Bloc de notas) para que la información se muestre en formato de número. En ocasiones la cadena no llega al control, esto puede sobrevenir por incontables y muy diversos motivos (véase anexo I, tabla XV con entradas de las ocurrencias más frecuentes).

AZTEC, DATAMATRIX y QRCODE

La lectura de estos códigos se realiza de manera similar al de la toma de una fotografía. En los lectores híbridos se puede configurar el formato de salida. La captura, si no se realiza automáticamente con los aplicativos instalados junto al driver del lector, se puede interceptar desde el puerto donde está establecida la entrada. También se pueden dirigir los datos hacia un control que sea capaz de contener imágenes de matrices de puntos. Lo más usual es que se guarden automáticamente en una carpeta establecida por el usuario, con un número, un prefijo y un sufijo, igual que se hace con los escáneres de sobremesa. Si se ha programado personalmente el aplicativo y se desea un objeto de automatización basta con comprobar desde la rutina, por medio de un contador, cuando se produce un cambio en la carpeta, capturar el archivo nuevo e incrustarlo en un control de imagen con un comando del tipo imageFromFile. Si la imagen no llega o presenta tramas, rallados, se observa aberrante, etc., hay que comprobar que el lector está configurado correctamente para la tarjeta gráfica instalada y que es soportado por el sistema operativo.

Adquisición desde aplicativos

Basta con abrir el cuadro de diálogo de apertura del explorador de archivos, escoger el archivo y pulsar aceptar. La rutina que lanza la decodificación se puede colocar en la captura de este evento, o en un botón colocado al efecto en un formulario. Sea cual fuere el suceso se abrirá la imagen en un control capaz de gestionar gráficos. Desde un aplicativo también es posible capturar directamente en vuelo las imágenes tomadas por una cámara o un teléfono móvil (fig. 107-110 y anexo III, código VIII).

Tolerancia de lectura

Envejecimiento y desgaste de material impreso en EAN8 y EAN13

La tolerancia es extrema hacia el desgaste, borrado, manchado, etc. en el sentido perpendicular a las barras. Muy escasa al deterioro en sentido longitudinal. Sobre todo la pérdida o fusión de dos líneas adyacentes (figs. 111- 130).

Envejecimiento y desgaste de material impreso códigos 2D

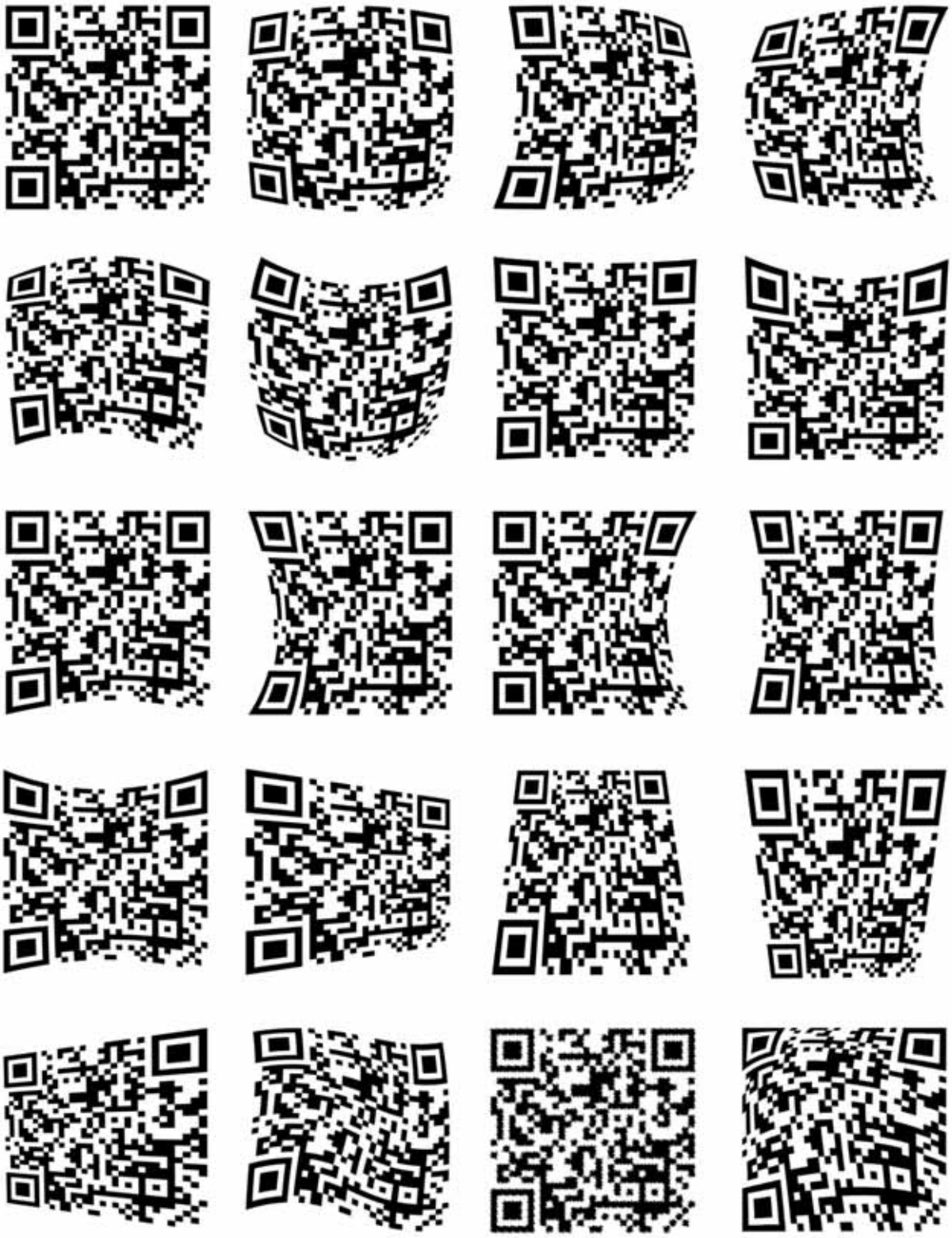
Estos códigos soportan quebrantos de datos de muy diversa índole, incluso se pueden llegar a leer etiquetas o gráficos con un 50% de pérdida, todo depende del nivel de redundancia con que se hayan generado.

Aberración, ruido, contraste, color y tonalidad en los códigos 2D

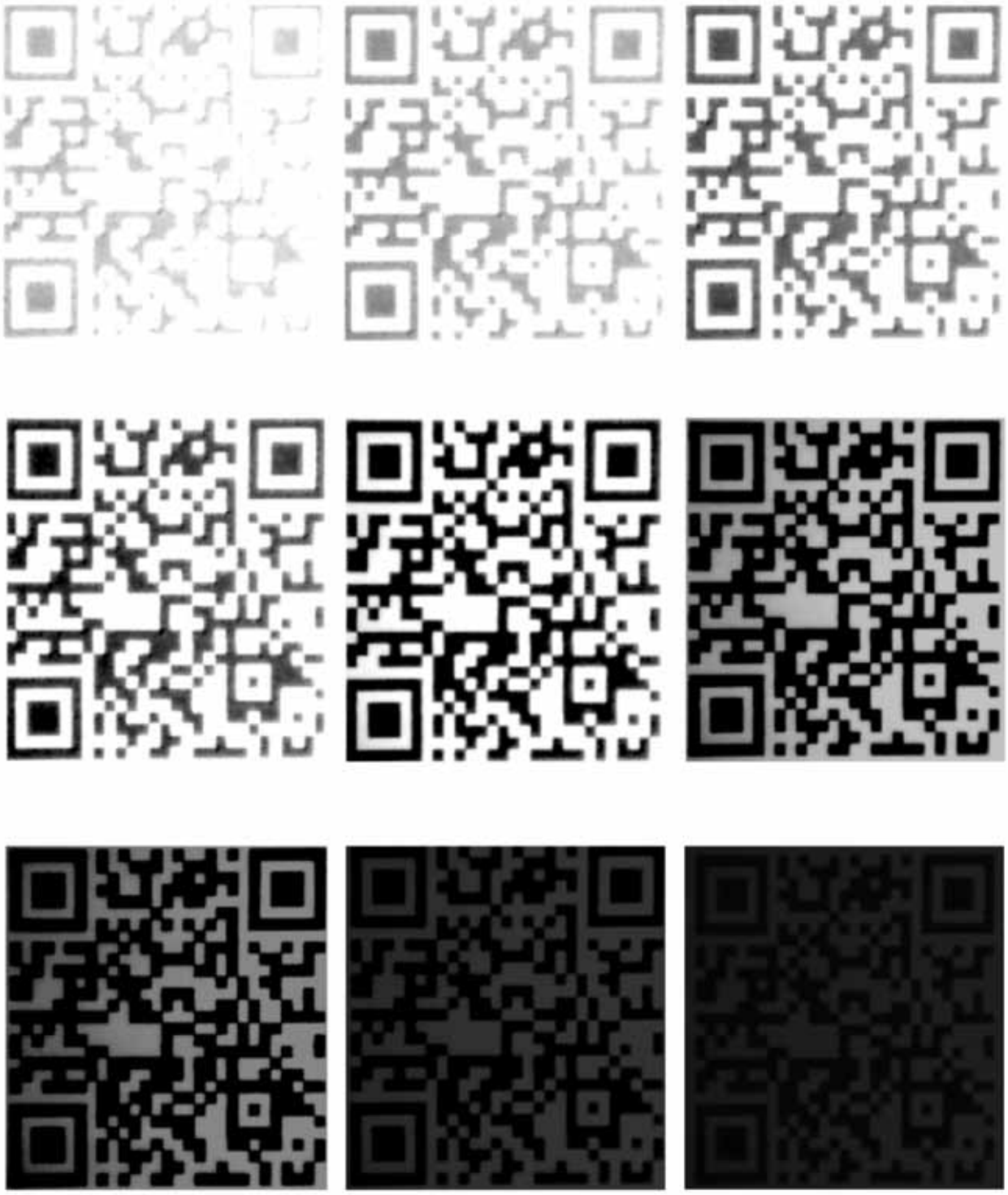
No soportan grandes aberraciones de imagen (figs., 131- 150). Tampoco resisten gráficos que presenten un contraste demasiado tenue entre el fondo y los cuadrados del símbolo (fig. 151-159). En estos casos puede ser posible aplicar algoritmia de binarización (Magro, 2013), y/o aplicativos editores de imagen para mejorar el dibujo. Sobre la tolerancia al color ya se ha escrito más arriba, ténganse en cuenta las advertencias al respecto.

Figura 107- 110: las figuras de arriba y del centro corresponden al módulo CODIFICADOR-DECODIFICADOR DE CÓDIGOS 2D (carpetas DECODIFICADOR UNIVERSAL y DECODIFICADOR POR CÁMARA WEB) para EPHESIA, se acaba de asignar la imagen de un código AZTEC y se ha decodificado respectivamente desde un archivo y con una captura de vídeo. Abajo, a la izquierda, lectura con el GENERADOR DE CÓDIGOS EAN para EPHESIA. A la derecha, abajo, lectura con un control ACTIVEX alojado en una hoja de EXCEL.

Figuras 111-130: etiquetas entomológicas. En las dos primeras filas se muestran ejemplos de códigos EAN con tipos de deterioros muy graves pero decodificables con lectores omnidireccionales. Las dos últimas filas no se pueden descifrar al presentar defectos verticales y pérdida o fusión de alguna o varias líneas de las barras.



Figuras 131-150: primera fila, la imagen 131 no está deformada, en las siguientes se aprecian distintas aberraciones, esfericidad convexa, esfericidad convexa derecha y esfericidad convexa izquierda. Segunda fila: esfericidad convexa hacia arriba, esfericidad convexa hacia abajo, esfericidad cóncava y esfericidad cóncava hacia arriba. Tercera fila: esfericidad cóncava hacia abajo, esfericidad cóncava izquierda, esfericidad derecha y deformación de tonel (Pin Barrel) horizontal. Cuarta fila: tonel vertical, fuga derecha, fuga superior y fuga inferior. Quinta fila: fuga izquierda, aberración sinusoidal, tramado y de curvatura irregular en el cuadrante superior.



Figuras 151-159: gradaciones de tonalidad que dificultan la decodificación del símbolo por el lector o el aplicativo. Las tres figuras centrales son perfectamente legibles. Las de la primera línea sólo se pueden descifrar con un alto índice de errores en las dos últimas etiquetas, es decir, la dificultad en la decodificación es directamente proporcional a la mayor claridad de la muestra con respecto al fondo. Con las de la última línea ocurre lo mismo pero la dificultad en la decodificación es directamente proporcional al menor contraste de la muestra con respecto al fondo. A efectos demostrativos todas las figuras han sido tomadas desde una cámara Web configurada para blanco y negro, de ahí la poca resolución, una pequeña aberración en la verticalidad, sucinta pérdida de foco y la generación de ruido en la captura de la imagen. La luminosidad se ha forzado en la sensibilidad de exposición del aparato.

El lanzamiento dinámico en profundidad

Algoritmia

Por excesivamente prolijo no podemos incluir aquí todo el código fuente de las opciones propuestas. Hay que tener en cuenta que la rutina completa puede superar fácilmente las 20.000 líneas. En cualquier caso, gran parte de las instrucciones se deben a las especificaciones y funcionalidades de los controles alojados en los formularios (tamaño, interfaz gráfica, manejadores, estados, control de eventos, etc.). Aquí nos centraremos en las funciones y fórmulas más importantes. En el anexo III código IX se ofrece la algoritmia completa para realizar el CODIFICADOR-DECODIFICADOR DE CÓDIGOS 2D (carpetas DATAMATRIX, QRCODE, AZTEC, DECODIFICADOR UNIVERSAL y DECODIFICADOR POR CÁMARA WEB). Con ellas, adjuntando las librerías, se pueden codificar símbolos, AZTEC, DATAMATRIX, MICROQRCODE y QRCODE, y decodificar EAN8, EAN13, UPCA, CODE39, ITF, PDF417, CODABAR, MSI, PLESSEY, AZTEC, DATAMATRIX, MICROQRCODE y QRCODE.

Automatismo y bucles

El lanzamiento dinámico se puede realizar de dos formas, una con la intervención del usuario y la otra de manera transparente con el control de eventos en cascada y bucles.

Lanzamiento dinámico con intervención del usuario

Se trata de ofrecer por medio de controles una opción de bifurcación de la operación resultante. Por ejemplo, en un cuadro de texto se ha vinculado el comportamiento de la captura de su evento al contenido de una lista desplegable, de manera que al escoger una opción en la lista ésta trasmite un código que se almacena en una variable. A leer el código y capturar el impulso final INTRO en el evento pulsación de tecla del cuadro de texto, se evalúa el contenido de la variable de la lista desplegable que ha establecido el usuario y se actúa en consecuencia. Supongamos que la lista desplegable tiene dos entradas: “Crear informe de citas de lepidópteros según el año del registro activo” y “Establecer baja del registro activo”. El usuario abre la lista desplegable y escoge la primera opción que se almacena en una variable bit con el valor 1. Se acerca el lector y se lee el código numérico “69” + una pulsación INTRO. Al capturar el evento de la tecla RETURN en el cuadro de texto se evalúa el contenido de la variable, este caso es 1. Con una sentencia de bifurcación se discrimina la funcionalidad, si la cadena bit corresponde a 1 se llama a la algoritmia que crea informe de citas de lepidópteros según el año del registro activo. Esta dinámica se puede programar o aplicar a cualquier aplicativo que contenga controles sensibles a una pulsación de tecla y en el que se puedan utilizar macros o fórmulas. Si por ejemplo en Microsoft EXCEL colocamos en una celda una fórmula que multiplica por 2 que se activa con la tecla INTRO, y estacionamos el cursor donde se debe colocar el valor de cálculo, sucedería lo siguiente: el lector trasmite la cadena “69” + INTRO, el número se escribe donde hemos colocado el cursor y el RETURN provoca la pérdida de foco de esa celda colocando el cursor en la siguiente y lanzando la fórmula que arroja el resultado de 4761. Evidentemente valerse de un

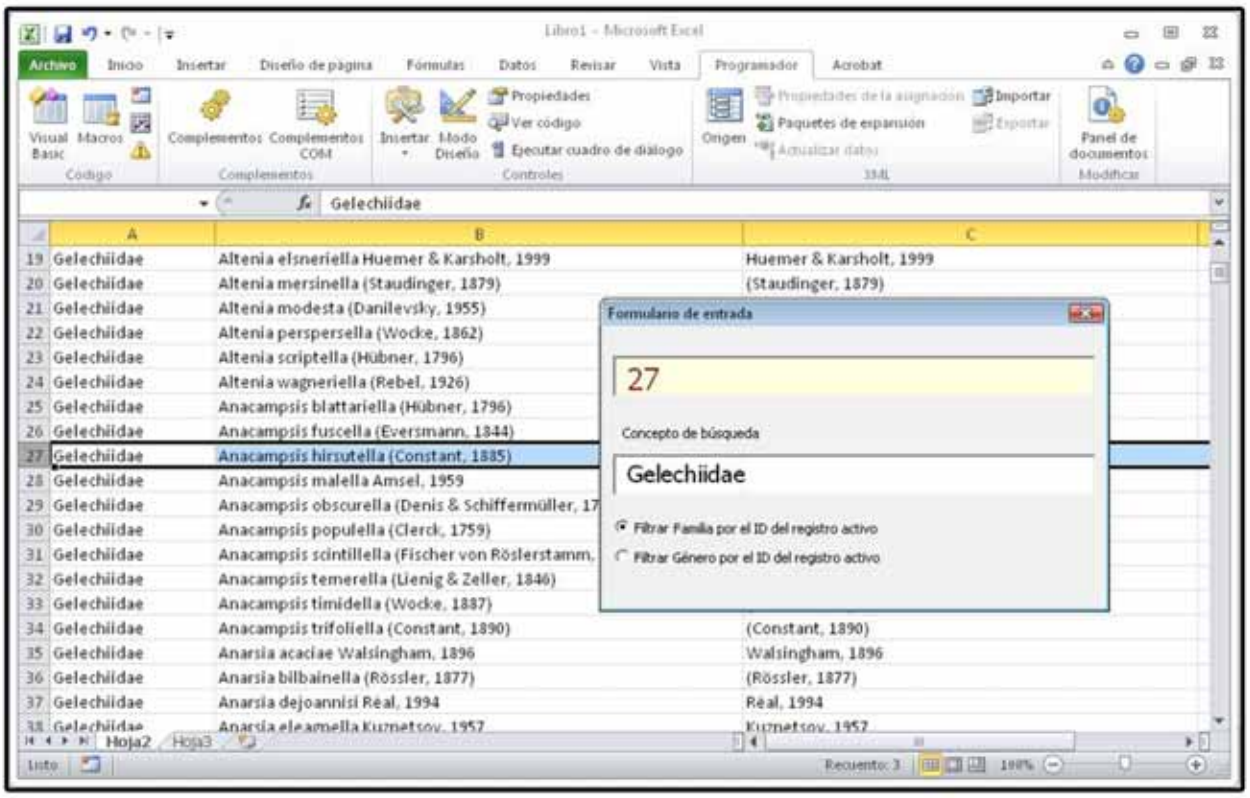
lector de códigos para multiplicar es banal. Se puede utilizar una lectura con intervención del usuario para realizar cualquier tarea que se pueda programar, incluir en una macro o en una fórmula. Las plataformas y los aplicativos son indiferentes. En las figuras 161 y 162, se muestran respectivamente dos documentos, uno en Microsoft WORD y otro en EXCEL (en modo programador). El usuario activa una casilla de verificación, decodifica el código con el lector (27 y 16) que lanza el resultado de la operación tras recibir la tecla INTRO y procesar el contenido de la algoritmia codificada en Visual Basic. Otros ejemplos: el operario abre un cuadro desplegable y establece la opción “Informe de caja” al decodificar con el lector la etiqueta el aplicativo se sitúa en el registro con dicho guarismo, lee el número de la caja y muestra un informe de la misma a la par que abre un cuadro de diálogo flotante durante unos segundos con las opciones más usuales (realizado para EPHESIA, 2014, figura 163). Esta operación se puede ejecutar de la misma forma leyendo los códigos impresos en etiquetas que se han pegado en las cajas. Otro modelo dinámico: el entomólogo hace clic en la imagen de la luna del registro activo de la tabla de datos y se lee el código de la etiqueta con el lector, el aplicativo busca la fecha de captura y la hora en el registro activo, genera las fases lunares para el día, la hora y el año establecido, calcula la edad de la luna, el factor de luminosidad y muestra un formulario con la imagen de la luna de los días anteriores y posteriores, un calendario con el día subrayado y a la vez señala la lunación en la posición anual correspondiente al día del registro activo de la decodificación mandada por el lector (todo ello realizado para EPHESIA, figura 164). La lista de posibilidades del lanzamiento dinámico con intervención del usuario sólo está limitada por nuestras necesidades e imaginación. En el anexo III, código X, puede verse un ejemplo de algoritmia de bifurcación de un cuadro de lista desplegable que sobrescribe la propiedad texto de una etiqueta que trasmite el valor de una variable que se utiliza para lanzar el proceso a continuación, tras la captura de la tecla emitida por el lector.

Lanzamiento dinámico transparente en cascada

En este tipo de activación no se requiere la participación de ningún operario ni antes ni después de la decodificación del código por el lector. Es posible programar las tareas de tres maneras diferentes, por separado o ambas entrelazadas. Una opción es lanzar la algoritmia en bucles de procesos en cascada inmediatamente después de la decodificación y la captura de la señal de tecla mandada por el lector. Otra configurar el lector para que envíe más de un impulso de codificación de tecla, antes y después de la decodificación. La última consiste en utilizar todos los eventos generados por el paso de los controles. El lanzamiento de los procesos en cascada ya se ha explicado suficientemente en el apartado correspondiente en la introducción a los usos de códigos 1D y 2D en la entomología. El uso de la captura de los eventos de los controles es igualmente útil cuando las tareas pasan por ellos, es decir, cuando la algoritmia no está basada exclusivamente en código o en procedimientos de consola. Por lo general, no es necesario nada más que la trasmisión de un evento de tecla al final de a decodificación pero todos los lectores de barras permiten la programación de varios sucesos de entrada y de salida. En el caso de la decodificación por medio de aplicativos no queda más remedio que programar estos lances de trasmisión. Una muestra del automatismo dinámico en cascada en entomología sobre una tabla de datos de preparaciones microscópicas podría ser que, al recibir la pulsación de la tecla RETURN se hiciera lo que se propone en la figura 160, como un diagrama secuencial de procesos aplicado a la gestión de las preparaciones y realización de informes vinculados al ID trasmitido por el lector.

Figura 160: diagrama de procesos de un lanzamiento dinámico sin intervención del usuario sobre el aplicativo.

En la figura 165 se muestra un ejemplo del diagrama expuesto, parado en el proceso de lanzar el cuadro de diálogo de impresión que espera la configuración de la impresora y la confirmación del usuario. Cuando se genera el evento aceptar en el cuadro de diálogo imprimir, este se captura para cerrar el módulo PREPARACIONES MICROSCÓPICAS, guardar los cambios, desconectar la base de datos del servidor y cerrar la aplicación (programado para para EPHESIA). Un ejemplo de lanzamiento dinámico con decodificación a través del aplicativo: se parte de la búsqueda del ID del ejemplar en la tabla de datos, si se encuentra se genera el código 1D o 2D y cuando se han terminado de pintar, se lanzan los procedimientos, se abre el módulo de generación de etiquetas entomológicas y se colocan automáticamente todos los datos justificados en los campos de la etiqueta que se han pasado por referencia; se genera la coordenada UTM de la localidad; se buscan preparaciones microscópicas y se realizan la etiquetas necesarias para el ejemplar, frascos y portas. Para terminar, se muestra una vista previa de las etiquetas y se ofrece el cuadro de diálogo impresión y cuando se pulsa el botón aceptar se imprimen las etiquetas generando un informe de actualización de etiquetas que se escribe en el módulo principal (fig. 166- 167 programado para EPHESIA). Sin dificultad es factible programar en WORD, EXCEL, etc. operaciones más sucintas (fig., 168). Valgan estos ejemplos para dejar meridianamente claro que simplemente con mover el lector por encima de un código de barras, se pueden efectuar instrucciones con un alto índice de sofisticación.



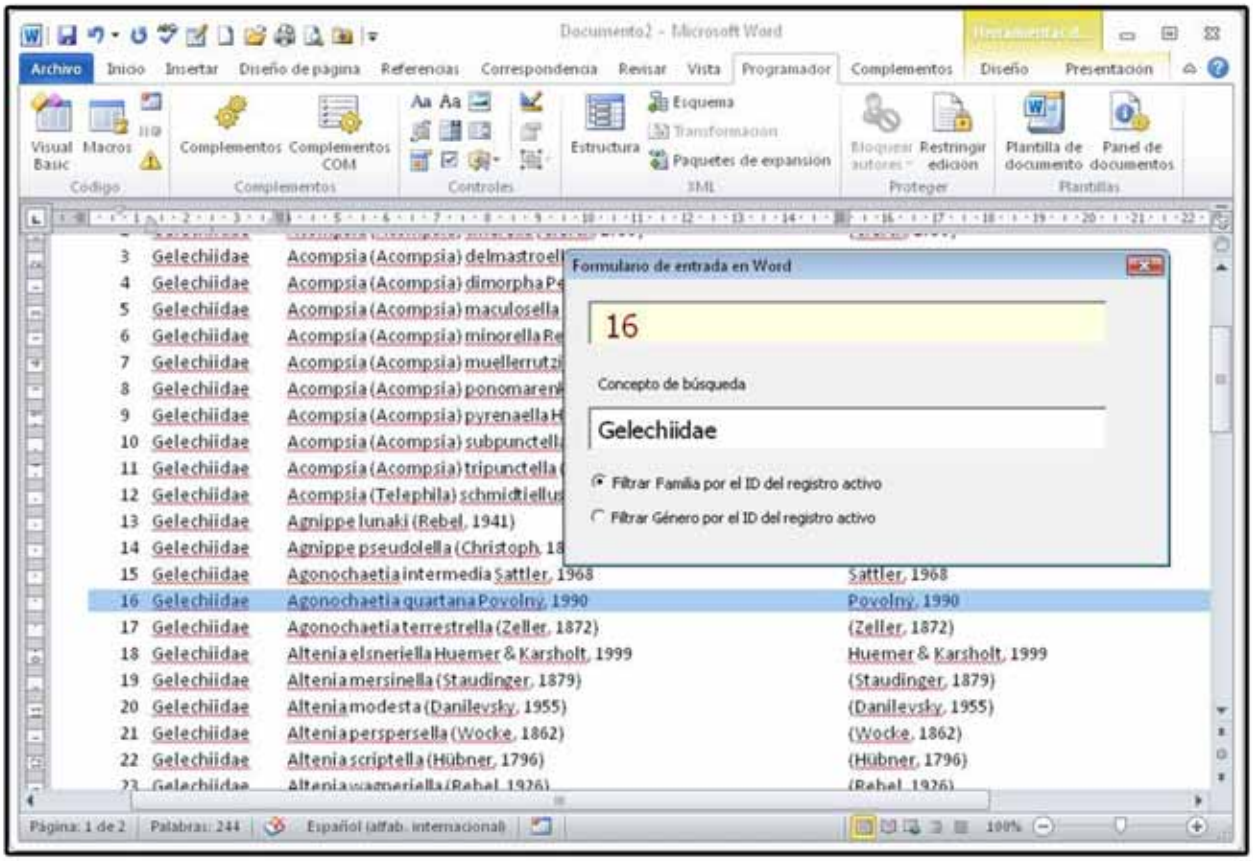


Figura 161- 162: lanzamiento dinámico en EXCEL y WORD. Al leer el código se busca en una lista y en una tabla, respectivamente, la especie que corresponde al ID decodificado.

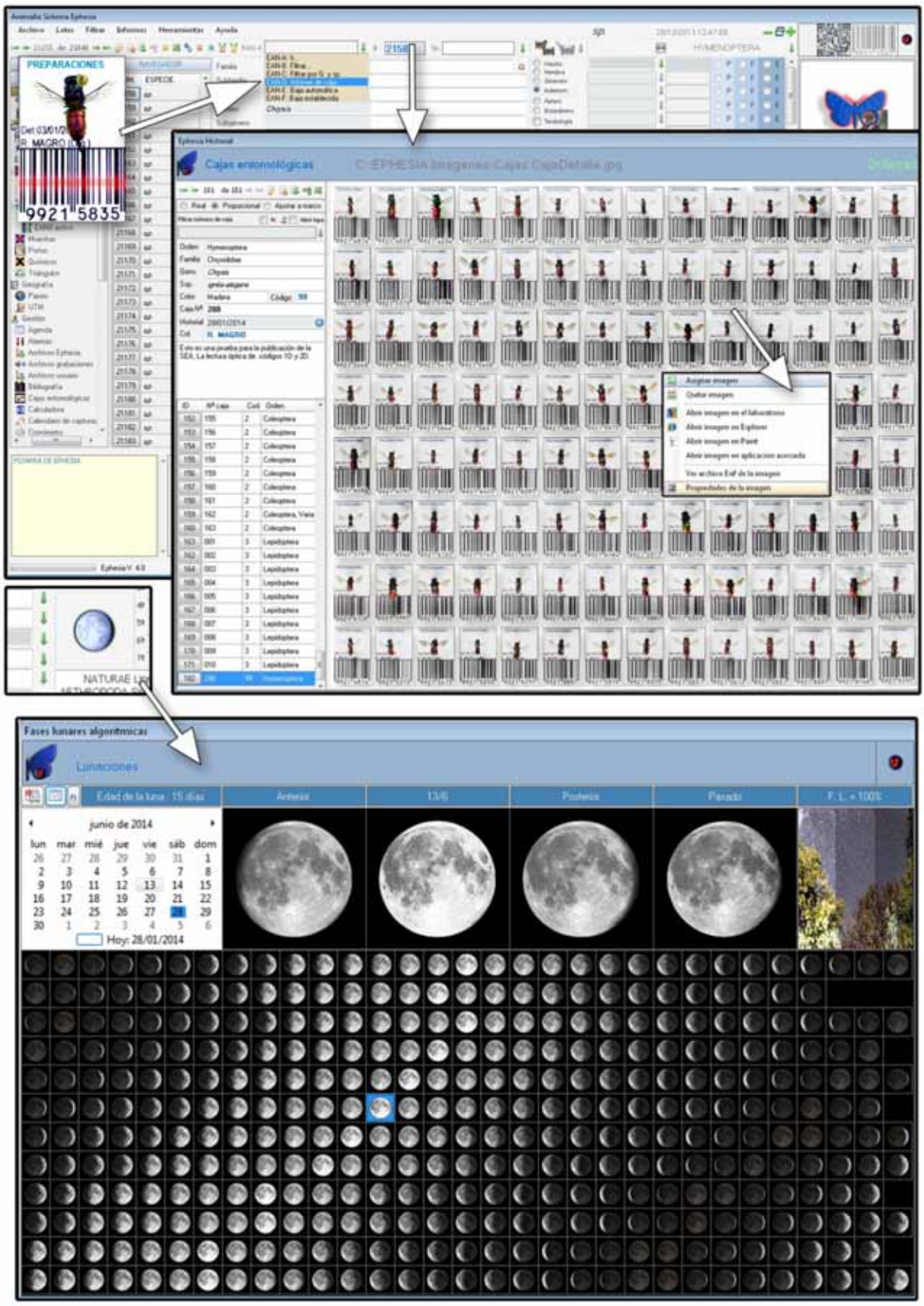


Figura 163- 164: lanzamiento dinámico con intervención del usuario sobre un cuadro desplegable que abre un aplicativo que gestiona cajas entomológicas, muestra la información según ID y abre un menú flotante. Abajo, lanzamiento que se activa según la lectura del ID de la etiqueta y haciendo clic en la luna del registro activo. Genera la fase lunar del día y del año tras el análisis por algoritmia de la fecha de los datos que figuran en la tabla principal.

