



*ugr* | Universidad  
de **Granada**

TRABAJO DE FIN DE GRADO  
GRADO EN INGENIERÍA INFORMÁTICA

# Ardufocuser: Una Solución Libre para Enfoque Astronómico

---

**Autor**

José Miguel López Pérez

**Tutor**

Dr. Sergio Alonso Burgos



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE  
TELECOMUNICACIÓN

—  
Granada, 12 de septiembre de 2016





---

# Ardufocuser: Una Solución Libre para Enfoque Astronómico

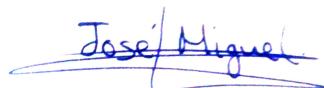
José Miguel López Pérez

---

## Declaración de Originalidad del TFG

D. **José Miguel López Pérez**, con DNI 15516308W, declara que el presente Trabajo de Fin de Grado es original, no habiéndose utilizado fuentes sin ser citadas debidamente. De no cumplir con este compromiso, soy consciente de que, de acuerdo con la Normativa de Evaluación y de Calificación de los estudiantes de la Universidad de Granada de 20 de mayo de 2013, esto conllevará automáticamente la calificación numérica de cero independientemente del resto de las calificaciones que el estudiante hubiera obtenido. Esta consecuencia debe entenderse sin perjuicio de las responsabilidades disciplinarias en las que pudieran incurrir los estudiantes que plagie.

Para que así conste lo firmo el 12 de septiembre de 2016



Firma del alumno

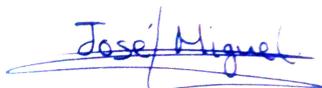
# Ardufocuser: Una Solución Libre para Enfoque Astronómico

José Miguel López Pérez

---

Yo, **José Miguel López Pérez**, alumno de Grado en Ingeniería Informática de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 15516308W, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Además, este mismo trabajo es realizado bajo licencia **Creative Commons Attribution-ShareAlike 4.0**, dando permiso para copiarlo y redistribuirlo en cualquier medio o formato, también de adaptarlo de la forma que se quiera, pero todo esto siempre y cuando se reconozca la autoría y se distribuya con la misma licencia que el trabajo original.



Firma del alumno  
Granada a 12 de septiembre de 2016

# Ardufocuser: Una Solución Libre para Enfoque Astronómico

José Miguel López Pérez

---

El Dr. Sergio Alonso Burgos, profesor del Departamento de Lenguajes y Sistemas Informáticos de la Universidad de Granada.

Informa:

Que el presente trabajo, titulado *Ardufocuser: Una Solución Libre para Enfoque Astronómico*, ha sido realizado bajo su supervisión por D. José Miguel López Pérez, y autoriza la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expido y firmo el presente informe en Granada a 12 de septiembre de 2016



Firma del tutor



# Agradecimientos

A mis padres por la cantidad de oportunidades que me han brindado con su esfuerzo, así como su cariño, en los buenos y malos momentos. Sin olvidarme de mi hermana siempre sacándome una sonrisa y mis abuelos siempre presentes.

Agradecer a mi tutor Dr. Sergio Alonso Burgos todo lo que he aprendido gracias a su labor en el presente proyecto. Transmitiendo su pasión y teniendo mucha paciencia.

Así como agradecer a mis compañeros de piso Javier, Marta, Antonio, David y Jorge por hacerme sentir en familia durante tanto tiempo. Sin olvidarme de mis compañero de clase Jaime, Oscar, Roge, Mario y Migue por hacer tantas horas de estudio más llevaderas.

Por último y no menos importante, dar las gracias a mis compañeros de trabajo, en especial a Diego, Francisco y Brian, por saber comprenderme y ayudar en todo lo posible.

*Somos polvo de estrellas que piensa acerca de las estrellas.*

- Carl Edward Sagan

# Índice general

<b>1. Resumen</b>	<b>1</b>
1.1. Breve resumen y palabras clave . . . . .	1
1.2. Extended abstract and key words . . . . .	2
<b>2. Introducción y motivación</b>	<b>5</b>
2.1. La Astronomía . . . . .	6
2.1.1. Astronomía Antigua . . . . .	7
2.1.2. Astronomía Moderna . . . . .	8
2.1.3. Astronomía en la actualidad . . . . .	9
2.1.4. Astronomía amateur . . . . .	10
2.2. Instrumental Astronómico . . . . .	11
2.2.1. Telescopios . . . . .	11
2.2.2. Cámaras CCD . . . . .	13
2.2.3. Monturas . . . . .	13
2.2.4. Rueda portafiltros . . . . .	15
2.2.5. Cúpulas . . . . .	15
2.2.6. Estaciones meteorológicas . . . . .	15
2.2.7. Enfocadores . . . . .	16
2.3. Software astronómico . . . . .	17
2.4. Imágenes Astronómicas . . . . .	18
2.5. Control remoto de dispositivos astronómicos . . . . .	19
2.5.1. INDI . . . . .	20
2.6. Hardware Libre . . . . .	20
2.6.1. Arduino . . . . .	22
2.6.2. Raspberry Pi . . . . .	24
2.7. Enfocadores astronómicos: estado del arte . . . . .	26
2.7.1. Opetec . . . . .	27
2.7.2. Robofocus . . . . .	27
<b>3. Objetivos</b>	<b>31</b>
<b>4. Metodología de Trabajo, Planificación y Costes</b>	<b>37</b>
4.1. Metodología de Trabajo . . . . .	37

4.2. Fases . . . . .	39
4.3. Temporización . . . . .	41
4.3.1. Estimación de tiempos . . . . .	41
4.3.2. Tiempos invertidos . . . . .	42
4.4. Recursos humanos . . . . .	43
4.5. Presupuesto . . . . .	43
<b>5. Módulo hardware</b>	<b>45</b>
5.1. Análisis . . . . .	45
5.1.1. Requisitos funcionales . . . . .	45
5.1.2. Requisitos no funcionales . . . . .	46
5.2. Planificación . . . . .	46
5.2.1. Mockup . . . . .	48
5.3. Diseño sistema hardware y electrónica: arquitectura del sistema	48
5.3.1. Núcleo: Arduino . . . . .	49
5.3.2. Módulo de motores . . . . .	51
5.3.3. Módulo de pantalla . . . . .	53
5.3.4. Módulo de sensores . . . . .	53
5.3.5. Módulo de control manual . . . . .	54
5.4. Presupuesto . . . . .	55
5.5. Implementación hardware . . . . .	55
5.5.1. Prototipos . . . . .	56
5.6. Diseño de la carcasa y piezas mecánicas . . . . .	58
5.7. Pruebas sobre el dispositivo hardware . . . . .	60
<b>6. Módulo firmware</b>	<b>65</b>
6.1. Diseño y Análisis . . . . .	65
6.1.1. Requisitos funcionales . . . . .	65
6.1.2. Requisitos no funcionales . . . . .	66
6.2. Arquitectura firmware . . . . .	68
6.3. Módulo de control de motores . . . . .	68
6.4. Módulo de control remoto . . . . .	70
6.4.1. Implementación del protocolo serie . . . . .	70
6.5. Bibliotecas adicionales . . . . .	72
6.6. Herramientas utilizadas . . . . .	72
6.7. Organización del código fuente . . . . .	73
6.8. Pruebas . . . . .	75
<b>7. Módulo Driver INDI</b>	<b>79</b>
7.1. Breve introducción a INDI . . . . .	79
7.1.1. Drivers, Servidores y Clientes INDI . . . . .	80
7.1.2. Propiedades INDI . . . . .	81
7.2. Diseño del Driver INDI . . . . .	83
7.2.1. Requisitos funcionales . . . . .	83

7.2.2. Requisitos no funcionales . . . . .	83
7.2.3. Arquitectura INDI for Java . . . . .	84
7.3. Implementación . . . . .	86
7.4. Instalación servidor . . . . .	88
7.4.1. Crear imagen Rasbindi . . . . .	90
7.5. Pruebas . . . . .	91
<b>8. Módulo Software</b>	<b>97</b>
8.1. Motivación . . . . .	97
8.2. Adquisición de Imágenes . . . . .	98
8.3. Herramienta de visualización y Análisis . . . . .	98
8.3.1. Planificación y temporización . . . . .	99
8.3.2. Análisis de requisitos . . . . .	100
8.3.3. Casos de uso . . . . .	101
8.3.4. Diagrama de casos de uso . . . . .	111
8.4. Diseño e implementación . . . . .	111
8.4.1. Diseño de clases . . . . .	111
8.4.2. Diseño de interfaz de usuario . . . . .	114
8.5. Algoritmos de detección . . . . .	114
8.6. Evaluación nivel de enfoque . . . . .	117
8.7. Enfoque . . . . .	119
8.7.1. Algoritmo búsqueda a ciegas . . . . .	120
8.7.2. Algoritmo autofocus básico . . . . .	123
8.7.3. Algoritmo autofocus para evitar problemas de seeing .	123
8.7.4. Algoritmo autofocus para evitar problemas de backlash	126
8.8. Métodos de simulación . . . . .	126
<b>9. Documentación y Difusión</b>	<b>129</b>
9.1. Página web del proyecto . . . . .	129
9.2. Canales en redes sociales . . . . .	130
9.3. Congresos Astronómicos . . . . .	130
9.4. Mantenimiento del repositorio GitHub . . . . .	131
<b>10. Conclusiones y trabajos futuros</b>	<b>135</b>
10.1. Trabajos futuros . . . . .	137
<b>A. Diagramas circuito</b>	<b>143</b>
<b>B. Diagramas de Gantt</b>	<b>149</b>
<b>C. Diseño Carcasa</b>	<b>153</b>
<b>D. Pruebas</b>	<b>157</b>

# Índice de figuras

1.1.	Ardufocuser: logo del proyecto . . . . .	1
1.2.	Ardufocuser: project logo . . . . .	2
2.1.	Hype Cycle for Emerging Technologies . . . . .	6
2.2.	Observatorio Amateur . . . . .	7
2.3.	Galaxia M88 . . . . .	9
2.4.	Sierra Nevada Night Sky Time Lapse . . . . .	11
2.5.	Telescopio . . . . .	12
2.6.	Diagrama telescopio reflector . . . . .	12
2.7.	Diagrama telescopio refractor . . . . .	13
2.8.	Cámara CCD . . . . .	14
2.9.	Montura . . . . .	14
2.10.	Rueda portafiltros . . . . .	15
2.11.	Cúpula . . . . .	16
2.12.	Enfocador modelo Tecnosky . . . . .	17
2.13.	Cabecera FITS . . . . .	18
2.14.	Observatorio Carsten Schmitt . . . . .	21
2.15.	Arduino . . . . .	22
2.16.	Diagrama Arduino . . . . .	22
2.17.	Interfaz de programación de Arduino . . . . .	25
2.18.	Raspberry Pi . . . . .	26
2.19.	Opetec Original TCF-S . . . . .	27
2.20.	Opetec Focus Lock . . . . .	28
2.21.	Kit Robofocus . . . . .	29
3.1.	Diagrama de los Objetivos del Proyecto . . . . .	34
4.1.	Iterativa basada en prototipos modulares . . . . .	38
4.2.	Iterativa basada en prototipos modulares . . . . .	40
4.3.	Evolución desarrollo de los módulos . . . . .	41
5.1.	Mockup del dispositivo hardware . . . . .	48
5.2.	Diagrama general, sistema hardware . . . . .	49
5.3.	Diagrama de pines de Arduino . . . . .	51

5.4.	Diagrama Pololu A4988 . . . . .	52
5.5.	Diagrama conexión pantalla LCD con interfaz I2C . . . . .	53
5.6.	Diagrama sensor de temperatura y humedad DHT11 . . . . .	54
5.7.	Diagrama de conexiones Wii Nunchuck . . . . .	54
5.8.	Diseño circuito con Frizing . . . . .	56
5.9.	Fase de soldadura de la placa . . . . .	57
5.10.	Foto del primer prototipo . . . . .	57
5.11.	Foto de la botonera manual . . . . .	58
5.12.	Prototipo en caja de aluminio . . . . .	59
5.13.	Foto de la carcasa en metacrilato . . . . .	60
6.1.	Gráfico de flujos de ejecución . . . . .	69
6.2.	IDE Arduino . . . . .	73
6.3.	Ejecución tests Travis CI . . . . .	78
7.1.	INDI Logo . . . . .	80
7.2.	Pipeline INDI . . . . .	81
7.3.	Arquitectura Servidor - Driver INDI . . . . .	82
7.4.	Diagrama de clases Driver INDI . . . . .	85
7.5.	Servidor INDI funcionando en Raspberry Pi . . . . .	89
7.6.	Diagrama completo del sistema Ardufocuser . . . . .	90
7.7.	Pruebas desde KStars . . . . .	91
7.8.	Pruebas desde Observatorio Remoto . . . . .	92
7.9.	Conectando el servidor INDI con el driver Ardufocuser . . . . .	93
7.10.	Conectando el Ardufocuser a través de Kstars . . . . .	94
8.1.	Logo Astropy . . . . .	98
8.2.	Ginga: Image Viewer and Toolkit . . . . .	99
8.3.	Diagrama de casos de uso. . . . .	112
8.4.	Diagrama de clases . . . . .	113
8.5.	Mockup interfaz . . . . .	114
8.6.	Aplicación Star Focusing . . . . .	115
8.7.	Mapa de calor de una estrella . . . . .	116
8.8.	Ajuste Gaussiana de una Estrella . . . . .	117
8.9.	Grafica FWHM . . . . .	118
8.10.	CCD ATIK-314L . . . . .	120
8.11.	Diagrama operaciones en el sistema . . . . .	121
8.12.	Algoritmo - A Ciegas . . . . .	122
8.13.	Algoritmo de autofocus básico . . . . .	124
8.14.	Algoritmo de autofocus (seeing) . . . . .	125
8.15.	Algoritmo de autofocus (backlash) . . . . .	127
9.1.	Poster presentando el proyecto en AstroAlcalá 2016 . . . . .	133
A.1.	Versión 0 del dispositivo . . . . .	144

A.2. Versión 1 del dispositivo . . . . .	145
A.3. Versión 2 del dispositivo . . . . .	146
A.4. Versión 3 del dispositivo . . . . .	147
B.1. Diagrama de Gantt Teórico . . . . .	150
B.2. Diagrama de Gantt Real . . . . .	151
C.1. Plano para cortar la carcasa con láser CNC . . . . .	154
C.2. Vista tridimensional de la carcasa . . . . .	155
D.1. Circuito para las pruebas lectura de valores potenciómetros .	158
D.2. Circuito para la prueba mover motor paso a paso . . . . .	159
D.3. Circuito para la prueba Wii nunchuck . . . . .	161

# Índice de tablas

2.1. Comandos Robofocus . . . . .	28
4.1. Costes y gastos requeridos en el desarrollo del proyecto: se comprueba que el mayor coste se encuentra en los gastos derivados del equipo humano. . . . .	44
5.1. Lista de componentes y costes . . . . .	55
5.2. Caso de prueba, lectura de potenciómetros . . . . .	61
5.3. Caso de prueba, mover motor a velocidad constante . . . . .	61
5.4. Caso de prueba, detectar pulsaciones en Wii Nunchuck . . . . .	62
5.5. Caso de prueba, temperatura de motor tras un tiempo de funcionamiento prolongado . . . . .	63
6.1. Comandos Ardufocuser . . . . .	71
6.2. Caso de prueba, ajustar posición enfocador mediante API serie. . . . .	76
6.3. Caso de prueba, apagar iluminación LCD mediante API serie. . . . .	76
6.4. Caso de prueba, los estados son persistentes de una sesión para otra . . . . .	77
7.1. Propiedades del Driver INDI del Ardufocuser . . . . .	84
7.2. Caso de prueba, conectar con servidor INDI . . . . .	92
7.3. Caso de prueba, conectar Ardufocuser . . . . .	93
7.4. Caso de prueba, modificar velocidad . . . . .	94
7.5. Caso de prueba, establecer posición del enfocador . . . . .	95
8.1. CU-1. Abrir imagen FITS. . . . .	103
8.2. CU-2. Modificar Gamma. . . . .	103
8.3. CU-3.Modificar ecuación Histograma de la Imagen. . . . .	104
8.4. CU-4.Modificar Zoom. . . . .	105
8.5. CU-5.Generar Gráfico luz de zona. . . . .	105
8.6. CU-6. Activar detección de objetos.. . . . .	106
8.7. CU-7. Generar Gráfico luz de objeto. . . . .	107
8.8. CU-8.Aplicar filtro margen. . . . .	107
8.9. CU-9.Aplicar filtro contraste. . . . .	108

8.10. CU-10.Aplicar filtro distancia. . . . .	109
8.11. CU-11.Aplicar filtro FWHM. . . . .	109
8.12. CU-12.Aplicar número de objetos. . . . .	110
8.13. CU-13. Aplicar filtro ajuste Gauss.. . . . .	111

# Índice de Fragmentos de Código

6.1.	Ejemplo de uso de AccelStepper, biblioteca usada para controlar motores paso a paso . . . . .	69
6.2.	Ejemplo de uso de la biblioteca SerialCommand . . . . .	72
6.3.	Núcleo implementación firmware ardufocuser . . . . .	74
6.4.	Script travis para realizar integración continua . . . . .	77
7.1.	Ejemplo biblioteca <b>SerialCommand</b> . . . . .	86
7.2.	Ejemplo iniciar una propieda INDI Numérica . . . . .	87
7.3.	Script de inicio del servidor INDI . . . . .	88
7.4.	Script de inicio del servidor INDI (2) . . . . .	89
D.1.	Pruebas lectura de valores potenciómetros . . . . .	157
D.2.	Prueba mover motor paso a paso . . . . .	158
D.3.	Prueba Wii nunchuck . . . . .	160

# Capítulo 1

## Resumen

De acuerdo con la normativa vigente para los Trabajos de Fin de Grado a continuación se presentan las palabras clave del proyecto así como el resumen breve del mismo, tanto en español como en inglés.

### 1.1. Breve resumen y palabras clave

**Palabras clave:** *Arduino, hardware libre, astronomía, robotización de telescopios, INDI, software libre, driver, internet of things, procesamiento imagen.*



Figura 1.1: Ardufocuser: logo del proyecto

Hoy día ya contamos con numerosas plataformas de control y automatización de dispositivos astronómicos, aunque en muchos casos se relacionan con grandes compañías privadas, y ello implica numerosos derechos intelectuales, el inconveniente de no encontrar detalles técnicos, libertad para conocer el funcionamiento y por tanto no poder realizar modificaciones a voluntad.

El objetivo principal de este proyecto es diseñar e implementar un sistema completo de enfoque automático, que permita acoplarse fácilmente a un telescopio y realizar una configuración de sus lentes o espejos y que consiga el mejor enfoque de imágenes estelares.

Para ello debemos hacer un recorrido que pase por el diseño y programación a bajo nivel, implementación hardware, protocolos de comunicación con los clientes, métricas y heurísticas para evaluar la calidad de las imágenes, así como los distintos algoritmos para encontrar los máximos deseados.

Al introduciéndonos en el proyecto nos hemos encontrado un amplio y rico abanico de retos y problemas relacionados con el diseño, la computación, algorítmica, desarrollo software y protocolos de comunicación.

Uno de los retos más grandes y amplios al que nos enfrentamos en este proyecto es acercar hardware y software libre a la astronomía, creando herramientas avanzadas con un bajo presupuesto, alcance libre de todas sus interioridades, detalles técnicos, así como la libertad para mejorar y evolucionar el producto por todo aquel interesado.

Por tanto, pretendo que este proyecto además de ser un producto perfectamente funcional, sea una semilla para multitud de nuevos proyectos en el contexto de la astronomía y el software y hardware libre.

## **1.2. Extended abstract and key words**

### **Ardufocuser: An Automatic Astronomical Focusing Solution**

José Miguel López Pérez

**Key words:** *Arduino, free hardware, astronomy, robotic telescopes, INDI, free software, driver, internet of things, image processing.*



Figura 1.2: Ardufocuser: project logo

Nowadays we can find several different solutions to control and automate astronomical devices. However, those solutions are usually developed by big private companies and thus their products do have many patents and copyright issues that hinder their technical details and restrict users freedom to know the inner workings of their devices thus avoiding the possibility of improving, modifying and repairing them.

The main objective of this project is to design and implement a complete automatic focusing solution. It should be easily attached to a telescope allowing to correctly align its lenses or mirrors to obtain the best possible focus for stellar imaging.

To do so we must cover several different areas including lower level programming, make hardware implementations, develop communication protocols with client software, design different metrics and heuristics to measure image quality and even design different algorithms to obtain the best focused images.

Once we have started the project we have found a rich and big amount of challenges and problems related to the design of the platform, computation, algorithms, software development and even communication protocols.

One of the main and broader challenges in this project is to narrow the distance among free hardware and software to the astronomy field by creating some advanced tools with a lower cost, completely free specs and documentation, technical details and, more important, the freedom to improve and evolve the product by anyone who is interested.

