ACOPIOS

Revista Ibérica de Mineralogía

ISSN 2171-7788

V52014

MTIEDIT

ACOPIOS

An Iberian Mineralogist Journal

Revista Ibérica de Mineralogía

MTIEDIT ISSN 2171-7788

ACOPIOS
An Iberian Mineralogist Journal
Revista Ibérica de Mineralogía

Volumen 5, 2014 ISSN 2171-7788

EDITA
MTIEDIT, Vitoria-Gasteiz, 2014

Versión impresa de su original *on line*
revistas.ojs.es/index.php/acopios/issue/view/136
DOI: 10.7597/acopios2171-7788.2014

Sumario

ACOPIOS2014 V5: 1-13
Revista Ibérica de Mineralogía
MTIEDIT
DOI: 10.7597/acopios2171-7788.2014.1

Caracterización por espectroscopía Raman de los sulfatos básicos de cobre del filón "Terreras" (Villanueva del Duque, Córdoba, España)

César MENOR SALVÁN[(1)], Antonio CARMONA RUIZ[(2)], Inmaculada RAMOS MÁRQUEZ[(3)]

[(1)] Geospectra Scientific Solutions.
cmenor@geospectra.es

[(2)] AMYP
a.carruiz@gmail.com

[(3)] AMYP
inma.ramos@gmail.com

Resumen

C. MENOR SALVÁN, A. CARMONA RUIZ, I. RAMOS MÁRQUEZ (2014) Caracterización por espectroscopía Raman de los sulfatos básicos de cobre del filón "Terreras" (Villanueva del Duque, Córdoba, España). *Acopios,* **5**: 1-13.

En este trabajo se describen algunos sulfatos básicos de cobre, determinados mediante espectrometría Raman, que constituyen los minerales secundarios más significativos del filón "Terreras" (Villanueva del Duque, Córdoba, España). El filón "Terreras", es una mineralización hidrotermal tipo Pb-Cu-Zn (Ag), formado por calcopirita, galena y esfalerita principalmente, junto con cuarzo como constituyente principal del filón y situado en la aureola de metamorfismo asociada al batolito de Los Pedroches. La paragénesis secundaria incluye calcosina, covellita, óxidos de cobre, carbonatos (malaquita, cerusita, auricalcita y zincrosasita) y sulfatos (anglesita, linarita, brochantita y devillina).

Palabras clave: Espectroscopía Raman; Minerales de Cobre; Villanueva del Duque; Linarita; Brochantita; Devillina.

Abstract

C. MENOR SALVÁN, A. CARMONA RUIZ, I. RAMOS MÁRQUEZ (2014) Raman spectroscopy identification of supergenic basic copper sulfate minerals associated to the "Terreras" vein (Villanueva del Duque, Cordoba, Spain). *Acopios,* **5**: 1-13.

The "Filón Terreras" is a peribatolitic hydrothermal vein-type Pb-Cu-Zn (Ag) mineralization, constituted by a quartz hosted sulfide assemblage, with chalcopyrite and galena as main minerals, and associated to the "Los Pedroches" intrusion. The secondary paragenesis includes secondary sulfides (chalcocite and covellite), copper oxides, carbonates (malachite, cerussite, aurichalcite and zincrosasite) and sulfates. This work describes the Raman spectroscopic determination of the basic copper sulfates that constitutes the main secondary copper minerals.

Keywords: Raman spectroscopy, Copper minerals, Villanueva del Duque, Linarite, Brochantite, Devilline.

INTRODUCCIÓN

Se tiene noticia documentada de la presencia romana en este y otros yacimientos de plomo de la zona. Ya en época moderna (desde finales del siglo XIX), el filón "Terreras" fue explotado mediante varias concesiones mineras, siendo la mina de igual nombre la más importante de todas ellas. La mina "Terreras" se sitúa dentro del término municipal de Villanueva del Duque (Córdoba), aunque muy cerca también de la localidad de Alcaracejos, cuyo núcleo urbano se encuentra a unos 4 km de distancia por la carretera N-502. La actividad romana en la mina "Terreras" queda demostrada por la presencia de hallazgos tales como tenazas de hierro, clavos, cerámica, tuberías de plomo e incluso el de un lingote o galápago de este último material, donado a la Escuela de Ingenieros de Minas de Madrid en 1913. García Romero (2002) data este lingote como perteneciente al siglo I D.C.

Villanueva del Duque dista 76 km de Córdoba capital y tiene una superficie de 136,7 km^2, es decir, un término no muy grande, pero que tuvo a gala contar con las más importantes explotaciones de plomo de la provincia, hasta que el cierre en los años treinta de sus minas más destacadas inició la decadencia del municipio y el éxodo de una gran parte de la población que vivía de las minas. Hoy día sigue aún mirando a sus abandonadas minas (Figura 1), pero desde el punto de vista turístico y patrimonial, ya que sin duda estos aspectos siguen atrayendo visitantes a la zona.

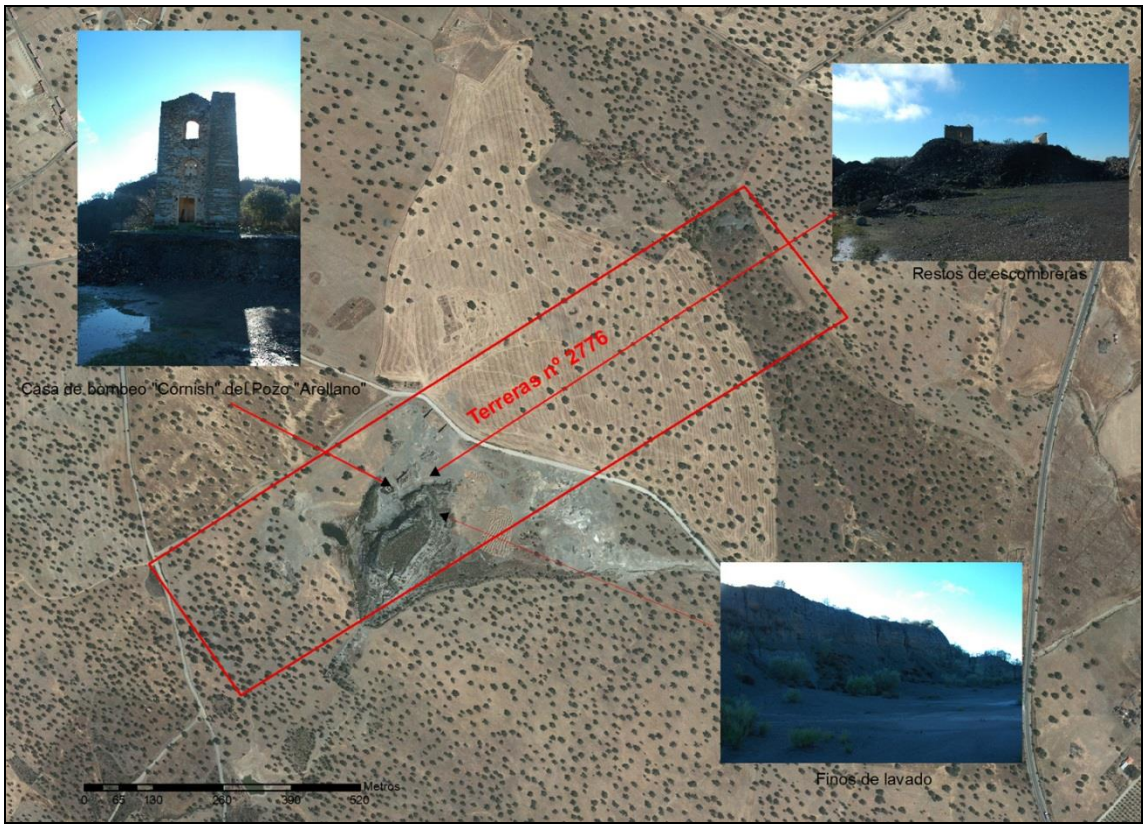

Figura 1: *Situación de la antigua concesión "Terreras", nº 2776, sobre Ortofotografía Rigurosa Color de Andalucía 2010 2011. Incluye montaje con la ubicación de algunas fotos de instalaciones y escombreras.*

A unos 2 km de la mina "Terreras" se encuentra la antigua estación de ferrocarril de El Soldado, contigua a las minas del mismo nombre. En su época de esplendor "Terreras" fue de las minas más importantes del Distrito y la segunda en producción de plomo, tras "El Soldado", lo que contribuyó sin duda a que la provincia de Córdoba llegase a ser, a comienzos del siglo XX, la primera productora nacional de Plomo.

MARCO GEOLÓGICO

El yacimiento se ubica en la parte centro-sur de la Hoja nº 858-El Viso del MTN, emplazado en la Zona Centro Ibérica, y más en concreto en la banda meridional de la Cuenca Carbonífera de Los Pedroches -donde aflora extensamente la facies Culm- en la que intruyó el Batolito del mismo nombre, provocando una aureola de metamorfismo de contacto, de espesor kilométrico en algunos casos, y un variado cortejo de diques ígneos. Cabe destacar que en el área estudiada la extensión de la aureola de metamorfismo es mayor por la presencia muy próxima de las apófisis granodioríticas de El Soldado y Las Morras. Las mineralizaciones se asocian a filones de cuarzo hidrotermales peribatolíticos de texturas principalmente brechoides, encajados en la aureola de metamorfismo y orientados en dirección NE-SO, que presentan espesores decamétricos y longitudes kilométricas. La paragénesis para las minas de la zona consiste en galena (argentífera) mayoritaria, esfalerita, calcopirita, pirita y como ganga cuarzo y carbonatos (calcita, dolomita, siderita) (Figura 2).

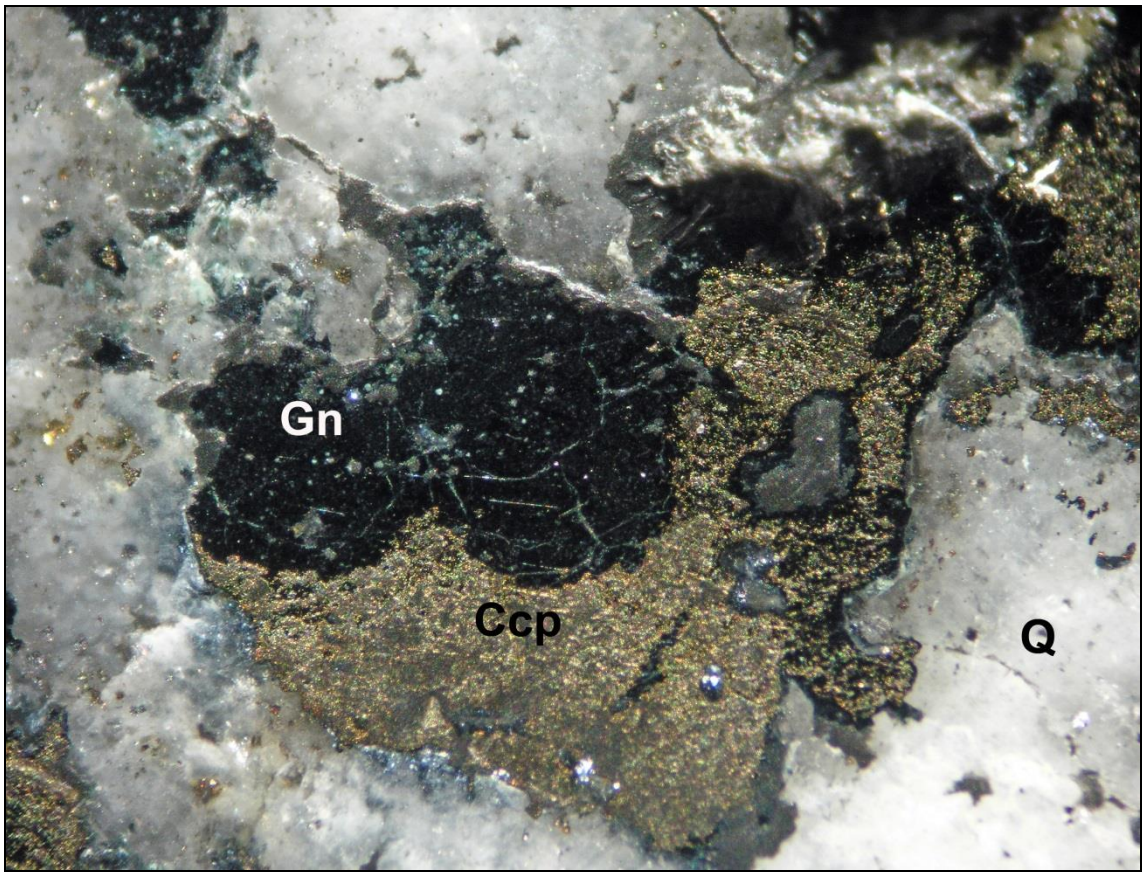

Figura 2: *Aspecto de visu de la mineralización, formada fundamentalmente por calcopirita y galena, del filón "Terreras". Se aprecia la formación de fases secundarias, de color verde. Campo de visión 1 cm. Gn: galena. Ccp: calcopirita: Q: cuarzo.*

El filón "Terreras" se extiende en una longitud de 3 km, encaja en pizarras y grauvacas del Carbonífero Inferior (Viseense) y en la mina "Terreras" presenta dirección N50ºE, aunque se curva hacia el N. al Este de la falla del Pozo "Arellano". Según Antonio Carbonell (1929) se trataba sin duda de un filón "de libro". El buzamiento medio indicado por el mismo autor era de 35ºSE. y su potencia, muy regular, entre 90 centímetros y 1 metro.

MINERÍA

La concesión “Terreras”, nº 2.776, con 42 hectáreas de superficie, se otorgó el 27 de julio de 1891 para explotar plomo, en un primer momento a nombre de una sociedad vasca de escasa entidad, que la arrendó poco después a la Sociedad Anónima “La Argentífera de Córdoba”, constituida en Bilbao en 1896, con capital español, bilbaíno en su mayor parte. En un breve período de tiempo esta última sociedad pasó a ser la siguiente Titular del derecho. Según la Estadística Minera de 1896, en ese año ya estaba investigando el filón Terreras y otros de la zona la sociedad referida anteriormente.

Las concesiones colindantes, sobre las que se prolongaba el filón “Terreras” y que constituyeron junto a ella un grupo minero, fueron explotadas también por la misma empresa, destacando entre otras las denominadas “Los Ingleses” y “Guido”.

Para la explotación del filón en profundidad la mina dispuso de varios pozos, dos de ellos maestros: “Terreras” y “Arellano”. Este último fue el principal, sobre el que se montó una máquina de vapor Cornwall, que llegó a desaguar 600 m^3 diarios, y a la que daba cobijo una Casa tipo “Cornish” que aún hoy día muestra su majestuosa silueta fácilmente identificable, aunque en ruinas (Figura 3).

Figura 3: *Vista de una zona de la mina "Terreras". A la derecha, la casa de bombeo tipo "Cornish" del Pozo "Arellano", uno de los pozos maestros. A la izquierda, las ruinas de lo que fue el lavadero. Al fondo, detrás de la casa "Cornish", se aprecia la altura que alcanzan los finos de lavado del mineral. A la izquierda también, pero en primer término, pueden verse algunos escombros, entre los que es posible recoger muestras de mineral.*

La mina llegó hasta los 550 metros de profundidad, con 15 plantas explotadas. Se dispone de datos de producción desde 1897 hasta 1922, aunque los trabajos se iniciaron ya en 1892. Con posterioridad a 1922 únicamente se llevaron a cabo labores de lavado de antiguas escombreras, en las que aún quedaba mucha esfalerita debido a que en los primeros compases de extracción era desechada, engrosando así sus escombreras.

Hacia los años 30 del pasado siglo XX la bibliografía consultada recoge que se efectuaba el lavado a pequeña escala de las antiguas escombreras, por lo que podría afirmarse que no hubo extracción propiamente dicha a partir de 1922.

Los datos de producción señalan que entre 1897 y 1922 se produjeron algo más de 66.000 t de Pb y, entre 1912-1922, poco más de 3000 t de esfalerita. Como media se daba un contenido en plata de 900 g/t, aunque hubo años que esa cifra superó los 2 kg/t.

La mina "Terreras" fue caducada en 1940, aunque décadas más tarde, concretamente en 1962, se instaló un lavadero denominado "Virgen de la Caridad", para la obtención de plomo y plata a partir de sus escombreras, que estuvo en funcionamiento hasta la década de los 70. En el mismo lavadero se trató material procedente de la conocida como mina de "Las Monjas", antigua concesión "Dificultades", situada en término de El Viso (Córdoba). Sobre esta última mina no consta que su paragénesis sea siquiera similar a la del filón "Terreras", no incluyendo a la calcopirita, que sin embargo sí aparece y está citada por todos los estudios que se han realizado para "Terreras". Decimos esto último para aclarar que por la tipología de las muestras estudiadas, en la que es plenamente identificable la paragénesis descrita para "Terreras", se descarta que pudieran proceder de la mina "Dificultades".

Actualmente se conservan algunos restos de escombreras, de las cuales se tomaron las muestras, las ruinas de la Casa "Cornish" del Pozo "Arellano", un gran volumen de finos procedentes del lavado del mineral y algunas ruinas de edificios que en su día formaron la estructura de la explotación, que incluía, en su momento, las oficinas administrativa y facultativa, el almacén de minerales, talleres de herrería y carpintería, corrales para depósito de maquinaria, carbones y maderas, vestuario para los obreros, polvorín, etc.

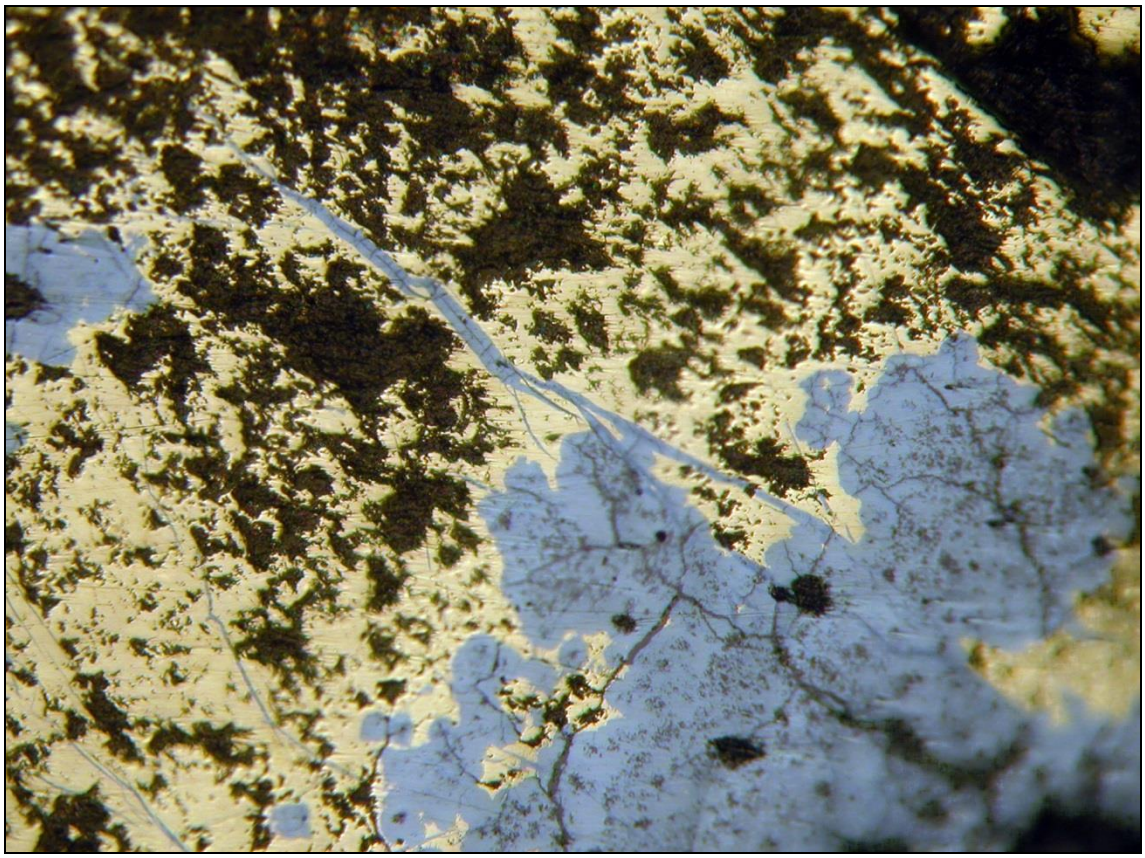

Figura 4: *Textura del mineral primario del Filón "Terreras". Campo de visión 5 mm. Gn: galena. Ccp: Calcopirita.*

SULFATOS SECUNDARIOS DEL FILÓN TERRERAS

En las muestras recogidas para este estudio, el sulfuro primario contenido en las muestras del filón "Terreras" está formado fundamentalmente por calcopirita y galena, con cantidades menores de esfalerita y pirita. La calcopirita se encuentra muy brechificada y la galena se presenta rellenando fracturas y rodeando los granos de calcopirita (Figura 4). La alteración de ésta es importante, siendo posible observar que el cobre movilizado tiñe de verde los cantos de carbonatos que forman la brecha del filón, así como la formación de pátinas compuestas de malaquita y de sulfatos básicos (Figura 2). En el filón puede observarse una variada paragénesis de minerales secundarios, que incluyen sulfuros y oxidos de cobre (calcosina, covellita y tenorita), así como carbonatos, que junto con los sulfatos básicos, constituyen los principales minerales de la fase supergénica. Los carbonatos identificados incluyen, por orden de abundancia, aragonito, cerusita, malaquita, auricalcita y zincrosasita. El aragonito, muy extendido, puede ser fácilmente confundido con otros minerales al estar en muchas ocasiones teñido de verde por la presencia de Cu(II). La movilización del cobre y plomo lleva a la formación de abundantes sulfatos básicos, que se acumulan especialmente en huecos y fracturas y que ocasionalmente forman cristales idiomorfos. Los sulfatos secundarios de cobre son los minerales de alteración más importantes encontrados en el filón, tanto por su abundancia como por su diversidad. La linarita, brochantita y devillina constituyen las especies más significativas, en especial las dos primeras, muy extendidas en las muestras, asociadas a los sulfuros primarios.

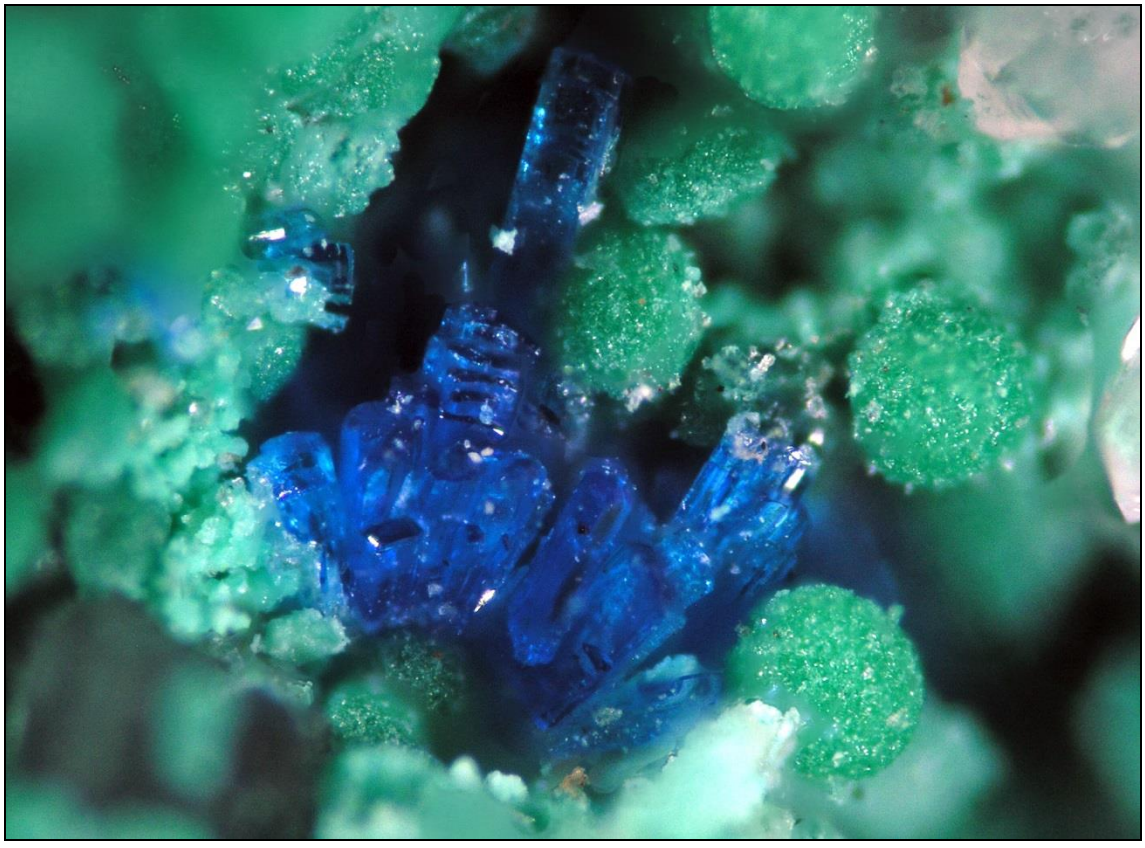

Figura 5: *Esférulas formadas por grupos radiales de cristales de brochantita y grupo de cristales de linarita. Estas agrupaciones de cristales se forman en huecos y fracturas del material del filón, en proximidad con los sulfuros primarios. Filón "Terreras". Campo de visión 1.2 mm.*

Brochantita $Cu_4SO_4(OH)_6$

Los sulfatos básicos de cobre, en especial la brochantita, son productos muy frecuentes de la alteración oxidativa de minerales y materiales ricos en cobre, desde calcopirita a materiales como bronce o cobre en tuberías. La brochantita se puede presentar como pátinas verdes, con hábito masivo, granular y en forma de cristales aciculares, prismáticos o tabulares, a veces aislados y a veces formando masas o grupos esferoidales. En las muestras del filón "Terreras", la brochantita se encuentra tanto en forma de pátinas verdes y recubrimientos terrosos como de esférulas formadas por grupos radiales de cristales prismáticos elongados y vistoso color verde (Figura 5). La brochantita no puede distinguirse de otros sulfatos básicos de cobre mediante su espectro de emisión de rayos X (ya sea EDS o fluorescencia). Por ello, la técnica de elección es la espectroscopía Raman. Las muestras del filón "Terreras" se analizaron usando como excitación un láser de longitud de onda λ=532 nm. Se observa una banda mayor a 975 cm^{-1}, correspondiente con el modo ν1 de stretching del grupo sulfato (Figura 6).

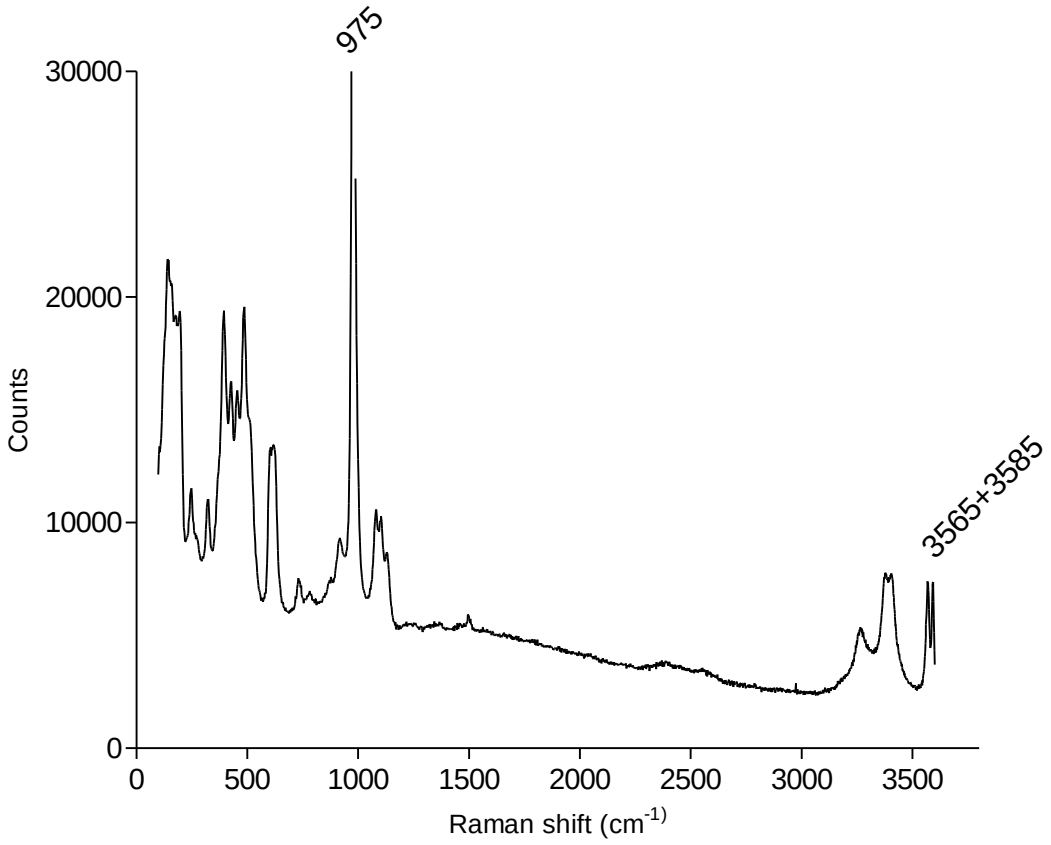

Figura 6: *Espectro Raman de la brochantita fotografiada en la figura 5.*

Otros aspectos distintivos son un doblete, correspondiente a la vibración del grupo hidroxilo, en 3565 y 3585 cm^{-1} y la intensidad relativa de las bandas de deformación del grupo hidroxilo a 389 y 483 cm-1. Estos datos están en perfecta coincidencia con datos reportados en la bibliografía para la brochantita (Bouchard y Smith, 2005), así como con el espectro de referencia de la base de datos RRUFF (http://rruff.info/). Sin embargo, W. Martens et al. (2003), sugieren que el modo principal de streching del sulfato en la brochantita aparece como una banda simple a 990 cm-1, y que los datos descritos anteriormente se corresponden realmente con la posnjakita ($Cu_4SO_4(OH)_6 \cdot H_2O$). En efecto, la posnjakita muestra la banda del modo ν1 del sulfato

a una frecuencia similar, sin embargo las intensidades relativas cambian y, por la presencia de agua adicional, esperaríamos un ensanchamiento en la zona de números de onda elevados (>3000 cm^{-1}), que no se observa en éste caso. Asimismo, los análisis realizados como contraste en muestras de brochantita confirmadas mediante difracción de rayos-X coinciden plenamente con los datos observados en las muestras del filón Terreras. No obstante, teniendo en cuenta además la química de la brochantita, que en atmósferas húmedas y en presencia de sulfato en solución se pseudomorfiza a posnjakita o langita, sería recomendable un estudio más profundo de los espectros Raman de éstos sulfatos básicos de cobre. En cualquier caso, dados los datos actuales, identificamos las muestras del filón "Terreras" como brochantita.

Devillina $CaCu_4(SO_4)_2(OH)_6 \cdot 3H_2O$

Dada la abundancia de sulfatos básicos en el filón "Terreras", la formación de devillina es esperable teniendo en cuenta la presencia de carbonatos en la brecha filoniana. La formación de sulfatos mixtos de calcio y cobre tiene lugar mediante la reacción de soluciones de sulfato de cobre con carbonato cálcico (Zhizhaev et al. 2007). En el filón Terreras la devillina se presenta en forma de láminas micáceas y rosetas de color azul verdoso o azul pálido y brillo nacarado, así como grupos de cristales hojosos de color azul verdoso y brillo subvítreo o nacarado (Figura 7).

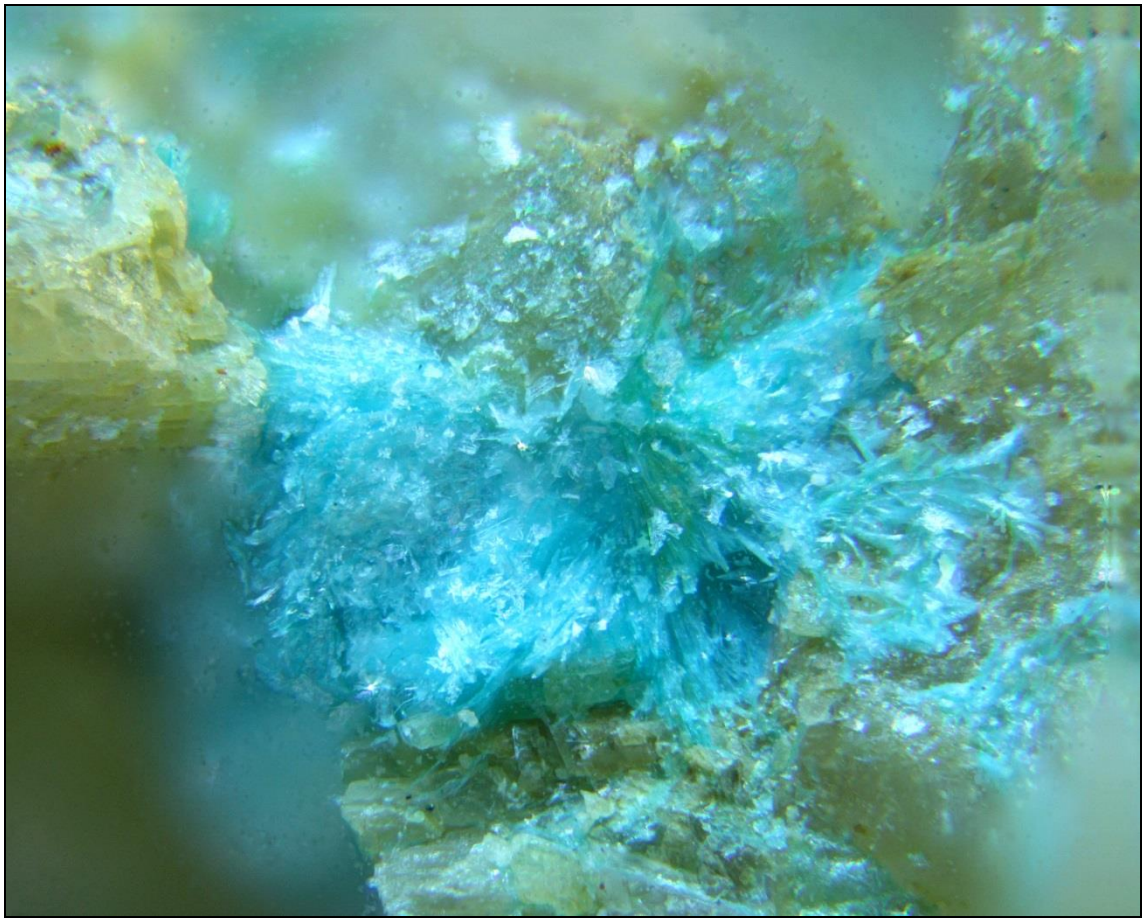

Figura 7: *Devillina. Filón "Terreras". Campo fotografiado 2.5 mm.*

Como en el caso anterior, la espectroscopía de emisión de rayos X no permite distinguir la devillina de otros sulfatos de cobre y calcio, como la ortoserpierita. El espectro Raman a λ=532 nm muestra una banda estrecha correspondiente al modo ν1 de

stretching simétrico del grupo sulfato a 1001 cm^{-1} y una banda característica correspondiente de stretching antisimétrico a 1135 cm^{-1}. El conjunto de bandas en 437 cm^{-1}, en la zona de bending de OH y sulfato, confirman la identificación de la devillina (Figura 8).

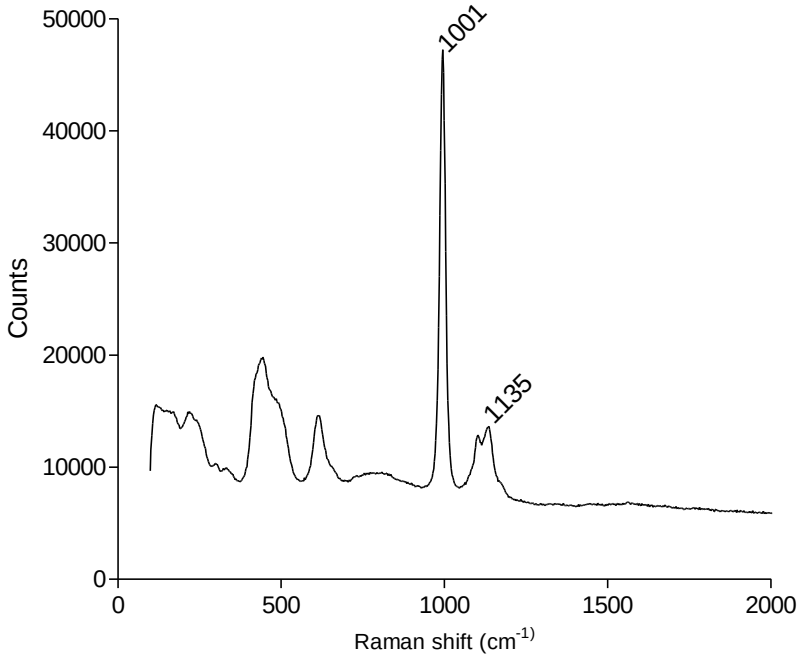

Figura 8: *Espectro Raman de la devillina del filón Terreras.*

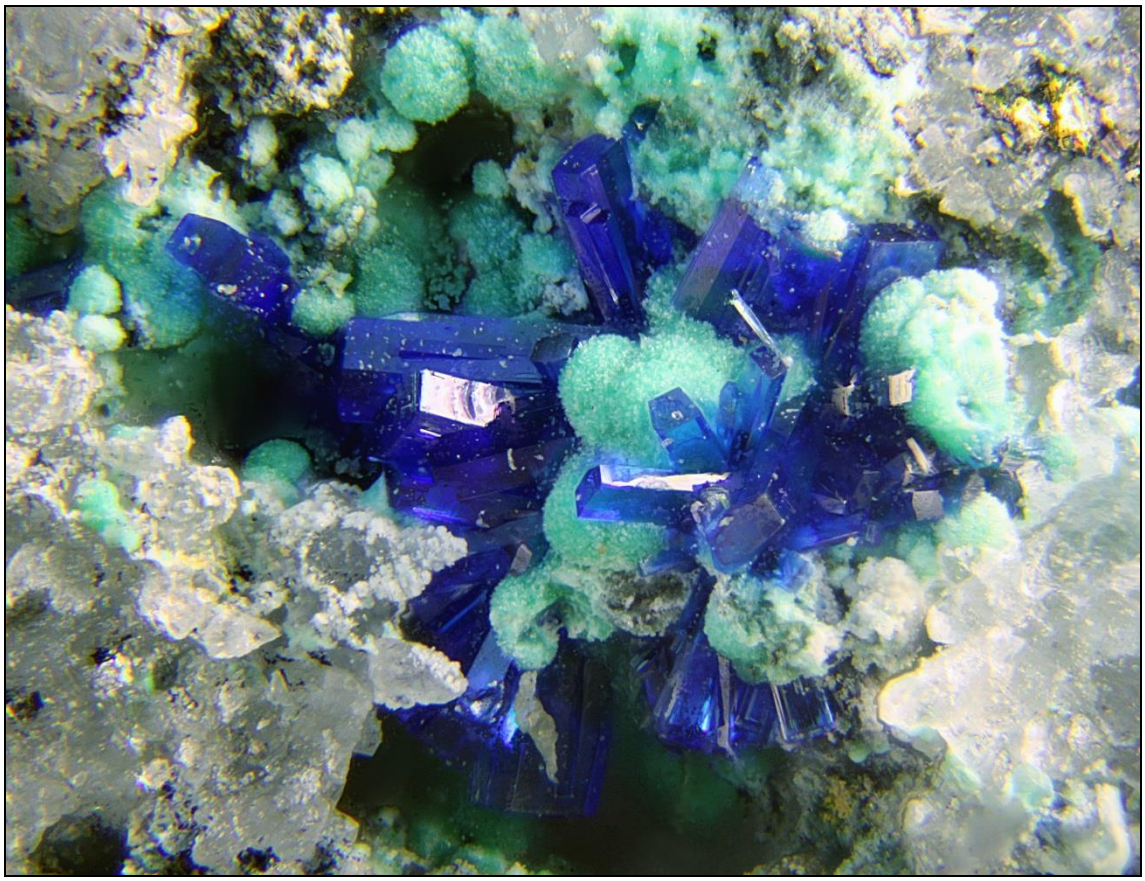

Figura 9: *Grupo de cristales de linarita, acompañados de sulfatos básicos de cobre. Filón "Terreras". Campo de visión 1 mm.*

Linarita $PbCuSO_4(OH)_2$

Posiblemente sea el mineral supergénico más interesante del yacimiento, tanto por su abundancia como por el tamaño y calidad de los cristales observados. Producto directo de la oxidación de sulfuros en yacimientos de Cu-Pb, como es el caso que nos ocupa, la linarita se presenta en el filón "Terreras" en forma de cristales tabulares o crecimientos paralelos o subparalelos de cristales muy finos, agrupados en grupos más o menos radiales, de color azul intenso y de intenso brillo vítreo. Los cristales más desarrollados observados son maclas de contacto en {100}, agrupadas radialmente (Figura 9). La linarita no se muestra aislada, asociándose con otros sulfatos básicos, principalmente brochantita. La identificación se realizó mediante espectroscopías EDS y Raman a λ=780 nm, observándose un espectro Raman caracterizado por una banda muy intensa a 968 cm^{-1} interpretada como el modo ν1 de stretching simétrico del anión sulfato (Figura 10A). El patrón de bandas característico entre 400 y 700 cm^{-1}, correspondiente a modos ν2 y ν4 del sulfato y las bandas situadas en la zona 80-200 cm^{-1}, interpretadas como vibraciones de red asociadas a Cu-OH y Pb-OH son coherentes con posible linarita.