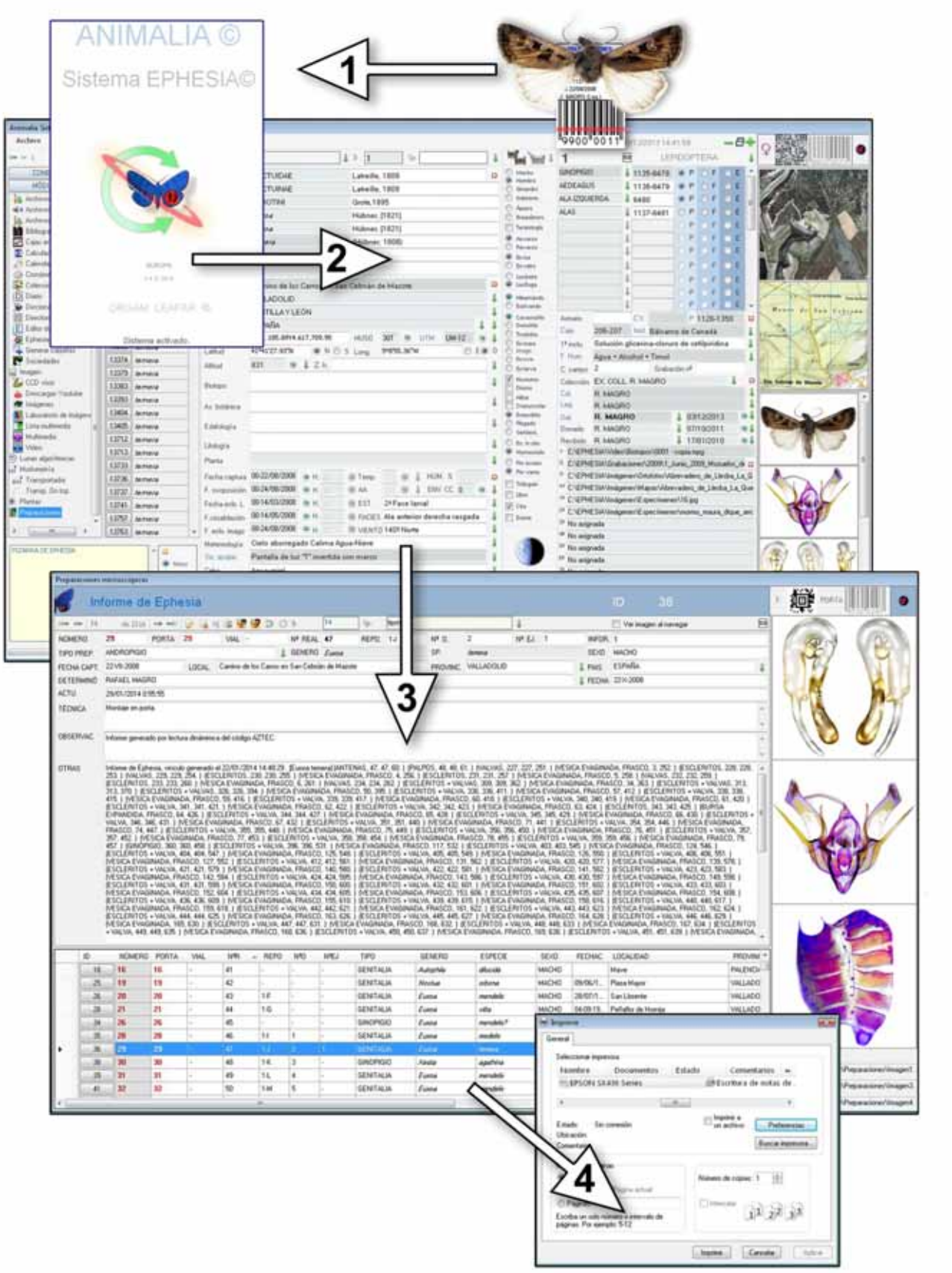
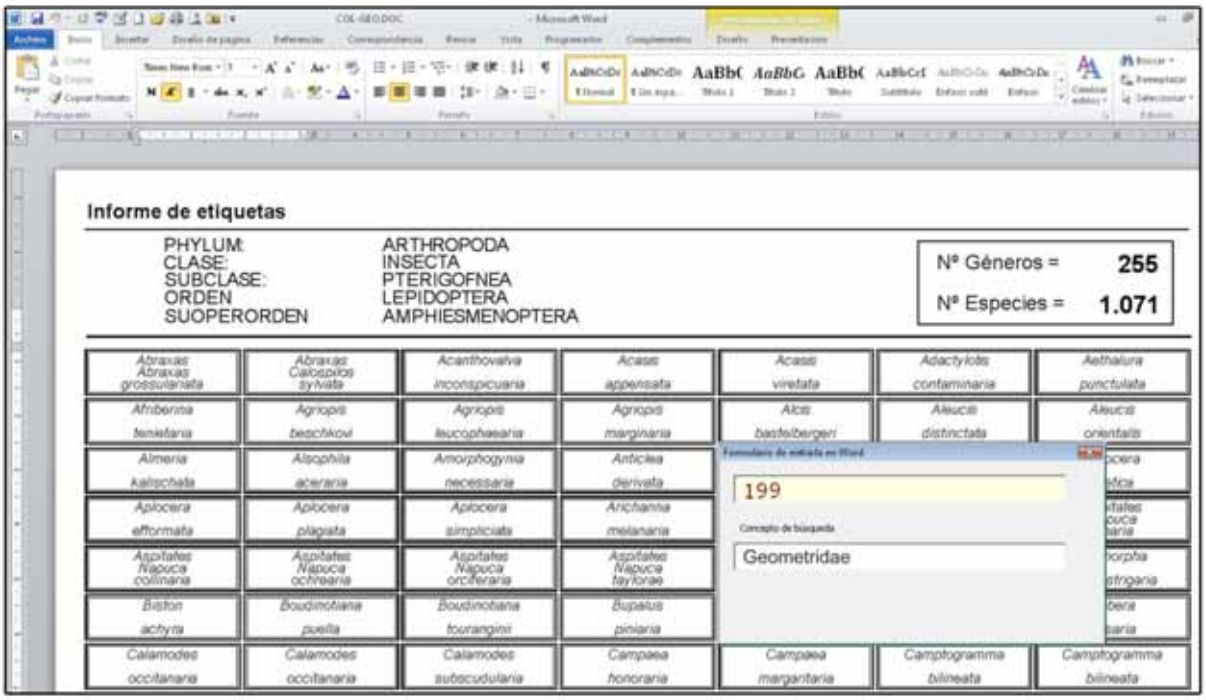


Figura 165: desarrollo del diagrama 160. **1**: el lector lanza el aplicativo con el parámetro del ID decodificado de la etiqueta. **2**: se muestra el registro por el ID. **3**: se lanza el módulo "PREPARACIONES MICROSCÓPICAS" y se genera un informe según el ID que muestra todas las preparaciones de la misma especie que existen en la tabla de datos, por tipos, ubicaciones, inclusión, etc. Para terminar se muestra un cuadro de diálogo de impresión.

Figura 166-68: arriba, ejemplos de lanzamiento dinámico para la generación de un juego completo de etiquetas a partir de la lectura del ID de un código de barras provisional de la especie. Módulos EDITOR DE MICROETIQUETAS y MICROETIQUETAS EAN para EPHESIA. Abajo una muestra de generación de etiquetas de colección de género y especie a partir del ID de la familia, en WORD. Se lanza en un documento incrustado con una tabla de ACCESS.



Impresión de códigos

Tipos de impresoras adecuadas para los códigos

Para micro-etiquetas, es válida cualquier impresora fotográfica de chorro de tinta o sublimación de cera que ofrezca calidad alta (4000- 6000 DPI.). En blanco y negro también se pueden utilizar impresoras láser. En las etiquetas grandes no es necesario que la impresión sea de extrema calidad. Los lectores las decodifican sin problemas pero para exposiciones las etiquetas deberían poseer una impecable presentación. Una última y significativa consideración en relación con el tamaño del código es saber qué alternativas brinda el proceso de impresión designado. El tamaño mínimo y el reajuste de ancho de banda (BWR) de cada símbolo varían según el proceso de impresión escogido e incluso según el modelo de impresora.

Tipos de soportes para impresión de códigos

La impresión de micro-etiquetas en papel se deberá realizar a una resolución mínima de 4000 DPI. El soporte no podrá ser poroso, rugoso, tramado, etc. Lo ideal es papel fotográfico ultra brillante del gramaje adecuado para el tipo de ejemplar, con el fin de poder penetrar el soporte con el alfiler entomológico sin doblarlo (anexo I, tabla XVI). Con respecto a las etiquetas más grandes, por ejemplo para productos químicos, es posible utilizar papel corriente, sin embargo estos frascos suelen ensuciarse con facilidad por lo que se recomienda un papel plastificado o lavable. Para ello es igualmente práctico pero es más laborioso, imprimir el texto invertido en acetato cristalino y pegarlo con un adhesivo transparente. Para las tapaderas de los frascos de preparaciones resultan muy ventajosas las transparencias adherentes. Otro sistema consiste en utilizar papel termo-fusible ya que es un tipo de material con el que se puede transferir la tinta por medio de calor al material donde se va a imprimir. El sistema denominado de “Calcas” consiste en el uso en una película especial muy fina que una vez impresa se recorta y se pega con agua o con un disolvente adecuado a la superficie definitiva. Una vez seca se remueve la cutícula como si fuera una calcomanía, dejando en la superficie la tinta original de la impresión. Este sistema proporciona acabados de extrema limpieza.

Resultados

Posibles aplicaciones de los códigos 1D y 2D en entomología y biología

Durante el transcurso del texto ya se han esbozado algunas de las aplicaciones. Los usos de los códigos en este campo pueden ser muy variados. Aquí resumimos algunas que a nuestro parecer son especialmente útiles:

Bajas y altas
Marcar como borrado o dar de baja un registro o grabar uno nuevo según el ID.

Ejemplares de colección
Es posible etiquetar los ejemplares de colección con códigos 1D y 2D (figs. 51, 54-90, 91-92, 104 y 163)

Preparaciones microscópicas
Las preparaciones microscópicas también se pueden etiquetar con códigos de barras (fig. 169). El código AZTEC resulta muy ventajoso en la codificación de cajas con frascos de preparaciones (fig. 170).

Exposiciones y museística
Potencialmente práctico es colocar QRCODE en las cajas de exposición con información relativa al espécimen mostrado (fig., 171).

Cajas entomológicas
Igualmente eficaz es ubicar una etiqueta en el frontal de las cajas entomológicas que codifique su contenido (figs., 163 y 172).

Recipientes
Los tubos de ensayo y recipientes de procesos químicos en serie etiquetados con AZTEC o DATAMATRIX simplifican la lectura de datos.

Químicos
Los tarros de sustancias también se pueden marcar con QRCODE (fig. 173).

Libros
La decodificación del ISBN y/o ISSN es recomendable para la tipificación de los libros científicos que hay en la biblioteca entomológica, sobre todo cuando las referencias están escritas en una base de datos y es factible la realización de un lanzamiento dinámico (fig. 174).

La gestión de separatas
Ayuda a la búsqueda de separatas.

Marcaje
Se pueden utilizar etiquetas para codificar ejemplares vivos para estudios de dinámica poblacional, migraciones, etc. (fig. 175).

Almacenamiento
Por ejemplo para codificar especímenes almacenados en triángulos (fig.176).

Creación de directorios
En ocasiones es necesario crear cientos de carpetas en un directorio, con un número, una secuencia, etc. La lectura dinámica de un código facilita mucho la tarea.

Binarización de imágenes
Si la base de datos es documental y/o posee una ruta del vínculo de la imagen a un directorio, la búsqueda de la especie por el ID puede generar por lanzamiento dinámico el dibujo binarizado vinculado al registro de dicha imagen (fig. 177-179).

Generar micro-etiquetas
Con un aplicativo se pueden generar etiquetas entomológicas de manera automática a partir de los datos biológicos, geográficos, climáticos, etc., que se han tomado de la búsqueda de un espécimen por su ID en la tabla de datos (figs. 66-68).

Secuenciación ADN
Un solo código puede contener una secuencia.

Lanzamiento de búsquedas en Internet
Por medio del ID se puede buscar la especie en Internet (fig. 180). De la misma manera, se ejecuta una búsqueda asociada a un libro escrito en una tabla de datos documental de bibliografía filtrada por el ISBN (véase anexo I, tabla XVII y anexo III, código XI, con unos ejemplos de lanzamiento de una aplicación externa).

Geografía
Tras decodificar el ID del espécimen encriptado en el código de una etiqueta y situarse en el registro correspondiente de la tabla de datos, se toma la localidad. Con eventos en cascada, a través de un archivo KMZ para GOOGLE EARTH se lanza el visor de ortofotografías y se coloca en la localidad tras ejecutar otro KMZ que muestra la coordenada UTM (en el anexo III, código XII: una parte del código del KMZ que muestra las coordenadas UTM 10 x 10 para la Península Ibérica).

Gestión de préstamos
En un museo, por ejemplo, se puede leer el código de un espécimen para saber si el mismo está en préstamo o donde queda momentáneamente depositado.

Se podría incluir una larga lista da aplicaciones, pero con los ejemplos descritos creemos que es suficiente. No nos cansaremos de insistir acerca de que la verdadera funcionalidad de la lectura de códigos reside en el lanzamiento dinámico de aplicativos, macros y fórmulas. No resulta muy laborioso programar estas funciones. Es totalmente rentable el esfuerzo y tiempo en comparación a los complejos y completos resultados que se pueden obtener, por ejemplo, en la elaboración de informes con complicados filtrados de tablas de datos entomológicas. Un alto nivel de sofisticación en una algoritmia inteligente, elegante y bien constituida puede llegar a proporcionar resultados tan sumamente elaborados, ricos, y con tal calado a la par que velocidad de procesamiento, que su desarrollo manual se tornaría tedioso e impracticable. En el anexo III código IX se ofrece la algoritmia completa para hacer el CODIFICADOR-DECODIFICADOR DE CÓDIGOS 2D (carpetas DATAMATRIX, QRCODE, AZTEC, DECODIFICADOR UNIVERSAL y DECODIFICADOR POR CÁMARA WEB) como el de EPHESIA. Para utilizar la algoritmia se deben crear los controles (fig. 106 y 107), generar los eventos, pegar el código brindado y vincular las librerías. Con las rutinas se pueden codificar AZTEC, DATAMATRIX, MICROQRCODE y QRCODE, y decodificar EAN8, EAN13, UPCA, CODE39, ITF, PDF417, CODABAR, MSI, PLESSEY, AZTEC, DATAMATRIX, MICROQRCODE y QRCODE.

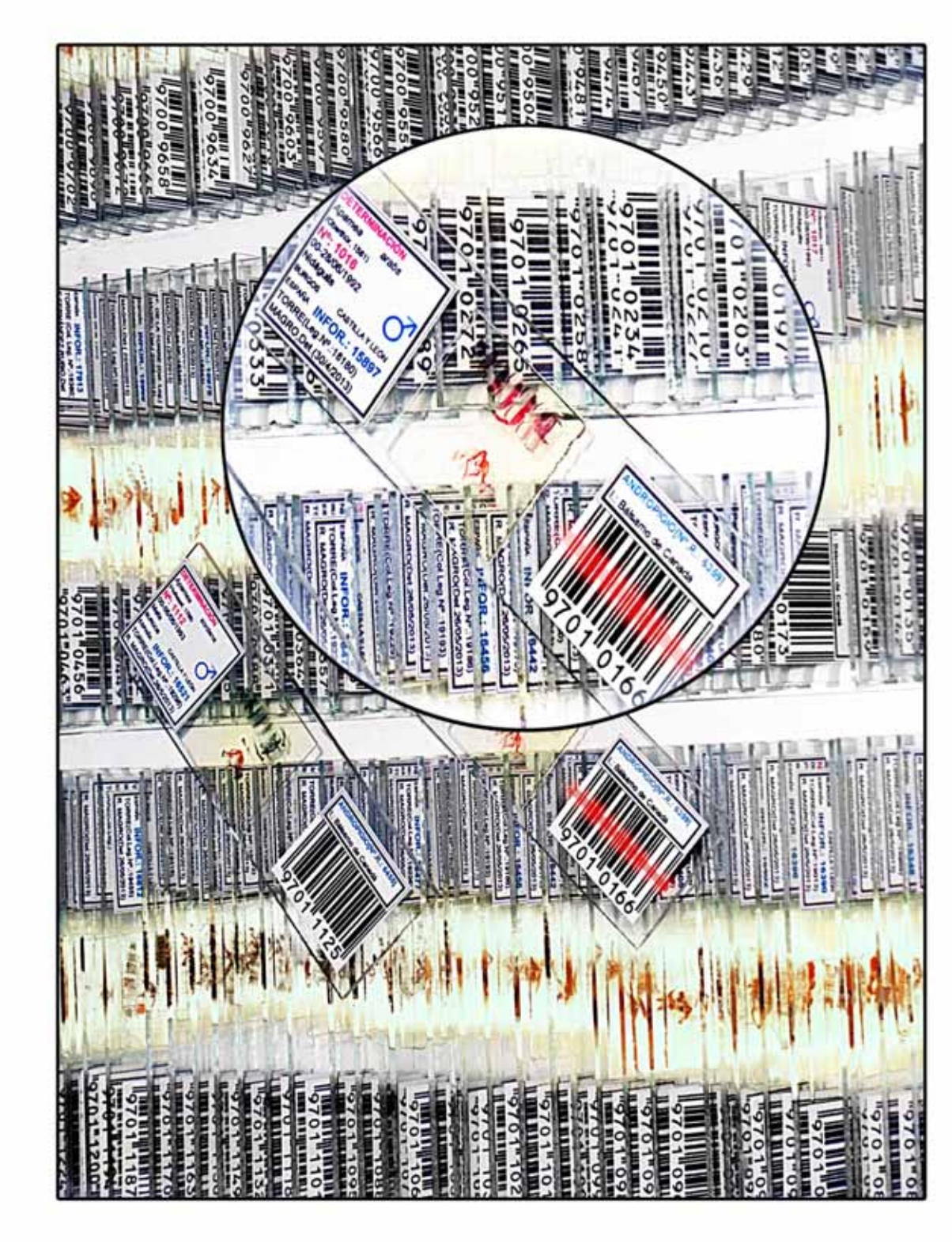


Figura 169: detalle de una caja de preparaciones microscópicas con estructuras genitales en portas con códigos EAN8. La imagen muestra el preciso instante de la lectura con el haz de barrido láser preparado para el eventual procesamiento de la decodificación numérica y su utilización inmediata en un aplicativo gestor con tablas de datos. El círculo delimita una zona ampliada.

Figura 170: detalle de una caja de frascos, con preparaciones microscópicas de vesicas evaginadas con la técnica del silopreno. Se han utilizado etiquetas con código AZTEC utilizando el sistema de calcas. Obsérvese que no se ha prescindido de la numeración.

Figura 171: dos falenas en exposición con datos en etiquetas QRCODE.

Figura 172- 173: arriba, cajas entomológicas con la información de su contenido en etiquetas con códigos AZTEC. Abajo, tarros con productos químicos con rótulos QRCODE.

Figura 174: lanzamiento dinámico de una tabla de datos documental con información bibliográfica a partir de la decodificación del lector del código ISBN de la publicación.

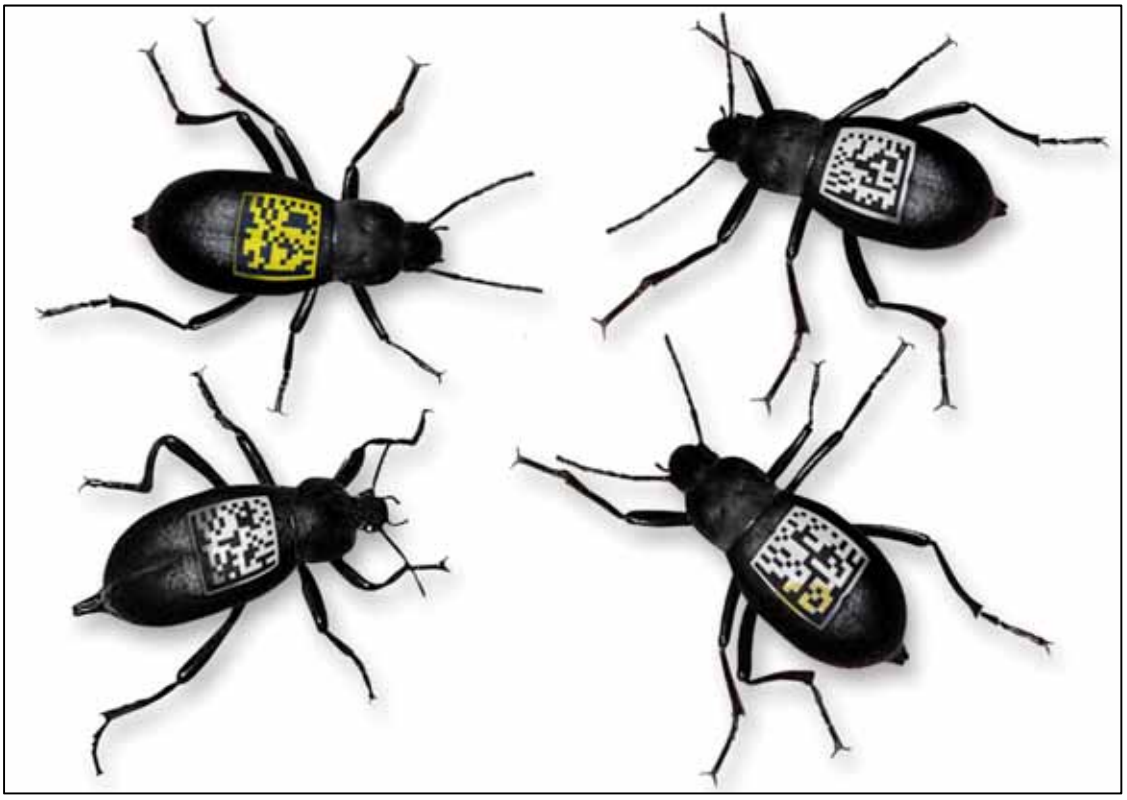


Figura 175: varios coleópteros etiquetados con DATAMATRIX en etiquetas indelebles, con datos y un número de identificación.

Figura 176: triángulos de almacenamiento con etiquetas UPCE.

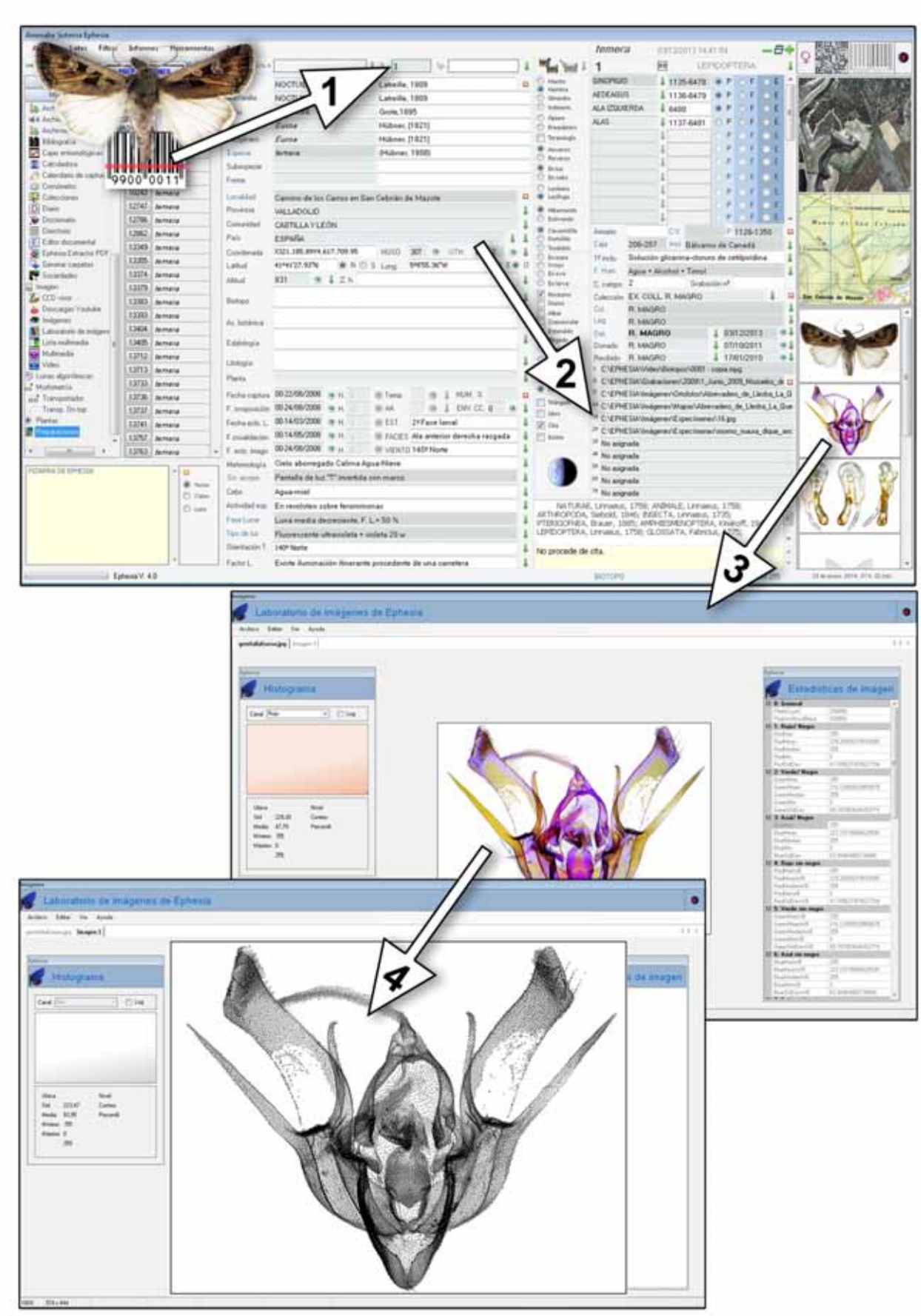


Figura 177- 179: 1: el lector lanza la búsqueda con el parámetro del ID de la especie. **2**: se toma la ruta del vínculo de la imagen. **3**: se abre el módulo “LABORATORIO DE IMÁGENES” y se pinta el gráfico. **4**: se realiza una binarización por desviación de píxel de Sauvola.

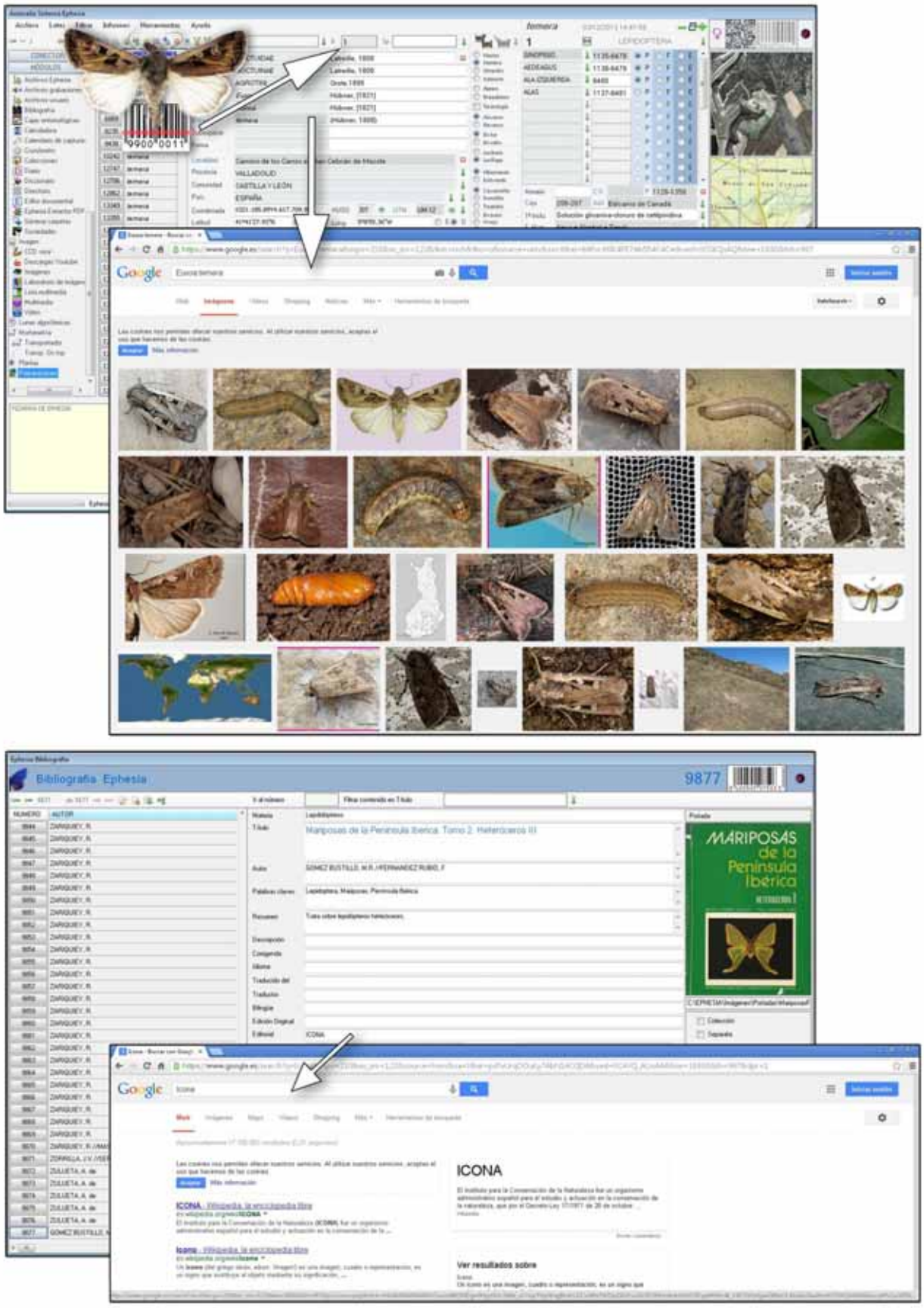


Figura 180- 183: arriba, lanzamiento dinámico de una búsqueda en Internet a partir del ID decodificado de una micro-etiqueta. Abajo, apertura de un buscador en Internet tras la decodificación del ISBN del libro. Se toma la información de un cuadro de texto de una tabla de datos bibliográfica vinculada a dicho código de barras.