Therefore, apart from developing a perfectly functional product, I pretend it to be a seed to many other different projects which bring together free software and hardware and astronomy.



## Capítulo 2

# Introducción y motivación

Toda observación de cualquier fenómeno astronómico, requiere de múltiples trabajos laboriosos y operaciones de control, para cada componente del observatorio. Ya sea la montura del telescopio (sección 2.2.3), la cámara CCD (sección 2.2.2), el enfocador (sección 2.2.7), la rueda de filtros (sección 2.2.4) etc., asumiendo una pérdida de tiempo útil y el riesgo que se puedan producir errores humanos o accidentes, que pongan en peligro la noche de observación.

Muchas de las operaciones siguen patrones claros y tienen relación con el estado de un sensor o periférico, permitiendo definir rutinas automáticas que agilicen los trabajos inherentes a la observación.

Un ejemplo claro puede ser la decisión de abrir o cerrar la cúpula en un momento dado en función de los datos proporcionados por la estación meteorológica (sección 2.2.6).

En el mercado existen ya gran cantidad de recursos que se encargan de automatizar estas tareas, sin embargo, el acceso a estos recursos es caro y no todo aficionado a la astronomía puede permitírselo.

Otro punto en contra de estas soluciones, es que suelen ser muy cerradas. Las compañías ofrecen muy pocos detalles técnicos (los justos para poder instalarlos e integrarlos), no permiten ninguna modificación y deben funcionar exactamente bajo la plataformas definidas, en la mayoría de los casos con el sistema operativo Windows.

El presente proyecto se puede englobar dentro de las soluciones “Internet of Things” que hoy día están en una de las fases de expectativas más alta, como podemos observar en el *Hype Cycle for Emerging Technologies* de Gartner [102] (figura 2.1).

Para poner en contexto el proyecto, en las siguientes secciones se propone



Figura 2.1: Hype Cycle for Emerging Technologies, 2015. Fuente: [102]

un recorrido por algunas de las ciencias, tecnologías y disciplinas que están involucradas en el desarrollo del presente TFG:

- Astronomía
- Instrumental astronómico
- Software astronómico
- Formato imágenes astronómicas
- Control remoto de observatorios
- Hardware libre
- Enfocadores astronómicos: estado del arte

## 2.1. La Astronomía

Significa literalmente el estudio de las **leyes** que rigen los **astros** o cuerpos celestes, según su etimología griega y latina.

Se define más formalmente como [17]:

*“Ciencia que se ocupa del estudio de los cuerpos celestes del universo, incluidos los planetas y sus satélites, los cometas y meteo-*



Figura 2.2: **Astrónomo amateur** en plena observación, utilizando telescopio y prismáticos. **Fuente:** [12]

*ritos, las estrellas y la materia interestelar, los sistemas de materia oscura, estrellas, gas y polvo llamados galaxias y los cúmulos de galaxias; por lo que estudiaba sus movimientos y fenómenos ligados a ellos.”*

Es una de las ciencias más remotas por su impacto visual, emocional y su gran utilidad en la agricultura. Captó la atención de nuestros antepasados, motivándolos al estudio de los objetos del firmamento y su movimiento, fenómeno que muchas veces no conseguían explicar por completo, y por eso los llegaban a divinizar en múltiples culturas.

Por tanto la astronomía es una ciencia antigua y moderna a la vez: Antigua porque se remonta prácticamente al origen de la humanidad; Moderna por proporcionar uno de los campos de estudio e investigación más avanzados.

Se trata de una ciencia donde aún perduran muchos interrogantes como por ejemplo, si existe vida fuera de la Tierra o si somos la única civilización inteligente [12].

### 2.1.1. Astronomía Antigua

Civilizaciones tan antiguas como la asiria, babilónica o sumeria, ya comenzaron a transmitirnos los primeros conocimientos sobre el universo que conocemos gracias a la difusión realizada por la cultura griega [45].

Tienen especial importancia los conocimientos astronómicos de los egipcios,

dado que para ellos el estudio del cielo y elaborar su **calendario egipcio** [21] les era de vital importancia, porque les permitía controlar los ciclos de la agricultura y prever la gran inundación del río Nilo, que sigue ciclos anuales.

En el Nuevo Mundo, los mayas llegaron a alcanzar importantes conocimientos de los cuerpos celestes y elaborando un calendario bastante preciso.

Los incas se consideraban a sí mismos descendientes del Sol y los aztecas adoraban al dios **Huitzilopochtli**, símbolo del Sol que amanecía cada mañana para hacer la lucha con sus hermanas, las estrellas y así imponer su reinado diurno.

En el siglo III a.C, el astrónomo griego **Aristarco de Samos** [10], puso en duda todo el modelo geocéntrico griego y postuló que la Tierra gira en 24 horas y se traslada en torno al Sol en un año. Realizando también dibujos de las órbitas planetarias en el orden que ahora las conocemos.

**Pitágoras** [72] en el siglo VI a.C. ya tenía ideas sobre los movimientos de rotación Terrestre y de traslación en torno al Sol, así como conocimiento de la esfericidad de la Tierra, Luna y Sol [30].

### 2.1.2. Astronomía Moderna

La Astronomía moderna [55] inicia su desarrollo con **Nicolás Copérnico (1473-1543)** [67], quien el año de su muerte publica su trabajo más importante **De revolutionibus orbium caelestium**.

Se mantiene que la Tierra tiene un doble movimiento: de rotación sobre ella misma, en 24 horas, y de revolución alrededor del Sol, en un año. También establece movimientos similares para los planetas y satélites.

Una figura muy importante es **Tycho Brahe (1546-1601)** [93] con sus observaciones, facilitó a su discípulo **Johannes Kepler (1571-1630)** [52], el descubrimiento de las famosas leyes que rigen el movimiento de los planetas, el abandono de las órbitas circulares y la ruptura definitiva con unos conceptos tradicionales que estaban profundamente arraigados.

**Galileo Galilei (1564, 1642)** [38] fue un astrónomo, filósofo, ingeniero, matemático y físico italiano. En 1609 construye su primer telescopio, motivado por el rumor de la existencia de un telescopio fabricado en Holanda. Su telescopio no deforma los objetos y es capaz de aumentarlos 6 veces. Tras muchas versiones consigue aumentar los objetos hasta 20 veces. Con su instrumento realiza numerosos descubrimientos de la morfología de la Luna y del universo en general.

**Isaac Newton (1643-1727)** [51] publica **los Principia** en 1685 [71] y explica una serie de fenómenos naturales como las estaciones del año, las

mareas, los movimientos de los astros, mediante un conjunto de leyes que podían ser probadas en un laboratorio. En este punto la **Astronomía** [17] y **Astrología** [16] inician caminos diferentes y desde entonces no tienen ningún punto en común.

Durante el siglo XVIII tienen lugar aportaciones importantes en el campo de la astronomía observacional para el estudio del Universo. **Charles Messier (1730-1817)** [56], presentó en la Academia de Ciencias de Francia en 1771 el primer catálogo de estrellas y asociaciones de cúmulos estelares, descubiertas u observadas por él.

El descubrimiento de la **fotografía** produjo un rápido avance en la aplicación de la astronomía. En 1863 Huggins [96], obtiene los primeros espectros estelares [5] abriendo una nueva era en la Astronomía.

Desde finales del siglo XIX y principios del XX la Física pasa a desempeñar un papel decisivo en la interpretación de los fenómenos astronómicos. La Astrofísica [14], adquiere una progresiva importancia sobre la astronomía clásica, utilizando en la actualidad ambos términos de forma casi sinónima.

### 2.1.3. Astronomía en la actualidad

Viendo el recorrido de esta ciencia en el pasado, nos preguntamos por su repercusión en la sociedad presente. Poniendo en valor los beneficios que tiene su estudio y los interrogantes que a día de hoy científicos de todo el mundo trabajan por dar respuesta [103].

Los beneficios que obtiene nuestra sociedad, los podemos catalogar según su alcance:

- **Puramente científico**, el impulso por el conocimiento. Responder interrogantes existenciales como: “¿De dónde venimos?. ¿A dónde vamos?”.



Figura 2.3: Galaxia en espiral **M88**, a 49 millones de años luz, descubierta por **Charles Messier** en 1781. **Fuente:** [34]

- **La expansión del Universo, el Big Bang, Agujeros Negros.** Conceptos cosmológicos complejos que poco a poco llegan a la sociedad con el esfuerzo de los investigadores a lo largo de décadas, junto con el desarrollo de instrumentos cada vez más sofisticados, para llegar a nuevas conclusiones.
- **Desarrollo tecnológico**, que posteriormente han tenido aplicaciones directas en la sociedad. Por ejemplo: [101]
  - **Detectores CCD** (10.1), que usan nuestras cámaras de fotos fueron inventados en 1969, pero lograron desarrollarse rápidamente gracias a sus aplicaciones en instrumentos astronómicos [29].
  - **Detectores de rayos X**, que existen en los aeropuertos fueron desarrollados para aplicar los conocimientos adquiridos en los detectores de rayos X para instrumentos astronómicos [79].
  - **Televisión satélite**, con la que estamos familiarizados y usamos a diario en nuestra vida cotidiana [92].

#### 2.1.4. Astronomía amateur

La astronomía **amateur** es la realizada por astrónomos no profesionales, normalmente sin formación reglada en la materia, y que su interés primordial esta en aprender, conocer esta ciencia, asistir a charlas o quedadas astronómicas y compartir esta afición con otras personas.

La labor de este conjunto es muy valorada, dado que suelen compartir sus trabajos, colaborando con asociaciones y nutriendo de numeroso material muy variopinto: observaciones desde múltiples ubicaciones, momentos en el tiempo o equipo de observación diferente, ampliando así la cantidad de muestras. No puede en este caso llevar más razón el refranero español: “*Más ven cuatro ojos que dos*”.

Todo este trabajo posteriormente puede ser organizado y estudiado por profesionales y para sacar conclusiones valiosas.

Además, muchos astrónomos aficionados, dedican tiempo y energías imparatiendo conferencias divulgativas que acercan esta ciencia, sus métodos y sus conclusiones al gran público.

Otro de los perfiles más visuales en el mundo de la astronomía amateur es el de la **astrofotografía** [13] (figura 2.4), que consiste en la captación fotográfica de las imágenes de cuerpos celestes, teniendo gran valor artístico en muchos casos.

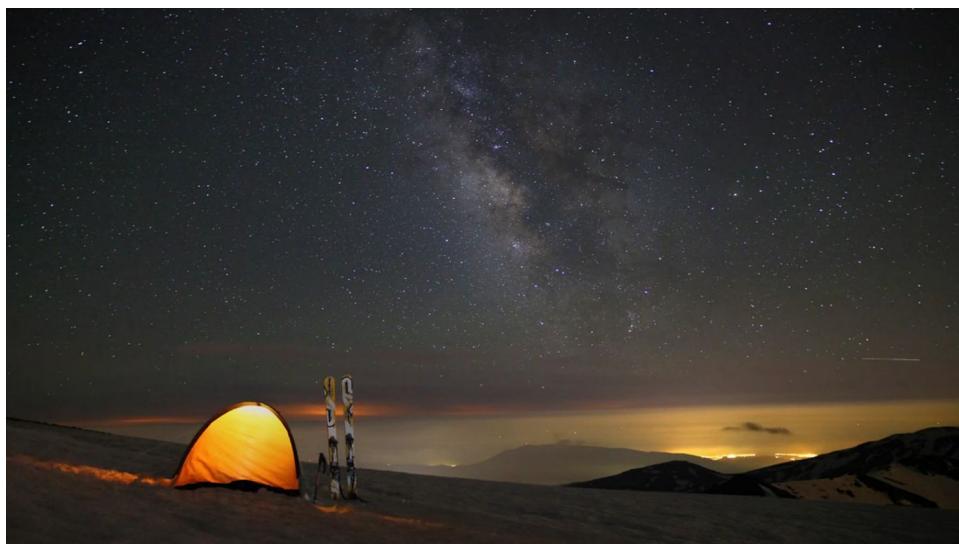


Figura 2.4: **Isidro Villo - Sierra Nevada Night Sky Time Lapse**, fotograma del vídeo tomado entre junio y agosto de 2011 en Sierra Nevada en memoria de su compañero Iker Canales Onaindia, muestra el lado más creativo y artístico de la astronomía. **Fuente:** [86]

En muchas ocasiones, la frontera entre astrónomos profesionales y amateur es muy tenue porque ambos contribuyen de manera destacada al conocimiento del cielo.

## 2.2. Instrumental Astronómico

La astronomía está intimamente relacionada con una serie de herramientas e instrumentos que permiten expandir los sentidos del observador, permitiendo llegar a ver objetos más lejanos y tenues y apreciar más características de ellos. A continuación se describen algunos de estos instrumentos.

### 2.2.1. Telescopios

El telescopio es un instrumento que básicamente recoge la mayor cantidad posible de luz emitida por un objeto situado fuera de la atmósfera y la concentrar para así permitir la detección de imágenes que a simple vista son inapreciables [91] (figura 2.5).

Es una herramienta fundamental en astronomía, y cada mejora de este instrumento ha permitido avances en la comprensión del Universo.



Figura 2.5: Vista de un telescopio refractor **Fuente:** [91]

Debemos agradecer este instrumento en gran parte a **Galileo** [38], cuyos avances permitieron usar el aparato como instrumento astronómico.

Existen dos grandes tipologías entre los telescopios, según el tipo de sistema óptico que utilizan: los **reflectores** y los **refractores**.

**Los reflectores** (figura 2.6) se constituyen de un espejo principal (espejo primario u objetivo), el cual no es plano como los espejos convencionales, sino que es provisto de cierta curvatura que le permite concentrar la luz en un punto.

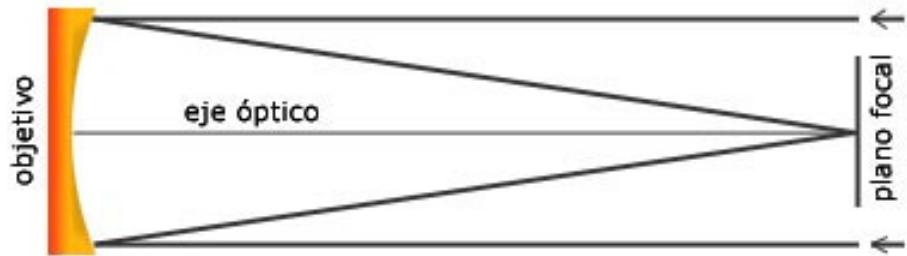


Figura 2.6: Diagrama funcionamiento telescopio reflector  
**Fuente:** [18]

**Los refractores** (figura 2.7) poseen como objetivo una lente (o serie de lentes, la cantidad varía según el diseño y calidad) que de forma análoga al

funcionamiento de una lupa, concentran la luz en el plano focal.

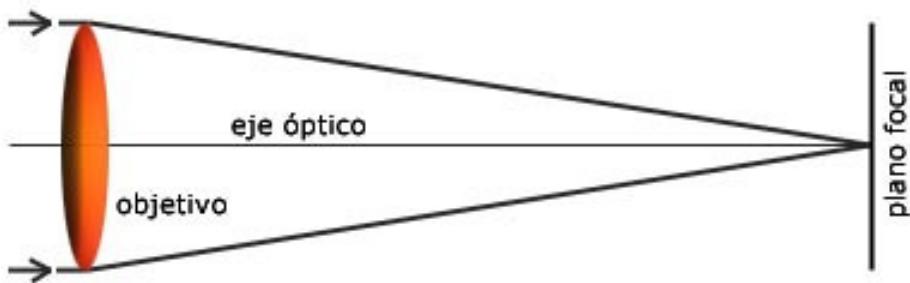


Figura 2.7: Diagrama funcionamiento telescopio Refractor

Fuente:[18]

### 2.2.2. Cámaras CCD

Un *dispositivo de carga acoplada*, conocido también como **CCD** (figura 2.8) es un circuito integrado que contiene un número determinado de condensadores enlazados bajo el control de un circuito interno. Se encarga de convertir la señal luminosa en una señal eléctrica, componiendo una imagen digital.

La capacidad de resolución de la imagen, depende del número de células foteléctricas del CCD. A mayor número de píxeles, mayor nitidez en relación con el tamaño. Actualmente las cámaras fotográficas digitales incorporan CCD con capacidades de hasta 160 megapíxeles [29].

### 2.2.3. Monturas

La montura de un telescopio (figura 2.9) es la parte mecánica que une el trípode o base al instrumento óptico o telescopio. Existen varios tipos de monturas, algunas muy simples, otras mas complejas. Hoy en día son comunes las que tienen correctores electrónicos y dispositivos de localización y seguimiento sofisticados (sistemas **GOTO**).

La montura tiene como objetivo proveer de movimiento controlado al telescopio [63].

La más simple es la montura **altacimutal**, que realiza movimientos horizontales y verticales. Este tipo de diseño lo traen incorporados los telescopios pequeños, por lo general **telescopios refractores** de uso terrestre, dado que su uso es simple, y también varios modelos de equipos automatizados.



Figura 2.8: Cámara CCD modelo Starlight Xpress. **Fuente:**[25]



Figura 2.9: Montura robotizada modelo SkyWatcher EQ-8 **Fuente:** [64]

Le sigue la **montura ecuatorial**, que utiliza como plano fundamental el ecuador celeste (proyección del ecuador terrestre). Este diseño usa las coordenadas ecuatoriales, ascensión recta (A.R. o R.A.) y declinación (Dec.), que son proyecciones de las coordenadas terrestres longitud y latitud, respectivamente, sobre la esfera celeste.

#### 2.2.4. Rueda portafiltros

La rueda porta-filtros (figura 2.10), consiste en un cuerpo generalmente de aluminio, que en su interior puede alojar varios **filtros**. El tamaño de los filtros así como el número de los mismos que puede alojar una rueda porta-filtros varía según los distintos modelos existentes.

Unos filtros comunes son los de colores, utilizados para resaltar las características de los objetos observados, sobre todo la atmósferas y superficies de los planetas.

Gracias a los filtros, podemos fragmentar el espectro de luz dejando pasar luz de una determinada longitud de onda infiriendo a partir de estas mediciones características de los objetos observados.



Figura 2.10: Rueda portafiltros **Fuente:** [82]).

#### 2.2.5. Cúpulas

Las **cúpulas** (figura 2.11) son recintos cerrados más o menos grandes que nos permiten albergar y proteger el instrumental astronómico. De esta forma, las **cúpulas** pueden ser abiertas o cerradas para exponer los instrumentos en el momento de las observaciones.

#### 2.2.6. Estaciones meteorológicas

Las **estaciones meteorológicas** son sistemas compuestos por un conjunto de sensores que nos proporcionan datos de las distintas magnitudes meteorológicas, tales como la temperatura, humedad, presión barométrica,



Figura 2.11: Cúpula - **Fuente:** [26].

presencia de nubes, viento, etc... permitiéndonos generar modelos a partir de los cuales conocer la situación climática y su posible evolución.

Gracias a los datos aportados por las **estaciones meteorológicas**, podemos conocer la climatología en el momento de realizar observaciones astronómicas. De esta forma podemos decidir si las condiciones son óptimas o si debemos cerrar la cúpula y abortar una observación para evitar daños en los instrumentos por lluvias o similar.

### 2.2.7. Enfocadores

El **enfocador** (figura 2.12), es una pieza fundamental del telescopio que nos permitirá ver con nitidez las imágenes formadas tras la reflexión de la luz en el espejo primario y su desviación por el espejo secundario.

El enfoque viene determinado por la convergencia de la mayor parte de los rayos justo en el plano focal, que es donde colocamos el ojo o una cámara.

El enfocador, es una pieza adaptada al telescopio, usualmente con una rueda dentada sobre una cremallera que podemos manipular para desplazar el portaocular y variar la distancia focal para encontrar el punto de foco óptimo [28].

Como en observación astronómica se suele trabajar a muchos aumentos, un pequeño error de enfoque se magnifica en una imagen poco nítida o desenfocada.

Para ayudar en esta operación los astrónomos han perfeccionado algunas

técnicas, una de las más extendidas es hacer uso de una **máscara de enfoque**, [97] que consiste en unas rendijas por las que hacemos pasar la luz de un objeto luminoso, difractando los rayos y observando la dirección que toman.

Otra característica que incorporan muchos enfocadores comerciales es la **compensación por temperatura**, para ello incorpora un sensor térmico en la óptica que informa de oscilaciones en la temperatura (que puedan producir dilatación en la lente o tubo). Cuando se detecta una oscilación considerable en la temperatura, se ejecuta un ciclo de autoenfoque o se compensa según alguna regla establecida.



Figura 2.12: Enfocador modelo Tecnosky- **Fuente:** [32]

Otra técnica más reciente pero muy extendida entre los astrónomos, es el uso de **software** especializado, que mediante procesamiento de imágenes es capaz de obtener una medida de la nitidez del objeto observado con el fin de establecer algoritmos automáticos de enfoque.