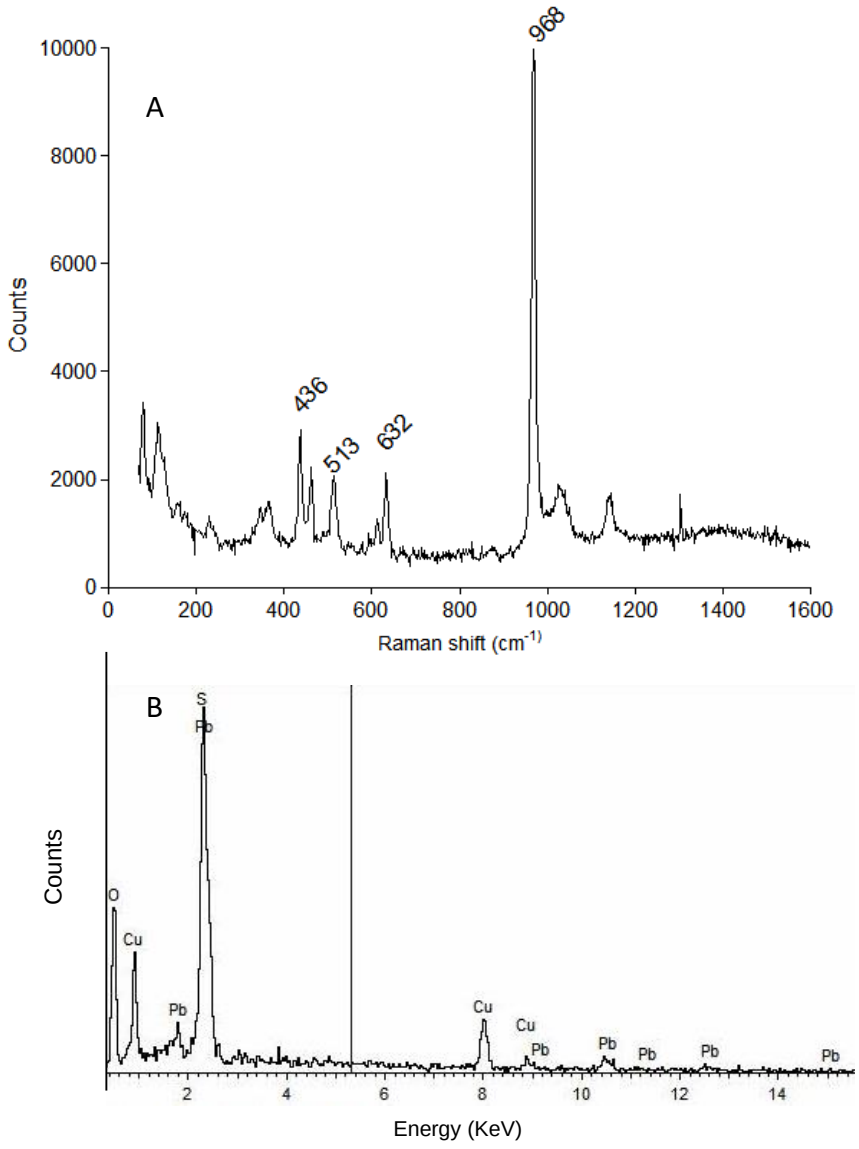

Figura 10: *A: Espectro Raman de la linarita del filón "Terreras". B: espectro EDS de los cristales de linarita mostrados en la figura 5.*

Tabla I

Filón Terreras	Bouchard y Smith (2005)	Buzgar *et al*. (2009)	Asignación
230	230	232	Cu-OH; Pb-OH
347	345	347	
365	365	366	
436	436	437	$\nu2$ (SO_4^{2-})
462	461	465	
513	513	518	
632	632	634	$\nu4$ (SO_4^{2-})
968	968	967	$\nu1$ (SO_4^{2-})
1024	1019	1021	$\nu3$ (SO_4^{2-})
1142	1141	1143	$\nu4$ (SO_4^{2-})

Tabla I: *Bandas Raman de la linarita del filón Terreras, en comparación con los datos publicados. Datos de desplazamiento Raman en número de onda (cm^{-1}).*

Los datos obtenidos coinciden perfectamente con los reportados en la bibliografía (Bouchard y Smith, 2005; Buzgar et al., 2009). El espectro EDS muestra la siguiente composición: 16% Cu, 55% Pb y 10% S, indicando una relación atómica Pb:Cu aproximadamente 1:1 y confirmando la identificación de la linarita del filón "Terreras" (Figura 10B).

METODOLOGÍA

Las determinaciones mineralógicas se realizaron utilizando dos espectrómetros Raman BW-Tek modulares, equipados con sensores de array CCD lineal de alta resolución enfriados mediante célula Peltier y con módulos láseres de excitación a 532 y 785 nm y 300 mW de potencia. La resolución espectral es de 2.5 cm^{-1} y el rango de medida es de 65 a 4200 cm^{-1}. El sistema de espectrometría se acopla en un microscopio Olympus, realizándose las medidas a través de objetivos de 20x PL NA: 0.40, WD 16mm. Dada la sensibilidad de los sulfatos básicos de cobre al laser (Figura 11), las adquisiciones se realizan a una potencia de láser del 10%, incrementando el tiempo de adquisición. El tratamiento de la señal es estándar, con reducción de ruido mediante filtrado Savitzky-Golay e identificación de picos mediante acoplamiento a una función Lorentziana y realizando el reprocesamiento de los espectros por cálculo de la segunda derivada de la señal.

Los análisis mediante espectrometría dispersiva EDS se realizaron en un microscopio electrónico de barrido JEOL 5600LV equipado con un sistema de microanálisis INCA X-sight de Oxford Instruments, a un voltaje de aceleración de 20 KV.

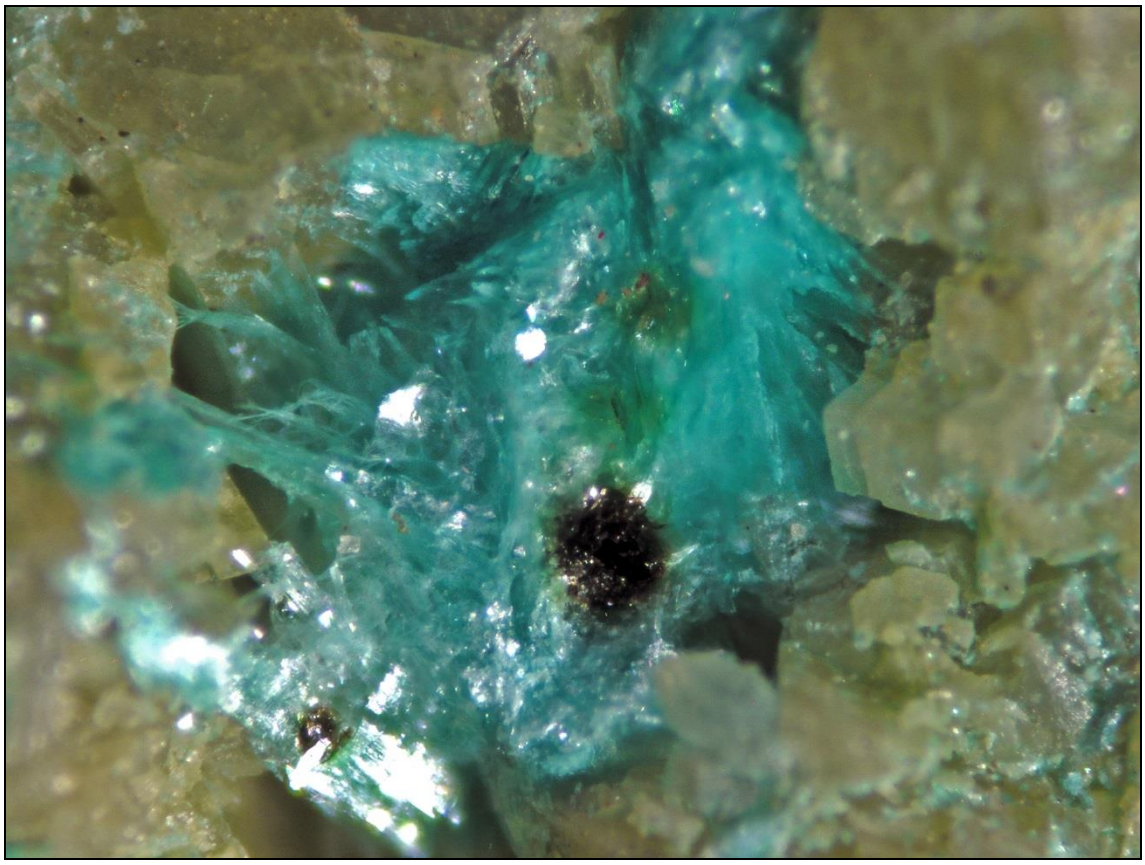

Figura 11: *Detalle de una muestra de devillina del filón "Terreras" mostrando un problema habitual en la espectroscopía Raman de este tipo de sales: son muy sensibles al láser de excitación, lo que ocasionalmente provoca la alteración o volatilización de la muestra. Campo de visión 0.9 mm.*

BIBLIOGRAFÍA

E. BOIXEREU, F. PALERO, J. MARIMÓN, C. FREIXAS (2011) Descripción de las mineralizaciones de la Zona Centro Ibérica. En: *Cartografía de recursos minerales de Andalucía* (A. García-Cortés, Ed. ppal.), IGME-Consejería de Economía, Innovación y Ciencia de la Junta de Andalucía, Madrid. Pp. 61-82.

M. BOUCHARD, D.C. SMITH (2005) Database of 74 Raman spectra of standard minerals of relevance to metal corrosion, stained glass or prehistoric rock art. In: *Raman spectroscopy in Archaeology and Art History* (H.G.M. Edwards y J.M. Chalmers, Eds.), RSC Analytical Spectroscopy Monographs, Cambridge, UK. Pp. 429-464.

N. BUZGAR, A. BUZATU, I.V. SANISLAV (2009) The Raman study of certain sulfates. *Analele Ştiintifice ale Universitatii "AL. I. CUZA" IAŞI Geologie,* **55**: 6-23.

J.M. CANO-SANCHIZ (2010) Tecnología córnica para el desagüe de minas: motores y casas de bombeo tipo Cornish. *De Re Metallica*, **15**: 13-20.

A. CARBONELL TRILLO-FIGUEROA (1929) Importancia Minero-Metalúrgica de la provincia de Córdoba. *Boletín de la Cámara Oficial Minera de Córdoba*, Año III, **12**: 17-22.

A. CARBONELL TRILLO-FIGUEROA (1929) *Criaderos de plomo de la provincia de Córdoba*. Tomo VII. IGME, Madrid.

J. GARCÍA ROMERO (2002) *Minería y Metalurgia en la Córdoba romana*. Universidad de Córdoba, Servicio de Publicaciones.

W. MARTENS, R.L. FROST, J.T. KLOPROGGE, P. WILLIAMS (2003) Raman spectroscopic study of the basic copper sulphates-implications for copper corrosion and 'bronze disease'. *Journal of Raman Spectroscopy*, **34**: 145–151. doi: 10.1002/jrs.969.

VV.AA. (1967) *Proyecto de Investigación para minerales de cobre. Zona de Villanueva de Córdoba-Alcaracejos*. IGME.

A.M. ZHIZHAEV, E.N. MERKULOVA, I.V. BRAGIN (2007) Copper precipitation from sulfate solutions with calcium carbonates. *Russian Journal of Applied Chemistry*, **80:** 1632-1635.

Manuscrito original recibido el 10 de febrero de 2014.
Publicado: 15 de febrero de 2014.

ACOPIOS2014 V5: 15-132
Revista Ibérica de Mineralogía

MTIEDIT
ISSN 2171-7788

Stack_Duino, un sistema automático y autónomo para la adquisición de fotografías multifoco destinado a documentación científica

Honorio CÓCERA LA PARRA

Museo de Geología, Dpto. de Geología, Universitat de València
Doctor Moliner, 50, E-46100 Burjassot (Spain)
hocolapa@gmail.com

Resumen

H. CÓCERA LA PARRA (2014) **Stack_Duino, un sistema automático y autónomo para la adquisición de fotografías multifoco destinado a documentación científica.** *Acopios,* **5**: 15-132.

La fotografía científica sigue siendo uno de los métodos más usados en numerosos campos de estudio de la ciencia, permitiendo el registro, documentación y comunicación de información. La fotografía científica es un conjunto de técnicas entre la que se encuentra la fotografía multifoco o focus stacking, usada principalmente en fotomicrografía, que produce imágenes bidimensionales de objetos tridimensionales con una profundidad de campo mayor que la permitida por la difracción, combinando las zonas en foco de una pila o serie de imágenes tomadas sobre un objeto a diferentes planos focales cambiando la distancia entre el objeto y la cámara a pequeños intervalos. Una vez tomada la pila, un software adecuado se encarga de la fusión para obtener una imagen totalmente enfocada. A pesar de que el software para esta técnica ha avanzado rápidamente en los últimos años en eficacia y facilidad de uso, el proceso manual de toma de imágenes es tedioso y tiene un coste elevado de tiempo. El siguiente trabajo describe el diseño, construcción y programación de un sistema automático y autónomo para la toma de esta serie de imágenes basado en la plataforma Arduino, además de discutir sobre las bases y uso de la fotografía multifoco.

Palabras clave: Fotografía científica, multifoco, focus stacking, Arduino, Stack_Duino, profundidad de campo, fusión de imágenes, construcción y programación.

Abstract

H. CÓCERA LA PARRA (2014) **Stack_Duino, an automatic and autonomous system for the acquisition of multifocus photography for scientific documentation.** *Acopios,* **5**: 15-132.

The scientific photography remains one of the most used methods in many scientific disciplines, allowing the registration, documentation and communication of important information. The scientific photography includes a set of techniques, among them the focus stacking points out. This technique, mainly used in photomicrography, produces two-dimensional images with a wider depth of field than that reached by diffraction, combining the focus areas of a series of images taken from an object in different focal plane by changing the distance between the object and the camera at small intervals. Once these series of pictures are taken, using specialized software, the images are combined using the different focused areas of each one, giving as a result an image completely focused. While the software for this technique has advanced rapidly in recent years, both in accuracy and use, the manual process of taking pictures is tedious and high time consuming. The following work discribes the design, construction and programming of an automatic and autonomous system for making this series of images based on the platform Arduino, besides discuss the utility and practices of the multifocus photography.

Keywords: Scientific photography, multifocus, focus stacking, Arduino, Stack_Duino, depth of field, image fusión, construction and programming.

1. INTRODUCCIÓN

El hombre siempre ha sentido la necesidad de representar la realidad que le rodea y transmitirla. Esta representación ha variado en correspondencia con la comprensión que se ha tenido de la realidad y de los medios, formatos y tecnologías alcanzados en cada época y sociedad. Desde los pictogramas, la primera forma de representar la información, hasta las actuales herramientas surgidas gracias a las posibilidades que brinda el mundo digital, se han sucedido una serie de medios que han contribuido a ampliar el cuantioso espectro de la información, su representación y difusión; y han contribuido especialmente en lo que nos atañe, a la evolución de la ciencia y a la comunicación y divulgación científica.

La documentación como disciplina académica, surge en respuesta al gran desarrollo de las nuevas necesidades de la ciencia en el siglo XX: el crecimiento incesante de los documentos, su gestión, la pluriformidad de éstos, los avances y consumos científicos. Se comienza a concebir el documento como la información que se recoge en cualquier soporte. El nacimiento en el siglo XX y XXI de nuevos medios técnicos, entre los que se encuentra la fotografía científica, provocaron la aparición de este campo que tiene como objeto de estudio del documento, la gestión de la documentación y las operaciones implicadas en ella.

Al igual que las otras fuentes documentales, la fotografía, se compone por mensaje, formato y soporte, se distingue por contener información, por transmitir conocimiento y por su intención comunicativa (Sánchez, 2011). La fotografía emerge en el siglo XIX como nueva forma de representación, trascendiendo todas las esferas de la vida humana y la comunicación visual; por su capacidad para la reproducción exacta, fue concebida como una herramienta de observación y un medio de documentación preciso y fidedigno (Schwartz, 2000), dando lugar así, a la fotografía científica.

La fotografía es el proceso que utiliza la captación del espectro electromagnético por un material sensible a él, para capturar, registrar y producir imágenes duraderas. La fotografía, en su visión general, no es la realidad, sino uno de los muchos modos de representarla y tiene por tanto un amplio grado de subjetividad. En contraste, la fotografía científica, que debe ser objetiva y unívoca conforme con el método científico, debe poseer de valor documental al ser un canal visual de transmisión de información y un contenedor de conocimientos, con el objetivo último de servicio y herramienta a la comunidad científica.

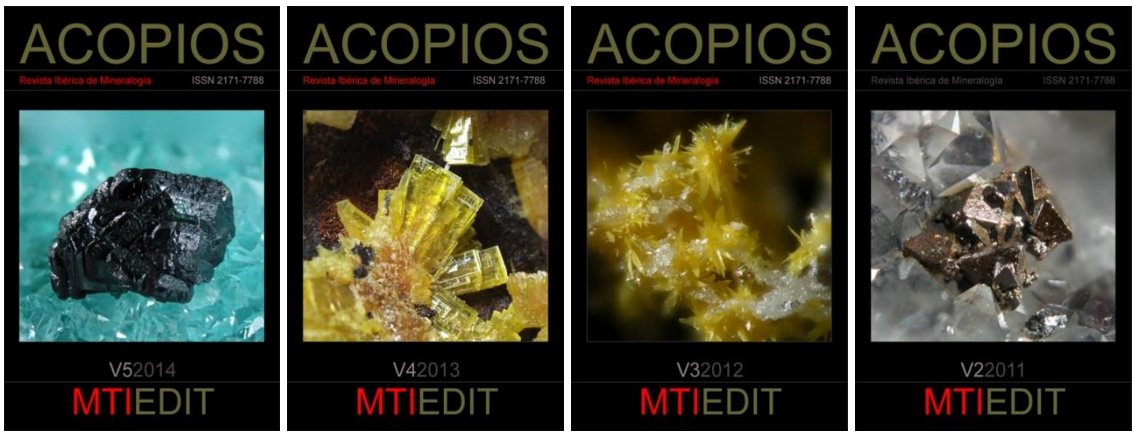

Figura 1: *Diferentes imágenes científicas obtenidas mediante la técnica de multifoco ilustrando la portada de la revista Acopios. (CC BY NC) MTI Project-Acopios.*

La fotografía científica, también llamada *fotografía aplicada*, no es una especialidad, es un grupo de técnicas fotográficas destinadas a obtener información valiosa en forma de

imágenes para la investigación o el control de procesos, en todas las ramas de la ciencia, la industria y la educación. (Monje, 2010). De manera resumida, podríamos decir que una imagen científica es una herramienta de interpretación y de reconstrucción del mundo real para el científico.

El uso de la fotografía científica tiene dos grandes ámbitos de aplicación. El primero comprende la obtención de información, documentación e ilustración sobre los procesos, objetos o eventos experimentales inherentes a la investigación. Tiene como función visualizar, fijar y demostrar esas observaciones durante el proceso de investigación (Savazzi, 2011), tanto como de ilustrar sus conclusiones finales para la literatura científica, requeridas en numerosos campos como la química, ciencia de los materiales, física, la biología, medicina, geología, arqueología, etc…En algunos campos no es necesaria para documentar los datos experimentales, pero si, para mostrar el equipamiento usado o el montaje experimental. También es de uso común en la conservación de arte, historia o colecciones científicas de diferentes temáticas, teniendo un gran auge en los últimos tiempos para la creación de bases de datos sobre estas colecciones que posteriormente son compartidas de forma *online* con el resto de la comunidad científica o público general.

El segundo comprende el ámbito divulgativo y educacional, con el fin de acercar y difundir la ciencia al público general. Para su uso en prensa, televisión, exposiciones, libros de texto, etc… En este contexto, se presta una mayor atención al aspecto estético que al estrictamente científico.

La fotografía óptica sigue siendo una de las técnicas de documentación preferidas, por su facilidad de uso y economía, aunque continúa sufriendo ciertas limitaciones impuestas por las leyes de la óptica y la naturaleza de la luz. Los humanos tenemos la habilidad de ver en 3 dimensiones una escena a pesar de que nuestro ojo solo es capaz de vez enfocada una parte de ella, y construir un mapa mental con todas las regiones de la escena en foco en nuestro cerebro, únicamente nuestro sistema visual capta la zona central enfocada. Sin embargo, en un sistema óptico, cuando se intenta capturar una escena en tres dimensiones, solo una parte de ésta, el plano focal, aparece enfocada y nítida; fuera de este campo la imagen aparece borrosa y desenfocada. Se llama profundidad de campo, PDC o DOF en inglés, al espacio por delante y por detrás del plano focal, comprendido entre el primer y el último punto aceptablemente nítido de la escena. A pesar de ello, el uso combinado de técnicas de adquisición digital y de software adecuado permite la obtención de imágenes completamente enfocadas. A continuación discutiremos la base teórica de este fenómeno, y ello nos permitirá entender con mayor facilidad la base de la fotografía multifoco, que describiremos posteriormente.

1.1. Formación de la imagen

Una imagen es una representación en dos dimensiones (2D) de un objeto o motivo tridimensional (3D). Cuando la luz procedente de diferentes puntos de un motivo pasa a través de una lente o un objetivo y son reconstruidos como una imagen, los distintos puntos del motivo se muestran en la imagen como pequeños patrones y no como puntos. Este fenómeno es causado por la difracción y dispersión de la luz a medida que pasa a través de las diminutas partes y espacios de la muestra y a través del diafragma del objetivo.

Uno de los parámetros más importantes en un sistema óptico, sea un microscopio, lente o telescopio, es la resolución, que es la capacidad que tiene un sistema óptico de discernir entre dos puntos. Podríamos pensar, que la imagen de un objeto, se compone

de un número de puntos infinitamente pequeños, pero esto es incorrecto, la resolución es finita, y es función de los parámetros de diseño del sistema óptico y de la naturaleza de la luz. Como comentábamos, cuando la luz que procede de diferentes puntos de un motivo pasa a través de un objetivo, es reconstruida como una imagen, los varios puntos del motivo se muestran en la nueva imagen formada como pequeños patrones, esos puntos del motivo aparecerán como patrones de difracción en el plano de focal, conocidos como patrones de Airy. Este fenómeno es producido por la difracción y por dispersión de la luz al pasar a través de la apertura o diafragma del objetivo.

El máximo central del patrón de Airy es normalmente conocido como disco de Airy, que es definido como la región encerrada por el primer mínimo del patrón de Airy y contiene el 84% de la energía luminosa (Davinson, 2009). La figura 2(A) muestra un disco de Airy y la distribución de intensidad lumínica.

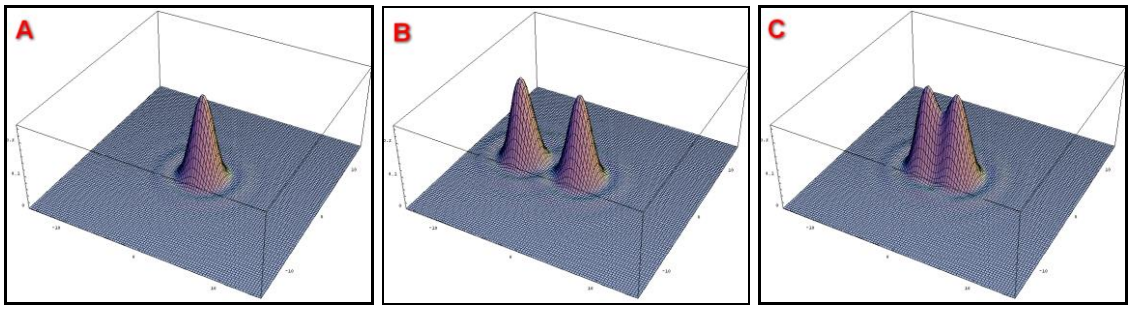

Figura 2: *Representación de discos de Airy. (CC BY 2.0) (Padley, 2005).*

Entonces podemos definir la resolución como la mínima distancia entre dos discos de Airy a la que ambos discos todavía pueden ser resueltos, y al límite en que pueden ser resueltos se le llama criterio de Rayleigh. Cuanto más pequeño sea el disco de Airy proyectado por el objetivo en la formación de la imagen, más pequeño será el valor de la resolución, y más detalles será posible discernir en la muestra o en el espécimen, más nítida será la imagen. La figura 2(B) muestra dos discos de Airy bien resueltos, mientras la figura 2(C) muestra dos discos sin resolver, dando lugar a una muestra menos nítida y más borrosa.

La resolución antes discutida se puede descomponer en dos componentes. La primera es la llamada resolución horizontal, es perpendicular al eje óptico y puede ser descrita matemáticamente como (Davidson, 2009):

Ecuación 1: $R = \frac{0{,}61 * \lambda}{NA}$

donde:
R= resolución en nm.
λ = longitud de onda de la luz de nm.
NA= apertura numérica del objetivo o lente, es función diafragma efectivo, el cual a su vez, depende diafragma real del sistema óptico usado y de la magnificación a la que estemos trabajando, según:

Ecuación 2: $NA = \frac{1}{2 * F\ (M + 1)}$

donde:
F= diafragma real del objetivo.
M= magnificación a la que estamos trabajando.

Cuanto más alto sea el valor de NA, más pequeños serán los discos de Airy, más bajo será el valor de R y mayor será el detalle del espécimen que será discernible. Objetivos que tengan una mayor apertura numérica serán capaces de producir imágenes más detalladas. La resolución horizontal es la responsable de la nitidez y detalle de la imagen.

La segunda resolución, o resolución axial, que posee todo sistema óptico, se mide en el plano paralelo al eje óptico, es más conocida como profundidad de campo (PDC). Representa la distancia por delante y por detrás del plano enfocado donde el objeto aparenta estar en foco. PDC es finito y es función de la longitud de onda de la luz y de la inversa al cuadrado de la apertura numérica, se expresa matemáticamente como:

Ecuación 3: $D = \frac{1{,}22 * \lambda}{NA^2}$

donde:
D= profundidad de campo, PDC, en nm.
λ= longitud de onda de la luz en nm.
NA = apertura numérica del objetivo.

Combinando la ecuación 1 y la ecuación 2 podemos obtener la relación de R frente a PDC (D).

Ecuación 4: $D = \frac{3{,}27 * R^2}{\lambda}$

la Figura 3 muestra la dependencia entre la PDC y la Resolución horizontal, la nitidez.

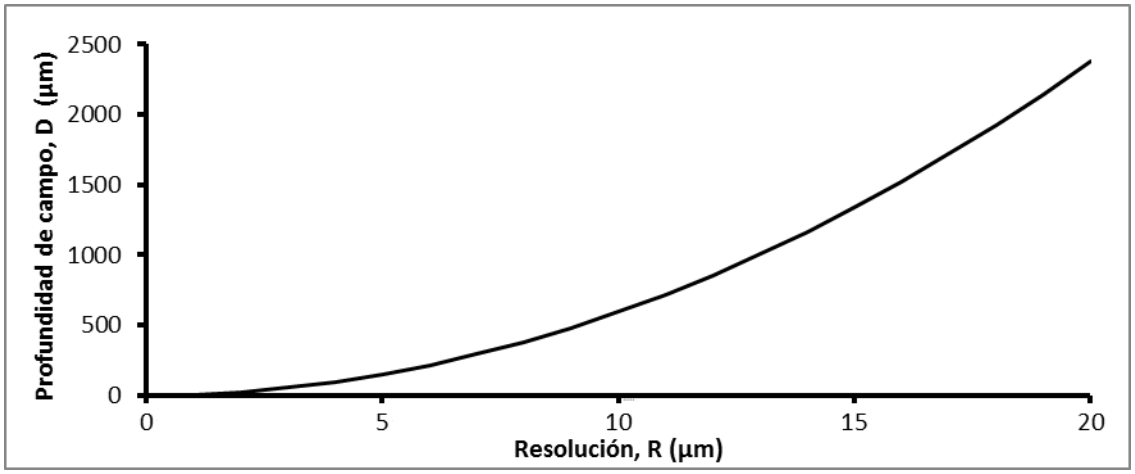

Figura 3: *Profundidad de campo frente a resolución horizontal para λ=550nm.*

Podemos observar en la figura 3 como, la PDC disminuye a medida que R se hace más pequeño, o lo que es lo mismo, aumenta el detalle que nuestro sistema óptico es capaz de discernir con la disminución de la PDC; y al contrario, imágenes con una gran profundidad de campo tendrán poca nitidez, todo ello debido a la difracción y la dispersión de la luz. Para objetos voluminosos esto no es un gran problema. Sin embargo, cuando trabajamos con objetos muy pequeños, este problema se acrecienta, llegando a ser crítico en microscopía. No hay que olvidar que el Valor de NA es dependiente de la magnificación, y conforme trabajamos a más aumentos, el valor de R aumenta, perdiendo detalle y nitidez rápidamente. La tabla 1 muestra los valores de PDC para diferentes objetivos de microscopio.

Recordemos que valores altos de R corresponden a menor resolución y menor detalle en la imagen.

Tabla I

Magnificación	Apertura Numérica	PDC (μm)
4x	0.10	15.5
10x	0.25	8.5
20x	0.40	5.8
40x	0.65	1.0
60x	0.85	0.40
100x	0.95	0.19

Tabla I: *PDC en función de NA y la magnificación.*

Debido a todo esto, el uso de microscopios ópticos muestra los problemas generados por la limitada profundidad de campo. Para la observación en vivo, el objeto se moverá arriba y abajo a lo largo del eje óptico, si esta operación se produce de forma rápida, muchas capas podrán ser observadas en foco en un corto intervalo de tiempo, produciendo en nuestro celebro la ilusión de una imagen global enfocada.

Sin embargo, este procedimiento no puede ser usado para la obtención de imágenes fotográficas, debido, a que solo una imagen puede ser vista y deberemos decidir por que capa debe ser retenida. Por ejemplo, a pequeñas magnificaciones usando un objetivo de 10x, la PDC baja hasta un valor de 8,5 micras, (tabla 1), si la muestra tiene una profundidad mayor que este valor, tan solo podremos tener una parte de ésta en foco, registrando solo parte de la información de la muestra y perdiendo el resto de la información que puede llegar a ser de gran interés.

Resolver el problema de la limitada PDC al trabajar a altas ampliaciones, es de gran interés para muchas ramas de la ciencia. Se han propuesto diferentes técnicas en los últimos años. Básicamente se pueden basar en dos grupos: El primer grupo es el basado en la variación de los parámetros ópticos del instrumento, según la ecuación 3 y 4, PDC es una función del diafragma del sistema óptico, si cerramos el diafragma lo que corresponde a valores F mayores, conseguimos incrementar la profundidad, pero a costa de una pérdida de resolución y de detalle de la imagen, aunque se consigue aumentar la cantidad de información registrada se degrada la calidad de ésta. En la figura 4 podemos ver con facilidad este fenómeno sobre imágenes tomadas sobre la superficie de una moneda colocada a 45º respecto el sensor de la cámara, la imagen 4(A) y 4(C) están tomada con el diafragma completamente abierto, podemos observar como la parte en foco es más pequeña pero mucho más nítida, las imágenes 4(B) y 4(D) están tomada con el diafragma completamente cerrado, aunque la imagen aparece totalmente en foco pero mucho más degradada, la calidad de la información es mucho menor, llegando a ser indiscernible cualquier detalle.

El otro grupo está basado en el uso de algoritmos aplicados sobre una pila de imágenes tomadas de tal forma que cada imagen tiene una parte en foco de la escena. Estos algoritmos son capaces de detectar las áreas enfocadas de cada una de las imágenes de la pila, y combinarlas en una única imagen que contenga toda la PDC del objeto en cuestión. Esta técnica es llamada *multifocus* o *focus stacking*. Los avances que se han realizado con el paso del tiempo y la aparición y uso en la actualidad de la fotografía digital y la capacidad de cálculo de los ordenadores actuales, han ampliado y extendido su rango de uso, haciéndose imprescindible en algunos campos de trabajo como la mineralogía, paleontología o entomología.

Figura 4: *Imagen de la superficie de una moneda puesta a 45º respecto del plano de la cámara. A y B tomadas a 3 aumentos, C y D tomadas a 10 aumentos. A y C a diafragma totalmente abierto, B y D a diafragma totalmente cerrado mostrando mayor PDC pero menor calidad de imagen.*

1.2. Fotografía multifoco

La fotografía multifoco pertenece a las llamadas técnicas de fusión de imágenes. En la fusión de imágenes, la información relevante que proveniente de dos o más imágenes de la misma escena es combinada en una sola para generar una nueva que contenga toda la información relevante dispersa en las demás. El objetivo de la fusión de imágenes es aumentar la información, que puede provenir de una misma fuente o de diferentes, para posteriormente ser usada en una aplicación particular (Ardehir, 2006). Para ello, la obtención de la pila de imágenes debe ser tomada de tal forma que la información de interés varíe entre cada una de las tomas o imágenes, para ello, cada una estará tomada a diferentes tiempos o con pequeñas variaciones de algún parámetro. Un ejemplo clásico es la fotografía astronómica, en la que la imagen final es suma de la exposición de las diferentes tomas para obtener una con la exposición correcta.

Como ya se ha comentado, la fotografía multifoco, permite franquear los límites de la PDC y la difracción, obteniendo una mayor PDC y en consecuencia una mayor cantidad y calidad de la información sin la correspondiente pérdida de detalle. La técnica consiste por tanto, en la toma de una serie sucesiva de imágenes enfocadas a diferentes planos sobre el objeto, tomadas con los parámetros que permiten la mayor resolución posible, aunque presenten poca PDC. Durante la toma de la pila, se varía la distancia del sistema cámara+lentes frente al objeto, en incrementos constantes, con lo que el plano focal cambia en cada fotografía levemente, manteniendo constantes la iluminación, el tiempo de exposición y demás parámetros.

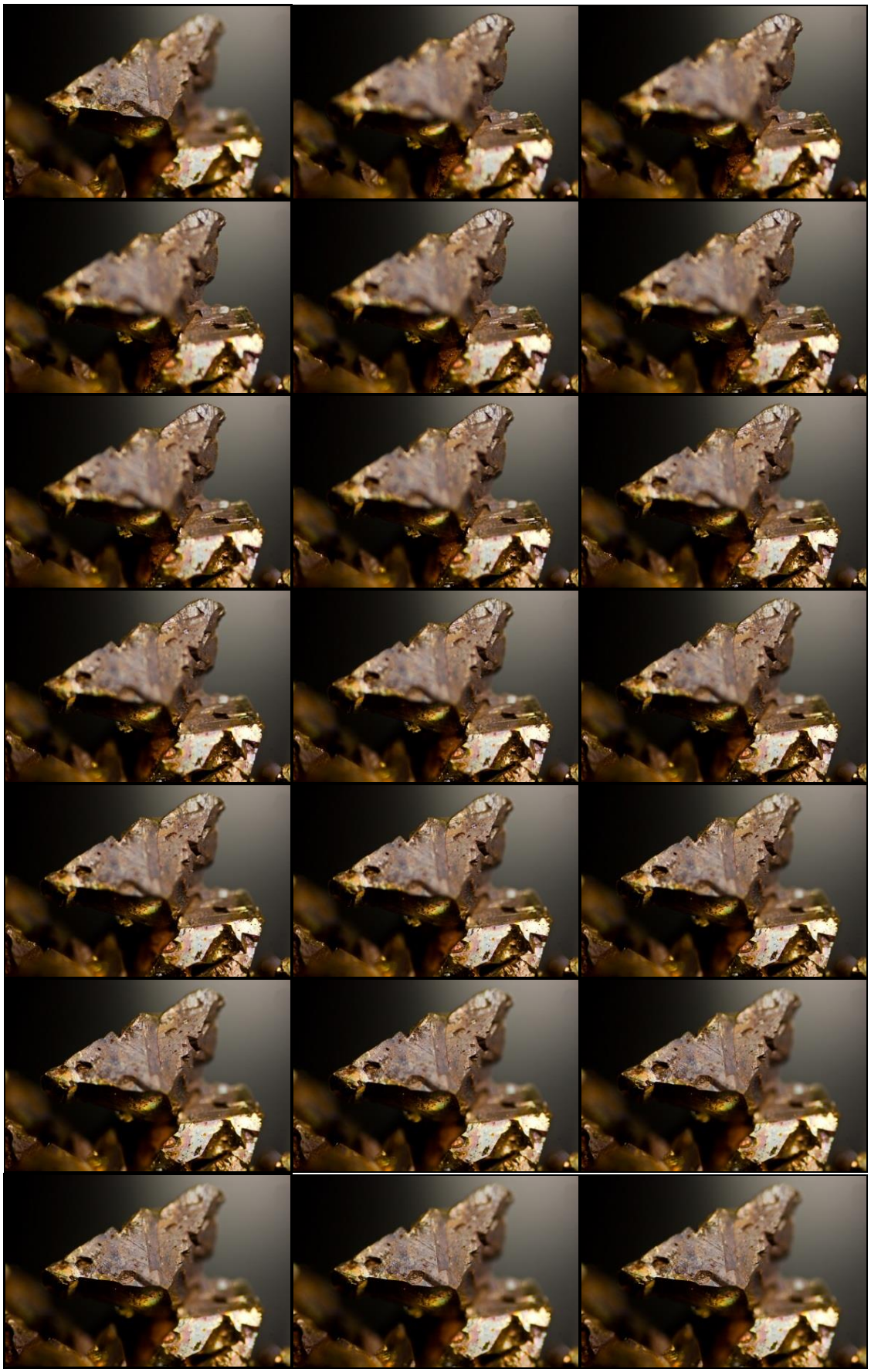

Figura 5: *Pila de imágenes mostrando la variación del plano focal entre cada una de ellas sobre un cristal de cobre nativo de 4 mm. (Col. y fot. H. Cócera).*

Posteriormente, cargando la pila en el software adecuado, se generará la imagen que contiene todas las áreas del sujeto en foco. Como principal desventaja cabe decir que esta técnica solo es aplicable a sujetos estáticos. La Figura 5 muestra una pila de imágenes tomada sobre un cristal de cobre nativo de 4mm de arista. La Figura 6 muestra el resultado de combinar la pila de imágenes del ejemplo anterior.

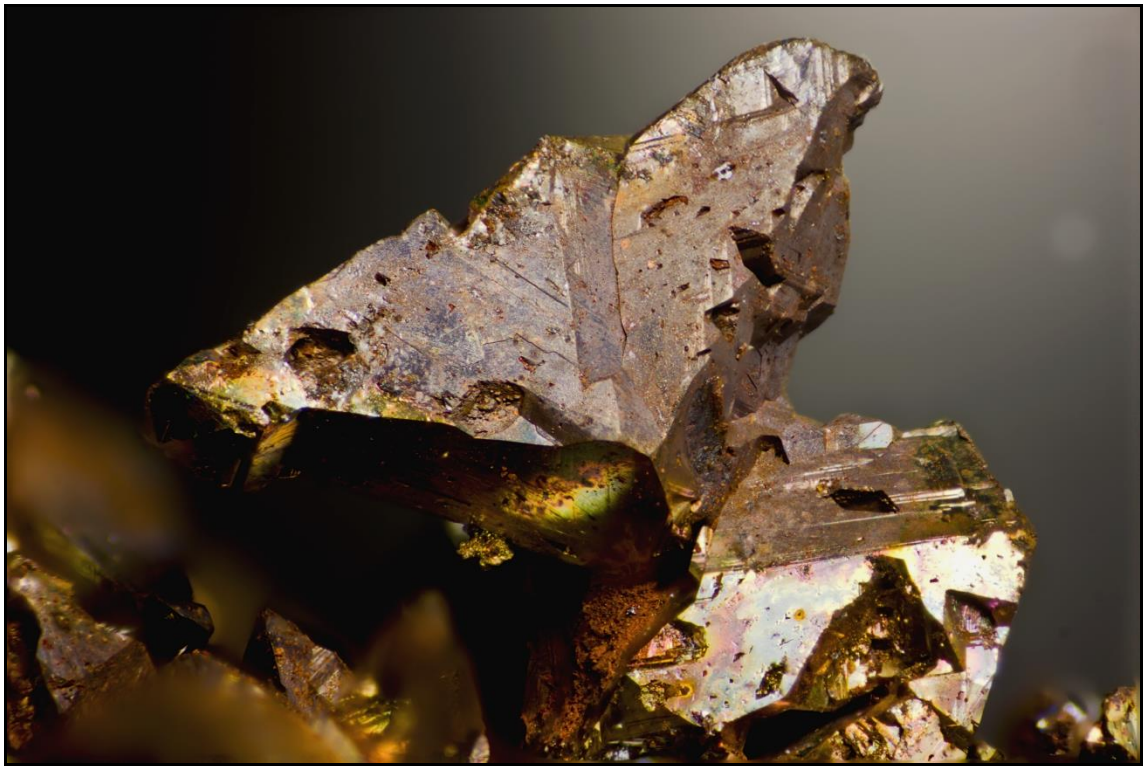

Figura 6: *Resultado de combinar la pila de imágenes de la figura 5. Cristal de cobre de El Valle, Begega, Asturias. 4mm. (Col. y fot. H. Cócera).*

2. OBJETIVOS

La toma de la pila de imágenes de forma manual es una labor lenta y tediosa, pudiendo llegar a ser necesaria la toma de varios cientos de tomas, consumiendo una gran cantidad de tiempo e introduciendo errores debidos al operador. Sin embargo en el mercado existen equipos que hacen este trabajo de forma automática, aunque su precio suele ser elevado y su software propietario, muchas veces solo aplicable a un modelo de microscopio, no pudiendo ser cambiado o adaptado a nuestras necesidades. También se pueden encontrar en la red, diferentes montajes de código abierto, aunque son muy rudimentarios y poco configurables.

El presente trabajo propone el diseño y construcción de un equipo automático y autónomo para la toma de la pila de imágenes, al que llamaremos **Stack_Duino**. Pretende ser una alternativa de bajo coste y fácil montaje, permitiendo un flujo de trabajo elevado y pudiendo ser adaptado fácilmente a cualquier sistema de toma de pilas de imágenes, compatible con cámaras DSLR y de microscopía.

El diseño, construcción y programación del sistema automático y autónomo para la adquisición de fotografías multifoco, debe permitir:

- Motorización y automatización de equipos ópticos, microscopios y estereomicroscopios así como la toma de pilas de imágenes a través de ellos.
- Control de cámaras DSLR y de microscopía.

- Diseño de bajo coste económico basado en la plataforma Arduino, con componentes fácilmente accesibles y fácilmente construible.
- El diseño debe permitir un alto flujo de trabajo.
- Debe ser fácilmente adaptable a las necesidades particulares de cada operador o disciplina en la que se use.
- El dispositivo debe ser simple e intuitivo de operar.
- Alta precisión y repetitividad en el control del posicionamiento de la cámara y de las lentes en relación con el sujeto.

3. MATERIALES Y MÉTODOS

Durante mucho tiempo, el control de los microscopios o sistemas ópticos fue un proceso laborioso que implicaba la manipulación de muchos componentes de hardware. Históricamente, las imágenes fueron registradas por primera vez mediante el uso de dibujos a mano, posteriormente en película fotográfica y más tarde en archivos digitales. El desarrollo de las cámaras digitales, junto con el control por ordenador de otros componentes del microscopio, ha incrementado drásticamente la facilidad de uso y ampliado las posibilidades para los investigadores. La informatización del control del disparador motorizado, ruedas de filtros y otros dispositivos, permite ahora a los usuarios controlar de forma precisa y repetidamente el color, la intensidad y la duración de la luz y de iluminación que incide sobre una muestra. La motorización, permite el automatizado del enfoque y la adquisición rápida de diferentes planos de la imagen a lo largo del eje óptico. Las cámaras digitales facilitan al investigador adquirir un gran número de imágenes. El almacenamiento de imágenes digitales y su procesamiento ofrecen a los usuarios la oportunidad de guardar, anotar y analizar estas grandes colecciones de imágenes.

Con la aparición y alza de las cámaras digitales y la motorización de los componentes mecánicos, el control informatizado de los microscopios y el desarrollo del software, se ha convertido en una herramienta prácticamente imprescindible para el investigador, al permitir nuevas técnicas antes impensables que aportan nueva información, como por ejemplo la microscopía confocal, de fluorescencia o sobre la que versa este trabajo, la multifoco.

Un sistema de adquisición de imágenes multifoco consta de tres partes, electrónica, firmware y mecánica.

3.1. Electrónica

3.1.1. Arduino

El corazón de Stack_Duino es una placa Arduino. Arduino es una plataforma de electrónica abierta para la creación de prototipos basada en software libre y hardware flexibles y fáciles de usar (Arduino, 2014). Arduino puede tomar información del entorno monitorizándolo a través de sus pines de entrada de toda una gama de sensores y puede actuar sobre aquello que le rodea controlando luces, motores y otros actuadores. El microcontrolador en la placa Arduino se programa mediante el lenguaje de programación Arduino (basado en Wiring) y el entorno de desarrollo Arduino (basado en Processing). Los proyectos hechos con Arduino pueden ejecutarse sin necesidad de conectarse a un ordenador, si bien tienen la posibilidad de hacerlo y comunicar con diferentes tipos de software (p.ej. Flash, Processing, MaxMSP). Detrás de Arduino existe una gran comunidad usuarios en cuya base radica el gran éxito y expansión que ha tenido. Existe al alcance de cualquier usuario final una serie de documentos,

librerías, proyectos y ejemplos que permiten a cualquier persona con pocos conocimientos de electrónica y programación adentrarse en este mundo (*Ibíd.*).