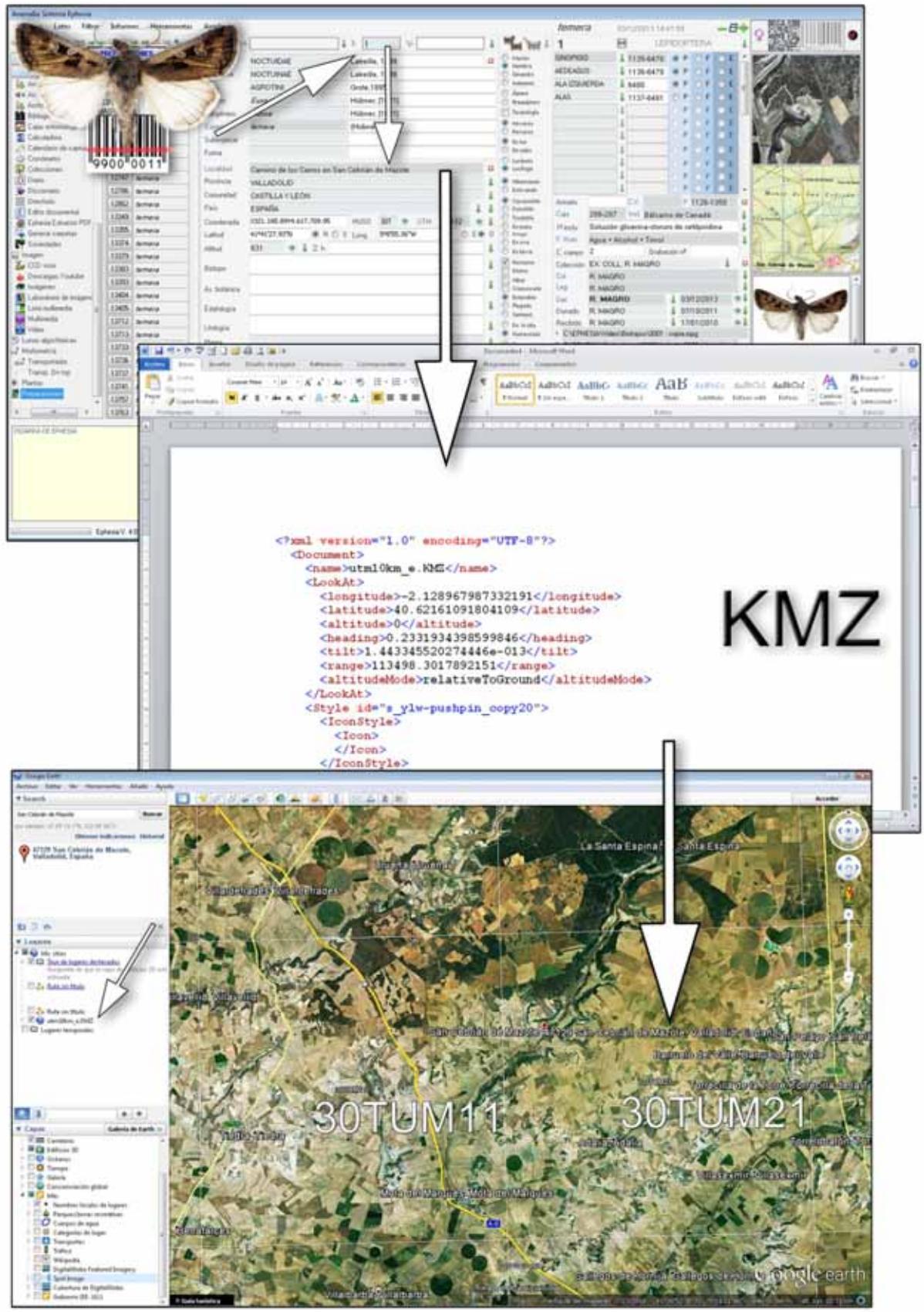


Figura 184- 186: lanzamiento dinámico en cascada de GOOGLE EARTH a partir del ID de una micro-etiqueta de un espécimen. Partiendo de EPHESIA (se puede hacer con WORD, EXCEL, ACCESS, etc.) se pasa la localidad como parámetro. Se ejecuta un KMZ que calcula la coordenada UTM y la muestra en el visor.

Agradecimiento

Nuestra más sincera gratitud para Adolfo Cordero Rivera, Santos Eizaguirre Orbe, Antonio Melic, Pedro Ramos Rodríguez, Loreto Sancho Casín y Tamar Zadí.

Bibliografía

BLAHUT, R. E., 1983. Theory and practice of error control codes. *Addison-Wesley Publishing Company, Reading*.

BOCK, F., STREIT, & S., TAUTZ, J. 2004. Automatic insect identification with RFID. First European *Conference of Apidology, Udine. University of Wuerzburg*: 60.

BULAN, O., MONGA, V. & SHARMA, G., 2009. High capacity color barcodes using dot orientation and color separability. Proc. SPIE IS&T *Electronic Imaging: Media Forensics and Security*, **7524-7254:** 17-71.

CHUAN CHONG C. & KHEE MENG, K., 1992. Principles and techniques in combinatorics, World *Scientific, Singapore*.

CLARK, G. C. CAIN, J. R. & J. B., 1981. Error-correction coding for digital communications, *Plenum Press, New York*.

CONSTANTINE, G. M. 1987. Combinatorial theory and statistical design, *John Willey & Sons, New York.*

ESPINOSA GARCÍA, F. J., HERNÁNDEZ ENCINAS, L. & MARTÍN DEL REY A. [Sin fecha]. Codificación de información mediante códigos bidimensionales. *Boletín de la Sociedad Española de Matemática Aplicada*: **0** (0000), 1-21.

FERNALD, R. D. 2006. Casting a genetic light on the evolution of eyes. *Science* 313, 5795, 1914–1918.

GRILLO, A., LENTINI, A., QUERINI M., ITALIANO, G.F., 2010. High Capacity Colored Two Dimensional *Codes Proc. International Multiconference on Computer Science and Information Technology* (IMCSIT): 709–716.

HARRISON, R. R., FOTOWAT, H., CHAN, R., KIER, R. J., LEONARDO, A., & GABBIANI, F., 2010. A wireless neural/EMG telemetry system for freely moving insects, to appear in: *Proceedings of the 2010 IEEE International Symposium on Circuits and Systems* (ISCAS 2010), Paris, France

HANKERSON, D., HOFFMAN, D. R., LEONARD, D. A., LINDNER, C. C., PHELPS, K. T., RODGER, C. A. & J. R. WALL, 2000. Coding theory and cryptography. The essentials, 2nd edition, Marcel Dekker, *Pure and Applied Mathematics*: 234.

HECHT, D. L., 2001a, Printed embedded data graphical user interfaces. *IEEE Computer*: 47–55.

HECHT, D. L., 2001b. Embedded data glyph technology for hardcopy digital documents. Proc. SPIE: *Color hard copy and graphic arts III*, J. Bares, ed., **2171:** 341–352.

KUTSCH, W., SCHWARZ, G., FISCHER, H., & KAUTZ, H. 1993. Wireless transmission of muscle potentials during free flight of a locust, J. *Exp. Biol.,* **185:** 367-373.

KUWANA, Y. ANDO, N. KANZAKI, R. & SHIMOYAMA, I., 1999- A radiotelemetry system for muscle potential recordings from freely flying insects, In: *Proc. BMES/EMBS Conf.*, **2:** 846.

MAGRO, R. 2008. Técnicas de tinción y corrección digital para preparaciones microscópicas en biología y entomología. *Boletín de la Sociedad Entomológica Aragonesa*, **43**: 525-548.

MAGRO, R. 2013. Binarización de imágenes digitales y su algoritmia como herramienta aplicada a la ilustración entomológica. *Boletín de la Sociedad Entomológica Aragonesa*, **53**: 443-464.

MATSUSHITA, N., HIHARA, D., USHIRO, T., YOSHIMURA, S., REKIMOTO, J., & YAMAMOTO, Y. 2003. ID CAM: A smart camera for scene capturing and id recognition. *In International Symposium on Mixed and Augmented Reality,* 227–236.

MOHAN, A., GRACE WOO, A, HIURAY, S. SMITHWICK, Q. & RASKAR, R., 2010. Bokode: Imperceptible Visual tags for Camera Based Interaction from a Distance. *Camera Culture Group, MIT Media Lab:* Sin numeración*,* 10 pp.

NIBLACK, W. 1986. *An Introduction to Image Processing*, Prentice-Hall, Englewood Cliffs, NJ: 115-116.

PARIKH, D. & JANCKE, G., 2008. Localization and Segmentation of a 2D High Capacity Color Barcode. Proc. 2008 IEEE Workshop on Applications of Computer Vision, *IEEE Computer Society*: 1–6.

PETERSON W. W. & WELDON, E. J., 1972. Error-correcting codes, *The MIT Press, 2nd edition, Cambridge*.

RASKAR, R., BEARDSLEY, P., VAN BAAR, J., WANG, Y., DIETZ, P., LEE, J., LEIGH, D., & WILLWACHER, T. 2004. RFIG Lamps: Interacting with a self-describing world via photosensing wireless tags and projectors. *In SIGGRAPH*, 406–415.

SATO, H., PEERI, Y., BAGHOOMIAN, E., BERRY, C. W., & MAHARBIZ, M. M., 2009. "Radio-controlled cyborg beetles: a radio-frequency system for insect neural flight control, in *Proceedings of IEEE International Conference on Micro Electro Mechanical System (MEMS 2009)*, Sorrento, 216–219.

SATO, H., BERRY, C. W., PEERI, Y., BAGHOOMIAN, E., CASEY, B. E., LAVELLA, G., VANDENBROOKS, J. M., HARRISON, J. F., AND MAHARBIZ, M. M. 2009. Remote radio control of insect flight. *Front. Integral Neuroscience.* 3:24. doi: 10.3389/neuro.07.024.2009.

SATO, H. & MAHARBIZ, M. M., 2010. Recent developments in the remote radio control of insect flight. *Frontiers in Neuroscience.* **4**:199. doi: 10.3389/fnins.2010.00199

SIONG, K. O., CHAI, D., & TAN, K. T., 2008. The use of border in colour 2D barcode. *Proc. 2008 International Symposium on Parallel and Distributed Processing with Applications (ISPA 2008)*: 999–1005.

STEEN, L. 1999. *Las Matemáticas en la vida cotidiana, 3ª edición*. Addison Wesley.

TREMBLAY, E. J., STACK, R. A., MORRISON, R. L., & FORD, J. E. 2007. Ultrathin cameras using annular folded optics. *Applied Optics* **46**, 4, 463–471.

WICKER, S. B. & BHARGAVA, V. K., 1994. Reed-Solomon codes and their applications, *IEEE Press, Piscataway, NJ.*

WIKELSKI, M., MOSKOWITZ, D., ADELMAN, J.S., COCHRAN, J., WILCOVE, D.S. & MAY, M.L. 2006. Simple rules guide dragonfly migration. *Biology Letters* **2**, 325-329

ZHANG, L., CURLESS, B., & SEITZ, S. M. 2002. Rapid shape acquisition using color structured light and multi-pass dynamic programming. *In IEEE 3DPVT*, 24–36.

ZHANG, X., FRONZ, S., & NAVAB, N. 2002. Visual marker detection and decoding in AR systems: A comparative study. *In ISMAR*, 97–106.

ZHANG, L., SUBRAMANIAM, N., LIN, R., RASKAR, R., AND NAYAR, S. 2008. Capturing images with sparse informational pixels using projected 3D tags. *In IEEE Virtual Reality.*

Glosario

Aberración de imagen: alteración o curvatura que se produce en una imagen cuando se la observa con una lupa o una cámara con objetivo gran angular, ojo de pez, etc.

Aberración sinusoidal: deformación de la imagen en forma de "s" o abanderada.

ACTIVEX: es un pequeño programa que aloja un control que se puede incrustar en una plataforma de programación o en un aplicativo, por ejemplo, un control generador de códigos de barras EAN para EXCEL.

AERO: modo gráfico de Windows en el cual hay que partes del sistema y de los aplicativos que se muestran semitransparentes, por ejemplo, las barras de cabecera.

Algoritmia: conjunto de algoritmos.

Algoritmo: función, procedimiento o fórmula que por medio de una rutina de programación y un intérprete de comandos realiza una o varias operaciones determinadas.

Anidamiento hexagonal: sistema de dibujado de pequeños hexágonos que utilizan algunos tipos de códigos de barras.

App: pequeños aplicativos, generalmente independientes de plataforma y sistema operativo. Muy utilizados en los teléfonos móviles.

ARAT: chip, por ejemplo RFID encapsulados en etiquetas con transpondedor y antena que son activos, es decir, trasmiten una señal.

ASCII: juego de códigos de caracteres.

BACKCOLOR: en el orden Z, color del fondo, por ejemplo la tonalidad del papel.

Base de datos documental: dícese de la base de datos que contiene tablas que almacenan como archivos, información documental de muy diversa índole, documentos de texto, imágenes y/o sonidos. La gestión de estas bases es lenta puesto que su tamaño suele ser muy grande.

Base de datos documental de vínculos: dícese de la base de datos que contiene tablas con información de los vínculos para acceder a la información documental. La gestión de estas bases es muy rápida dado que no almacenan ningún tipo de archivo, sólo la referencia.

BAT: es un tipo de archivo que contiene instrucciones en lotes para efectuar determinadas operaciones**.**

Binarización: proceso por el que una imagen de una matriz de píxeles con multitud de valores de luminosidad y color se trasfieren a otra matriz con información binaria en blanco y negro a través de un umbral de decisión.

Bitmap: imagen de mapa de bites.

BOOKODE: etiquetas emisoras de luz con micro-puntos ordenados de tal manera que almacenan información. Pueden ser decodificadas con cámaras fotográficas.

Bucle: forma de programar una función de manera que se repite hasta que una condición sea cierta o falsa. La condición se puede evaluar antes, mientras y después del lanzamiento del bucle.

Bucle infinito: función que por error no termina nunca hasta que el sistema se colapsa.

BWR: el tamaño mínimo y el reajuste de ancho de banda de un código.

Calca: tipo de plástico imprimible que permite trasvasar el material impreso con agua o un disolvente específico de manera que la impresión queda alojada en el soporte.

Check sum: dígito verificador.

Código de barras: un sistema de codificación- decodificación de cadenas numéricas, alfanuméricas y booleanas que utiliza barras y espacios para encriptar información.

Código de barras 1D: consiste en un gráfico de líneas verticales que suele además incluir un símbolo de entrada, de salida o de parada. Esta codificación puede descifrarse por un lector óptico o por un aplicativo.

Código de barras 2D: símbolo de muy diversas morfologías, cuadrado, rectangular, hexagonal, redondo, etc., en el que se encripta una información que puede ser decodificada por una cámara y/o un aplicativo.

Código propietario: el código de barras que no se puede usar libremente y está sujeto a patente. Evidentemente está disponible previo pago. El precio depende del tipo de licencia de uso y de la cantidad de códigos que incluya la librería o el control ACTIVEX.

Convolución: técnica de comparación de matrices de datos binarios que encapsulan una imagen de manera que se puede manipular, mejorar y modificar.

Control de usuario: cualquier control que se añade a un formulario en un entorno de programación, por ejemplo: colocar un botón en un documento de Word o un cuadro de texto en Excel.

Deformación de tonel (Pin Barrel): aberración en forma de tonel típica de algunas cámaras y monitores convencionales.

Desencriptar: operación para decodificar una cadena que está encriptada en otra o en un símbolo de forma que sea posible entenderla.

Dígito verificador: número que se codifica dentro de la información de un símbolo y que ayuda a su decodificación. Dependiendo del tipo de código se calcula de una manera u otra.

Dispositivos de lectura automática: aparatos que sirven para decodificar símbolos sin necesidad de introducción manual de datos.

DPI: puntos de impresión.

Driver: aplicativo que contiene la algoritmia, los parámetros, configuraciones, etc., que sirven para su comunicación con el sistema operativo y la interfaz de usuario.

Dll: archivo de código compilado que sirve para alojar una librería o una parte de un aplicativo o un control.

Encriptar: operación que se hace para esconder una cadena dentro de otra o en un símbolo, de forma que sea inteligible y que para su uso sea necesario descifrarla.

ENTER: tecla entrar del teclado QWERTY.

Esfericidad cóncava: aberración de una imagen que provoca un objetivo de mala calidad de una cámara. También puede presentarse en monitores TFT.

Esfericidad convexa: aberración de una imagen que provoca el objetivo angular de una cámara. También puede presentarse en monitores TFT.

Etiqueta de abeja: código que utilizan estructuras hexagonales.

Etiquetas entomológicas: micro-etiquetas que se colocan en los ejemplares de colección.

Eventos en cascada: señales de respuesta que genera un control de usuario ante cualquier eventualidad, por ejemplo, cuando se hace clic y se escribe en un control de texto se generan los siguiente eventos: ganancia de foco ► el control se ha pintado ►el control se ha validado ►el ratón ha entrado en el control ►el control ha recibido un clic del botón del ratón ►el botón del ratón ha pulsado hacia abajo ►el botón del ratón se ha levantado ►el texto de control ha cambiado ►la tecla se ha pulsado en el control ►la tecla se ha levantado en el control ►el cursor ha cambiado ►el cursor ha permanecido en el control cierto tiempo ►el control ha comprobado la cadena ►el control se ha validado ►el control ha perdido el foco. Estos eventos se puede capturar con condiciones y bucles, por ejemplo: si en el control ha recibido el código de la tecla RETURN, muestra la cadena “¡Hola mundo!”. Véase: Función.

Exe: tipo de extensión que suelen tener los aplicativos.

FORECOLOR: en el orden Z, color del frontal, por ejemplo la tonalidad de la fuente.

Fuente, de códigos: el código puede dibujarse en el control o utilizarse una fuente como si fuera un tipo de letra. Estas fuentes no suelen ser gratuitas.

Fuga, deformación: aberración que provoca que la imagen gane perspectiva en una o dos de sus dimensiones.

Función: un trozo de código que realiza una operación, por ejemplo: static saludo (string t) { t =+ “ mundo!”; return t; }, la función recibe la cadena “¡Hola” y devuelve el texto “¡Hola mundo!

Función recursiva: una función que se llama a sí misma hasta que se cumple una condición, por ejemplo: static sumaHasta () { i++; if (i <= 100) { sumaHasta }; return i;}

GB: unidad de medida de almacenamiento, Giga Bytes

HHLC: Hand Held Laser Compatible, luz láser para pistolas lectoras de códigos.

ID: siglas que se suelen utilizar para referirse al índice principal establecido en una tabla de datos y que por lo general, tiene relación con otros índices alojados en diferentes tablas. El uso de índices agiliza la manera en que los procedimientos acceden a las tablas relacionadas, por ejemplo, para la realización de una consulta de anexión.

Interfaz: medio que encapsula ciertas funcionalidades. Por ejemplo, interfaz gráfica: lo que ve el usuario de un aplicativo; interfaz de impresión: toda la codificación interna y los procesos necesarios para imprimir. Cada periférico tiene su propia interfaz.

INTRO: tecla entrar del teclado QWERTY. RETURN.

KMZ: se llama popularmente a unos archivos para GOOGLE EARTH que almacenan rutinas, coordenadas, ubicaciones, trazados, puntos, trayectorias aplicables a mapas y a ortofotografías.

Lanzamiento dinámico: forma de proceder que aprovecha la captura de un código de finalización del lector para generar eventos en cascada.

Lector láser: aparato que lanza un haz de luz que sirve para decodificar símbolos.

Logo, logotipo: pequeño dibujo o icono que se puede insertar en algunos modelos de códigos 2D.

Magnificación: alteración en el tamaño de un código.

Marca de agua: dibujo en color o semitransparente que se coloca encima o debajo de un gráfico para su adorno o especificar una autoría.

Matriz: conjunto de datos que se ordenan en columnas y líneas como si fuera una tabla. Puede ser de varias dimensiones.

Multi-hilo: manera de lanzar la algoritmia en diferentes hilos de procesos de manera que ambos sean independientes del gasto de procesador y de memoria y el aplicativo se ejecute de manera más eficaz. Para el control y prioridad de los mismos se utilizan semáforos de procesos.

Multimedia: dícese de los soportes o eventos que emplean todo tipo de recursos de comunicación; por ejemplo, un documento que posee texto, gráficos, imágenes en tres dimensiones y música.

ON RELESE: versión compilada y depurada independiente de un aplicativo. La opción contraria es ON DEBUG (en depuración).

Quiet zone: zona blanca de los laterales de un código de barras en los que en ocasiones se imprime un carácter de comienzo y finalización (<>) para facilitar la decodificación del símbolo por el lector.

Paridad: algunos códigos sólo admiten números pares y nunca vacíos por lo que la cadena en el caso de ser menor al tamaño requerido se rellena con ceros.

Pdf: documento de formato “portable” que puede ser de texto, imágenes e incluso multimedia.

Periféricos: dícese de los aparatos externos que están conectados al computador, por ejemplo el teclado.

Porta, porta-objetos: plaquita de cristal muy fino que se utiliza como base para inclusiones de materiales para efectuar su análisis bajo lupa o microscopio.

PRAT: chip, por ejemplo RFID encapsulados en etiquetas con transpondedor y antena que son pasivos.

Randomize: generación de semillas aleatorias, por ejemplo de un número o de un carácter, de manera que la cadena se componga de una serie estocástica.

Recursividad: manera de proceder en las llamadas a una rutina que consiste en realizarlas desde y para sí misma hasta que se cumple una condición. Si no se establece la condición el procedimiento entra en un bucle infinito. Véase: Función recursiva.

Redundancia alta: código que posee una información para su recuperación de nivel máximo.

Redundancia baja: código que posee una información para su recuperación de nivel pequeña.

Redundancia, en el código: información adicional (Back-up) que se almacena dentro del código para la eventual lectura y recuperación en el caso de deterioros o pérdida de información.

Redundancia normal: código que posee una información para su recuperación de nivel medio.

Reed Solomon, algoritmo de: función matemática aplicada a la algoritmia para el procesado de valores redundantes a modo de copia de seguridad codificada en el cuerpo del símbolo (anexo III, código VI).

Retorno de carro: en las antiguas máquinas de escribir operación de pasar del final de una línea al principio de la siguiente. Los lenguajes de programación suelen poseer retorno de carro automatizado sensible al contexto pero por lo general los nombres de parámetros y comandos no se pueden partir. Para que el intérprete de comandos entienda que se trata de un retorno de carro cada lenguaje utiliza un símbolo especial. Por ejemplo “Hola” + “ ” + “mundo.” = “Hola mundo” (en C#). Si en este caso se omite el símbolo “+” el compilador arroja una excepción en tiempo de ejecución.

RFID: chip Radio Frequency Identification, conocido con diversos nombres: MIFARE, HITANG, I-CODE, U-CODE, ATMELRFID, HTKRFID.

RETURN: tecla entrar del teclado QWERTY. INTRO.

Sangría: tabulación del texto.

Símbolo > y <: carácter “mayor que” y “menor que” situado en algunos códigos de barras. Su utilidad reside en la identificación de espacios blancos al principio y al final de las líneas y /o como cierre y apertura para facilitar la lectura por el aparato decodificador.

Singulación: algoritmo que discrimina múltiples recepciones de ondas de radio frecuencia por medio de árboles analíticos.

SyntaxHighlighting: sistema que utilizan los editores de código para resaltar con diferentes colores las palabras, órdenes, funciones y otros parámetros de un lenguaje de programación determinado. Esto sucede, normalmente, mientras se escribe. Cada tipo de lenguaje tiene una codificación de color única establecida por su creador.

TAB: tecla de tabular del teclado QWERTY.

Tabla de datos: cada una de las secciones en que se divide una base de datos, por ejemplo: “bases de datos LEPIDOPTEROLOGIA” contiene tablas “LOCALIDADES”, “SISTEMÁTICA”, “BIBLIOGRAFÍA, etc. Cada una de estas tablas suele estar relacionada por medio de un ID.

Tabulación: sangría del texto, establecer un espacio en blanco antes del mismo.

TB: unidad de medida de almacenamiento, Tera Bytes.

Tramado, deformación de: mala resolución que genera una trama en una imagen o sector que dificulta la lectura de un código.

Transpondedor: dispositivo utilizado en un gran número de aparatos electrónicos cuyo nombre se deriva del ingés (Transmitter + Responder). Se conocen varias siglas para denominar el aparato: XPRD, XPNDR, TPDR y TP. El circuito acoge, amplifica y/o re-emite en una banda distinta de la señal recibida con una respuesta normalmente automática.

URL: dirección Web.

Variable: es un espacio de memoria con muy diversos formatos y tipos para almacenamiento fugaz de información

Vesica evaginada: el interior del a*edeagus* o pene de un insecto expandido o eviscerado.

Threads: multi-hilo o proceso en hilos independientes.

WiFi: una manera de interactuar de forma inalámbrica con el computador, un teléfono, tableta, etc.

Anexo I

Tablas

TABLA I

	AZTEC				
	Tipo	**Pref. estático**	**Milisegundos ***	**Prefijo aleatorio**	**Milisegundos ***
1	Número	3748	153	3748	161
2	Carácter	2999	182	± 2157	213

Tabla I: caracteres que puede contener el código AZTEC. Los prefijos estáticos de letra se generaron con el carácter 'A' y los numéricos con el dígito '9'.

TABLA II

DATAMATRIX (Ascii y C40)											
A	**B**	**C**	**D**	**E**	**F**	**A**	**B**	**C**	**D**	**E**	**F**
Ascii	10 x 10	4	10	5	04	C40	10 x 10	4	10	2	03
Ascii	12 x 12	4	10	9	03	C40	12 x 12	4	10	5	05
Ascii	14 x 14	4	10	15	07	C40	14 x 14	4	10	9	06
Ascii	16 x 16	4	10	23	05	C40	16 x 16	4	10	15	08
Ascii	18 x 18	4	10	35	07	C40	18 x 18	4	10	24	11
Ascii	20 x 20	4	10	43	09	C40	20 x 20	4	10	30	16
Ascii	22 x 22	4	10	59	12	C40	22 x 22	4	10	42	19
Ascii	24 x 24	4	10	71	13	C40	24 x 24	4	10	51	20
Ascii	26 x 26	4	10	87	15	C40	26 x 26	4	10	63	22
Ascii	32 x 32	4	10	123	20	C40	32 x 32	4	10	90	33
Ascii	36 x 36	4	10	171	24	C40	36 x 36	4	10	126	50
Ascii	40 x 40	4	10	227	31	C40	40 x 40	4	10	168	52
Ascii	44 x 44	4	10	287	37	C40	44 x 44	4	10	213	62
Ascii	48 x 48	4	10	347	43	C40	48 x 48	4	10	258	81
Ascii	52 x 52	4	10	407	49	C40	52 x 52	4	10	303	87
Ascii	64 x 64	4	10	559	76	C40	64 x 64	4	10	417	163
Ascii	72 x 72	4	10	735	95	C40	72 x 72	4	10	549	210
Ascii	80 x 80	4	10	911	129	C40	80 x 80	4	10	681	233
Ascii	88 x 88	4	10	1151	157	C40	88 x 88	4	10	861	303
Ascii	96 x 96	4	10	1391	172	C40	96 x 96	4	10	1041	324
Ascii	104 x	4	10	1631	201	C40	104 x 104	4	10	1221	376
Ascii	120 x	4	10	2099	269	C40	120 x 120	4	10	1572	463
Ascii	132 x	4	10	2607	601	C40	132 x 132	4	10	1953	680
Ascii	144 x	4	10	2844	793	C40	144 x 144	4	10	2130	793
Ascii	8 x 18	4	10	9	07	C40	8 x 18	4	10	5	05
Ascii	8 x 32	4	10	19	08	C40	8 x 32	4	10	12	13
Ascii	12 x 26	4	10	31	10	C40	12 x 26	4	10	21	17
Ascii	12 x 36	4	10	43	29	C40	12 x 36	4	10	30	21
Ascii	16 x 36	4	10	63	36	C40	16 x 36	4	10	45	31
Ascii	16 x 48	4	10	97	45	C40	16 x 48	4	10	71	35

Tabla II: parámetros y número de dígitos que pueden contener algunos tipos de códigos DATAMATRIX (Ascii y C40). **A** = modo; **B** = tamaño; **C** = píxel; **D** = margen; **E** = caracteres; **F** = milisegundos que consume la generación del código (*).

TABLA III

DATAMATRIX (Text y X12)											
A	**B**	**C**	**D**	**E**	**F**	**A**	**B**	**C**	**D**	**E**	**F**
Text	10 x 10	4	10	2	02	X12	10 x 10	4	10	3	02
Text	12 x 12	4	10	5	03	X12	12 x 12	4	10	6	03
Text	14 x 14	4	10	9	04	X12	14 x 14	4	10	10	04
Text	16 x 16	4	10	15	05	X12	16 x 16	4	10	16	05
Text	18 x 18	4	10	24	07	X12	18 x 18	4	10	25	06
Text	20 x 20	4	10	30	08	X12	20 x 20	4	10	31	07
Text	22 x 22	4	10	42	09	X12	22 x 22	4	10	43	09
Text	24 x 24	4	10	51	11	X12	24 x 24	4	10	52	15
Text	26 x 26	4	10	63	19	X12	26 x 26	4	10	64	19
Text	32 x 32	4	10	90	21	X12	32 x 32	4	10	91	22
Text	36 x 36	4	10	126	27	X12	36 x 36	4	10	127	24
Text	40 x 40	4	10	168	40	X12	40 x 40	4	10	169	30
Text	44 x 44	4	10	213	53	X12	44 x 44	4	10	214	36
Text	48 x 48	4	10	258	54	X12	48 x 48	4	10	259	76
Text	52 x 52	4	10	303	70	X12	52 x 52	4	10	304	96
Text	64 x 64	4	10	417	77	X12	64 x 64	4	10	418	106
Text	72 x 72	4	10	549	96	X12	72 x 72	4	10	550	133
Text	80 x 80	4	10	681	119	X12	80 x 80	4	10	682	158
Text	88 x 88	4	10	861	198	X12	88 x 88	4	10	862	166
Text	96 x 96	4	10	1041	208	X12	96 x 96	4	10	1042	172
Text	104 x	4	10	1221	247	X12	104 x 104	4	10	1222	201
Text	120 x	4	10	1572	268	X12	120 x 120	4	10	1573	268
Text	132 x	4	10	1953	315	X12	132 x 132	4	10	1954	370
Text	144 x	4	10	2130	325	X12	144 x 144	4	10	2132	399
Text	8 x 18	4	10	5	03	X12	8 x 18	4	10	6	03
Text	8 x 32	4	10	12	05	X12	8 x 32	4	10	13	05
Text	12 x 26	4	10	21	06	X12	12 x 26	4	10	22	06
Text	12 x 36	4	10	30	08	X12	12 x 36	4	10	31	08
Text	16 x 36	4	10	45	14	X12	16 x 36	4	10	46	14
Text	16 x 48	4	10	71	16	X12	16 x 48	4	10	72	22

Tabla III: parámetros y número de dígitos que pueden contener algunos tipos de códigos DATAMATRIX (Text y X12). **A** = modo; **B** = tamaño; **C** = píxel; **D** = margen; **E** = caracteres; **F** = milisegundos que consume la generación del código (*).