### 2.3. Software astronómico

Según su utilidad podemos distinguir y catalogar el software astronómico en los siguientes conjuntos básicos:

- **Planetarios y cartas celestes:** Permiten orientarnos en el cielo, permitiendo así reconocer estrellas, planetas, galaxias, constelaciones. Tiene conexión con bases de datos de todos estos objetos, teniendo información en tiempo real, de su posición. Un ejemplo es **Stellarium** [87] o **Google Sky** [42].
- **Alineación y Guiado:** Su función es calibrar y sincronizar los elementos del observatorio (entre ellos y con el movimiento del cielo),

para ello también pueden incorporar un “planetario”. Ejemplos de este tipo de software son **KStars** [53] (solución libre) y **MaxIm DL** [60].

- **Captura y procesado de imágenes:** se encargan especialmente de capturar imágenes y procesarlas de forma automática. Podemos destacar **ImagesPlus** (privativo), **Deep Sky Stacker** y **Ginga** (software libre) [40].

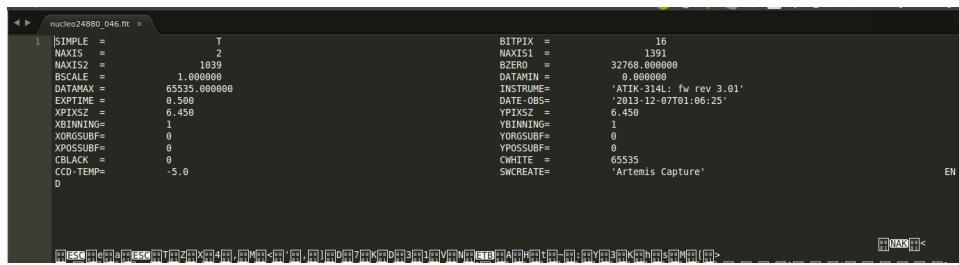
## 2.4. Imágenes Astronómicas

En las imágenes astronómicas interesa almacenar la máxima cantidad de información sobre la imagen, es por ello que siempre se usan formatos **sin compresión o con compresión sin pérdidas**, como puede ser RAW [78] o FITS (figura 2.13) [35].

**FITS** es un formato de imágenes especialmente concebido para el mundo de la astronomía, por permitir almacenar información más allá de la visible, así como espectros electromagnéticos.

Una característica muy interesante para los astrónomos, es la incorporación de cabeceras en texto plano y legibles sin software adicional: en esta cabecera se introducen **metadatos** sobre la observación realizada, posición geográfica, marcas de tiempo, características de la cámara, filtros empleados, etc.

FITS está soportado por bibliotecas disponibles en los lenguajes más utilizados en el ámbito científico, incluyendo C, FORTRAN, Java, Perl, PHP, Python.



```

nucleo24880_046.fits

BITPIX = 16
MAXISI = 1391
BZERO = 32768.000000
DATAMIN = 0.000000
INSTRUME= 'ATIK-314L: fw rev 3.01'
DATE-OBS= '2013-12-07T01:06:25'
YPIXSZ= 6.450
YDELTANG= 1
XORGSUBF= 0
YORGSUBF= 0
XPOSUSBF= 0
YPOSUSBF= 0
CBLACK = 0
CMWHITE = 65535
SWCREATE= 'Artemis Capture'

EN

```

Figura 2.13: **Cabecera FITS.** Observamos como se puede leer la cabecera de una imagen en formato FITS usando un editor de texto.

Además existen numerosos entornos de procesamiento de imágenes que permiten manipular este tipo de imágenes, como por ejemplo **ImageJ** [48].

## 2.5. Control remoto de dispositivos astronómicos

En la actualidad se están implantando diversos protocolos y estándares de control remoto al campo de la astronomía para agilizar y facilitar las observaciones. Estos estándares pretenden liberar al astrónomo de tareas tediosas controlar la seguridad en la operación de los instrumentos (por ejemplo condiciones climatológicas adversas) y permitirle centrarse en la propia observación, que puede realizarse desde cualquier parte del mundo y a cualquier hora.

También permite multiplicar el número de puestos, dado que un equipo de astrónomos puede controlar varios observatorios en remoto, mientras que de forma física se ven limitados a uno solo.

Existen diversas formas de controlar los dispositivos astronómicos pero la mayoría presenta los mismos inconvenientes:

- Normalmente se controlan los dispositivos directamente: conectando cada dispositivo a un PC y se trabaja desde dicho ordenador.
- En ocasiones se utilizan herramientas para el control remoto del PC como el **escritorio remoto** [33], lo que puede ocasionar algunos inconvenientes como lag excesivo o un consumo de ancho de banda alto.

La evolución de las plataformas se puede resumir como sigue:

1. **Esquema monolítico:** Cada dispositivo, funciona con **su propio cliente** y usa un **protocolo particular**.
2. **Esquema extensible:** Donde algunos **fabricantes comparten el código de control**, pero nadie de forma independiente puede implementar nuevos plugins.
3. **ASCOM**, 10.1 intenta crear una **capa intermedia entre los programas cliente y los dispositivos astronómicos**, dado que es un estándar abierto, cualquiera puede implementar nuevos driver para sus dispositivos. Como inconveniente encontramos que solo puede utilizarse en sistemas **Microsoft Windows**. Su diseño tiene una relación bastante profunda con el sistema operativo lo cual dificulta el desarrollo basado en red. [11]
4. **INDI**, es un protocolo **más abierto que el anterior y dispone de múltiples implementaciones** en C y Java [49], se puede desplegar en cualquier sistema operativo, ya sea **Windows, Linux o Mac** [46].

### 2.5.1. INDI

*“The Instrument Neutral Distributed Interface (INDI) Library is a cross-platform software designed for automation control of astronomical instruments. It supports a wide variety of telescopes, CCDs, focusers , filter wheels, etc., and it has the capability to support virtually any device. INDI is small, flexible, easy to parse, and scalable. It supports common DCS functions such as remote control, data acquisition, monitoring, and a lot more. With INDI, you have a total transparent control over your instruments so you can get more science with less time.”*

**Fuente:** [1]

El protocolo **INDI** es una plataforma software diseñada para el control de instrumental astronómico, aunque podría usarse con cualquier dispositivo, incluidos “virtuales”. La biblioteca **INDI** permite controlar cualquier dispositivo para el que se haya desarrollado un driver **INDI**. Funciona mediante el paso de paso de información en formato XML (10.1).

Sus principales ventajas frente a otras soluciones para el control de dispositivos son:

- Es una biblioteca **ligera, flexible** y **escalable**.
- Es de código abierto por lo que cualquiera puede ver su código y mejorarlo o crear drivers para cualquier dispositivo.
- El intercambio de información entre clientes, servidores y drivers es mínimo.
- Es **multiplataforma**.
- Separa clara y totalmente el cliente del servidor.
- Los fabricantes comienzan a desarrollar drivers para sus dispositivos o liberan las especificaciones para que la comunidad pueda desarrollarlos.
- Existen **numerosos clientes INDI** como kstars, **Cartes Du Ciel**, **Xephem**, **Observatorio Remoto** (Android) [68], **Stellarium** [87].

## 2.6. Hardware Libre

Se llama hardware libre a aquellos dispositivos cuyas especificaciones y esquemáticos son de acceso público, ya sea bajo algún tipo de pago, o de forma

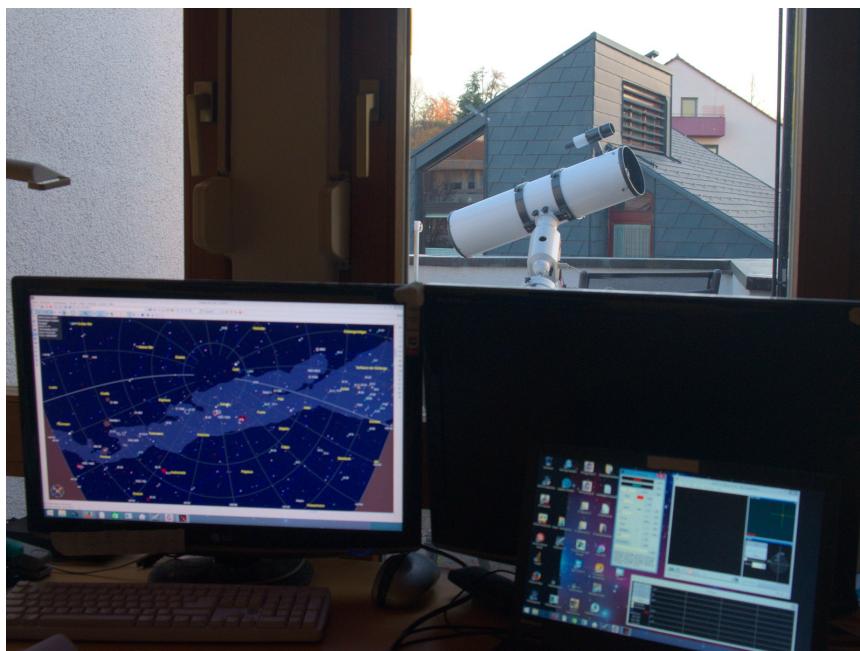


Figura 2.14: Observatorio Carsten Schmitt - (Lost Infinity), haciendo uso de control remoto y software astronómico especializado. **Fuente:** [58]

gratuita. La filosofía del software libre es aplicable a la del hardware libre, y por eso forma parte de la **cultura libre** [43].

Los problemas que trata de solventar el hardware libre son los siguientes:

1. **Conocimiento restringido**, el conocimiento lo poseen las empresas. El hardware libre trata de hacer pública toda la documentación, diálogos, data shield, fichas, etc.
2. **Falta de materiales o herramientas para la fabricación**. Tratan de utilizar componentes estándar, que se puedan encontrar fácilmente así como hacer recomendaciones de algunas tiendas online donde puedes comprar tales componentes.
3. **Altos costes de producción**. Al ser de diseño libre, diferentes fábricas pueden ocuparse de proveer los componentes, preocupándose por optimizar y agilizar el proceso, llegando a bajar los costes de producción.
4. **Gran inversión en realizar trabajos redundantes**. No hay que preocuparse por solucionar una y otra vez los mismo problemas: para nuestros diseños podemos partir de otros proyectos consolidados.

Al auspicio de este movimiento han surgido muchísimas plataformas y tecno-



Figura 2.15: Logo oficial de Arduino - **Fuente:** [9]

logías, pero en especial cabe remarcar dos de las más importantes, **Arduino** [9] y **Raspberry Pi** [76].

### 2.6.1. Arduino

**Arduino** es una compañía de hardware libre, la cual desarrolla placas de desarrollo (figura 2.16) que integran un **microcontrolador** y un entorno de desarrollo [4].

Está diseñado para facilitar el uso de la electrónica en proyectos multidisciplinares [9].

Tal como marcan los principios del hardware libre, los esquemáticos de diseño del Hardware, están disponibles bajo licencia Libre, permitiendo a cualquier persona crear su propia placa Arduino sin necesidad de comprar una prefabricada.

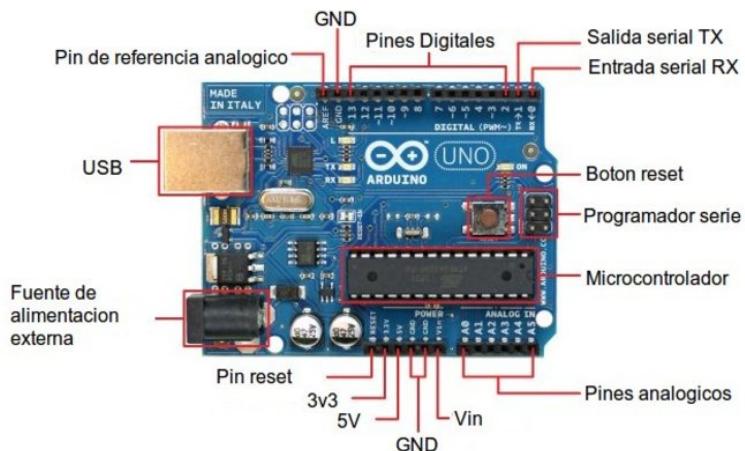


Figura 2.16: Vista principal placa Arduino, podemos diferenciar los diferentes componentes. **Fuente:** [9]

**Modelos de Arduino** Existen multitud de ediciones o modelos de placa, cada una pensada para un público concreto o para una serie de tareas o proyectos específicos. También han surgido muchos modelos no oficiales.

Por nombrar algunas placas de las más famosas:

- **Arduino UNO**: es la plataforma más extendida y la primera que salió al mercado.
- **Arduino Yun**: Combina un chip ATmega32u4 y en un chip Atheros AR9331, que se comunican mediante un puente. El chip Atheros soporta distribuciones ligeras de Linux.
- **Arduino Mega**: Incorpora un chip ATmega2560, dando más rendimiento que el ATmega320 del Arduino UNO
- **Arduino Ethernet**: Idéntico al Arduino UNO, pero con conexión a red.
- **Arduino Nano y Micro y TinyDuino**: De muy pequeño tamaño y optimizadas para mejorar el consumo.
- **Arduino LilyPad**: está pensado para insertarse en prendas y textiles, es lavable.

**Shields para Arduino** Son placas de circuitos modulares que se montan unas encima de otras para dar funcionalidad extra a las placas de Arduino.

Las shields, se comunican con Arduino usando algunos de los pines digitales/analógicos, por algún bus como el SPI 10.1, I2C 10.1 o bien por puerto serie. Además estas shields se alimentan generalmente a través del Arduino mediante los pines de 5V y GND [85].

Podemos destacar algunos de los shield más comunes y recomendables para multitud de proyectos:

- **Arduino Wifi Shield**, que añade comunicación wireless WIFI,
- **Arduino GSM Shield**, comunicación GPRS (usando una tarjeta SIM),
- **Arduino Motor Shield**, permite manejar motores DC,
- **GPS Shield**, añade localización GPS,
- **Xbee Shield**, comunicación inalámbrica XBee [99], entre muchos otros.

**Framework desarrollo Arduino** El lenguaje de programación de Arduino (basado en Wiring) está implementado en C++. Posee su propio entorno de programación (figura 2.17), con opciones para compilar y cargar el código fuente en la placa que tenemos conectada. Los programas de Arduino se llaman “sketch” [8].

También podemos encontrar multitud de bibliotecas y módulos implementados por terceros [7], que se pueden incorporar directamente a nuestros proyectos, realizando algunas abstracciones sobre el hardware, comunicación o algún aspecto de más bajo nivel.

### 2.6.2. Raspberry Pi

Raspberry Pi [76] fue lanzado en 2006 por la Fundación Raspberry Pi con el objeto de facilitar la formación de informática en las escuelas de todo el mundo, especialmente en los lugares más desfavorecidos.

Esta placa (figura 2.18), a diferencia que Arduino, es capaz de ejecutar un sistema operativo completo, incluyendo el correspondiente servidor gráfico y servicios básicos como SSH o FTP.

Actualmente contamos con la versión 3, que incorpora un procesador ARM de 4 núcleos con una velocidad de 1.2 GHz. El otro punto destacable es su GPU, capaz de llegar a resoluciones Full HD sin problema alguno. Junto a las características anteriores, cabe destacar 1 GB de memoria RAM, que permite ejecutar varias aplicaciones al mismo tiempo.

Otro punto a destacar de la Raspberry Pi es su conectividad, ya que incorpora en la misma placa el hardware necesario para dotar al sistema tanto de **Bluetooth**, de **WiFi** o de **Ethernet**.

También puede usarse para prototipado, por lo que incorpora 40 pines **GPIO** a los que se les puede conectar cualquier dispositivo sensor o actuador.

Recientemente ha aparecido una placa de la misma serie pero de reducidas dimensiones, solo 6x3 cm, a diferencia de la anterior tiene una potencia inferior y no cuenta con módulo de conexión a red.

Por su reducido coste y consumo estas plataformas son ideales para ejecutar servicios básicos, convertirse en concentradores de los datos que proporciona una red de sensores y dar acceso a ellos mediante Internet (función de puerta de acces o gateway).

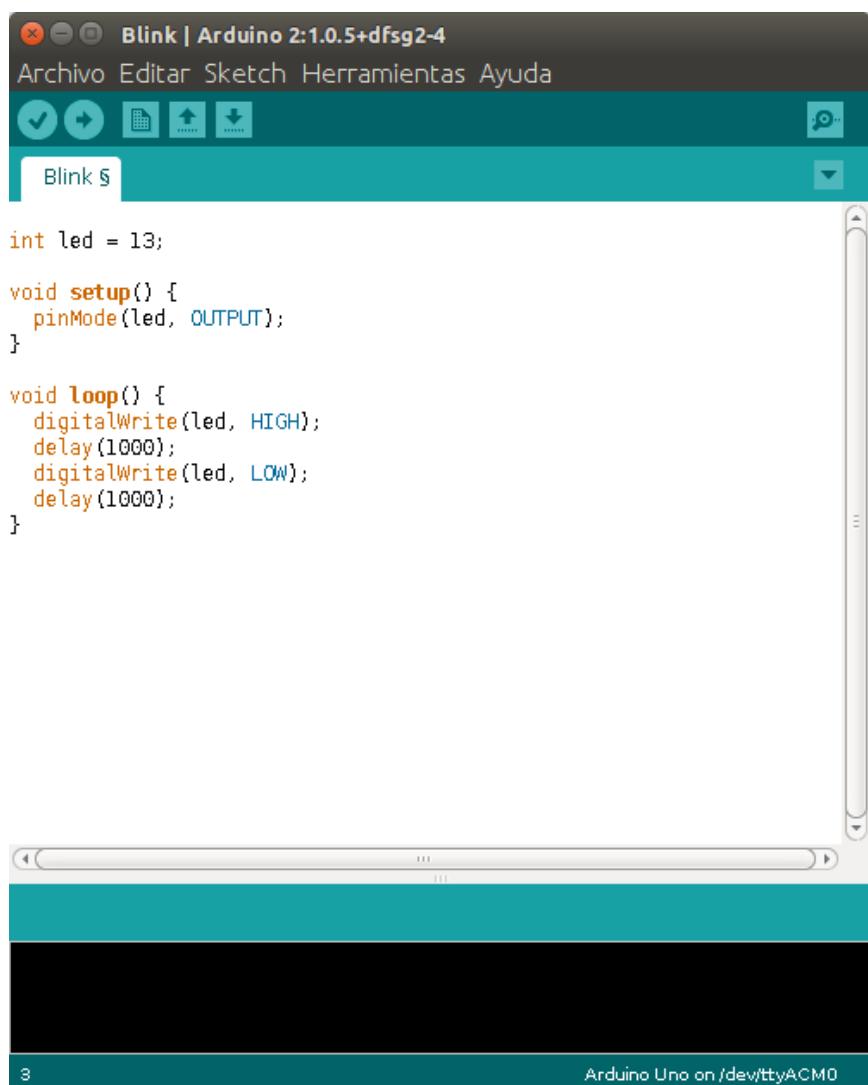


Figura 2.17: Interfaz de programación de Arduino, con un **sketch** básico “Hola Mundo”

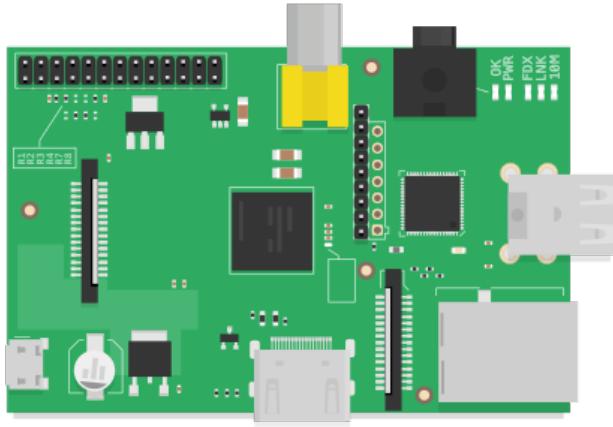


Figura 2.18: Raspberry Pi **Fuente:** [23]

## 2.7. Enfocadores astronómicos: estado del arte

Entendemos estado del arte como aquellos desarrollos de última tecnología realizados sobre un producto, que han sido probados en la industria y han sido acogidos y aceptados por diferentes fabricantes.

Dado el alcance e interés de este proyecto es de obligado cumplimiento el hacer una investigación entre los diferentes fabricantes de las soluciones existentes para el enfoque automático en astronomía. Una búsqueda en Internet nos ofrece algunas alternativas:

- **Orion AccuFocus** [65]: Muy básico: ni control por ordenador, ni pantalla, ni botones de control preciso. Tiene un precio en el mercado de 99\$.
- **FocusMaster** [90], cuenta con las funciones básicas de control, incluido ajuste de velocidad y ajuste de movimiento fino. Su precio oscila en torno a los 200\$.
- **Opetec** es uno de los fabricantes más especializados y dispone de varios modelos, el más económico tiene un precio de 725\$.
- **Robofocus**, una de las soluciones más avanzadas y extendidas, tiene un precio de 495\$.