La placa Arduino contiene el microcontrolador, el encargado del control de todo el sistema. Es un circuito integrado que nos ofrece las posibilidades de un pequeño computador, en cuyo interior, encontramos un procesador, la memoria (RAM, ROM y EPROM), líneas de entrada/salida para comunicarse con el exterior, y varios periféricos (temporizadores, puertos serie y paralelo, CAD: Conversores Analógico/Digital, CDA: Conversores Digital/Analógico, etc.). La placa electrónica dispone de todo lo necesario para proporcionar soporte al microcontrolador como el reloj o la fuente de alimentación.

Arduino dispone de diferentes modelos de placas, con diferentes características. Stack_Duino es compatible con cualquier placa que monte un chip ATmega32u4; éste permite la comunicación USB con nuestro ordenador, facilitando que **Arduino** tome el control, por ejemplo, de nuestro teclado. Esta función será necesaria para alguna de las funciones. En concreto, se ha elegido el modelo Arduino Leonardo, por su facilidad de adquisición, funciones disponibles y bajo coste. Cuenta con 20 pines de entradas/salidas digitales, 12 entradas analógicas, un oscilador de cristal de 16 MHz, una conexión micro USB, un conector de alimentación, puertos ICSP e I2C, y un botón de reset.

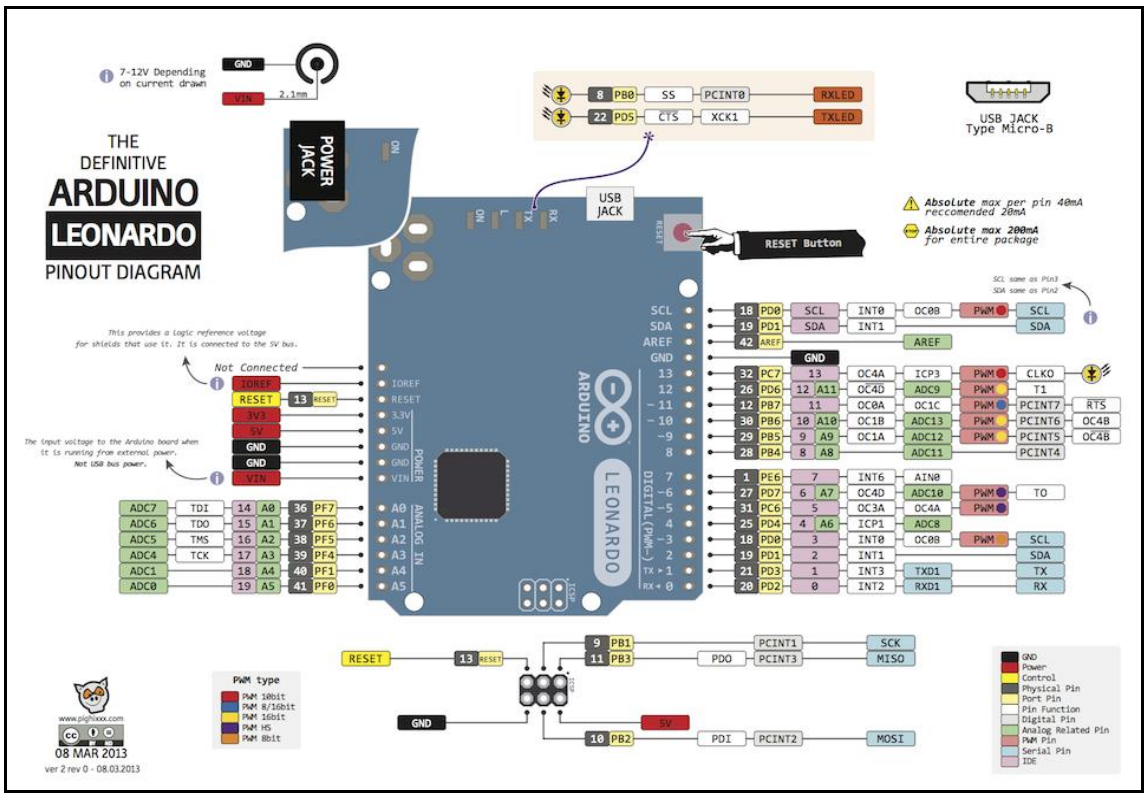

Figura 7: *Diagrama pines de Arduino Leonardo. (CC BY-SA 3.0) www.pighixxx.com.*

3.1.2. Interfaz de Usuario

La comunicación del usuario con Stack_Duino se realiza por dos medios. Mediante un teclado matricial de 12 teclas, el cual cuenta con filas y columnas, de forma que cuando se presiona un botón, se activa la salida correspondiente a la fila y la columna en la que se encuentra dicho botón, leyendo estos datos, Arduino puede saber que tecla ha sido presionada y en función de ella, realizar la tarea para la que ha sido programado. Permite la navegación dentro de los menús y la introducción de los valores y parámetros para los diferentes datos necesarios para su configuración o funcionamiento.

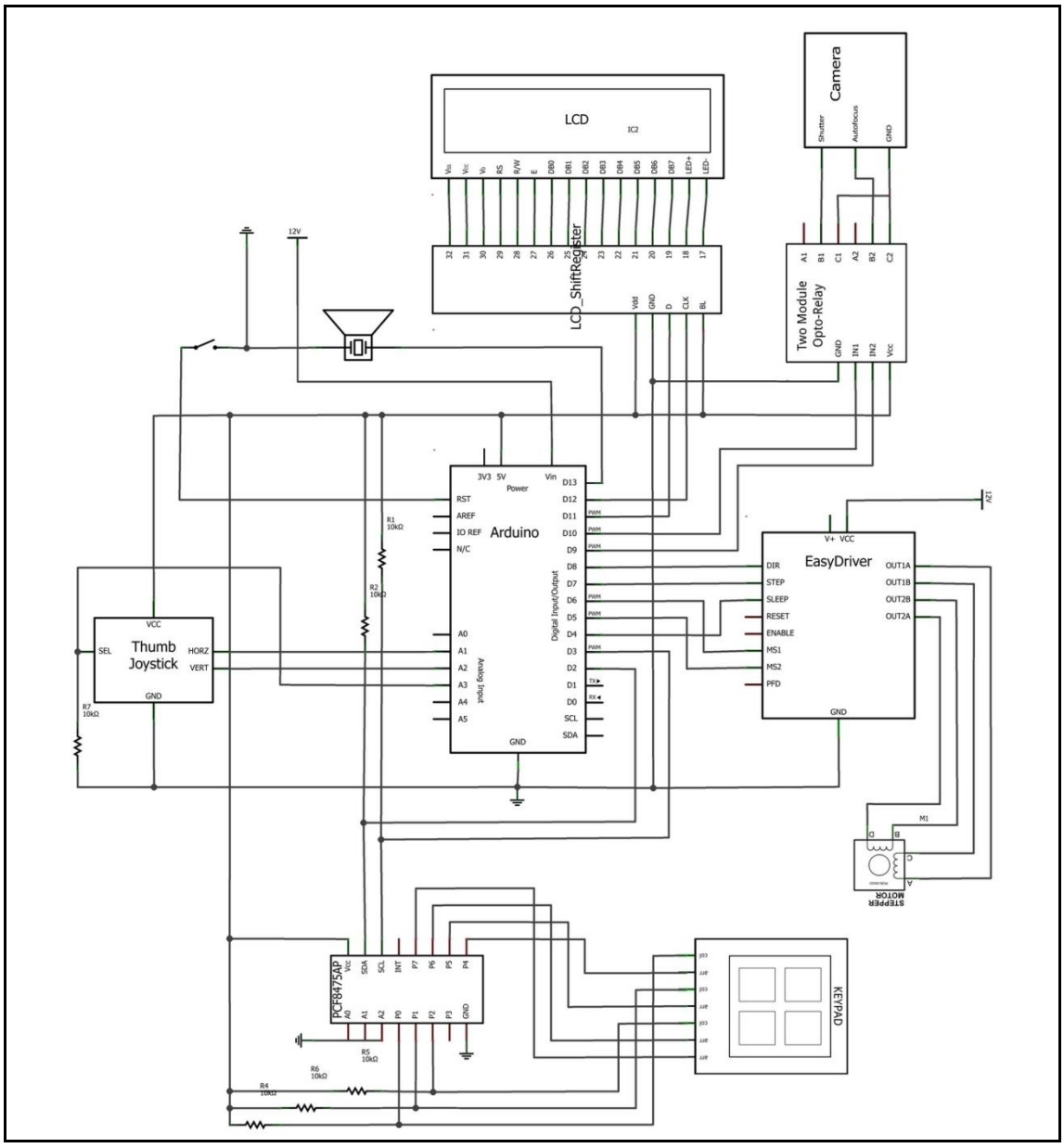

Figura 8: *Diagrama conexiones y componentes de Stack_Duino.*

La comunicación entre el teclado y Arduino se realiza a través del protocolo I2C, siendo éste un bus de comunicaciones en serie; únicamente se necesitan dos líneas, la de datos (SDA) y la del reloj (SCL). Cada dispositivo conectado al bus tiene un código de dirección seleccionable, lo que permite conectar varios dispositivos en serie a Arduino, únicamente mediante las dos entradas SDA y SCL y especificando la dirección de cada uno. En Stack_Duino la conexión al bus se realiza a través del chip PCF8574, que es un expansor de E/S compatible con la mayoría de microcontroladores y permite una comunicación bidireccional.

Mediante el joystick es posible mover el motor en la dirección adecuada, para poder posicionar el sistema en el lugar deseado. El joystick se comunica con Arduino a través de las entradas analógicas; en función del voltaje de entrada podremos controlar la velocidad con la que se mueve el motor, fijándose la velocidad mínima a través del menú de configuración. El joystick también actúa como un botón, que al presionarlo manda al sistema que tome una fotografía.

La salida de datos se realiza a través de un LCD de 2 líneas y 16 dígitos basado en el controlador HD44780 de Hitachi ya integrado, ampliamente usado, económico y fácil de obtener; éste se encarga de gestionar el display líquido: polarizar los puntos de la pantalla, generar los caracteres, desplazar la pantalla, mostrar el cursor, etc. Se trata de un display alfanumérico, que permite que se puedan mostrar mensajes formados por distintos caracteres: letras, números, símbolos, etc. Para la conexión se usa como puente el Shift_Registred entre el LCD y el teclado, esto permite reducir el número de pines de 8 del bus paralelo a tan solo 2, liberando pines para otros usos y aumentando la velocidad de respuesta enormemente.

3.1.3. Motor y Driver

El motor es el encargado de comunicar el movimiento al sistema óptico o al microscopio. Se ha optado por el uso de un motor paso a paso de 1,8º o 200 pasos por revolución, bipolar de 0,4A por fase. El motor paso a paso es un dispositivo electromecánico que convierte una serie de impulsos eléctricos en desplazamientos angulares discretos, lo que significa que es capaz de avanzar una serie de grados (paso) dependiendo de sus entradas de control. El motor paso a paso se comporta de la misma manera que un conversor digital-analógico (D/A) y puede ser gobernado por impulsos procedentes de sistemas lógicos, como Easydriver.

Easydriver es un controlador o driver de motores paso a paso (pap), es el encargado de transformar las señales de Arduino en pulsos de corriente, que harán girar el motor. Es compatible con cualquier dispositivo capaz de suministrar un pulso digital de 0V o 5V. Easydriver requiere una fuente de alimentación de 7V a 20V y proporciona al motor la tensión necesaria para su movimiento. Tiene integrado en la placa directamente un regulador de voltaje para el control de la parte lógica que puede ser ajustada a 5V o 3.3V. Conectándolo a un motor pap de 4 hilos y a un microcontrolador se obtiene un controlador de motor muy preciso. Easydriver controla motores bipolares pap de 4, 6 u 8 hilos (Schulmalz, 2014).

Dispone de microsteps (pines MS1 Y MS2) que permite más flexibilidad y control sobre el motor aumentando la resolución de éste. Los microsteps o micropasos, en español, permiten aumentar el número de pasos del motor. Los valores disponibles para el integrado A3967 que monta Easydriver, son: 1, 2, 4, 8. El motor permite un número de pasos fijo, el uso de los micropasos permite aumentar las posiciones que puede adoptar el motor por cada vuelta, y por tanto, aumentar el número de pasos por vuelta y la resolución del equipo. Si elegimos el valor de 2, se duplican los pasos; para valor de 4, se cuadruplica y así sucesivamente. Una ventaja de su uso, es una menor vibración del motor, pero para valores mayores hay una pérdida de reproducibilidad de los pasos.

3.1.4. Cámara

Stack_Duino permite el uso de dos tipos de cámaras. Cámaras controladas a través de software como las cámaras microscopía, siempre y cuando el software que maneja la cámara desde el PC, permita el disparo mediante una pulsación del teclado; o cámaras accionadas mecánicamente como las DSLR.

Para el primer tipo, Stack_Duino emula una pulsación del teclado, enviándola al software que controla la cámara, que es el encargado del disparo y toma de la imagen. El disparo de cámaras DSLR se hace de forma mecánica, conectando ésta a Stack_Duino mediante el cable adecuado según la marca, al jack de 3,5mm de la caja de control. La conexión entre Arduino y la cámara se realiza a través del relé con opto-acoplador, esto permite aislar totalmente la cámara del sistema y protegerla de posibles

cortocircuitos. Stack_Duino está pensado principalmente para trabajar con este tipo de cámaras ya que dan una mejor calidad de imagen a un costo mucho menor.

3.2. Firmware

Para programar la tarjeta se necesita el ambiente de desarrollo Arduino. El entorno de desarrollo Arduino, IDE, es libre y de código abierto, disponible para Windows, Mac OS, y Linux. Está constituido por un editor de texto para escribir el código, un área de mensajes, una consola de texto, una barra de herramientas con botones para las funciones comunes, y una serie de menús. Permite la conexión al hardware de Arduino para cargar los programas y comunicarse con ellos.

La descarga del IDE se puede realizar desde la página web del proyecto Arduino (http://arduino.cc/), el sketch de Stack_Duino, es compatible con la versión 1.05. Una vez realizada la descarga, descomprimimos la carpeta en el lugar de destino. El Stack_Duino utiliza una librería diferente a la que trae por defecto para el control del LCD, para su descarga (ver Anexo II), seleccionaremos la última versión de LiquidCrystal, y la copiamos a la carpeta "*libraries*" del IDE de Arduino, sobrescribiendo la que allí se encuentre. Podemos encontrar más instrucciones de funcionamiento e información del IDE en el mismo sitio web de Arduino. Junto con los archivos suministrados en este documento podemos encontrar la versión del IDE 1.05 en la que ya ha sido reemplazada la librería, y por lo tanto el paso anterior no será necesario. Para instalar la tarjeta Arduino y poder cargar el sketch debemos instalar los drivers de éste, estos se encuentran en el directorio "*drivers***"** de la distribución del IDE. Dependiendo del sistema operativo que tengamos los pasos serán diferentes.

Arduino utiliza para escribir el firmware lo que denomina *"sketch"* (programa). Estos programas son escritos en el editor de texto. Existe la posibilidad de cortar/pegar y buscar/remplazar texto. En el área de mensajes se muestra información mientras se cargan los programas y también muestra errores. La consola muestra el texto de salida para el entorno de Arduino incluyendo los mensajes de error completos y otras informaciones.

La estructura básica de un sketch es muy similar a la de cualquier programa en lenguaje "C" con algunas diferencias impuestas por el hecho de que la plataforma interactúa directamente con el medio físico que le rodea. Así la primera parte de un Sketch será el llamamiento a las librerías, la declaración de variables donde definiremos las variables globales del Sketch, declarando su tipo y si queremos, su valor inicial. Estas variables serán accesibles desde cualquier parte de nuestro Sketch al contrario que las variables locales, que son las que se declaran dentro de una función o dentro de una estructura de control.

Seguidamente tenemos las funciones. Una función es un bloque de código que tiene un nombre y un conjunto de instrucciones que son ejecutadas cuando se llama a la función. La segunda parte de un Sketch, es una parte que solo se ejecuta una vez tras el arranque de la placa Arduino, es la función "void setup()". Suele utilizarse para definir los pines que vamos a utilizar y declararlos como entradas o como salida, esta parte del código solo se ejecuta una vez cuando Arduino arranca.

La siguiente parte importante de un Sketch, la función "void loop()", es la parte principal de nuestro programa, en ella tendremos que poner el código para que el microcontrolador lo ejecute de manera repetida, como un bucle, es decir, esta parte del Sketch se repetirá una y otra vez mientras la placa Arduino esté en marcha.

No todo el código tiene que estar escrito en el Loop, ciertas partes pueden ser escritas fuera de este bucle como funciones, y serán llamadas dentro del Loop para su ejecución. En el Anexo IV tenemos el código necesario para el funcionamiento de Stack_Duino, donde podemos encontrar un pequeño comentario antes de cada función para saber su utilidad, y en caso necesario, poder modificarlas a nuestra conveniencia, para poder adaptar Stack_Duino a las necesidades del operador.

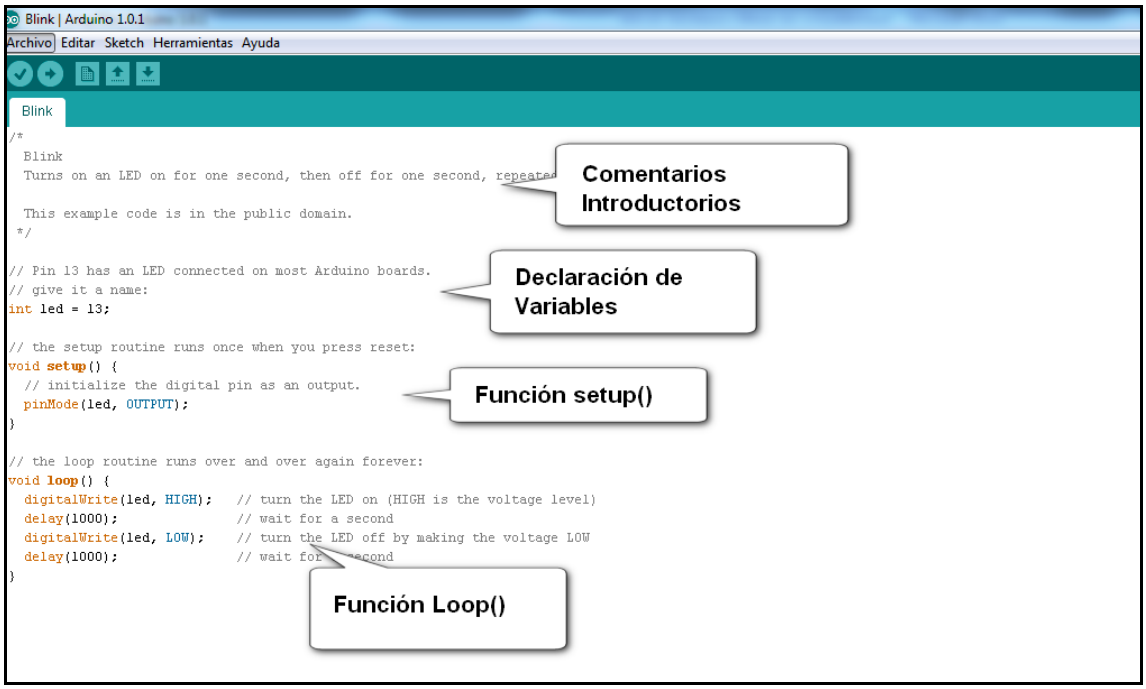

Figura 9: *Estructura del sketch.*

3.3. Hardware, ejemplos de montaje

La forma en que Stack_Duino se une a al sistema óptico que usaremos dependerá en cada caso del diseño de éste. No será lo mismo el montaje para un microscopio que para un sistema que se desliza sobre un rail de enfoque o de la forma en la que el motor transmita el movimiento, bien mediante poleas y una correa o directamente por fricción, como se usa en astronomía. Seguidamente, como ejemplo, mostraremos el montaje de dos sistemas diferentes, un macroscopio casero y un microscopio poligráfico.

Un macroscopio viene a ser un microscopio para trabajar a pocos aumentos. Posee un único camino óptico que es paralelo al eje de desplazamiento del sistema óptico (figura 10).

La transmisión del movimiento del motor al piñón de enfoque, se realiza mediante una correa dentada y dos ruedas dentadas de 20 dientes estándar imperial XL. Siendo el recorrido del piñón de enfoque de 200 micras/vuelta, con un motor de 200 pasos/vuelta y la relación entre las dos ruedas dentadas de 1:1, el conjunto permite una resolución mínima de 0,5 micras. Jugando con el tamaño de las poleas podemos aumentar la resolución, con una relación de 2:1 pudiendo llegar a 0,25 micras de resolución.

En el siguiente ejemplo (figura 13), Stack_Duino se ha montado sobre un microscopio petrográfico. La transmisión es similar al modelo anterior, si bien, un motor de 200 pasos/vuelta, poleas de 20 dientes estándar imperial xl y correa dentada, la relación entre las dos ruedas dentadas de 1:1. Si bien ahora lo que cambia es el desplazamiento del piñón de enfoque, de 100 micras por vuelta, llegando entonces a una resolución de

0,25 micras ó 0,12 con una relación de poleas 2:1, más que suficiente para la mayoría de las aplicaciones.

Figura 10: *Macroscopio casero. Un fuelle de fotografía se encuentra unido a un piñón de enfoque con movimiento macro y micrométrico, en la parte inferior se encuentran los objetivos mientras en la superior se coloca la cámara, en este caso una DSLR Canon Eos 40D. (Fot. H. Cócera).*

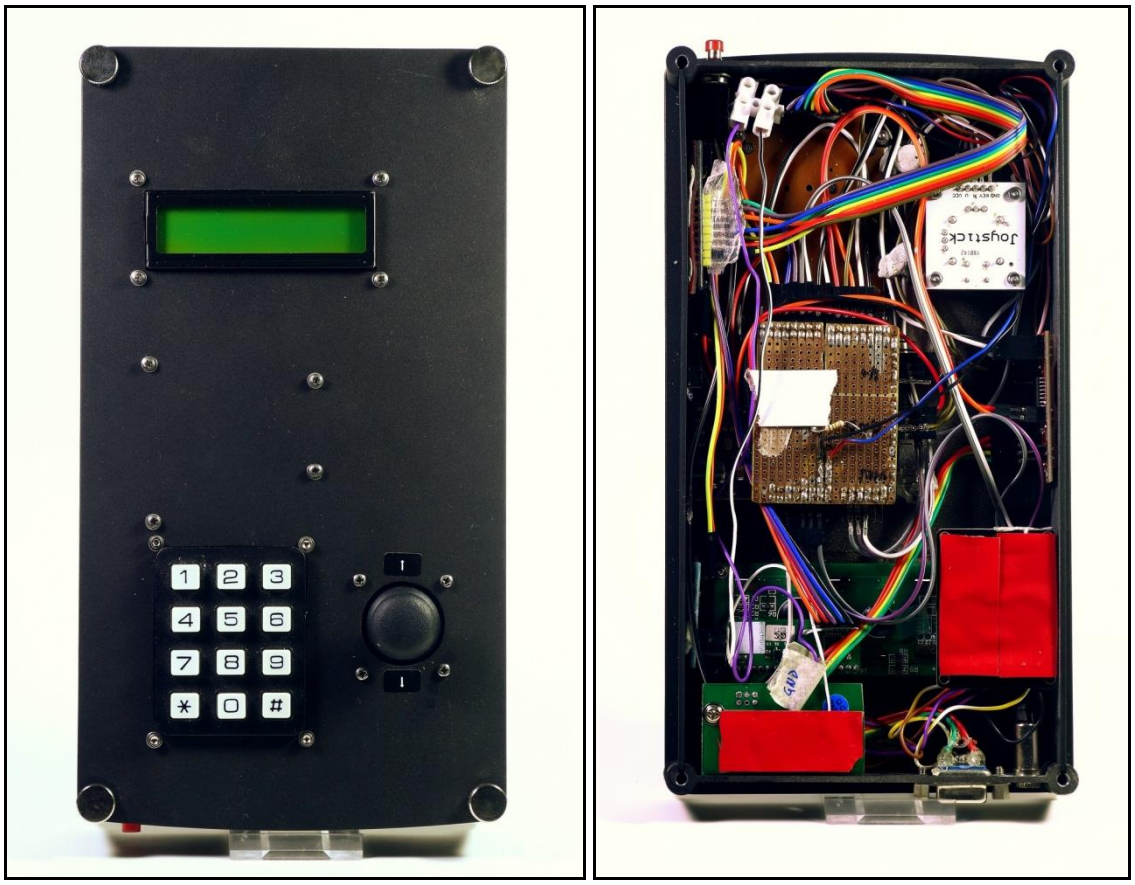

Figura 11: *Caja de control de Stack_Duino. Imagen izquierda: parte frontal. Imagen derecha: vista de conexiones y montaje. (Fot. H. Cócera).*

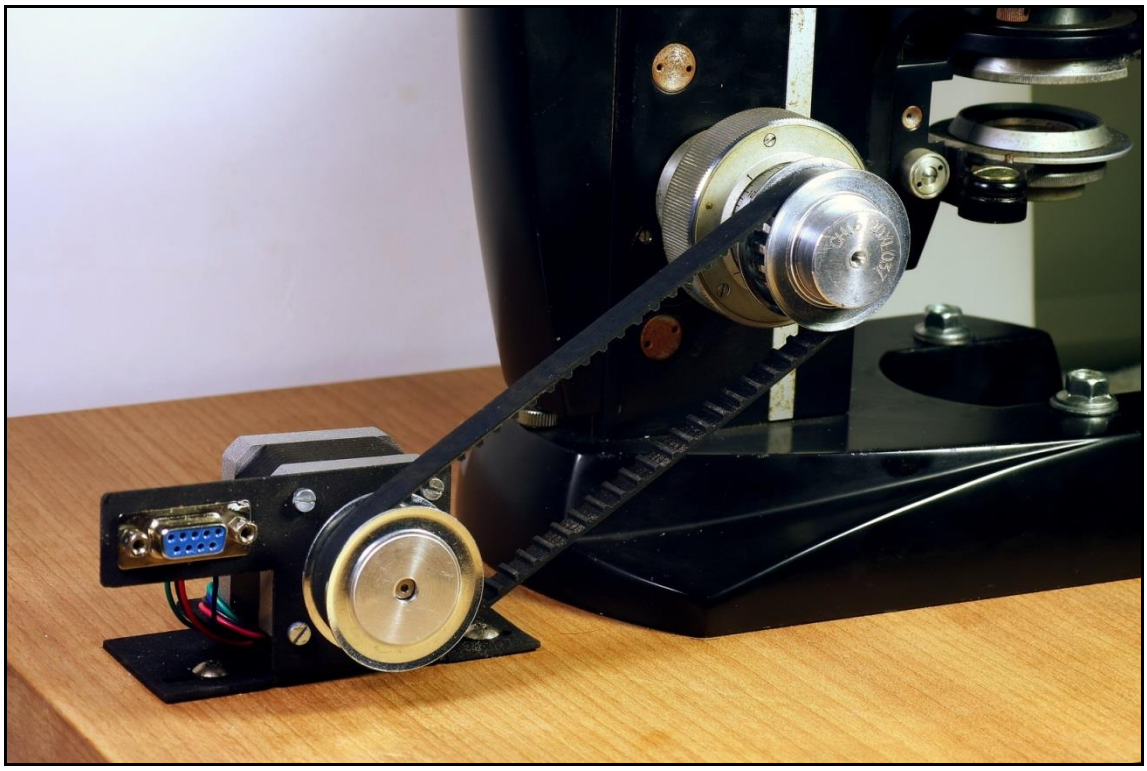

Figura 12: *Detalle transmisión del movimiento del motor a un microscopio petrográfico. Poleas dentadas y*

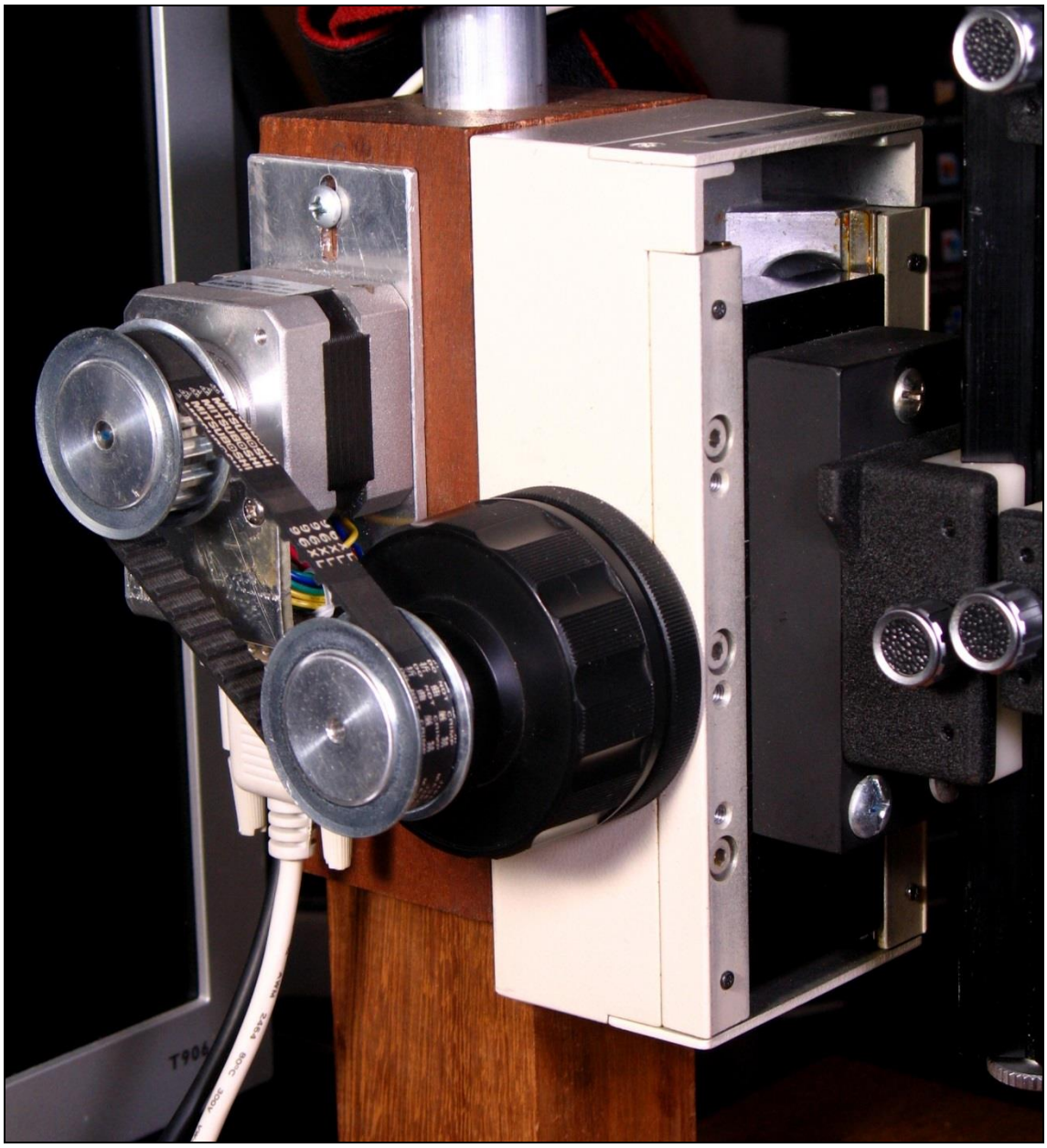

Figura 13: *Detalle transmisión del movimiento del motor. Poleas dentadas y correa estándar imperial xl. (Fot. H. Cócera).*

4. RESULTADOS Y DISCUSIÓN

El registro y documentación de colecciones es un proceso esencial, la recopilación permanente de información textual y visual es un requisito para el manejo de los fondos y colecciones científicas. La documentación exhaustiva de las piezas que conforman una colección y el proceso de digitalización de la información obliga entre otras, al registro fotográfico de los objetos que la conforman, convirtiéndose la fotografía en una herramienta indispensable para su funcionamiento como fuente de información.

La gestión y documentación de las colecciones científicas (geológicas, paleontológicas, mineralógicas, etc…), se ha realizado tradicionalmente mediante el uso de archivos de fichas en papel o cartulina, que incluían toda la información disponible del ejemplar en cuestión. Este tipo de soporte tiene bastantes inconvenientes como son la utilización de un espacio físico muy grande sobre todo en museos con colecciones extensas, la

incomodidad a la hora de la modificación de los datos y sobre todo en los procesos de búsqueda de ejemplares o grupos de ejemplares asociados por diferentes criterios. Actualmente los recursos informáticos solventan gran parte de estos inconvenientes y nos permiten almacenar en soportes físicos (y virtuales), grandes bases de datos en un espacio muy reducido. Las ventajas de la digitalización y la publicación en línea de los datos de las colecciones, permite un incremento del valor (en todos los aspectos) de los ejemplares de una colección científica. Una adquisición de imágenes de alta calidad es un documento de muy gran valor científico:

- Permite conocer el ejemplar desde su adquisición, e incluso en diversas fases: restauración, limpieza, posibles alteraciones, evolución…
- Permite una identificación del ejemplar y una visualización evitando, en la medida de lo posible, la manipulación del ejemplar original, por lo que se pueden evitar pérdidas, alteraciones, etc.
- Su publicación en una base de datos on-line, pone a disposición de los investigadores, aficionados, visitantes, el acceso a ejemplares cuya consulta, en el caso de no disponer de esta técnica, solo sería posible con la visita del investigador al centro, cuestión que no siempre es posible.

La correcta toma de imágenes requiere tanto de personal con los conocimientos técnicos necesarios como del equipamiento adecuado. Stack_Duino permite integrar los últimos avances en adquisición y tratamiento digital de imágenes. En la Figura 14 tenemos un ejemplo de uso de estas imágenes en una ficha de registro.

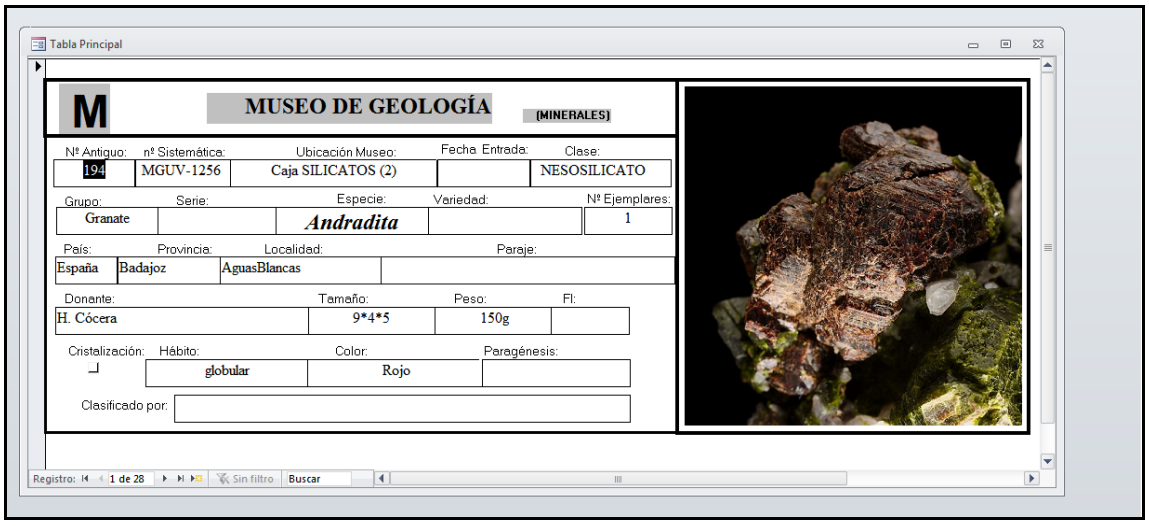

Figura 14: *Ejemplo de registro.*

4.1. Funcionamiento y modos de operación

Automatizar el proceso de disparo y adquisición de la serie minimiza la posibilidad de errores y fluctuaciones internas, ahorra tiempo y prácticamente es indispensable para la toma de series compuestas por un gran número de imágenes.

Una vez configurado Stack_Duino con los parámetros del sistema, marcados el punto de inicio, y el punto final o la distancia entre foto y foto a mover, podremos elegir entre dos modos de operación o de toma de la serie de imágenes:

4.1.1. Modo A-B

En este método se marca el punto de inicio A y el punto final B de la pila, se introduce el número de micras a desplazar entre foto y foto; el programa calcula el número de fotos a realizar y muestra los datos en pantalla.

4.1.2. Modo A/nFotos

En este modo se marca el punto de inicio A y el número de fotos a hacer y se introduce el número de micras a desplazar entre foto y foto.

Una vez acabada la toma de la serie, nos mostrará el desplazamiento total y nos pregunta que queremos hacer, las opciones disponibles son:

- Mueve el motor y toma una foto adicional.
- Vuelve a realizar la pila con los mismos parámetros, esto nos permitiría hacer fotos panorámicas sobre un objeto, conservando los puntos de inicio y final de la serie, moveríamos un poco el objeto y volveríamos a repetir la serie con el nuevo encuadre.
- Realiza *n* fotos adicionales.
- Volver al menú principal reseteando los valores de posición y estado de A y B, y la posición del motor Z.
- Volver al menú principal pero sin resetear los valores anteriores, esto nos permitiría repetir la serie exactamente igual.

Stack_Duino no solamente es capaz de controlar movimientos en sistemas lineales, sino también rotatorios, para ser usado en otras técnicas como en el control de mesas rotatorias para la fotografía de 360º. Tan solo hay que cambiar las unidades de un par de parámetros, en grados, definiendo el valor de “Distancia_vuelta” como 360º, y valor de “Distancia_x_foto” n grados. De esta forma, al girar el motor, el motivo girará n grados respecto al eje óptico; a diferencia del movimiento lineal, en el que el motivo se alejará o acercará a la cámara. Para más información ver Anexo II, funcionamiento.

4.2. Apilado de imágenes

Una vez realizada la toma de la serie de imágenes es necesaria su fusión. Existen numerosos paquetes de software en el mercado, con diferentes características y precios. A continuación haremos una breve descripción de los que por nuestra experiencia ofrecen los mejores resultados.

Combine ZP. (Hadley, 2014) Es un programa gratuito de código abierto creado por Alan Hadley. Aunque es de uso complicado, dispone de muchos métodos de apilado, y la posibilidad de modificarlos o crear los nuestros propios a través de macros. Permite el retoque de las imágenes finales, aunque no es sencillo, pero sí bastante efectivo. Solo está disponible para Windows y no soporta imágenes de 16 bits ni raw.

Helicon Focus, (Helicon Soft., 2014), es un programa de pago creado y publicado por Helicon Soft Limited, cuya primera versión aparece en 2003. Posee tres métodos de apilado diferentes, destaca por su fácil manejo, rapidez, diferentes métodos de interpolación, posibilidad de retoque, creación de archivos 3D y otras opciones. Permite trabajar en cualquier formato, incluso raw. Muy buenos resultados con imágenes con poco contraste. Disponible para cualquier plataforma. Como curiosidad, este programa nace de la necesidad de un químico de fotografiar pequeños cristales obtenidos en el laboratorio, su cuñado aficionado a la fotografía lo creó para resolver su

problema, posteriormente fue puesto en el mercado a la disposición de cualquier usuario.

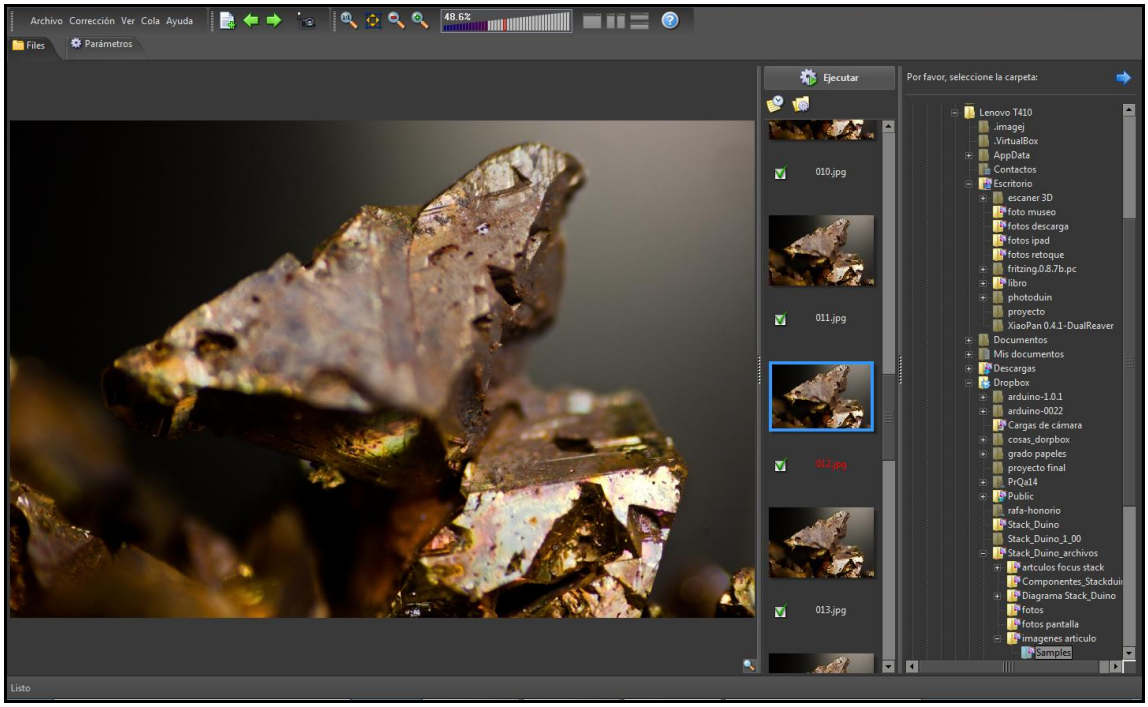

Figura 15: *Interfaz de usuario de Helicon Focus.*

Figura 16: *Cristal romboédrico de carbonato cálcico obtenido sintéticamente mediante cristalización en gel en el laboratorio. (Fot. César Menor-Salvan).*

Zerene Stacker, (Littlefield, 2014). Es uno de los últimos programas en aparecer, creado por Rick_Littlefield. Al igual que Helicon focus es de fácil uso. Presenta dos métodos de apilado, Pmax y Dmax, el primero está pensado para imágenes que muestran detalles muy finos como pelos o agujas; el otro está pensado para motivos más voluminosos, consigue una mejor reproducción del color y del contraste. También permite retoque de las imágenes, si bien trabaja de forma algo diferente que le aporta ciertas ventajas, al poder combinar varias imágenes finales; por ejemplo, combinar la salida de Dmax con la de Pmax. Permite trabajar con imágenes de 16 bits y está disponible para cualquier plataforma.

Las imágenes resultantes del apilado pueden ser post-procesadas, en ocasiones el proceso de apilado puede producir artefactos no deseados, como halos, presencia de manchas del sensor, reflejos no deseados, etc... Estos artefactos, y el ajuste final de contraste, luminosidad, balance de blancos pueden ser fácilmente editados con un programa de edición de imagen. Como ilustración de los resultados obtenidos, la Figura 6 muestra una imagen final, correspondiente a un cristal de cobre nativo de 4 mm de arista, otro ejemplo lo encontramos en la Figura 16, correspondiente a un cristal romboédrico de carbonato cálcico obtenido sintéticamente mediante cristalización en gel en el laboratorio. Esta técnica tiene una gran aplicación dentro del campo del estudio de ciencias de los materiales, la Figura 17 aplicada al estudio de procesos de cristalización, en la que al variar las condiciones experimentales se obtienen diferentes morfologías de los cristales.