TABLA IV

DATAMATRIX (Edifact y Base256)											
A	**B**	**C**	**D**	**E**	**F**	**A**	**B**	**C**	**D**	**E**	**F**
Edifact	10 x 10	4	10	1	02	Base256	10 x 10	4	10	1	02
Edifact	12 x 12	4	10	5	03	Base256	12 x 12	4	10	3	04
Edifact	14 x 14	4	10	9	04	Base256	14 x 14	4	10	6	05
Edifact	16 x 16	4	10	14	05	Base256	16 x 16	4	10	10	07
Edifact	18 x 18	4	10	22	06	Base256	18 x 18	4	10	16	10
Edifact	20 x 20	4	10	27	07	Base256	20 x 20	4	10	20	12
Edifact	22 x 22	4	10	38	11	Base256	22 x 22	4	10	28	14
Edifact	24 x 24	4	10	46	18	Base256	24 x 24	4	10	34	15
Edifact	26 x 26	4	10	57	21	Base256	26 x 26	4	10	42	20
Edifact	32 x 32	4	10	81	23	Base256	32 x 32	4	10	60	23
Edifact	36 x 36	4	10	113	25	Base256	36 x 36	4	10	84	26
Edifact	40 x 40	4	10	150	30	Base256	40 x 40	4	10	112	30
Edifact	44 x 44	4	10	190	36	Base256	44 x 44	4	10	142	36
Edifact	48 x 48	4	10	230	43	Base256	48 x 48	4	10	172	44
Edifact	52 x 52	4	10	270	50	Base256	52 x 52	4	10	202	83
Edifact	64 x 64	4	10	371	106	Base256	64 x 64	4	10	277	96
Edifact	72 x 72	4	10	489	120	Base256	72 x 72	4	10	365	106
Edifact	80 x 80	4	10	606	149	Base256	80 x 80	4	10	453	118
Edifact	88 x 88	4	10	766	167	Base256	88 x 88	4	10	573	143
Edifact	96 x 96	4	10	926	203	Base256	96 x 96	4	10	693	173
Edifact	104 x	4	10	1086	227	Base256	104 x 104	4	10	813	204
Edifact	120 x	4	10	1398	319	Base256	120 x 120	4	10	1047	265
Edifact	132 x	4	10	1737	325	Base256	132 x 132	4	10	1301	389
Edifact	144 x	4	10	1894	385	Base256	144 x 144	4	10	1419	730
Edifact	8 x 18	4	10	5	03	Base256	8 x 18	4	10	3	03
Edifact	8 x 32	4	10	12	05	Base256	8 x 32	4	10	8	05
Edifact	12 x 26	4	10	20	06	Base256	12 x 26	4	10	14	07
Edifact	12 x 36	4	10	27	11	Base256	12 x 36	4	10	20	08
Edifact	16 x 36	4	10	40	13	Base256	16 x 36	4	10	30	11
Edifact	16 x 48	4	10	63	16	Base256	16 x 48	4	10	47	14

Tabla IV: parámetros y número de dígitos que pueden contener algunos tipos de códigos DATAMATRIX (Edifact y Base256). **A** = modo; **B** = tamaño; **C** = píxel; **D** = margen; **E** = caracteres; **F** = milisegundos que consume la generación del código (*).

TABLA V

DATAMATRIX (AsciiGS1)											
AsciiGS1	10 x 10	4	10	3	02	AsciiGS1	96 x 96	4	10	1389	171
AsciiGS1	12 x 12	4	10	7	03	AsciiGS1	104 x 104	4	10	1629	202
AsciiGS1	14 x 14	4	10	13	05	AsciiGS1	120 x 120	4	10	2097	272
AsciiGS1	16 x 16	4	10	21	06	AsciiGS1	132 x 132	4	10	2605	363
AsciiGS1	18 x 18	4	10	33	07	AsciiGS1	144 x 144	4	10	2841	832
AsciiGS1	20 x 20	4	10	41	08	AsciiGS1	8 x 18	4	10	7	03
AsciiGS1	22 x 22	4	10	57	09	AsciiGS1	8 x 32	4	10	17	05
AsciiGS1	24 x 24	4	10	69	11	AsciiGS1	12 x 26	4	10	29	06
AsciiGS1	26 x 26	4	10	85	13	AsciiGS1	12 x 36	4	10	41	08
AsciiGS1	32 x 32	4	10	121	19	AsciiGS1	16 x 36	4	10	61	11
AsciiGS1	36 x 36	4	10	169	37	AsciiGS1	16 x 48	4	10	95	14
AsciiGS1	40 x 40	4	10	225	40	AsciiGS1	96 x 96	4	10	1389	171
AsciiGS1	44 x 44	4	10	285	44	AsciiGS1	104 x 104	4	10	1629	202
AsciiGS1	48 x 48	4	10	345	48	AsciiGS1	120 x 120	4	10	2097	272
AsciiGS1	52 x 52	4	10	405	50	AsciiGS1	132 x 132	4	10	2605	363
AsciiGS1	64 x 64	4	10	557	77	AsciiGS1	144 x 144	4	10	2841	832
AsciiGS1	72 x 72	4	10	733	96	AsciiGS1	8 x 18	4	10	7	03
AsciiGS1	80 x 80	4	10	909	118	AsciiGS1	8 x 32	4	10	17	05
AsciiGS1	88 x 88	4	10	1149	133	AsciiGS1	12 x 26	4	10	29	06

Tabla V: parámetros y número de dígitos que pueden contener algunos tipos de códigos DATAMATRIX (AsciiGS1). **A** = modo; **B** = tamaño; **C** = píxel; **D** = margen; **E** = caracteres; **F** = milisegundos que consume la generación del código (*).

TABLA VI

1D				
Tipo	**Mínima altura**	**Mínima anchura**	**Máxima altura**	**Máxima anchura**
EAN13	7, 9 mm	16,3	79 mm	116 mm
EAN8	6,5 mm	11,5	66 mm	115 mm

Tabla VI: tamaño mínimo y máximo de los códigos de barras 1D. La dimensión menor está en el límite de miniaturización, la impresión debe realizarse a máxima resolución y en papel fotográfico. La lectura tiene que realizarse con un escáner adecuado a esta resolución, consulte la ficha técnica del mismo.

TABLA VII

Factor *	Ancho ideal del módulo	Dimensiones EAN13		Dimensiones EAN8	
		Anchura	Altura	Anchura	Altura
0.47	0.155	17.54	12.19	12.57	10.02
0.60	0.222	24.85	17.27	17.81	14.02
0.80	0.264	29.83	20.73	21.38	17.05
0.85	0.281	31.70	22.02	22.72	18.11
0.90	0.297	33.56	23.32	24.06	19.18
0.95	0.313	35.43	24.61	25.39	20.24
1.00	0.330	37.29	25.91	26.73	21.31
1.05	0.346	39.15	27.21	28.07	22.38
1.10	0.363	41.02	28.50	29.40	23.44
1.15	0.379	42.88	29.80	30.74	24.51
1.20	0.396	44.75	31.09	32.08	25.57
1.25	0.412	46.61	32.39	33.41	26.64
1.30	0.429	48.48	33.68	34.75	27.70
1.35	0.445	50.34	34.98	36.09	28.77
1.40	0.462	52.21	36.27	37.42	29.83
1.45	0.478	54.07	37.57	38.76	30.90
1.50	0.495	55.94	38.87	40.10	31.97
1.55	0.511	57.80	40.16	41.43	33.03
1.60	0.528	59.66	41.46	42.77	34.10
1.65	0.544	61.53	42.75	44.10	35.16
1.70	0.561	63.39	44.05	45.44	36.23
1.75	0.577	65.26	45.34	46.78	37.29
1.80	0.594	67.12	46.64	48.11	38.36
1.85	0.610	68.99	47.93	49.45	39.42
1.90	0.627	70.85	49.23	50.79	40.49
1.95	0.643	72.72	50.52	52.12	41.55
2.00	0.660	74.58	51.82	53.46	42.62

Tabla VII: en la tabla se muestran los valores de tamaño y factor de magnificación en EAN8 y 13 (válidos también para UPC, GTIN, etc.). Éstos deberán estar entre los formatos estándar que aseguran que el código podrá decodificarse en cualquier lector o escáner del mundo. Los números están expresados en milímetros. No se incluyen factores de truncamiento. Incluso respetando las proporciones, para poder utilizar tamaños por debajo de 10.02 mm de anchura es indispensable informarse adecuadamente de las capacidades de lectura y las dimensiones de mínimas de los símbolos que la pistola admite y puede decodificar. Muy pocos lectores pueden acometer con éxito lecturas de símbolos tan diminutos. Con respecto a la decodificación por medio de aplicativos el límite de miniaturización está directamente relacionado con la calidad de impresión del símbolo y la resolución de lectura del escáner. Por lo general, ninguna impresora puede imprimir en la actualidad un código por debajo de los 0,5 mm con garantías de poderse escanear y decodificar.

TABLA VIII

QRCODE (byte con redundancia leve y normal)											
A	B	C	D	E	F	A	B	C	D	E	F
1	Leve	Byte	4	17	01	1	Normal	Byte	4	14	01
2	Leve	Byte	4	32	02	2	Normal	Byte	4	26	02
3	Leve	Byte	4	53	03	3	Normal	Byte	4	42	03
4	Leve	Byte	4	78	04	4	Normal	Byte	4	62	04
5	Leve	Byte	4	106	05	5	Normal	Byte	4	84	05
6	Leve	Byte	4	134	06	6	Normal	Byte	4	106	06
7	Leve	Byte	4	154	07	7	Normal	Byte	4	122	07
8	Leve	Byte	4	192	08	8	Normal	Byte	4	152	08
9	Leve	Byte	4	230	09	9	Normal	Byte	4	180	09
10	Leve	Byte	4	271	11	10	Normal	Byte	4	213	11
11	Leve	Byte	4	321	12	11	Normal	Byte	4	251	12
12	Leve	Byte	4	367	18	12	Normal	Byte	4	287	13
13	Leve	Byte	4	425	22	13	Normal	Byte	4	331	15
14	Leve	Byte	4	458	26	14	Normal	Byte	4	362	19
15	Leve	Byte	4	520	29	15	Normal	Byte	4	412	20
16	Leve	Byte	4	586	30	16	Normal	Byte	4	450	21
17	Leve	Byte	4	644	31	17	Normal	Byte	4	504	22
18	Leve	Byte	4	718	32	18	Normal	Byte	4	560	23
19	Leve	Byte	4	792	34	19	Normal	Byte	4	624	26
20	Leve	Byte	4	858	36	20	Normal	Byte	4	666	31
21	Leve	Byte	4	929	39	21	Normal	Byte	4	711	33
22	Leve	Byte	4	1003	41	22	Normal	Byte	4	779	35
23	Leve	Byte	4	1091	43	23	Normal	Byte	4	857	36
24	Leve	Byte	4	1171	46	24	Normal	Byte	4	911	37
25	Leve	Byte	4	1273	49	25	Normal	Byte	4	997	38
26	Leve	Byte	4	1367	52	26	Normal	Byte	4	1059	40
27	Leve	Byte	4	1465	54	27	Normal	Byte	4	1125	43
28	Leve	Byte	4	1528	58	28	Normal	Byte	4	1190	48
29	Leve	Byte	4	1628	60	29	Normal	Byte	4	1264	49
30	Leve	Byte	4	1732	63	30	Normal	Byte	4	1370	52
31	Leve	Byte	4	1840	66	31	Normal	Byte	4	1452	55
32	Leve	Byte	4	1952	69	32	Normal	Byte	4	1538	58
33	Leve	Byte	4	2068	73	33	Normal	Byte	4	1628	66
34	Leve	Byte	4	2188	71	34	Normal	Byte	4	1722	67
35	Leve	Byte	4	2303	79	35	Normal	Byte	4	1809	69
36	Leve	Byte	4	2431	89	36	Normal	Byte	4	1911	93
37	Leve	Byte	4	2563	98	37	Normal	Byte	4	1989	98
38	Leve	Byte	4	2699	128	38	Normal	Byte	4	2099	104
39	Leve	Byte	4	2809	138	39	Normal	Byte	4	2213	110
40	Leve	Byte	4	2953	88	40	Normal	Byte	4	2331	86

Tabla VIII: tipo de codificación byte**,** parámetros y número de dígitos que pueden contener algunos tipos de códigos QRCODE (redundancia leve y normal). **A** = versión; **B** = niveles de corrección; **C** = tipo; **D** = tamaño; **E** = caracteres; **F** = milisegundos que consume la generación del código (*).

TABLA IX

QRCODE (byte con redundancia mediana y grande)											
A	B	C	D	E	F	A	B	C	D	E	F
1	Mediana	Byte	4	11	01	1	Grande	Byte	4	7	01
2	Mediana	Byte	4	20	02	2	Grande	Byte	4	14	02
3	Mediana	Byte	4	32	03	3	Grande	Byte	4	24	03
4	Mediana	Byte	4	46	04	4	Grande	Byte	4	34	04
5	Mediana	Byte	4	60	05	5	Grande	Byte	4	44	05
6	Mediana	Byte	4	74	06	6	Grande	Byte	4	58	06
7	Mediana	Byte	4	86	07	7	Grande	Byte	4	64	07
8	Mediana	Byte	4	108	08	8	Grande	Byte	4	84	08
9	Mediana	Byte	4	130	09	9	Grande	Byte	4	98	09
10	Mediana	Byte	4	151	11	10	Grande	Byte	4	119	10
11	Mediana	Byte	4	177	18	11	Grande	Byte	4	137	11
12	Mediana	Byte	4	203	19	12	Grande	Byte	4	155	12
13	Mediana	Byte	4	241	20	13	Grande	Byte	4	177	13
14	Mediana	Byte	4	258	21	14	Grande	Byte	4	194	14
15	Mediana	Byte	4	292	22	15	Grande	Byte	4	220	15
16	Mediana	Byte	4	322	23	16	Grande	Byte	4	250	18
17	Mediana	Byte	4	364	24	17	Grande	Byte	4	280	19
18	Mediana	Byte	4	394	25	18	Grande	Byte	4	310	21
19	Mediana	Byte	4	442	26	19	Grande	Byte	4	338	23
20	Mediana	Byte	4	482	27	20	Grande	Byte	4	382	25
21	Mediana	Byte	4	509	28	21	Grande	Byte	4	403	27
22	Mediana	Byte	4	565	30	22	Grande	Byte	4	439	29
23	Mediana	Byte	4	857	33	23	Grande	Byte	4	461	33
24	Mediana	Byte	4	661	35	24	Grande	Byte	4	511	35
25	Mediana	Byte	4	715	37	25	Grande	Byte	4	535	37
26	Mediana	Byte	4	751	40	26	Grande	Byte	4	593	38
27	Mediana	Byte	4	805	42	27	Grande	Byte	4	625	42
28	Mediana	Byte	4	868	46	28	Grande	Byte	4	658	49
29	Mediana	Byte	4	908	48	29	Grande	Byte	4	698	50
30	Mediana	Byte	4	982	51	30	Grande	Byte	4	742	52
31	Mediana	Byte	4	1030	53	31	Grande	Byte	4	790	55
32	Mediana	Byte	4	1112	57	32	Grande	Byte	4	842	57
33	Mediana	Byte	4	1168	60	33	Grande	Byte	4	898	59
34	Mediana	Byte	4	1228	65	34	Grande	Byte	4	958	62
35	Mediana	Byte	4	1283	67	35	Grande	Byte	4	983	82
36	Mediana	Byte	4	1351	71	36	Grande	Byte	4	1051	90
37	Mediana	Byte	4	1423	75	37	Grande	Byte	4	1093	102
38	Mediana	Byte	4	1499	76	38	Grande	Byte	4	1139	104
39	Mediana	Byte	4	1579	80	39	Grande	Byte	4	1219	120
40	Mediana	Byte	4	1663	95	40	Grande	Byte	4	1273	137

Tabla IX: tipo de codificación byte**,** parámetros y número de dígitos que pueden contener algunos tipos de códigos QRCODE (redundancia mediana y grande). **A** = versión; **B** = niveles de corrección; **C** = tipo; **D** = tamaño; **E** = caracteres; **F** = milisegundos que consume la generación del código (*).

TABLA X

QRCODE (alfanumérico redundancia leve y normal)											
A	B	C	D	E	F	A	B	C	D	E	F
1	Leve	Alfanumérico	4	25	01	1	Normal	Alfanumérico	4	20	02
2	Leve	Alfanumérico	4	47	02	2	Normal	Alfanumérico	4	38	03
3	Leve	Alfanumérico	4	77	03	3	Normal	Alfanumérico	4	61	04
4	Leve	Alfanumérico	4	114	04	4	Normal	Alfanumérico	4	90	08
5	Leve	Alfanumérico	4	154	05	5	Normal	Alfanumérico	4	122	09
6	Leve	Alfanumérico	4	195	06	6	Normal	Alfanumérico	4	154	10
7	Leve	Alfanumérico	4	224	08	7	Normal	Alfanumérico	4	178	12
8	Leve	Alfanumérico	4	279	09	8	Normal	Alfanumérico	4	221	13
9	Leve	Alfanumérico	4	335	10	9	Normal	Alfanumérico	4	262	14
10	Leve	Alfanumérico	4	395	11	10	Normal	Alfanumérico	4	311	15
11	Leve	Alfanumérico	4	468	15	11	Normal	Alfanumérico	4	366	16
12	Leve	Alfanumérico	4	535	17	12	Normal	Alfanumérico	4	419	17
13	Leve	Alfanumérico	4	619	23	13	Normal	Alfanumérico	4	483	18
14	Leve	Alfanumérico	4	667	25	14	Normal	Alfanumérico	4	528	19
15	Leve	Alfanumérico	4	758	26	15	Normal	Alfanumérico	4	600	21
16	Leve	Alfanumérico	4	854	28	16	Normal	Alfanumérico	4	656	22
17	Leve	Alfanumérico	4	938	30	17	Normal	Alfanumérico	4	734	25
18	Leve	Alfanumérico	4	1046	31	18	Normal	Alfanumérico	4	816	26
19	Leve	Alfanumérico	4	1153	33	19	Normal	Alfanumérico	4	909	27
20	Leve	Alfanumérico	4	1249	35	20	Normal	Alfanumérico	4	970	28
21	Leve	Alfanumérico	4	1352	37	21	Normal	Alfanumérico	4	1035	29
22	Leve	Alfanumérico	4	1460	38	22	Normal	Alfanumérico	4	1134	30
23	Leve	Alfanumérico	4	1588	39	23	Normal	Alfanumérico	4	1248	34
24	Leve	Alfanumérico	4	1704	40	24	Normal	Alfanumérico	4	1326	35
25	Leve	Alfanumérico	4	1853	41	25	Normal	Alfanumérico	4	1451	37
26	Leve	Alfanumérico	4	1990	42	26	Normal	Alfanumérico	4	1542	40
27	Leve	Alfanumérico	4	2132	45	27	Normal	Alfanumérico	4	1637	49
28	Leve	Alfanumérico	4	2223	48	28	Normal	Alfanumérico	4	1732	51
29	Leve	Alfanumérico	4	2369	51	29	Normal	Alfanumérico	4	1839	53
30	Leve	Alfanumérico	4	2520	55	30	Normal	Alfanumérico	4	1994	54
31	Leve	Alfanumérico	4	2677	61	31	Normal	Alfanumérico	4	2113	56
32	Leve	Alfanumérico	4	2840	65	32	Normal	Alfanumérico	4	2238	58
33	Leve	Alfanumérico	4	3009	68	33	Normal	Alfanumérico	4	2369	60
34	Leve	Alfanumérico	4	3183	75	34	Normal	Alfanumérico	4	2506	93
35	Leve	Alfanumérico	4	3351	92	35	Normal	Alfanumérico	4	2632	105
36	Leve	Alfanumérico	4	3537	81	36	Normal	Alfanumérico	4	2780	116
37	Leve	Alfanumérico	4	3729	117	37	Normal	Alfanumérico	4	2894	125
38	Leve	Alfanumérico	4	3927	234	38	Normal	Alfanumérico	4	3054	138
39	Leve	Alfanumérico	4	4087	386	39	Normal	Alfanumérico	4	3220	140
40	Leve	Alfanumérico	4	4296	442	40	Normal	Alfanumérico	4	3391	161

Tabla X: tipo de codificación alfanumérica**,** parámetros y número de caracteres que pueden contener algunos tipos de códigos QRCODE (redundancia leve y normal). **A** = versión; **B** = niveles de corrección; **C** = tipo; **D** = tamaño; **E** = caracteres; **F** = milisegundos que consume la generación del código (*).

TABLA XI

QRCODE (alfanumérico con redundancia mediana y grande)											
A	B	C	D	E	F	A	B	C	D	E	F
1	Mediana	Alfanumérico	4	16	01	1	Grande	Alfanumérico	4	10	02
2	Mediana	Alfanumérico	4	29	02	2	Grande	Alfanumérico	4	20	04
3	Mediana	Alfanumérico	4	47	03	3	Grande	Alfanumérico	4	35	05
4	Mediana	Alfanumérico	4	67	04	4	Grande	Alfanumérico	4	50	06
5	Mediana	Alfanumérico	4	87	05	5	Grande	Alfanumérico	4	64	07
6	Mediana	Alfanumérico	4	108	06	6	Grande	Alfanumérico	4	84	08
7	Mediana	Alfanumérico	4	125	07	7	Grande	Alfanumérico	4	93	09
8	Mediana	Alfanumérico	4	157	08	8	Grande	Alfanumérico	4	122	10
9	Mediana	Alfanumérico	4	189	09	9	Grande	Alfanumérico	4	143	11
10	Mediana	Alfanumérico	4	221	10	10	Grande	Alfanumérico	4	174	12
11	Mediana	Alfanumérico	4	259	11	11	Grande	Alfanumérico	4	200	13
12	Mediana	Alfanumérico	4	296	12	12	Grande	Alfanumérico	4	227	14
13	Mediana	Alfanumérico	4	352	13	13	Grande	Alfanumérico	4	259	15
14	Mediana	Alfanumérico	4	376	14	14	Grande	Alfanumérico	4	283	17
15	Mediana	Alfanumérico	4	426	15	15	Grande	Alfanumérico	4	321	17
16	Mediana	Alfanumérico	4	470	16	16	Grande	Alfanumérico	4	365	18
17	Mediana	Alfanumérico	4	531	19	17	Grande	Alfanumérico	4	408	20
18	Mediana	Alfanumérico	4	574	20	18	Grande	Alfanumérico	4	452	21
19	Mediana	Alfanumérico	4	644	23	19	Grande	Alfanumérico	4	493	23
20	Mediana	Alfanumérico	4	702	24	20	Grande	Alfanumérico	4	557	25
21	Mediana	Alfanumérico	4	742	25	21	Grande	Alfanumérico	4	587	28
22	Mediana	Alfanumérico	4	823	27	22	Grande	Alfanumérico	4	640	30
23	Mediana	Alfanumérico	4	890	31	23	Grande	Alfanumérico	4	672	32
24	Mediana	Alfanumérico	4	963	34	24	Grande	Alfanumérico	4	744	34
25	Mediana	Alfanumérico	4	1041	38	25	Grande	Alfanumérico	4	779	39
26	Mediana	Alfanumérico	4	1094	39	26	Grande	Alfanumérico	4	864	40
27	Mediana	Alfanumérico	4	1172	42	27	Grande	Alfanumérico	4	910	42
28	Mediana	Alfanumérico	4	1263	49	28	Grande	Alfanumérico	4	958	49
29	Mediana	Alfanumérico	4	1322	51	29	Grande	Alfanumérico	4	1016	50
30	Mediana	Alfanumérico	4	1429	52	30	Grande	Alfanumérico	4	1080	52
31	Mediana	Alfanumérico	4	1499	56	31	Grande	Alfanumérico	4	1150	64
32	Mediana	Alfanumérico	4	1618	57	32	Grande	Alfanumérico	4	1226	72
33	Mediana	Alfanumérico	4	1700	59	33	Grande	Alfanumérico	4	1307	76
34	Mediana	Alfanumérico	4	1787	64	34	Grande	Alfanumérico	4	1394	78
35	Mediana	Alfanumérico	4	1867	66	35	Grande	Alfanumérico	4	1431	80
36	Mediana	Alfanumérico	4	1966	70	36	Grande	Alfanumérico	4	1530	85
37	Mediana	Alfanumérico	4	2071	73	37	Grande	Alfanumérico	4	1591	94
38	Mediana	Alfanumérico	4	2181	79	38	Grande	Alfanumérico	4	1658	125
39	Mediana	Alfanumérico	4	2298	81	39	Grande	Alfanumérico	4	1774	156
40	Mediana	Alfanumérico	4	2420	85	40	Grande	Alfanumérico	4	1852	367

Tabla XI: tipo de codificación alfanumérica**,** parámetros y número de caracteres que pueden contener algunos tipos de códigos QRCODE (redundancia mediana y grande). **A** = versión; **B** = niveles de corrección; **C** = tipo; **D** = tamaño; **E** = caracteres; **F** = milisegundos que consume la generación del código (*).

TABLA XII

QRCODE (numérico redundancia leve y normal)											
A	B	C	D	E	F	A	B	C	D	E	F
1	Leve	Numérico	4	41	01	1	Normal	Numérico	4	34	01
2	Leve	Numérico	4	77	02	2	Normal	Numérico	4	63	04
3	Leve	Numérico	4	127	03	3	Normal	Numérico	4	101	05
4	Leve	Numérico	4	187	04	4	Normal	Numérico	4	149	06
5	Leve	Numérico	4	255	05	5	Normal	Numérico	4	202	07
6	Leve	Numérico	4	322	06	6	Normal	Numérico	4	255	08
7	Leve	Numérico	4	370	07	7	Normal	Numérico	4	293	11
8	Leve	Numérico	4	461	09	8	Normal	Numérico	4	365	14
9	Leve	Numérico	4	552	10	9	Normal	Numérico	4	432	16
10	Leve	Numérico	4	652	11	10	Normal	Numérico	4	513	17
11	Leve	Numérico	4	772	12	11	Normal	Numérico	4	604	18
12	Leve	Numérico	4	883	13	12	Normal	Numérico	4	691	19
13	Leve	Numérico	4	1022	14	13	Normal	Numérico	4	796	20
14	Leve	Numérico	4	1101	15	14	Normal	Numérico	4	871	21
15	Leve	Numérico	4	1250	16	15	Normal	Numérico	4	991	22
16	Leve	Numérico	4	1408	18	16	Normal	Numérico	4	1082	23
17	Leve	Numérico	4	1548	20	17	Normal	Numérico	4	1212	24
18	Leve	Numérico	4	1725	24	18	Normal	Numérico	4	1346	25
19	Leve	Numérico	4	1903	26	19	Normal	Numérico	4	1500	26
20	Leve	Numérico	4	2061	27	20	Normal	Numérico	4	1600	27
21	Leve	Numérico	4	2232	28	21	Normal	Numérico	4	1708	28
22	Leve	Numérico	4	2409	32	22	Normal	Numérico	4	1872	30
23	Leve	Numérico	4	2620	33	23	Normal	Numérico	4	2059	31
24	Leve	Numérico	4	2812	36	24	Normal	Numérico	4	2188	35
25	Leve	Numérico	4	3057	44	25	Normal	Numérico	4	2395	37
26	Leve	Numérico	4	3283	46	26	Normal	Numérico	4	2544	42
27	Leve	Numérico	4	3517	48	27	Normal	Numérico	4	2701	49
28	Leve	Numérico	4	3669	48	28	Normal	Numérico	4	2857	50
29	Leve	Numérico	4	3909	56	29	Normal	Numérico	4	3035	51
30	Leve	Numérico	4	4158	63	30	Normal	Numérico	4	3289	52
31	Leve	Numérico	4	4417	72	31	Normal	Numérico	4	3486	54
32	Leve	Numérico	4	4686	76	32	Normal	Numérico	4	3693	57
33	Leve	Numérico	4	4965	77	33	Normal	Numérico	4	3909	60
34	Leve	Numérico	4	5253	78	34	Normal	Numérico	4	4134	64
35	Leve	Numérico	4	5529	79	35	Normal	Numérico	4	4343	67
36	Leve	Numérico	4	5836	80	36	Normal	Numérico	4	4588	71
37	Leve	Numérico	4	6153	82	37	Normal	Numérico	4	4775	74
38	Leve	Numérico	4	6479	85	38	Normal	Numérico	4	5039	79
39	Leve	Numérico	4	6743	86	39	Normal	Numérico	4	5313	81
40	Leve	Numérico	4	7089	118	40	Normal	Numérico	4	5596	85

Tabla XII: tipo de codificación numérica**,** parámetros y número de caracteres que pueden contener algunos tipos de códigos QRCODE (redundancia leve y normal). **A** = versión; **B** = niveles de corrección; **C** = tipo; **D** = tamaño; **E** = caracteres; **F** = milisegundos que consume la generación del código (*).

TABLA XIII

QRCODE (numérico redundancia mediana y grande)											
A	B	C	D	E	F	A	B	C	D	E	F
1	Mediana	Numérico	4	27	01	1	Grande	Numérico	4	17	01
2	Mediana	Numérico	4	48	02	2	Grande	Numérico	4	34	02
3	Mediana	Numérico	4	77	03	3	Grande	Numérico	4	58	03
4	Mediana	Numérico	4	111	04	4	Grande	Numérico	4	82	04
5	Mediana	Numérico	4	144	05	5	Grande	Numérico	4	106	05
6	Mediana	Numérico	4	178	06	6	Grande	Numérico	4	139	06
7	Mediana	Numérico	4	207	07	7	Grande	Numérico	4	154	07
8	Mediana	Numérico	4	259	08	8	Grande	Numérico	4	202	08
9	Mediana	Numérico	4	312	09	9	Grande	Numérico	4	235	09
10	Mediana	Numérico	4	364	10	10	Grande	Numérico	4	288	10
11	Mediana	Numérico	4	427	11	11	Grande	Numérico	4	331	11
12	Mediana	Numérico	4	489	12	12	Grande	Numérico	4	374	12
13	Mediana	Numérico	4	580	13	13	Grande	Numérico	4	427	13
14	Mediana	Numérico	4	621	14	14	Grande	Numérico	4	468	18
15	Mediana	Numérico	4	703	15	15	Grande	Numérico	4	530	20
16	Mediana	Numérico	4	775	16	16	Grande	Numérico	4	602	22
17	Mediana	Numérico	4	876	19	17	Grande	Numérico	4	674	23
18	Mediana	Numérico	4	948	21	18	Grande	Numérico	4	746	24
19	Mediana	Numérico	4	1063	23	19	Grande	Numérico	4	813	25
20	Mediana	Numérico	4	1159	25	20	Grande	Numérico	4	919	26
21	Mediana	Numérico	4	1224	27	21	Grande	Numérico	4	969	27
22	Mediana	Numérico	4	1358	30	22	Grande	Numérico	4	1056	29
23	Mediana	Numérico	4	1468	33	23	Grande	Numérico	4	1108	32
24	Mediana	Numérico	4	1588	35	24	Grande	Numérico	4	1228	34
25	Mediana	Numérico	4	1718	37	25	Grande	Numérico	4	1286	40
26	Mediana	Numérico	4	1804	39	26	Grande	Numérico	4	1425	42
27	Mediana	Numérico	4	1933	41	27	Grande	Numérico	4	1501	44
28	Mediana	Numérico	4	2085	42	28	Grande	Numérico	4	1581	46
29	Mediana	Numérico	4	2181	45	29	Grande	Numérico	4	1677	48
30	Mediana	Numérico	4	2358	48	30	Grande	Numérico	4	1782	51
31	Mediana	Numérico	4	2473	54	31	Grande	Numérico	4	1897	54
32	Mediana	Numérico	4	2670	55	32	Grande	Numérico	4	2022	57
33	Mediana	Numérico	4	2805	59	33	Grande	Numérico	4	2157	60
34	Mediana	Numérico	4	2949	66	34	Grande	Numérico	4	2301	65
35	Mediana	Numérico	4	3081	68	35	Grande	Numérico	4	2361	68
36	Mediana	Numérico	4	3244	70	36	Grande	Numérico	4	2524	73
37	Mediana	Numérico	4	3417	74	37	Grande	Numérico	4	2625	75
38	Mediana	Numérico	4	3599	78	38	Grande	Numérico	4	2735	86
39	Mediana	Numérico	4	3791	83	39	Grande	Numérico	4	2927	97
40	Mediana	Numérico	4	3993	89	40	Grande	Numérico	4	3057	115

Tabla XIII: tipo de codificación numérica**,** parámetros y número de caracteres que pueden contener algunos tipos de códigos QRCODE (redundancia mediana y grande). **A** = versión; **B** = niveles de corrección; **C** = tipo; **D** = tamaño; **E** = caracteres; **F** milisegundos que consume la generación del código (*).