A continuación detallamos un poco más las soluciones existentes más populares y completas.

### 2.7.1. Opetec

**Opetec** es uno de los fabricantes más importantes en el mundo de los productos electro-ópticos usados en astronomía. Su sede central se localiza en Lowell (Michigan, EEUU).

Entre los productos más económicos para enfoque astronómico contamos con el modelo, **17670 - Original TCF-S** (figura 2.19), cuyas características estrella son la **compensación térmica** y un **backlash pequeño**.

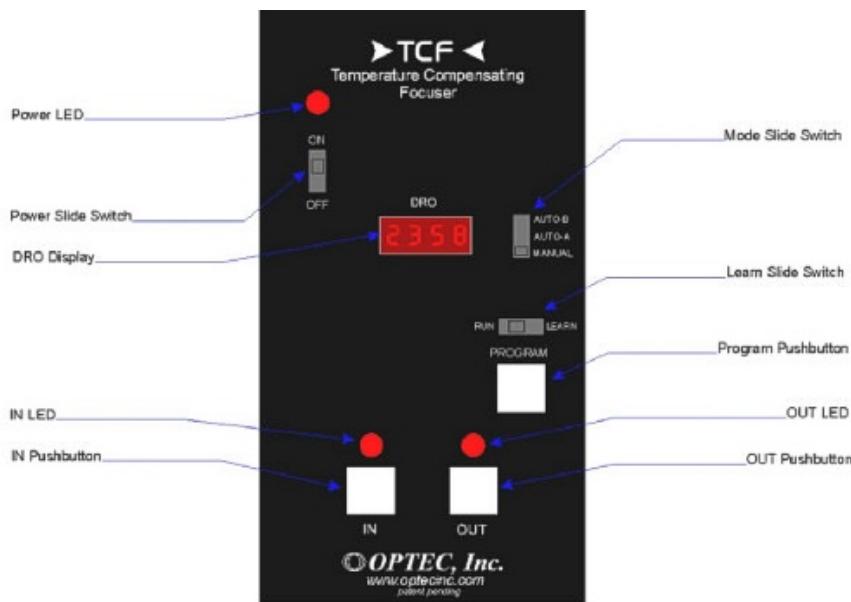


Figura 2.19: Mando de control Opetec Original TCF-S. **Fuente:** [70]

La misma compañía cuenta con un software específico para corregir el foco en tiempo real, **FocusLock Focusing Software** (figura 2.20).

Se trata de un producto de alta calidad, pero su precio es elevado y el software requiere de la plataforma **ASCOM** (10.1) lo que limita su uso.

### 2.7.2. Robofocus

Robofocus (figura 2.21) es uno de los enfocadores astronómicos más populares que existen en el mercado es **Robofocus** [95].

Además una de contar con todas las características anteriores de la solución ofrecida por Opetec, incluida la compensación por cambio de temperatura, Robofocus es compatible con la plataforma INDI [80].

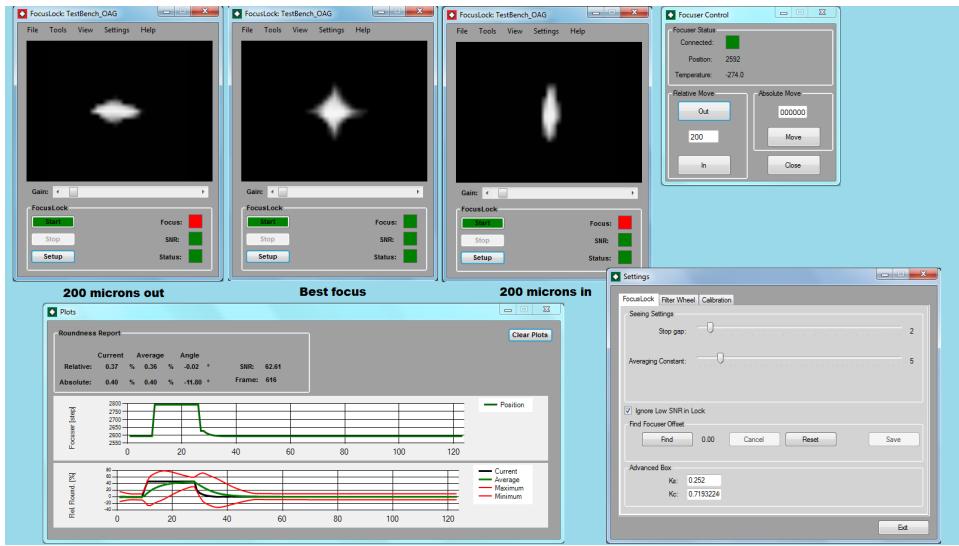


Figura 2.20: Software Opetec Focus Lock en un ciclo de funcionamiento  
**Fuente:** [70]

Investigando el funcionamiento del dispositivo se puede encontrar el protocolo serie (tabla 2.1) que utiliza, que utiliza el siguiente formato:

- **F** : Indica que es un comando del enfocador.
- **X** : Caracter alpha, selecciona el comando.
- **?** : Separador
- **NNNNNN** : Seis caracteres decimales (completando con ceros sí es necesario).
- **Z** : Checksum, suma de verificación del comando.

Tabla 2.1: Descripción de los comandos de Robofocus

Comando	Parámetro	Descripción
FV	-	Consulta versión del firmware.
FG	XXXXXX	Mueve a posición.
FI	XXXXXX	Mueve X pasos hacia el interior.
FO	XXXXXX	Mueve X pasos hacia el exterior.
FS	XXXXXX	Asigna posición actual como posición X.
FL	XXXXXX	Asigna máximo recorrido.
FP	XXXXXX	Activa salidas de corriente electrica adicional, para alimentar dispositivo adicional.
FB	NXXXXXXZ	Cambia la compensación de backlash.
FT	-	Responde con la temperatura.



Figura 2.21: Componentes de un kit Robofocus. **Fuente:** [95]

A la vista de lo presentado podemos concluir que pese a que existen diversas soluciones para control del foco en astronomía, dichas soluciones:

- Son privativas (lo que dificulta su mejora, estudio, modificación o arreglo).
- Solo funcionan bajo la plataforma ASCOM (Windows).
- Son caras (para lo que ofrecen).

Estos motivos son los que llevan a pensar que el desarrollo de una solución completa de enfoque que sea libre es algo interesante y que puede tener una cierta apreciación por los astrónomos (sobretodo los amateur).



# Capítulo 3

## Objetivos

El objetivo global de este proyecto **OBJ-G** es desarrollar un sistema completo y libre de **enfoque** astronómico. Dicho sistema debe permitir un modo de funcionamiento remoto utilizando el protocolo **INDI**.

Este objetivo global se desglosa en los siguientes objetivos principales:

- **OBJ-P1.** Diseño dispositivo hardware, usando plataforma libre **Arduino**, que interactúe directamente con los mandos de enfoque del **telescopio**, mediante un motor y distintos sensores.
- **OBJ-P2.** Implementar un firmware, que permita realizar el control de todos los periféricos y electrónica.
- **OBJ-P3.** Implementar un **Driver INDI** que provea de la funcionalidad básica para controlar todas las características o propiedades del dispositivo enfocador.
- **OBJ-P4.** Diseñar un algoritmo de autofocus, basándonos en una medida del foco y coordinada con las funciones del dispositivo y las imágenes capturadas por una CCD [*opcional, puesto que ya existen soluciones software que lo hacen usando INDI*].
- **OBJ-P5.** Documentar y difundir extensivamente el trabajo para que sea reproducible por todos los interesados.

Dado los objetivos principales del proyecto, podemos hacer un desglose en objetivos menores, separándolos a su vez en objetivos **Obligatorios**, que deben cumplirse para llegar a completar su correspondiente objetivo principal, y objetivos **Secundarios**, objetivos igualmente interesantes para el proyecto, pero que carecen de importancia vital y su incumplimiento no evita que se pueda completar su correspondiente objetivo central.

- **OBJ-P1.O1.** Investigación de los periféricos y sensores existentes para la plataforma Arduino,
- **OBJ-P1.O2.** Realizar una estimación de precios y gastos en componentes, sopesando distintas alternativas.
- **OBJ-P1.O3.** Investigar sobre los distintos métodos de mecanización para los materiales empleados, así como posibles alternativas para crear la PCB y carcasa del dispositivo.
- **OBJ-P1.O4.** Implementar un prototipo completo.
- **OBJ-P1.S1.** Implementación de distintas versiones, completa, solo funcionamiento remoto o solo control manual.
- **OBJ-P2.O1.** Investigar librerías de control de los periféricos, buscando alternativas y realizando pruebas y ejemplos.
- **OBJ-P2.O2.** Implementar un **Firmware**, basándonos en un diseño modular, permitiendo así que futuras ampliaciones sea fáciles.
  - **OBJ-P2.O2.M1** Módulo de control de motores paso a paso.
  - **OBJ-P2.O2.M2** Módulo visualización de datos en pantalla LCD.
  - **OBJ-P2.O2.M3** Módulo control manual.
  - **OBJ-P2.O2.M4** Módulo control remoto y comunicación con host.
  - **OBJ-P2.O2.M4** Módulo de sensores externos.
- **OBJ-P2.O3.** Definir protocolo de comunicación, mensajes y parámetros, entre ordenador y el dispositivo.
- **OBJ-P2.O4.** Realizar las pruebas pertinentes para comprobar el buen comportamiento de la lógica, sobre la electrónica previamente diseñada, así como la integración de los distintos módulos.
- **OBJ-P2.S1.** Implementar protocolo de intercambio de mensajes **Rofocus** (sección 2.7.2).
- **OBJ-P3.O1.** Que cumpla el entandar INDI para enfocadores
- **OBJ-P3.S1.** Que el driver proporcione información sobre los sensores.
- **OBJ-P3.S2.** Realizar script de despliegue automatizado, para instalar y correr el Servidor INDI con su correspondiente driver.
- **OBJ-P4.O1** Implementar un pequeño framework de visualización y procesamiento de imágenes, que permita evaluar y parametrizar de forma simple los algoritmos que aplicamos sobre las imagenes estelares.

- **OBJ-P4.S2** Diseño y implementación de algoritmos de detección de objetos celestes, con propiedades estelares, basándose en la curva de luz característica.
- **OBJ-P4.O3** Realizar cálculo de distintas medidas de enfoque, calculo del FWHM.
- **OBJ-P4.S4** Dada la medida anterior, implementar algoritmos de búsqueda de máximo enfoque, coordinado con el movimiento del enfocador y la adquisición de las imágenes por parte de la CCD.

Además de los objetivos anteriores, se persigue alcanzar los siguientes objetivos secundarios globales:

- **OBJ-P5-O1.** Completar una buena documentación técnica, así como información variada para desarrolladores, a fin de que cualquier interesado sea capaz de reproducir fácilmente y a bajo costo el proyecto.
- **OBJ-P5-O2.** Publicar código fuente, diseños y documentación bajo una licencia libre (GNU3 o similar), que permita que la comunidad de desarrolladores interesados pueda realizar modificaciones y personalizaciones.
- **OBJ-P5-S1.** Difusión en la comunidad astronómica y desarrollo de una web propia del proyecto.

Estos objetivos se han representado gráficamente en la figura 3.1.

Para la realización de los objetivos se pondrán en práctica los conocimientos alcanzados en las siguientes materias impartidas en la Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada:

- **Ingeniería del software** para el análisis y diseño del proyecto, así como modelar el sistema.
- **Programación orientada a objetos** para la estructura y la organización del código **Java**.
- **Programación de sistemas multimedia** para poder implementar las interfaces de usuario en **Java Swing**, así como visualizar y tratar las imágenes.
- **Infraestructuras virtuales** para poder gestionar los sistemas, teniendo habilidad para realizar instalaciones y aprovisionamiento en servidores.
- **Transmisión de datos y redes de computadores** para comprender el funcionamiento de las capas de red y los puertos, base para

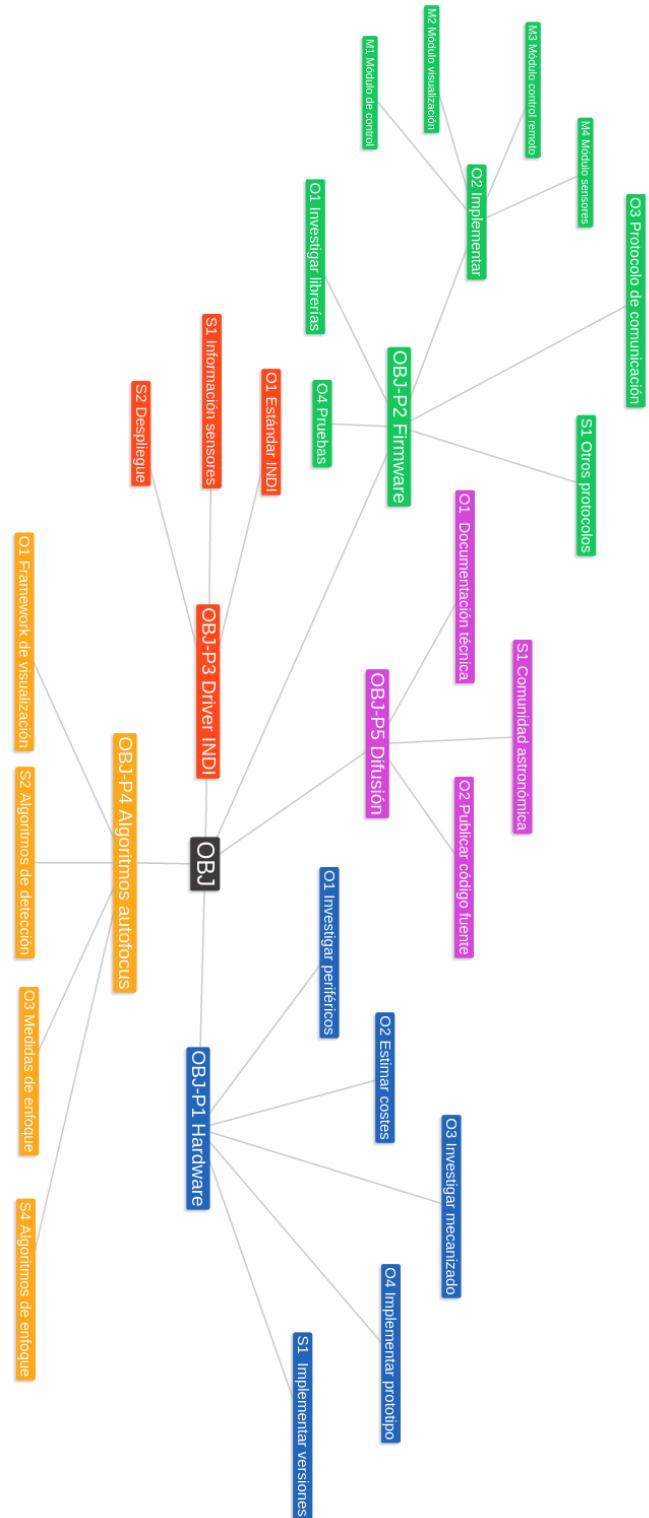


Figura 3.1: Diagrama de los Objetivos del Proyecto

comprender el protocolo **INDI** y configurar correctamente las redes para las pruebas.

- **Algorítmica** para optimizar las rutinas de tratamiento de imágenes.
- **Calculo matemático** para poder implementar las funciones matemáticas necesarias.
- **Estructura de datos** para modelar la representación de las imágenes de una forma eficiente.
- **Tecnologías Emergentes** para conocer las plataformas de hardware libre y domótica.

Por otro lado, han sido necesarios alcanzar conocimientos en otras áreas:

- **Astronomía básica y equipos astronómicos** para entender a los usuarios potenciales y poder acomodar la aplicación a sus necesidades.
- **Electrónica básica y Arduino** para diseñar y construir dispositivo hardware.
- **Raspberry Pi** para montar un servidor permanente de pruebas o acceso público para probar la aplicación.
- **LATEX** para la realización del presente documento y la ampliación de conocimientos para futuros textos científicos.
- **Git** para la gestión de versiones y la publicación de código abierto que permita a otros desarrolladores participar.



## Capítulo 4

# Metodología de Trabajo, Planificación y Costes

En este capítulo se desglosa información sobre la metodología de trabajo desarrollada, la planificación inicialmente propuesta (e información adicional sobre su cumplimiento) así como una estimación de costes de desarrollo del proyecto.

### 4.1. Metodología de Trabajo

Podemos diferenciar grandes bloques o subsistemas (de acuerdo con los objetivos principales marcados en el capítulo 3 que dada su diferente naturaleza debemos analizar, diseñar, implementar y testear por separado, para posteriormente dedicarnos a la integración de unos módulos con otros.

Es importante mencionar que dado la novedad del proyecto, se necesita un trabajo extra de investigación, tanto a nivel conceptual, de dominio y técnico. Dado que no se ha tratado con un diseño inicial estanco y cerrado, se ha dejado abierto en muchos aspectos, y a medida que hanido surgiendo nuevas ideas, se han descubierto tecnologías o el cliente ha propuesto un nuevo requisito, no se ha tenido limitación en tratar de incorporarlo, siempre con el objetivo de optimizar la solución mediante refinamiento.

La metodología seguida en el proyecto podemos enmarcarla como **Iterativa basada en prototipos** (figura 4.1): para cada módulo se crean versiones cada vez más óptimas y aproximadas a la solución definitiva.

Esta decisión se ha tomado en base a las siguientes características de dicha tipo de metodología, muy beneficiosas en nuestro proyecto:

- Rápida retroalimentación por parte del cliente y validación de los requisitos.
- Optimización de la solución según se tiene un conocimiento más fuerte del dominio y la tecnología manejada.
- Los prototipos son **rápidos** y **económicos** de construir.
- Los prototipos garantizan que el producto se adapta a las necesidades del cliente, dado que **el cliente es participe del desarrollo mediante revisiones continuas**.
- Los prototipos constituyen un **medio de comunicación mejor** que los modelos formales, ya sean diagramas o listados de requisitos.
- Permiten **refinar la estimación de tiempos de desarrollo**, observando de forma más realista las **dificultades** que se van dando.

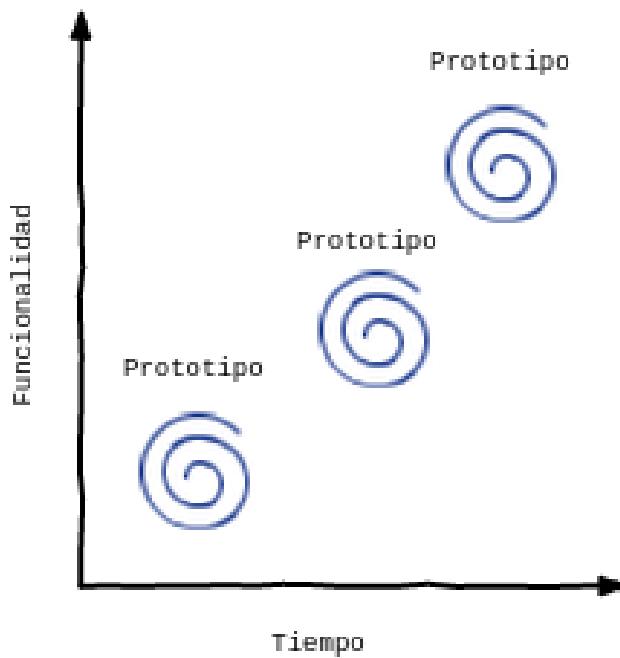


Figura 4.1: Iterativa basada en prototipos

Podemos observar en la figura 4.2 como cada módulo cicla por sus correspondientes fases, **planteamiento, análisis y diseño, implementación, pruebas**.

Tras las pruebas en **feedback** se recopila información de retroalimentación, que puede proceder de errores detectados en pruebas o de informes por parte del cliente, manifestando la falta o mal entendimiento de algunos requisitos.

Cuando se encuentran deficiencias, debemos valorar si son exclusivas e internas al propio módulo o por el contrario también incumben al módulo anterior. En caso de estar relacionado con el módulo anterior debemos realizar una iteración sobre tal módulo.

## 4.2. Fases

Dado que el proyecto cuenta con módulos (uno por cada uno de los objetivos primarios comentados en el capítulo 3) que pueden considerarse subsistemas de un sistema de magnitud mayor, podemos a su vez dividir el desarrollo de cada uno de los módulos en distintas fases. Ello nos ayuda a evaluar las tareas y hacer estimaciones con alcances más reducidos. Adicionalmente debemos contemplar un coste extra de integración de los módulos. En la figura 4.3 se muestra una distribución ideal del tiempo y solapamiento en la consecución de las distintas fases de cada uno de los módulos.

### ■ Módulo Hardware

- **Fase 0:** Reunión con cliente y planteamiento del problema.
- **Fase 1:** Investigar tecnologías implicadas.
- **Fase 2:** Análisis y diseño.
- **Fase 3:** Implementación.
- **Fase 4:** Pruebas.