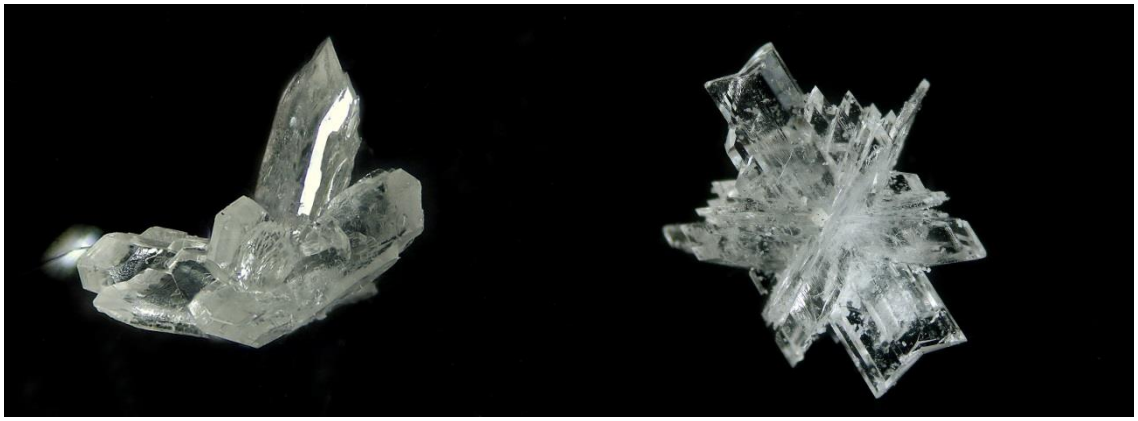

Figura 17: *Cristales microscópicos de sulfato cálcico obtenidos sintéticamente en el laboratorio mediante cristalización en gel. La imagen de la izquierda muestra un clúster de cristales monoclínicos, al bajar la concentración de los reactivos se obtienen morfologías diferentes, en este caso cristales en macla de golondrina, imagen derecha. Multifoco. (Fot. César Menor-Salvan).*

4.3 Imágenes 3D

Otra aplicación directa es la generación de archivos 3D de los objetos en estudio. En el mercado existe multitud de escáneres capaces de generar archivos 3D de objetos de tamaño grande o medio. Sin embargo, cuando nos movemos a tamaños de orden de los pocos milímetros, los precios de los pocos equipos que son capaces de realizarlo, se disparan. Con la información que suministran los programas de apilado, tras procesar la pila de imágenes obtenidas mediante Stack_Duino, es posible reconstruir un archivo 3D del objeto.

Al combinar la parte enfocada de cada una de las imágenes de la serie obtenemos, por una parte, una imagen formada por pixeles de color, y otra imagen llamada “mapa de

profundidad" (Figura 18), que contiene las coordenadas x-y-z de cada uno de esos pixeles, donde la escala de grises representa la coordenada z.

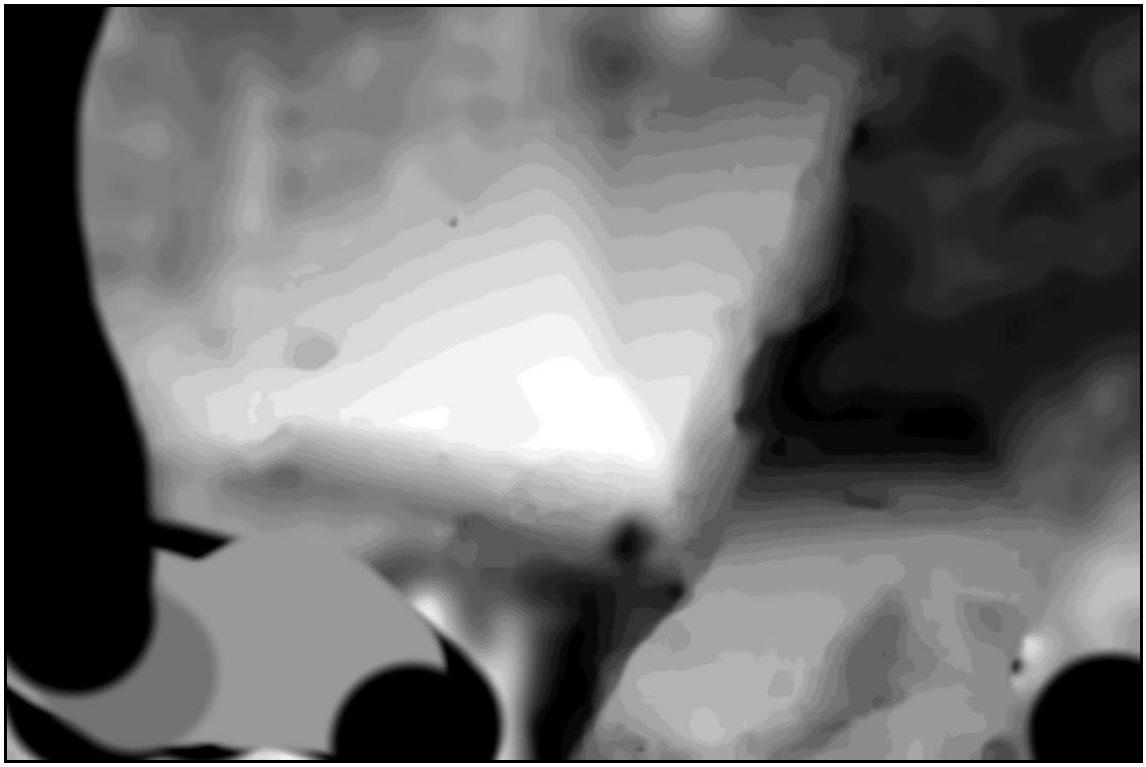

Figura 18: *Mapa de profundidad para el cristal de cobre de la figura 6.*

Una vez obtenidas estas dos imágenes, es posible generar el archivo 3D. Helicon Focus, dispone de una opción que genera directamente el archivo. En el mercado hay disponibles otros programas específicos que realizan esta acción, dando archivos de mayor calidad, y posteriormente, permitiendo hacer medida de superficies, distancias y ángulos sobre el archivo, siendo de aplicación directa por ejemplo en estudios de cristalografía. Y por supuesto, podemos imprimirla en una impresora 3D a una escala mucho mayor que nos permita ver los detalles con mayor facilidad.

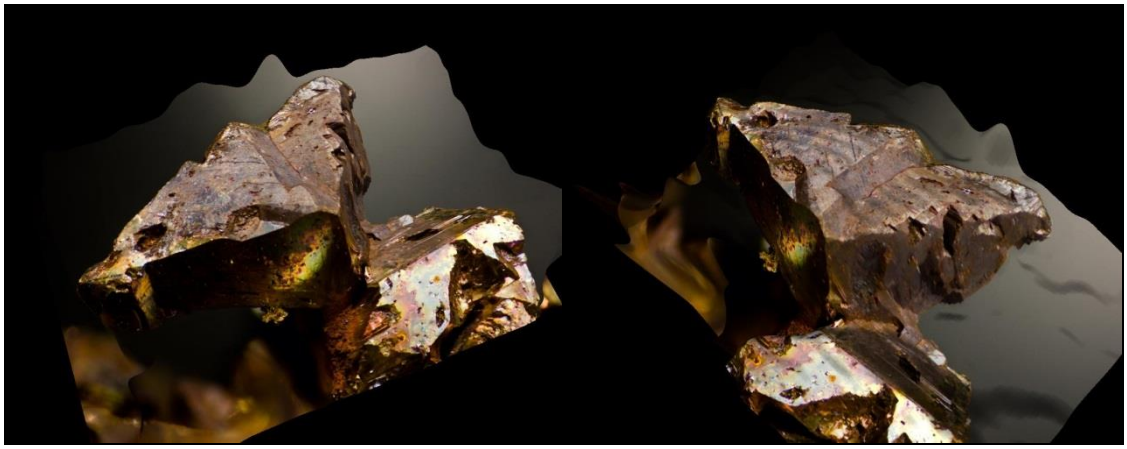

Figura 19: *Imágenes 2D generadas sobre el archivo 3D del cristal de cobre de la figura 6. (Col. y Fot. H. Cócera).*

4.4. Comparación con otras técnicas

Hasta la aparición de esta la fotografía multifoco, el método más empleado para la obtención de imágenes en detalle de objetos microscópicos era el uso del microscopio electrónico de barrido (Scanning electrón microscope o SEM), éste es capaz de obtener imágenes a alta magnificación con mayor EDF, pero en una escala de grises, sin color. Las muestras deben ser cuidadosamente preparadas para que su superficie pueda conducir la electricidad, para ello se recubren bien con una fina capa de un metal o de carbono. El haz de electrones incide sobre la superficie de la muestra, rebotando sobre ella y llegando al detector que transforma en señal que crea una imagen en el computador. Este es un método caro tanto en el coste del equipamiento como en el tiempo que consume. El uso de Stack_Duino junto al multifoco permite obtener las mismas imágenes, hasta el rango de los 50x, a un costo ostensiblemente menor. Además, ofrece mayor información, ya que es capaz de obtener imágenes en color, imprescindibles en campos como la mineralogía, la figura 21 muestra una comparación entre una imagen obtenida mediante SEM y otra usando multifoco. También presenta ventajas en campos donde el motivo de estudio se encuentre incluido en el interior de un objeto, como es el caso del estudio de inclusiones en gemología, figura 20, por poner un ejemplo.

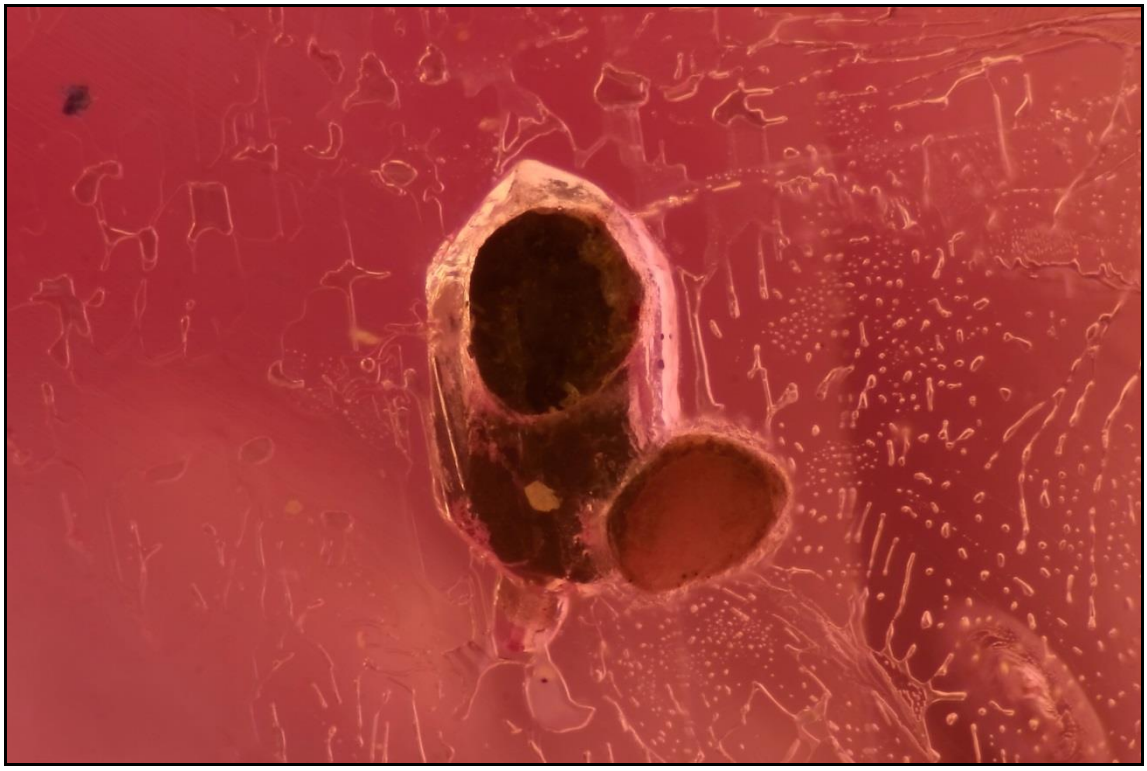

Figura 20: *Imagen de una inclusión en cristal de rubí obtenida mediante Stack_Duino + multifoco. (Col. M.A. Maroto, Fot. H. Cócera).*

También presenta ventajas frente al uso de microscopios confocales laser de barrido. En este equipo, se combina el microscopio de fluorescencia con imagen electrónica y puntos de luz, permitiendo obtener imágenes en tres dimensiones de secciones sobre el objeto deseado. Al igual que para SEM, es una técnica cara, donde Stack_Duino junto con la técnica del multifoco puede obtener resultados comparables a un coste mucho más reducido.

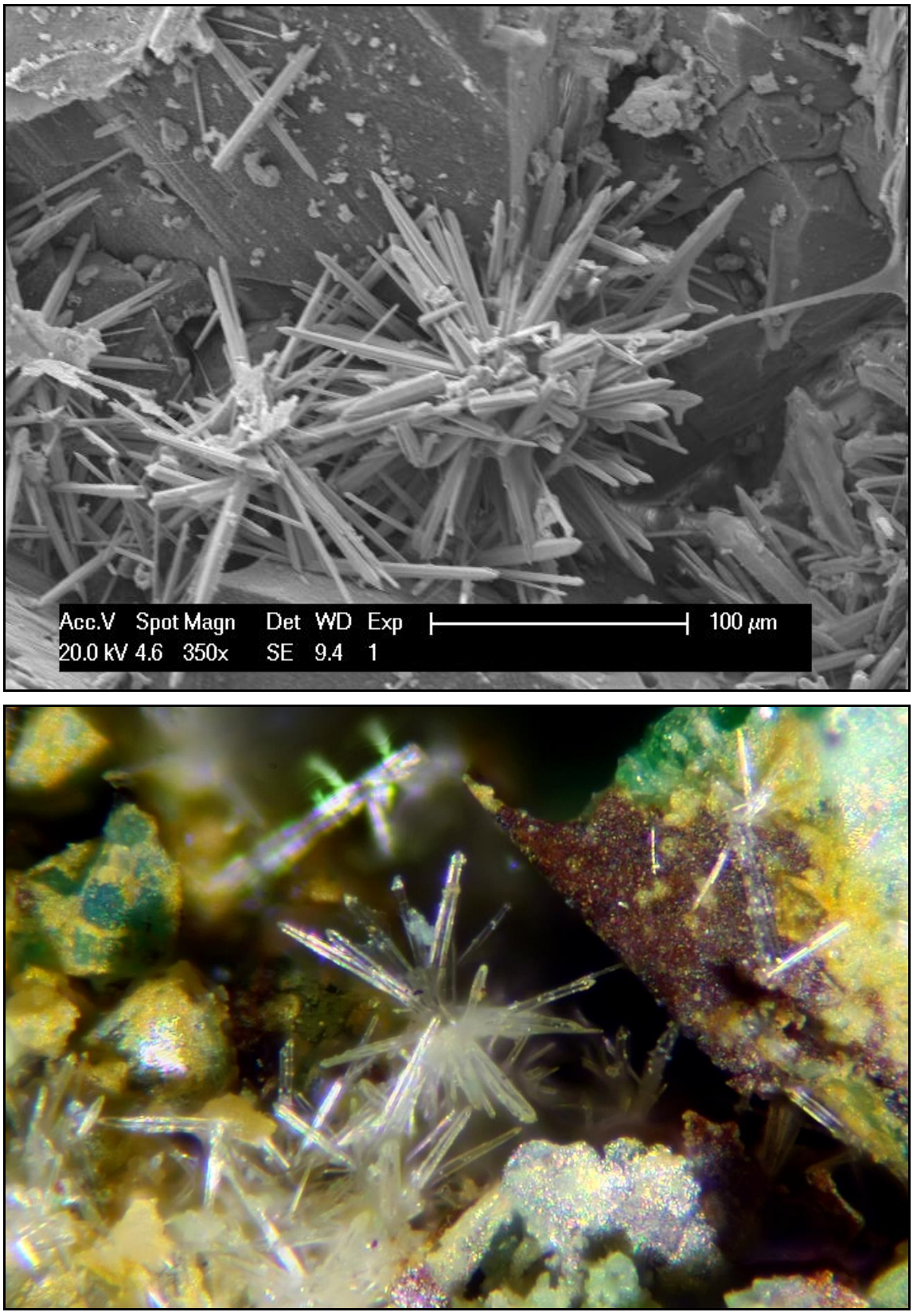

Figura 21: *Fotografía de cristales de mimetita de la mina Amorosa, Villahermosa del Río Castellón. La imagen superior está obtenida mediante SEM, la imagen inferior esta obtenida mediante multifoco, mostrando características no posibles de registrar mediante SEM. (Col. y Fot. H. Cócera).*

4.5. Posibles aplicaciones, cambios y adaptación a otros proyectos

Stack_Duino ha sido diseñado pensando en la mayoría de las necesidades que el usuario final pueda necesitar. Además, y al ser de código abierto, puede ser fácilmente modificado para incluir nuevas opciones, tanto en la parte de hardware como en el firmware.

Stack_Duino ha sido diseñado pensando en poder ser adaptado fácilmente a otros proyectos. Podemos dividirlo en dos partes, una que es la parte necesaria para la fotografía multifoco en sí, y otra que es el corazón que lo hace funcionar, la placa Arduino, el teclado y el LCD más el código necesario para que se integren: los menús, lectura teclado, salida en pantalla, etc... Utilizando esta segunda parte y eliminando la primera, podemos adaptar la caja de control a cualquier otro proyecto, como por ejemplo un control de un baño térmico. Adaptando un termopar podemos controlar la temperatura, programar la caja de control para que active una resistencia o la apague en función de la temperatura que hayamos indicado mediante el menú +teclado, o incluso pueda abrir un grifo o cerrarlo para por ejemplo añadir líquido en caso de que este disminuyera. Otro ejemplo podría ser el seguimiento de una reacción por lectura de la absorbancia, mediante un sensor LDR que mide la intensidad de la luz incidente y un láser de la longitud de onda adecuada, podríamos registrar la variación de la concentración de un producto y guardar la información en un archivo; o esto mismo en industria, para el control y monitorización de la producción de una sustancia, actuando sobre la maquinaría de forma automática si fuera necesario.

En el mercado existe una gran multitud de periféricos que podrían adaptarse a nuestras necesidades y una enorme comunidad, donde encontrar información y siempre dispuesta a ayudar. Aquí es donde radica el gran éxito que ha tenido Arduino.

5. CONCLUSIONES

Stack_Duino es un sistema automatizado y autónomo para la toma de pilas de imágenes, dispone de dos modos de toma de imágenes; en el primero se marcan el punto de inicio A, el punto final B y la distancia entre fotos, calculando el software el número de fotos a realizar; en el segundo, se marca el punto de inicio A, el intervalo entre fotos y el número de fotos a realizar. Dispone además de un menú de configuración que le permite ser adaptado fácilmente a cualquier sistema, como un rail de enfoque, un estereomicroscopio o un microscopio compuesto. El diseño presentado en este trabajo permite movimientos mínimos de hasta 0,5 micras, lo que permite trabajar hasta ampliaciones de 50x o más, aunque es posible llegar a mayores aumentos con pequeñas modificaciones, si bien, la máxima ampliación dependerá en gran medida de la calidad de nuestro sistema óptico. El campo de aplicación de este dispositivo puede abarcar diversas disciplinas como: mineralogía, biología, medicina, arqueología, etc… presentando ventajas frente a otras técnicas.

6. AGRADECIMIENTOS

Dedicado a la memoria de Joan Viñals, un maestro, un amigo, un ejemplo. A Anna García, Luis E. Ochando, Jorge Soria, Rafa Muñoz, César Menor, Jesús Alonso, Carlos Martínez, M. Angeles Maroto, J. Antonio Villena, Victoria Puchol y Vicky Martínez, por la ayuda y apoyo para la realización del presente trabajo..

7. BIBLIOGRAFÍA

ARDEHIR GOSHTABY, A. (2006) Fusion of multifocus images to maximize image information. *Proc. SPIE 6229, Intelligent Computing: Theory and Applications IV*, 62290L (May 09, 2006); doi:10.1117/12.663706

ARDUINO (2014) http://www.arduino.cc/

BERCOVICI, A., HADLEY, A., VILLANUEVA-AMADOZ U. (2009) Improving depth of field resolution for palynological photomicrography. *Paleontologia Electronica*, **12 (2).**
Http://paleoelectronica/2009_2/170/170.pdf

BEREJNOV, V. (2009) Rapid and inexpensive reconstruction of 3D structures for micro-objects using common optical microscopy. Xxx.lanl.gov.

BETZ, V. (2005) Micromineral Photography with Multifocus Processing. *Mineralogical Record*, **36**:365-369.

CISNEROS, E., (2010) Automated Multi-focus Imaging. *Axis*. 6 (2).
www.MineralogicalRecoord.com

DAVIDSON, M. (2009) Numerical Aperture and Resolution. *Molecular Expressions*.
http://micro.magnet.fsu.edu/primer/anatomy/numaperture.html

DAVISON, M., ABRAMOWITZ, M. (2014) *Optical Microscopy*. Olympus Microscopy Resource center.
http://www.medic.ula.ve/histologia/anexos/microscopweb/MONOWEB/anexos/Olympusmicroscopy15.pdf

HADLEY, A. (2014) Combine ZP. http://www.hadleyweb.pwp.blueyonder.co.uk/

HAEBERLI, P. (1994. A multifocus method for controlling depth of field.
http://graficaobscura.com/depth/index.html

HELICON SOFT. (2014) Helicon focus.
http://www.heliconsoft.com/heliconsoft-products/helicon-focus/

LITTLEFIELD, R. (2014) Zerene Stacker. http://zerenesystems.com/cms/stacker

LONGSON, J., COOPER, G., GIBSON, R., GIBSON, M., RAWLINS, J., SARGENT, R. (2010) Adapting traditional macro and micro photography for scientific gigapixel imaging. *Fine international conference on gigapixel imaging, conference and events.*
http://repository.cmu.edu/gigapixel/1/

MADHAVI, R., ASHOK, K. (2011) An all approach for Multi-focus image fusion using neural network. *International journal of computer science and telecommunication*, Volume 2, Issue 8. www.ijcst.org

MONJE, L. (2010) Fotografía científica. El arte de captar lo invisible.
http://www.uned.es/ca-guadalajara/actividades/09-10/JuevesCiencia10/FotografiaCientifica.pdf

NAVAS, J., KULAWIK, K., ALCANTARÁ, R., FERNÁNDEZ-LORENZO, J., MARTÍN-CALLEJA, J. (2010) Instrumental development for obtaining extended depth of field images: study of the influence of the chromatic coordinates used as basis of the calculation algorithm. *Microscopy: Science, Technology, Applications and Education, Formatex*, **4**:1474-1482.

NIEDEROST, M., NIEDEROST, J., SCUCKA, J. (2012) Automatic 3D reconstruction and visualization of microscopic objects from a monoscopic multifocus image

sequences. *International Archives of the Photogrammetry. Remote Sensing and Spatial Information Sciences,* Vol. XXXIV-5/W10.

PADLEY, P. (2005) Diffraction from a Circular Aperture. *Retrieved from the OpenStax-CNX*. http://cnx.org/content/m13097/1.1/

PHOTODUINO (2014) http://photoduino.com/es/index.html

SÁNCHEZ-ORTEGA, N. (2011) De la fotografía como representación de la realidad a documento representado: el análisis documental de contenido. *Contribuciones a las Ciencias Sociales*, octubre 2011. http://www.eumed.net/rev/cccss/14/

SAVAZZI, E. (2011) *Digital Photography for Science: Close-up Photography, Macrophotography and Photomicrography*. Lulu Press, Railigh NC, USA (704 pp).

SAVAZZI. E. (2011) Construction and programing of an autonomous focus stacker. *Computers and Geosciences*. **37**:1670-1676.

SCHULMALZ, B. (2014) EasyDriver. http://www.schmalzhaus.com/EasyDriver/

WIKIPEDIA. (2014) Confocal scanning laser microscope.
http://es.wikipedia.org/wiki/Microscopio_confocal_l%C3%A1ser_de_barrido

WIKIPEDIA. (2014) Focus Stacking Software.
http://en.wikipedia.org/wiki/Focus_stacking

ZHAO, H., SHANG, Z., YANG-TANG,Y., FANG, B. 2013. Multi-focus image fusion based on the neighbor distance. *Pattern Recognition*. **46**: 1002-1011.

Anexo I

Imágenes de muestra

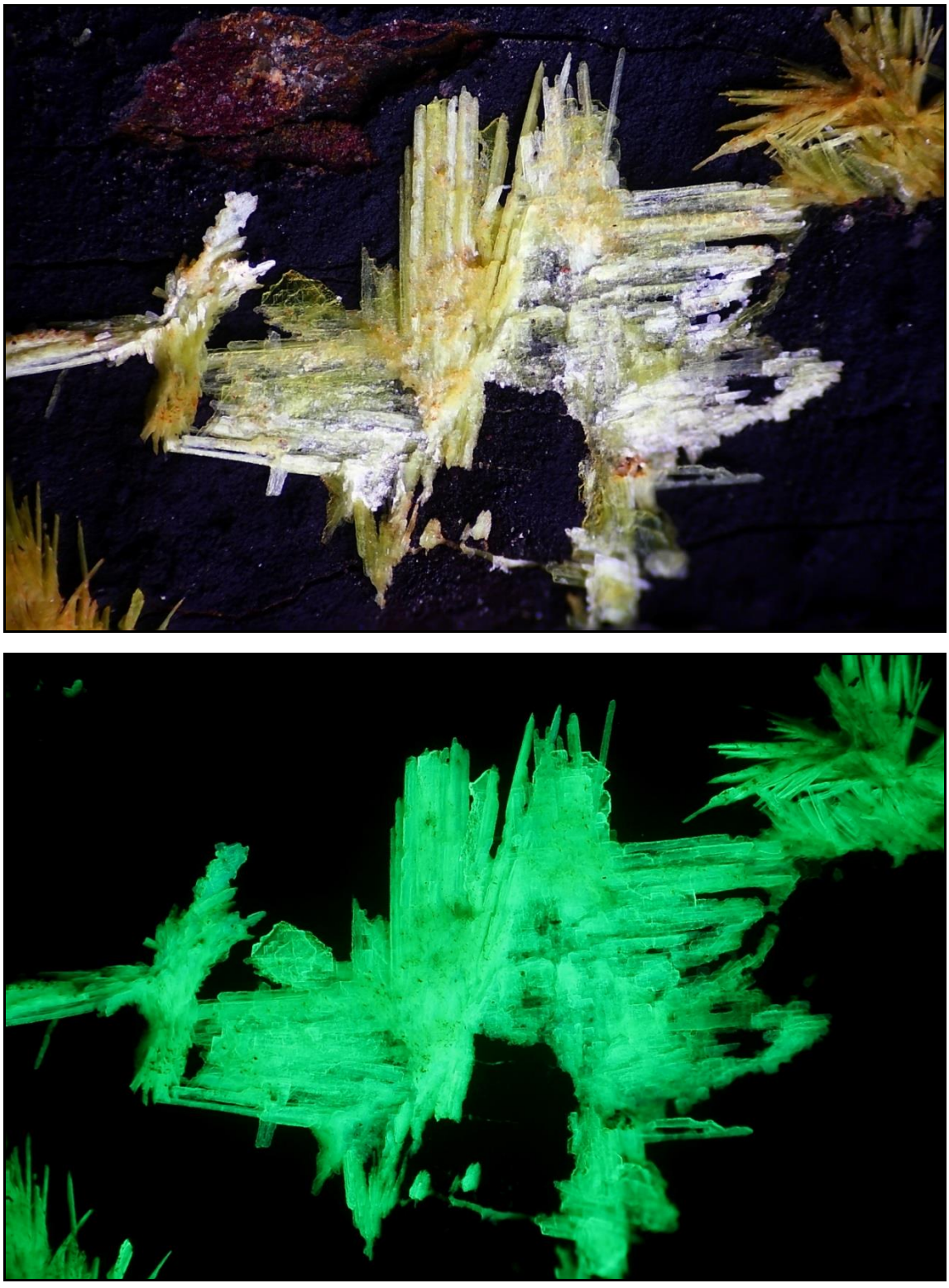

Figura I.1

Imagen de cristales de sabugalita, un fosfato de uranio, mina El Lobo, Don Benito, Badajoz. La imagen superior está tomada usando luz visible, la imagen inferior está tomada usando UV de onda larga, mostrando la fluorescencia. Multifoco + Stack_Duino. Encuadre 6 mm (Col. y Fot. H. Cócera).

Figura I.2

Cacoxenita, fosfato de hierro y aluminio, procedente de la mina La Paloma, Zarza la Mayor, Badajoz. 8º puesto en Nikon Small World 2010.
Multifoco. Encuadre 1,5 mm (Col. y Fot. H. Cócera).

Figura I.3

Cacoxenita, fosfato de hierro y aluminio, procedente de la mina La Paloma, Zarza la Mayor, Badajoz. 5º puesto en Nikon Small World 2012.
Multifoco. Encuadre 1,5 mm (Col. y Fot. H. Cócera).

Figura I.4

Fósil de díptero en ámbar del Báltico.
Multifoco+Stack_Duino. 2 mm (Col. y fot. H. Cócera).

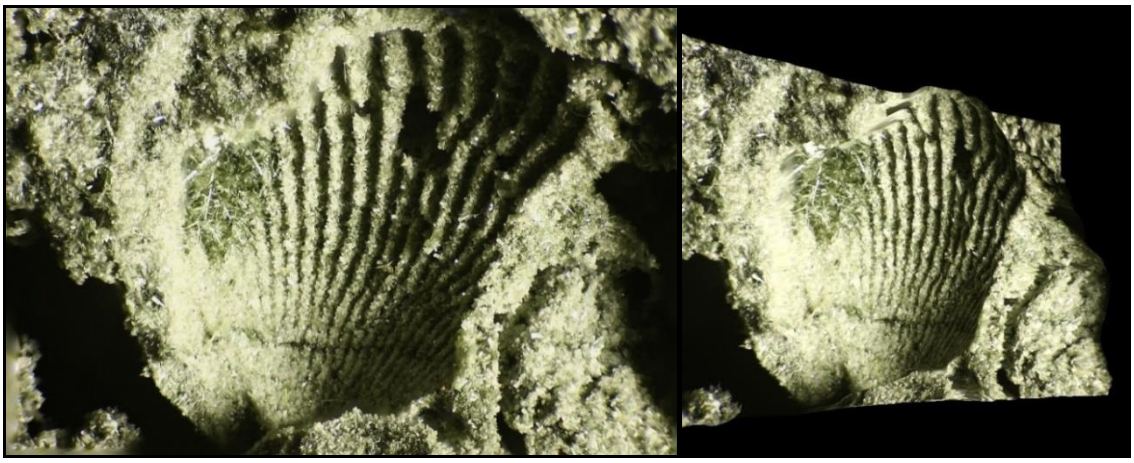

Figura I.5

Izquierda: molde de bivalvo en roca metamórfica, skarn.
Derecha: reconstrucción 3D del molde, mostrando como los costillas están formadas por pequeños cristales de un silicato no determinado.
Encuadre 2 cm (Fot. H. Cócera).

Figura I.6

Macla de rutilo.
Diamantina, Jequitinhonha valley, Minas Gerais, Brazil.
Macla de 5 mm.

Figura I.7

Cobre.
El Valle-Boinás, Begega, Belmonte de Miranda, Asturias.
Anchura 4 mm (Col. y Fot. H. Cócera).

Figura I.8

Calcosina.
Mina Las Cruces, Gerena, Sevilla.
Cristal de 5 mm. Col. G. García (Fot. H. Cócera).

Figura I.9

Conicalcocita pseudomorfica de azurita.
Calicata Dolores, Pastrana, Mazarrón, Murcia.
Encuadre 6 mm (Col. y Fot. H. Cócera).

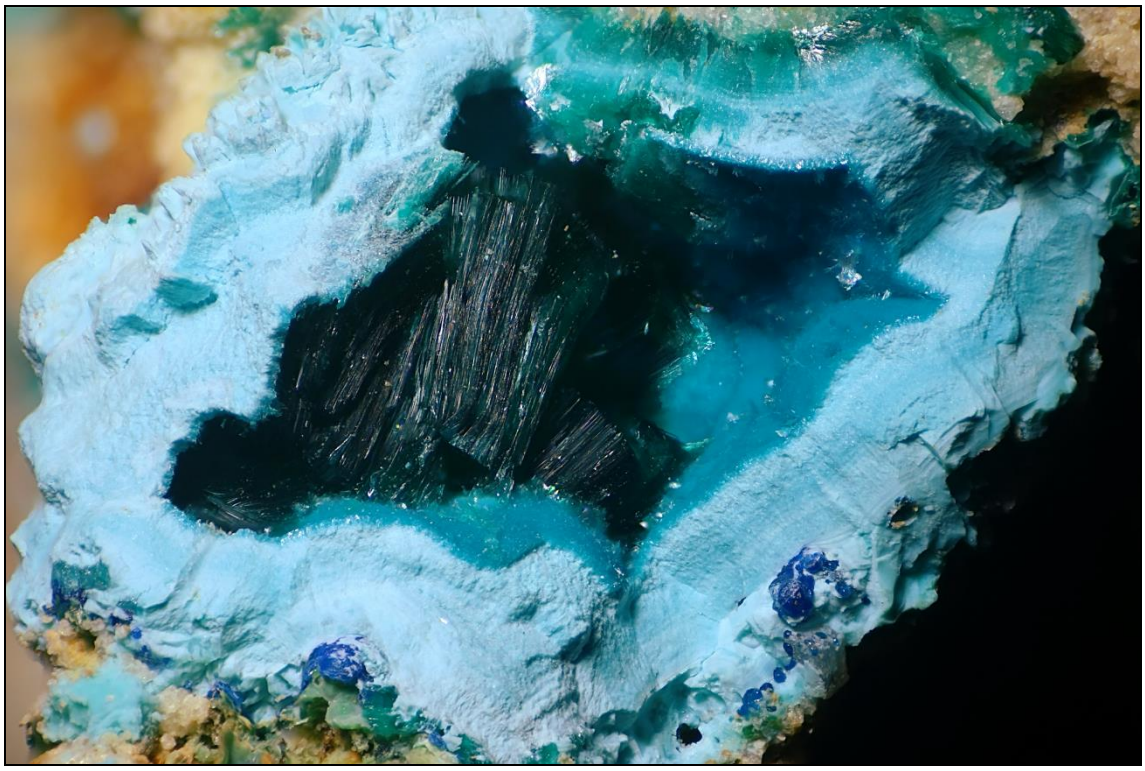

Figura I.10

Claraita y tirolita.
Mina la Amorosa, Villahermosa del Río, Castellon.
Encuadre 5 mm (Col. y Fot. H. Cócera).

Figura I.11

Auricalcocita.
Mina Pedreo, Arcentales, Vizcaya.
Encuadre 4 mm (Col. y Fot. H. Cócera).

Figura I.12

Armacolita.
Cantera Aljorra, La Aljorra, Murcia.
Cristal de 4 mm (Col. y Fot. H. Cócera).

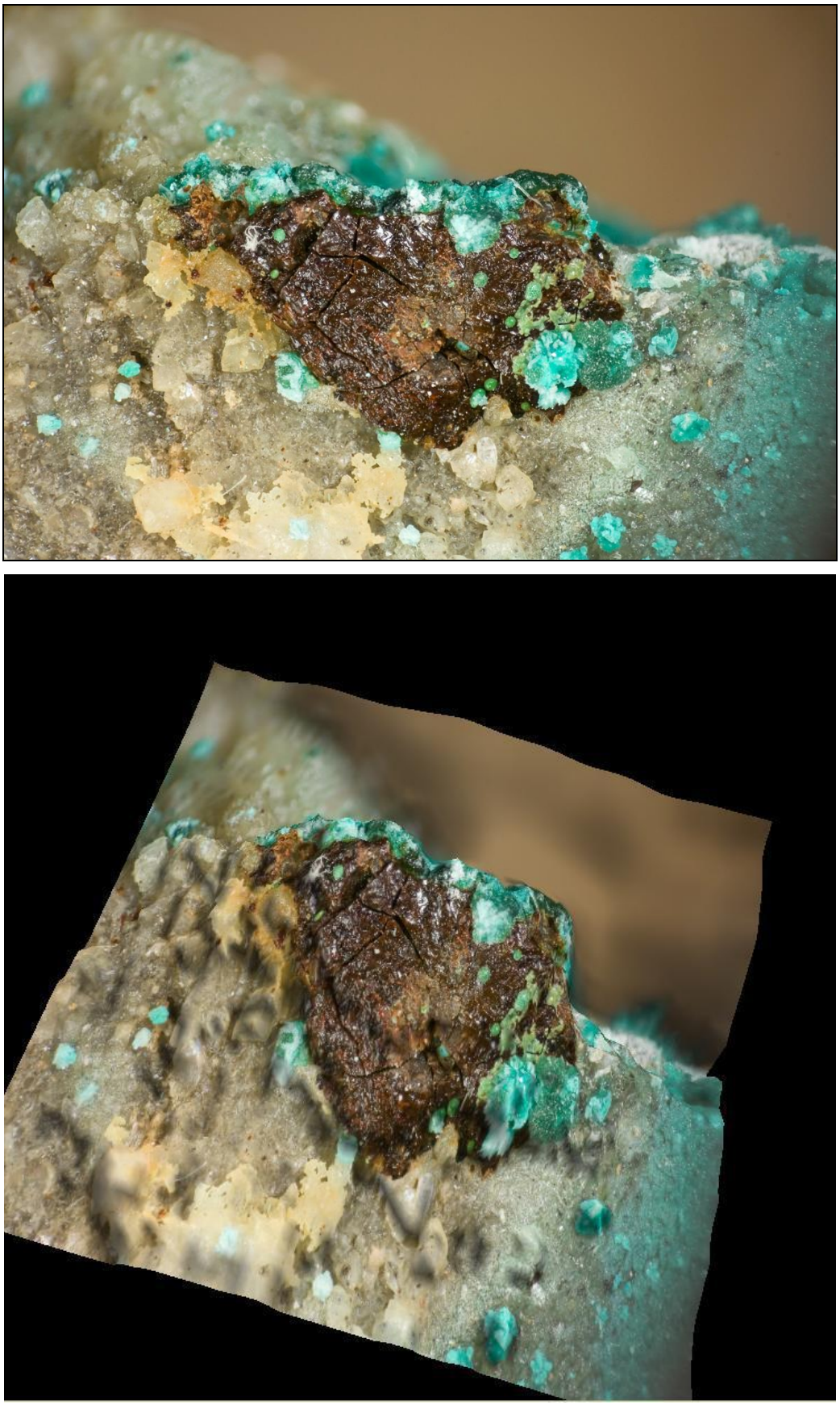

Figura I.13

Yakhontovita psm. de tenantita y reconstrucción 3D mostrando la forma del tetraedro.
Mina la Amorosa, Villahermosa del Río, Castellon.
Encuadre 7 mm (Col. y Fot. H. Cócera).

Figura I.14

Clinozoisita.
Cantera Los Arenales, Torás, Castellón.
Cristal de 1,5 mm (Col. y Fot. H. Cócera).

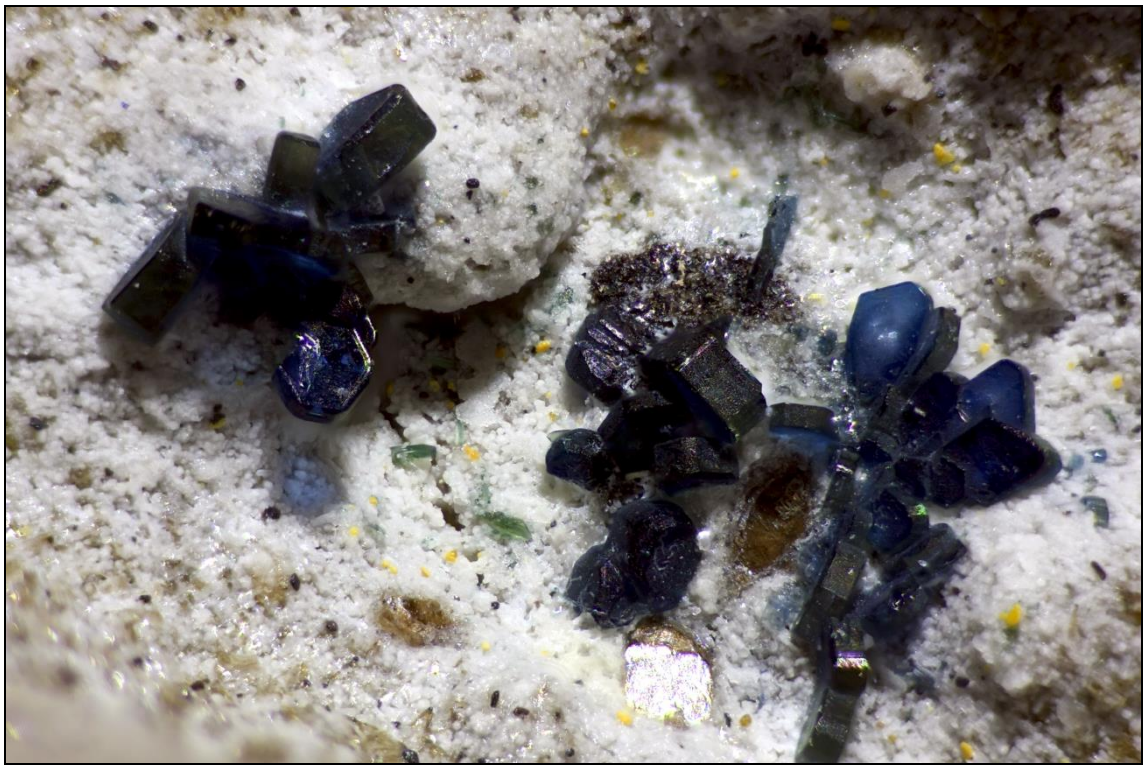

Figura I.12

Roedderita.
Pitón de Cancarix, Sierra de las Cabras, Hellín, Albcete.
Encuadre 1,5 mm. Col. R. Muñoz (Fot. H. Cócera).

Diseño, construcción y programación de un sistema automático y autónomo para la adquisición de fotografías multifoco destinado a documentación científica

Anexo II

Stack_Duino: Instrucciones de funcionamiento

1. STACK_DUINO 1.0

Apilado, multifoco o focus stacking, son algunos de los nombres que recibe la té fotográfica que produce imágenes de objetos tridimensionales con una profundid campo mayor que la permitida por la difracción, combinando las zonas en foco de una pila o serie de imágenes tomadas sobre un objeto a diferentes planos focales, cambiando la distancia entre el objeto y la cámara a pequeños intervalos. El combinado de la pila de imágenes es realizado mediante software, que es fácilmente accesible y de fácil manejo; pero la toma de la pila de fotografías, que posteriormente será cargada en dicho software, es posible hacerla de forma manual, tediosa y lenta, o bien mediante equipamiento automático. En el mercado podemos encontrar dispositivos comerciales que nos permiten hacer la serie de fotografías de forma automática, aunque por lo general suelen tener un coste económico alto; sin embargo, también es posible encontrar en la red diversos diseños "caseros" de bajo coste y relativo fácil montaje, aunque pobres en opciones y configuración.