(*) Como plataforma de generación y mediciones se emplearon las carpetas DATAMATRIX, QRCODE, AZTEC, DECODIFICADOR y CAMARA WEB, del módulo “CODIFICADOR-DECODIFICADOR DE CÓDIGOS 2D” para ANIMALIA: Sistema Ephesia V.4, 64 bits, 2014 (ON RELEASE). Las pruebas se realizaron añadiendo cadenas de semillas aleatorias de cifras (randomize). Se procedió de esta manera porque entendemos que el uso de números pudiera ser la rutina más habitual en la búsqueda secuencial de dígitos para el acceso dinámico a los índices de las tablas de datos relacionales. En cualquier caso, se sobreentiende que en el supuesto de utilizar cadenas híbridas compuestas de guarismos y caracteres, el cómputo de signos por código en relación al tamaño del mismo resultaría inferior de manera directamente proporcional al uso de mayor cantidad letras y menor de cifras. Debido a que las medidas se han tomado en milisegundos, es decir, con extrema precisión, estos valores no se han de entender como categóricos. Los algoritmos de medición de tiempo se han independizado de la compilación del código principal en procesos multi-hilo (threads) pero aun así es factible que existan variaciones en decenas de milisegundos. El resultado pudiera ser variable debido a muchos factores: procesos internos del sistema operativo, uso de los procesadores, acceso a la memoria, servicios activos y otros elementos estocásticos que incumben al automatismo de la frecuencia de reloj y a los recursos del sistema utilizados por los aplicativos en ejecución en el preciso instante de la generación del código.

TABLA XIV

ESPECTROFOTOMETRÍA ACS

F = fondos S = símbolos.	Tipo de luz	Lum.	Eje R-G	Eje Y-B	Sat.	Ton.
F-rojo	D65 10º 6500 o K	52.47	53.46	39.53	66.49	36.48
	10º Tungsteno	60.67	56.29	52.64	77.07	43.08
	CWF 10º Luz Día	52.70	42.98	40.46	59.03	43.27
F-violeta	D65 10º 6500 o K	52.69	31.85	-31.65	44.90	315.18
	10º Tungsteno	53.92	26.81	-27.49	38.40	314.28
	CWF 10º Luz Día	51.06	25 11	-35.89	43.80	304.98
F-azul	D65 10º 6500 o K	60.99	-16.26	-38.97	42.23	347.36
	10º Tungsteno	55.10	-27.21	-49.09	56.12	241.00
	CWF 10º Luz Día	55.76	-11.34	-47.82	49.15	256.66
F-verde	D65 10º 6500 o K	57.59	-53.58	30.07	61.44	150.70
	10º Tungsteno	53.96	-48.45	19.28	52.14	158.30
	CWF 10º Luz Día	55.38	-41.30	28.16	49.99	145.71
F-amarillo	D65 10º 6500 o K	87.69	0.02	97.79	97.79	89.99
	10º Tungsteno	91.45	7.95	98.07	98.39	85.37
	CWF 10º Luz Día	90.97	-2.15	104.68	104.70	91.18
F-naranja	D65 10º 6500 o K	66.65	26.61	62.34	67.78	66.89
	10º Tungsteno	72.30	29.45	69.51	75.49	67.04
	CWF 10º Luz Día	70.34	19.72	69.63	72.37	74.18
S-verde	D65 10º 6500 o K	57.34	-54.96	23.27	59.73	156.96
	10º Tungsteno	53.20	-46.51	12.18	48.08	165.33
	CWF 10º Luz Día	53.38	-41.08	17.91	44.81	156.45
S-azul	D65 10º 6500 o K	31.51	26.20	-52.74	58.89	296.42
	10º Tungsteno	28.01	8.13	-56.47	57.05	278.19
	CWF 10º Luz Día	28.26	22.22	-59.74	63.74	290.40
S-marrón	D65 10º 6500 o K	42.07	5.80	20.39	21.20	74.11
	10º Tungsteno	43.87	8.81	22.31	23.98	68.46
	CWF 10º Luz Día	43.25	4.43	23.04	23.47	79.12

Tabla XIV: espectrofotometría ACS y combinación de colores adecuada para la codificación-decodificación de los códigos considerando el tipo de luz arrojada sobre la micro-etiqueta.

TABLA XV

FALLOS MÁS COMUNES EN LA LECTURA DE MICRO-ETIQUETAS POR LECTORES DE CÓDIGOS

Síntomas	Posibles causas	Posibles soluciones
No se enciende nada.	No hay alimentación.	Verificar está bien conectado.
Suena dos veces y parpadean las luces alternativamente al encender el lector.	Error en la memoria ROM.	Se requiere actualizar la memoria ROM.
Suena 3 veces al encender el lector.	Error en la memoria RAM.	Contactar centro de reparaciones.
Hay un zumbido al encender el lector.	Error memoria RAM o ROM	Contactar centro de reparaciones.
Hay un zumbido un led parpadea, al encender el lector.	Error con el láser.	Contactar centro de reparaciones.
Hay un zumbido y ambos leds parpadean, al encender el lector.	Error en el mecanismo de lectura.	Contactar centro de reparaciones.
El lector lee, transmite y suena dos o varias veces.	El tiempo para lectura de un mismo código es muy corto.	Configurar el tiempo de lectura a un lapso mayor de tiempo.
El lector no suena pero lee y decodifica.	La bocina puede estar apagada.	Habilitar bocina.
El lector no suena y tampoco lee al ser encendido, pero si hay barrido.	Se intenta leer un código no admitido.	Verificar que el código es compatible con el modelo de lector.
El lector no suena y tampoco hace la lectura al ser encendido, pero si hay barrido e intenta leer hasta que se para.	El tamaño mínimo de lectura es inadecuado y la bocina puede estar apagada.	Verificar que el tamaño mínimo del código está entre los parámetros soportados. Activar bocina.
Después de leer un código de barras el lector se bloquea y no se muestra nada.	El PC no está recibiendo la señal del lector	Verificar el cable de PC o el sistema de trasmisión inalámbrica.
El lector suena, lanza barrido, no ofrece resultado y tarda en reaccionar, esto sucede para algunos códigos con idéntica información y con la misma simbología, pero con distinto color, tamaño, sistema de impresión o diferente soporte.	La calidad de impresión es baja o el papel es malo o inadecuado. El código está deteriorado o presenta aberraciones notorias. La combinación de colores del fondo y el gráfico no es la soportada. El código no tiene la proporción que necesita el tipo al que se refiere.	Imprimir en alta resolución y mejorar el tipo de papel de soporte. Volver a generar e imprimir el código deteriorado. Utilizar la combinación de colores adecuada y soportada por el lector. Aumentar tamaño del símbolo. Controlar la proporción entre el ancho y la altura.
El lector lee, pero el dato no es correcto.	La configuración no es correcta.	Verificar el tipo de sistema operativo.

El lector puede leer, pero el dato transmitido a la PC es incorrecto.	Los parámetros de comunicación son incompatibles con los del PC.	Verificar que el lector está conectado al puerto de comunicaciones correcto.
El lector transmite cada carácter dos veces o más.	La configuración del lector no es correcta.	Incrementar los parámetros de retardo entre lecturas de caracteres.
Caracteres alfabéticos se muestran en minúsculas o mayúsculas.	El teclado está habilitado de esta manera.	Cambie el modo en el teclado.
Muestra una letra que no corresponde a la codificada.	El lenguaje del teclado es incorrecto.	Habilitar la configuración regional del teclado.
Aparecen uno o varios espacios de línea con retorno de carro, antes, después o en ambas posiciones de la cadena de lectura del código.	El lector está configurado de esta manera.	Cambiar configuración de los ENTER que trasmite y en qué posición. La configuración ideal es un RETURN después de leer el código.
La anchura del haz de lectura es muy pequeña y el lector 1D no reacciona.	El lector está configurado de esta manera.	Configurar la anchura del haz láser correctamente.
Se leen varios códigos alternativamente de manera muy rápida y el resultado se muestra en varias líneas con retorno de carro entre resultados.	Las micro-etiquetas están demasiado cerca unas de otras.	Separar las etiquetas. El barrido es demasiado grande, ancho o alto.
Algunos símbolos de otros alfabetos aparecen como espacios en blanco o como guiones u otro carácter.	El símbolo que se desea codificar no es admitido por el PC.	Instalar la fuente adecuada para la decodificación del símbolo.
El lector no hace nada con códigos 2D, pero si funcionan las luces, bocina y decodifica perfectamente códigos 1D.	El lector no admite códigos 2D o esta funcionalidad está apagada en la configuración por defecto.	Habilitar la funcionalidad 2D. Si esto no es posible, el lector no sirve para decodificar códigos 2D.
Aparecen tabulaciones.	El lector está configurado así.	Configure las tabulaciones del lector.

Tabla XV: problemas en los lectores y síntomas más frecuentes en la lectura de micro-etiquetas con códigos. Posibles soluciones. Se proporciona una guía genérica para los lectores más corrientes en el mercado. Con algunos tipos de aparatos menos comunes, la información del parpadeo de las luces y secuencia de pitidos pudieran diferir, en este caso consulte la documentación de su modelo.

TABLA XVI

Alfiler	Gr.
000 y 00	100 gr. x m^2
0 y 1	150 gr. x m^2
2 y 3	200 gr. x m^2
4 y 5	250 gr. x m^2
>5	300 gr. x m^2

Tabla XVI: Gramos máximos del papel de impresión de códigos, en relación al grosor del alfiler entomológico.

TABLA XVII

ShellExecute		
	Tipo	**Descripción**
1	HWND	La aplicación que tiene el control. Este parámetro no es indispensable y por lo general si pasamos NULL es suficiente.
2	LPCTSTR	Una cadena de texto que indica el tipo de comando a procesar. Puede ser: edit, open, explore, find, print y NULL. Para convertir una cadena de texto a LPCTSTR podemos utilizar la función TEXT.
3	LPCTSTR	Una cadena de texto con la ruta completa del archivo o carpeta a procesar. Para convertir una cadena de texto a LPCTSTR podemos utilizar la función TEXT.
4	LPCTSTR	Una cadena de texto con los parámetros a enviar a la aplicación. Si no existen parámetros se puede mandar NULL. Para convertir una cadena de texto a LPCTSTR podemos utilizar la función TEXT.
5	LPCTSTR	Una cadena de texto con la ruta del directorio que se utilizará como base al momento de ejecutar el comando. Este parámetro por lo general es NULL. Para convertir una cadena de texto a LPCTSTR podemos utilizar la función TEXT.
6	INT	Un entero que indica cómo se desea mostrar la aplicación. Se recomienda utilizar las constantes predefinidas siendo las más comunes las siguientes: SW_SHOWNORMAL, SW_SHOWMINIMIZED, SW_SHOWMAXIMIZED y S W_HIDE.

Tabla XVII: Parámetros ShellExecute (HWND, LPCTSTR, LPCTSTR, LPCTSTR, LPCTSTR, INT). Para lanzar aplicaciones externas (véase anexo III, código XI).

Anexo II

Fórmulas

Fórmula I

	Posiciones																	
COD											**N1**	**N2**	**N3**	**N4**	**N5**	**N6**	**N7**	**N8**
12							N1	N2	N3	N4	N5	N6	N7	N8	N9	N10	N11	N12
13						N1	N2	N3	N4	N5	N6	N7	N8	N9	N10	N11	N12	N13
14					N1	N2	N3	N4	N5	N6	N7	N8	N9	N10	N11	N12	N13	N14
18	N1	N2	N3	N4	N5	N6	N7	N8	N9	N10	N11	N12	N13	N14	N15	N16	N17	N18

x

x3	**x1**	**x3**	**x1**	**x3**	**x1**	**x3**	**x1**	**x3**	**x1**	**x3**	**x1**	**x3**	**x1**	**x3**	**x1**	**x3**

+

Decena anterior - resultado	**=**	**Dígito verificador**

Fórmula I: tabla de cálculo de para códigos de barras 8-18.

Fórmula II

1	**2**	**3**	**4**	**5**	**6**	**7**	
9	**9**	**2**	**1**	**6**	**4**	**6**	
x 3	x 1	x 3	x 1	x 3	x 1	x 3	
27	9	6	1	18	4	18	
27 + 9 + 6 + 1 + 18 + 4 + 18							**83**
90 - 83							**7**

Fórmula II: tabla de cálculo para códigos de barras EAN8. Los números (9921646) se han tomado de la figura 52.

Anexo III

Algoritmia

Código I

```
byte procesoFinalizado = 0;

private void txtCodigoBarras_KeyPress(object sender, KeyPressEventArgs e)
    {
       if (e.KeyChar == (Char)Keys.Return)
       {
          string ID = this txtCodigoBarras.Text;
          while (procesoFinalizado <= 9)
          {
             procesoFinalizado = lanzarModulo();
             procesoFinalizado = conectarBaseDeDatos();
             procesoFinalizado = generarInformeConID(ID);
             procesoFinalizado = ordenarPorFecha();
             procesoFinalizado = organizarTiposPrepaciones();
             procesoFinalizado = enumerarSistemasDeInclusión();
             procesoFinalizado = armariosCajas(ID);
             procesoFinalizado = actualizarFecha(ID);
             procesoFinalizado = comprobarIntegridadActualizarTabla(ID);
             procesoFinalizado = 10;
          }
          this.Dispose();
          Application.Exit();
       }
    }

   private byte lanzarModulo()
   {
      //Aquí código de ejecución
      return procesoFinalizado = 1;
   }

   private byte conectarBaseDeDatos()
   {
      //Aquí código de ejecución
      return procesoFinalizado = 2;
   }

   private byte generarInformeConID(string ID)
   {
      //Aquí código de ejecución
      return procesoFinalizado = 3;
   }

   private byte ordenarPorFecha()
   {
      //Aquí código de ejecución
      return procesoFinalizado = 4;
   }
```

```
private byte organizarTiposPrepaciones()
{
   //Aquí código de ejecución
   return procesoFinalizado = 5;
}

private byte enumerarSistemasDeInclusión()
{
   //Aquí código de ejecución
   return procesoFinalizado = 6;
}

private byte armariosCajas(string ID)
{
   //Aquí código de ejecución
   return procesoFinalizado = 7;
}

private byte actualizarFecha(string ID)
{
   //Aquí código de ejecución
   return procesoFinalizado = 8;
}

private byte comprobarIntegridadActualizarTabla(string ID)
{
   //Aquí código de ejecución
   return procesoFinalizado = 9;
}
```

Código II

```
public static string EANOcho(string valor)
    {
       int i;
       double m_validador;
       string CodigoBarra = "";
       m_validador = 0;
       if (valor.Length == 7)
       {
          for (i = 1; i <= 7; i++)
          {
             int L1 = Convert.ToChar(valor.Substring(i - 1, 1));
             if ((L1 < 48) || (L1 > 57))
             {
                i = 0;
                break;
             }
          }
          if (i == 8)
          {
             for (i = 7; i > 0; i = i - 2)
             {
                m_validador =+ Convert.ToInt32(valor.Substring(i - 1, 1));
             }
```

```
            m_validador = m_validador * 3;

            for (i = 6; i > 0; i = i - 2)
            {
               m_validador = m_validador + Convert.ToInt32(valor.Substring(i - 1, 1));
            }
            valor = valor + (10 - m_validador % 10) % 10;

            CodigoBarra = ":";

            for (i = 1; i <= 4; i++)
            {
               CodigoBarra = CodigoBarra + Convert.ToChar(65 +
                     Convert.ToInt32(valor.Substring(i - 1, 1)));
            }
            CodigoBarra = CodigoBarra + "*";
            for (i = 5; i <= 8; i++)
            {
               CodigoBarra = CodigoBarra + Convert.ToChar(97 +
                     Convert.ToInt32(valor.Substring(i - 1, 1)));
            }
            CodigoBarra = CodigoBarra + "+";
         }
      }
      return CodigoBarra;
   }
```

Código III

```
private void GenerarAZTEC ()
    {
       try
       {
          Cursor = Cursors.WaitCursor;
          this.txtCronoAZTEC.Text = EphesiaUtilidades.cronometro(1);
          this.pctCodigoAZTEC.SizeMode = PictureBoxSizeMode.AutoSize;
          int tamañoImagenAZTEC = Convert.ToInt16
                (this.cboAZTEC.SelectedItem.ToString().Substring(10, 3));
          this.pctCodigoAZTEC.Image = EphesiaUtilidades.generarAZTEC
                (this.txtAZTEC.Text.Trim(), tamañoImagenAZTEC);
          this.pctMuestraAZTEC.Image = this.pctCodigoAZTEC.Image;
          this.txtCronoAZTEC.Text = EphesiaUtilidades.cronometro(0);
          tamañoImagen = this.pctCodigoAZTEC.Size;
          this.hScrollBarAZTEC.Value = tamañoImagen.Height;
          if (this.pctCodigoAZTEC.Height <= this.pctMuestraAZTEC.Height)
          {
             this.pctMuestraAZTEC.SizeMode = PictureBoxSizeMode.Normal;
             this.pctMuestraAZTEC.Image = this.pctCodigoAZTEC.Image;
          }
          else
          {
             this.pctMuestraAZTEC.SizeMode = PictureBoxSizeMode.StretchImage;
             this.pctMuestraAZTEC.Image = this.pctCodigoAZTEC.Image;
          }
          EphesiaUtilidades.CentrarControlEnPanel(this.pnlAZTEC, this.pctCodigoAZTEC);
                this.lblDatosAZTEC.Text = "Código AZTEC (" +
                this.pctCodigoAZTEC.Width.ToString()
```

```
                + " x " + this.pctCodigoAZTEC.Height.ToString() + " píxeles; "  +
                EphesiaUtilidades.cadenaPixelesMilimetros(this.pctCodigoAZTEC.Width) +
                " x "   +
                EphesiaUtilidades.cadenaPixelesMilimetros(this.pctCodigoAZTEC.Height) + "
                milímetros).";
            contAZTEC();
            Cursor = Cursors.Default;
        }
        catch (System.NullReferenceException)
        {
            EphesiaError.errorGenerarCodigo("El código no se puede generar,
                  el texto contenido es demasiado grande.");
            this.pctMuestraEscala.Image = null;
            Cursor = Cursors.Default;
        }
        catch (IndexOutOfRangeException)
        {
            EphesiaError.errorFueraDeRangoCadena("Demasiadas generaciones en cascada
                  para la memoria del sistema.");
            indexOutOfRangeException = 1;
            Cursor = Cursors.Default;
        }
        catch (System.Exception err)
        {
            EphesiaError.errorDesconocido(err);
            Cursor = Cursors.Default;
        }
    }
```

Código IV

```
private void generarDataMatrix(string textoCodificar, Int32 modo, Int32 tamañoSimbolo)
    {
        try
        {
            Cursor = Cursors.WaitCursor;
            this.pctDataMatrix.SizeMode = PictureBoxSizeMode.AutoSize;
            this.txtCronometradoDT.Text = EphesiaUtilidades.cronometro(1);
            this.pctDataMatrix.Image = EphesiaUtilidades.generarDATAMATRIX
                  (textoCodificar, this.dfMargin.Text,
                this.dfPixelSize.Text, modo, tamañoSimbolo);
            this.txtCronometradoDT.Text = EphesiaUtilidades.cronometro(0);
            tamañoImagen = this.pctDataMatrix.Size;
            this.hScrollBar2.Value = tamañoImagen.Height;
            if (this.pctDataMatrix.Height <= this.pctMuestraEscala.Height)
            {
                this.pctMuestraEscala.SizeMode = PictureBoxSizeMode.Normal;
                this.pctMuestraEscala.Image = this.pctDataMatrix.Image;
            }
            else
            {
                this.pctMuestraEscala.SizeMode = PictureBoxSizeMode.StretchImage;
                this.pctMuestraEscala.Image = this.pctDataMatrix.Image;
            }
            EphesiaUtilidades.CentrarControlEnPanel(this.panCodificarDTM,
            this.pctDataMatrix);
            this.lblDATAMATRIX.Text = "Código DATAMATRIX (" +
            this.pctDataMatrix.Width.ToString()
```

```
                + " x " + this.pctDataMatrix.Height.ToString() + " píxeles; "
                + EphesiaUtilidades.cadenaPixelesMilimetros(this.pctDataMatrix.Width)
                + " x " + EphesiaUtilidades.cadenaPixelesMilimetros(this.pctDataMatrix.Height)
                + " milímetros).";
            contDT();
            Cursor = Cursors.Default;
        }
        catch (System.NullReferenceException)
        {
            EphesiaError.errorGenerarCodigo("El código no se puede generar,
                    el texto contenido es demasiado grande.");
            this.pctMuestraEscala.Image = null;
            Cursor = Cursors.Default;
        }
        catch (IndexOutOfRangeException)
        {
           EphesiaError.errorFueraDeRangoCadena("Demasiadas generaciones en cascada
                    para la memoria del sistema.");
            indexOutOfRangeException = 1;
            Cursor = Cursors.Default;
        }
        catch (System.Exception err)
        {
            EphesiaError.errorDesconocido(err);
            Cursor = Cursors.Default;
        }
    }
```

Código V

```
private void GennerarQR()
    {
        try
        {
            Cursor = Cursors.WaitCursor;
            if (this.txtEncodeData.Text.Trim() == String.Empty)
            {
                EphesiaError.errorSimpleVacios("de datos", "");
            }
            else
            {
                this.pctEncode.SizeMode = PictureBoxSizeMode.AutoSize;
                contQR();
                this.txtCrono.Text = EphesiaUtilidades.cronometro(1);
                Image image = EphesiaUtilidades.asignarQR(this.txtEncodeData.Text.Trim(),
                    this.rbtByte.Checked,this.rbtAlfanumerico.Checked,
                    this.rbtNumerico.Checked, this.rbtL.Checked, this.rbtN.Checked,
                    this.rbtQ.Checked == true, this.rbtH.Checked, txtSize.Text, cboVersion.Text,
                    pixelColor, pixelColor1);
                this.txtCrono.Text = EphesiaUtilidades.cronometro(0);
                Image image1 = EphesiaUtilidades.asignarQR(this.txtEncodeData.Text.Trim(),
                    this.rbtByte.Checked,this.rbtAlfanumerico.Checked,
                    this.rbtNumerico.Checked,
                    this.rbtL.Checked, this.rbtN.Checked,
                    this.rbtQ.Checked == true, this.rbtH.Checked, txtSize.Text, cboVersion.Text,
                    Color.Black, Color.White);
```

```
                this.pctEncode.Image = image;
                tamañoImagen = this.pctEncode.Size;
                this.hScrollBar3.Value = tamañoImagen.Height;
                if (image.PhysicalDimension.Height <= this.pctMuestra.Height)
                {
                    this.pctMuestra.SizeMode = PictureBoxSizeMode.Normal;
                    this.pctMuestra.Image = image1;
                }
                else
                {
                    this.pctMuestra.SizeMode = PictureBoxSizeMode.StretchImage;
                    this.pctMuestra.Image = image1;
                }
                EphesiaUtilidades.CentrarControlEnPanel(this.panCodificarQR, this.pctEncode);
                this.lblCodigoTamaño.Text = "QRCODE (" + this.pctEncode.Width.ToString()
                      + " x "
                    + this.pctEncode.Height.ToString() + "  píxeles; "
                    + EphesiaUtilidades.cadenaPixelesMilimetros(this.pctEncode.Width) + " x "
                    + EphesiaUtilidades.cadenaPixelesMilimetros(this.pctEncode.Height) +
                     "  milímetros).";
                this.Refresh();
                Cursor = Cursors.Default;
            }
        }
        catch (System.NullReferenceException)
        {
            this.pctMuestra.Image = null;
            Cursor = Cursors.Default;
        }
        catch (System.Exception err)
        {
            EphesiaError.errorDesconocido(err);
            Cursor = Cursors.Default;
        }
    }
```

Código VI

d(x; y) = jfi : 1 _ i _ n y xi 6 = yigj : (1) / d = minfd(x; y) : x; y 2 C y x 6= yg:

Código VII

```
public static void guardarImagenCodigo(SaveFileDialog saveFileDialog, Image imagen)
    {
        try
        {
            saveFileDialog.InitialDirectory = CadenasEphesia.D +
           "\\EPHESIA\\Codigos\\IMAGENES\\";
            saveFileDialog.Filter = "JPg Image|*.jpg|Bitmap Image|*.bmp|Gif Image|*.gif|PNG
           Image|*.png";
            saveFileDialog.Title = "Guardar";
            saveFileDialog.FileName = CadenasEphesia.formatoFechaActualCarpeta() +
            "_ImageneGenerada";
            saveFileDialog.ShowDialog();
```

```
        if (saveFileDialog.FileName != "")
        {
            System.IO.FileStream fs = (System.IO.FileStream)saveFileDialog.OpenFile();
            switch (saveFileDialog.FilterIndex)
            {
                case 1:
                    imagen.Save(fs, System.Drawing.Imaging.ImageFormat.Jpeg);
                    break;
                case 2:
                    imagen.Save(fs, System.Drawing.Imaging.ImageFormat.Bmp);
                    break;
                case 3:
                    imagen.Save(fs, System.Drawing.Imaging.ImageFormat.Gif);
                    break;
                case 4:
                    imagen.Save(fs, System.Drawing.Imaging.ImageFormat.Png);
                    break;
            }
            fs.Close();
        }
    }
    catch (System.IO.IOException err)
    {
        EphesiaError.errorGuardandoImagen("Inspector de formatos de imagen.");
    }
    catch (Exception err)
    {
        EphesiaError.errorDesconocido(err);
    }
}
```

Código VIII

```
private const int WM_CAP = 0x400;
private const int WM_CAP_DRIVER_CONNECT = 0x40a;
private const int WM_CAP_DRIVER_DISCONNECT = 0x40b;
private const int WM_CAP_EDIT_COPY = 0x41e;
private const int WM_CAP_GET_FRAME = 1084;
private const int WM_CAP_COPY = 1054;
private const int WM_CAP_SET_PREVIEW = 0x432;
private const int WM_CAP_SET_OVERLAY = 0x433;
private const int WM_CAP_SET_PREVIEWRATE = 0x434;
private const int WM_CAP_SET_SCALE = 0x435;
private const int WS_CHILD = 0x40000000;
private const int WS_VISIBLE = 0x10000000;
private const int SWP_NOMOVE = 0x2;
private const int SWP_NOZORDER = 0x4;
private const int HWND_BOTTOM = 1;

[DllImport("avicap32.dll")]
protected static extern bool capGetDriverDescriptionA(short wDriverIndex,
[MarshalAs(UnmanagedType.VBByRefStr)]ref String lpszName,
int cbName, [MarshalAs(UnmanagedType.VBByRefStr)] ref String lpszVer, int cbVer);

[DllImport("avicap32.dll")]
protected static extern int
capCreateCaptureWindowA([MarshalAs(UnmanagedType.VBByRefStr)]
```

```
ref string lpszWindowName,
int dwStyle, int x, int y, int nWidth, int nHeight, int hWndParent, int nID);

[DllImport("user32")]
protected static extern int SetWindowPos(int hwnd, int hWndInsertAfter, int x, int y, int cx,
int cy,
int wFlags);

[DllImport("user32", EntryPoint = "SendMessageA")]
protected static extern int SendMessage(int hwnd, int wMsg, int wParam,
[MarshalAs(UnmanagedType.AsAny)] object lParam);

[DllImport("user32")]
protected static extern bool DestroyWindow(int hwnd);

int DeviceID = 0;
int hHwnd = 0;
ArrayList ListOfDevices = new ArrayList();

public PictureBox Container { get; set; }

private void btnConectarCam_MouseDown(object sender, MouseEventArgs e)
 {
    this.lblAnimacion.Visible = true;
    this.lblOcultaVideo.Visible = true;
    if (wCam == null)
    {
       wCam = new WebCam {Container = this.pctCam};
       wCam.OpenConnection();
       this.btnConectarCam.Enabled = false;
       this.btnApagarCamara.Enabled = true;
       this.btnCapturar.Enabled = true;
       this.txtDatosCam.Text = "";
    }
    timer1.Start();
 }

 private void btnApagarCamara_MouseDown(object sender, MouseEventArgs e)
 {
    wCam.Dispose();
    wCam = null;
    this.btnConectarCam.Enabled = true;
    if (pctModi.Image == null)
    {
       this.btnDecodificarCam.Enabled = false;
       this.btnApagarCamara.Enabled = false;
       this.btnCapturar.Enabled = false;
       this.btnGuardarImagen.Enabled = false;
       this.btnImprimirImagen.Enabled = false;
       this.ScrollAlfa.Enabled = false;
       this.hScrollBarColor.Enabled = false;
    }
    else
    {
       this.btnApagarCamara.Enabled = false;
    }
    this.pctCam.Image = null;
 }
```

```
    private void btnCapturar_MouseDown(object sender, MouseEventArgs e)
     {
        this.pctCaptura.Image = wCam.GetCurrentImage();
        btnApagarCamara_MouseDown(sender, e);
        btnConectarCam_MouseDown(sender, e);
        this.btnGuardarImagen.Enabled = true;
        this.btnImprimirImagen.Enabled = true;
        this.btnDecodificarCam.Enabled = true;
        this.ScrollAlfa.Enabled = true;
        this.hScrollBarColor.Enabled = true;
        this.pctModi.Image = this.pctCaptura.Image;
        this.pctModi.Visible = true;
     }
```