### ■ Módulo Firmware

- **Fase 0:** Reunión con cliente y planteamiento del problema.
- **Fase 1:** Investigar librerías y lenguaje programación.
- **Fase 2:** Análisis y diseño.
- **Fase 3:** Implementación.
- **Fase 4:** Pruebas.

### ■ Módulo Driver INDI

- **Fase 0:** Reunión con cliente y planteamiento del problema.
- **Fase 1:** Familiarizarme con arquitectura INDI.
- **Fase 2:** Análisis y diseño.
- **Fase 3:** Implementación.
- **Fase 4:** Pruebas.

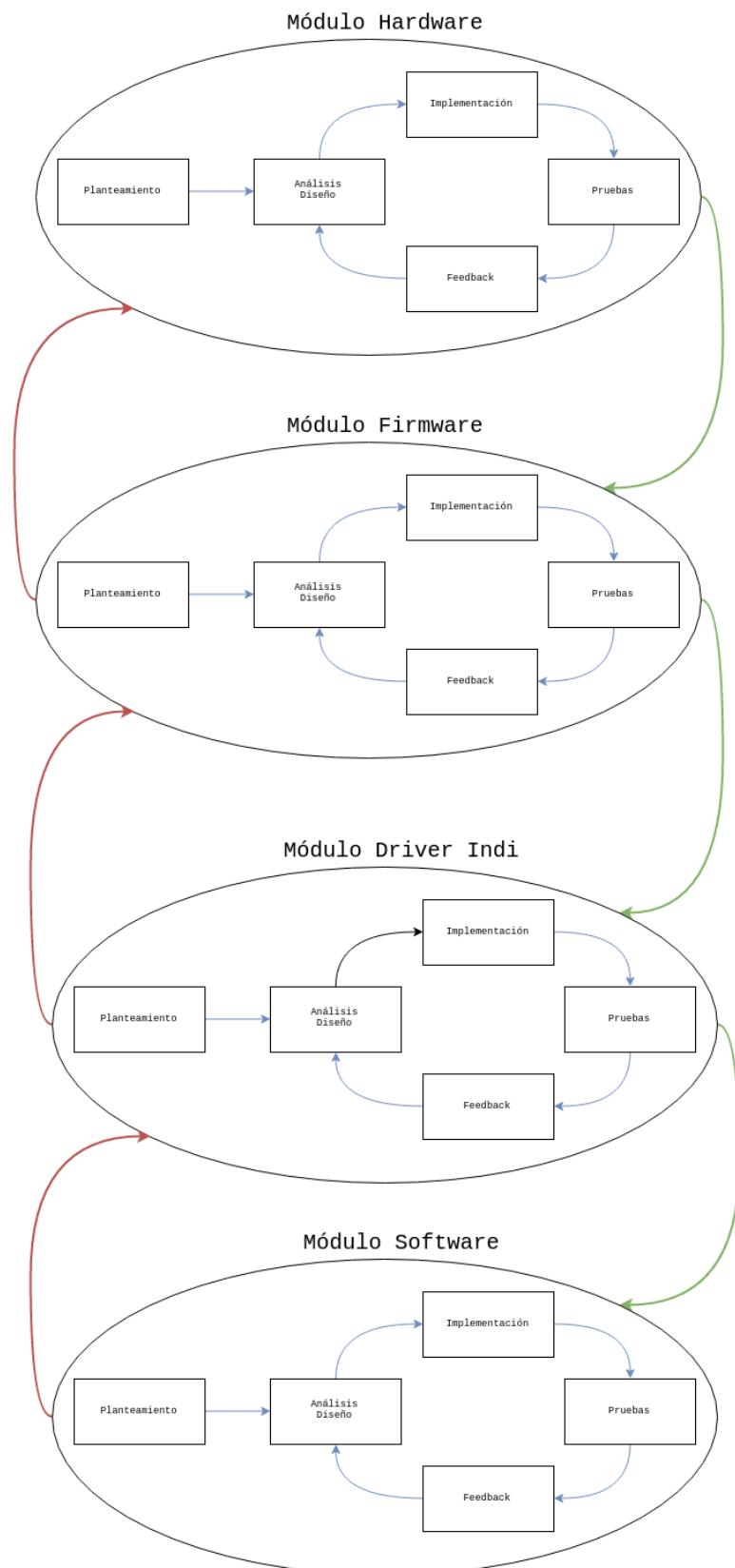


Figura 4.2: Diagrama iteraciones basados en iteraciones basadas en prototipos modulares

### ■ Módulo Software

- **Fase 0:** Reunión con cliente y planteamiento del problema.
- **Fase 1:** Investigar entorno de procesamiento de imágenes.
- **Fase 2:** Análisis y diseño.
- **Fase 3:** Implementación.
- **Fase 4:** Pruebas.

### ■ Integración

#### ■ Pruebas de integración

#### ■ Documentación

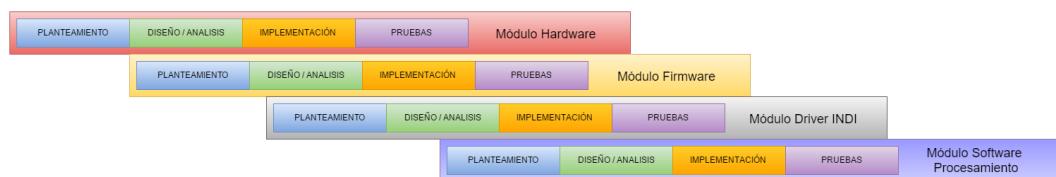


Figura 4.3: Evolución desarrollo de los módulos

## 4.3. Temporización

Por limitaciones dadas por el horario laboral y trabajos del resto de asignaturas del grado, se hace una estimación del tiempo que se puede dedicar al proyecto: **2 horas los 5 días a la semana**. Se cuenta con fines de semana para suplir posibles imprevistos que surjan durante la semana.

### 4.3.1. Estimación de tiempos

Es necesario hacer una estimación de tiempos para cada tarea a desarrollar a fin de organizar correctamente el trabajo realizar. En principio se estima que se necesitarán unas 38 semanas de tiempo, totalizando **380 horas** de trabajo. Dichas horas se distribuyen teóricamente tal y como se muestra en el diagrama de **Gantt** situado en el apéndice apéndice B (figura B.1).

Dada la estimación anterior se prevé que la fecha de entrega será para la convocatoria de **Diciembre del 2015**.

#### 4.3.2. Tiempos invertidos

Dada la complejidad del proyecto, así como la incorporación de diversas mejoras no previstas al inicio del mismo, así como una subestimación del esfuerzo necesario para implementar ciertas partes del proyecto, finalmente hemos observado que hemos consumido **58 semanas efectivas**, trabajando de media durante 2 horas los 5 días de la semana. Por tanto, el tiempo empleado total ha sido de 580 horas. Como resultado de este incremento en las horas de trabajo, finalmente la fecha de entrega real es **Septiembre del 2016**.

Podemos ver un diagrama de Gantt del desarrollo (real) en el apéndice B (figura B.2).

Tal y como se puede observar el tiempo real esta muy por encima del tiempo estimado, exactamente pasamos de 38 a 58 semanas (380 a 580 horas), con un **desfase de 200 horas**. Analizando las causas de tal desfase, podemos encontrar:

- **Varias iteraciones sobre cada módulo**, en un principio se pensó que con un primer prototipo se conseguirían alcanzar todos los objetivos. En la práctica esto no ha sido posible y para llegar a resolver de forma satisfactoria todos los objetivos propuestos se han tenido que realizar varios prototipos y refinamientos de cada uno de ellos.
- **Requerir dos años académicos**, en el primero de ellos tras realizar un estudio del dominio, la tecnología y una planificación inicial, por diferentes motivos personales se abandonó el desarrollo del proyecto tal y como se puede ver en el diagrama de Gantt (figura B.2) en el periodo Abril-Noviembre de 2016. Al reanudar el proyecto en Noviembre se necesita volver a invertir tiempo en planificación y estudio del dominio.
- **Tiempo de experimentación**, al ser un proyecto en el marco académico y orientado también a mi formación, he dedicado tiempo extra en realizar experimentos, probar diferentes alternativas, explorar Internet para recabar toda la información posible relacionada.

Creo que el desarrollo de este proyecto es un buen ejemplo que muestra la importancia de realizar una estimación de tiempos realista, ya que en un TFG las consecuencias de un retraso no son excesivamente graves, pero en un proyecto empresarial con presupuestos fijos y *deadlines*, un retraso de esta magnitud puede tener consecuencias serias. Por ello el caso dado es una excelente lección para mi futuro en el mundo laboral.

#### 4.4. Recursos humanos

A nivel de recursos humanos he sido el único integrante del equipo técnico, a excepción de tareas que requerían manejo de maquinaria avanzada, como cortadora láser, CNC o torno en el cual se ha contado con la ayuda de especialistas.

El cliente también ha sido participante en todo momento del desarrollo del proyecto, validando los requisitos del sistema, los prototipos y participando de forma activa en las pruebas.

Su cooperación y experiencia en el campo de la astronomía ha hecho que el desarrollo del proyecto sea mucho más fluido y el resultado de mayor calidad.

#### 4.5. Presupuesto

Para el desarrollo del presente proyecto se tendrán en cuenta los siguientes costes (resumidos en la tabla 4.1):

- **Costes de componentes y hardware necesario:** se hace un desglose pormenorizado en el capítulo 5. Total 62.5 €. Debemos sumarle unos 50 € adicionales en costes de componentes estropeados, pruebas de componentes que finalmente no se han usado o material fungible.
- **Costes por trabajo del equipo humano:** inicialmente se presupuestaron 380 horas de trabajo que equivalen a 9.5 semanas de trabajo (40 horas semanales). Suponiendo un sueldo mensual de 2000 €, a 4.5 semanas por mes, tenemos un coste asociado al trabajo del equipo humano de 4222 €.
- **Costes asociados a licencias necesarias para publicar o desarrollar el software:** dado que todo el trabajo se ha realizado con programas con licencias libres y gratuitas (estilo GNU3 [41] y similares), no tenemos que invertir dinero en este aspecto: 0 €.
- **Máquina de desarrollo:** portátil valorado en 400 € con sistema operativo Ubuntu 16.04.
- **Servidor INDI:** instalado en una **Raspberry Pi, B+** que tiene un coste asociado de 35 €. Utiliza el sistema operativo **Raspbian** [77] que es libre.
- **Herramientas de desarrollo:** Entorno de desarrollo **Netbeans** [66], gratuito, editor de texto **SublimeText** [88] versión gratuita, **IDE Arduino** [6] gratuito: 0 €.

- **Clientes INDI:** para interactuar con el servidor INDI y sus periféricos, **Kstars** [53] y desde Android **Observatorio Remoto** [68], ambas herramientas son gratuitas: 0 €.
- **Dominio y hosting de Internet para difundir el proyecto:** 12 € (hosting cedido por el cliente).
- **Costes de impresión:** impresión de pósters para comunicación en congresos y reuniones de astronomía: 20 €.

Elemento	Coste (euros)
Componentes y hardware	62.5 €
Componentes de pruebas y fungibles	50 €
Costes equipo humano	4222 €
Licencias	0 €
Máquina de desarrollo	400 €
Herramientas de desarrollo	0 €
Clientes INDI	0 €
Dominio y hosting	12 €
Impresión	20 €
<b>Total</b>	<b>4766.5 €</b>

Tabla 4.1: Costes y gastos requeridos en el desarrollo del proyecto: se comprueba que el mayor coste se encuentra en los gastos derivados del equipo humano.

# Capítulo 5

## Módulo hardware

En esta parte nos centraremos en profundizar en las parte más tangible del proyecto, introduciendo la electrónica y mecánica del mismo.

Es importante destacar que gracias a las diferentes plataformas de hardware libre he podido reducir en gran parte el alcance del proyecto, pudiéndome centrar en implementar las funcionalidades y no tanto en montar todo el marco de trabajo necesario para llegar a ello.

Tenemos presentes las herramientas de última generación, que están teniendo un gran auge y aceptación en la comunidad DIY, como lo son las **impresoras en 3D** o las **cortadoras láser**, que agilizan la mecanización y fabricación de las partes mecánicas.

### 5.1. Análisis

Después de las reuniones con el cliente y el estudio del dominio puedo llegar a extraer los requisitos que debe satisfacer el dispositivo para el objetivo propuesto. Dichos requisitos se desglosan a continuación.

#### 5.1.1. Requisitos funcionales

- **RF-1.:** Controlar motor paso a paso:
  - **RF-1.1.:** Controlar la dirección.
  - **RF-1.2.:** Controlar velocidad.
  - **RF-1.3.:** Controlar microstepping, para conseguir la máxima resolución en los pasos. [62]

- **RF-2.**: Mostrar información del sistema mediante una pantalla LCD.
- **RF-3.**: Modificar parámetros de forma manual.
  - **RF-3.1.**: Modificar velocidad.
  - **RF-3.2.**: Modificar mover  $n$  pasos por pulsación.
- **RF-4.**: Ejecutar funciones de movimiento del motor desde una botonera o control manual.
  - **RF-4.1.**: Ejecutar función de movimiento continuo en la dirección deseada.
  - **RF-4.2.**: Ejecutar función de movimiento ( $n$  pasos) en la dirección deseada.
- **RF-5.**: Monitorizar temperatura y mostrar aviso cuando se detecte un cambio.
- **RF-6.**: Controlar límites físicos en el desplazamiento de las partes mecánicas del enfocador.

### **5.1.2. Requisitos no funcionales**

- **RNF1.**: El movimiento del motor debe tener una precisión alta y una suavidad adecuada.
- **RNF2.**: Los controles deben ser intuitivos y responder de forma interactiva.
- **RNF3.**: La pantalla LCD no debe tener excesiva iluminación para conservar la oscuridad.
- **RNF4.**: Fácil instalación y transporte.
- **RNF5.**: Dispositivo robusto y tolerante a condiciones adversas, sesiones de trabajo prolongadas, así como posibles golpes y movimientos.
- **RNF6.**: Bajo consumo, dado que puede ser alimentado por baterías.

## **5.2. Planificación**

- **Fase 0: Planteamiento del problema.**
  - **Reunión inicial con el cliente:** Donde se propone la idea y los puntos clave del proyecto.

- **Investigar la temática:** Buscar enfocadores comerciales, ver su funcionalidad y posibles nuevas funcionalidades.
- **Segunda reunión con cliente:** Se proponen las funcionalidades más interesantes y se compara con los dispositivos comerciales estudiados, el cliente corrobora que las funciones resultan más o menos interesantes desde su punto de vista como astrónomo amateur.
- **Fase 1: Investigar tecnologías implicadas.**
  - **Estudiar tecnologías que se emplearán,** para ello se estudian las diferentes plataformas de hardware libre, comparando a grandes rasgos sus puntos fuertes, limitaciones, así como otros aspectos como precio, soporte por la comunidad, componentes y repuestos.
  - **Buscar componentes** a utilizar y comparar precios con los diferentes proveedores.
- **Fase 2: Análisis y diseño.**
  - Redactar formalmente los **requisitos** del sistema.
  - Realizar **esquemáticos** de la electrónica.
  - Realizar **diseños** para el corte de la caja exterior.
- **Fase 3: Implementación.**
  - **Montaje versión 0** del dispositivo usando placas de prototipado y cables.
  - **Corte de piezas** para construir la carcasa del dispositivo.
  - **Soldadura de la PCB**, con todos los componentes,
  - **Montaje prototipo 1**, versión 1, haciendo uso de la caja y la placa PCB.
- **Fase 4: Pruebas.**
  - **Pruebas de los componentes**, asegurando que no tienen defectos de fábrica.
  - **Pruebas prototipo versión 0.**
  - **Pruebas prototipo versión 1.**

### 5.2.1. Mockup

Un primer boceto que se realiza del sistema con los requisitos anteriores es el mostrado en la figura 5.1.

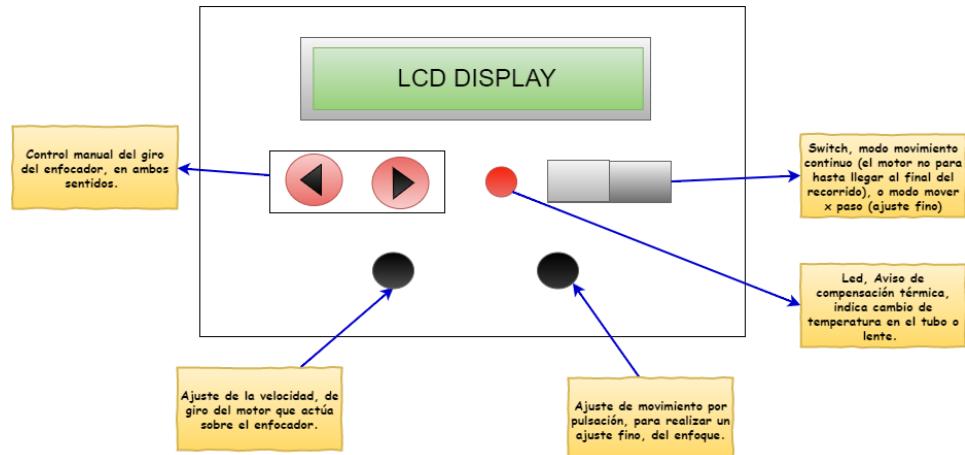


Figura 5.1: Diseño parte externa del dispositivo, donde podemos ver los diferentes controles.

## 5.3. Diseño sistema hardware y electrónica: arquitectura del sistema

Para la elaboración y diseño del sistema hardware hemos seleccionado distintos componentes que hemos agrupado en los siguientes módulos:

- Núcleo: Arduino
- Módulo de motores
- Módulo de pantalla
- Módulo de sensores
- Módulo de control manual

Estos módulos se describen en las siguientes secciones.

En la figura 5.2 mostramos un esquemático en el que se integran los distintos componentes hardware.

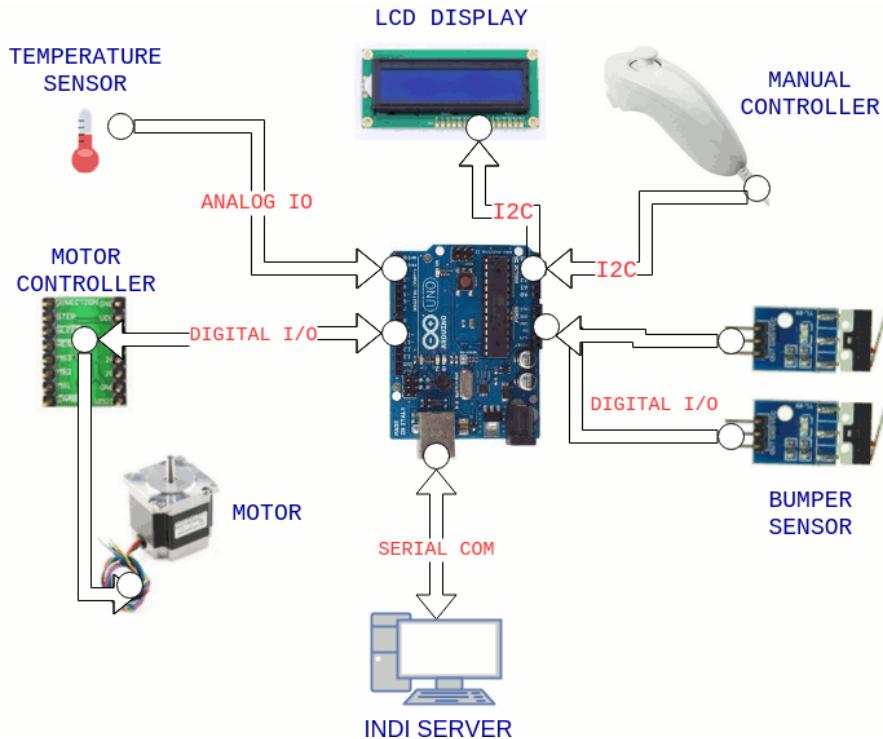


Figura 5.2: **Diagrama general, sistema hardware.** En el diagrama podemos ver los elementos que intervienen y de forma simplificada la relación que existe entre ellos.

### 5.3.1. Núcleo: Arduino

Para el diseño de la electrónica, partimos como base de una placa de prototipo **Arduino** [9].

El hardware consiste en una placa de circuito impreso con un microcontrolador, usualmente **Atmel AVR** [20], junto con puertos digitales y analógicos para entrada/salida.