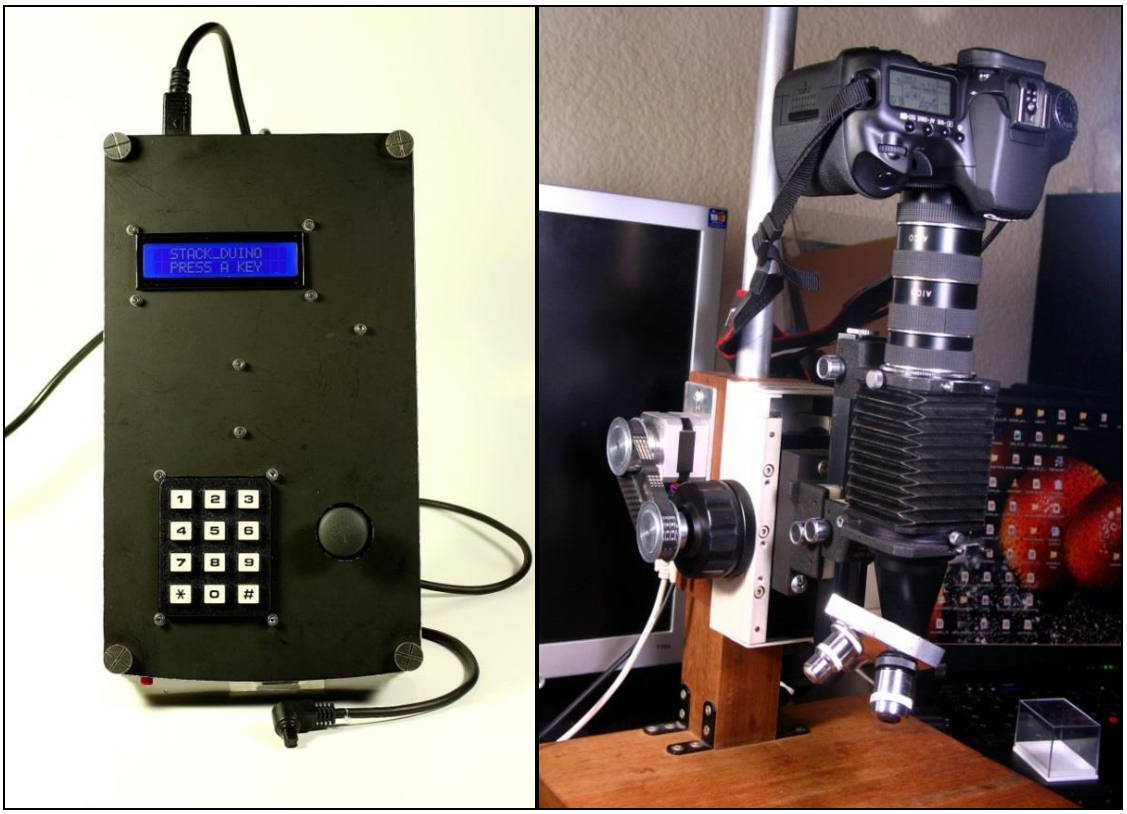

Figura II.1

Stack_Duino, caja de control a la izquierda. A la derecha, montado sobre un macroscopio.

Stack_Duino es un sistema automatizado y autónomo para la toma de pilas de imágenes. Stack_Duino dispone de dos modos de toma de imágenes; en el primero se marcan el punto de inicio (A), el punto final (B) y la distancia entre fotos, calculando el software el número de fotos a realizar; y un segundo método, en el que se marca el punto de inicio (A), el intervalo entre fotos y el número de fotos a realizar a partir de (A). Dispone además de un menú de configuración que le permite ser adaptado fácilmente a cualquier sistema, permite configurar micropasos del driver, pasos del motor, recorrido por vuelta del piñón de enfoque, sentido de giro del motor, tiempo de espera tras movimiento del motor y tiempo de espera tras realizar la foto.

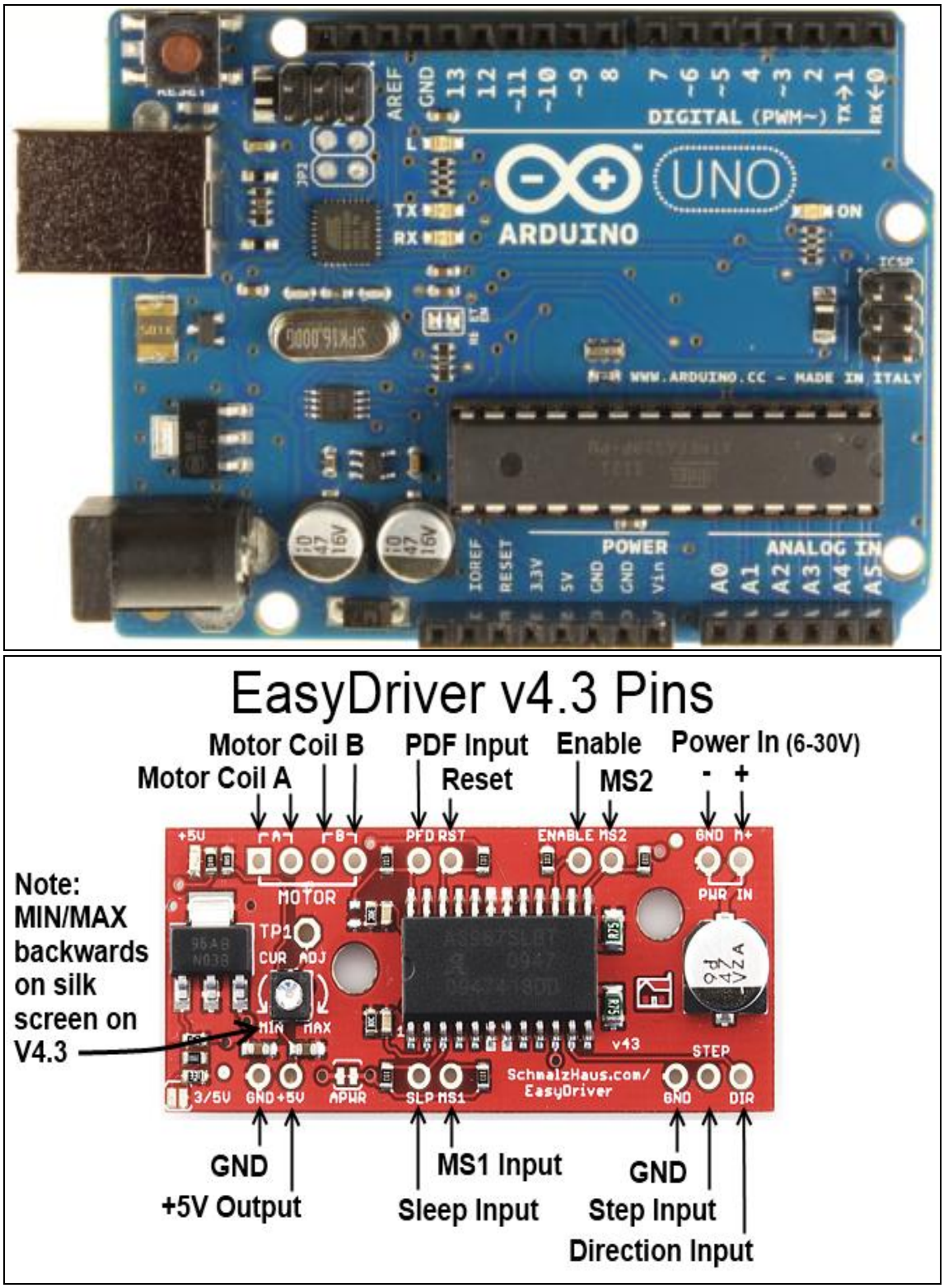

Figura II.2

Arriba: placa Arduino.
Abajo: Easydriver V4.4. (CC BY-SA 3.0) arduino.cc.

El corazón que controla este dispositivo está formado por una placa Arduino. Arduino es una plataforma de hardware libre, basada en una placa con un microcontrolador y un entorno de desarrollo, el hardware consiste en una placa con un microcontrolador Atmel AVR y puertos de entrada/salida; por otro lado el software consiste en un entorno de

desarrollo que implementa el lenguaje de programación Processing/Wiring y el cargador de arranque (boot loader) que corre en la placa.

La entrada de datos u órdenes se realiza a través de un teclado matricial de 12 teclas o a través de un joystick, el microcontrolador los recibe e interpreta, se encarga de realizar los cálculos necesarios, y una vez recibida la orden de realizar la pila, el microcontrolador manda las secuencias necesarias para hacer mover el motor y disparar la cámara. Al mismo tiempo, a través del LCD podemos seguir los pasos que vamos realizando.

Figura II.3

Conexión motor al Piñón de enfoque mediante poleas y correa dentada (Fot. H. Cócera)

Durante la toma de la serie se varia la distancia del sistema cámara+lentes frente al objeto a fotografiar, con lo que el plano focal en cada foto cambia levemente. Para ello o bien la cámara, o bien el objeto, se encuentra sobre un sistema que permita un movimiento lineal como pueda ser sobre un rail de enfoque o un piñón de un microscopio. En un dispositivo automático el encargado de realizar y transmitir este movimiento es un motor paso a paso, que se comunica con el microcontrolador a través de un driver, que se encarga de interpretar las señales del microcontrolador, y según estas, aplicar la tensión necesaria al motor para su movimiento. En concreto para Stack_Duino se ha elegido el driver “Easydriver”, que permite controlar motores paso a paso bipolares de hasta 35V y 0,75A por fase y permite hasta 1/8 micropasos. Como motor PAP se ha escogido un motor nema17, 200 pasos por vuelta, 12V y 0,33A por fase. Para el disparo de la cámara se ha optado por un circuito relé +optoacoplador aislando la cámara del resto del circuito. Todo el sistema se ha montado en el interior de una caja de un disco duro, que además aporta la alimentación al dispositivo, siendo esta de 12V y 2A.

En el montaje de la figura II.1, Stack_Duino ha sido conectado a un equipo óptico. La transmisión movimiento del motor al piñón de enfoque, se realiza mediante una correa dentada y dos ruedas dentadas de 20 dientes estándar imperial XL. Siendo el recorrido del piñón de enfoque de 200 micras/vuelta, con un motor de 200 pasos/vuelta y la relación entre las dos ruedas dentadas de 1:1, el conjunto permite una resolución mínima de 0,5 micras.

Figura II.4

Cristal de cobre nativo de El Valle, Begega, Asturias. Imagen tomada mediante Stack_Duino + Multifoco. (Col. y Fot. H. Cócera).

2. SOFTWARE

Para programar la tarjeta se necesita el ambiente de desarrollo Arduino. El entorno de desarrollo Arduino, IDE, es libre y de código abierto, disponible para Windows, Mac OS, y Linux. Está constituido por un editor de texto para escribir el código, un área de mensajes, una consola de texto, una barra de herramientas con botones para las funciones comunes, y una serie de menús. Permite la conexión al hardware de Arduino para cargar los programas y comunicarse con ellos.

Arduino utiliza para escribir el software lo que denomina "sketch" (*programa*). Estos programas son escritos en el editor de texto. Existe la posibilidad de cortar/pegar y buscar/remplazar texto. En el área de mensajes se muestra información mientras se cargan los programas y también muestra errores. La consola muestra el texto de salida para el entorno de Arduino incluyendo los mensajes de error completos y otras informaciones.

La descarga del IDE se puede realizar del siguiente link:

http://arduino.cc/

El sketch de Stack_Duino, es compatible con la versión 1.05. Una vez realizada la descarga, descomprimimos la carpeta y la situamos donde queramos. El dispositivo

utiliza una librería diferente a la que trae por defecto para el control del LCD. Esta librería se puede descargar del siguiente link:

https://bitbucket.org/fmalpartida/newliquidcrystal/downloads

Seleccionamos la última versión de LiquidCrystal, y la copiamos a la carpeta *"libraries"* del IDE de Arduino, sobrescribiendo la que allí se encuentre. Las instrucciones de funcionamiento e información del IDE podemos encontrarla en la página web del proyecto Arduino. Junto con los archivos suministrados en este documento podemos encontrar una la versión del IDE 1.05 en la que ya ha sido reemplazada la librería, y por lo tanto el paso anterior no será necesario.

Para instalar la tarjeta Arduino y poder cargar el sketch debemos instalar los drivers de éste, que se encuentran en el directorio *"drivers"* de la distribución del IDE. Dependiendo del sistema operativo que tengamos los pasos serán diferentes; en Windows, iremos a *"Agregar nuevo hardware"* del panel de control y seguiremos los pasos que nos vaya indicando. Para mayor información nos dirigiremos a la página oficial de Arduino:

http://arduino.cc/

Cargar el programa en la tarjeta:

Abre el sketch de Stack_Duino: ***Archivo>Abrir>Stack_Duino_1_00>Stack_Duino_1_00.***

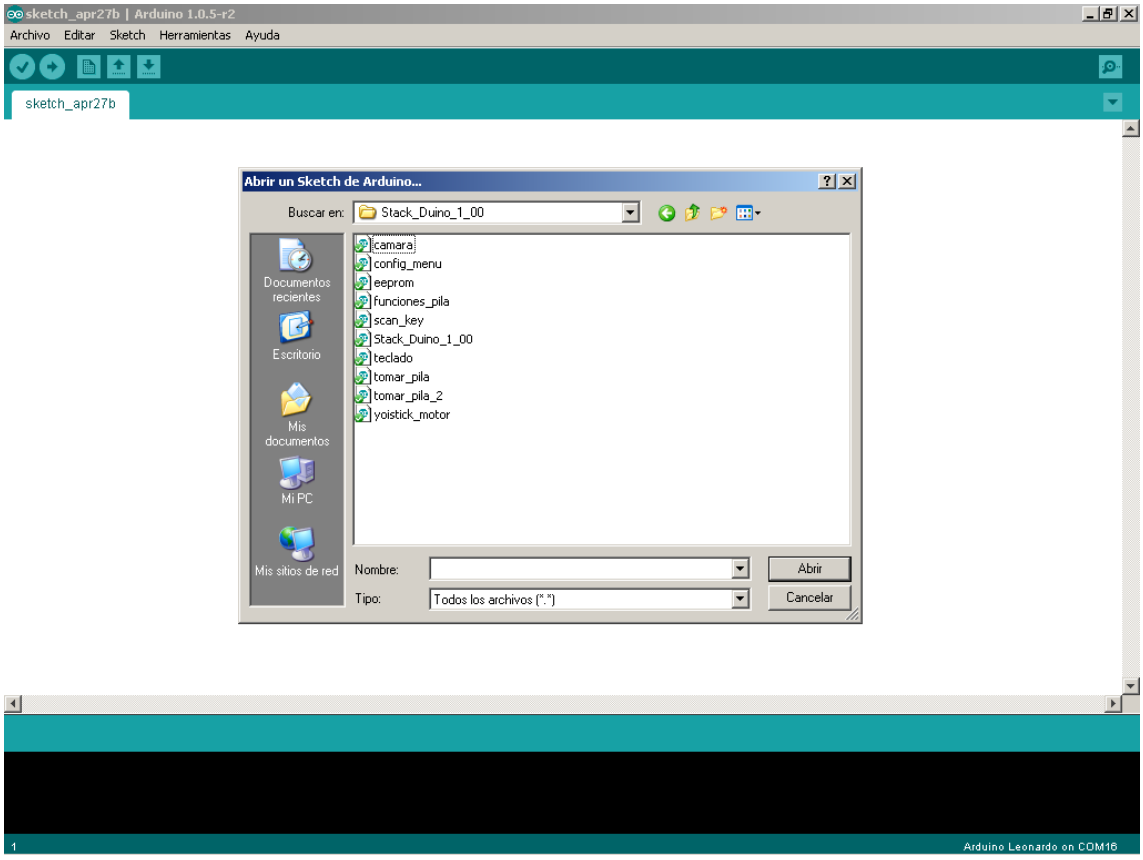

Figura II.5 : *Carga del sketch.*

Selecciona el puerto COM en el que tienes conectada la tarjeta Arduino en el menú *Herramientas > Puerto Serial***.** Para ver cuál es el puerto COM en el que está conectado tienes que ir al Administrador de Dispositivos de Windows (*Panel de control > Sistema > Hardware > Administrador de Dispositivos*). Busca "USB Serial Port" en la sección de puertos; esa es la tarjeta Arduino.

Necesitamos también especificar la tarjeta que se está utilizando. Hay que asegurarse de seleccionar la correcta en el menú *Herramientas> Tarjeta >Arduino Leonardo*.

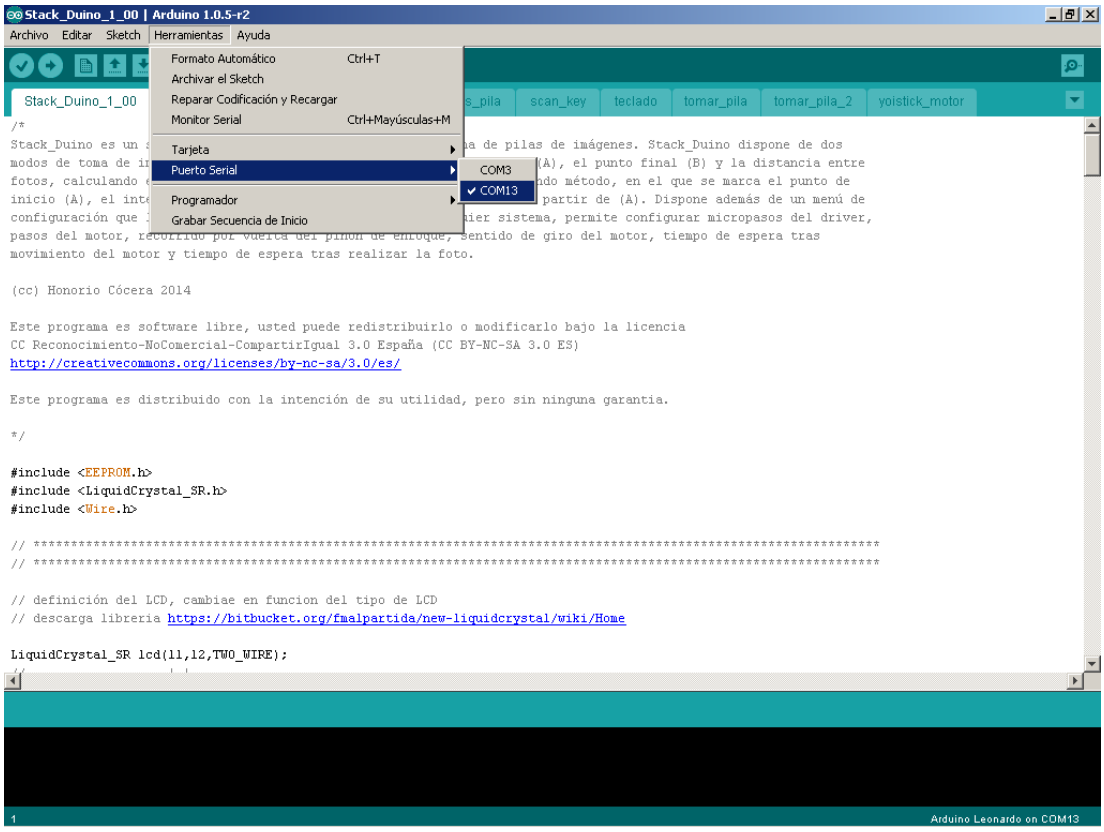

Figura II.6: *Pantalla configuración puerto COM.*

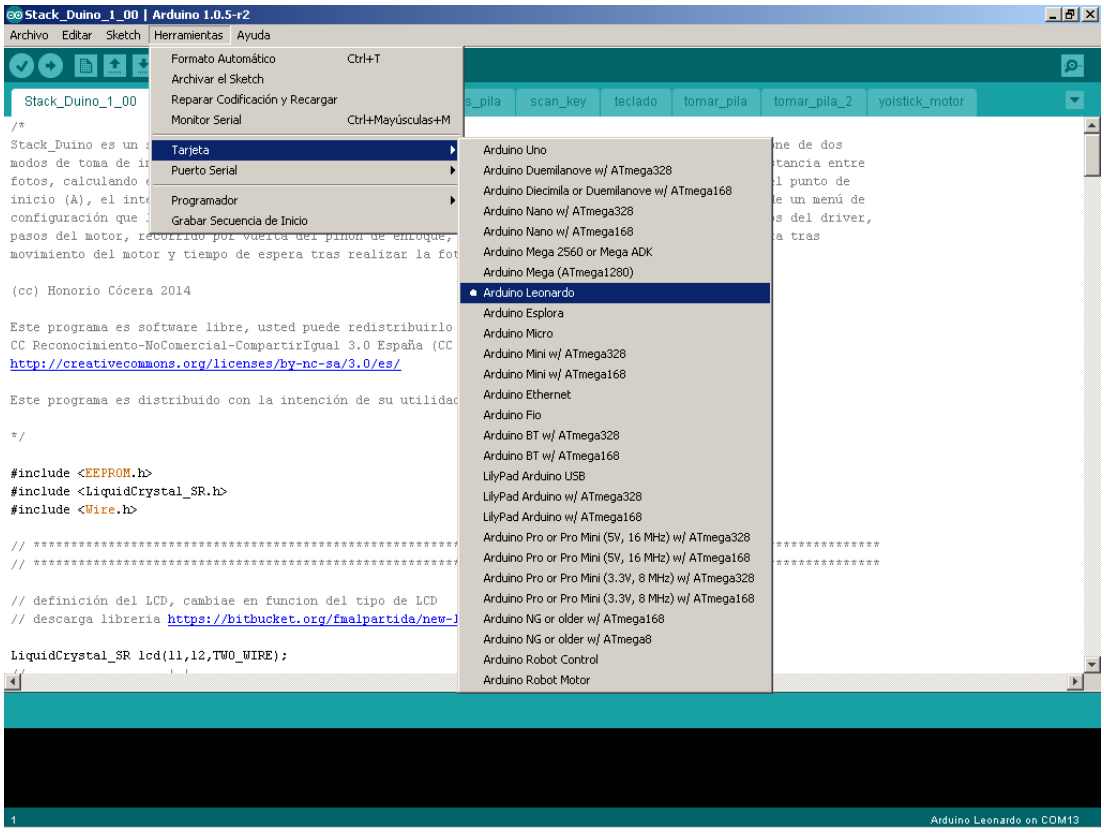

Figura II.7: *Pantalla configuración tarjeta Arduino Leonardo.*

Ahora para acabar, simplemente hacemos clic en el botón “Cargar”, con lo que comenzará la carga del sketch. Esperamos unos segundos y deberíamos ver los leds RX y Tx de la tarjeta parpadeando. Si el programa se cargó satisfactoriamente un mensaje aparecerá en la barra de status “Carga terminada”. Y ya podemos comenzar a usarlo.

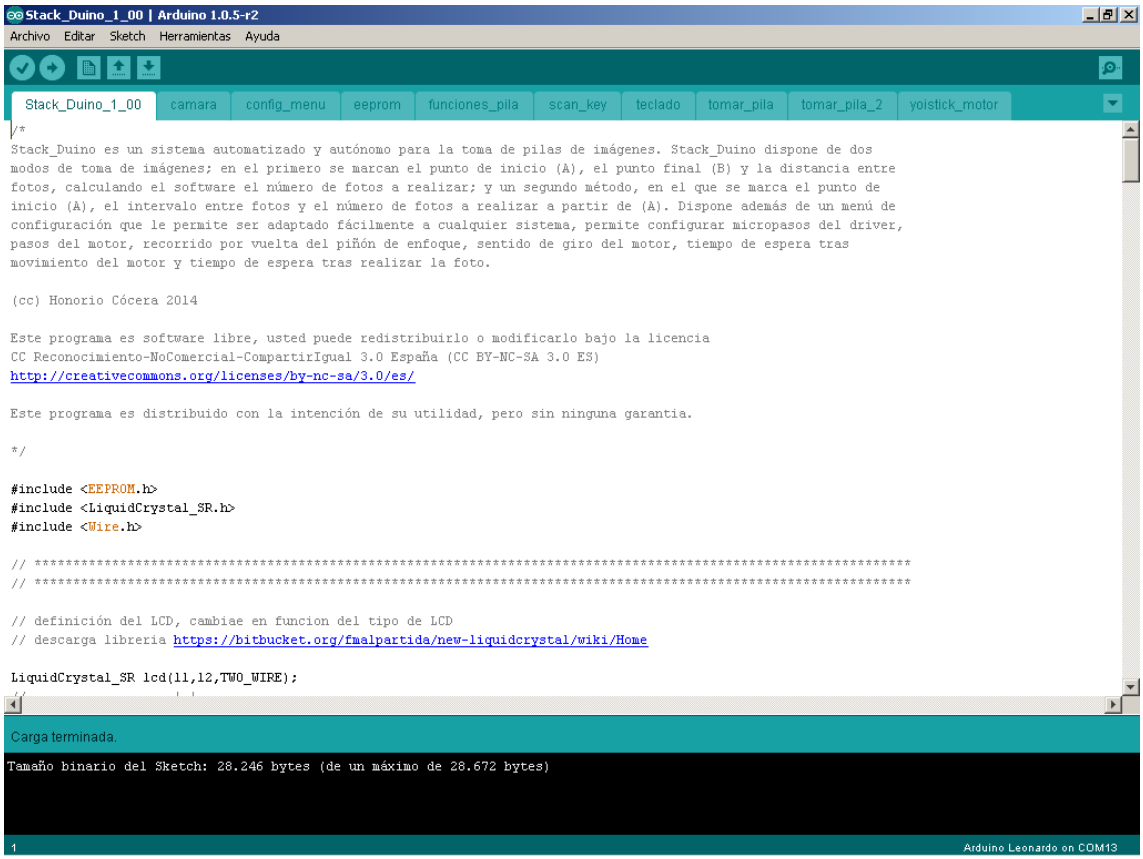

Figura II.8: *Pantalla carga del sketch.*

3. INSTRUCCIONES DE FUNCIONAMIENTO

Para el control del dispositivo y entrada de datos, Stack_Duino dispone de dos elementos, un joystick con movimiento XY más un botón, y un teclado matricial de 12 teclas. Adicionalmente, Stack_Duino dispone además de un botón de reseteo, que permite reiniciar el dispositivo en caso de problemas. Seguidamente explicaremos el funcionamiento de cada uno de los elementos.

3.1. Joystick

Dispone de dos ejes de movimiento, un en sentido vertical y otro en sentido horizontal. El movimiento en sentido vertical, ascendente o descendente, permite hacer mover el motor en uno u otro sentido respectivamente, rápidamente y con una velocidad variable dependiendo del grado de desplazamiento. La velocidad mínima puede ser fijada a través del menú de configuración general. El movimiento en el eje horizontal, permite un movimiento mucho más lento del motor, ideal para un ajuste fino de la posición, siendo la velocidad del movimiento fija.

3.2. Teclado

En concreto se trata de un teclado matricial de 12 teclas. Haciendo una pulsación corta o larga, el dispositivo permite realizar diferentes funciones. Para notación aquí,

añadiremos **_C** (clic, pulsación corta de 60 milisegundos) o **_H** (hold, pulsación larga de 800 milisegundos) tras el símbolo de la tecla, por ejemplo:

7_C	Pulsación corta de la tecla 7.
7_H	Pulsación larga de la tecla 7.

A continuación pasamos a describir el funcionamiento general y método de toma de pilas de imágenes.

Una vez conectada la alimentación, Stack_Duino muestra el mensaje de bienvenida, tras él, espera a que se pulse una tecla para comenzar. Una vez pulsada entra en el menú principal.

3.3. Menú Principal

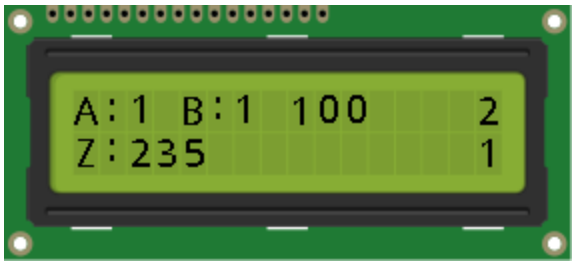

Figura II.9: *Pantalla menú principal.*

Pantalla – Línea 1	
A:1	Estado del punto **A:** **0** *no marcado* **1** *marcado*
B:1	Estado del punto **B**: **0** *no marcado* **1** *marcado*
100	Muestra la distancia a mover entre foto y foto
2	Muestra los micropasos

Pantalla – Línea 2	
Z:235	Posición del motor medido en pasos
1	Estado del motor: **0** *no conectado* **1** *conectado*

1_C	Mueve el motor 1 micra o unidad de distancia en sentido ascendente.
1_H	Mueve el motor 5 micras o unidades de distancia en sentido ascendente.
2_C	Mueve el motor 100 micras o unidades de distancia en sentido ascendente.

2_H	Mueve el motor una vuelta entera del piñón de enfoque en sentido ascendente.
3_C	Marca la posición de inicio, A, de toma de la pila.
3_H	Mueve el motor a la posición de inicio, A, de la pila.
4_C	Mueve el motor 1 micra o unidad de distancia en sentido descendente.
4_H	Mueve el motor 5 micras o unidades de distancia en sentido descendente.
5_C	Mueve el motor 100 micras o unidades de distancia en sentido descendente.
5_H	Mueve el motor una vuelta entera del piñón de enfoque en sentido descendente.
6_C	Marca la posición final, B, de toma de la pila.
6_C	Mueve el motor a la posición final, B, de la pila.
7_C	Menú configurar micras o distancia entre foto y foto. Permite configurar la cantidad de micras o distancia a mover entre foto y foto en otras unidades, cuando se está tomando la pila.
7_H	Menú de configuración general. Permite entrar a dicho menú y ajustar diversos parámetros.
8_C	Permite ver el estado y posición de los puntos de inicio y final de la pila.
8_C	Permite ajustar a 0, el estado, posición del motor (z), y los puntos de inicio y final de la pila.
9_C	Mueve el motor el valor ajustado para micras o distancia por foto en 7_C, el número de micras o distancia a mover entre foto y foto al realizar la pila, permitiendo ver el solapamiento que habrá entre fotos.
9_H	Inicia el modo de toma de pila A/nFotos.
***_C**	Vuelve a atrás.
***_H**	Ninguna función en el menú principal.
0_C	Conecta o desconecta el motor, permitiendo que el piñón de enfoque pueda girar libremente cuando el motor esta desconectado.
#_C	Confirmar.
#_H	Inicia el modo de toma de pila A-B.

La primera vez que conectemos Stack_Duino, debemos introducir los parámetros de configuración de nuestro equipo. Con estos valores, se realizaran los cálculos necesarios para las diferentes funciones. Para ello pulsamos 7_H y entraremos en el menú de configuración general, que a continuación describimos. Una vez introducidos, estos quedan grabados en memoria y permanecen en ella aunque se apague el dispositivo.

Para poder pasar de pasos, el valor de Z, a unidades de distancia, podemos usar la siguiente ecuación:

$$Distancia = Z * \frac{Distancia\ por\ vuelta}{Total\ pasos}$$

Donde el total de pasos, será el producto de los pasos del motor por el valor de microsteps. Esta ecuación también nos permite medir la distancia entre dos puntos, con tan solo sustituir en Z, la diferencia la diferencia entre esos dos puntos.

3.4. Menú de Configuración General

En la pantalla aparecerá el parámetro a configurar y el valor guardado en la memoria. La primera vez que inicialicemos el dispositivo mostrará los valores por defecto del programa. Para movernos entre el menú e introducir los valores utilizaremos las siguientes teclas:

8_C	Sube en el menú.
0_C	Baja en el menú.
***_C**	Sale del menú de configuración general para volver al menú principal.
#_C	Entra en el menú de configuración del valor mostrado en pantalla. Para introducir el valor deseado, una vez pulsado #_C, veremos que aparece un signo de interrogación y a continuación introducimos el valor y damos a confirmar. Únicamente permite introducir valores enteros. Las teclas a usar son las siguientes **0_C**, **1_C** ….**9_C**: Introducimos el valor. ***_C**: Borra el último número introducido. ***_H**: Borra todos los números. **#_C**: Confirma el valor y lo guarda en la memoria, saliendo posteriormente y volviendo al menú de configuración general. **@_H**: Salir del menú sin guardar el valor, se mantiene el valor que había anteriormente.

Los valores que podemos configurar son los siguientes:

Microsteps

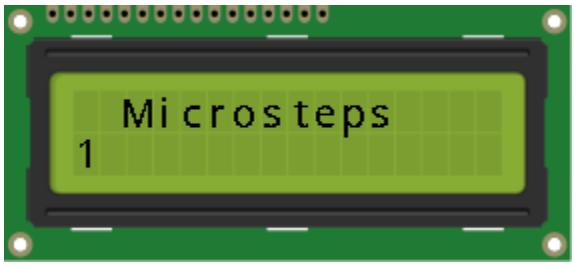

Figura II.10: *Pantalla menú configuración: microsteps.*

Los microsteps o micropasos en español, permiten aumentar el número de pasos del motor. Los valores permitidos son 1, 2, 4, 8; cualquier valor diferente introducido dará error. El motor permite un número de pasos fijo, el uso de los micropasos permite aumentar las posiciones que puede adoptar el motor por cada vuelta, y por tanto, aumentar el número de pasos y la resolución del equipo. El driver que maneja el motor

en Stack_Duino, Easydriver, permite unos valores de micropasos de 1,2,4 y 8. Si elegimos el valor de 2, se duplican los pasos; para valor de 4, se cuadriplica y así sucesivamente. Una ventaja de su uso, es una menor vibración del motor, pero para mayores valores, hay una pérdida de reproducibilidad de los pasos. En la práctica tan solo los valores 1 y 2 son efectivos, no se recomienda usar los valores 4 y 8.

Time_1_Motor

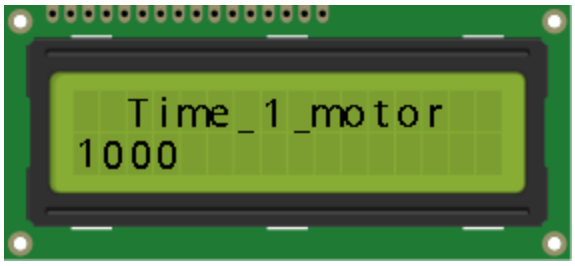

Figura II.11: *Pantalla menú configuración: tiempo 1.*

Permite configurar el tiempo de espera entre el movimiento del motor y el inicio de la toma de la foto, esto permite eliminar las vibraciones del equipo. El tiempo debe introducirse en milisegundos (1s = 1000 milisegundos). Por ejemplo, para introducir 3,5 segundos, escribiríamos 3500.

Time_2_foto

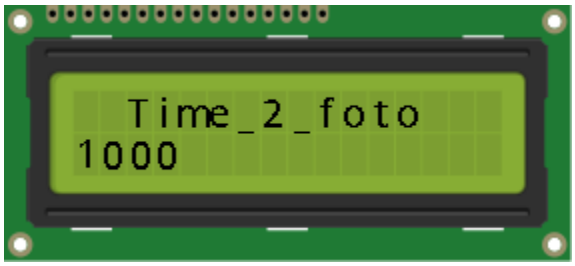

Figura II.12: *Pantalla menú configuración: tiempo 2.*

Permite configura el tiempo tras la toma de la foto y el siguiente movimiento del motor. Este tiempo comprende el tiempo necesario para que la cámara tome la foto más el tiempo necesario para que guarde y procese el archivo. Por ejemplo en una foto con un tiempo de exposición de 2 segundo, y un tiempo de procesado y guardado de otros 2 segundos, deberíamos poner un tiempo mayor de 4 segundos, dando algo más de margen. Al igual que antes, debemos introducir el tiempo en milisegundos.

Pasos_vuelta

Figura II.13: *Pantalla menú configuración: pasos por vuelta.*

Es el número de pasos del motor por vuelta. El valor depende del motor usado, siendo el más común, el de 200 pasos por vuelta. Debe ser mayor que 0.

Sentido_giro

Figura II.14: *Pantalla menú configuración: sentido de giro del motor.*

Permite cambiar el sentido de giro del motor, en función de cómo hayamos construido nuestro equipo y el conexionado del motor, el motor funcionará en uno u otro sentido, para hacer coincidir este sentido con el sentido de giro de joystick o el teclado podemos cambiar este valor. Los valores permitidos son 0 y 1, que corresponden a uno y otro sentido, introducir un valor diferente dará error.

Sonido

Figura II.15: *Pantalla menú configuración: activar-desactivar sonido.*

Permite conectar o desconectar el sonido,. Los valores permitidos son 0 y 1, 0 desconecta el sonido y 1 lo conecta, introducir un valor diferente dará error.

Velocidad

Figura II.16: *Pantalla menú configuración: velocidad mínima movimiento motor.*

Permite establecer una velocidad mínima a la que se moverá el motor, la velocidad se mide en Hz. Velocidades bajas en motores paso a paso pueden producir problemas de vibraciones mientras que velocidades muy altas hacen que driver del motor entre en resonancia impidiendo el movimiento del motor, el driver utilizado, Easydriver, permite valores máximos en torno a 500 Hz. El movimiento mediante el joystick permite variar la velocidad de movimiento del motor, siendo la menor la velocidad introducida y la

mayor la velocidad introducida*5, por lo que un valor adecuado se encuentra entre 70 a 100 Hz.

Distancia_vuelta

Figura II.17: *Pantalla menú configuración: distancia por vuelta.*

Número de micras o unidades que se mueve el equipo por cada revolución del piñón de enfoque, el valor debe estar en micras, aunque es posible usar otras unidades, siempre que "***Distancia_vuelta***" y "***Distancia_x_foto***" estén en las mismas unidades. Debe ser mayor que 0. En caso de que la transmisión del motor al piñón de enfoque se realice mediante poleas, el valor a introducir será:

$$Distancia\ \ vuelta = Desplazamiento\ piñon\ enfoque * \frac{Diametro\ polea\ piñon}{Diametro\ polea\ motor}$$

Replica_Fotos

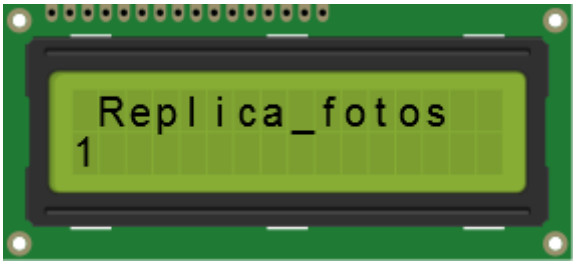

Figura II.18: *Pantalla menú configuración: número de fotos por plano.*

Permite indicar el número de fotos a realizar a un mismo nivel de foco. Esto permite la toma de varias imágenes iguales, que posteriormente pueden ser combinadas para la reducción de ruido u otras necesidades.

Tipo cámara

Figura II.19: *Pantalla menú configuración: cámara.*

Stack_Duino permite el uso de dos tipos de cámaras. Cámaras controladas a través de software como las cámaras microscopía, siempre y cuando el software permita el

disparo mediante una pulsación del teclado, o cámaras accionadas mecánicamente como las DSLR.

Para el primer tipo de cámara, Stack_Duino emula una pulsación del teclado, enviándola al software que controla la cámara, que es el encargado del disparó y toma de la imagen. Para ello deberemos tenerlo conectado a nuestro ordenador y el programa de control de la cámara seleccionado durante la toma de la pila de imágenes. La tecla a emular, depende de un software a otro, para indicar cuál es, únicamente debemos tomar en la tabla II.1, el valor numérico correspondiente a la tecla en cuestión e introducirlo, presionando #_C para confirmar. Por ejemplo, para el uso de cámaras de microscopía de la marca Leica, su manejo se hace a través del programa Leica Adquisition Suite (LAS), el cual, permite el disparo mediante la presión de la tecla F3. En este caso para configurar Stack_Duino, únicamente deberemos introducir el valor 26.

El disparo de cámaras DSLR se hace de forma mecánica, conectando ésta a Stack_Duino mediante el cable adecuado según la marca al jack de 3,5mm de la caja de control. Para configurarlo, únicamente debemos introducir el valor 0 en el menú, que activa el disparo mecánico. Stack_Duino está pensado principalmente para el uso de este tipo de cámaras, debido por una parte a que podemos prescindir del ordenador y por tanto, eliminar las vibraciones provenientes del ventilador de éste, que puede llegar a afectar a la calidad de la imagen al trabajar a grandes ampliaciones; por otra parte este tipo de cámaras dan una mayor calidad de imagen y un costo mucho menor que las cámaras de microscopía.

Tabla II.I

Valor	Tecla	Valor	Tecla	Valor	Tecla
0	Cámara réflex	46	9	92	r
1	KEY_LEFT_CTRL	47	A	93	s
2	KEY_LEFT_SHIFT	48	B	94	t
3	KEY_LEFT_ALT	49	C	95	u
4	KEY_LEFT_GUI	50	D	96	v
5	KEY_RIGHT_CTRL	51	E	97	w
6	KEY_RIGHT_SHIFT	52	F	98	x
7	KEY_RIGHT_ALT	53	G	99	y
8	KEY_RIGHT_GUI	54	H	100	z
9	KEY_UP_ARROW	55	I	101	:
10	KEY_DOWN_ARROW	56	J	102	;
11	KEY_LEFT_ARROW	57	K	103	,
12	KEY_RIGHT_ARROW	58	L	104	.
13	KEY_BACKSPACE	59	M	105	-
14	KEY_TAB	60	N	106	<
15	KEY_RETURN	61	Ñ	107	>
16	KEY_ESC	62	O	108	!
17	KEY_INSERT	63	P	109	"
18	KEY_DELETE	64	Q	110	·
19	KEY_PAGE_UP	65	R	111	$
20	KEY_PAGE_DOWN	66	S	112	%
21	KEY_HOME	67	T	113	&
22	KEY_END	68	U	114	(
23	KEY_CAPS_LOCK	69	V	115	)

24	KEY_F1	70	W	116	?
25	KEY_F2	71	X	117	¿
26	KEY_F3	72	Y	118	`
27	KEY_F4	73	Z	119	´
28	KEY_F5	74	a	120	+
29	KEY_F6	75	b	121	ç
30	KEY_F7	76	c	122	Ç
31	KEY_F8	77	d	123	*
32	KEY_F9	78	e	124	º
33	KEY_F10	79	f	125	ª
34	KEY_F11	80	g	126	¡
35	KEY_F12	81	h	127	~
36	espacio	82	i	128	[
37	0	83	j	129	]
38	1	84	k	130	^
39	2	85	l	131	#
40	3	86	m		
41	4	87	n		
42	5	88	ñ		
43	6	89	o		
44	7	90	p		
45	8	91	q		

Tabla II.I: *Valores para emular tecla.*

3.5. Menú configurar distancia entre foto y foto

Figura II.20: *Pantalla menú configuración: distancia a mover entre foto y foto.*

Aquí podemos configurar el desplazamiento a mover del equipo por el motor entre foto y foto durante la toma de imágenes. Para acceder al mismo presionamos 7_C, apareciendo en la pantalla el valor guardado en memoria. Aunque Stack_Duino está pensado principalmente para trabajar con microscopios o similares, y por lo tanto pensado para trabajar en micras, esta distancia, se podrá especificar en cualquier unidad, siempre que "Distancia_vuelta" y "Distancia_x_foto" estén en las mismas unidades.

Además, también es posible aplicarlo para movimientos no lineales, como en movimientos rotatorios, en los que el motivo gira en torno al eje óptico del sistema; como por ejemplo en la fotografía de 360º o para toma pilas de imágenes para generación de modelos 3D. En este caso, las unidades a usar son grados, definiendo el valor de "Distancia_vuelta" como 360º, y valor de "Distancia_x_foto" n grados. De esta forma, al girar el motor, el motivo girará n grados respecto al eje óptico; a diferencia del movimiento lineal, en el que el motivo se alejará o acercará a la cámara.

La distancia mínima que es capaz de resolver o mover el sistema vendrá dada por:

$$Distancia\ mínima = \frac{Distancia\ por\ vuelta}{Total\ pasos}$$

Donde el total de pasos, será el producto de los pasos del motor por el valor de microsteps. Por ejemplo, para un sistema de 200 micras por vuelta del piñón de enfoque, un motor de 200 pasos y el valor de microsteps ajustado a 1; la distancia mínima que será capaz de mover Stack_Duino será de 1 micra.