Código IX

```
using System;
using System.Drawing;
using System.Windows.Forms;
using System.Drawing.Imaging;

namespace Especimenes
{
   public partial class frmQr : Form
   {
      private string especimen = "";
      private Color pixelColor = Color.Black;
      private Color pixelColor1 = Color.White;
      private byte tabPulsada = 0;
      private byte indexOutOfRangeException = 0;
      private int numeroRbt = 1;
      private int numeroRbtDM = 1;
      private int numeroRbtAZTEC = 1;
      private String nombreRuta = "";
      private Size tamañoImagen;
      private ToolTip toolTip1 = new ToolTip();
      private WebCam wCam;//Camara
      private byte contadorRefresco = 1;
      //Tratamiento de imagen
      ColorMatrix colorMatrix;
      float c00 = 1, c01 = 0, c02 = 0, c03 = 0, c04 = 0, c10 = 0,
                  c11 = 1, c12 = 0, c13 = 0, c14 = 0;
      float c20 = 0, c21 = 0, c22 = 1, c23 = 0, c24 = 0, c30 = 0,
                  c31 = 0, c32 = 0, c33 = 1, c34 = 0;
      float c40 = 0, c41 = 0, c42 = 0;
      float c43 = 0, c44 = 0;
      byte filtroNumero = 0;
      Image image;
      ImageAttributes imageAttributes;
      Bitmap Bs = null;
      Graphics Gs;
      float auxFormula = 0f;

      public frmQr(string ESPECIMEN)
      {
         especimen = ESPECIMEN;
         InitializeComponent();
```

```
}

private void frmQr_Load(object sender, EventArgs e)
{
   if (especimen != "")
   {
      this.txtEncodeData.Text = especimen;
      GennerarQR();
   }
}

private void GennerarQR()
{
   try
   {
      Cursor = Cursors.WaitCursor;
      if (this.txtEncodeData.Text.Trim() == String.Empty)
      {
         EphesiaError.errorSimpleVacios("de datos", "");
      }
      else
      {
         this.pctEncode.SizeMode =
               PictureBoxSizeMode.AutoSize;
         contQR();
         this.txtCrono.Text =
               EphesiaUtilidades.cronometro(1);
         Image image =
               EphesiaUtilidades.asignarQR
         (this.txtEncodeData.Text.Trim(), this.rbtByte.Checked,
         this.rbtAlfanumerico.Checked, this.rbtNumerico.Checked, this.rbtL.Checked,
         this.rbtN.Checked,
         this.rbtQ.Checked == true, this.rbtH.Checked, txtSize.Text, cboVersion.Text,
         pixelColor, pixelColor1);
         this.txtCrono.Text = EphesiaUtilidades.cronometro(0);
         Image image1 = EphesiaUtilidades.asignarQR(
               this.txtEncodeData.Text.Trim(), this.rbtByte.Checked,
               this.rbtAlfanumerico.Checked, this.rbtNumerico.Checked, this.rbtL.Checked,
               this.rbtN.Checked, this.rbtQ.Checked == true, this.rbtH.Checked,
               this.txtSize.Text, cboVersion.Text, Color.Black, Color.White);
         this.pctEncode.Image = image;
         tamañoImagen = this.pctEncode.Size;
         this.hScrollBar3.Value = tamañoImagen.Height;
         if (image.PhysicalDimension.Height <=
                     this.pctMuestra.Height)
         {
            this.pctMuestra.SizeMode =
                     PictureBoxSizeMode.Normal;
            this.pctMuestra.Image = image1;
         }
         else
         {
            this.pctMuestra.SizeMode =
                     PictureBoxSizeMode.StretchImage;
            this.pctMuestra.Image = image1;
         }
               EphesiaUtilidades.CentrarControlEnPanel(
               this.panCodificarQR, this.pctEncode);
              this.lblCodigoTamaño.Text = "QRCODE (" +
```

```
                this.pctEncode.Width.ToString() + " x "
              + this.pctEncode.Height.ToString() + "
              píxeles; " +
              EphesiaUtilidades.cadenaPixelesMilimetros(
              this.pctEncode.Width) + " x " + EphesiaUtilidades.cadenaPixelesMilimetros
              (this.pctEncode.Height) + "  milímetros).";
          this.Refresh();
          Cursor = Cursors.Default;
       }
    }
    catch (System.NullReferenceException)
    {
       this.pctMuestra.Image = null;
       Cursor = Cursors.Default;
    }
    catch (System.Exception err)
    {
       EphesiaError.errorDesconocido(err);
       Cursor = Cursors.Default;
    }
}

private void printDocument1_PrintPage(object sender,
            System.Drawing.Printing.PrintPageEventArgs e)
{
    switch(tabPulsada)
    {
       case 1:
          e.Graphics.DrawImage(this.pctEncode.Image, 0, 0);
          break;
       case 2:
          e.Graphics.DrawImage(this.pctDataMatrix.Image, 0,
                                0);
          break;
       case 3:
          e.Graphics.DrawImage(this.pctCodigoAZTEC.Image,
                                0, 0);
          break;
       case 4:
          e.Graphics.DrawImage(this.pctModi.Image, 0, 0);
          break;
    }
}

private void btnSalir_MouseUp(object sender, MouseEventArgs
            e)
{
    this.pctEncode.Size = new Size(45, 45);
    this.pctDataMatrix.Size=new Size(45, 45);
    this.picDecode.Size = new Size(45, 45);
    this.Close();
    this.Dispose();
}

private void btnEspecimen_MouseUp(object sender,
            MouseEventArgs e)
{
    if (especimen != "")
    {
```

```
            this.txtEncodeData.Text = especimen;
            GennerarQR();
        }
    }

    private void frmQr_Shown(object sender, EventArgs e)
    {
        generarPrimeraVez();
    }

    private void generarPrimeraVez()
    {
        this.cbMode.SelectedIndex = 0;
        this.cbSymbolSize.SelectedIndex = 2;
        this.dfText.Text = especimen;
        generarDataMatrix(this.dfText.Text,
                    this.cbMode.SelectedIndex,
                    (this.cbSymbolSize.SelectedIndex - 2));
        this.txtAZTEC.Text = especimen;
        btnGenerarAZTEC_MouseDown(null, null);
    }

    private void btnGenerar_MouseDown(object sender,
                    MouseEventArgs e)
    {

        generarDataMatrix(this.dfText.Text,
                    this.cbMode.SelectedIndex,
                    (this.cbSymbolSize.SelectedIndex - 2));
    }

    private void generarDataMatrix(string textoCodificar, Int32
                            modo, Int32 tamañoSimbolo)
    {
        try
        {
            Cursor = Cursors.WaitCursor;
            this.pctDataMatrix.SizeMode =
                    PictureBoxSizeMode.AutoSize;
            this.txtCronometradoDT.Text =
                    EphesiaUtilidades.cronometro(1);
            this.pctDataMatrix.Image =
                    EphesiaUtilidades.generarDATAMATRIX(
                    textoCodificar,
                    this.dfMargin.Text,
                    this.dfPixelSize.Text, modo, tamañoSimbolo);
            this.txtCronometradoDT.Text =
                    EphesiaUtilidades.cronometro(0);
            tamañoImagen = this.pctDataMatrix.Size;
            this.hScrollBar2.Value = tamañoImagen.Height;
            if (this.pctDataMatrix.Height <=
                    this.pctMuestraEscala.Height)
            {
                this.pctMuestraEscala.SizeMode =
                    PictureBoxSizeMode.Normal;
                this.pctMuestraEscala.Image =
                    this.pctDataMatrix.Image;
            }
            else
```

```
        {
            this.pctMuestraEscala.SizeMode =
                PictureBoxSizeMode.StretchImage;
            this.pctMuestraEscala.Image =
                this.pctDataMatrix.Image;
        }
                EphesiaUtilidades.CentrarControlEnPanel(
                        this.panCodificarDTM, this.pctDataMatrix); this.lblDATAMATRIX.Text =
                        "Código DATAMATRIX (" + this.pctDataMatrix.Width.ToString()
         + " x " + this.pctDataMatrix.Height.ToString() +
                        " píxeles; " +
                        EphesiaUtilidades.cadenaPixelesMilimetros(
                        this.pctDataMatrix.Width)  + " x " +
                        EphesiaUtilidades.cadenaPixelesMilimetros(
                        this.pctDataMatrix.Height) + " milímetros).";
          contDT();
          Cursor = Cursors.Default;
    }
    catch (System.NullReferenceException)
    {
                        EphesiaError.errorNoCodigo("El código no se puede
                        generar.");
         this.pctMuestraEscala.Image = null;
         Cursor = Cursors.Default;
    }
    catch (IndexOutOfRangeException)
    {
        EphesiaError.errorFueraDeRangoCadena("Demasiadas
                          generaciones para la memoria del sistema.");
        indexOutOfRangeException = 1;
        Cursor = Cursors.Default;
    }
    catch (System.Exception err)
    {
        EphesiaError.errorDesconocido(err);
        Cursor = Cursors.Default;
    }
}

private void btnGuardar_MouseDown(object sender,
                MouseEventArgs e)
{
    guardarCodigoDATAMATRIX(1);
}

private void guardarCodigoDATAMATRIX(byte respuesta)
{
    //respuesta 1= muestra mensaje, 2= no lo muestra
    try
    {
        Cursor = Cursors.WaitCursor;
        string nombreTexto = this.dfText.Text + "-" +
                          this.dfMargin.Text + "-" +
                          this.dfPixelSize.Text
                + "-" + this.cbMode.SelectedIndex + "-" +
                          this.cbSymbolSize.SelectedIndex;
        Cursor = Cursors.Default;
    }
    catch (System.IO.IOException err)
```

```
    {
        Cursor = Cursors.Default;
        EphesiaError.errorF();
    }
}

private void btnVolverGen_MouseDown(object sender,
            MouseEventArgs e)
{
    generarPrimeraVez();
}

private void btnVolverGenerarEspecieAZTEC_MouseDown(object
            sender, MouseEventArgs e)
{
    generarPrimeraVez();
}

private void imprimirCodigo()
{
    printDialog1.Document = printDocument1;
    DialogResult respuesta = printDialog1.ShowDialog();
    if (respuesta == DialogResult.OK)
    {
        printDocument1.Print();
    }
}

private void btnPrint_MouseDown(object sender, MouseEventArgs
            e)
{
    tabPulsada = 1;
}

private void btnImprimir_MouseDown(object sender,
            MouseEventArgs e)
{
    tabPulsada = 2;
}

private void btnImprimirAZTEC_MouseDown(object sender,
            MouseEventArgs e)
{
    tabPulsada = 3;
}

private void btnPrint_MouseUp(object sender, MouseEventArgs
            e)
{
    imprimirCodigo();
}

private void btnImprimir_MouseUp(object sender,
            MouseEventArgs e)
{
    imprimirCodigo();
}

private void btnImprimirAZTEC_MouseUp(object sender,
```

```
                MouseEventArgs e)
{
   imprimirCodigo();
}

private void pctBarraColor_MouseDown(object sender,
                MouseEventArgs e)
{
   Bitmap image1 = new Bitmap(pctBarraColor.Image);
   if (this.rbtCodigo.Checked == true)
   {
      pixelColor = image1.GetPixel(e.X, e.Y);
      this.lblColoCodigo.BackColor = pixelColor;
   }
   if (this.rbtFondo.Checked == true)
   {
      pixelColor1 = image1.GetPixel(e.X, e.Y);
      this.lblColor.BackColor = pixelColor1;
   }
}

private void btnContar_MouseDown(object sender,
                MouseEventArgs e)
{
   contQR();
}

private void contQR()
{
   this.txtContar.Text =
    this.txtEncodeData.Text.Trim().Length.ToString();
}

private void btnContarDataMa_MouseDown(object sender,
                MouseEventArgs e)
{
   contDT();
}

private void contDT()
{
   this.txtContaDataMa.Text =
                this.dfText.Text.Trim().Length.ToString();
}

private void btnGenerarMenos_MouseDown(object sender,
                MouseEventArgs e)
{

   this.dfText.Text =
                EphesiaUtilidades.devolverTextoNumeroRadio(
                this.dfText.Text, numeroRbtDM);
   generarDataMatrix(this.dfText.Text,
                this.cbMode.SelectedIndex,
                (this.cbSymbolSize.SelectedIndex - 2));
}

private void btnGemerarMas_MouseDown(object sender,
                MouseEventArgs e)
```

```
{
   for (int i = 1; i <= numeroRbtDM; i++)
   {
      this.dfText.Text = this.dfText.Text +
                  this.txtCaracterDTM.Text;
   }
   generarDataMatrix(this.dfText.Text,
            this.cbMode.SelectedIndex,
            (this.cbSymbolSize.SelectedIndex - 2));
}

private void btnGenerarMasQR_MouseDown(object sender,
            MouseEventArgs e)
{
   for (int i = 1; i <= numeroRbt; i++)
   {
      this.txtEncodeData.Text = this.txtEncodeData.Text +
                  this.txtCaracter.Text;
   }
   GennerarQR();
}

      private void btnMasAZTEC_MouseDown(object sender,
            MouseEventArgs e)
{

   for (int i = 1; i <= numeroRbtAZTEC; i++)
   {
      this.txtAZTEC.Text = this.txtAZTEC.Text +
                  this.txtPatronAZTEC.Text;
   }
   btnGenerarAZTEC_MouseDown(sender, e);
}

private void btnMenosAZTEC_MouseDown(object sender,
            MouseEventArgs e)
{

   this.txtAZTEC.Text =
            EphesiaUtilidades.devolverTextoNumeroRadio(this.txtAZTEC.
                  Text, numeroRbtAZTEC);
   btnGenerarAZTEC_MouseDown(sender, e);
}

private void btnGenerarMenosQR_MouseDown(object sender,
            MouseEventArgs e)
{

   this.txtEncodeData.Text =
            EphesiaUtilidades.devolverTextoNumeroRadio(
                  this.txtEncodeData.Text, numeroRbt);
   GennerarQR();
}

private void rbtPor1_CheckedChanged(object sender, EventArgs
            e)
{
   Cursor = Cursors.WaitCursor;
   numeroRbt = Convert.ToInt16(rbtPor1.Text);
```

```
    Cursor = Cursors.Default;
}

private void rbtPor10_CheckedChanged(object sender, EventArgs
            e)
{
    Cursor = Cursors.WaitCursor;
    numeroRbt = Convert.ToInt16(rbtPor10.Text);
    Cursor = Cursors.Default;
}

private void rbtPor25_CheckedChanged(object sender, EventArgs
            e)
{
    Cursor = Cursors.WaitCursor;
    numeroRbt = Convert.ToInt16(rbtPor25.Text);
    Cursor = Cursors.Default;
}

private void rbtPor50_CheckedChanged(object sender, EventArgs
            e)
{
    Cursor = Cursors.WaitCursor;
    numeroRbt = Convert.ToInt16(rbtPor50.Text);
    Cursor = Cursors.Default;
}

private void rbtPor100_CheckedChanged(object sender,
            EventArgs e)
{
    Cursor = Cursors.WaitCursor;
    numeroRbt = Convert.ToInt16(rbtPor100.Text);
    Cursor = Cursors.Default;
}

private void rbtPor500_CheckedChanged(object sender,
            EventArgs e)
{
    Cursor = Cursors.WaitCursor;
    numeroRbt = Convert.ToInt16(rbtPor500.Text);
    Cursor = Cursors.Default;
}

private void rbt1_CheckedChanged(object sender, EventArgs e)
{
    Cursor = Cursors.WaitCursor;
    numeroRbtDM = Convert.ToInt16(rbt1.Text);
    Cursor = Cursors.Default;
}

private void rbt10_CheckedChanged(object sender, EventArgs e)
{
    Cursor = Cursors.WaitCursor;
    numeroRbtDM = Convert.ToInt16(rbt10.Text);
    Cursor = Cursors.Default;
}

private void rbt25_CheckedChanged(object sender, EventArgs e)
{
```

```
    Cursor = Cursors.WaitCursor;
    numeroRbtDM = Convert.ToInt16(rbt25.Text);
    Cursor = Cursors.Default;
}

private void rbt50_CheckedChanged(object sender, EventArgs e)
{
    Cursor = Cursors.WaitCursor;
    numeroRbtDM = Convert.ToInt16(rbt50.Text);
    Cursor = Cursors.Default;
}

private void rbt100_CheckedChanged(object sender, EventArgs
            e)
{
    Cursor = Cursors.WaitCursor;
    numeroRbtDM = Convert.ToInt16(rbt100.Text);
    Cursor = Cursors.Default;
}

private void rbt500_CheckedChanged(object sender, EventArgs
            e)
{
    Cursor = Cursors.WaitCursor;
    numeroRbtDM = Convert.ToInt16(rbt500.Text);
    Cursor = Cursors.Default;
}

private void rbt1AZTEC_CheckedChanged(object sender,
            EventArgs e)
{
    Cursor = Cursors.WaitCursor;
    numeroRbtAZTEC = Convert.ToInt16(rbt1AZTEC.Text);
    Cursor = Cursors.Default;
}

private void rbt10AZTEC_CheckedChanged(object sender,
            EventArgs e)
{
    Cursor = Cursors.WaitCursor;
    numeroRbtAZTEC = Convert.ToInt16(rbt10AZTEC.Text);
    Cursor = Cursors.Default;
}

private void rbt25AZTEC_CheckedChanged(object sender,
            EventArgs e)
{
    Cursor = Cursors.WaitCursor;
    numeroRbtAZTEC = Convert.ToInt16(rbt25AZTEC.Text);
    Cursor = Cursors.Default;
}

private void rbt50AZTEC_CheckedChanged(object sender,
            EventArgs e)
{
    Cursor = Cursors.WaitCursor;
    numeroRbtAZTEC = Convert.ToInt16(rbt50AZTEC.Text);
    Cursor = Cursors.Default;
}
```

```
private void rbt100AZTEC_CheckedChanged(object sender,
            EventArgs e)
{
   Cursor = Cursors.WaitCursor;
   numeroRbtAZTEC = Convert.ToInt16(rbt100AZTEC.Text);
   Cursor = Cursors.Default;
}

private void rbt500AZTEC_CheckedChanged(object sender,
            EventArgs e)
{
   Cursor = Cursors.WaitCursor;
   numeroRbtAZTEC = Convert.ToInt16(rbt500AZTEC.Text);
   Cursor = Cursors.Default;
}

private void btnEstablecerMaxLe_MouseDown(object sender,
            MouseEventArgs e)
{
   this.dfText.MaxLength = 999999999;
}

private void btnMaxAZTEC_MouseDown(object sender,
            MouseEventArgs e)
{
   this.txtAZTEC.MaxLength = 3748;
}

private void btnEstablecerMaximo_MouseDown(object sender,
            MouseEventArgs e)
{
   try
   {

      this.txtCronometradoDT.Text =
                EphesiaUtilidades.sumarCronometrados(1);
      if (this.rdbN.Checked == true)
      {
         this.dfText.Text =
            EphesiaUtilidades.
            devolvelCadenaNumeroAleatorio();
      }
      else if (this.rbtT.Checked == true)
      {
         this.dfText.Text =
                  EphesiaUtilidades.devolverCadenaAleatoria();
      }
      else if (this.rbtVacio.Checked == true)
      {
         this.dfText.Text = this.txtCaracterDTM.Text;
      }
      generarDataMatrix(this.dfText.Text,
                  this.cbMode.SelectedIndex,
                        (this.cbSymbolSize.SelectedIndex - 2));
      Cursor = Cursors.WaitCursor;
      for (int i = 1;
            this.pctMuestraEscala.Image != null; i++)
      {
```

```
            try
            {
               if (this.rdbN.Checked == true)
               {
                  this.dfText.Text = this.dfText.Text +
                              EphesiaUtilidades.
                              devolvelCadenaNumeroAleatorio();
               }
               else if (this.rbtT.Checked == true)
               {
                  this.dfText.Text = this.dfText.Text +
                              EphesiaUtilidades.
                              devolverCadenaAleatoria();
               }
               else if (this.rbtVacio.Checked == true)
               {
                  this.dfText.Text = this.dfText.Text +
                                          this.txtCaracterDTM.Text;
               }
               generarDataMatrix(this.dfText.Text,
                                 this.cbMode.SelectedIndex,
                                 (this.cbSymbolSize.SelectedIndex - 2));
               if (indexOutOfRangeException == 1)
               {
                  indexOutOfRangeException = 0;
                  return;
               }
               this.pctMuestraEscala.Refresh();
               this.dfText.Refresh();
               this.lblNumero.Refresh();
            }
            catch (IndexOutOfRangeException)
            {
                     EphesiaError.errorFueraDeRangoCadena(
                     "Demasiadas generaciones para la memoria del
                     sistema.");
               Cursor = Cursors.Default;
               return;
            }
         }
         this.btnGenerarMenos_MouseDown(sender, e);
         Cursor = Cursors.Default;
         this.txtCronometradoDT.Text =
                        EphesiaUtilidades.sumarCronometrados(0);
      }
      catch (System.Exception err)
      {
         Cursor = Cursors.Default;
         EphesiaUtilidades.cronometro(0);
         this.txtCronometradoDT.Text = "";
         EphesiaError.errorDesconocido(err);
      }
   }

   private void btnGenerarMax_MouseDown(object sender,
                MouseEventArgs e)
   {
      try
      {
```

```
            this.txtCrono.Text =
                                    EphesiaUtilidades.sumarCronometrados(1);
            if (this.rbtNumeroQr.Checked == true)
            {
               this.txtEncodeData.Text =
                        EphesiaUtilidades.
                        devolvelCadenaNumeroAleatorio();
            }
            else if (this.rbtTextoQr.Checked == true)
            {
               this.txtEncodeData.Text =
                      EphesiaUtilidades.devolverCadenaAleatoria();
            }
            else if (this.rbtNadaQr.Checked == true)
            {
               this.txtEncodeData.Text = this.txtCaracter.Text;
            }
            GennerarQR();
            Cursor = Cursors.WaitCursor;
            for (int i = 1; this.pctMuestra.Image != null; i++)
            {
               if (this.rbtNumeroQr.Checked == true)
               {
                  this.txtEncodeData.Text =
                        this.txtEncodeData.Text + EphesiaUtilidades.
                        devolvelCadenaNumeroAleatorio();
               }
               else if (this.rbtTextoQr.Checked == true)
               {
                  this.txtEncodeData.Text =
                          this.txtEncodeData.Text +
                          EphesiaUtilidades.
                          devolverCadenaAleatoria();
               }
               else if (this.rbtNadaQr.Checked == true)
               {
                  this.txtEncodeData.Text =
                                this.txtEncodeData.Text + this.txtCaracter.Text;
               }
               GennerarQR();
               this.txtEncodeData.Refresh();
               this.lblNumero.Refresh();
            }
            this.btnGenerarMenosQR_MouseDown(sender, e);
            Cursor = Cursors.Default;
            this.txtCrono.Text =
                      EphesiaUtilidades.sumarCronometrados(0);
         }
         catch (System.Exception err)
         {
            Cursor = Cursors.Default;
            EphesiaError.errorDesconocido(err);
         }
      }

      private void btnMayorAZTEC_MouseDown(object sender,
                  MouseEventArgs e)
      {
         try
```

```
    {
        this.txtCronoAZTEC.Text =
                    EphesiaUtilidades.sumarCronometrados(1);
        if (this.rbtNumeroAZTEC.Checked == true)
        {
            this.txtAZTEC.Text =
                    EphesiaUtilidades.
                    devolvelCadenaNumeroAleatorio();
        }
        else if (this.rbtTextoAZTEC.Checked == true)
        {
            this.txtAZTEC.Text =
                EphesiaUtilidades.devolverCadenaAleatoria();
        }
        else if (this.rbtDefectoAZTEC.Checked == true)
        {
            this.txtAZTEC.Text = this.txtPatronAZTEC.Text;
        }
        this.btnGenerarAZTEC_MouseDown(sender, e);
        Cursor = Cursors.WaitCursor;
        for (int i = 1;
                    this.pctMuestraAZTEC.Image != null; i++)
        {
            if (this.rbtNumeroAZTEC.Checked == true)
            {
                this.txtAZTEC.Text = this.txtAZTEC.Text +
                        EphesiaUtilidades.
                        devolvelCadenaNumeroAleatorio();
            }
            else if (this.rbtTextoAZTEC.Checked == true)
            {
                this.txtAZTEC.Text = this.txtAZTEC.Text +
                          EphesiaUtilidades.
                          devolverCadenaAleatoria();
            }
            else if (this.rbtDefectoAZTEC.Checked == true)
            {
                 this.txtAZTEC.Text = this.txtAZTEC.Text +
                            this.txtPatronAZTEC.Text;
            }
            this.btnGenerarAZTEC_MouseDown(sender, e);
            this.pctMuestraAZTEC.Refresh();
            this.txtAZTEC.Refresh();
            this.lblNumero.Refresh();
        }
        this.btnMenosAZTEC_MouseDown(sender, e);
        Cursor = Cursors.Default;
        this.txtCronoAZTEC.Text =
                        EphesiaUtilidades.sumarCronometrados(0);
    }
    catch (System.Exception err)
    {
        Cursor = Cursors.Default;
        EphesiaError.errorDesconocido(err);
    }
}

private void txtContaDataMa_TextChanged(object sender,
                EventArgs e)
```

```
{
   this.lblNumero.Text = this.txtContaDataMa.Text;
}

private void txtContar_TextChanged(object sender,
    EventArgs e)
{
   this.lblNumero.Text = this.txtContar.Text;
}

private void txtNumeroCaracteresAZTEC_TextChanged(object
            sender, EventArgs e)
{
   this.lblNumero.Text = this.txtNumeroCaracteresAZTEC.Text;
}

private void btnInvertir_MouseDown(object sender,
            MouseEventArgs e)
{
   Color color1 = this.lblColor.BackColor;
   pixelColor= color1;
   Color color2 = this.lblColoCodigo.BackColor;
   pixelColor1 = color2;
   lblColoCodigo.BackColor = color1;
   lblColor.BackColor = color2;
   GennerarQR();
}

private void btnBN_MouseDown(object sender, MouseEventArgs e)
{
   pixelColor = Color.Black;
   this.lblColoCodigo.BackColor = pixelColor;
   pixelColor1 = Color.White;
   this.lblColor.BackColor = pixelColor1;
   GennerarQR();
}

private void btnEncode_MouseDown(object sender,
            MouseEventArgs e)
{
    GennerarQR();
}

private void tabMain_Selecting(object sender,
            TabControlCancelEventArgs e)
{
   switch (e.TabPageIndex)
   {
      case 0:
         this.lblNumero.Text = this.txtContar.Text;
         tamañoImagen = this.pctEncode.Size;
         this.hScrollBar3.Value = tamañoImagen.Height;
         break;
      case 1:
         this.lblNumero.Text = this.txtContaDataMa.Text;
         tamañoImagen = this.pctDataMatrix.Size;
         this.hScrollBar2.Value = tamañoImagen.Height;
         break;
      case 2:
```

```
            this.lblNumero.Text =
             this.txtNumeroCaracteresAZTEC.Text;
            tamañoImagen = this.pctCodigoAZTEC.Size;
            this.hScrollBarAZTEC.Value = tamañoImagen.Height;
            break;
        case 3:
            this.lblNumero.Text = "";
            break;
        case 4:
            this.lblNumero.Text = "";
            break;
    }
}

private void txtEncodeData_TextChanged(object sender,
                EventArgs e)
{
    contQR();
}

private void dfText_TextChanged(object sender, EventArgs e)
{
    contDT();
}

private void txtAZTEC_TextChanged(object sender, EventArgs e)
{
    contAZTEC();
}

private void txtDatosCam_TextChanged(object sender, EventArgs
                e)
{
    this.lblNumeroCaracteresCam.Text =
                this.txtDatosCam.Text.Length.ToString();
}

private void btnEstablecerMaximoYGenerar_MouseDown(object
                sender, MouseEventArgs e)
{
    try
    {
        Cursor = Cursors.WaitCursor;
        this.btnEstablecerMaximo_MouseDown(sender, e);
        this.btnGenerar_MouseDown(sender, e);
        string contenidoCombo =
                        this.cbSymbolSize.SelectedItem.ToString();
        if (contenidoCombo.Contains("Código de "))
        {
            contenidoCombo =
                        contenidoCombo.Replace("Código de ", "");
        }
        else if (contenidoCombo.Contains("Código auto "))
        {
            contenidoCombo =
                          contenidoCombo.Replace("Código ", "");
        }
        contenidoCombo =
                        CadenasEphesia.quitarEspaciosEnBlancoCadena
```

```
                    (contenidoCombo);
        if (this.chkSobrescrbirPortapapelesQr.
                    CheckState == CheckState.Checked)
        {
            llenarPortapapelesDT(1, contenidoCombo);
        }
        else if (this.chkSobrescribirPortapapeles.CheckState
                        == CheckState.Unchecked)
        {
            llenarPortapapelesDT(2, contenidoCombo);
        }
        llenarListaPortapapelesDT(contenidoCombo);
    }
    catch (Exception err)
    {
        Cursor = Cursors.Default;
        EphesiaError.errorDesconocido(err);
    }
}

private void btnEstablecerMaximoYGenerarQr_MouseDown(object
            sender, MouseEventArgs e)
{
    try
    {
        Cursor = Cursors.WaitCursor;
        this.btnGenerarMax_MouseDown(sender, e);
        GennerarQR();
        if (this.chkSobrescribirPortapapeles.CheckState ==
                    CheckState.Checked)
        {
            llenarPortapapeles(1);
        }
        else if (this.chkSobrescribirPortapapeles.CheckState
                    == CheckState.Unchecked)
        {
            llenarPortapapeles(2);
        }
        llenarListaPortapapeles();
        Cursor = Cursors.Default;
    }
    catch (Exception err)
    {
        Cursor = Cursors.Default;
        EphesiaError.errorDesconocido(err);
    }
}

private void btnGenenrarMaximoAZTEC_MouseDown(object sender,
            MouseEventArgs e)
{
    try
    {
        Cursor = Cursors.WaitCursor;
        this.btnMayorAZTEC_MouseDown(sender, e);
        this.btnGenerarAZTEC_MouseDown(sender, e);
        string contenidoCombo =
            this.cboAZTEC.SelectedItem.ToString();
        if (contenidoCombo.Contains("Código de "))
```

```
        {
            contenidoCombo =
                    contenidoCombo.Replace("Código de ", "");
        }
        contenidoCombo =
                CadenasEphesia.
                quitarEspaciosEnBlancoCadena(contenidoCombo);
        if (this.chkPortapapelesAZTEC.CheckState ==
                        CheckState.Checked)
        {
            llenarPortapapelesAZTEC(1, contenidoCombo);
        }
        else if (this.chkPortapapelesAZTEC.CheckState ==
                        CheckState.Unchecked)
        {
            llenarPortapapelesAZTEC(2, contenidoCombo);
        }
        llenarListaPortapapelesAZTEC(contenidoCombo);
    }
    catch (Exception err)
    {
        Cursor = Cursors.Default;
        EphesiaError.errorDesconocido(err);
    }
}

private void llenarPortapapeles(byte tipoOp)
{
    //tipoOP= 1, portapapeles lleno, 2= vacío
    string textoPegar = "QRCODE " +
                this.cboVersion.SelectedItem.ToString() + " " +

                        EphesiaUtilidades.
                        devolverNombreRadioBotonesTruePanel(this.panel6) +
          " " +
                        EphesiaUtilidades.
                        devolverNombreRadioBotonesTruePanel(this.panel5) +
         " " + this.txtSize.Text + " " +
                        this.txtContar.Text + " " + this.txtCrono.Text;
    if (tipoOp == 1)
    {
        if (Clipboard.GetText().Trim() == "")
        {
            textoPegar = textoPegar.Trim();
            Clipboard.SetText(textoPegar);
        }
        else if (Clipboard.GetText().Trim() != "")
        {
            Clipboard.SetText(Clipboard.GetText() +
                                    "\r\n" + textoPegar);
        }
    }
    else if (tipoOp == 2)
    {
        textoPegar = textoPegar.Trim();
        Clipboard.SetText(textoPegar);
    }
}
```

```
private void llenarListaPortapapeles()
{
   string textoPegar = "QRCODE " +
               this.cboVersion.SelectedItem.ToString() + " " +
               EphesiaUtilidades.devolverNombreRadioBotonesTruePanel
               (this.panel6) +" " +
               EphesiaUtilidades.devolverNombreRadioBotonesTruePanel
               (this.panel5) +" " + this.txtSize.Text + " " + this.txtContar.Text + " " +
               this.txtCrono.Text;
    textoPegar = textoPegar.Trim();
    this.lstListaProcesos.Items.Add(textoPegar);
}

private void llenarPortapapelesDT(byte tipoOp, string
            contenidoCombo)
{
   //tipoOP= 1, portapapeles lleno, 2= vacío
   string textoPegar = "DATAMATRIX " +
               this.cbMode.SelectedItem.ToString() +
      " " + contenidoCombo + " " + this.dfPixelSize.Text +
                  " " + this.dfMargin.Text +
      " " + this.txtContaDataMa.Text + " " +
               this.txtCronometradoDT.Text;
   Clipboard.SetText(Clipboard.GetText() + "\r\n" +
                  textoPegar);
   if (tipoOp == 1)
   {
      if (Clipboard.GetText().Trim() == "")
      {
         textoPegar = textoPegar.Trim();
         Clipboard.SetText(textoPegar);
      }
      else if (Clipboard.GetText().Trim() != "")
      {
         Clipboard.SetText(Clipboard.GetText() + "\r\n" +
                           textoPegar);
      }
   }
   else if (tipoOp == 2)
   {
      textoPegar = textoPegar.Trim();
      Clipboard.SetText(textoPegar);
   }
}

private void llenarPortapapelesAZTEC(byte tipoOp, string
            contenidoCombo)
{
   //tipoOP= 1, portapapeles lleno, 2= vacío
   string textoPegar = "AZTEC " + contenidoCombo + " " +
   this.txtNumeroCaracteresAZTEC.Text + " " +
               this.txtCronoAZTEC.Text;
   Clipboard.SetText(Clipboard.GetText() + "\r\n" +
               textoPegar);
   if (tipoOp == 1)
   {
      if (Clipboard.GetText().Trim() == "")
      {
         textoPegar = textoPegar.Trim();
```

```
            Clipboard.SetText(textoPegar);
        }
        else if (Clipboard.GetText().Trim() != "")
        {
            Clipboard.SetText(Clipboard.GetText() + "\r\n" +
                                textoPegar);
        }
    }
    else if (tipoOp == 2)
    {
        textoPegar = textoPegar.Trim();
        Clipboard.SetText(textoPegar);
    }
}

private void llenarListaPortapapelesDT(string contenidoCombo)
{
    string textoPegar = "DATAMATRIX " +
            this.cbMode.SelectedItem.ToString() +
             " " + contenidoCombo + " " + this.dfPixelSize.Text +
            " " + this.dfMargin.Text +
            " " + this.txtContaDataMa.Text + " " + this.txtCronometradoDT.Text;
    Clipboard.SetText(Clipboard.GetText() + "\r\n" +
            textoPegar);
    textoPegar = textoPegar.Trim();
    this.lstDATAMATRIX.Items.Add(textoPegar);
}

private void llenarListaPortapapelesAZTEC(string
            contenidoCombo)
{
    string textoPegar = "AZTEC " + contenidoCombo + " " +
    this.txtNumeroCaracteresAZTEC.Text + " " +
            this.txtCronoAZTEC.Text;
    textoPegar = textoPegar.Trim();
    this.lstListaAZTEC.Items.Add(textoPegar);
}

private void btnDecode_MouseDown(object sender,
            MouseEventArgs e)
{
    try
    {
        Cursor = Cursors.WaitCursor;

        Image imagen = (Bitmap)Bitmap.FromFile(nombreRuta);
        if (this.rbtQr.Checked == true)
        {
            if (nombreRuta.Contains("QRCODE"))
            {
                this.txtDecodedData.Text =
                        EphesiaUtilidades.decodificarQR(imagen);
            }
            else
            {
                EphesiaError.errorCodigo2D("QRCODE");
                this.txtDecodedData.Text =
                        EphesiaUtilidades.
                        decodificarUniversal(nombreRuta, "");
```

```
                this.rbtUniversal.Checked = true;
            }
        }
        if (this.rbtDatamatrix.Checked == true)
        {
            if (nombreRuta.Contains("DATAMATRIX"))
            {
                this.txtDecodedData.Text =
                        EphesiaUtilidades.
                        decodificarDATAMATRIX(nombreRuta);
            }
            else
            {
                EphesiaError.errorCodigo2D("DATAMATRIX");
                this.txtDecodedData.Text =
                        EphesiaUtilidades.
                        decodificarUniversal(nombreRuta, "");
                this.rbtUniversal.Checked = true;
            }
        }
        if (this.rbtAZTEC.Checked == true)
        {
            if (nombreRuta.Contains("AZTEC"))
            {
                this.txtDecodedData.Text =
                        EphesiaUtilidades.
                        decodificarUniversal(nombreRuta, " AZTEC");
            }
            else
            {
                EphesiaError.errorCodigo2D("AZTEC");
                this.txtDecodedData.Text =
                    EphesiaUtilidades.
                    decodificarUniversal(nombreRuta, "");
                this.rbtUniversal.Checked = true;
            }
        }
        if (this.rbtUniversal.Checked == true)
             this.txtDecodedData.Text =
                EphesiaUtilidades.decodificarUniversal
                (nombreRuta, "");
             this.lblCaracter.Text =
                    this.txtDecodedData.Text.Length.ToString();
        Cursor = Cursors.Default;
    }
    catch (Exception err)
    {
        Cursor = Cursors.Default;
        EphesiaError.errorDesconocido(err);
    }
}

private void btnOpen_MouseDown(object sender,
    MouseEventArgs e)
{
    if (this.rbtQr.Checked == true)
    {
        openFileDialog1.InitialDirectory = EphesiaApp.AP +
                    "\\EPHESIA\\Codigos\\QRCODE\\";
```

```
    }
    else if (this.rbtDatamatrix.Checked == true)
    {
        openFileDialog1.InitialDirectory = EphesiaApp.AP +
                        "\\EPHESIA\\Codigos\\DATAMATRIX\\";
    }
    else if (this.rbtAZTEC.Checked == true)
    {
        openFileDialog1.InitialDirectory = EphesiaApp.AP +
                        "\\EPHESIA\\Codigos\\AZTEC\\";
    }
    else if (this.rbtUniversal.Checked == true)
    {
        openFileDialog1.InitialDirectory = EphesiaApp.AP +
                        "\\EPHESIA\\Codigos\\";
    }
    openFileDialog1.Filter = "JPg Image|*.jpg|Bitmap
            Image|*.bmp|Gif Image|*.gif|PNG Image|*.png|
            All files (*.*)|*.*";
    openFileDialog1.FilterIndex = 1;
    openFileDialog1.RestoreDirectory = true;
    openFileDialog1.FileName = string.Empty;
    if (openFileDialog1.ShowDialog() == DialogResult.OK)
    {
        nombreRuta = openFileDialog1.FileName;
        this.txtRuta.Text = nombreRuta;
        this.picDecode.Image = new Bitmap(nombreRuta);
                    EphesiaUtilidades.CentrarControlEnPanel(
                    this.panDecodificar, this.picDecode);
        this.txtDecodedData.Text = "";
        tamañoImagen = this.picDecode.Size;
        this.hScrollBar1.Value = tamañoImagen.Height;
        this.hScrollBar1.Enabled = true;
    }
}

private void btnAbrirYDecodificar_MouseDown(object sender,
            MouseEventArgs e)
{
    btnOpen_MouseDown(sender, e);
    btnDecode_MouseDown(sender, e);
}

private void btnGenerarSecuencia_MouseDown(object sender,
            MouseEventArgs e)
{
    Cursor = Cursors.WaitCursor;

    int principioSecuencia =
            Convert.ToInt16(this.txtPartidaSecuencia.Text);
    int finalSecuencia =
            Convert.ToInt16(this.txtFinalSecuencia.Text);
    if (principioSecuencia < finalSecuencia)
    {
        int j = 0;
        for (int i = principioSecuencia;
                        i <= finalSecuencia; i++)
        {
            string cadenaSecuencia =
```

```
                    this.txtCadenaPrefijo.Text.Trim();
            if (cadenaSecuencia != "")
            {
                this.dfText.Text = cadenaSecuencia + "-" +
                 i.ToString();
            }
            else if (cadenaSecuencia == "")
            {
                this.dfText.Text = i.ToString();
            }
            this.dfText.Refresh();
            this.pctMuestraEscala.Refresh();
            generarDataMatrix(this.dfText.Text,
             this.cbMode.SelectedIndex, (this.cbSymbolSize.SelectedIndex - 2));
            try
            {
                guardarCodigoDATAMATRIX(2);
            }
            catch (System.NullReferenceException)
            {

              EphesiaError.errorGuardar(
              " No se puede guardar ");
              this.lblNumero.Text = "";
              Cursor = Cursors.Default;
              return;
            }
            j++;
            this.lblNumero.Text = j.ToString();
            this.lblNumero.Refresh();
          }
          Cursor = Cursors.Default;
          }
      else if (principioSecuencia == finalSecuencia)
      {
          Cursor = Cursors.Default;
                      EphesiaError.finalsecuencia();
      }
      else if (principioSecuencia > finalSecuencia)
      {
          Cursor = Cursors.Default;
                      EphesiaError. finalsecuencia ();
      }
      Cursor = Cursors.Default;
}

private void btnGenerarSecuenciaAZTEC_MouseDown(object
              sender, MouseEventArgs e)
{
      Cursor = Cursors.WaitCursor;
      int principioSecuencia =
                          Convert.ToInt16(this.txtInicioAZTEC.Text);
      int finalSecuencia =
                          Convert.ToInt16(this.txtFinalAZTEC.Text);
      if (principioSecuencia < finalSecuencia)
      {
          int j = 0;
          for (int i = principioSecuencia; i
                          <= finalSecuencia; i++)
```

```
        {
            string cadenaSecuencia =
            this.txtCadenaContenidoAZTEC.Text.Trim();
            if (cadenaSecuencia != "")
            {
                this.txtAZTEC.Text = cadenaSecuencia + "-" +
                                     i.ToString();
            }
            else if (cadenaSecuencia == "")
            {
                this.txtAZTEC.Text = i.ToString();
            }
            this.txtAZTEC.Refresh();
            this.pctMuestraAZTEC.Refresh();
            this.btnGenerarAZTEC_MouseDown(sender, e);
            EphesiaUtilidades.
                guardarCodigosAutomaticamente(
            this.txtAZTEC.Text.Trim(),
            this.cboAZTEC.SelectedItem.ToString().Substring(10, 3),
                  CadenasEphesia.usuarioApellido,
            this.pctCodigoAZTEC.Image, "AZTEC");
            j++;
            this.lblNumero.Text = j.ToString();
            this.lblNumero.Refresh();
        }
        Cursor = Cursors.Default;
    }
    else if (principioSecuencia == finalSecuencia)
    {
        Cursor = Cursors.Default;
                    EphesiaError.errorDecodificacion();
         this.lblNumero.Text = "";
    }
    else if (principioSecuencia > finalSecuencia)
    {
        Cursor = Cursors.Default;
                    EphesiaError.errorSecuencia();
    }
    Cursor = Cursors.Default;
}

private void txtPartidaSecuencia_KeyUp(object sender,
             KeyEventArgs e)
{
    numerosEphesia.comprobarNumeroTecleadoTexto(e,
             this.txtPartidaSecuencia);
}

private void txtFinalSecuencia_KeyUp(object sender,
             KeyEventArgs e)
{
    numerosEphesia.comprobarNumeroTecleadoTexto(e,
             this.txtFinalSecuencia);
}

private void txtPrincipioSecuenciaQR_KeyUp(object sender,
             KeyEventArgs e)
{
    numerosEphesia.comprobarNumeroTecleadoTexto(e,
```

```
            this.txtPrincipioSecuenciaQR);
}

private void txtFinalSecuenciaQR_KeyUp(object sender,
            KeyEventArgs e)
{
   numerosEphesia.comprobarNumeroTecleadoTexto(e,
            this.txtPrincipioSecuenciaQR);
}

private void txtInicioAZTEC_KeyUp(object sender,
            KeyEventArgs e)
{
   numerosEphesia.comprobarNumeroTecleadoTexto(e,
            this.txtInicioAZTEC);
}

private void txtFinalAZTEC_KeyUp(object sender,
            KeyEventArgs e)
{
   numerosEphesia.comprobarNumeroTecleadoTexto(e,
            this.txtFinalAZTEC);
}

private void btnGenerarSecuenciaQR_MouseDown(object sender,
            MouseEventArgs e)
{
   Cursor = Cursors.WaitCursor;

   int principioSecuencia =
            Convert.ToInt16(this.txtPrincipioSecuenciaQR.Text);
   int finalSecuencia =
            Convert.ToInt16(this.txtFinalSecuenciaQR.Text);
   string cadenaSecuencia =
            this.txtCadenaSecuencia.Text.Trim();
   if (principioSecuencia < finalSecuencia)
   {
      int j = 0;
      for (int i = principioSecuencia; i <=
                  finalSecuencia; i++)
      {
         if (cadenaSecuencia != "")
         {
            this.txtEncodeData.Text =
                        cadenaSecuencia + "-" + i.ToString();
         }
         else if (cadenaSecuencia == "")
         {
            this.txtEncodeData.Text = i.ToString();
         }
         this.txtEncodeData.Refresh();
         this.pctMuestra.Refresh();
         this.pctEncode.Refresh();
         GennerarQR();

          EphesiaUtilidades.
          guardarCodigosAutomaticamente
          (this.txtEncodeData.Text.Trim(),
           this.cboVersion.Text,
```

```
            this.txtSize.Text.Trim(),
            this.pctEncode.Image, "QRCODE");
            j++;
            this.lblNumero.Text = j.ToString();
            this.lblNumero.Refresh();
        }
        Cursor = Cursors.Default;
    }
    else if (principioSecuencia == finalSecuencia)
    {
                    EphesiaError.errorSecuencia();
    }
    else if (principioSecuencia > finalSecuencia)
    {
        Cursor = Cursors.Default;
                    EphesiaError.errorSecuencia();
    }
    Cursor = Cursors.Default;
}

private void btnGuardarQR_MouseDown(object sender,
            MouseEventArgs e)
{
    Cursor = Cursors.WaitCursor;
    EphesiaUtilidades.guardarCodigo("QRCODE", saveFileDialog1,
            this.txtEncodeData.Text.Trim(),
        this.cboVersion.Text, this.txtSize.Text.Trim(),
            this.pctEncode.Image);
    Cursor = Cursors.Default;
}

private void hScrollBar1_Scroll(object sender,
            ScrollEventArgs e)
{
    cambiarTamañoConScroll(this.picDecode,
            this.hScrollBar1.Value, e, this.panDecodificar);
}

private void hScrollBar2_Scroll(object sender,
            ScrollEventArgs e)
{
    cambiarTamañoConScroll(this.pctDataMatrix,
            this.hScrollBar2.Value, e, this.panCodificarDTM );
}

private void hScrollBar3_Scroll(object sender,
            ScrollEventArgs e)
{
    cambiarTamañoConScroll(this.pctEncode,
            this.hScrollBar3.Value, e, this.panCodificarQR );
}

private void hScrollBarAZTEC_Scroll(object sender,
            ScrollEventArgs e)
{
    cambiarTamañoConScroll(this.pctCodigoAZTEC,
            this.hScrollBarAZTEC.Value, e, this.pnlAZTEC );
}
```

```
private void cambiarTamañoConScroll(PictureBox pic, int
            valorScroll, ScrollEventArgs e, Panel pan)
{
   pic.SizeMode = PictureBoxSizeMode.StretchImage;
   if (e.NewValue > e.OldValue)
   {
      pic.Size = new Size(tamañoImagen.Width + valorScroll,
                  tamañoImagen.Height + valorScroll);
      EphesiaUtilidades.CentrarControlEnPanel(pan, pic);
   }
   else if (e.NewValue < e.OldValue)
   {
      pic.Size = new Size(valorScroll - tamañoImagen.Width,
                  valorScroll - tamañoImagen.Height);
      EphesiaUtilidades.CentrarControlEnPanel(pan, pic);
   }
}

private void btnLimpiarPortapapeles_MouseDown(object sender,
            MouseEventArgs e)
{
   Clipboard.Clear();
}

private void btnLimpiarP_MouseDown(object sender,
            MouseEventArgs e)
{
   Clipboard.Clear();
}

private void btnBorrarPortapapelesAZTEC_MouseDown(object
            sender, MouseEventArgs e)
{
   Clipboard.Clear();
}

private void btnMuestraListaPortaQR_MouseDown(object sender,
            MouseEventArgs e)
{
   if (this.lstListaProcesos.Items.Count != 0)
   {
      this.lstListaProcesos.Size = new Size(458, 458);
      this.lstListaProcesos.Visible = true;
   }
}

private void lstListaProcesos_MouseLeave(object sender,
            EventArgs e)
{
   this.lstListaProcesos.Visible = false;
   this.lstListaProcesos.Size = new Size(455, 10);
}

private void btnAbrirListaDATAMATRIX_MouseDown(object sender,
            MouseEventArgs e)
{
   if (this.lstDATAMATRIX.Items.Count != 0)
   {
      this.lstDATAMATRIX.Visible = true;
```

```
        this.lstDATAMATRIX.Size = new Size(455, 455);
    }
}

private void btnAbrirListaAZTEC_MouseDown(object sender,
            MouseEventArgs e)
{
    if (this.lstListaAZTEC.Items.Count != 0)
    {
        this.lstListaAZTEC.Visible = true;
        this.lstListaAZTEC.Size = new Size(455, 455);
    }
}

private void lstDATAMATRIX_MouseLeave(object sender,
            EventArgs e)
{
    this.lstDATAMATRIX.Visible = false;
    this.lstDATAMATRIX.Size = new Size(455, 10);
}

private void lstListaAZTEC_MouseLeave(object sender,
            EventArgs e)
{
    this.lstListaAZTEC.Visible = false;
    this.lstListaAZTEC.Size = new Size(455, 10);
}

private void btnMandarPortapapelesQr_MouseDown(object sender,
            MouseEventArgs e)
{
    operacionesListas(this.lstListaProcesos);
}

private void btnMandarPortapapelesListaAZTEC_MouseDown(object
            sender, MouseEventArgs e)
{
    operacionesListas(this.lstListaAZTEC);
}

private void operacionesListas(ListBox listaProcesos)
{
    if (listaProcesos.Items.Count != 0)
    {
        Clipboard.Clear();
        string resultadoTexto = "";
        for (int i = 0; i <= (listaProcesos.Items.Count - 1); i++)
        {
            if (resultadoTexto != "")
            {
                resultadoTexto = resultadoTexto + "\r\n" +
                        listaProcesos.Items[i].ToString();
            }
            else if (resultadoTexto == "")
            {
                resultadoTexto =
                        listaProcesos.Items[i].ToString();
            }
        }
```

```
        Clipboard.SetText(resultadoTexto);
    }
}

private void btnPegarPortapapelesDTM_MouseDown(object sender,
            MouseEventArgs e)
{
    operacionesListas(this.lstDATAMATRIX);
}

private void btnBorrarLista_MouseDown(object sender,
            MouseEventArgs e)
{
    this.lstListaProcesos.Items.Clear();
}

private void btnBorrarListaDTM_MouseDown(object sender,
            MouseEventArgs e)
{
    this.lstDATAMATRIX.Items.Clear();
}

private void btnBorrarListaAZTEC_MouseDown(object sender,
            MouseEventArgs e)
{
    this.lstListaAZTEC.Items.Clear();
}

private void btnGenerarAZTEC_MouseDown(object sender,
            MouseEventArgs e)
{
    try
    {
        Cursor = Cursors.WaitCursor;
        this.txtCronoAZTEC.Text =
                    EphesiaUtilidades.cronometro(1);
        this.pctCodigoAZTEC.SizeMode =
                    PictureBoxSizeMode.AutoSize;
        int tamañoImagenAZTEC =
                    Convert.ToInt16(this.cboAZTEC.SelectedItem.ToString
                    ().Substring(10, 3));
        this.pctCodigoAZTEC.Image =
                    EphesiaUtilidades.generarAZTEC(this.txtAZTEC.Text.
                    Trim(), tamañoImagenAZTEC);
        this.pctMuestraAZTEC.Image =
                    this.pctCodigoAZTEC.Image;
        this.txtCronoAZTEC.Text =
                    EphesiaUtilidades.cronometro(0);
        tamañoImagen = this.pctCodigoAZTEC.Size;
        this.hScrollBarAZTEC.Value = tamañoImagen.Height;
        if (this.pctCodigoAZTEC.Height <=
                    this.pctMuestraAZTEC.Height)
        {
            this.pctMuestraAZTEC.SizeMode =
                    PictureBoxSizeMode.Normal;
            this.pctMuestraAZTEC.Image =
                    this.pctCodigoAZTEC.Image;
        }
        else
```

```
        {
            this.pctMuestraAZTEC.SizeMode =
                        PictureBoxSizeMode.StretchImage;
            this.pctMuestraAZTEC.Image =
                        this.pctCodigoAZTEC.Image;
        }

                        EphesiaUtilidades.CentrarControlEnPanel
                        (this.pnlAZTEC, this.pctCodigoAZTEC);
              this.lblDatosAZTEC.Text = "Código AZTEC (" +
                        this.pctCodigoAZTEC.Width.ToString()
             + " x " +
                        this.pctCodigoAZTEC.Height.ToString()
                        + " píxeles; "
                        + EphesiaUtilidades.cadenaPixelesMilimetros
                        (this.pctCodigoAZTEC.Width)+ " x "  +
                        EphesiaUtilidades.cadenaPixelesMilimetros
                        (this.pctCodigoAZTEC.Height) + " milímetros).";
        contAZTEC();
        Cursor = Cursors.Default;
    }
    catch (System.NullReferenceException)
    {
       EphesiaError.errorTamaño();
        this.pctMuestraEscala.Image = null;
        Cursor = Cursors.Default;
    }
    catch (IndexOutOfRangeException)
    {
        EphesiaError.errorFueraDeRangoCadena("Demasiadas
                        generaciones para la memoria del sistema.");
        indexOutOfRangeException = 1;
        Cursor = Cursors.Default;
    }
    catch (System.Exception err)
    {
        EphesiaError.errorDesconocido(err);
        Cursor = Cursors.Default;
    }
}

private void contAZTEC()
{
    this.txtNumeroCaracteresAZTEC.Text =
              this.txtAZTEC.Text.Trim().Length.ToString();
    this.lblNumero.Text = this.txtNumeroCaracteresAZTEC.Text;
}

private void btnContarAZTEC_MouseDown(object sender,
              MouseEventArgs e)
{
    contAZTEC();
}

private void btnGuardarAZTEC_MouseDown(object sender,
              MouseEventArgs e)
{
    Cursor = Cursors.WaitCursor;
    EphesiaUtilidades.guardarCodigo("AZTEC", saveFileDialog1,
```

```
            this.txtAZTEC.Text.Trim(),
        this.cboAZTEC.SelectedItem.ToString().
            Substring(10, 3),
        CadenasEphesia.usuarioApellido,
            this.pctCodigoAZTEC.Image);
    Cursor = Cursors.Default;
}

private void rbtUniversal_MouseEnter(object sender,
            EventArgs e)
{
    toolTip1.SetToolTip(this.rbtUniversal, "EAN8" + "\r\n" +
            "EAN13" + "\r\n" + "UPC-A" +
        "\r\n" + "QRCODE" + "\r\n" + "CODE39" + "\r\n" +
                    "CODE128" + "\r\n" + "ITF" +
        "\r\n" + "PDF417" + "\r\n" + "CODABAR" + "\r\n" +
                    "MSI" + "\r\n" + "PLESSEY" +
        "\r\n" + "DATAMATRIX" + "\r\n" + "AZTEC");
}

private void btnConectarCam_MouseDown(object sender,
            MouseEventArgs e)
{
    this.lblAnimacion.Visible = true;
    this.lblOcultaVideo.Visible = true;
    if (wCam == null)
    {
        wCam = new WebCam {Container = this.pctCam};
        wCam.OpenConnection();
        this.btnConectarCam.Enabled = false;
        this.btnApagarCamara.Enabled = true;
        this.btnCapturar.Enabled = true;
        this.txtDatosCam.Text = "";
    }
    timer1.Start();
}

private void btnApagarCamara_MouseDown(object sender,
            MouseEventArgs e)
{
    wCam.Dispose();
    wCam = null;
    this.btnConectarCam.Enabled = true;
    if (pctModi.Image == null)
    {
        this.btnDecodificarCam.Enabled = false;
        this.btnApagarCamara.Enabled = false;
        this.btnCapturar.Enabled = false;
        this.btnGuardarImagen.Enabled = false;
        this.btnImprimirImagen.Enabled = false;
        this.ScrollAlfa.Enabled = false;
        this.hScrollBarColor.Enabled = false;
    }
    else
    {
        this.btnApagarCamara.Enabled = false;
    }
    this.pctCam.Image = null;
}
```

```
private void btnDecodificarCam_MouseDown(object sender,
            MouseEventArgs e)
{

   if (this.pctCaptura.Image != null)
   {
      Bitmap bitmap = new Bitmap(this.pctModi.Image);
      this.txtDatosCam.Text =
                  EphesiaUtilidades.
                  decodificarUniversalCamara(bitmap);
   }
}

private void timer1_Tick(object sender, EventArgs e)
{
   contadorRefresco++;
   this.lblAnimacion.Text = contadorRefresco.ToString();
   if (contadorRefresco >= 35)
   {
      this.lblAnimacion.Visible = false;
      this.lblOcultaVideo.Visible = false;
      this.timer1.Stop();
      contadorRefresco = 0;
   }
}

private void btnCapturar_MouseDown(object sender,
            MouseEventArgs e)
{
   this.pctCaptura.Image = wCam.GetCurrentImage();
   btnApagarCamara_MouseDown(sender, e);
   btnConectarCam_MouseDown(sender, e);
   this.btnGuardarImagen.Enabled = true;
   this.btnImprimirImagen.Enabled = true;
   this.btnDecodificarCam.Enabled = true;
   this.ScrollAlfa.Enabled = true;
   this.hScrollBarColor.Enabled = true;
   this.pctModi.Image = this.pctCaptura.Image;
   this.pctModi.Visible = true;
}

private void ScrollAlfa_Scroll(object sender,
            ScrollEventArgs e)
{
   c40 = asignarMatrix1(ScrollAlfa);
   c41 = asignarMatrix1(ScrollAlfa);
   c42 = asignarMatrix1(ScrollAlfa);
   this.lbl45.Text = auxFormula.ToString();
   filtroNumero = 1;
   aplicarFiltros(this.pctCaptura, this.pctModi, 0, 0);
}

private void hScrollBarColor_Scroll(object sender,
            ScrollEventArgs e)
{
   c00 = asignarMatrix1(hScrollBarColor);
   c01 = asignarMatrix1(hScrollBarColor);
   c02 = asignarMatrix1(hScrollBarColor);
```

```
    c10 = asignarMatrix1(hScrollBarColor);
    c11 = asignarMatrix1(hScrollBarColor);
    c12 = asignarMatrix1(hScrollBarColor);
    c20 = asignarMatrix1(hScrollBarColor);
    c21 = asignarMatrix1(hScrollBarColor);
    c22 = asignarMatrix1(hScrollBarColor);
    this.label62.Text = auxFormula.ToString();
    filtroNumero = 1;
    aplicarFiltros(this.pctCaptura, this.pctModi, 0, 0);
}

private void aplicarFiltros(PictureBox pictureOrigen,
            PictureBox pictureDestino, int facCW, int facCH)
{
    Cursor = Cursors.WaitCursor;
    if (filtroNumero == 1)
    {
        image = new Bitmap(pictureOrigen.Image);
        imageAttributes = new ImageAttributes();
        float[][] colorMatrixElements = {
        new float[] {c00,  c01,  c02,  c03, c04},
                        //Escala roja factor of 2
        new float[] {c10,  c11,  c12,  c13, c14},
                        //Escala verde factor of 1
        new float[] {c20,  c21,  c22,  c23, c24},
                        //Escala azul factor of 1
        new float[] {c30,  c31,  c32,  c33, c34},
                        //Escala alfa factor of 1
        new float[] {c40,  c41,  c42,  c43, c44}};
                        //Transacción ficticia of 0.2
        colorMatrix = new ColorMatrix(colorMatrixElements);
        imageAttributes.SetColorMatrix(colorMatrix,
                        ColorMatrixFlag.Default, ColorAdjustType.Bitmap);
        Bs = (Bitmap)(image.Clone());
        Gs = Graphics.FromImage(Bs);
        pictureDestino.Image = Bs;
        Gs.DrawImage(image, new Rectangle(0, 0,
                        pictureOrigen.Width, pictureOrigen.Height),
        0, 0, pictureOrigen.Width, pictureOrigen.Height,
                GraphicsUnit.Pixel, imageAttributes);
    }
    Cursor = Cursors.Default;
}

private float asignarMatrix1(HScrollBar hs)
{
    try
    {
        if (hs.Value != 100)
        {
            auxFormula = (Convert.ToSingle((hs.Value - 90))
                                / 100);
        }
        else if (hs.Value == 100)
        {
            auxFormula = 0;
        }
        return auxFormula;
    }
```

```
            catch (System.ArgumentOutOfRangeException err)
            {
                EphesiaError.errorFueraDeRango(err);
                return 0;
            }
            catch (System.FormatException err)
            {
                EphesiaError.errorFormatoInspector(err,
                            "conceptos en matrices, código interno de Ephesia:
                                    asignarMatrix1.");
                return 0;
            }
        }

        private void btnGuardarImagen_MouseDown(object sender,
                    MouseEventArgs e)
        {
            EphesiaUtilidades.guardarImagenCodigo(saveFileDialog1,
                    this.pctModi.Image);
        }

    }
}
```