Las características principales de las placas Arduino, que pueden servirnos para decidirnos entre un tipo de placa u otro son las siguientes:

- **Microcontrolador**, por su arquitectura y su frecuencia de reloj.
- **Tamaño de las memorias**,
  - **Flash** (espacio del programa) es donde Arduino almacena el **sketch**. Esta memoria es no volátil y su tamaño oscila entre los 16 kilobytes.
  - **SRAM** Memoria estática de acceso aleatorio, de tipo volátil. Es

el espacio donde los sketches (programas) almacenan y manipulan variables al ejecutarse. La información guardada en esta memoria será eliminada cuando Arduino pierda la alimentación. Esta memoria es de uso exclusivo para el programa en ejecución. Su tamaño oscila en torno a los 1024 bytes.

- **EEPROM** es un espacio de memoria que puede ser utilizado por los programadores para almacenar información a largo plazo. Este tipo de memoria es no volátil, por lo que los datos guardados en ella permanecerán aunque Arduino pierda la alimentación. Permite almacenar configuraciones de la sesión. Es importante resaltar que tiene un número de escrituras muy limitado por lo cual no se puede usar de forma extensiva. Su tamaño oscila en torno a los 512 bytes.
- **Puertos de entrada y salida I/O.** Son los puertos que disponemos para comunicarnos con los periféricos, ya sean sensores o actuadores.
  - **Digitales:** Estos **pines**, tal como podemos intuir trabajan con señales digitales de 5V, (estados LOW y HIGH), existiendo un tipo de especial de ellos denominados **PWM**, que permiten “emular” señales analógicas. Otro punto a tener en cuenta es el número de pines que permiten ejecutar **interrupciones hardware**.
  - **Analógicos:** Permiten lectura de valores analógicos que se encuentren en la escala de 0V a 5V. Las entradas analógicas disponen de 10 bits de resolución, lo que proporciona 1024 niveles digitales, lo que a 5V supone una precisión de la medición de  $\pm 2,44\text{mV}$ .
- **Otros factores**
  - **Factores de forma:** Características como las dimensiones, el peso, la forma.
  - **Conectores auxiliares:** Tamaño y forma de los conectores USB, ya sea de Tipo A, Tipo B o mini-USB.
  - **Consumo:** Muy importante si alimentamos la placa con baterías.

Para este proyecto hemos seleccionado la placa **Arduino UNO** (figura 5.3) en base a los siguientes criterios:

- **Precio reducido**, que oscila en torno a los 20 €.
- **Potencia del procesador y memoria**, suficiente para soportar el firmware a implementar.
- **Conektor USB tipo B**, al ser de buen tamaño, hace la conexión robusta (importante en telescopios donde existe movimiento o posibles

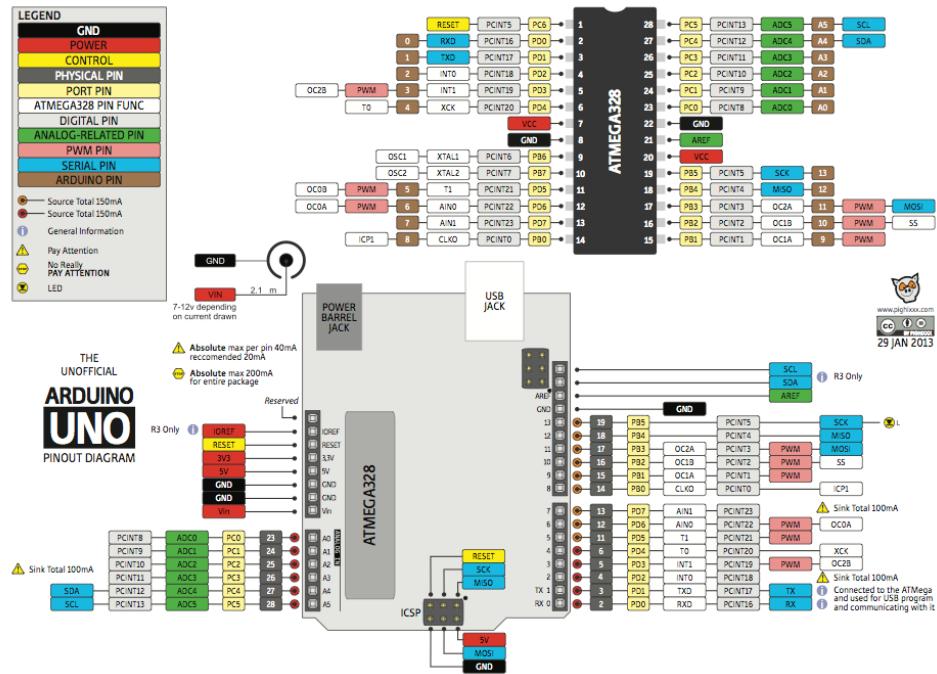


Figura 5.3: **Diagrama de pines de Arduino**, muestra los diferentes pines de entrada/salida de la placa Arduino UNO, así como sus conexiones con el microcontrolador. También podemos ver los pines de alimentación a 5V y 3V3. **Fuente:** [9]

tensiones en los cables).

- **Número de pines**, suficiente para conectar todos los periféricos previstos.
- **Características especiales**, (PWM y interrupciones hardware).
- **Comunicación I2C**.

### 5.3.2. Módulo de motores

Para permitir el control del motor de un paso a paso, usaremos un Pololu A4988 [73] (figura 5.4). Las características más reseñables de tal controlador y que los hacen destacar frente a sus competidores son las siguientes:

- Interfaz simple para controlar pasos y dirección.
- Cinco resoluciones de paso diferentes, paso completo, 1/2 paso, 1/4 de paso, 1/8 de paso y 1/16 de paso.

- Control de corriente de paso ajustable.
- Protecciones contra sobretensiones y sobrecalentamientos.

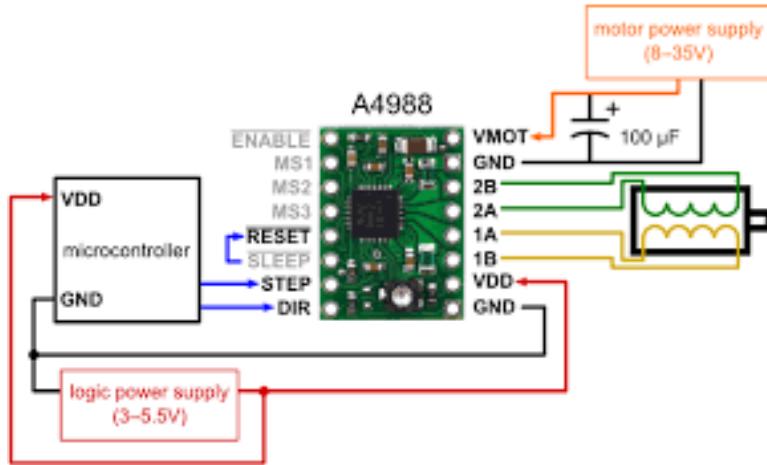


Figura 5.4: **Pololu A4988** - Vista del controlador de motor paso a paso utilizado. Podemos observar las diferentes conexiones, puertos de control (STEP y DIR), alimentación del controlador (VDD), alimentación del motor (VMOT), salidas para el motor (1A, 1B, 2A, 2B) y pines para ajuste de la resolución de pasos (MS1, MS2, MS3) **Fuente:** [73]

El motor empleado para nuestro sistema es un motor paso a paso **NEMA 17** / 3.2Kg/cm, de uso muy habitual, en el montaje de impresoras 3D. Sus características principales son las siguientes:

- **Tamaño:** 42.3x48mm (sin incluir el eje) (NEMA 17)
- **Peso:** 350 gramos (13 oz)
- **Diámetro del eje:** 5 mm "D"
- **Longitud del eje:** 25 mm
- **Pasos por vuelta:** 200 ( $1.8^\circ$ /paso)
- **Corriente:** 1.2 Amperios por bobinado
- **Tensión:** 12 V
- **Resistencia:** 3.3 Ohm por bobina
- **Torque:** 3.2 kg/cm (44 oz-in)
- **Inductancia:** 2.8 mH por bobina

### 5.3.3. Módulo de pantalla

Para visualizar los estados del sistema fácilmente, se incorporará una pequeña pantalla LCD. Uno de los displays LCD más corrientes son los alfanuméricicos de 16x2 o 16x4 caracteres.

Para su utilización en un principio se estima controlarla mediante los pines digitales usando 4 señales digitales D7, D6, D5, y D4, junto con dos pines adicionales de control tal y como se puede ver en las figuras A.1 y A.2 del apéndice A.

En una fase de rediseño optimizamos el circuito incorporando un **interfaz I2C**, reduciendo las señales a solo dos (figura 5.5).

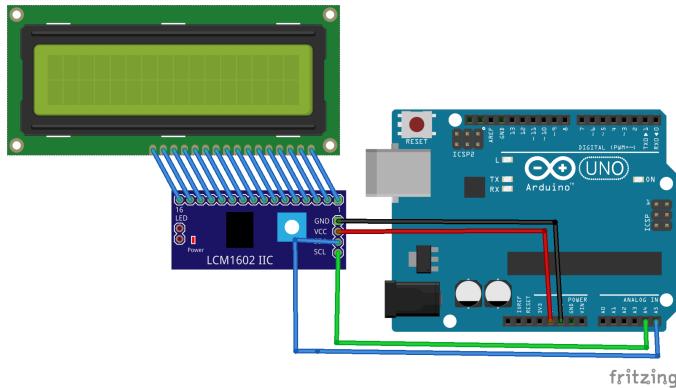


Figura 5.5: Diagrama conexión pantalla LCD con interfaz I2C

### 5.3.4. Módulo de sensores

El dispositivo incorpora dos tipos de sensores:

- **Sensor térmico:** Hacemos uso de un sensor LM35 [98]. Es un sensor de temperatura con una precisión calibrada de 1 °C. Su rango de medición abarca desde -55°C hasta 150 °C. La salida es lineal y cada grado Celsius equivale a 10 mV.

En una fase de rediseño se reemplaza el sensor por uno del tipo DHT11 [84], que mide temperatura en el rango -10 °C y 50°C. Adicionalmente tambien mide humedad, en un rango del 20 % al 95 % (figura 5.6). Usa un protocolo de un solo hilo digital, (1-wire).

- **Endstop sensor**, o contactos fines de carrera. Son interruptores del tipo todo o nada, que se activan cuando se llega a un límite físico.

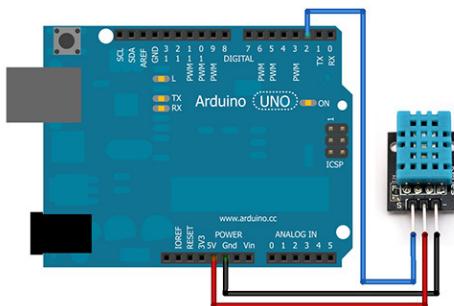


Figura 5.6: Diagrama de conexión del sensor de temperatura y humedad DHT11

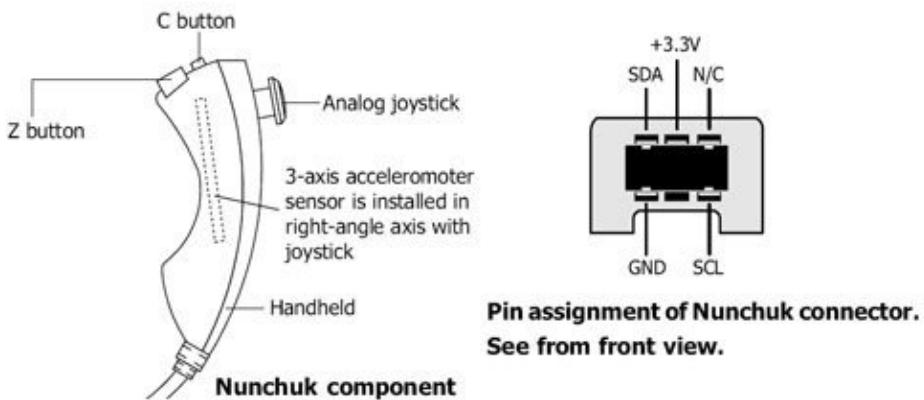


Figura 5.7: Diagrama de conexiones Wii Nunchuck: observamos que contamos con las señales (SDA y SCL) y se alimenta a 3.3V **Fuente:** [27]

### 5.3.5. Módulo de control manual

Para permitir el control manual del dispositivo usamos los siguientes componentes:

- **Potenciómetros:** para regular algunos valores, en concreto usamos 10 kOhms.
- **Botones:** en los primeros prototipos una botonera para activar las distintas funciones.
- **Wii Nunchuck:** se utiliza un mando de la consola Wii, para de una forma ergonómica (y barata) activar los comandos, mediante el botón zeta y algunos pulsadores (figura 5.7). Este mando totalmente compatible con Arduino dado que funciona bajo el protocolo I2C.

## 5.4. Presupuesto

En todo momento se ha tratado de optimizar el precio del desarrollo buscando componentes de reducido precio comparando entre los diferentes proveedores. Sin embargo es importante destacar que en componentes críticos (núcleo, controladora de motor) no es adecuado arriesgarse a introducir elementos de proveedores dudosos. Es por ello que se usa placa de Arduino UNO Oficial así como una controladora de motor Pololu. El desglose costes para cada componente hardware se muestra en la tabla 5.1 (algunos precios pueden variar ligeramente).

Componente	Precio (€)
Arduino UNO	22 €
Motor PaP Nema 17	10 €
Controladora Motor (Pololu o similar)	7 €
Pantalla LCD + controladora I2C	2.5 €
Nunchaku Wii (compatible)	4 €
Sensor Temperatura, resistencias, condensador, led	1 €
Cables, soldadura, pines, placas, etc.	4 €
Potenciómetros	2 €
Caja	5 €
Clavijas aviador	3 €
Fines de carrera	2 €
<b>Total</b>	<b>62.5 €</b>

Tabla 5.1: Lista de componentes y desglose de precios

## 5.5. Implementación hardware

La implementación del dispositivo hardware se puede definir como un proceso totalmente iterativo, en el marco del prototipado. Comenaremos siempre con pruebas de cada uno de los módulos por separado y trabajando con placas que permiten quitar y cambiar la distribución de los componentes.

Para realizar el diseño del circuito se utiliza la herramienta **Frizing** [36] (figura 5.8), de la que llama la atención su fácil uso dado que esta orientada al prototipado. Cuenta un repertorio de componentes genéricos de Arduino.

Una vez definido el esquemático pasamos a la fase de soldadura. Usaremos la tecnología de agujeros pasantes o **THT**, donde el circuito se monta aprovechando los orificios que trae una placa, y mediante pequeños puentes se van

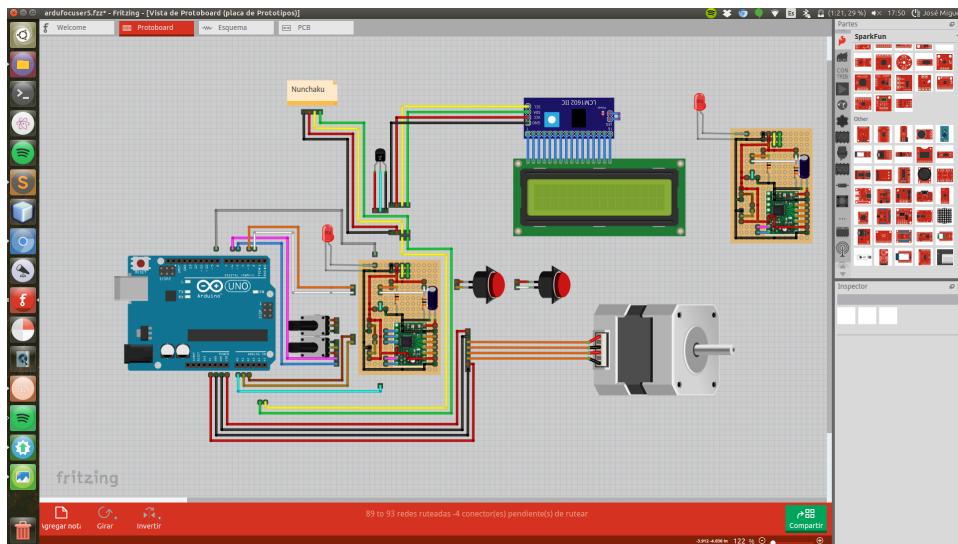


Figura 5.8: Diseño de la circuitería haciendo uso de la herramienta **Fritzing**

uniendo los agujeros creando pistas. En el diseño del circuito nos tenemos que asegurar que las pistas no se crucen (figura 5.9).

### 5.5.1. Prototipos

Durante el proceso iterativo se han implementado cuatro versiones progresivamente más avanzadas. Detallo las características más importantes de cada uno de ellos.

- **Versión 0:** Usando una placa de prototipado que permite poner y quitar los componentes sin realizar soldaduras. Se usa **LCD Keypad Shield** [57] (figura 5.10) para reducir la dificultad inicial. LCD Keypad Shield permite utilizar una pantalla LCD y una botonera acoplándolo dicho módulo sobre Arduino MEGA. Podemos ver el diagrama seguido en la figura A.1 del apéndice A.
- **Versión 1:** Se realiza una completa reestructuración de los componentes, se modifica **LCD Keypad Shield** por una pantalla LCD (interfaz de conexión clásico), una **botonera analógica** (figura 5.11) y conectores RS232 [81]. Se puede ver el diagrama correspondiente en la figura A.2 del apéndice A. En la figura 5.12 se muestra la primera carcasa de aluminio realizada para el prototipo (mecanizada manualmente). Pese al buen acabado de la misma, hacen falta muchas herramientas y tiempo para construir dicha carcasa.
- **Versión 2:** Se rediseña modificando interfaz de la pantalla a I2C.



Figura 5.9: **Fase de soldadura de la placa.** Haciendo uso de un soldador de estaño y un polímetro para comprobar continuidad, procedo a soldar un prototipo de PCB en una placa de baquelita perforada

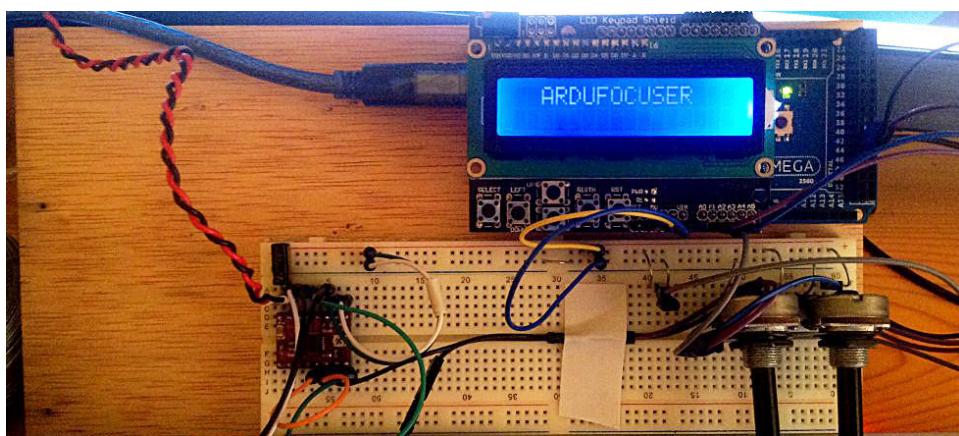


Figura 5.10: Foto del primer prototipo, montada en placa de prototipado. Carece de carcasa y todas las conexiones se hacen usando cables.



Figura 5.11: Foto de la botonera manual analógica

Se intercambia la botonera manual por un mando **Wii Nunchuck** reciclado (tambien por I2C). Esto permite que se reduzca la cantidad de cables usados. Podemos ver el circuito completo en en la figura A.3 del apéndice A.

- **Versión 3:** Modificamos sensor de temperatura analógico LM35 por DHT11 (digital) (figura A.3 del apéndice A).

## 5.6. Diseño de la carcasa y piezas mecánicas

Para dar soporte a toda la electrónica, así como hacer el dispositivo robusto y fácilmente transportable, se decide diseñar una carcasa que permita cubrir toda la electrónica y concentrar todos los conectores y botones.

Para ello debemos hacer una buena estimación del espacio y la distribución de los elementos. Otro punto a tener en cuenta es que debemos permitir la fácil apertura y manipulación del los elementos internos, con el fin de poder realizar mejoras o simplemente permitir a los interesados abrir el dispositivo y conocer sus interioridades.

En un primer momento se optó por adquirir cajas prefabricadas a un proveedor externo y se estudiaron diferentes diseños.

El acabado era excelente, pero se encontraron varios inconvenientes:

- **Dependencia de un proveedor**, que facilite siempre el mismo modelo de caja.



Figura 5.12: **Prototipo en caja de aluminio**, con un acabado muy sólido.

- **Diseño poco personalizable**, debemos asegurarnos de que el dispositivo se adapte perfectamente a las dimensiones.
- **Difícil mecanización**, dado el material metálico, realizar agujeros pasantes, soldaduras, grabados, roscas etc, se convierte en tareas tediosas.

Por estos motivos decidimos plantear la alternativa de fabricar una carcasa propia. Dado que contamos con una **cortadora láser** controlada numéricamente decidimos crear una carcasa totalmente propia a partir de láminas de **metacrilato** de 3mm de espesor (figura 5.13).