Las teclas a usar son las siguientes:

***_C**	Sale del menú de configuración micras/fotos para volver al menú principal.
#_C	Entra en el menú de configuración del valor. Para introducir el valor deseado, una vez pulsado #_C, veremos que aparece un signo de interrogación y a continuación lo introducimos y damos a confirmar. Las teclas a usar se expresan en el cuadro siguiente.

0_C, **1_C** ….**9_C**	Introducimos el valor.
***_C**	Borra el último número introducido.
***_H**	Borra todos los números.
#_C	Confirma el valor y lo guarda en la memoria, saliendo posteriormente y volviendo a al menú de configuración de micras/fotos.
#_H	Salir del menú sin guardar el valor, se mantiene el valor que había anteriormente en memoria.

3.6. Toma de la pilas de imágenes

Una vez configurado por primera vez, podemos pasar a tomar la pila de imágenes. Stack_Duino permite dos tipos de tomas de pila.

Modo A-B

En este método se marca el punto de inicio A y el punto final B de la pila, se introduce el número de micras a desplazar entre foto y foto; el programa calcula el número de fotos a realizar y muestra los datos en pantalla. Los pasos generales son los siguientes.

Posicionamos la muestra a fotografiar, ajustamos los parámetros de la cámara a usar.

0_C	Conectamos el motor, una vez hecho, no debemos desconectarlo hasta que no acabe la toma de imágenes, tampoco debemos mover el piñón de enfoque de forma manual, de lo contrario el dispositivo no podría saber la posición en la que se encuentra.

Búsqueda del punto de inicio. Bien mediante el teclado o bien mediante el joystick, movemos el motor arriba o abajo, con lo que el plano focal va cambiando. Buscamos el punto de inicio, y una vez encontrado lo marcamos. Las teclas a usar son:

3_C	Marca la posición en la que esté como punto de inicio A, la pantalla mostrará que el punto A ha sido marcado y la posición.
#_C	Volver al menú principal.

Búsqueda del punto final. Ahora, igual que antes, movemos el plano focal hacia abajo para buscar el punto final. Una vez encontrado lo marcamos como el punto anterior.

6_C	Marca la posición en la que esté como punto de final B, la pantalla mostrará que el punto B ha sido marcado y la posición. Si A y B han sido marcados, la pantalla también mostrará la diferencia de posición entre los dos en pasos.
#_C	Volver al menú principal.

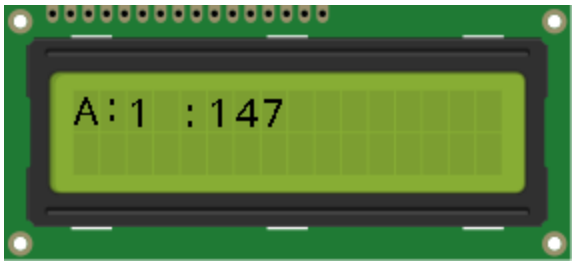

Figura II.21: *Pantalla de confirmación punto A.*

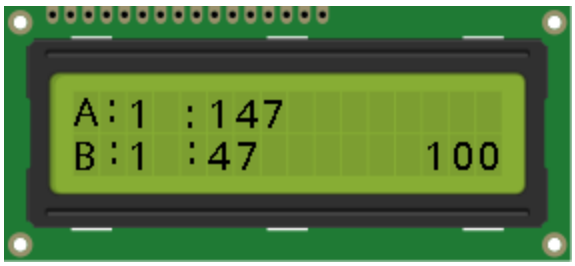

Figura II.22: *Pantalla de confirmación de punto B.*

Si queremos comprobar la posición marcada nuevamente, pulsaríamos 8_C, con lo que nuevamente mostraría la posición, estado y diferencia de los puntos de inicio y final. Presionando cualquier tecla volveríamos al menú principal. También es posible resetear todos los parámetros, posición y estado de los puntos A y B, y posición del motor z, a 0; para ello pulsamos 8_H, para salir presionamos cualquier tecla.

El siguiente paso es el ajuste del desplazamiento del plano focal entre foto y foto, para ello presionamos 7_C, con lo que se muestra en pantalla el valor que tiene en memoria.

Podemos usar este valor con lo que simplemente salimos mediante *_C, pero si deseamos cambiarlo teclas a usar son las siguientes:

***_C**	Sale del menú de configuración “Distancia_x_foto” para volver al menú principal.
#_C	Entra en el menú de configuración del valor. Para introducir el valor deseado, una vez pulsado #_C, veremos que aparece un signo de interrogación y a continuación lo introducimos y damos a confirmar. Las teclas a usar se expresan en el cuadro siguiente.

0_C, 1_C ….9_C	Introducimos el valor.
***_C**	Borra el último número introducido.
***_C**	Borra todos los números.
#_C	Confirma el valor y lo guarda en la memoria, saliendo posteriormente y volviendo a al menú de configuración de "Distancia_x_foto".
#_H	Salir del menú sin guardar el valor, se mantiene el valor que había anteriormente.

Una vez todo ajustado, ya podemos comenzar a tomar la pila de imágenes, para ello simplemente presionamos 0_H, la pantalla mostrará la diferente información, si quisiéramos volver atrás pulsaríamos *_C, para confirmar y comenzar la pila pulsaríamos #_C.

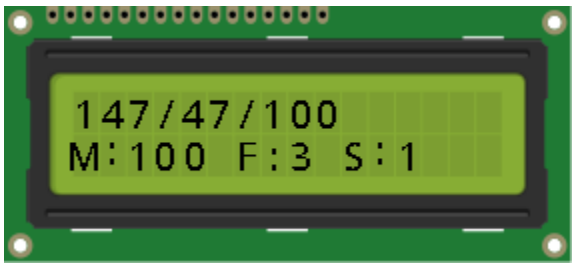

Figura II.23: *Pantalla de modo A-B.*

Pantalla - Línea 1	
147	Muestra la posición del punto A, en pasos.
47	Muestra la posición del punto B, en pasos.
100	Diferencia en pasos entre el punto A y B.

Pantalla - Línea 2	
M:100	Muestra las micras a mover entre foto y foto.
F:3	Muestra las fotos a realizar.
S:1	Muestra los micropasos.

El dispositivo comenzaría a tomar la pila. Durante la toma el dispositivo mantiene conectado el bloqueo de exposición de la cámara. Durante la toma también podemos:

***_C**	Abortar el proceso y volver al menú principal.
0_C	Pausa manteniendo el bloqueo de exposición, si se quiere eliminar el bloqueo pulsaríamos 8_C. Para continuar presionar 0_C.
8_C	Pausa sin mantener el bloqueo de exposición. Para continuar pulsar 0_C.

Una vez acabada la toma, nos mostrará el desplazamiento total en micras y nos pregunta que queremos hacer, las opciones disponibles son:

7_C	Mueve el motor y toma una foto adicional.
8_C	Vuelve a realizar la pila con los mismos parámetros.
9_C	Hace n fotos adicionales.
#_C	Volver al menú principal reseteando los valores de posición y estado de A y B, y la posición del motor Z.
0_C	Volver al menú principal pero sin resetear los valores anteriores.

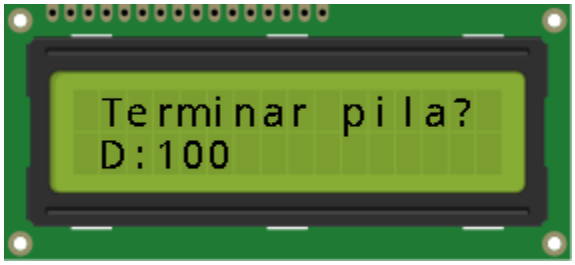

Figura II.24: *Pantalla de final de toma de pila.*

Modo A/nFotos

En este modo se marca el punto de inicio A y el número de fotos a hacer y se introduce el número de micras a desplazar entre foto y foto. Los pasos generales son los siguientes:

0_C	Conectamos el motor, una vez hecho, no debemos desconectarlo hasta que no acabe la toma de imágenes, tampoco debemos mover el piñón de enfoque de forma manual, de lo contrario el dispositivo no podría saber la posición en la que se encuentra.

Búsqueda del punto de inicio. Bien mediante el teclado o bien mediante el joystick, movemos el motor arriba o abajo, con lo que el plano focal va cambiando. Buscamos el punto de inicio, y una vez encontrado lo marcamos. Las teclas a usar son:

3_C	Marca la posición en la que esté como punto de inicio A, la pantalla mostrará que el punto A ha sido marcado y la posición.
#_C	Volver al menú principal.

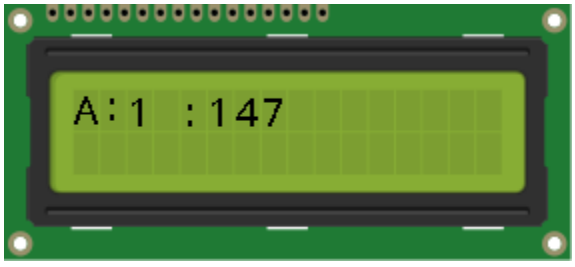

Figura II.25: *Pantalla de confirmación de punto A.*

El siguiente paso es el ajuste del desplazamiento del plano focal entre foto y foto, para ello presionamos 7_C, con lo que se muestra en pantalla el valor que tiene en memoria. Podemos usar este valor con lo que simplemente salimos mediante *_C, pero si deseamos cambiarlo teclas a usar son las siguientes:

***_C**	Sale del menú de configuración “Distancia_x_foto” para volver al menú principal.
#_C	Entra en el menú de configuración del valor. Para introducir el valor deseado, una vez pulsado #_C, veremos que aparece un signo de interrogación y a continuación lo introducimos y damos a confirmar. Las teclas a usar se expresan en el cuadro siguiente.

0_C, 1_C ….9_C	Introducimos el valor.
***_C**	Borra el último número introducido.
***_C**	Borra todos los números.
#_C	Confirma el valor y lo guarda en la memoria, saliendo posteriormente y volviendo a al menú de configuración de “Distancia_x_foto”.
#_H	Salir del menú sin guardar el valor, se mantiene el valor que había anteriormente.

Para introducir el número de fotos, pulsamos 9_H. Acto seguido se nos preguntará el número de fotos a realizar; introducimos el valor deseado y confirmamos mediante #_C. en caso de que queramos abortar pulsaríamos #_H, y volveríamos al menú principal.

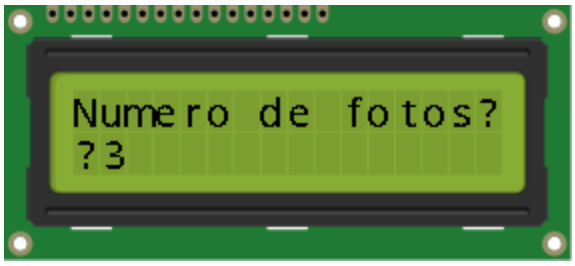

Figura II.26: *Pantalla configuración número de fotografías a tomar en modo A-nFotos.*

Ahora la pantalla mostrará la diferente información, si quisiéramos volver atrás pulsaríamos *_C, para confirmar y comenzar la pila pulsaríamos #_C.

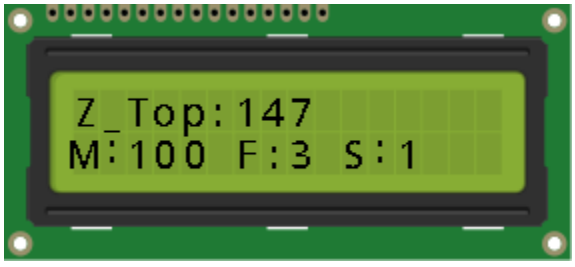

Figura II.27: *Pantalla de modo A-nFotos.*

Pantalla - Línea 1	
Z_Top: 147	Muestra la posición del punto A, en pasos.

Pantalla - Línea 2	
M:100	Muestra las micras a mover entre foto y foto.
F:3	Muestra las fotos a realizar
S:1	Muestra los micropasos

Una vez confirmado, El dispositivo comenzaría a tomar la pila. Durante la toma el dispositivo mantiene conectado el bloqueo de exposición de la cámara. Durante la toma también podemos:

***_C**	Abortar el proceso y volver al menú principal.
0_C	Pausa manteniendo el bloqueo de exposición, si se quiere eliminar el bloqueo pulsaríamos 8_C. Para continuar presionar 0_C.
8_C	Pausa sin mantener el bloqueo de exposición. Para continuar pulsar 0_C.

Una vez acabada la toma, nos mostrará el desplazamiento total en micras y nos pregunta que queremos hacer, las opciones disponibles son:

7_C	Mueve el motor y toma una foto adicional.
8_C	Vuelve a realizar la pila con los mismos parámetros.
9_C	Hace n fotos adicionales.
#_C	Volver al menú principal reseteando los valores de posición y estado de A y B, y la posición del motor Z.
0_C	Volver al menú principal pero sin resetear los valores anteriores.

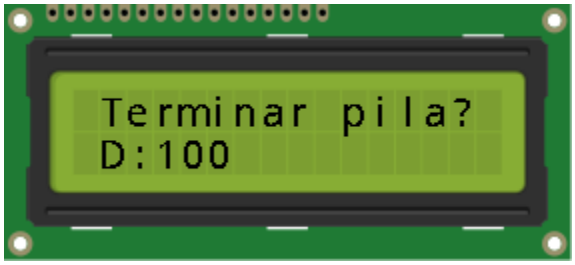

Figura II.28: *Pantalla de final de toma de pila.*

Diseño, construcción y programación de un sistema automático y autónomo para la adquisición de fotografías multifoco destinado a documentación científica

Anexo III

Stack_Duino: Diagrama, Componentes y Conexiones

1. DIAGRAMA

A continuación se muestran las instrucciones para el montaje de la caja de control Stack_Duino. La siguiente figura muestra el diagrama de los componentes y las conexiones entre ellos.

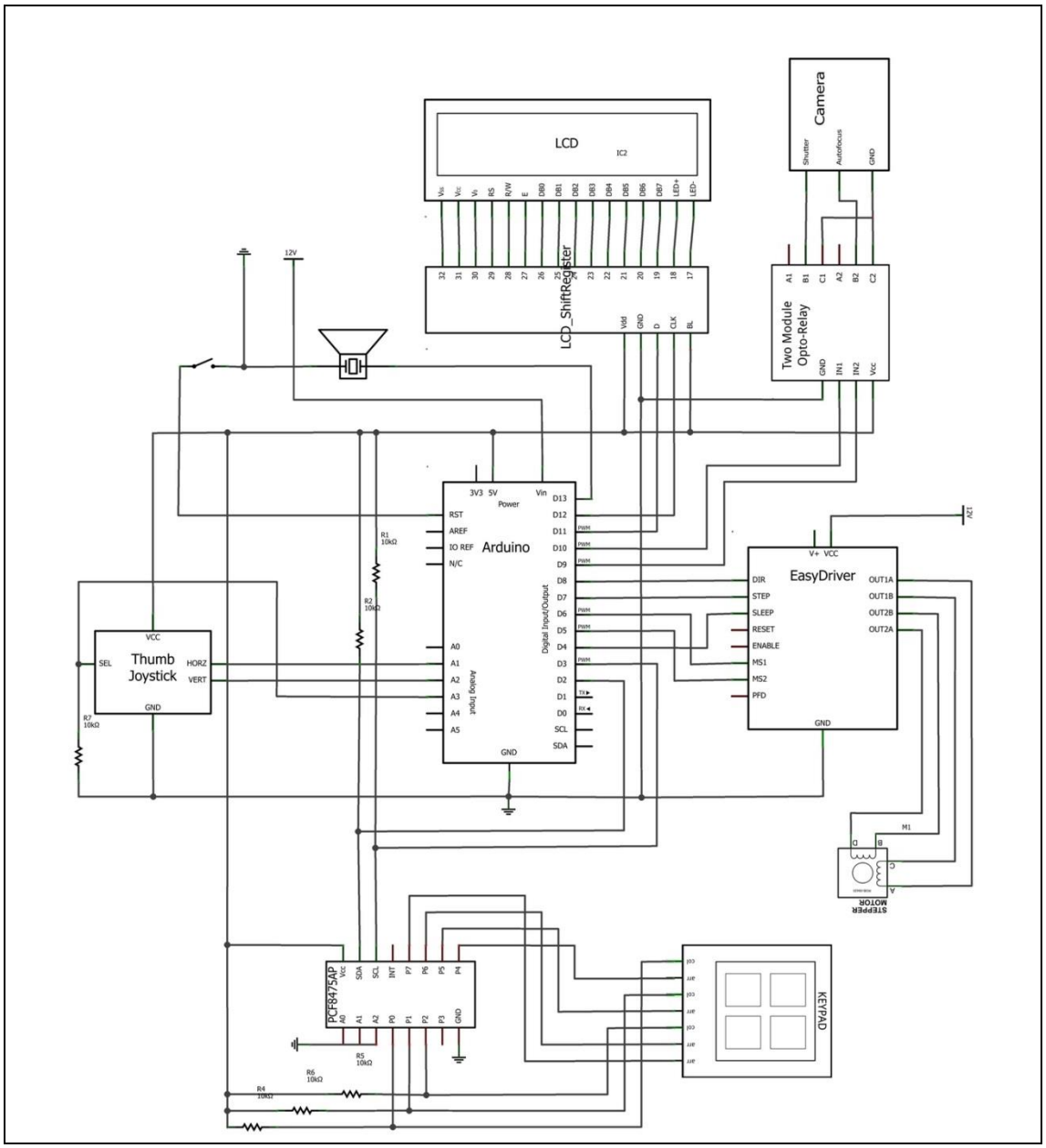

Figura III.1: *Diagrama de las conexiones.*

Seguidamente se describen cada uno de los componentes más detalladamente, las conexiones y donde poder obtenerlos.

2. COMPONENTES Y CONEXIONES

2.1. Placa Arduino Leonardo

Arduino dispone de diferentes modelos de placas, con diferentes características. Stack_Duino es compatible con cualquier placa que monte un chip ATmega32u4; este permite la comunicación USB con nuestro ordenador, facilitando que **Arduino** tome el

control, por ejemplo, de nuestro teclado. Esta función será necesaria para alguna de las funciones. En concreto, se ha elegido el modelo Arduino Leonardo, por su facilidad de adquisición, funciones disponible y bajo coste. Cuenta con 20 pines de entradas/salidas digitales, 12 entradas analógicas, un oscilador de cristal de 16 MHz, una conexión micro USB, un conector de alimentación, puertos ICSP e I2C, y un botón de reset.

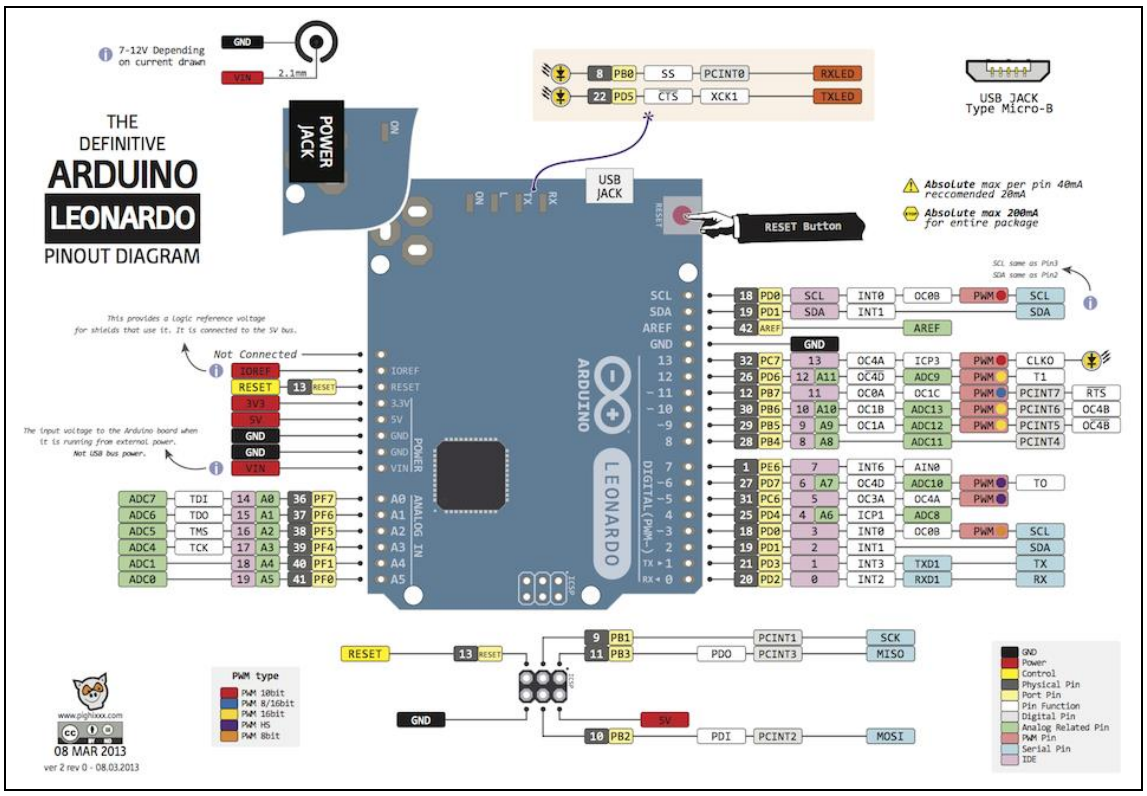

Figura III.2: *Diagrama pines de Arduino Leonardo. (CC BY-SA 3.0) www.pighixxx.com*

Más información: http://arduino.cc/en/Main/arduinoBoardLeonardo

Conexiones	
Vin	A la alimentación de 12V.
GND	A masa.

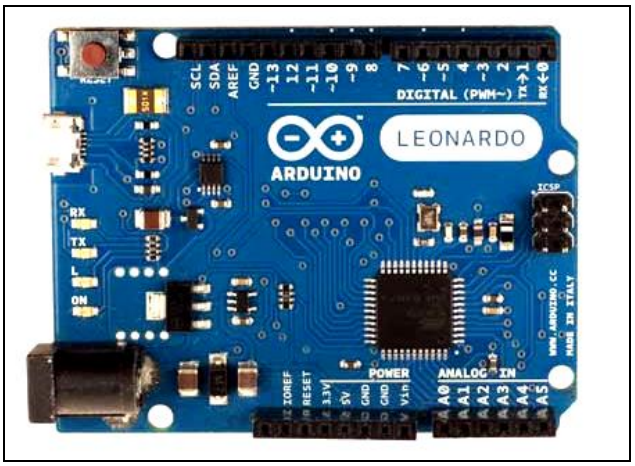

Figura III.3: *Placa Arduino Leonardo. (CC BY-SA 3.0) arduino.cc.*

2.2. Driver del motor. Easydriver

Easydriver es un controlador o driver de motores paso a paso (pap), compatible con cualquier dispositivo capaz de suministrar un pulso digital de 0V o 5V. Easydriver requiere una fuente de alimentación de 7V a 20V y proporciona al motor la tensión necesaria para su movimiento. Tiene integrado en la placa directamente un regulador de voltaje para el control de la parte lógica que puede ser ajustada a 5V o 3.3V. Conectándolo a un motor pap de 4 hilos y a un microcontrolador obtienes un controlador de motor muy preciso. Easydriver controla motores bipolares pap de 4, 6 u 8 hilos.

Dispone de microsteps (pines MS1 Y MS2) que permite más flexibilidad y control sobre el motor aumentando la resolución de éste. Los microsteps o micropasos en español, permiten aumentar el número de pasos del motor. Los valores disponibles para el integrado A3967 que monta Easydriver, son 1, 2, 4, 8. El motor permite un número de pasos fijo, el uso de los micropasos permite aumentar las posiciones que puede adoptar el motor por cada vuelta, y por tanto, aumentando el número de pasos por vuelta y la resolución del equipo. Si elegimos el valor de 2, se duplican los pasos; para valor de 4, se cuadriplica y así sucesivamente. Una ventaja de su uso, es una menor vibración del motor, pero para mayores valores hay una pérdida de reproducibilidad de los pasos.

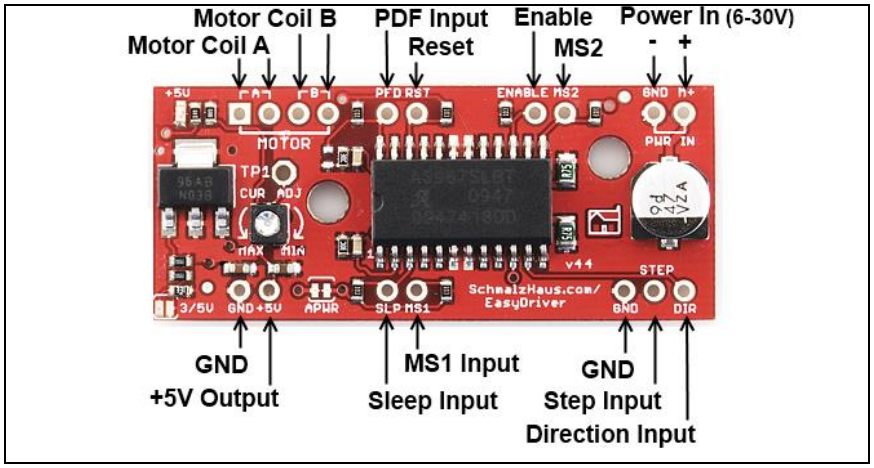

Figura III.4: *Easydriver V4.4. (CC BY-SA 3.0) arduino.cc.*

Más información: http://schmalzhaus.com/EasyDriver/

Conexiones	
M+	Conectado a la fuente de alimentación de 12V.
+5V	A 5V de Arduino, aunque no es necesaria esta conexión, si es conveniente.
GND	A GND, a masa.
DIR	A pin 8 de Arduino.
STEP	A pin 7 de Arduino.
MS1	A pin 6 de Arduino.
MS2	A pin 5 de Arduino.
SLP	A pin 4 de Arduino.

2.3. Teclado numérico 12 teclas

Un teclado matricial cuenta con filas y columnas, cuando se presiona un botón, se activa la salida correspondiente a la fila y la columna en la que se encuentra dicho botón, leyendo estos datos, Arduino puede saber que tecla ha sido el presionada. En la figura 4 se muestra el diagrama de conexionado para el teclado usado. Según el fabricante la disposición y correspondencia de las filas y columnas puede variar, por que previamente a su montaje, es conveniente su identificación de la correspondencia de los pines con las filas o columnas.

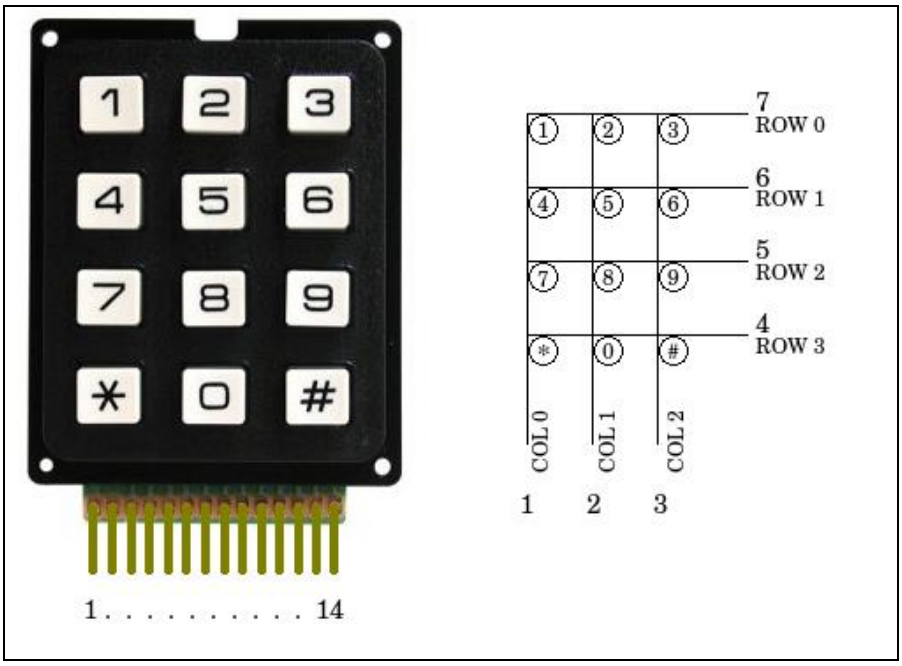

Figura III.5: *Teclado matricial 4x3, 12 teclas y diagrama. (CC BY-SA 3.0) arduino.cc.*

Más información: http://playground.arduino.cc/Main/KeypadTutorial

Conexiones	
Col0 (columna 1 del teclado)	P0 del PCF8574AP
Col1 (columna 2 del teclado)	P1 del PCF8574AP
Col2 (columna 3 del teclado)	P2 del PCF8574AP
Row0 (fila 1 del teclado)	P4 del PCF8574AP
Row1 (fila 2 del teclado)	P5 del PCF8574AP
Row2 (fila 3 del teclado)	P6 del PCF8574AP
Row3 (fila 4 del teclado)	P7 del PCF8574AP

2.4. LCD 16x2 compatible con Hitachi HD44780

El LCD (Liquid Crystal Dysplay) o pantalla de cristal líquido es un dispositivo empleado para la visualización de contenidos o información de una forma gráfica, mediante caracteres, símbolos o pequeños dibujos dependiendo del modelo. Está gobernado por el microcontrolador de Arduino, el cual dirige todo su funcionamiento. Permite mostrar datos en un proceso de monitoreo y control. Su interfaz con los

controladores se realiza a través de un conector de 14 pines que se unen al ShfitRegistered mediante una tira de pines.

En este caso vamos a emplear un LCD de 16x2, dispone de 2 filas de 16 caracteres cada una. Los píxeles de cada símbolo o carácter, varían en función de cada modelo. Se ha optado por esta opción dada su amplia disponibilidad y bajo coste.

Mediante una tira de pines se suelda al ShiftRegister LCD según el diagrama de la figura III.1.

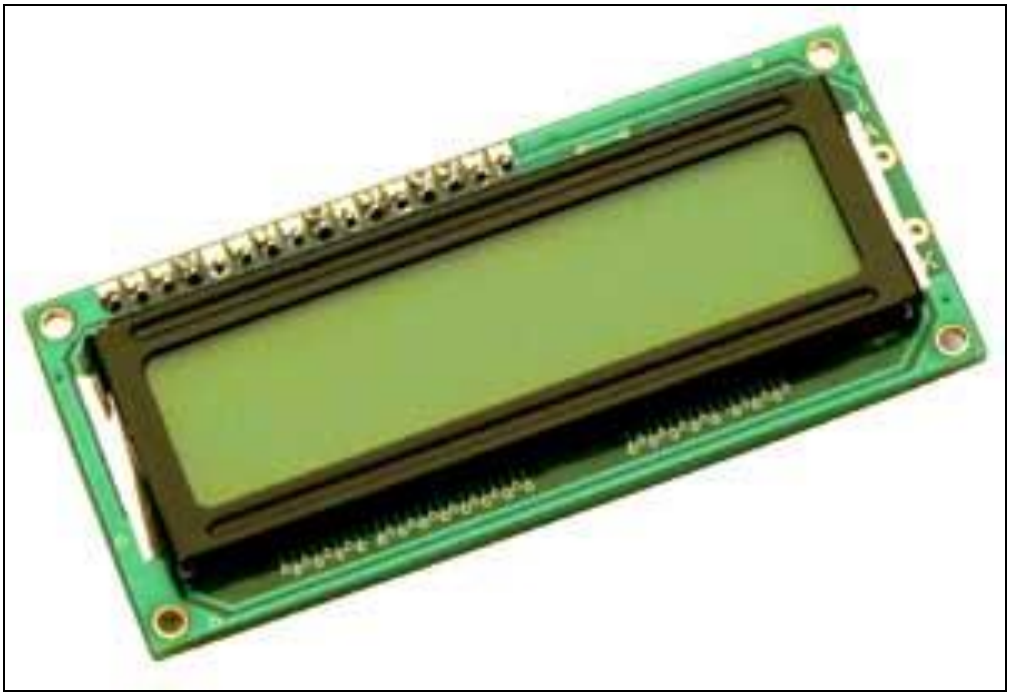

Figura III.6: *LCD 16x2 caracteres. (CC BY-SA 3.0) arduino.cc.*

2.5. ShiftRegister LCD (SR LCD)

El LCD SR se utiliza con una pantalla LCD, permitiendo reducir el número de pines necesarios para controlarla y amentando notablemente la velocidad de funcionamiento de éste. Para hacer funcionar este módulo y el LCD necesitamos una librería específica, que debemos descargarla y copiarla en la carpeta "libraries" del IDE de Arduino y sustituir la que trae por defecto. Descarga librería ver anexo II. Dispone además de un potenciómetro mediante el que ajustamos el contraste de la pantalla.

Conexiones	
La parte superior se une al LCD mediante una tira de pines soldada	
Vdd	A 5V de la placa Arduino
GND	A masa
D	Al pin 11 de Arduino
CLK	Al pin 12 de Arduino
BL	A 5V de la placa Arduino
Mediante una tira de 16 pines al LCD	

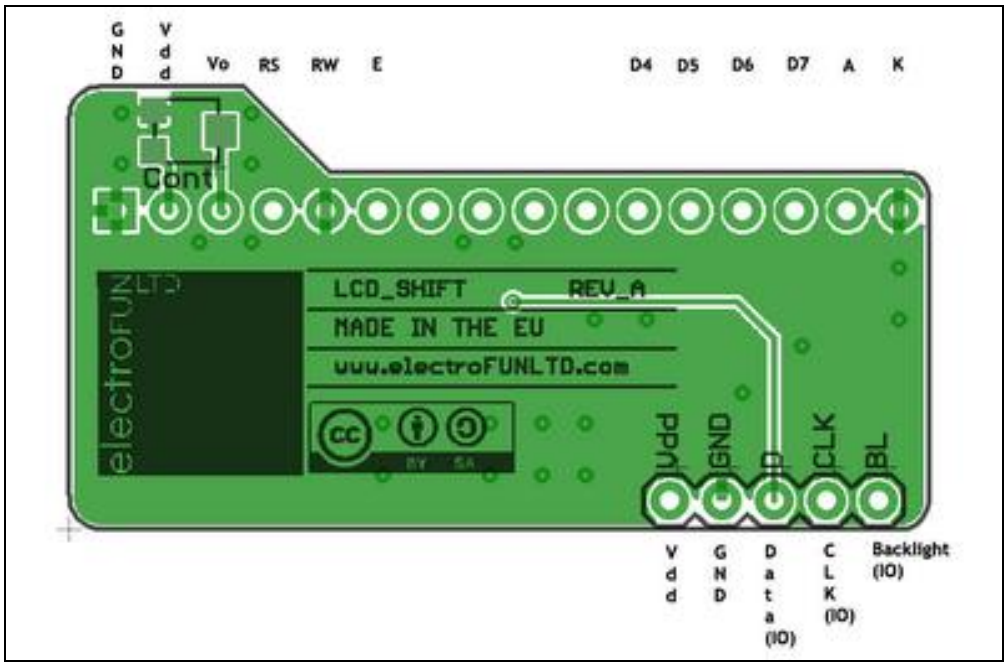

Figura III.7: *ShiftRegistered LCD. (CC BY-SA 3.0) F. Malpartida.*

Más información sobre el modulo SR LCD:
http://www.electrofunltd.com/p/shiftregister-lcd
https://bitbucket.org/fmalpartida/lcd-shiftregister/wiki/Home

Diagramas:
http://code.google.com/p/arduinoshiftreglcd/

2.6. Altavoz Piezoeléctrico

Permite la reproducción de sonidos de aviso.

Figura III.8: *Altavoz piezoeléctrico. (CC BY-SA 3.0) arduino.cc.*

Conexiones	
Los dos pines son equivalentes	
Pin1 del piezo	Al pin 13 de Arduino
Pin2 del piezo	A GND, a masa

2.7. Botón Momentario

Se une al pin reset de Arduino, permite el reseteo de la placa en caso de que haya problemas.

Figura III.9: *Botón momentario. (CC BY-SA 3.0) arduino.cc.*

Conexiones	
Los dos pines son equivalentes	
Pin1 del botón	Al pin 13 de Arduino
Pin2 del botón	A GND, a masa

2.8. Módulo de dos relés con optoacoplador

Permite el disparo de la cámara de forma mecánica y al mismo tiempo, aísla a ésta del circuito principal, para evitar dañarla.

Figura III.10: *Módulo de dos relés con optoacoplador. (CC BY-SA 3.0) arduino.cc.*

Conexiones	
GND	A masa.
IN1	Al pin 5 de Arduino.
IN2	Al pin 4 de Arduino.
Vcc	A 5V de la placa Arduino.

Salidas	
A1	Cable del autofoco de la cámara.
A3	Cable de masa de la cámara.
B1	Cable del disparador de la cámara.
B3	Cable de masa de la cámara.

2.9. Cable control de la cámara

El cable y los diferentes pines dependerán de la marca y modelo de la cámara, en la referencia se pueden encontrar los datos necesarios para cada tipo de cámara. Como ejemplo ponemos el de una Canon Eos 40D.

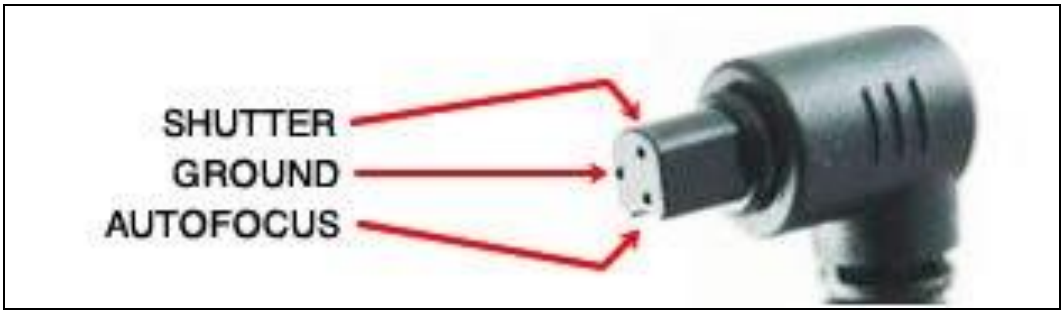

Figura III.11: *Cable control para DSLR Canon EOS XX. (CC BY-SA 3.0) photoduino.com*

Conexiones	
Shutter	B2 del módulo de relés.
Ground	A3 y B3 del módulo de relés.
Autofocus	A2.

Referencia:
http://photoduino.com/es/documentacion/referencia/conectores-control-remoto-de-camaras/

2.10. Joystick

El joystick consiste en dos potenciómetros que nos permiten medir el movimiento en 2D. Los potenciómetros son resistencias variables y, por lo tanto, actúan como sensores proporcionándonos una tensión variable dependiendo de la rotación del dispositivo sobre su eje, que Arduino es capaz de leer, a través de las entradas analógicas. Además, presionándolo, actúa como un botón.

Figura III.12: *Joystick. (CC BY-SA 3.0) arduino.cc.*

Concesiones	
GND	A GND, A masa
X	Al pin A1 de Arduino
Y	Al pin A2 de Arduino
B	Al pin A3 de Arduino
+	A 5V de la placa Arduino

2.11. PCF8574AP. Expansor Remoto de puerto 8-bit I/O por I^2C-bus

Permite la comunicación mediante el protocolo I2C del teclado con la placa Arduino. I2C es un bus de comunicaciones en serie, únicamente se necesitan dos líneas, la de datos (SDA) y la del reloj (SCL), cada dispositivo conectado al bus tiene un código de dirección seleccionable, lo que permite conectar varios dispositivos en serie a Arduino únicamente mediante las dos entradas SDA y SCL y especificando la dirección de cada uno.

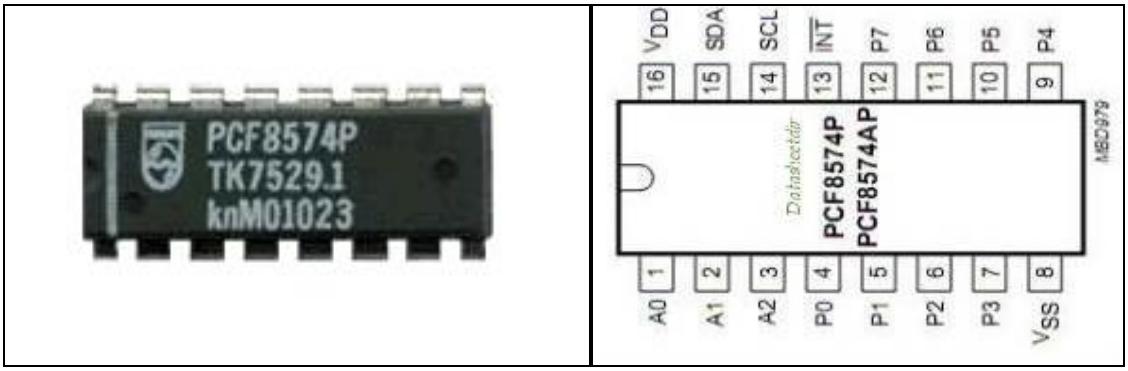

Figura III.13: *Chip PCF8574. (CC BY-SA 3.0) arduino.cc.*

En Stack_Duino la conexión al bus se realiza a través del chip PCF8574, es un expansor de E/S compatible con la mayoría de microcontroladores, permite una comunicación bidireccional, necesitando para ello solo dos líneas a través del bus I2C. El microcontrolador es está configurado como maestro y el módulo PCF8574 conectado

al bus se configuran como esclavo. A continuación detallamos como se debe conectar al resto de componentes. Podremos adquirir el chip suelto y construir nosotros el circuito según el diagrama anterior, o bien, recientemente ha aparecido en el mercado el circuito ya montado, que aun siendo más con un coste bastante superior, es aconsejable por la facilidad de montaje, pequeño tamaño y el ahorro de tiempo en su montaje.

Si optamos por montar nosotros mismos el circuito, el diagrama y las conexiones son las que se expresan en la Fig. III.14.

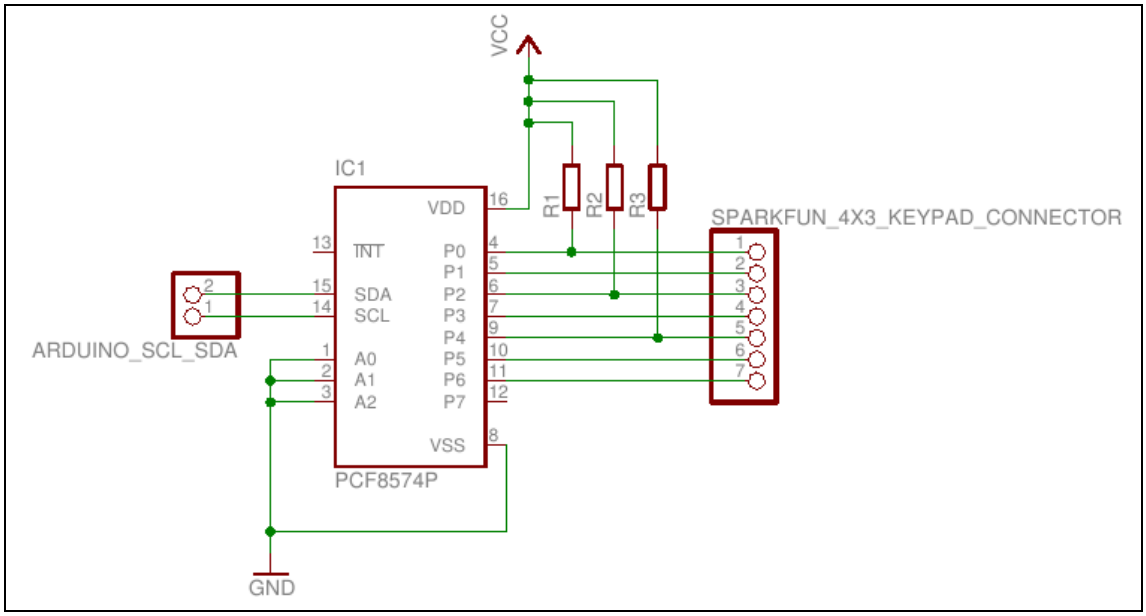

Figura III.14: *Diagrama conexiones teclado. (CC BY-SA 3.0) arduino.cc.*

Conexiones	
A0	A GND, a masa.
A1	A GND, a masa.
A2	A GND, a masa.
Vss	A GND, a masa.
Vdd	A 5V de la placa Arduino.
SDA	Al pin D2 de Arduino, y a 5V de la placa Arduino junto con una resistencia de 10K ohm, ver diagrama.
SCL	Al pin D3 de Arduino, y a 5V de la placa Arduino junto con una resistencia de 10K ohm, ver diagrama.
P0	Col1, columna 1 del teclado, y a 5V de la placa Arduino junto con una resistencia de 10K ohm, ver diagrama.
P1	Col2, columna 2 del teclado, y a 5V de la placa Arduino junto con una resistencia de 10K ohm, ver diagrama.
P2	Col3, columna 3 del teclado, y a 5V de la placa Arduino junto con una resistencia de 10K ohm, ver diagrama.
P4	Row1, fila 1 del teclado.
P5	Row2, fila 2 del teclado.
P6	Row3, fila 3 del teclado.
P7	Row4, fila 4 del teclado.