Código X

```
        private void txtEAN_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (e.KeyChar == (char)Keys.Return)
            {
                switch (this.toolStripLabel2EAN.Text)
                {
                    case "EAN-A":
                                EphesiaUtilidades.borrarFiltro();
                                controlEanNavegar(FiltrosEphesia.navegarATextoDevolucionNumero
                                    ((Convert.ToInt32(this.txtEAN.Text.Trim().Substring(2, 5))).ToString(),
                                this.cAPTURASBindingSource, "NUMERO"));
                                this.txtEAN.Text = "";  break;
                    case "EAN-D":
                                EphesiaUtilidades.borrarFiltro();
                                string formula = "{CAPTURAS.CAJANUMERO} ='" +
                                (Convert.ToInt32(this.txtEAN.Text.Trim().Substring(9, 3))).
                                    ToString() + "'";
                                generarInformes("/Informes/inforcajas.rpt", formula);
                                activarOpcionMenuBorrarFiltro();
                                this.txtEAN.Text = "";  break;
                    case "EAN-E"://Baja automática
                                EphesiaUtilidades.borrarFiltro();
                                controlEanNavegar(FiltrosEphesia.navegarATextoDevolucionNumero
                                    ((Convert.ToInt32(this.txtEAN.Text.Trim().Substring(2, 5))).ToString(),
                                    this.cAPTURASBindingSource, "NUMERO"));
                                toolStripButtonCausarBaja_Click(sender, e);
                                this.txtEAN.Text = "";  break;
                    }
                }
            }
```