Ventajas de esta opción:

- Bajo coste de los materiales.
- Totalmente personalizada y a medida.
- Replicable al 100 %.

En el apéndice C se muestran algunas imágenes del diseño de la caja con un programa de modelado 3D así como los el fichero SVG con los planos de las piezas utilizado para el corte de la misma.

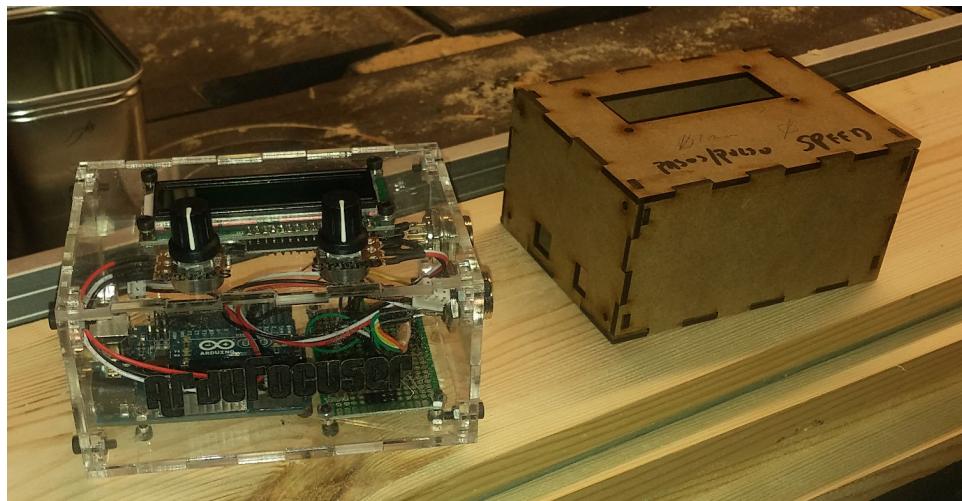


Figura 5.13: **Resultado después del corte de la carcasa en metacrilato.** Previamente se realizan ensayos en materiales más baratos como DM [89].

## 5.7. Pruebas sobre el dispositivo hardware

La sección de pruebas las dividimos en varias partes, pasando pruebas de concepto de cada uno de los componentes y módulos individuales, pruebas de integración, pruebas de rendimiento.

- **Pruebas de concepto y de componentes:** Es necesario revisar que los componentes que hemos adquirido no tienen ningún defecto que de ser detectados más adelante pueden ser costosos de cambiar. En las tablas 5.2, 5.3 y 5.4 se muestran algunos de los casos de prueba realizados.
- **Pruebas de rendimiento:** Comprobar que el sistema es tolerante a fatigas, no se sobrecalienta ningún componente y continua siendo estable después de un tiempo prolongado. En la tabla 5.5 se muestran uno de los casos de prueba realizados.

ID caso de prueba	1
Nombre prueba	Prueba lectura potenciómetros
Autor de la prueba	José Miguel López
Responsable diseño	José Miguel López
Pasos y condiciones ejecución	<ul style="list-style-type: none"> <li>- Compilar y cargar sketch (D.1)</li> <li>- Conectar placa a la alimentación</li> <li>- Variar resistencia en los potenciómetros.</li> </ul>
Resultado deseado	Se debe pintar por pantalla valores de 0 a 100 proporcionales a la resistencia.
Resultado obtenido	Se pintan por pantalla valores de 0 a 100 proporcionales a la resistencia.
Estado caso de prueba	Éxito
Errores asociados	Ninguno
Comentario	

Tabla 5.2: Caso de prueba, lectura de potenciómetros

ID caso de prueba	2
Nombre prueba	Prueba mover motor velocidad constante
Autor de la prueba	José Miguel López
Responsable diseño	José Miguel López
Pasos y condiciones ejecución	<ul style="list-style-type: none"> <li>- Compilar y cargar sketch (D.2)</li> <li>- Conectar placa a la alimentación</li> </ul>
Resultado deseado	El motor debe girar a velocidad constante y de forma uniforme.
Resultado obtenido	El motor gira a velocidad constante y de forma uniforme.
Estado caso de prueba	Éxito
Errores asociados	Ninguno
Comentario	

Tabla 5.3: Caso de prueba, mover motor a velocidad constante

ID caso de prueba	3
Nombre prueba	Prueba mando Nunchuck
Autor de la prueba	José Miguel López
Responsable diseño	José Miguel López
Pasos y condiciones ejecución	<ul style="list-style-type: none"> <li>- Compilar y cargar sketch (D.3)</li> <li>- Conectar placa a la alimentación</li> <li>- Pulsar sobre los diferentes botones del mando.</li> </ul>
Resultado deseado	<p>Se debe pintar en la pantalla LCD:</p> <ul style="list-style-type: none"> <li>- LEFT: Cuando movemos la palanca hacia la izquierda.</li> <li>- RIGHT: Cuando movemos la palanca hacia la derecha.</li> <li>- Z: Al pulsar el botón Z.</li> <li>- C: Al pulsar el botón C.</li> </ul>
Resultado obtenido	<p>Se pinta en la pantalla LCD:</p> <ul style="list-style-type: none"> <li>- LEFT: Cuando movemos la palanca hacia la izquierda.</li> <li>- RIGHT: Cuando movemos la palanca hacia la derecha.</li> <li>- Z: Al pulsar el botón Z.</li> <li>- C: Al pulsar el botón C.</li> </ul>
Estado caso de prueba	Éxito
Errores asociados	Ninguno
Comentario	

Tabla 5.4: Caso de prueba, detectar pulsaciones en Wii Nunchuck

ID caso de prueba	4
Nombre prueba	Prueba temperatura motor
Autor de la prueba	José Miguel López
Responsable diseño	José Miguel López
Pasos y condiciones ejecución	<ul style="list-style-type: none"><li>- Compilar y cargar sketch (D.2)</li><li>- Conectar placa a la alimentación</li><li>- Dejar el circuito conectado durante 120 minutos.</li></ul>
Resultado deseado	La temperatura del motor debe ser normal no debe existir sobrecalentamiento.
Resultado obtenido	El motor presenta una temperatura por encima de la normal
Estado caso de prueba	Error
Errores asociados	Se puede quemar el motor con un uso más prolongado.
Comentario	Se diagnostica posible causa, mal reglaje en la alimentación del motor.

Tabla 5.5: Caso de prueba, temperatura de motor tras un tiempo de funcionamiento prolongado



# Capítulo 6

## Módulo firmware

En este capítulo se expone el trabajo relacionado con la programación a bajo nivel realizada sobre la electrónica propuesta en el capítulo anterior.

Dado que la electrónica se basa en la plataforma Arduino, tal programación se realiza usando su propio lenguaje basado en C/C++.

El firmware es el encargado de detectar y controlar cambios en los estados de los periféricos y sensores que se produzcan por cualquier evento, interno o externo.

### 6.1. Diseño y Análisis

#### 6.1.1. Requisitos funcionales

Pasamos a determinar los requisitos que debe satisfacer nuestro firmware.

Los requisito funcionales están estrechamente relacionados con los requisitos propuestos en el anterior capítulo.

- **RF-1.:** Controlar motor paso a paso:
  - **RF-1.1.:** Controlar la dirección.
  - **RF-1.2.:** Controlar velocidad.
  - **RF-1.3.:** Controlar microstepping, para conseguir la máxima resolución en los pasos. [62]
- **RF-2.:** Mostrar información del sistema mediante una pantalla LCD.
- **RF-3.:** Modificar parámetros de forma manual

- **RF-3.1.:** Modificar velocidad.
- **RF-3.2.:** Modificar mover n pasos por pulsación.
- **RF-4.:** Ejecutar funciones de movimiento del motor desde una botónera o control manual.
  - **RF-4.1.:** Ejecutar función de movimiento continuo en la dirección deseada.
  - **RF-4.2.:** Ejecutar función de movimiento (n pasos) en la dirección deseada.
- **RF-5.:** Monitorizar temperatura y mostrar aviso cuando se detecte un cambio.
- **RF-6.:** Controlar límites físicos en el desplazamiento de las partes mecánicas del enfocador.
- **RF-7.:** Hacer las sesiones persistentes, recordando los estados después de una desconexión.
- **RF-8.:** Modificar y consultar los estados del dispositivo, mediante una API.
- **RF-9.:** Se debe permitir el encender y apagar la retro-iluminación de la pantalla.

### 6.1.2. Requisitos no funcionales

- **RNF1.:** El sistema debe responder a las órdenes en tiempo real y de forma interactiva.
- **RNF2.:** La velocidad de giro del motor no debe estar condicionada por la velocidad del microcontrolador.
- **RNF3.:** La pantalla LCD debe refrescarse solo cuando existan cambios en los datos a presentar.
- **RNF4.:** La comunicación serie debe ser robusta y tolerante y el protocolo debe ser tolerante a perdida de mensajes.
- **RNF5.:** El protocolo serie debe ser flexible y permitir extenderse con nuevos mensajes o cambiar por completo el repertorio de los mismos.
- **RNF6.:** La escritura en memoria EEPROM tiene que hacerse de forma eficiente y limitando el número de escrituras en lo posible, dado que es un elemento que se deteriora.

Todo ello debe implementarse con una buena calidad de código permitiendo añadir más módulos y consiguiendo que la API sea ampliable a nuevos comandos.

Con la experiencia y el trabajo realizado nos hemos ido familiarizándonos con la plataforma, conociendo nuevas bibliotecas de utilidad, nuevos periféricos, así como mediante sesiones de **refactorización** se consigue hacer el código más **limpio, mantenible y modular**.

Las entrada al sistema hardware de enfoque, proviene de cuatro fuentes.

1. Los sensores inherentes al dispositivo.
2. Variables de configuración o estados de la sesión anterior.
3. Controles manuales, ya sean botoneras o potenciómetros.
4. Controles remotos que llegan al dispositivo en forma de comandos por el puerto serie.

Por tanto el sistema, se descompone en los siguiente módulos.

- **Módulo para el control de motores**, encargado de actuar sobre el motor, de forma precisa. Debe controlar variables y estados tales como posición actual, velocidad, aceleración, sentido de giro y respetando los módulos de sensores y control.
- **Módulo de sensores**, que se ocupa de manejar la información proveniente de sensores externos, tales como sensores finales de recorrido o un sensor de temperatura. Debe controlar el estado de dichos sensores (presencia de un tope que limite el movimiento en uno de los sentidos, temperatura de último enfoque y actual).
- **Módulo control manual**, que se encarga de leer los diferentes botones y potenciómetros, con los que el usuario puede manejar directamente el dispositivos. Tenedremos estados para cada uno de los botones y los comunicara al módulo de motores. Si cambia la configuración o sensor lo comunicará por puerto serie y/o módulo de visualización LCD.
- **Módulo de visualización**, que se encargará de pintar la información en la pantalla LCD.
- **Módulo de control remoto**, su trabajo será estar a la escucha de los posibles comandos que puedan llegar por el puerto serie y comunicarlo al modulo que corresponda.
- **Modulo de configuración y almacenamiento de sesión**, que se encargará de manejar todas las variables de configuración, así como al-

macenarlos en EEPROM para que se mantengan de forma persistente para la próxima sesión de trabajo.

## 6.2. Arquitectura firmware

Para la programación del firmware se sigue un diseño donde se diferencian cuatro grandes bloques funcionales.

- **Initializer:** Método `init()` de la API de Arduino, que se ejecuta en primera instancia, prepara todo el entorno de ejecución, declarando las entradas y salidas, reservando memoria, iniciando interrupciones, así como ejecutar rutinas como cargar sesión anterior o crear instancias a modo de *singleton* para los manejadores de los diferentes periféricos.
- **Controlador principal:** Bucle principal que se ejecuta periódicamente (el periodo no se puede modificar y lo marca la velocidad del reloj del micro), por lo tanto no es adecuado ejecutar rutinas con restricciones temporales fuertes. Se corresponde con el método `loop()` de la API de Arduino y se ocupa de manejar la mayoría de las entradas y salidas.
- **Interrupciones software,** se ejecutan en un periodo marcado por un *timer*, similar al loop pero con periodos fijos. Se usa para rutinas con restricciones temporales fuertes.
- **Interrupciones hardware,** se ejecutan rutinas ligadas directamente a eventos de entrada y salida en alguno de los pines habilitados para tal fin.

Podemos ver un diagrama simplificado en la figura 6.1.

## 6.3. Módulo de control de motores

Para conseguir precisión en el control del motor, hacemos uso de la biblioteca **AccelStepper** [3], compatible con el controlador usado POLOLU A4988. Sus características son las siguientes:

- Soporta aceleración y desaceleración.
- Soporta múltiples motores paso a paso, de forma concurrente.
- La API no añade bloqueos o retardos por `delays()`.
- Soporta diversos tipos de controladores.
- Permite velocidades muy lentas.

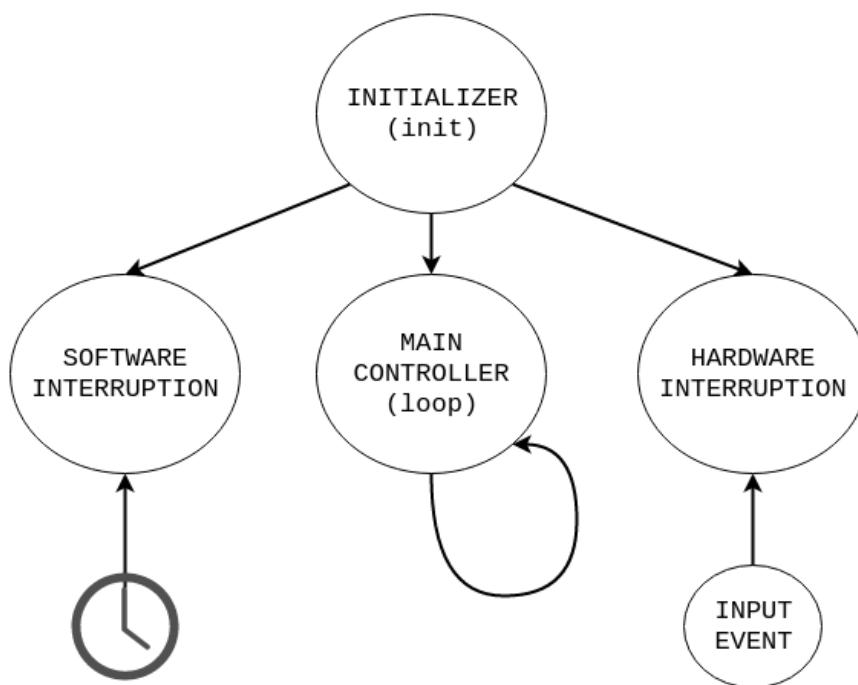


Figura 6.1: Flujos de ejecución del firmware y los eventos que activan los flujos.

- La API es extensible.

En el fragmento de código 6.1 podemos ver un ejemplo de uso sencillo de esta biblioteca.

```

1  #include <AccelStepper.h>
2  // Declaramos pines digitales donde se conecta el controlador.
3  #define PINDIR 3
4  #define PINSTEP 2
5
6  AccelStepper motor(1, PINSTEP, PINDIR);
7
8  void setup() {
9    // Asigna velocidad.
10   motor.setMaxSpeed(200);
11   // Asigna Aceleración.
12   motor.setAcceleration(1000);
13 }
14
15 void loop() {
16   // Inicia movimiento.
17   motor.run();
18   // Asigna posición de destino.
19   motor.moveTo(3000);
}
  
```

Fragmento de Código 6.1: Ejemplo de uso de AccelStepper, biblioteca usada para controlar motores paso a paso

## 6.4. Módulo de control remoto

Para comunicar el dispositivo con las aplicaciones que se ejecutan en un ordenador o servidor, hacemos uso de una comunicación mediante puerto serie. Para ello creamos un protocolo basado en comandos y parámetros.

El puerto serie envía la información mediante una secuencia de bits por los conectores, RX (recepción) y TX (transmisión). En Arduino UNO y Mini Pro los pines empleados son 0 (RX) y 1 (TX). En nuestro caso la comunicación utilizará una velocidad de transferencia de 9600 baudios.

La nomenclatura seguida todos los comandos tiene el siguiente formato:

COMANDO?ARGUMENTO1?ARGUMENTO2

Y la lista de comandos completa que hemos definido se puede encontrar en la tabla 6.1.

Todo comando tiene un mensaje de respuesta, bien indicando el nuevo estado del sistema que se ha cambiado (comandos escritura), bien información sobre el estado solicitado (comandos lectura).

El protocolo de comandos es extensible, pudiendo incorporar nuevos comandos o cambiando por completo el repertorio.

### 6.4.1. Implementación del protocolo serie

Para el parseo de los comando que se comunican por puerto serie, en un principio se estimó la solución de implementar una rutina propia. Conforme el repertorio de instrucciones creció y el formato de los parámetros se complicó, el código empezó a ser poco mantenible.

Por ello se introdujo la biblioteca **SerialCommand** [54]. Tal librería abstrae al programador del problema de interpretar los caracteres que se comunican por el puerto serie. Para ello debemos asociar cada comando válido con una función *callback* que se ejecutará inmediatamente después de su recepción, todo ello haciendo un uso eficiente del buffer serie y sin necesidad de analizar manualmente ningún mensaje recibido. En el fragmento de código 6.2 se puede ver un ejemplo de uso de esta biblioteca.

Comando	Parámetro	Descripción
AINIT	-	Iniciar modo funcionamiento Ardufocuser.
AMODE	MODE (int)	Establecer modo de funcionamiento.
AG	POSITION (int)	Mover enfocador hasta posición pasada como dato.
APOSITION	-	Solicitar posición actual del enfocador.
ATEMP	-	Solicitar temperatura de la lente.
AMICRO	MICRO (int)	Establecer multiplicación del micropaso.
AFINE	STEEP PER PULSE (int)	Establecer pasos que da el enfocador por cada pulso.
ASPEED	SPEED (int)	Establecer la velocidad del movimiento.
AACC	ACCELERATION (int)	Establecer la aceleración del movimiento.
AR	POSITION (int)	Fijar posición relativa del enfocador en un valor personalizado.
AHLIMIT	-	Consultar si el enfocador ha llegado a un límite hardware.
ASLIMIT	-	Consultar si el enfocador ha llegado a un límite software.
ASILIMIT	POSITION (int)	Establecer límite software inware.
ASOLIMIT	POSITION (int)	Establecer límite software outware.
AVERS	-	Consultar versión del firmware.
AMOV	-	Consulta si el enfocador esta moviéndose.
ALUX	LUX (bool)	Enciende o apaga la iluminación de la pantalla LCD.
ARUNA	-	Comando para debug: Mueve el motor en una dirección.
ARUNB	-	Comando para debug: Mueve el motor en una dirección.
ASTOP	-	Fuerza el motor a detener la marcha.
ALCDPRINT	MESSAGE (String)	Imprime el mensaje deseado en la pantalla LCD.

Tabla 6.1: Repertorio de Comandos Ardufocuser

```

1 #include <SerialCommand.h>
2 #define arduinoLED 13
3 // Objeto SerialCommand
4 SerialCommand sCmd;
5 void setup() {
6     Serial.begin(9600);
7     pinMode(arduinoLED, OUTPUT);
8     digitalWrite(arduinoLED, LOW);
9     // Callback, cuando se introduce el comando LED_on
10    sCmd.addCommand("ON", LED_on);
11    // Callback, cuando se introduce el comando LED_off
12    sCmd.addCommand("OFF", LED_off);
13    // Callback por defecto. se ejecuta con un comando
14    desconocido.
15    sCmd.setDefaultHandler(unrecognized);
16 }
17 void loop() {
18     // Ejecuta lectura y parseo de comando serie.
19     sCmd.readSerial();
}

```

Fragmento de Código 6.2: Ejemplo de uso de la biblioteca SerialCommand

## 6.5. Bibliotecas adicionales

A continuación se enumeran otras bibliotecas que han resultado especialmente útiles:

- **TimerOne:** Maneja las interrupciones programadas: Se ocupa de ejecutar ciertas funciones de forma periódica, sin necesidad de estar comprobando continuamente la hora.
- **Nunchuck:** Nos proporciona una API cómoda para utilizar el controlador Wii Nunchuck usando el puerto I2C.
- **EEPROMEx:** Maneja la lectura y escritura de datos en EEPROM, pudiendo escribir objetos de forma eficiente.

## 6.6. Herramientas utilizadas

Las herramientas utilizadas para realizar la programación del firmware han sido:

- **Arduino IDE**, para compilar y cargar el código fuente [9] (figura 6.2).