En caso de optar por el circuito ya montado, las conexiones serán las siguientes:

Conexiones	
Vss	A GND, a masa.
Vdd	A 5V de la placa Arduino.
SDA	Al pin D2 de Arduino.
SCL	Al pin D3 de Arduino.
P0	Col1, columna 1 del teclado.
P1	Col2, columna 2 del teclado.
P2	Col3, columna 3 del teclado.
P4	Row1, fila 1 del teclado.
P5	Row2, fila 2 del teclado.
P6	Row3, fila 3 del teclado.
P7	Row4, fila 4 del teclado.

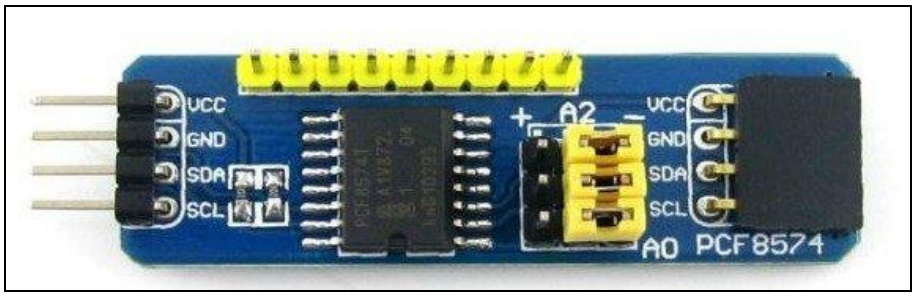

Figura III.15: *Expansor PCF8574. (CC BY-SA 3.0) arduino.cc.*

2.12. Motor paso a paso bipolar 12V y 0,33A

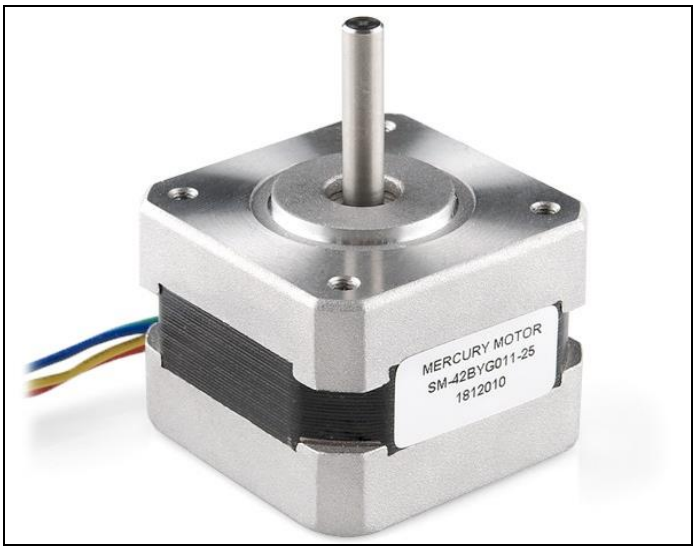

Figura III.16: *Motor paso a paso. (CC BY-SA 3.0) arduino.cc.*

El motor paso a paso es un dispositivo electromecánico que convierte una serie de impulsos eléctricos en desplazamientos angulares discretos, lo que significa que es

capaz de avanzar una serie de grados (paso) dependiendo de sus entradas de control. El motor paso a paso se comporta de la misma manera que un conversor digital-analógico (D/A) y puede ser gobernado por impulsos procedentes de sistemas lógicos, como Easydriver.

Puede servir cualquier motor paso a paso bipolar de hasta 35V y 0,75 amperios por fase como máximo. En caso de usar un motor de mayor amperaje, debemos usar un driver del motor distinto, como por ejemplo el Big-EasyDriver.

Conexiones	
A1	pin A de EasyDriver.
A2	pin A de EasyDriver.
B1	Pin B de EasyDriver.
B2	Pin B de EasyDriver

2.13. Placa de prototipado

Aunque no es necesaria, sí que permite un montaje más sencillo, ya que sirve como placa para soldar algunos de los elementos, así como para poder hacer la conexión de los cables. En el mercado existen diferentes modelos, según el modelo de la placa y las conexiones que ofrece, también nosotros podemos construir una con una placa de prototipado a la que le hemos soldado los pines necesarios para la conexión de los cables.

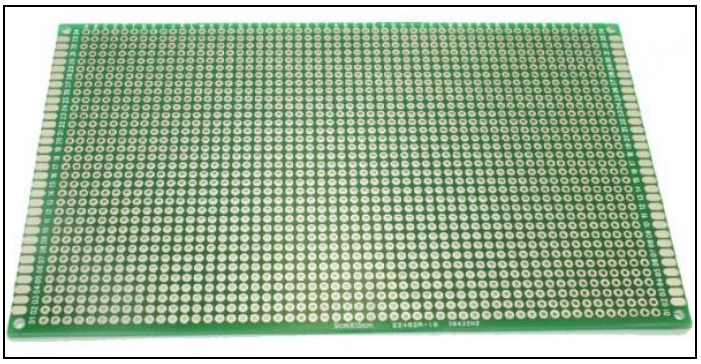

Figura III.17: *Placa de prototipado. (CC BY-SA 3.0) arduino.cc.*

2.14. Tira de pines

Únicamente son necesarios si optamos por montar nosotros la placa de prototipado. Existen varios modelos, rectos o en ángulo.

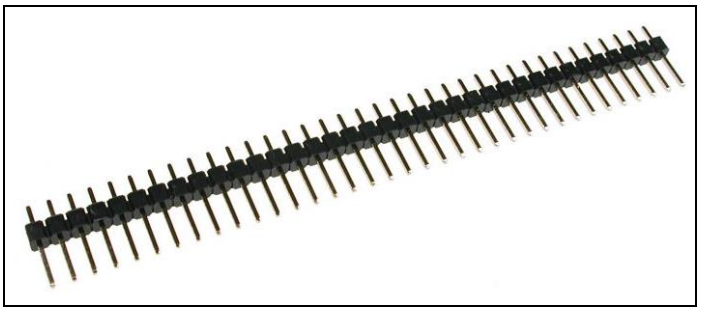

Figura III.18: *Tira de pines. (CC BY-SA 3.0) arduino.cc.*

2.15. Resistencias 10K Ohm 0,25A

Necesarias para conectar algunos pines a masa.

Figura III.19: *Resistencia 10k ohmios. (CC BY-SA 3.0) arduino.cc.*

2.16. Caja y alimentación

Se ha optado por montar todos los componentes en una caja de un disco duro ya que ésta lleva la alimentación ya acoplada. La alimentación debe ser de 12V y 2A si se usan los componentes antes descritos.

Figura III.20: *Caja externa disco duro. (CC BY-SA 3.0) arduino.cc.*

2.17. Cables Dupont

Permite una fácil conexión entre los diferentes módulos. Están disponibles hembra-macho y hembra-hembra, así como en diferentes longitudes.

Figura III.21: *Cables dupont hembra-hembra. (CC BY-SA 3.0) arduino.cc.*

3. COMPRA DE LOS COMPONENTES

A continuación daremos diferentes links de tiendas virtuales donde poder adquirir los diferentes componentes y su precio aproximado. El mejor lugar para adquirirlos es www.ebay.com, por precio y facilidad de adquisición, dada la temporalidad de los links en este sitio, se dará un link a búsquedas por palabras claves adecuadas.
También se darán otras direcciones alternativas, como Sparkfun u otras tiendas virtuales.

3.1. Placa Arduino Leonardo

Precio aproximado 10 euros.

- http://www.ebay.com/sch/i.html?_sacat=0&_from=R40&_nkw=arduino+leonardo+board&_sop=12
- https://www.sparkfun.com/products/11286
- http://www.dx.com/es/p/leonardo-r3-atmega32u4-module-blue-black-works-with-official-arduino-boards-261685#.U1lplVfT3xU

3.2. Driver del motor. Easydriver

Precio aproximado 10 euros.

- http://www.ebay.com/sch/i.html?_odkw=arduino+leonardo+board&_sop=12&_osacat=0&_from=R40&_trksid=p2045573.m570.l1313.TR4.TRC2.A0.H0.Xeasydriver&_nkw=easydriver&_sacat=0
- http://www.goodluckbuy.com/a3967-easydriver-drive-driver-board-for-stepper-motor.html
- http://www.dx.com/es/p/easydriver-v4-4-stepper-motor-driver-board-for-arduino-red-161537
- https://www.sparkfun.com/products/10267

3.3. Teclado numérico 12 teclas

Precio aproximado 4 euros.

- https://www.sparkfun.com/products/8653
- http://www.dx.com/es/p/diy-3-x-4-12-key-numeric-keypad-black-150134#.U1lraVfT3xU

3.4. LCD 16x2 compatible con Hitachi HD44780

Precio aproximado 3 euros.

- http://www.ebay.com/sch/i.html?_odkw=arduino+lcd&LH_BIN=1&_sop=15&_osacat=0&_from=R40&_trksid=p2045573.m570.l1313.TR2.TRC1.A0.H0.Xarduino+lcd+16x2&_nkw=arduino+lcd+16x2&_sacat=0
- https://www.sparkfun.com/products/709
- http://www.dx.com/es/p/16-x-2-character-lcd-display-module-with-blue-backlight-121356

3.5. ShiftRegister LCD (SR LCD)

Precio aproximado 7 euros.

- http://www.electrofunltd.com/p/buy.html

3.6. Altavoz Piezoeléctrico

Precio aproximado 2 euros.

- http://www.ebay.com/sch/i.html?_odkw=piezo&LH_BIN=1&_sop=15&_osacat=0&_from=R40&_trksid=p2045573.m570.l1313&_nkw=piezo+buzzer&_sacat=0
- https://www.sparkfun.com/products/7950
- http://www.dx.com/es/p/13mm-x-2mm-piezo-buzzers-black-dc-9v-5-pack-141433#.U1lu11fT3xU

3.7. Botón Momentario

Precio aproximado 1 euro.

- http://www.ebay.com/sch/i.html?_odkw=piezo+buzzer&LH_BIN=1&_sop=15&_osacat=0&_from=R40&_trksid=p2045573.m570.l1313.TR12.TRC2.A0.H0.Xmomentary+button&_nkw=momentary+button&_sacat=0
- https://www.sparkfun.com/products/11992
- http://www.dx.com/es/p/plastic-copper-push-button-power-switches-red-10-pcs-153209#.U1l0HFfT3xU

3.8. Módulo de dos relés con optoacoplador

Precio aproximado 3 euros.

- http://www.ebay.com/sch/i.html?_odkw=relay+opto&LH_BIN=1&_sop=15&_osacat=0&_from=R40&_trksid=m570.l3201&_nkw=relay+opto+2&_sacat=0
- http://www.dx.com/es/p/produino-diy-pc817-2-channel-5v-relay-module-w-optocoupler-extension-board-blue-290132#.U1l1U1fT3xU

3.9. Cable control de la cámara

El cable a usar dependerá de la marca de nuestra cámara, en el siguiente link, podemos encontrar información específica sobre cada modelo y marca.

- http://photoduino.com/es/documentacion/referencia/conectores-control-remoto-de-camaras/

3.10. Joystick

Precio aproximado 3 euros.

- http://www.ebay.com/sch/i.html?LH_BIN=1&_sop=15&_from=R40&_sacat=0&_nkw=joystick+arduino&_nkwusc=joystick+arduinio&_rdc=1
- http://www.dx.com/es/p/cg05sz-050-joystick-module-black-266570#.U1l29lfT3xU

3.11. PCF8574AP. Expansor Remoto de puerto 8-bit I/O por I^2C-bus

Precio aproximado 2 euros chip suelto, 7 euros circuito montado.

- http://www.ebay.com/sch/i.html?_odkw=joystick+arduino&LH_BIN=1&_sop=15&_osacat=0&_from=R40&_trksid=p2045573.m570.l1313.TR0.TRC0.H0.XPCF8574AP&_nkw=PCF8574AP&_sacat=0
- http://www.ebay.com/sch/i.html?_odkw=PCF8574&LH_BIN=1&_sop=15&_osacat=0&_from=R40&_trksid=p2045573.m570.l1313.TR0.TRC0.H0.XPCF8574+expansion&_nkw=PCF8574+expansion&_sacat=0

- http://www.dx.com/es/p/pcf8574-io-expansion-board-blue-151551#.U1l4BVfT3xU

3.12. Motor paso a paso bipolar 12V y 0,33A

- http://www.ebay.com/sch/i.html?_odkw=stepper+motor+12v&LH_BIN=1&_sop=15&_osacat=0&_from=R40&_trksid=p2045573.m570.l1313.TR0.TRC0.H0.Xstepper+motor+12v+nema17&_nkw=stepper+motor+12v+nema17&_sacat=0
- https://www.sparkfun.com/products/9238

3.13. Tira de pines

Precio aproximado 1 euro.

- http://www.ebay.com/sch/i.html?LH_BIN=1&_from=R40&_sacat=0&_nkw=pin+header&_sop=12
- http://www.dx.com/es/s/pin+header

3.14. Cables Dupont

Precio aproximado 2,5 euros.

- http://www.ebay.com/sch/i.html?_odkw=pin+header&LH_BIN=1&_sop=12&_osacat=0&_from=R40&_trksid=p2045573.m570.l1313.TR8.TRC1.A0.H0.Xdupont+cable&_nkw=dupont+cable&_sacat=0
- http://www.dx.com/es/p/pvc-male-to-female-arduino-dupont-cables-multicolored-30cm-307052#.U1l7oFfT3xU

3.15. Caja y alimentación

Se ha optado por el uso de una caja de un disco duro externo. Debemos seleccionar una que permita que quepan todos los componentes, que sea fácilmente accesible para su montaje y lleve alimentación acoplada de 12V y 2A. También es posible fabricarnos nuestra propia caja con cualquier otro material, acoplándole la alimentación necesaria. El coste de la caja, tornillería y pintura, ronda los 40 euros.

Diseño, construcción y programación de un sistema automático y autónomo para la adquisición de fotografías multifoco destinado a documentación científica