Código XI

```
System.Diagnostics.Process p = new System.Diagnostics.Process();
p.StartInfo.FileName = @"MiAplicativo.exe";
p.Start();
p.WaitForExit(); //Abre calculadora
```

ó

```
System.Diagnostics.Process.Start("iexplore.exe",   "http://www.google.es");
//Abre explorador por defecto con la página GOOGLE
```

Abrir un programa externo:

```
ShellExecute(NULL, TEXT("abrir"), TEXT(CadenasEphesia.Ap
+ "\\ANIMALIA\\Ephesia.exe"), NULL, NULL, SW_SHOWNORMAL);
```

Abrir un programa externo y que comience maximizado:

```
ShellExecute(NULL, TEXT("abrir"), TEXT(CadenasEphesia.Ap
+ "\\ANIMALIA\\Ephesia.exe"), NULL, NULL, SW_MAXIMIZE);
```

Abrir un programa externo y que permanezca oculto:

```
ShellExecute(NULL, TEXT("abrir"), TEXT(CadenasEphesia.Ap
+ "\\ANIMALIA\\Ephesia.exe"), NULL, NULL, SW_HIDE);
```

Abrir una página web:

```
ShellExecute(NULL, TEXT("abrir"), TEXT("http://Minglanillos.com"), NULL, NULL,
SW_SHOWNORMAL);
```

Abrir un directorio:

```
ShellExecute(NULL, TEXT("abrir"), TEXT(CadenasEphesia.Ap
+ "\\MiDirectorio \\"), NULL, NULL, SW_SHOWNORMAL);
```

Abrir un directorio mostrando la jerarquía de carpetas:

```
ShellExecute(NULL, TEXT("exoplorar"), TEXT(CadenasEphesia.Ap
+ "\\MiDirectorio\\"), NULL, NULL, SW_SHOWNORMAL);
```

Imprimir un documento de texto:

```
ShellExecute(NULL, TEXT("imprimir"), TEXT(CadenasEphesia.Ap
+ "\\MIDirectorio\\colodrillosEnAlgara.txt"), NULL, NULL, SW_SHOWNORMAL);
```

Código XII

```
<?xml version="1.0" encoding="UTF-8"?>
 <Document>
  <name>utm10km_e.KMZ</name>
  <LookAt>
   <longitude>-2.128967987332191</longitude>
   <latitude>40.62161091804109</latitude>
   <altitude>0</altitude>
```

```
  <heading>0.2331934398599846</heading>
  <tilt>1.443345520274446e-013</tilt>
  <range>113498.3017892151</range>
  <altitudeMode>relativeToGround</altitudeMode>
</LookAt>
<Style id="s_ylw-pushpin_copy20">
  <IconStyle>
    <Icon>
    </Icon>
  </IconStyle>
  <LabelStyle>
    <scale>0.7</scale>
  </LabelStyle>
</Style>
<Style id="s_ylw-pushpin_hl_copy20">
  <IconStyle>
    <Icon>
    </Icon>
  </IconStyle>
  <LabelStyle>
    <scale>0.7</scale>
  </LabelStyle>
</Style>
<StyleMap id="m_ylw-pushpin_copy3">
  <Pair>
    <key>normal</key>
    <styleUrl>#s_ylw-pushpin_copy20</styleUrl>
  </Pair>
  <Pair>
    <key>highlight</key>
    <styleUrl>#s_ylw-pushpin_hl_copy20</styleUrl>
  </Pair>
</StyleMap>
<Folder>
  <name>CuadrÃ¬cula CUTM 10 km</name>
  <open>1</open>
  <Folder>
    <name>Etiquetas</name>
    <open>1</open>
    <Style>
    </Style>
    <Placemark>
      <name>31SBC47</name>
      <styleUrl>#m_ylw-pushpin_copy3</styleUrl>
      <Point>
        <coordinates>0.07127699999999999,38.5851,0</coordinates>
      </Point> //A continuación el mismo código para cada coordenada UTM
```

Anexo IV

Listas

Lista I

1-3: Phylum Acanthocephala
- 1- Clase Archiacanthocephala
- 2- Clase Eoacanthocephala
- 3- Clase Palaeacanthocephala

4-9: Phylum Annelida
- 4- Clase Acanthobdellea
- 5- Clase Aphanoneura
- 6- Clase Branchiobdellea
- 7- Clase Hirudinea
- 8- Clase Oligochaeta
- 9- Clase Polychaeta

10-19 y 99: Phylum Arthropoda
- 10- Clase Arachnida
- 11- Clase Branchiopoda
- 12- Clase Malacostraca
- 13- Clase Maxillopoda
- 14- Clase Ostracoda
- 15- Clase Entognatha
- 99- Clase Insecta
- 16- Clase Chylopoda
- 17- Clase Diplopoda
- 18- Clase Pauropoda
- 19- Clase Symphyla

20-21: Phylum Bryozoa
- 20- Clase Gymnolaemata
- 21- Clase Phylactolaemata

22-27: Phylum Chordata
- 22- Clase Cephalaspidomorphi
- 23- Clase Actinopterygii
- 24- Clase Amphibia
- 25- Clase Aves
- 26- Clase Mammalia
- 27- Clase Reptilia

28-29: Phylum Cnidaria
- 28- Clase Leptolida
- 29- Clase Scyphomedusa

30-32: Phylum Entoprocta
- 30- Clase Entoproctoda
- 31- Phylum Gastrotricha
- 32- Clase Chaetonotoidea

33-34: Phylum Mollusca
- 33- Clase Bivalvia
- 34- Clase Gastropoda

35-36: Phylum Nematoda
- 35- Clase Adenophorea
- 36- Clase Secernentea

37: Phylum Nematomorpha
- 37- Clase Gordioida

39-39: Phylum Nemertea
- 38- Clase Anopla
- 39- Clase Enopla

40-43: Phylum Platyhelminthes
- 40- Clase Cestoda
- 41- Clase Monogenea
- 42- Clase Trematoda
- 43- Clase Turbellaria

44: Phylum Rotifera
- 44- Clase Eurotatoria

45-47: Phylum Tardigrada
- 45- Clase Eutardigrada
- 46- Clase Heterotardigrada
- 47- Clase Mesotardigrada

48: Phylum Porifera
- 48- Clase Demospongiae

49-96: Usuario
- 49-96

97: Administración
- 97

98: Preparaciones
- 98

Lista I: numeración utilizada por nosotros de las clases de Animalia para el dígito codificador de entrada en los códigos de barras, en sustitución de los guarismos de procedencia por producto y países. Los números 97 y 98 se reservan para labores de administración y para las bases de datos de preparaciones microscópicas. La clase Insecta, debido a la gran cantidad de especies, se coloca la última para facilitar futuras ampliaciones hacia arriba restando números a la lista de cifras personalizables por el usuario.

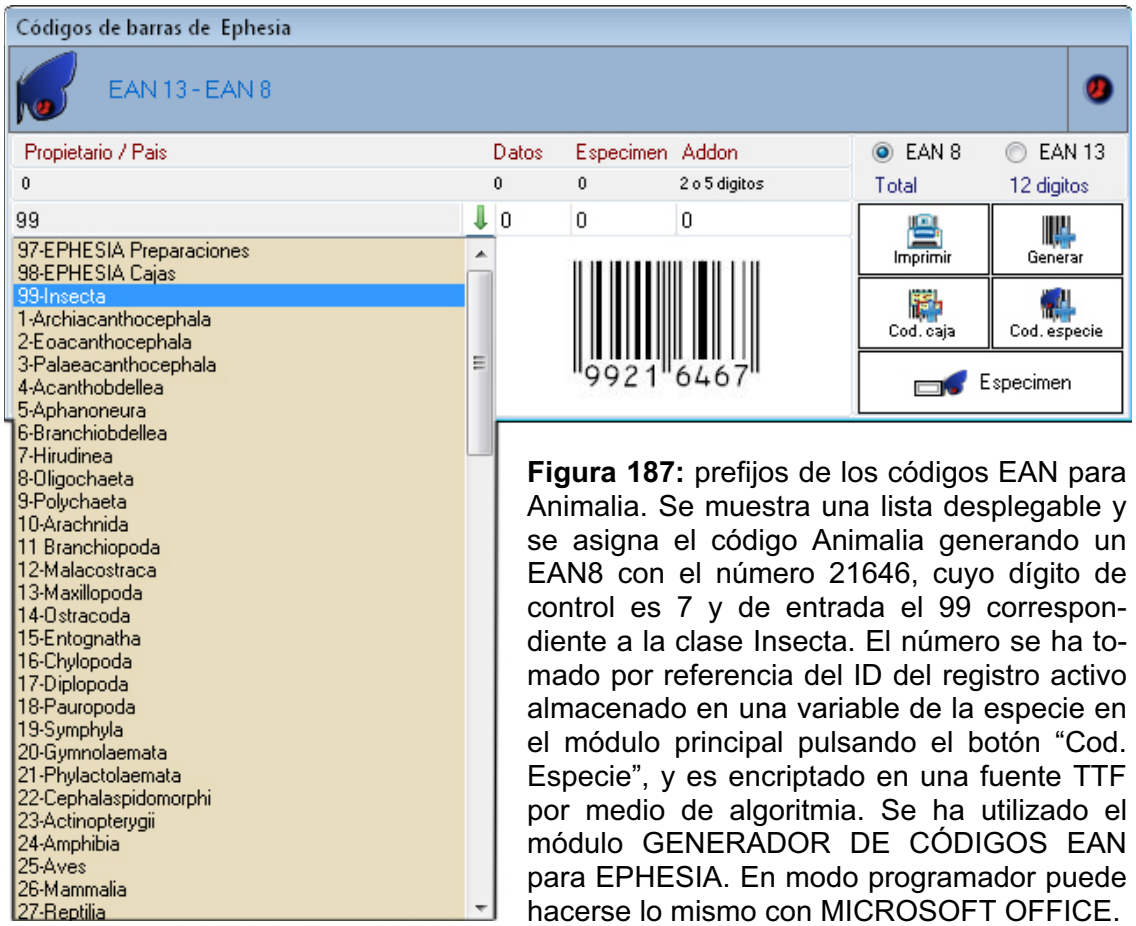


Figura 187: prefijos de los códigos EAN para Animalia. Se muestra una lista desplegable y se asigna el código Animalia generando un EAN8 con el número 21646, cuyo dígito de control es 7 y de entrada el 99 correspondiente a la clase Insecta. El número se ha tomado por referencia del ID del registro activo almacenado en una variable de la especie en el módulo principal pulsando el botón “Cod. Especie”, y es encriptado en una fuente TTF por medio de algoritmia. Se ha utilizado el módulo GENERADOR DE CÓDIGOS EAN para EPHESIA. En modo programador puede hacerse lo mismo con MICROSOFT OFFICE.

Lista II

40-Alemania
41-Alemania
42-Alemania
43-Alemania
44-Alemania
779-Argentina
485-Armenia
93-Australia
90-Austria
91-Austria
54-Bélgica y Luxemburgo
777-Bolivia
789-Brasil
780-Chile
690-China
691-China
692-China
529-Chipre
770-Colombia
880-Corea del Sur
744-Costa Rica
850-Cuba
57-Dinamarca
786-Ecuador
00-EE.UU. y Canadá
01-EE.UU. y Canadá
02-EE.UU. y Canadá
03-EE.UU. y Canadá
04-EE.UU. y Canadá
05-EE.UU. y Canadá
06-EE.UU. y Canadá
07-EE.UU. y Canadá
08-EE.UU. y Canadá
09-EE.UU. y Canadá
10-EE.UU. y Canadá
11-EE.UU. y Canadá
12-EE.UU. y Canadá
13-EE.UU. y Canadá
622-Egipto
741-El Salvador
20-En la tienda Funciones
21-En la tienda Funciones
22-En la tienda Funciones
23-En la tienda Funciones

24-En la tienda Funciones
25-En la tienda Funciones
26-En la tienda Funciones
27-En la tienda Funciones
28-En la tienda Funciones
29-En la tienda Funciones
858-Eslovaquia
84-España
474-Estonia
46-Federación de Rusia
480-Filipinas
64-Finlandia
30-Francia
31-Francia
32-Francia
33-Francia
34-Francia
35-Francia
36-Francia
37-Francia
486-Georgia
520-Grecia
740-Guatemala
742-Honduras
489-Hong Kong
599-Hungría
890-India
899-Indonesia
626-Irán
539-Irlanda
569-Islandia
729-Israel
80-Italia
81-Italia
82-Italia
83-Italia
45-Japón
49-Japón (JAN-13)
625-Jordania
487-Kazajstán
475-Letonia
528-Líbano
477-Lituania
531-Macedonia
955-Malasia
535-Malta
611-Marruecos
609-Mauricio
750-México
484-Moldavia
743-Nicaragua
978-Norma Internacional de Numeración libro (ISBN)
70-Noruega
94-Nueva Zelanda
979-Número Internacional Estándar Música (ISMN)
977-Número Internacional Normalizado de Publicaciones Seriadas de Publicaciones Periódicas (ISSN)
87-Países Bajos
784-Paraguay
775-Perú
785-Perú
590-Polonia
560-Portugal
980-recibos de reembolso
50-Reino Unido
859-República Checa
746-República Dominicana
594-Rumania
860-Servia
888-Singapur
479-Sri Lanka
600-Sudáfrica
601-Sudáfrica
73-Suecia
76-Suiza
885-Tailandia
471-Taiwán
619-Túnez
869-Turquía
482-Ucrania
773-Uruguay
759-Venezuela
893-Vietnam

Lista II: numeración correspondiente a los países y otras configuraciones para el dígito de codificación de entrada de los códigos de barras.

Anexo V

Referencias

Todas las direcciones con acceso el 25 de febrero del 2014.

Lectores

http://www.intermec.es/products/bar_code_scanners/index.aspx

http://www.plusexpress.es/7-lectores-codigo-de-barras?gclid=CM2pycD-t7wCFSTmwgodWHIARw

http://www.motorolasolutions.com/XL-ES/Product+Lines/Symbol/Symbol+Bar+Code+Scanners

http://www.cognex.com/we-can-read-it.aspx?id=48&locale=es&cm_campid=6B8CD92A-3588-E311-A002-005056B20035&langtype=1034&gclid=CIz7hqCAuLwCFYYBwwodRh4AIQ

Fuentes de códigos de barras

http://www.codigoean.com/codigo-de-barras-imagen-codigos-propios.html

http://www.comercialtpv.com/accesorios-para-tpv/scanner-lector-codigo-barras.html?gclid=CN3f87_5t7wCFQ_HtAodykwASA

http://www.tortugaproductiva.galeon.com/docs/ean128/files/ean128ttf.zip

http://crazyhouse.e-mision.net/blogs/dotnet/archive/2002/06/21/fuente-c-243-digo-de-barras-ean-13.aspx

http://www.letramania.com/Fuentes-De-Codigos-De-Barras/index3.htm

http://dl.dropbox.com/u/26490398/c/ean13.zip

http://www.fontpalace.com/font-details/EAN-13/

http://www.fpress.com/revista/Num9905/fuentes.zip

Aplicativos de adquisición

http://zbar.sourceforge.net/

http://zbar.sourceforge.net/iphone/index.html

http://sourceforge.net/projects/zbar/files/iPhoneSDK/ZBarSDK-1.2.dmg/download

http://www.bcwebcam.de/en/

http://www.qualitysoft.de/index.php?id=15

https://play.google.com/store/apps/details?id=la.droid.QRCODE&hl=es

https://play.google.com/store/apps/details?id=appinventor.ai_progetto2003.SCAN&hl=es

http://www.codigoean.com/codigo-de-barras-grafica-movil.html

http://www.labeljoy.com/es/QRCODE-code-es/generador-codigo-QRCODE/

http://www.labeljoy.com/es/soporte/generar-codigo-barras/datamatrix-es/

http://www.labeljoy.com/es/QRCODE-code-es/

http://www.labeljoy.com/es/soporte/generar-codigo-barras/ean-13-es/

http://www.labeljoy.com/es/soporte/generar-codigo-barras/ean-8-es/

http://www.aulux.com/

http://www.tec-it.com/es/software/barcode-software/barcode-creator/barcode-studio/Default.aspx

http://www.tec-it.com/es/software/barcode-software/office/word-excel/Default.aspx

http://www.tec-it.com/es/start/Default.aspx

http://www.nchsoftware.com/barcode/

http://www.autoreseditores.com/es-es/generar-codigo-barras-isbn.html

http://www.itsapparent.com/barcodeproducer/

http://www.windowsphone.com/es-es/store/app/scan-QRCODE-code-and-barcode-reader/c62d7a1c-c336-4394-84d0-2ea4913fc891

Aplicativos decodificadores

http://www.beetagg.com/en/download-QRCODE-reader/

http://www.upc.fi/en/upcode/download/

http://www.i-nigma.com/Downloadi-nigmaReader.html

http://www.quickmark.com.tw/En/basic/downloadMain.asp

http://reader.kaywa.com/getit

https://play.google.com/store/apps/details?id=com.google.zxing.client.android&hl=es

https://play.google.com/store/apps/details?id=me.scan.android.client&hl=es

https://play.google.com/store/apps/details?id=com.bidi&hl=es

https://play.google.com/store/apps/details?id=com.dodo.scannersecure&hl=es

https://play.google.com/store/apps/details?id=com.manateeworks.barcodescanners&hl=es

http://www.jaxo-systems.com/solutions/barcapture/

http://sourceforge.net/projects/zbar/files/zbar/0.10/zbar-0.10-setup.exe/download

https://itunes.apple.com/cl/app/scan/id411206394?l=es&mt=8

http://store.ovi.com/content/2322

http://app.scanlife.com/appdownload/dl

http://qrdroid.com/get/

http://keremerkan.net/downloads/

http://redlaser.com/

https://play.google.com/store/apps/details?id=com.tingiz

https://scan.me/download

https://play.google.com/store/apps/details?id=com.alberovalley.qr1

http://quickmark.com.tw/En/basic/downloadMain.asp

Librerías

https://code.google.com/p/zxing/

https://code.google.com/p/zxing/downloads/list?q=label:Featured

http://community.bartdesmet.net/blogs/bart/archive/2006/09/18/4432.aspx

http://www.codeproject.com/cs/miscctrl/barcodectl.asp

http://www.codeproject.com/cs/webservices/wsbarcode.asp

http://www.planet-source-code.com/vb/scripts/BrowseCategoryOrSearchResults.asp?lngWId=1&txtCriteria=barcode&blnWorldDropDownUsed=TRUE&txtMaxNumberOfEntriesPerPage=10&blnResetAllVariables=TRUE&optSort=Alphabetical

http://cid-9f0a0ef4955e0126.skydrive.live.com/browse.aspx/Public/Codigo%20de%20Barras/Codigo%20Vb.NET?uc=1

http://cid-9f0a0ef4955e0126.skydrive.live.com/browse.aspx/Public/Codigo%20de%20Barras/Codigo%20C%7C3?uc=1

http://sourceforge.net/projects/javabarcoderead/

http://java.dzone.com/articles/java-barcode-api?utm_source=feedburner&utm_medium=feed&utm_campaign=Feed%3A+javalobby%2Ffrontpage+(Javalobby+%2F+Java+Zone)

http://barbecue.sourceforge.net/apidocs/net/sourceforge/barbecue/BarcodeImageHandler.html

http://zbar.sourceforge.net/

http://sourceforge.net/projects/zbar/files/iPhoneSDK/

http://www.ajpdsoft.com/modules.php?name=Downloads&d_op=viewdownloaddetails&lid=298

http://jocr.sourceforge.net/

http://asprise.com/product/ocr/selector.php

http://www.meeddy.com/java-barcodereader/

http://www.meeddy.com/net-barcodereader/csharp-netreader.shtml

http://www.meeddy.com/net-barcode/csharp-net.shtml

http://www.meeddy.com/java-barcode/

Fotografía, aplicativos, control remoto

http://www.europe-nikon.com/support/

http://imaging.nikon.com/lineup/software/control_pro/

http://imaging.nikon.com/lineup/software/control_pro/spec.htm

http://imaging.nikon.com/lineup/software/

http://imaging.nikon.com/lineup/software/capturenx2/

http://www.nikon.es/es_ES/product/software/camera-control-pro-2

http://digicamcontrol.com/

http://digicamcontrol.com/manual

http://digicamcontrol.com/sites/default/files/user_manual.pdf

http://digicamcontrol.com/news/release-candidate-1-version-110

http://sourceforge.net/projects/digicamcontrol/files/1.1.0%20-%20RC1/digiCamControlsetup_1.0.692.exe/download

http://www.breezesys.com/DSLRRemotePro/

http://www.breezesys.com/MultiCamera/index.htm

http://learn.usa.canon.com/app/pdfs/quickguides/CDLC_EOSUtility_RemoteOp_QuickGuide.pdf

http://www.diyphotobits.com/download-diyphotobitscom-camera-control/

http://nikonasia-en.custhelp.com/app/answers/detail/a_id/6459/~/nikon-transfer-1.5.1-software-download-(windows)

http://www.usa.canon.com/cusa/page.action

http://www.usa.canon.com/cusa/support/consumer/eos_slr_camera_systems/eos_digital_slr_cameras/digital_rebel_xti#DriversAndSoftware

http://sabsik.com/Cam2Com/

http://sabsik.com/Cam2Com/inwp_cam2com_screenshots.htm

http://canoncanada.custhelp.com/app/answers/detail/a_id/18708/related/1/session/L2F2LzEvdGltZS8xMzkyNDA0MTc5L3NpZC9zZ05SRFdNbA%3D%3D

http://canoncanada.custhelp.com/app/answers/detail/a_id/7886/~/remote-live-view-shooting-using-eos-utility

http://canoncanada.custhelp.com/app/answers/detail/a_id/13056/related/1

http://www.linuxhispano.net/2010/07/16/control-remoto-para-nuestra-camara-de-fotos/

Índice de la monografía

1

1D · 2, 4, 6, 9, 11, 23, 24, 42, 53, 54, 60, 75, 83, 84, 93

2

2D · 2, 4, 5, 6, 9, 11, 13, 14, 16, 17, 20, 21, 22, 23, 24, 42, 47, 52, 53, 54, 60, 61, 72, 73, 75, 77, 91, 93

3

3DI · 4, 13

A

Aberración sinusoidal · 74
Aberración, de imagen · 74
aberraciones · 45, 47, 92
ACTIVEX · 38, 74, 75
AERO · 11, 74
algoritmos · 8, 9, 11, 24, 25, 29, 38, 74, 91
ancho de banda · 59, 75
Anidamiento hexagonal · 74
ANIMALIA · 11, 91, 138
aplicativo · 10, 22, 25, 26, 40, 41, 44, 47, 52, 54, 61, 74, 75, 76, 77, 78
App · 9, 25, 74
ARAT · 9, 74
ARRAYTAG · 4, 14
ASCII · 14, 16, 35, 74
ASCIIGS1 · 35
ATMELRFID · 9, 79
AZTEC · 3, 4, 5, 6, 11, 13, 14, 24, 34, 36, 39, 41, 47, 52, 60, 61, 80, 91, 98, 122, 123, 124, 125, 127, 133, 134

B

BackColor · 29, 110, 118
Barcode · 21, 72
Base de datos documental · 74
BAT · 22, 74
BEETAGG · 4, 13, 14
BIDI · 4, 13, 14

Binarización · 61, 72, 75
bitmap · 25, 135
Bloc de notas · 46
BLOTCODE · 4, 15
BOKODE · 9
BWR · 59, 75

C

C# · 8, 78
C40 · 35, 80, 81
Calcas · 59
cámara Web · 11, 26, 45
CCD · 11, 12, 42, 44, 45
CHARGE COUPLED DEVICE · 42
Check-Sum · 25
CODABAR · 4, 12, 15, 52, 62, 134
CODACBLOCK · 4, 15
CODE11 · 4, 15
CODE128 · 4, 12, 16, 134
CODE16K · 4, 12, 15
CODE39 · 4, 12, 15, 52, 61, 134
CODE49 · 4, 12, 15
CODE93 · 4, 16
CODEONE · 4, 16
código de barras · 22, 27, 32, 33, 54, 75, 78, 92
COLORCODE · 4, 16
Convolución · 75
COOLDATAMATRIX · 13
coordenada UTM · 54, 61, 139
CPCODE · 4, 16
CRONÓMETRO DE SISTEMA · 11

D

DATAGLYPHS · 4, 17
DATAMATRIX · 4, 5, 6, 11, 13, 14, 16, 17, 24, 35, 36, 41, 47, 52, 60, 61, 80, 81, 82, 83, 91, 99, 108, 122, 123, 124, 125, 134
DATASTRIP · 4, 17
DECODIFICADOR POR CÁMARA WEB · 11, 52, 61
DECODIFICADOR UNIVERSAL · 11, 52, 61
dígito de control · 5, 28, 33, 140
DIP · 44
DLL · 38
DOTCODE · 4, 18
DVD · 45

E

EAN · 5, 12, 24, 28, 29, 32, 33, 34, 36, 38, 40, 41, 74, 137, 140, 143
EAN128 · 4, 5, 18, 23
EAN13 · 4, 5, 6, 12, 18, 23, 24, 33, 40, 46, 47, 52, 61, 83, 134
EAN2 · 18
EAN5 · 18
EAN8 · 4, 5, 6, 12, 18, 20, 22, 23, 24, 28, 29, 33, 40, 46, 47, 52, 61, 83, 84, 95, 134
EANIC · 12
EANUCC · 12
EDITOR DE MICROETIQUETAS · 11
ENTER · 26, 76, 93
EPHESIA · 11, 53, 54, 61, 101, 124, 125, 140
escáner · 9, 11, 12, 21, 26, 84
Esfericidad cóncava · 76
Esfericidad convexa · 76
ESPECTROFOTOMETRÍA · 91
Etiqueta de abeja · 14, 76
etiquetas entomológicas · 54, 61
Eventos en cascada · 76
EXCEL · 11, 28, 52, 74
Exe · 76

F

ForeColor · 29
frames · 45

G

GENERADOR DE CÓDIGOS EAN · 11, 140
GESTOR DE BIBLIOGRAFÍA · 11
GESTOR DE CAJAS ENTOMOLÓGICAS · 11
GS1128 · 12, 18, 20

H

HCCB · 4, 13, 19
HHLC · 42, 77
HITANG · 9, 79
HTKRFID · 9, 79

I

ICODE · 9
ID · 20, 28, 34, 39, 53, 54, 60, 61, 72, 77, 79, 96, 97, 140
imageFromFile · 47
impresora fotográfica · 59
Interfaz · 77
INTERLEAVED · 4, 19
INTRO · 10, 27, 52, 77
ISBN · 4, 19, 60, 61, 142
ISSN · 4, 19, 60, 142
ITF · 4, 12, 19, 52, 61, 134
ITF14 · 4, 20

K

KEYBOARD WEDGE · 42

L

LABORATORIO DE IMÁGENES · 11
lanzamiento dinámico · 6, 27, 28, 32, 34, 52, 53, 54, 60, 61
lector láser · 41, 42, 46
LUNAS ALGORÍTMICAS · 11

M

marcas de agua · 5, 6, 35, 39
matriz de la imagen · 26
MAXICODE · 4, 20
MCODE · 4, 13, 20
micro-etiqueta entomológica · 26, 32
micro-etiquetas · 2, 9, 18, 23, 25, 29, 31, 34, 42, 59, 61, 76, 93
MICROETIQUETAS EAN · 11
MICROQR · 4, 13, 14, 17, 20
MICROSOFT WORD · 53
MICROSOFT-TAG · 13
MIFARE · 9, 79
MINICODE · 5, 21

O

OCR-B · 29
ON RELESE · 11, 78
OT · 25

P

papel termo-fusible · 59
PDF · 21, 25, 40
PDF417 · 5, 13, 21, 52, 62, 134
POSNETCODE · 5, 21
PRAT · 9, 78
PREPARACIONES MICROSCÓPICAS · 54
procesos en cascada · 53

Q

QRCODE · 5, 6, 13, 20, 21, 22, 24, 34, 37, 38, 39, 41, 47, 52, 60, 61, 85, 86, 87, 88, 89, 90, 91, 101, 105, 121, 122, 123, 124, 129, 134
QUICKMARK · 5, 13, 22

R

RAM · 11, 92
RAW · 45
recursividad · 26, 27
Reed Solomon · 23, 78
RETURN · 26, 53, 76, 79, 93
RFID · 9, 71, 74, 78, 79
RS-232 · 42

S

SHOTCODE · 5, 13, 22
singulación · 9
SISTEMA EPHESIA · 11
SLR · 10, 45
SNOWFLAKECODE · 5, 22
SPOTCODE · 13
Supresión de datos · 5, 6, 32, 34, 36, 38
SWITCHES · 44
SyntaxHighlighting · 8, 79

T

TAB · 10, 26, 79
teléfonos móviles · 6, 9, 11, 13, 22, 45, 74
transpondedor · 9, 74, 78
Trasmutación · 5, 25
TRILLCODE · 5, 13, 22

TRIPCODE · 13
TSR · 42
TT · 25
TTF · 25, 29, 140

U

UCODE · 9
ULTRACODE · 5, 23
UPC · 12, 24, 29, 32, 134
UPCA · 5, 19, 23, 24, 52, 61
UPCE · 5, 23, 24
URL · 22, 79
USB · 42, 45

W

WAND EMULATION · 42

X

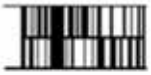
X12 · 35, 81
XML · 8








D5021

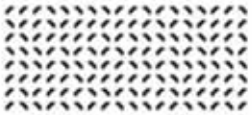

1234567890


34565678


1234567890

12345678

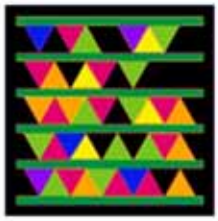





4123 4563

3 123456 789019


(00)123456789012345675


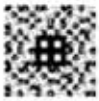
12345678

ISBN 958-12-3456-X
9 789581 234561

1144875X9031993307665201
1144-875X(199303)52;1-0


12345678901231







123456ABCdef

123456 78901 2

Sociedad Entomológica Aragonesa
www.sea-entomologia.org
Zaragoza, 30-06-2014