Anexo IV

Código Stack_Duino 1.00

CÓDIGO STACK_DUINO 1.00

```
/*
Stack_Duino es un sistema automatizado y autónomo para la toma de pilas de
imágenes.
Stack_Duino dispone de dos modos de toma de imágenes; en el primero se marcan el
punto de inicio (A), el punto final (B) y la distancia entre fotos, calculando el software
el número de fotos a realizar; y un segundo método, en el que se marca el punto de
inicio (A), el intervalo entre fotos y el número de fotos a realizar a partir de (A).
Dispone además de un menú de configuración que le permite ser adaptado fácilmente a
cualquier sistema, permite configurar micropasos del driver, pasos del motor, recorrido
por vuelta del piñón de enfoque, sentido de giro del motor, tiempo de espera tras
movimiento del motor y tiempo de espera tras realizar la foto.
(cc) Honorio Cócera 2014
Este programa es software libre, usted puede redistribuirlo o modificarlo bajo la licencia
CC Reconocimiento-NoComercial-CompartirIgual 3.0 España (CC BY-NC-SA 3.0 ES)
http://creativecommons.org/licenses/by-nc-sa/3.0/es/
Este programa es distribuido con la intención de su utilidad, pero sin ninguna garantia.
*/

#include <EEPROM.h>
#include <LiquidCrystal_SR.h>
#include <Wire.h>

//************************************************************************
// definición del LCD, cambia en funcion del tipo de LCD
// descarga libreria https://bitbucket.org/fmalpartida/new-liquidcrystal/wiki/Home

LiquidCrystal_SR lcd(11,12,TWO_WIRE);
//                   | |
//                   | \-- Clock Pin
//                   \---- Data/Enable Pin

const int numRows = 2;
const int numCols = 16;

//************************************************************************
// Variables:

int     valx=0;  // valor del potenciometro-joystic eje x
int     valy=0;  // valor del potenciometro-joystic eje y
int     pasos=0; // pasos a mover el motor
long    z=0;     // cuenta los pasos, la posición en el eje z
unsigned long time=0;  // para calcular el tiempo
boolean clickbutton=0;// valor para el boton del joistick
long    ztop=10;  //marca el punto más alto de la pila
long    zbotton=1; //marca el punto más bajo de la pila

byte    microsteps=1;// microsteps por defecto, eeprom dir 4
int     num_pasos;  // numero de pasos por defecto a mover entre foto y foto,
```

```
int     num_pasos_vuelta=200;  // numero de pasos por vuelta, motor y reductor eepron dir 5 y 6
int     micras_vuelta=200;  // recorrido del piñon de enfoque por vuelta en micras*10
int     velocidad_defecto=100 ; // velocidad minima
unsigned long     tiempo_exposicion=6000;// tiempo necesario para la toma de foto
unsigned long     tiempo_vibracion=4000; // tiempo para eliminar la vibracion tras mover el motor
boolean sentido_giro=1;  // sentido giro del motor, eeprom dir 18 y 19
boolean estado_ztop=false; // comprobar si se ha marcado el punto ztop
boolean estado_zbotton=false;  // comprobar si se ha marcado zbotton
unsigned int     micras=10;  // micras a mover  , memoria 14 y 15
byte    key=0;
boolean confirmar=false;//  para salir sin guardar valor
int     valx_referencia;
int     valy_referencia;
int     valx_map; // para calcular el mapeado de xPin
int     valy_map; // para calcular el mapeado de yPin
int     error_valor=0;
byte    reply_fotos=1; // numero de fotos a repetir por toma a mismo foco
int     num_fotos; // numero de fotos para hacer la pila     pasos/ pasos_a_mover(calculado a partir de micras a mover)
int     num_fotos_restantes;//  numero de fotos que quedan por hacer
int     num_fotos_desplazamiento=0;// para calcular el desplazamiento
int     contador_fotos; // cuenta las fotos que se van haciendo
boolean estado_menu=false;  // para que vuelva  a imprimir cuando volvemos al menu
boolean estado_motor=false;//  conecta/ desconecta motor
boolean temp_estado_motor;
boolean     sonido=1; // estado del sonido, 1 encendido, 0 apagado, direccion de eeprom 20 y 22
byte    camara=0; // define el tipo de camara, 0 para reflex, >0 para camaras controladas por software, dir eeporm 26 y 27
byte    secuencia=1;// para saber el menu a mostrar
char    tecla; // asigna la tecla a mandar al pc para disparar simulando un keyboard

//*****************************************************************************
// direccion I2C del teclado y variables
int row,column;
#define expander B0100000

//*****************************************************************************
// direcciones memoria eeprom
#define eeprom_dir_microsteps        4 //  microspteps
#define eeprom_dir_num_pasos         2 //  num_pasos
#define eeprom_dir_num_pasos_vuelta   5 // num_pasos_vuelta
#define eeprom_dir_tiempo_exposicion  7 // tiempo_exposicion, tiempo1
#define eeprom_dir_tiempo_vivbracion  9 // tiempo_vivracion, tiempo2
#define eeprom_dir_sentido_giro      18 // sentido_giro
#define eeprom_dir_micras_vuelta     12 // micras_vuelta
#define eeprom_dir_micras           14 // micras
#define eeprom_dir_version          16 // direccion de eeprom para la version del programa
```

```
#define eeprom_dir_sonido          20// direccion de eeprom para el estado del sonido
#define eeprom_dir_velocidad_defecto 22// direccion eeprom velocidad defecto,
minimo valor 25
#define eeprom_dir_reply_fotos      24 // numero replicas por foto y foco
#define eeprom_dir_camara          26 // tipo de camara
#define version_programa          100//
#define EEPROM_SIZE               512 // tamaño eeprom para arduino demilanove

//*********************************************************************
#define velocidad_defecto_min   25  //
#define multiplicador_vel   2  // multiplica el valor de velocidad_defecto en algunas
funciones
                    // para aumentar la velocidad y evitar vibraciones

//*********************************************************************
// valor telas
#define no_key   10  //
#define key01_C  11  // tecla 1 cliclk
#define key02_C  12  // tecla 2 click
#define key03_C  13  // tecla 3 click
#define key04_C  14  // tecla 4 click
#define key05_C  15  // tecla 5 click
#define key06_C  16  // tecla 6 click
#define key07_C  17  // tecla 7 click
#define key08_C  18  // tecla 8 click
#define key09_C  19  // tecla 9 click
#define key10_C  20  // tecla * click
#define key11_C  21  // tecla 0 click
#define key12_C  22  // tecla # click
#define key01_H  31  // tecla 1 hold
#define key02_H  32  // tecla 2 hold
#define key03_H  33  // tecla 3 hold
#define key04_H  34  // tecla 4 hold
#define key05_H  35  // tecla 5 hold
#define key06_H  36  // tecla 6 hold
#define key07_H  37  // tecla 7 hold
#define key08_H  38  // tecla 8 hold
#define key09_H  39  // tecla 9 hold
#define key10_H  40  // tecla * hold
#define key11_H  41  // tecla 0 hold
#define key12_H  42  // tecla # hold

//*********************************************************************
// Definiciones para algunos tiempos
#define debounce       15 // tiempo de espera para evitar rebote del boton
#define AF_TIME        200 //ms
#define SHUTTER_LAG   200 //ms
#define PAUSE_TIME   1000 //ms
#define click_time     60 //ms, tiempo para eliminar repeticiones, debounce
#define hold_time     600 //ms, tiempo a esperar para mantener pulsado el boton
#define stop_time     800
```

```
//*********************************************************************
// Definiciones para los pines
#define shutterPin      10  // digital pin
#define autofocusPin   9   // digital pin
#define sleepPin         4   // conecta-desconecta la rotación, pin digital
#define dirPin            8   // pin que controla la direccion de giro, pin digital
#define pasosPin         7   // pin pasos de la tarjeta, pin digital,   steps
#define ms1Pin           6   // ms1 pin para microsteps, pin digital
#define ms2Pin           5   // ms2 pin para microsteps, pin digital
#define xPin             1   //pin analogico del joistic eje x
#define yPin             2   //pin analogico del joistic eje y
#define clickPin        A3  //pin boton digital del joistic
#define buzzerPin      13  //pin altavoz

//*********************************************************************
 void setup() {

 // inicializa la comunicación I2C
 Wire.begin();

 // Muestra comienzo en pantalla
 lcd.begin(numRows,numCols);
 lcd.print("  STACK_DUINO ");
 lcd.setCursor(0,1);
 lcd.print("  PRESS A KEY   ");

 // Declaracion de pines
 pinMode(buzzerPin, OUTPUT);
 pinMode(sleepPin, OUTPUT); digitalWrite( sleepPin, LOW ); // desconecta el motor
para que gire libremente
                                        // el piñon de enfoque
 pinMode(dirPin, OUTPUT); digitalWrite(dirPin,LOW);
 pinMode(ms1Pin, OUTPUT);
 pinMode(ms2Pin, OUTPUT);
 pinMode(pasosPin, OUTPUT); digitalWrite(pasosPin,LOW);
 pinMode(clickPin, INPUT);
 pinMode(shutterPin,OUTPUT); digitalWrite(shutterPin,HIGH);
 pinMode(autofocusPin,OUTPUT); digitalWrite(autofocusPin,HIGH);

 cargar_eeprom(); // comprueba la version que hay guardada en eeprom
                     //si no es la misma carga los valores anteriores de eeprom

 //valores de referencia para yoistick, calcular 0.
 valx_referencia= analogread_repetido(xPin,5);
 delayMicroseconds(200);
 valy_referencia= analogread_repetido(yPin,5);
 delayMicroseconds(200);

 // activar pines microsteps
 if (microsteps==1) {digitalWrite(ms1Pin,LOW) ;digitalWrite(ms2Pin,LOW);}
```

```
 if (microsteps==2) {digitalWrite(ms1Pin,HIGH);digitalWrite(ms2Pin,LOW);}
 if (microsteps==4) {digitalWrite(ms1Pin,LOW) ;digitalWrite(ms2Pin,HIGH);}
 if (microsteps==8) {digitalWrite(ms1Pin,HIGH);digitalWrite(ms2Pin,HIGH);}

 // activar sentido giro
 if (sentido_giro==1) {sentido_giro=1;}
 if (sentido_giro==0) {sentido_giro=0;}

 // activar camara controlada a traves del pc y asignar la tecla a mandar al pc
 if (camara !=0) asignar_tecla();

 //espera a presionar una tecla para comenzar
 while ( 1 ) { pulsaciones_teclado(); if (key>no_key) break;}

 // Mostrar menú principal en pantalla
 lcd.clear();
 lcd.setCursor(0,1);
 lcd.print("Z:"); lcd.setCursor(2,1); lcd.print(z);
 lcd.setCursor(0,0);lcd.print("A:"); lcd.print(int(estado_ztop)); lcd.print("
B:"),lcd.print(int(estado_zbotton));

 // imprimir micras en pantalla
 lcd.setCursor(9,0), lcd.print("      "); lcd.setCursor(9,0); lcd.print(micras);

 // imprimir microspets en pantalla
 lcd.setCursor(15,0), lcd.print(" "); lcd.setCursor(15,0); lcd.print(microsteps);

 // imprimir en pantalla estado del motor
 lcd.setCursor(15,1);lcd.print(" ");lcd.setCursor(15,1);if (estado_motor == 0
)lcd.print("0");
 if (estado_motor == 1 )lcd.print("1");
 } // final de set-up

//*****************************************************************************
 void loop() {

 // Leer teclado
 delayMicroseconds(10);  // tiempo entre lectura de dos entradas analogicas
 menu_teclado();

 //movimiento joisitck eje y
 delayMicroseconds(50);

 valy=analogRead (yPin);
 valy= valy - valy_referencia;
 valy_map= abs(valy);
 valy_map= map(valy_map, 0, (valy_referencia-30), 0, 5);
 if ( valy_map != 0 ) {
                    while ( valy_map != 0 )
                    {
```

```
          //joistick colocado del reves, si se coloca de otra bien hay que cambiar los
valores de valy<0 y valx>0
                  if (valy <= 0) { motor( -1,velocidad_defecto*valy_map); }// Mover el
motor con el joistick
                  if (valy > 0)  { motor(1,velocidad_defecto*valy_map); }
                  valy=analogRead (yPin);
                  valy= valy - valy_referencia;
                  valy_map= abs(valy);
                  valy_map= map(valy_map, 0, (valx_referencia-10), 0, 5);
                  }
                 }

 delayMicroseconds(50);

 // movimiento joisitck eje x
 valx=analogRead (xPin);
 valx= valx - valx_referencia;
 valx_map= abs(valx);
 valx_map= map(valx_map, 0, (valx_referencia-30), 0, 5);
 if ( valx_map != 0 ) {
                  while ( valx_map != 0 )
                  {
                    //para mover con más precisión el motor
        if ((valx > 1) && (valx_map > 2) ) { motor( 1,1);delay(100);} // el delay retrasa
la velocidad del motor
       if ((valx < -1) && (valx_map > 2) ) { motor( -1,1);delay(100);} // el delay retrasa
la velocidad del motor

                  valx=analogRead (xPin);
                  valx= valx - valx_referencia;
                  valx_map= abs(valx);
                  valx_map= map(valx_map, 0, (valx_referencia-10), 0, 5);
                  }
                 }

// Leer boton digital del joistick
if ( digitalRead(clickPin)==LOW) {
                          time=millis();
                          while ((digitalRead(clickPin) == LOW) )
                            {
                            if (millis()-time >= click_time) { click_shutter_button(); }
                            // if (millis()-time >= hold_time) , por si añado algo
                            }
                          }
}  // final del loop

//*********************************************************************
void imprimir_menu_inicial(){

 lcd.clear();
```

```
 //imprimir pasos
 imprimir_pasos(z);
 //imprimir micras
 { lcd.setCursor(9,0), lcd.print("      "); lcd.setCursor(9,0); lcd.print(micras);}
 // imprimir microsteps
 { lcd.setCursor(15,0), lcd.print(" "); lcd.setCursor(15,0); lcd.print(microsteps);}
 // imprimir estado motor
 { lcd.setCursor(15,1);lcd.print(" ");lcd.setCursor(15,1); if (estado_motor == 0
)lcd.print("0");
 if (estado_motor == 1 )lcd.print("1");}
 // imprimir estado ztop y zbotton
 {lcd.setCursor(0,0);lcd.print("A:"); lcd.print( int(estado_ztop)); lcd.print("
B:"),lcd.print(int(estado_zbotton));
 }
 } // fin imprimir:menu_inicial
 //*************************************************************************
  //  Para reducir el ruido en la lectura analogica, lee varias veces el pin analogico y hace
la media.

  int analogread_repetido( int Pin_leer, int repeticiones ) {
  int valor_sumatorio=0;
  analogRead(Pin_leer);
  delayMicroseconds(100);
  for ( int i=1; i <= repeticiones; i++) {
                                valor_sumatorio= analogRead(Pin_leer) + valor_sumatorio;
                                delayMicroseconds(100);
                                }
  int valor_medio_2= ( valor_sumatorio / repeticiones );
  return valor_medio_2;

  } // Fin analogread_repetido

//*************************************************************************
void disp_camara_keyboard()    {
                            Keyboard.begin();
                            delay(100);
                           // asignar_tecla();
                            Keyboard.write(tecla);
                            delay(100);
                            Keyboard.end();
                           }

//*************************************************************************
void asignar_tecla()    {

if    (camara==1) tecla= KEY_LEFT_CTRL        ;       //      1
if    (camara==2) tecla= KEY_LEFT_SHIFT       ;       //      2
if    (camara==3) tecla= KEY_LEFT_ALT         ;       //      3
if    (camara==4) tecla= KEY_LEFT_GUI         ;       //      4
if    (camara==5) tecla= KEY_RIGHT_CTRL       ;       //      5
if    (camara==6) tecla= KEY_RIGHT_SHIFT      ;       //      6
```

```
if   (camara==7) tecla= KEY_RIGHT_ALT      ;    //    7
if   (camara==8) tecla= KEY_RIGHT_GUI      ;    //    8
if   (camara==9) tecla= KEY_UP_ARROW       ;    //    9
if   (camara==10) tecla= KEY_DOWN_ARROW    ;    //    10
if   (camara==11) tecla= KEY_LEFT_ARROW    ;    //    11
if   (camara==12) tecla= KEY_RIGHT_ARROW   ;    //    12
if   (camara==13) tecla= KEY_BACKSPACE     ;    //    13
if   (camara==14) tecla= KEY_TAB           ;    //    14
if   (camara==15) tecla= KEY_RETURN        ;    //    15
if   (camara==16) tecla= KEY_ESC           ;    //    16
if   (camara==17) tecla= KEY_INSERT        ;    //    17
if   (camara==18) tecla= KEY_DELETE        ;    //    18
if   (camara==19) tecla= KEY_PAGE_UP       ;    //    19
if   (camara==20) tecla= KEY_PAGE_DOWN     ;    //    20
if   (camara==21) tecla= KEY_HOME          ;    //    21
if   (camara==22) tecla= KEY_END           ;    //    22
if   (camara==23) tecla= KEY_CAPS_LOCK     ;    //    23
if   (camara==24) tecla= KEY_F1       ;    //    24
if   (camara==25) tecla= KEY_F2       ;    //    25
if   (camara==26) tecla= KEY_F3       ;    //    26
if   (camara==27) tecla= KEY_F4       ;    //    27
if   (camara==28) tecla= KEY_F5       ;    //    28
if   (camara==29) tecla= KEY_F6       ;    //    29
if   (camara==30) tecla= KEY_F7       ;    //    30
if   (camara==31) tecla= KEY_F8       ;    //    31
if   (camara==32) tecla= KEY_F9       ;    //    32
if   (camara==33) tecla= KEY_F10      ;    //    33
if   (camara==34) tecla= KEY_F11      ;    //    34
if   (camara==35) tecla= KEY_F12      ;    //    35 espacio
if    (camara==36) tecla=' '; // 36
if    (camara==37) tecla='0'; // 37
if    (camara==38) tecla='1'; // 38
if    (camara==39) tecla='2'; // 39
if    (camara==40) tecla='3'; // 40
if    (camara==41) tecla='4'; // 41
if    (camara==42) tecla='5'; // 42
if    (camara==43) tecla='6'; // 43
if    (camara==44) tecla='7'; // 44
if    (camara==45) tecla='8'; // 45
if    (camara==46) tecla='9'; // 46
if    (camara==47) tecla='A'; // 47
if    (camara==48) tecla='B'; // 48
if    (camara==49) tecla='C'; // 49
if    (camara==50) tecla='D'; // 50
if    (camara==51) tecla='E'; // 51
if    (camara==52) tecla='F'; // 52
if    (camara==53) tecla='G'; // 53
if    (camara==54) tecla='H'; // 54
if    (camara==55) tecla='I'; // 55
if    (camara==56) tecla='J'; // 56
if    (camara==57) tecla='K'; // 57
```

```
if    (camara==58) tecla='L'; // 58
if    (camara==59) tecla='M'; // 59
if    (camara==60) tecla='N'; // 60
if    (camara==61) tecla='Ñ'; // 61
if    (camara==62) tecla='O'; // 62
if    (camara==63) tecla='P'; // 63
if    (camara==64) tecla='Q'; // 64
if    (camara==65) tecla='R'; // 65
if    (camara==66) tecla='S'; // 66
if    (camara==67) tecla='T'; // 67
if    (camara==68) tecla='U'; // 68
if    (camara==69) tecla='V'; // 79
if    (camara==70) tecla='W'; // 70
if    (camara==71) tecla='X'; // 71
if    (camara==72) tecla='Y'; // 72
if    (camara==73) tecla='Z'; // 73
if    (camara==74) tecla='a'; // 74
if    (camara==75) tecla='b'; // 75
if    (camara==76) tecla='c'; // 76
if    (camara==77) tecla='d'; // 77
if    (camara==78) tecla='e'; // 78
if    (camara==79) tecla='f'; // 79
if    (camara==80) tecla='g'; // 80
if    (camara==81) tecla='h'; // 81
if    (camara==82) tecla='i'; // 82
if    (camara==83) tecla='j'; // 83
if    (camara==84) tecla='k'; // 84
if    (camara==85) tecla='l'; // 85
if    (camara==86) tecla='m'; // 86
if    (camara==87) tecla='n'; // 87
if    (camara==88) tecla='ñ'; // 88
if    (camara==89) tecla='o'; // 89
if    (camara==90) tecla='p'; // 90
if    (camara==91) tecla='q'; // 91
if    (camara==92) tecla='r'; // 92
if    (camara==93) tecla='s'; // 93
if    (camara==94) tecla='t'; // 94
if    (camara==95) tecla='u'; // 95
if    (camara==96) tecla='v'; // 96
if    (camara==97) tecla='w'; // 97
if    (camara==98) tecla='x'; // 98
if    (camara==99) tecla='y'; // 99
if    (camara==100) tecla='z'; // 100
if    (camara==101) tecla=':'; // 101
if    (camara==102) tecla=';'; // 102
if    (camara==103) tecla=','; // 103
if    (camara==104) tecla='.'; // 104
if    (camara==105) tecla='-'; // 105
if    (camara==106) tecla='<'; // 106
if    (camara==107) tecla='>'; // 107
if    (camara==108) tecla='!'; // 108
```

```
if     (camara==109) tecla=""; // 109
if     (camara==110) tecla='·'; // 110 punto en medio
if     (camara==111) tecla='$'; // 111
if     (camara==112) tecla='%'; // 112
if     (camara==113) tecla='&'; // 113
if     (camara==114) tecla='('; // 114
if     (camara==115) tecla=')'; // 115
if     (camara==116) tecla='?'; // 116
if     (camara==117) tecla='¿'; // 117
if     (camara==118) tecla='`'; // 118
if     (camara==119) tecla='´'; // 119
if     (camara==120) tecla='+'; // 120
if     (camara==121) tecla='ç'; // 121
if     (camara==122) tecla='Ç'; // 122
if     (camara==123) tecla='*'; // 123
if     (camara==124) tecla='º'; // 124
if     (camara==125) tecla='ª'; // 125
if     (camara==126) tecla='¡'; // 126
if     (camara==127) tecla='~'; // 127
if     (camara==128) tecla='['; // 128
if     (camara==129) tecla=']'; // 129
if     (camara==130) tecla='^'; // 130
if     (camara==131) tecla='#'; // 131

 } //fin void asiganr_tecla()

//*****************************************************************************
// configurar menu general
void config_menu() {

byte temp_secuencia=0;// usado en config_menu
estado_menu=false;

  label_2:
  temp_secuencia= 0;   //  0 por poner un numero que no coincida con secuencia
 do { pulsaciones_teclado();

    if (key==key08_C) secuencia=secuencia--;  // si aprietas 2 sube
    if (key==key11_C) secuencia=secuencia++;  // si aprietas 8 baja

    if ((secuencia==1)  && (temp_secuencia != 1)) { imprimir1("  Microsteps");
imprimir2(microsteps);         } else
    if ((secuencia==2)  && (temp_secuencia != 2)) { imprimir1("  Time_1_motor");
imprimir2(tiempo_vibracion);   } else
    if ((secuencia==3)  && (temp_secuencia != 3)) { imprimir1("  Time_2_foto");
imprimir2(tiempo_exposicion);  } else
    if ((secuencia==4)  && (temp_secuencia != 4)) { imprimir1("  Pasos_vuelta");
imprimir2(num_pasos_vuelta);   } else
    if ((secuencia==5)  && (temp_secuencia != 5)) { imprimir1("  Sentido_giro");
imprimir2(sentido_giro);       } else
```

```
   if ((secuencia==6)  && (temp_secuencia != 6)) { imprimir1("  Sonido");
imprimir2(sonido);            } else
   if ((secuencia==7)  && (temp_secuencia != 7)) { imprimir1("  Velocidad");
imprimir2(velocidad_defecto);  } else
   if ((secuencia==8)  && (temp_secuencia != 8)) {
imprimir1("Distancia_vuelta");imprimir2(micras_vuelta);      } else
   if ((secuencia==9)  && (temp_secuencia != 9)) { imprimir1("
Replica_fotos");imprimir2(reply_fotos);        } else
   if ((secuencia==10) && (temp_secuencia != 10)){ imprimir1("  Tipo_camara");
imprimir2(camara);            }

   temp_secuencia=secuencia;
   if (secuencia==11)   {secuencia=1;}  // vuelve al principio, debe ser valor maximo+1
   if (secuencia==0)    {secuencia=10;}  // debe ser igual al maximo valor

   if ((secuencia==2) && (key==key12_C)) config_tiempo_vibracion();  else
   if ((secuencia==3) && (key==key12_C)) config_tiempo_exposicion(); else
   if ((secuencia==1) && (key==key12_C)) config_microsteps();        else
   if ((secuencia==4) && (key==key12_C)) config_pasos_vuelta();      else
   if ((secuencia==5) && (key==key12_C)) config_sentido_giro();      else
   if ((secuencia==6) && (key==key12_C)) config_sonido();           else
   if ((secuencia==7) && (key==key12_C)) config_velocidad_defecto(); else
   if ((secuencia==8) && (key==key12_C)) config_micras_vuelta();     else
   if ((secuencia==9) && (key==key12_C)) config_reply_fotos();       else
   if ((secuencia==10)&& (key==key12_C))config_camara();

   if ( estado_menu == true  )    { estado_menu=false; goto label_2;}

  } while ( key!=key10_C );  // tecla 10 clic ....salir de configuracion

 imprimir_menu_inicial(); // vuelve a mostrar menu principal

}  // Fin config_menu()

//*********************************************************************
void imprimir1(String titulo) { lcd.clear(); // imprime primera linea
                        lcd.home();
                        lcd.print(titulo);  }

void imprimir2(int titulo2) { lcd. setCursor(0,1);  // imprimir segunda linea

                        lcd.print(titulo2);  }

//*********************************************************************
 void menu_micras()
 {
 boolean secuencia=1;  // usado en config_menu
 key=0;

 do { pulsaciones_teclado();
```

```
  if (secuencia==1) { imprimir1("Distancia x Foto"); imprimir2(micras); secuencia=0;
} else
  if (key == key12_C) config_micras();    //key=22 igual a key12_C
  if ( estado_menu == true  )    { estado_menu=false; secuencia=1;}

  } while ( key != key10_C );  // tecla #-hold ....salir de configuracion  key12_H

 imprimir_menu_inicial();
}  // Fin config_menu()

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

void config_velocidad_defecto() {

 lcd.clear();
 lcd.setCursor(0,0);
 lcd.print("Cf_Velocidad_def");
 int anterior=velocidad_defecto;
 int valor_temp =leer_teclado();
 if ((velocidad_defecto != valor_temp) && (valor_temp >= velocidad_defecto_min)
&& (confirmar==true))
    {velocidad_defecto=valor_temp;} else error_velocidad_defecto();
 if (velocidad_defecto != anterior )
eeprom_writeInt(eeprom_dir_velocidad_defecto,velocidad_defecto); // memoria 22-23
estado_menu = !estado_menu;

 } // fin config_velocidad_defecto

//*********************************************************************
void config_sonido() {

  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Cfg_Sonido");
  int anterior=sonido;
  int valor_temp =leer_teclado();
  if  ((valor_temp != 1) && (valor_temp != 0)) { error_sonido(); }
  if ((valor_temp == 1) || (valor_temp == 0)) {if ((sonido != valor_temp) &&
(confirmar==true)  ) {sonido=valor_temp;} }
  if (sonido != anterior ) {EEPROM.write(eeprom_dir_sonido, sonido); // memoria 20
                            if (sonido==1) {sonido= 1;} else
                              if (sonido==0) {sonido= 0;}
                                 }
  estado_menu = !estado_menu;

} // fin config_sonido

//*********************************************************************
// configurar micras entre foto y foto para realizar la pila
 void config_micras(){
```

```
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Cnfg_Micras/foto");
  int anterior=micras;
  int valor_temp =leer_teclado();
  if ((valor_temp==0) && (confirmar==false)) { micras= micras;}
  if ((valor_temp==0) && (confirmar==true)) {error_micras();} else
  if ((micras != valor_temp)  && (confirmar==true) ) micras=valor_temp;
  if (micras != anterior ) eeprom_writeInt(eeprom_dir_micras,micras); // memoria 14 y
15

  estado_menu = !estado_menu;

 } // fin void config_micras()

//*********************************************************************
// microsteps, preguntar 1,2,4,8
void config_microsteps() {

  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Cnfg_microsteps");
  int anterior=microsteps;
  int valor_temp =leer_teclado();
  if ((valor_temp != 1) && (valor_temp != 2) && (valor_temp != 4) && (valor_temp !=
8)) { error_microsteps();}
       else  if ((microsteps != valor_temp) && (confirmar==true))
microsteps=valor_temp;
 if (microsteps != anterior )  { EEPROM.write(eeprom_dir_microsteps,microsteps); //
memoria en 4
                            if (microsteps==1)
{digitalWrite(ms1Pin,LOW);digitalWrite(ms2Pin,LOW);} else
                             if (microsteps==2)
{digitalWrite(ms1Pin,HIGH);digitalWrite(ms2Pin,LOW);} else
                              if (microsteps==4)
{digitalWrite(ms1Pin,LOW);digitalWrite(ms2Pin,HIGH);} else
                               if (microsteps==8)
{digitalWrite(ms1Pin,HIGH);digitalWrite(ms2Pin,HIGH);} else
                                    config_microsteps();}  // poner label acordarse..

  estado_menu = !estado_menu;
  }  // fin void config_microsteps()

//*********************************************************************
// configurar pasos cuantos pasos equivalen a una vuelta del piñon
void config_pasos_vuelta() {

  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Cfg_Pasos/Vuelta");
  int  anterior= num_pasos_vuelta;
```

```
  int valor_temp =leer_teclado();
  if ((num_pasos_vuelta != valor_temp) && (confirmar==true))
num_pasos_vuelta=valor_temp;
  if (num_pasos_vuelta != anterior)
eeprom_writeInt(eeprom_dir_num_pasos_vuelta,num_pasos_vuelta); // memoria en 5 y
6

  estado_menu = !estado_menu;
  }   // void config_pasos_vuelta()

//**********************************************************************
// configurar tiempo necesario para la toma de la foto, tiempo2
 void config_tiempo_exposicion() {

   lcd.clear();
   lcd.setCursor(0,0);
   lcd.print("Cfg_Tiempo2/Foto");
   int anterior=tiempo_exposicion;
   int valor_temp =leer_teclado();
   if ((valor_temp == 0 ) && (confirmar==true))error_tiempo(); else
   if ((tiempo_exposicion != valor_temp ) && (confirmar==true))
tiempo_exposicion=valor_temp;
   if (tiempo_exposicion != anterior )
eeprom_writeInt(eeprom_dir_tiempo_exposicion,tiempo_exposicion); // memoria 7-8
   estado_menu = !estado_menu;
 }  // fin void config_tiempo_exposicion()

//**********************************************************************
// configurar tiempo para eliminar vibraciones, tiempo1
void config_tiempo_vibracion(){

  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Cfg_Tiempo1/Mtor");
  int anterior=tiempo_vibracion;
  int valor_temp =leer_teclado();
  if ((valor_temp == 0 ) && (confirmar==true)) error_tiempo(); else
  if ((tiempo_vibracion != valor_temp) && (confirmar==true))
tiempo_vibracion=valor_temp;
  if (tiempo_vibracion != anterior )
eeprom_writeInt(eeprom_dir_tiempo_vivbracion,tiempo_vibracion); // memoria 9 y 10
 estado_menu = !estado_menu;
 } // fin void config_tiempo_vibracion()

//**********************************************************************
// configurar sentido giro
 void config_sentido_giro(){
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Cfg_Sentido_Giro");
  int anterior=sentido_giro;
```

```
  int valor_temp =leer_teclado();
  if  ((valor_temp != 1) && (valor_temp != 0)) { error_sentido_giro(); }  // hay que
cambiar config_sentido por error
  if ((valor_temp == 1) || (valor_temp == 0)) {if ((sentido_giro != valor_temp) &&
(confirmar==true) )
                                    {sentido_giro=valor_temp;} }
  if (sentido_giro != anterior ) {eeprom_writeInt(eeprom_dir_sentido_giro
,sentido_giro); // memoria 11
                              if (sentido_giro==1) {sentido_giro= 1;} else
                              if (sentido_giro==0) {sentido_giro= 0;}
                             }
  estado_menu = !estado_menu;
 } // fin void config_sentido_giro()

//*********************************************************************
// configurar micras que corresponden a una vuelta
void config_micras_vuelta(){

  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Cf_Micras/Vuelta");
  int anterior=micras_vuelta;
  int valor_temp =leer_teclado();
  if ((micras_vuelta != valor_temp) && (confirmar==true)) micras_vuelta=valor_temp;
  if (micras_vuelta != anterior )
eeprom_writeInt(eeprom_dir_micras_vuelta,micras_vuelta); // memoria 12 y 13

  estado_menu = !estado_menu;
 } // fin void config_micras_vuelta()
 //*********************************************************************
 void config_reply_fotos(){

  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Cfg_Replic_fotos");
  int anterior=reply_fotos;
  int valor_temp = leer_teclado();
  if ((valor_temp == 0 ) && (confirmar == true)) error_reply_fotos(); else
  if ((reply_fotos != valor_temp) && (confirmar == true)) reply_fotos=valor_temp;
  if (reply_fotos != anterior ) eeprom_writeInt(eeprom_dir_reply_fotos,reply_fotos);
  estado_menu = !estado_menu;
 } // fin void config_reply_fotos()
 //*********************************************************************
// configurar micras entre foto y foto para realizar la pila
 void config_camara(){

  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Cnfg Tipo Camara");
  byte anterior=camara;
  byte valor_temp =leer_teclado();
```

```
  if ((valor_temp >= 130) && (confirmar == true)) {error_camara();}
  if ((camara != valor_temp)  && (confirmar == true) && (valor_temp <= 130)  )
camara=valor_temp;
  if (camara != anterior  ) { asignar_tecla();
eeprom_writeInt(eeprom_dir_camara,camara); } // memoria 26 y 27

  estado_menu = !estado_menu;

 } // fin void config_micras()

//**********************************************************************
void error_sentido_giro() {
  lcd.clear(); lcd.print("   Error 1/0    "); lcd.setCursor(0,1); lcd.print("    Pres key    ");
  while ( 1 ) { pulsaciones_teclado(); if (key != 0) break; } // esperar a que se pulse una
tecla
}

//**********************************************************************
void error_sonido() {
  lcd.clear(); lcd.print("   Error 1/0    "); lcd.setCursor(0,1); lcd.print("    Press key   ");
  while ( 1 ) { pulsaciones_teclado(); if (key != 0) break; } // esperar a que se pulse una
tecla
}

//**********************************************************************
void error_velocidad_defecto() {
    lcd.clear(); lcd.print(" Error Vel > 25   "); lcd.setCursor(0,1); lcd.print("    Press key
");
    while ( 1 ) { pulsaciones_teclado(); if (key != 0) break; }  // esperar a que se pulse una
tecla
    velocidad_defecto= velocidad_defecto_min;
}

//**********************************************************************
void error_microsteps() {
  lcd.clear(); lcd.print("Error  S=1/2/4/8"); lcd.setCursor(0,1); lcd.print("    Press key
");
  while ( 1 ) { pulsaciones_teclado(); if (key != 0) break; } // esperar a que se pulse una
tecla
}

//**********************************************************************
void error_micras() {
  lcd.clear(); lcd.print("Error micras !0 "); lcd.setCursor(0,1); lcd.print("    Press key   ");
  while ( 1 ) { pulsaciones_teclado(); if (key != 0) break; } // esperar a que se pulse una
tecla
}
//**********************************************************************

void error_tiempo() {
```

```
 lcd.clear(); lcd.print("Error tiempo !0 "); lcd.setCursor(0,1); lcd.print("    Press key   ");
 while ( 1 ) { pulsaciones_teclado(); if (key != 0) break; } // esperar a que se pulse una
tecla
}
//*********************************************************************
void error_reply_fotos() {
 lcd.clear(); lcd.print("Error Rep_fot !0"); lcd.setCursor(0,1); lcd.print("    Press key
");
 while ( 1 ) { pulsaciones_teclado(); if (key != 0) break; } // esperar a que se pulse una
tecla
}

//*********************************************************************
void error_camara() {
  lcd.clear(); lcd.print("      Error     "); lcd.setCursor(0,1); lcd.print("    Press key   ");
  while ( 1 ) { pulsaciones_teclado(); if (key != 0) break; }  // esperar a que se pulse una
tecla
}

//*********************************************************************
// Write a unsigned int (two bytes) value to eeprom
void eeprom_writeInt(int address, unsigned int value){

 EEPROM.write(address, value/256);
 EEPROM.write(address+1, value % 256);
}

//*********************************************************************
// read a unsigned int (two bytes) value from eeprom
unsigned int eeprom_readInt(int address){

 return EEPROM.read(address)*256+EEPROM.read(address+1);
}

//*********************************************************************
void cargar_eeprom() {  if (( EEPROM.read(eeprom_dir_version)) ==
version_programa) { config_init();} else //carga los valores de eeprom
                          { // si es diferente la version grabar los valores nuevos
                          // clear the eeprom
                          for (int i = 0; i < EEPROM_SIZE; i++)  {EEPROM.write(i,
0xFF);}
                          // escribir valores
                             EEPROM.write(eeprom_dir_microsteps, microsteps);
                             EEPROM.write(eeprom_dir_version, version_programa);
                             eeprom_writeInt(eeprom_dir_reply_fotos, reply_fotos);
                             eeprom_writeInt(eeprom_dir_num_pasos_vuelta,
num_pasos_vuelta);
                             eeprom_writeInt(eeprom_dir_tiempo_exposicion,
tiempo_exposicion);
                             eeprom_writeInt(eeprom_dir_tiempo_vivbracion,
tiempo_vibracion);
```

```
                                eeprom_writeInt(eeprom_dir_sentido_giro, sentido_giro);
                                eeprom_writeInt(eeprom_dir_micras, micras);
                                eeprom_writeInt(eeprom_dir_micras_vuelta,
micras_vuelta);
                                EEPROM.write(eeprom_dir_sonido, sonido);
                                eeprom_writeInt(eeprom_dir_velocidad_defecto,
velocidad_defecto);
                                eeprom_writeInt(eeprom_dir_camara, camara);
                              }
  }// fin void cargar_eeprom

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

void config_init() {
  // Leer y sustituir parametros de la memoria

 if (microsteps       != (EEPROM.read(eeprom_dir_microsteps)))
                         {microsteps=EEPROM.read(eeprom_dir_microsteps);}
 if (num_pasos_vuelta  != (eeprom_readInt(eeprom_dir_num_pasos_vuelta)))

{num_pasos_vuelta=(eeprom_readInt(eeprom_dir_num_pasos_vuelta));}
 if (tiempo_exposicion != (eeprom_readInt(eeprom_dir_tiempo_exposicion)))

{tiempo_exposicion=(eeprom_readInt(eeprom_dir_tiempo_exposicion));}
 if (tiempo_vibracion  != (eeprom_readInt(eeprom_dir_tiempo_vivbracion)))

{tiempo_vibracion=(eeprom_readInt(eeprom_dir_tiempo_vivbracion));}
 if (sentido_giro     != (eeprom_readInt(eeprom_dir_sentido_giro)))
                         {sentido_giro=(eeprom_readInt(eeprom_dir_sentido_giro));}
 if (micras          != (eeprom_readInt(eeprom_dir_micras)))
                         {micras=(eeprom_readInt(eeprom_dir_micras));}
 if (micras_vuelta    != (eeprom_readInt(eeprom_dir_micras_vuelta)))
                         {micras_vuelta=(eeprom_readInt(eeprom_dir_micras_vuelta));}
 if (sonido          != (EEPROM.read(eeprom_dir_sonido)))
                         {sonido=EEPROM.read(eeprom_dir_sonido);}
 if (velocidad_defecto != (eeprom_readInt(eeprom_dir_velocidad_defecto)))

{velocidad_defecto=(eeprom_readInt(eeprom_dir_velocidad_defecto));}
 if (reply_fotos      != (eeprom_readInt(eeprom_dir_reply_fotos)))
                         {reply_fotos=(eeprom_readInt(eeprom_dir_reply_fotos));}
 if (camara          != (eeprom_readInt(eeprom_dir_camara)))
                         {camara=(eeprom_readInt(eeprom_dir_camara));}
 } // fin cofig_init

//*********************************************************************
// mover hasta ztop, punto más alto
void mover_ztop() {

 int mover_ztop_pasos= ztop-z;
 motor(mover_ztop_pasos, velocidad_defecto*multiplicador_vel);
}
```

```
//*********************************************************************
// mover hasta ztop, punto más alto
void mover_zbotton() {

  int mover_zbotton_pasos= zbotton-z;
  motor(mover_zbotton_pasos, velocidad_defecto*multiplicador_vel);
}
//*********************************************************************
void conectar_motor()  // conecta y desconecta el motor
{
 temp_estado_motor=estado_motor;
 if ( temp_estado_motor == false ) { digitalWrite(sleepPin, HIGH); estado_motor=true
;}
 if ( temp_estado_motor == true  ) { digitalWrite(sleepPin, LOW );
estado_motor=false;}

 imprimir_menu_inicial();
}
//*********************************************************************
/*
 Codigo de programa para PCF8574 I2C I/O expander
 https://sites.google.com/site/arduinoprojectst/i2c-keypad-and-lcd-controller/keypad-
and-port-expander-1

 On port-expander (port - row or column):
 P7 P6 P5 P4 P3 P2 P1 P0  : puerto del PCF8574
 R3 R2 R1 R0 -- C2 C1 C0  : columnas y filas ( column, row )

 void keyscan()
 The columns are connected to the three lowest ports (P2, P1, P0)
 and the columns have a weak pullup-resistor. We send a 1 to all three
 columns at the same time, we send a 0 on the row for which we want to test,
 and we send a 1 on the rows that we are not testing. The column on which
 a key has been pressed will turn from 1 to 0.
 Example: We send 11110111 which means we are testing on row 0 (R0). If the key
 on row 0 and colum 1 is pressed (number 2 on keypad), then we will read the
 pattern 11110101. In other words, the second lowest bit has been changed. We
 repeat the procedure for each existing row on the keypad. In short:
 with port connection [(-) R2 R1 R0 C2 C1 C0 ],
 send 11110111 receive 11110101 means R0 C1 key was pressed.
 Next, we send 11101111 to test row R1, etc.
 */
int key_scan()
{
  byte send_pattern, receive_pattern,test_pattern;
  byte send_pattern_array[]={
    B11101111, B11011111, B10111111, B01111111                    };
  byte test_pattern_array[]={   B00000001, B00000010, B00000100                    };
  int i=0;
  int scan_key=0;
  int scan_key_temp=0;
```

```
 row=5;
 column=5;

 for (i=0; i<4;i++)
 {
  // Try each row. Send 0 on R1 port, the bit on the pressed column
  // will turn from 1 to 0
  send_pattern=send_pattern_array[i];
  expanderWrite(send_pattern);
  receive_pattern=expanderRead();

  if(send_pattern!=receive_pattern)
  {
   row=i;

   for (int j=0;j<3;j++)
   {
    test_pattern=test_pattern_array[j]&receive_pattern;
    //Serial.println(btoa(test_pattern));
    if(test_pattern==0)
     column=j;
   } }  }

if (row==0) { if ( column == 0 ) { scan_key_temp = 11;} else
            if ( column == 1 ) { scan_key_temp = 12;} else
            if ( column == 2 ) { scan_key_temp = 13;}     }

if (row==1) { if ( column == 0 ) { scan_key_temp = 14;} else
            if ( column == 1 ) { scan_key_temp = 15;} else
            if ( column == 2 ) { scan_key_temp = 16;}     }

if (row==2) { if ( column == 0 ) { scan_key_temp = 17;} else
            if ( column == 1 ) { scan_key_temp = 18;} else
            if ( column == 2 ) { scan_key_temp = 19;}     }

if (row==3) { if ( column == 0 ) { scan_key_temp = 20;} else
            if ( column == 1 ) { scan_key_temp = 21;} else
            if ( column == 2 ) { scan_key_temp = 22;}     }

if ( scan_key_temp != scan_key ) { scan_key = scan_key_temp;}

 return scan_key;
}
//************************************************************************
void expanderWrite(byte _data ) {
 Wire.beginTransmission(expander);
 Wire.write(_data);
 Wire.endTransmission();
}
//************************************************************************
byte expanderRead() {
```

```
  byte _data;
  Wire.requestFrom(expander, 1);
  if(Wire.available()) {
    _data = Wire.read();
  }
  return _data;
}
//*****************************************************************************
// Entrar datos con el teclado
long leer_teclado() {
confirmar=false;
unsigned int sumatorio=0;
byte posicion=1;

lcd.setCursor(0,1);  lcd.print("?              "); lcd.setCursor(1,1); lcd.blink(); lcd.cursor();

for ( boolean exit=false; !exit ;) {   pulsaciones_teclado();

 if ( key==key01_C ) { sumatorio=sumatorio*10 + 1; lcd.setCursor(posicion,1);
lcd.print(1); posicion++;} else
 if ( key==key02_C ) { sumatorio=sumatorio*10 + 2; lcd.setCursor(posicion,1);
lcd.print(2); posicion++;} else
 if ( key==key03_C ) { sumatorio=sumatorio*10 + 3; lcd.setCursor(posicion,1);
lcd.print(3); posicion++;} else
 if ( key==key04_C ) { sumatorio=sumatorio*10 + 4; lcd.setCursor(posicion,1);
lcd.print(4); posicion++;} else
 if ( key==key05_C ) { sumatorio=sumatorio*10 + 5; lcd.setCursor(posicion,1);
lcd.print(5); posicion++;} else
 if ( key==key06_C ) { sumatorio=sumatorio*10 + 6; lcd.setCursor(posicion,1);
lcd.print(6); posicion++;} else
 if ( key==key07_C ) { sumatorio=sumatorio*10 + 7; lcd.setCursor(posicion,1);
lcd.print(7); posicion++;} else
 if ( key==key08_C ) { sumatorio=sumatorio*10 + 8; lcd.setCursor(posicion,1);
lcd.print(8); posicion++;} else
 if ( key==key09_C ) { sumatorio=sumatorio*10 + 9; lcd.setCursor(posicion,1);
lcd.print(9); posicion++;} else
 if ( key==key10_C ) { sumatorio=sumatorio/10; posicion-- ; if ( posicion <= 0)
{lcd.print("?              ");
                posicion=1;} lcd.setCursor(posicion,1);  lcd.print(" ");
lcd.setCursor(posicion,1); }
                else      // borrar tecla
 if ( key==key11_C ) { sumatorio=sumatorio*10; lcd.setCursor(posicion,1);
lcd.print(0); posicion++;}     else
 if ( key==key12_C ) { confirmar=true;exit=true;}                else// boton #, confirmar
y devolver valor
 if ( key==key10_H ) { sumatorio=0; posicion=1;lcd.setCursor(0,1);lcd.print("?
");
                lcd.blink();lcd.setCursor(1,1);} else
 if ( key==key12_H ) { confirmar=false;exit=true;}

                        }
```

```
 lcd.noBlink(); lcd.noCursor();
 return sumatorio;
} // fin void leer_teclado

//*********************************************************************
  void menu_teclado()   {   // hacer algo si se aprieta una tecla del teclado

  pulsaciones_teclado(); // pulsacion tecla, =1 para click; =2 para hold

  if ( key==key01_C )  { motor( calcular_pasos_micra(1),
velocidad_defecto*multiplicador_vel);} else//mover motor una micra
  if ( key==key01_H )  { motor( calcular_pasos_micra(5),
velocidad_defecto*multiplicador_vel);} else //mover motor un 5 micra arriba
  if ( key==key02_C )  { motor(
calcular_pasos_micra(100),velocidad_defecto*multiplicador_vel);} else // mover motor
100 micras
 if ( key==key02_H )  { motor(
calcular_pasos_micra(micras_vuelta),velocidad_defecto*multiplicador_vel);} else
//mueve el motor una vuelta completa arriba
 if ( key==key03_C )  { ztop = z; estado_ztop=true; imprimir_AB(); } else // punto alto
pila, tecla 3; marca que se ha marcado el punto alto
 if ( key==key03_H )  { mover_ztop();} else// mover a ztop
 if ( key==key04_C )  { motor( calcular_pasos_micra(-1),
velocidad_defecto*multiplicador_vel); } else //mover motor un 1 micra abajo
 if ( key==key04_H )  { motor( calcular_pasos_micra(-5),
velocidad_defecto*multiplicador_vel);} else //mover motor un 5 micra abajo
 if ( key==key05_C )  { motor( calcular_pasos_micra(-
100),velocidad_defecto*multiplicador_vel);} else //mover motor 100 micras abajo
if ( key==key05_H )  { motor( calcular_pasos_micra(-
micras_vuelta),velocidad_defecto*multiplicador_vel);} else // mueve el motor una
vuelta completa abajo
if ( key==key06_C )  { zbotton = z; estado_zbotton=true; imprimir_AB();} else// marca
punto más bajo de la pila, tecla 6, marca que ha marcado el punto bajo
if ( key==key06_H )  { mover_zbotton(); } else // mover a punto más bajo, tecla 6
 if ( key==key07_C )  { menu_micras();} else// else introducir numero pasos a recorrer
entre foto y foto, tecla 7 click
if ( key==key07_H )  { config_menu();} else// configuar menu general, tecla 7 hold
if ( key==key08_H )  { z=0; estado_ztop=false, estado_zbotton=false;
imprimir_reseteo();} else // reiniciar contadores
 if ( key==key08_C )  { imprimir_AB(); } else
 if ( key==key09_C )  { motor( -1*calcular_pasos_micra(micras),
velocidad_defecto*multiplicador_vel); } else// mueve la cantidad de micras entre foto y
foto
  if ( key==key09_H )  { if ( (estado_ztop == true) && ( micras != 0 ))
{tomar_pila_2();} else {error_valor=1; imprimir_error_2();}   }  // tomar pila de fotos ,
tecla 0 hold;} else
 if ( key==key12_H )  { if ( (estado_ztop == true) && (estado_zbotton == true) && (
micras != 0 ) && (ztop != zbotton) ) {tomar_pila();} else {error_valor=1;
imprimir_error();}  }   // tomar pila de fotos , tecla 0 hold
  if ( key==key11_C )  { conectar_motor();}
  }  // fin void menu_teclado
```

```
//*********************************************************************
// calcular pasos a mover para mover una micra
int calcular_pasos_micra(float micras_mover) {

     // total_pasos_vuelta=( num_pasos_vuelta*microsteps );
      float temp_num_pasos =
(micras_mover*(float(num_pasos_vuelta*microsteps))/(float(micras_vuelta)));
     // temp_num_pasos no ppuede ser menor que  1
     if ((temp_num_pasos >= 0) && (temp_num_pasos <= 1)) temp_num_pasos= 1;
     if ((temp_num_pasos <= 0) && (temp_num_pasos >= -1)) temp_num_pasos= -1;
      num_pasos = int ( temp_num_pasos);
      return num_pasos;
    }
//*********************************************************************
// Entrar datos con el teclado
void pulsaciones_teclado() {

key=0; // resetea key
byte estado=0;
time=millis();
boolean beep_click=false;
boolean beep_hold =false;
int temp_key=0;

 do { if( ((millis()-time)>35) && ((millis() - time) < 50)) temp_key=key_scan();
     if  ((millis() - time) >= stop_time  ) {break;} else
     if  ((millis() - time) >= hold_time  ) {estado=2; if (beep_hold==false && sonido)
{tone(13,1500,150);beep_hold=true;} } else
     if  ((millis() - time) >= click_time ) {estado=1; if (beep_click==false && sonido)
{tone(13,3000,50);beep_click=true;} } else estado=0;
                                //  tiempo para evitar rebotes, para calcular el valor medio
solo toma el intervalo entre
                                //  35 y 50 milisegundos.
     } while ( key_scan());

 // asignar tecla segun el vlaor temp_key

   if ( (temp_key==11) && (estado==1) ) key=key01_C;  else
   if ( (temp_key==11) && (estado==2) ) key=key01_H;  else

   if ( (temp_key==12) && (estado==1 ) ) key=key02_C;  else
   if ( (temp_key==12) && (estado==2 ) ) key=key02_H;  else

   if ( (temp_key==13) && (estado==1 ) ) key=key03_C;  else
   if ( (temp_key==13) && (estado==2 ) ) key=key03_H;  else
   if ( (temp_key==14) && (estado==1 ) ) key=key04_C;  else
   if ( (temp_key==14) && (estado==2 ) ) key=key04_H;  else
   if ( (temp_key==15) && (estado==1 ) ) key=key05_C;  else
   if ( (temp_key==15) && (estado==2 ) ) key=key05_H;  else
   if ( (temp_key==16) && (estado==1 ) ) key=key06_C;  else
   if ( (temp_key==16) && (estado==2 ) ) key=key06_H;  else
```

```
   if ( (temp_key==17) && (estado==1 ) ) key=key07_C;  else
   if ( (temp_key==17) && (estado==2 ) ) key=key07_H;  else
   if ( (temp_key==18) && (estado==1 ) ) key=key08_C;  else
   if ( (temp_key==18) && (estado==2 ) ) key=key08_H;  else
   if ( (temp_key==19) && (estado==1 ) ) key=key09_C;  else
   if ( (temp_key==19) && (estado==2 ) ) key=key09_H;  else
   if ( (temp_key==20) && (estado==1 ) ) key=key10_C;  else
   if ( (temp_key==20) && (estado==2 ) ) key=key10_H;  else
   if ( (temp_key==21) && (estado==1 ) ) key=key11_C;  else
   if ( (temp_key==21) && (estado==2 ) ) key=key11_H;  else
   if ( (temp_key==22) && (estado==1 ) ) key=key12_C;  else
   if ( (temp_key==22) && (estado==2 ) ) key=key12_H;  else key=0;
 }   // fin pulsaciones_teclado

//*********************************************************************
// tomar pila de imagenes, comenzar
void tomar_pila()   {

// calculo variables
float temp_num_pasos =
(((float(micras))*(float(num_pasos_vuelta*microsteps)))/(float(micras_vuelta)));
if ( temp_num_pasos <= 1 ) temp_num_pasos =1;
num_pasos = int( temp_num_pasos);  // calcula pasos a mover por para x micras*10, no
puede ser menor que 1
num_fotos= (abs((ztop-zbotton) / (num_pasos)))+1;  // hay que retocar, poner direccio,
                                    // poner la foto que falta en funcion del reto que quede
num_fotos_desplazamiento=0;
num_fotos_desplazamiento=num_fotos+num_fotos_desplazamiento;  //para calcular el
desplazamiento, suma el numero total de fotos

//  imprimir comienza pila
imprimir_pila_inicio();
}

//*********************************************************************
void imprimir_pila_inicio() {

 lcd.clear();
 lcd.setCursor(0,0);
 lcd.print(ztop);lcd.print("/");lcd.print(zbotton);lcd.print("/");lcd.print( (ztop - zbotton) );
 lcd.setCursor(0,1);
 lcd.print("M:");lcd.print(micras);lcd.print("  F:");lcd.print(num_fotos);lcd.print("
S:");lcd.print(microsteps);

 boolean temporal_exit=false;

 do
        {
        pulsaciones_teclado();
        if (key == key12_C) temporal_exit = true;
        if (key == key10_C) temporal_exit = true;
```

```
        }
        while ( temporal_exit == false ) ;

 if (key == key12_C) { key=0; tomar_pila_seguir(); }
 if (key == key10_C) { key=0; imprimir_menu_inicial(); }

}
//*********************************************************************
 //  sigue a tomar_pila()
void tomar_pila_seguir()   {

  lcd.clear();
  mover_ztop();  //vuelve al punto mas alto
  delay(500);

  motor((10+num_pasos), velocidad_defecto*multiplicador_vel); // para evitar holguras
y posicionar exactamente en ztop.
  delay(500);
  motor( -10, velocidad_defecto*multiplicador_vel);
  lcd.clear();
  delay(1); // tiempo necesario para despertar tarjeta
  digitalWrite(autofocusPin, LOW);  //  conectar bloquo exposición
  lcd.setCursor(0,0);lcd.print(" Iniciando pila ");
  delay(2000);
  lcd.clear();
  delay(AF_TIME);
  int num_fotos_restantes=0;
  contador_fotos=1;

  while( contador_fotos <= num_fotos )
                              {  // disparo con tomar_foto  y mover motor2
                              tomar_foto();
                              contador_fotos++;
                              if (key==key10_C) {key=0; break;}
                              }
   if (sonido) tone(13,20,2000);  // sonido al terminar la pila

   pila_terminar();

 //digitalWrite( sleepPin, LOW );    // desconectar motor
   digitalWrite(autofocusPin, HIGH);  // desconectar bloquo exposición
   key=0;
   imprimir_menu_inicial();
}
//*********************************************************************
void pila_terminar()  {

 lcd.clear();
 lcd.setCursor(0,0);
 lcd.print(" Terminar pila? ");
 lcd.setCursor(0,1);
```

```
 lcd.print(" D:"), lcd.print( micras*(num_fotos_desplazamiento-1));

 int temporal_exit=false;

 do
        {
        pulsaciones_teclado();
        if (key == key12_C) temporal_exit = true; //finalizar pila, ztop=0, zbotton=0, z=0
        if (key == key09_C) temporal_exit = true; //finalizar pila pero guardar valores de
ztop,
                                        //tbotton y z para realizar pila otra vez
        if (key == key08_C) temporal_exit = true; //tomar otra foto y mover motor
num_pasos
        if (key == key11_C) temporal_exit = true; //volver a hacer pila
        if (key == key07_C) temporal_exit = true; //volver a hacer pila
        }
        while ( temporal_exit == false ) ;

 if (key == key12_C) { key=0; ztop=0; estado_ztop=0; estado_zbotton=0; zbotton=0;
z=0;num_fotos_desplazamiento=0;
                imprimir_menu_inicial(); } // vuelve la principio y resetea todos los
valores
 if (key == key07_C) { key=0; lcd.clear();
tomar_foto();num_fotos_desplazamiento=(num_fotos_desplazamiento+1);
                pila_terminar();  } // toma una foto y vuelve a preguntar
 if (key == key11_C) { key=0;num_fotos_desplazamiento=0; imprimir_menu_inicial();
} // vuelve al principio
                                                        //manteniendo los valores
 if (key == key08_C) { key=0; if (estado_zbotton == 1) {tomar_pila();} else
                {error_valor=1; imprimir_error();}}//vuelve a hacer la pila con los
mismos valores, modo A.....B
 if (key == key09_C) { key=0; ztop= z - num_pasos ; tomar_pila_2(); }
// vuelve a hacer la pila con los mismos
//valores, asigna ztop a z y pregunta cuantas fotos hacer
}

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

//toma foto para la pila (tomar foto y mover motor x pasos)
void tomar_foto() {

  boolean  temporal_imprimir;
  long temp_time;

  // mueve motor y espera una tiempo para que pare la vibración
  motor( num_pasos*(-1) , velocidad_defecto*multiplicador_vel );
 // mirar dirección cuando este montado, quitar multriplicar si es necesario

  temporal_imprimir=false;
  temp_time=millis();
```

```
  while ( ((millis()- temp_time) ) <= tiempo_vibracion ) { if (temporal_imprimir==false)
{  lcd.setCursor(0,0);lcd.print("               ");
                                                          lcd.setCursor(0,0);
                                                          lcd.print("T_1");
                                                          lcd.setCursor(6,0);
                                                          lcd.print("F:");
                                                          lcd.print(contador_fotos);
                                                          lcd.print("/");
                                                          lcd.print(num_fotos);
                                                          temporal_imprimir=true;
                                                          imprimir_z2();
                                                         }
                                    pulsaciones_teclado();     // romper bucle y volver menu
principal
                                          if (key == key10_C )  goto final;
                                          if (key == key11_C )  pausa_pila(
tiempo_vibracion, 0);
                                          if (key == key08_C )  pausa_pila(
tiempo_vibracion, 1);
                                        }

  // toma foto
  for (int i=1; i <= reply_fotos; i++) {

  // disparo de camara por via mecanica, Reflex
  if (camara == 0) { digitalWrite(shutterPin, LOW); //toma foto
                  delay(SHUTTER_LAG);
                  digitalWrite(shutterPin, HIGH);
                 }
  // disparo de camara mediante software, enviando una letra del teclado
  if (camara >= 1)  {
                  disp_camara_keyboard();
                 }

  if (sonido) tone(13,200,100);  // pitido al hacer foto

  //espera un tiempo para exponer la foto
  temp_time=millis();
  lcd.clear();
  temporal_imprimir=false;
  while ( ((millis()- temp_time) ) <= tiempo_exposicion ) {
                           if (temporal_imprimir==false) { lcd.setCursor(0,0);lcd.print("
");
                                                          lcd.setCursor(0,0);
                                                          lcd.print("T_2");
                                                          lcd.setCursor(6,0);
                                                          lcd.print("F:");
                                                          lcd.print(contador_fotos);
                                                          lcd.print("/");
                                                          lcd.print(num_fotos);
                                                          temporal_imprimir=true;
```

```
                                                    imprimir_z2();
                                                   }
                                pulsaciones_teclado();    // romper bucle y volver menu
principal
                                    if (key == key10_C ) {goto final;}
                                    if (key == key11_C ) { pausa_pila(
tiempo_exposicion, 0);}
                                    if (key == key08_C ) { pausa_pila(
tiempo_exposicion, 1);}
                                  }
                        }
  // igual hay que poner que retroceda 1*num_pasos
  num_fotos_restantes++;
  final:;

}  // fin void tomar foto

//*********************************************************************
void imprimir_z2()  {

 lcd.setCursor(0,1);lcd.print("              "); lcd.setCursor(0,1);lcd.print("Z:"),
lcd.print(z);

} // fin imprimir_z2

//*********************************************************************
void pausa_pila( int tiempo_retorno, boolean estado_bloqueo_exposicion ) { /* pausa
mientras realiza la pila
  tiempo_retorno....tiempo a esperar despues que acabe la pausaestdo_bloqueo
exposición....desbloquea o
  bloquea el bloqueo de exposicion, pone en low o high el pin que controla el estado
esposicion, autofocusPin */

  if ( estado_bloqueo_exposicion == true ) digitalWrite(autofocusPin, HIGH ); //
desconecta bloqueo exposicion
  boolean temporal_imprimir=false; key=no_key;
  long temp_time2=millis();
  while ( key != key11_C) { if (temporal_imprimir==false)
                            {lcd.clear();
                            lcd.print("    Pausa     ");
                            lcd.setCursor(0,1);
                            lcd.print("  Presiona 0  ");
                            temporal_imprimir=true;
                            }
                    pulsaciones_teclado();
                    if (key == key08_C ) digitalWrite(autofocusPin, HIGH); // desconecta
bloqueo esposicion
                  }
  key=no_key;
  temp_time2=millis();
  temporal_imprimir=false;
```

```
  digitalWrite(autofocusPin, LOW );  // conecta bloqueo exposicion
  while ( ((millis()- temp_time2) ) <= tiempo_retorno ) { if ( temporal_imprimir==false)
                    {lcd.clear();lcd.setCursor(0,0),lcd.print("  REINICIANDO  ");
temporal_imprimir=true;}  }
}

//*********************************************************************
// tomar pila marcando inicio y haciendo x fotos...
void tomar_pila_2() {

 // calculo variables, calcula los pasos a mover por x micras
 // total_pasos_vuelta=( num_pasos_vuelta*microsteps );
  float temp_num_pasos =
(((float(micras))*(float(num_pasos_vuelta*microsteps)))/(float(micras_vuelta)));
  if ( temp_num_pasos <= 1 ) temp_num_pasos =1;
  num_pasos = int( temp_num_pasos);

  // mostrar informacion del comienzo de la pila
 lcd.clear();
 lcd.setCursor(0,0);
 lcd.print("Numero de fotos?");
 int temp_numero_fotos=leer_teclado();

 // el numero de fotos no puede ser 0
 if (key == key12_H)  imprimir_menu_inicial();
 if ( temp_numero_fotos==0 && key != key12_H ) { lcd.clear();
                                    lcd.print("  ERROR FOTOS=0 ");
                                    lcd.setCursor(0,1); lcd.print("    Press key   ");
  // esperar a que se pulse una tecla
   while ( 1 ) { pulsaciones_teclado(); if (key != 0) break; } // esperar a que se pulse una
tecla
                                    key=0;
                                    imprimir_menu_inicial();
                                   }
 if ( temp_numero_fotos >= 1 && key != key12_H ) { num_fotos=temp_numero_fotos;
                                     //para calcular el desplazamiento, suma el numero total
de fotos
                                     num_fotos_desplazamiento = num_fotos +
num_fotos_desplazamiento;
                                     seguir_pila_2();}
                                    }
//*********************************************************************
void seguir_pila_2() {
 //imprimir informacion de inicio.
 estado_zbotton = 0; // para que no permita hacer una pila de A....B  cuando pregunte
que hacer al terminar pila.

    lcd.clear();
    lcd.setCursor(0,0);lcd.print("Z_top:");
    lcd.print(ztop);
```

```
    lcd.setCursor(0,1);
    lcd.print("M:");lcd.print(micras);lcd.print("  F:");lcd.print(num_fotos);lcd.print("
S:");lcd.print(microsteps);

 int temporal_exit=false;

 do
        {
        pulsaciones_teclado();
        if (key == key12_C) temporal_exit = true;
        if (key == key10_C) temporal_exit = true;
        }
        while ( temporal_exit == false ) ;

 if (key == key12_C) { key=0; tomar_pila_seguir(); }
 if (key == key10_C) { key=0; imprimir_menu_inicial(); }
 }

//**********************************************************************
// disparo con el boton del joistic
void click_shutter_button() {

  if (sonido) tone(13,2000,100);
  if (camara == 0) {
          digitalWrite(autofocusPin, LOW); //  low cierra el rele y bloquea al exposicion
          delay(AF_TIME);
          digitalWrite(shutterPin, LOW);  //   low cierra el rele y activa el disparo
          delay(SHUTTER_LAG);
          digitalWrite(shutterPin, HIGH); // high abre el rele y desbloquea el disparo
          digitalWrite(autofocusPin, HIGH);  //  high abre el rele y desbloquea la
exposicion
          delay(PAUSE_TIME);
                }
  // disparo de camara mediante software, enviando una letra del teclado
  if (camara >= 1)  {
                 disp_camara_keyboard();
                }
  if (sonido) tone(13,200,100);

}  // fin void click_shutter_button()

//**********************************************************************
 // Función para mover el motor para hacer pila de imagenes, la conexion se realiza en
otra funcion
void motor( int pasos, int velocidad )
{

 if (estado_motor == true )  { // solo realiza esta funcion si el motor esta conectado

  if ( pasos < 0 ) { digitalWrite(dirPin, sentido_giro); } else
```

```
  if ( pasos > 0 ) { digitalWrite(dirPin, !sentido_giro); }
  //delay(1); // tiempo para que cambie la dirección

  int temp_pasos;
  temp_pasos=abs(pasos); // si pasos es negativo hacerlo positivo

  // calcula tiempo entre pasos
  int delay_medio_paso= tiempo_por_paso_motor(velocidad);

  // bucle para mover x pasos
  for (int i=0; i < temp_pasos; i++)
    {
     // si estado de motor no es igual a true no cuenta los pasos
    if ( pasos < 0 ) { z = z - 1; }
    if ( pasos > 0 ) { z = z + 1; }
    //int delay_medio_paso= tiempo_por_paso_motor(velocidad);
    digitalWrite(pasosPin,HIGH);
    delayMicroseconds(delay_medio_paso);
    digitalWrite(pasosPin,LOW);
    delayMicroseconds(delay_medio_paso);
    imprimir_pasos(z);
    }

   } //fin de if motor esta conectado

 } // fin void motor2( int pasos, int velocidad )
 //*************************************************************************
// funcion para calcular el tiempo entre paso y paso, para hacer variable la velocidad
    int tiempo_por_paso_motor ( int tiempo_por_paso )
    {
     int tiempo_paso;
     tiempo_paso= 1000000 / tiempo_por_paso;
                          // timepo necesario entre pasos para la velocidad en Hz, en
microsegundos
     tiempo_paso= tiempo_paso / 2; //  para calcular los dos delays que hay por cada paso
     return tiempo_paso;
   }

//*************************************************************************
 void imprimir_pasos(int valor_temporal_z)
  {
   lcd.setCursor(0,1);lcd.print("Z:");
   lcd.setCursor(2,1);lcd.print(valor_temporal_z);lcd.print("       ");
   //lcd.print("/"); lcd.print(micras);//sustituir micras por las micras que se mueve,
calcular
  }

//*************************************************************************
   void imprimir_error()
  {
    lcd.clear();
```

```
   lcd.print("    ERRROR    ");
   if ((error_valor == 1) && (estado_ztop == false) && (estado_zbotton == false))
                                        {lcd.setCursor(0,1); lcd.print("A y B no
marcados");} else
   if ((error_valor == 1) && (estado_ztop == false) && (estado_zbotton == true))
                                        {lcd.setCursor(0,1); lcd.print("  A no marcado  ");}
else
   if ((error_valor == 1) && (estado_ztop == true)  && (estado_zbotton == false))
                                        {lcd.setCursor(0,1); lcd.print("  B no marcado  ");}
else
   if ((error_valor == 1) && (estado_ztop == estado_zbotton))
                                        {lcd.setCursor(0,1); lcd.print("    A == B      ");}
  volver_principio();
  }
//*************************************************************************
 void imprimir_error_2()
  {
   lcd.clear();lcd.setCursor(0,0);
   lcd.print("    ERRROR    ");

     if ((error_valor == 1) && (estado_ztop == false))  {lcd.setCursor(0,1); lcd.print("
A no marcado  ");}

  volver_principio();

  }
//*************************************************************************
   void imprimir_reseteo()
  {
   lcd.clear();
   lcd.print("    RESETEAR    ");
   lcd.setCursor(0,1); lcd.print(" A - B - Z = 0  ");
   volver_principio();
  }
//*************************************************************************
 void imprimir_AB()
  {
  lcd.clear();
  lcd.setCursor(0,0);lcd.print("A:");lcd.print(int(estado_ztop)); if (estado_ztop==1)
{lcd.print(" :"); lcd.print(ztop);}
  lcd.setCursor(0,1);lcd.print("B:");lcd.print(int(estado_zbotton)); if
(estado_zbotton==1)
                                        {lcd.print(" :"); lcd.print(zbotton);}
  if ( estado_ztop && estado_zbotton ) {lcd.print(" ");lcd.print( ztop-zbotton);}

  volver_principio();
  }
//*************************************************************************
  void volver_principio() //  para imprimir error estado z_top o z_button, espera hasta
apretar una tecla
```

```
 {
 while ( 1 ) { pulsaciones_teclado(); if (key != 0) break; }

 imprimir_menu_inicial();
 }
//**************************************************************************
```

Diseño, construcción y programación de un sistema automático y autónomo para la adquisición de fotografías multifoco destinado a documentación científica

Anexo V

Descarga de Documentación

DESCARGA DE DOCUMENTACIÓN

Los archivos y programas necesarios para la construcción y funcionamiento de Stack_Duino pueden ser descargados de la siguiente dirección:

https://dl.dropboxusercontent.com/u/5032322/Stack_Duino.zip

Manuscrito original recibido el 28 de junio de 2014.
Publicado: 4 de agosto de 2014.

ACOPIOS

DOI: 10.7597/acopios2171-7788.2014

ACOPIOS
Revista Ibérica de Mineralogía
ISSN 2171-7788

V52014
MTIEDIT

Foto Portada:

Clinoatacamita
Mina Lily, Pisco Umay, Ica, Perú
Fot. César Menor

ACOPIOS

Revista Ibérica de Mineralogía

ISSN 2171-7788

DOI: 10.7597/acopios2171-7788.2014

http://mti-acopios.blogspot.com.es

http://issuu.com/malacate/docs/V5_2014

V52014

MTIEDIT