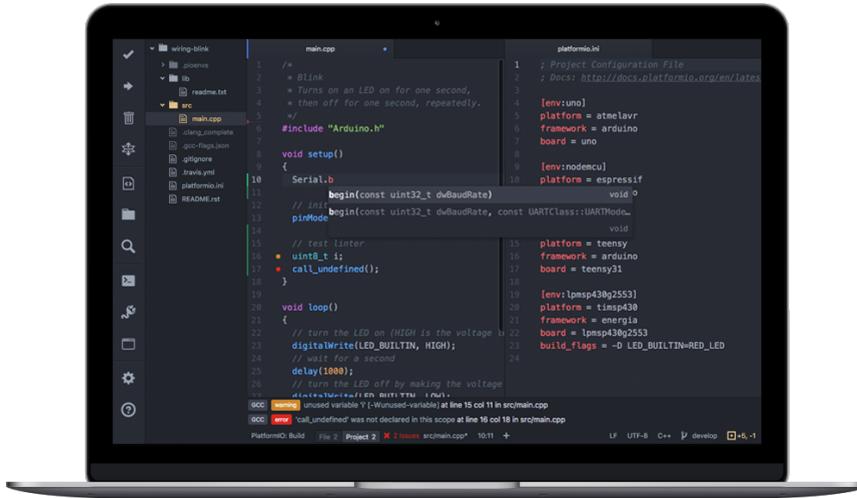


Figura 6.2: Interfaz de programación Arduino, con un fragmento de código. La interfaz permite compilar y cargar el ejecutable en la placa Arduino.

- **Sublime Text**, muy cómodo para editar rápidamente el código fuente. Además permite opciones avanzadas de búsqueda, reemplazo y refactorización [88].
- **Arduino UNO** junto con electrónica desarrollada en el capítulo anterior, para comprobar que funciona la lógica [9].
- **PlatformIO**, que además de ser un IDE completo para numerosas plataformas hardware, permite la compilación de hardware en el cloud, permitiendo también crear script **Travis** para realizar integración continua del código fuente [69].

## 6.7. Organización del código fuente

El código fuente del firmware queda distribuido en los siguientes ficheros:

- **Ardufocuser-config.h**: **Configuración**, así como el mapeo de pines y demás definiciones.
- **Ardufocuser-init.h**: **Inicializa el sistema**, creando las instancias de los objetos utilizados, variables globales, contadores y demás estados.
- **Ardufocuser-cmd.h**: **Comandos remotos**, junto con sus funciones callback asociadas.

- **Ardufocuser-utils.h:** Funciones de propósito general, útiles en algunos de los módulos.
- **Ardufocuser.ino:** Controlador principal, incluye los ficheros anteriores, contiene las funciones `setup()`, `loop()` y `callbacks` de las interrupciones hardware y software (fragmento de código 6.3).
- **library.json:** Referencia a las bibliotecas de terceros, catalogadas, y con referencias al sitio web y al desarrollador o equipo de desarrollo. Por seguridad también se incorporan al repositorio en el directorio `libs`.

```

void setup()
{
    // Inicia comunicación serie.
    Serial.begin(9600);

    // Inicia pantalla LCD.
    lcd.begin();
    lcd.backlight();

    // Saludo inicial.
    welcome(" ARDUFOCUSER ");

    // Velocidad y Aceleración inicial del motor.
    motor.setMaxSpeed(200);
    motor.setAcceleration(1000);

    // Iniciamos control Nunckuck
    chuck.begin();
    chuck.update();

    // Actualizamos con datos guardados en sesión
    // anterior
    load_session();

    // Iniciamos interrupciones Software.
    Timer1.initialize(50);
    Timer1.attachInterrupt(timerFunction);

    // Inicia interrupciones hardware a la escucha.
    attachInterrupt ( 0, finA,RISING);
    attachInterrupt ( 1, finB,RISING);

    // Registraremos y iniciamos comandos serie.
    registerCommand();
}

void loop()
{
    // Leemos nuevo comando serie.
    serial_cmd.readSerial();
    // Leemos control manual y sensores auxiliares.
}

```

```

    read_manual_controller();
    // Actualizamos LCD.
    update_lcd_display();
    // Leemos controles nunchuck.
    nunchuck_controller();
    // Almacenamos estados de forma persistente.
    save_current_session();
}

```

41  
42  
43  
44  
45  
46  
47  
48

Fragmento de Código 6.3: Núcleo implementación firmware ardufocuser

## 6.8. Pruebas

Se han llevado a cabo **pruebas unitarias** de cada módulo en concreto. Estas consisten en ejecutar individualmente las funciones involucradas en cada uno de los módulos observando si el comportamiento es adecuado.

Para ello hacemos uso de varios scripts adicionales:

- **test-eeprom-mem**: Testea módulo de almacenamiento en memoria EE-PROM.
- **test-hardw-intrp**: Testea módulo de interrupciones hardware.
- **test-lcd**: Testea módulo de visualización en la pantalla LCD.
- **test-motor**: Testea módulo de control del motor.
- **test-nunchuck**: Testea módulo controlador del Nunchuck.
- **test-serial-comand**: Testea módulo de comunicación serie.
- **test-temp-sensor**: Testea módulo del sensor de temperatura.
- **test-poten**: Testea el módulo de lectura de los potenciómetros.

La utilidad de estos script es inestimable, puesto que ayudan a detectar rápidamente posibles incidencias, tanto de **programación** como de **electrónica**. También ayudan a comprobar que los componentes que usamos en el montaje no tienen defectos de fábrica.

Adicionalmente se han llevado a cabo **pruebas de caja negra**, donde únicamente se tienen en cuenta las entradas que recibe el sistema y las salidas o respuestas que produce, sin preocuparnos del funcionamiento interno. Algunos de los casos de prueba que se han llevado a cabo se detallan en las tablas 6.2,6.3 y 6.4.

Por otro lado para asegurar que el firmware implementado permite ser compilado en las distintas placas hacemos uso del framework **PlatformIO** [69].

ID caso de prueba	8
Nombre prueba	Prueba ajustar posición enfocador mediante API serie.
Autor de la prueba	José Miguel López
Responsable diseño	José Miguel López
Pasos y condiciones de ejecución	<ul style="list-style-type: none"> <li>- Conectamos el dispositivo a la alimentación.</li> <li>- Conectamos el dispositivo mediante USB a un PC.</li> <li>- Abrirnos un cliente serie (Arduino IDE - Monitor Serial)</li> <li>- Configuramos una velocidad de <b>9600 baud</b>.</li> <li>- Mandamos comando <b>AG?50</b></li> </ul>
Resultado deseado	<ul style="list-style-type: none"> <li>- El motor del enfocador se moverá hasta la posición 50</li> <li>- Periódicamente responderá con el comando APOSITION?X, siendo X la posición actual.</li> <li>- Cuando alcance la posición 50, responderá con el comando ASTOP?</li> </ul>
Resultado obtenido	<ul style="list-style-type: none"> <li>- El motor del enfocador se mueve hasta la posición 50</li> <li>- Periódicamente responde con el comando APOSITION?X, siendo X la posición actual.</li> <li>- Cuando alcanza la posición 50, responde con el comando ASTOP?</li> </ul>
Estado caso de prueba	Éxito
Errores asociados	Ninguno
Comentario	

Tabla 6.2: Caso de prueba, ajustar posición enfocador mediante API serie.

ID caso de prueba	8
Nombre prueba	Prueba comando apagar iluminación LCD
Autor de la prueba	José Miguel López
Responsable diseño	José Miguel López
Pasos y condiciones de ejecución	<ul style="list-style-type: none"> <li>- Conectamos el dispositivo a la alimentación.</li> <li>- Conectamos el dispositivo mediante USB a un PC.</li> <li>- Abrirnos un cliente serie (Arduino IDE - Monitor Serial)</li> <li>- Configuramos una velocidad de <b>9600 baud</b>.</li> <li>- Mandamos comando <b>ALUX?</b></li> </ul>
Resultado deseado	Se debe apagar la retroiluminación de la pantalla.
Resultado obtenido	Se apaga la iluminación de la pantalla.
Estado caso de prueba	Éxito
Errores asociados	Ninguno
Comentario	

Tabla 6.3: Caso de prueba, apagar iluminación LCD mediante API serie.

ID caso de prueba	9
Nombre prueba	Prueba almacenamiento sesión
Autor de la prueba	José Miguel López
Responsable diseño	José Miguel López
Pasos y condiciones de ejecución	<ul style="list-style-type: none"> <li>- Conectamos el dispositivo a la alimentación.</li> <li>- Modificamos velocidad al valor 5.</li> <li>- Movemos enfocador hasta la posición 120.</li> <li>- Desconectamos alimentación y USB.</li> <li>- Conectamos alimentación.</li> </ul>
Resultado deseado	El dispositivo debe iniciar con velocidad 5 y posición del enfocador 120
Resultado obtenido	El dispositivo inicia en la posición 120 y velocidad 10.
Estado caso de prueba	Error
Errores asociados	No se almacena correctamente la velocidad.
Comentario	Se diagnostica error de almacenamiento.

Tabla 6.4: Caso de prueba, los estados son persistentes de una sesión para otra

En el fragmento de código 6.4 se muestra un script **Travis CI** que se ha añadido al repositorio del firmware. En tal script se añaden los comandos necesarios para compilar el firmware en el cloud así como realizar un test completo del firmware, con la integración de todos los módulos (figura 6.3).

#### Fragmneto de Código 6.4: Script travis para realizar integración continua

```

1 language: python
2 python:
3 - "2.7"
4
5 install:
6 - python -c "$(curl -fsSL https://raw.githubusercontent.com/platformio/platformio/master/scripts/get-platformio.py)"
7 - wget https://github.com/josemlp91/ardufocuser_firmware/raw/master/
     Ardufocuser/libs/TimerOne.zip
8 - unzip TimerOne.zip
9 - wget https://github.com/josemlp91/ardufocuser_firmware/raw/master/
     Ardufocuser/libs/i2clcd.zip
10 - unzip i2clcd.zip
11 - wget https://github.com/josemlp91/ardufocuser_firmware/raw/master/
     Ardufocuser/libs/AccelStepper-1.49.zip
12 - unzip AccelStepper-1.49.zip
13 - wget https://github.com/josemlp91/ardufocuser_firmware/raw/master/
     Ardufocuser/libs/nunchuck.zip
14 - unzip nunchuck.zip
15
16 script:
17 - platformio ci Ardufocuser --lib="TimerOne" --lib="i2clcd" --lib=
     nunchuck" --lib="AccelStepper" --board=uno

```

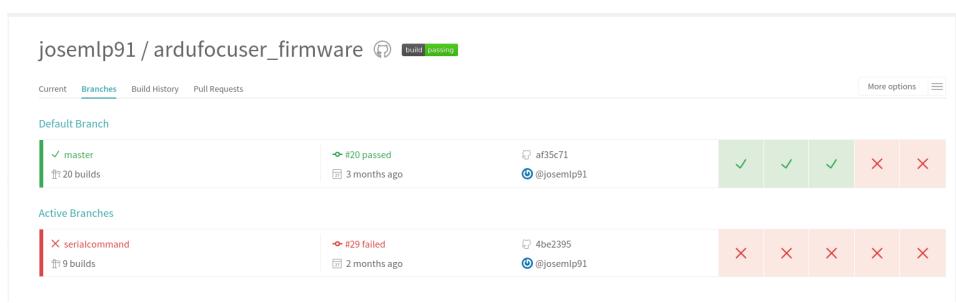


Figura 6.3: Ejecución del tests **Travis CI** tras realizar subidas al servidor de GitHub.

## Capítulo 7

# Módulo Driver INDI

En esta parte del proyecto nos centraremos en exponer como se ha creado el driver INDI específico para el dispositivo desarrollado bajo la tecnología Arduino en los capítulos anteriores de este mismo proyecto.

Este driver dotará a este dispositivo de la funcionalidad necesaria para poder funcionar en remoto, bajo el protocolo de control INDI, destinado a dispositivos astronómicos.

Para ello se trabajará con la versión implementada en Java de este protocolo, **INDI for Java** [50], por lo cual el lenguaje de programación que usaremos es Java.

### 7.1. Breve introducción a INDI

INDI consiste a su nivel más básico en un protocolo que permite el control, automatización, obtención de datos e intercambio de los mismos entre distintos dispositivos hardware y programas cliente.

La idea subyacente en el protocolo INDI es desacoplar aspectos específicos del hardware, haciendo que cambios en el hardware no impliquen necesariamente cambios en el software.

Para conseguir un desacople efectivo entre los clientes y el hardware se define un protocolo basado en **XML** que permite abstraer los dispositivos hardware como conjuntos de **propiedades** que pueden ser leídas, y modificadas por los clientes (siempre estableciendo las restricciones oportunas).



Figura 7.1: Logo de INDI Fuente: [46]

### 7.1.1. Drivers, Servidores y Clientes INDI

Pese a que nivel más básico INDI es “simplemente” una especificación de un protocolo basado en XML, a un nivel superior se distinguen tres entidades diferentes que interaccionan entre sí para tener un sistema de control plenamente funcional (figuras 7.2 y 7.3):

- **Drivers:** Son programas que se ejecutarán (normalmente) en la máquina donde se conectan los dispositivos hardware. Son los encargados de la comunicación directa con los dispositivos y su abstracción a propiedades INDI.
- **Servidor:** Es un programa cuya función principal es ejecutar los drivers y permitir la conexión a los mismos por parte de los clientes (funciona de un modo similar a un proxy). Normalmente reside en la máquina donde están conectados los dispositivos, aunque en principio se pueden crear estructuras de red tipo árbol de servidores. El intercambio de información entre el servidor y los drivers se realiza utilizando el protocolo INDI.
- **Cliente:** Es un programa que permite conectar con uno o más servidores y su función principal es hacer de **interfaz** con el usuario. Para ello conecta (usualmente a través de la red) con el servidor e intercambia información sobre los dispositivos utilizando el protocolo INDI. Es interesante recalcar que los clientes pueden ser de cualquier estilo: desde programas con interfaz de usuario avanzadas, hasta programas simples en línea de comandos scripts completamente automáticos que controlen o monitoricen los dispositivos.

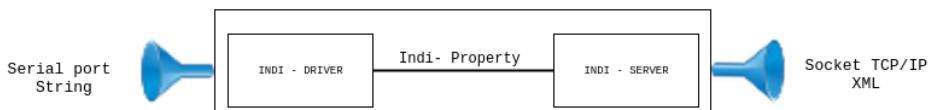


Figura 7.2: **Pipeline INDI**, flujo de información y conversión en diferentes entidades. (String Comando Serie - Propiedad Indi - Documento XML )

La biblioteca **INDI** original está escrita en lenguaje **C**, pero existe una implementación completa realizada en **Java** y que se encuentra en constante mejora. En la página oficial de INDI podemos encontrar toda la información sobre nuevas versiones y la documentación para poder utilizarla. La principal ventaja de poder usar **Java** es que podemos implementar drivers y clientes con la potencia de un lenguaje orientado a objetos y combinarlo con otras tecnologías como los dispositivos móviles basados en la plataforma **Android** ([enlace descarga de Remote Observatory \(App Android\)](#))

### 7.1.2. Propiedades INDI

El protocolo INDI define un conjunto de 5 propiedades que envían los diferentes drivers para formar la interfaz del cliente. Las propiedades se relacionan con el tipo de dato que maneja el dispositivo para controlar una característica concreta:

- **Textos:** Son cadenas de caracteres ordenados arbitrariamente.
- **Números:** Son cantidades numéricas. Además de la cantidad numérica se envían otros parámetros que sirven para el formato de su visualización y configuración (mínimo, máximo, paso, etc.).
- **Switches:** Son propiedades binarias con estado encendido o apagado. Sus agrupaciones pueden ser de tres tipos diferentes:
  - **Una de muchas:** para todas las opciones se tiene que seleccionar obligatoriamente una.
  - **Como máximo una:** de todas las opciones se puede seleccionar como máximo una.
  - **Cualquiera de muchas:** se podrán seleccionar todas las que se deseen.
- **Lights:** Son propiedades de solo lectura, que pueden estar en cada uno de los cuatro estados que se definen.
  - **Inactivo:** Luz de color gris.
  - **Alerta:** Luz de color rojo.

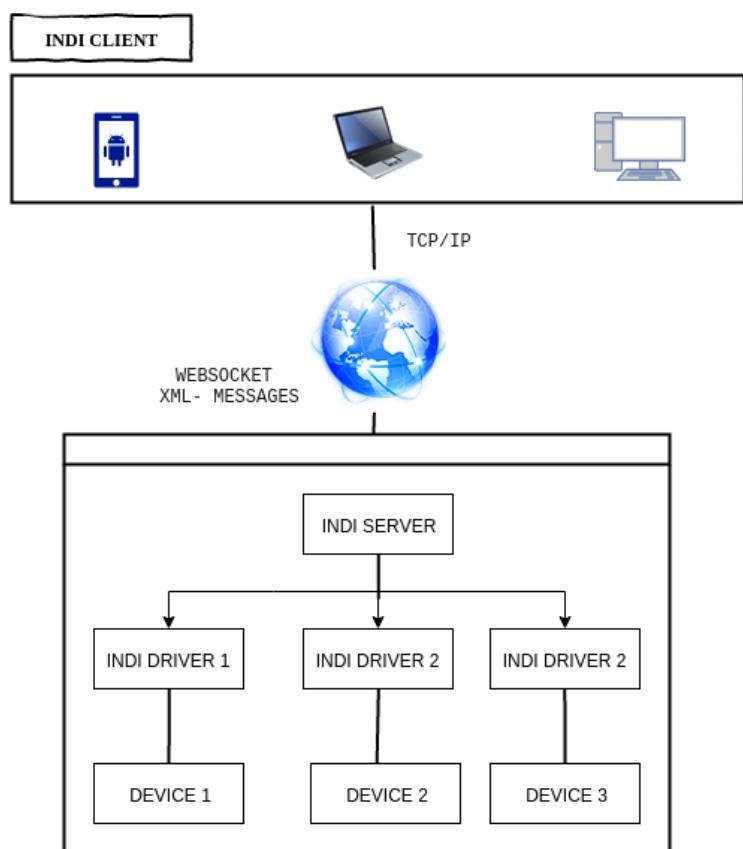


Figura 7.3: Arquitectura Servidor - Driver INDI

- **Ocupado:** Luz de color amarillo.
- **Ok:** Luz de color verde.
- **BLOBs:** Datos binarios cualesquiera.

## 7.2. Diseño del Driver INDI

A continuación se desglosa información sobre el proceso de diseño y planificación de este módulo.

### 7.2.1. Requisitos funcionales

- **RF-1.:** Crear conexión con dispositivo enfocador Ardufocuser.
- **RF-2.:** Controlar posición del enfocador Ardufocuser, permitir moverlo a la posición deseada.
- **RF-3.:** Modificar velocidad del enfocador.
- **RF-4.:** Modificar pasos por pulso que se mueve.
- **RF-5.:** Modificar posición relativa del enfocador.
- **RF-5.:** Mostrar temperatura del enfocador.
- **RF-5.:** Establece límite software del enfocador.
- **RF-6.:** Mostrar aviso si se alcanza un límite software.
- **RF-7.:** Mostrar aviso si se alcanza un límite hardware.
- **RF-8.:** Encender y apagar la iluminación de la pantalla lcd.

### 7.2.2. Requisitos no funcionales

- **RNF-1.:** El intercambio de mensajes por el puerto serie tiene que ser eficiente y tolerante a perdida de mensajes.
- **RNF-2.:** La información ha de presentarse al usuario de una forma clara e intuitiva.
- **RNF-3.:** Debe ser compatible con los clientes INDI que existen actualmente.

### 7.2.3. Arquitectura INDI for Java

Debemos conocer la estructura que tiene un Driver INDI. Para ello se recurrimos a la referencia oficial indiforjava, donde podemos encontrar una buena guía para implementar un Driver, junto un ejemplo simple.

**INDI for Java**, ya cuenta con una clase abstracta que define muchos de los atributos y métodos mínimos con los que debe contar un enfocador “Focuser” genérico. Debemos heredar de dicha clase para definir nuestro `ArduFocuserDriver` particular, añadiendo nuevos atributos e implementando los métodos abstractos existentes.

**Propiedades a implementar** Para ello lo primero que tenemos que hacer es definir exactamente las propiedades que tiene nuestro dispositivo, el tipo y si son lectura, escritura o ambas.

Dichas propiedades (tabla 7.1) serán comunicadas al cliente INDI para que este puede hacer una representación en la interfaz gráfica.

Descripción	RW/OW/OR	Tipo Propiedad	Mininimo Máximo Step
Posición Actual	RW	Number	MinimumAbsPos MaximumAbsPos 1
Velocidad	RW	Number	1 MaximumSpeed 1
Movimiento enfocador	OR	SwitchOneOrNone	–
Pasos por pulso	RW	Number	1 99 1
Temperatura	OR	Number	–
Límite Hardware	OR	Switch	–
Límite Software	OR	Switch	–
Posición Relativa	RW	Number	MinimumAbsPos MaximumAbsPos 1
Iluminación LCD	RW	Switch	–

Tabla 7.1: Propiedades del Driver INDI del Ardufocuser

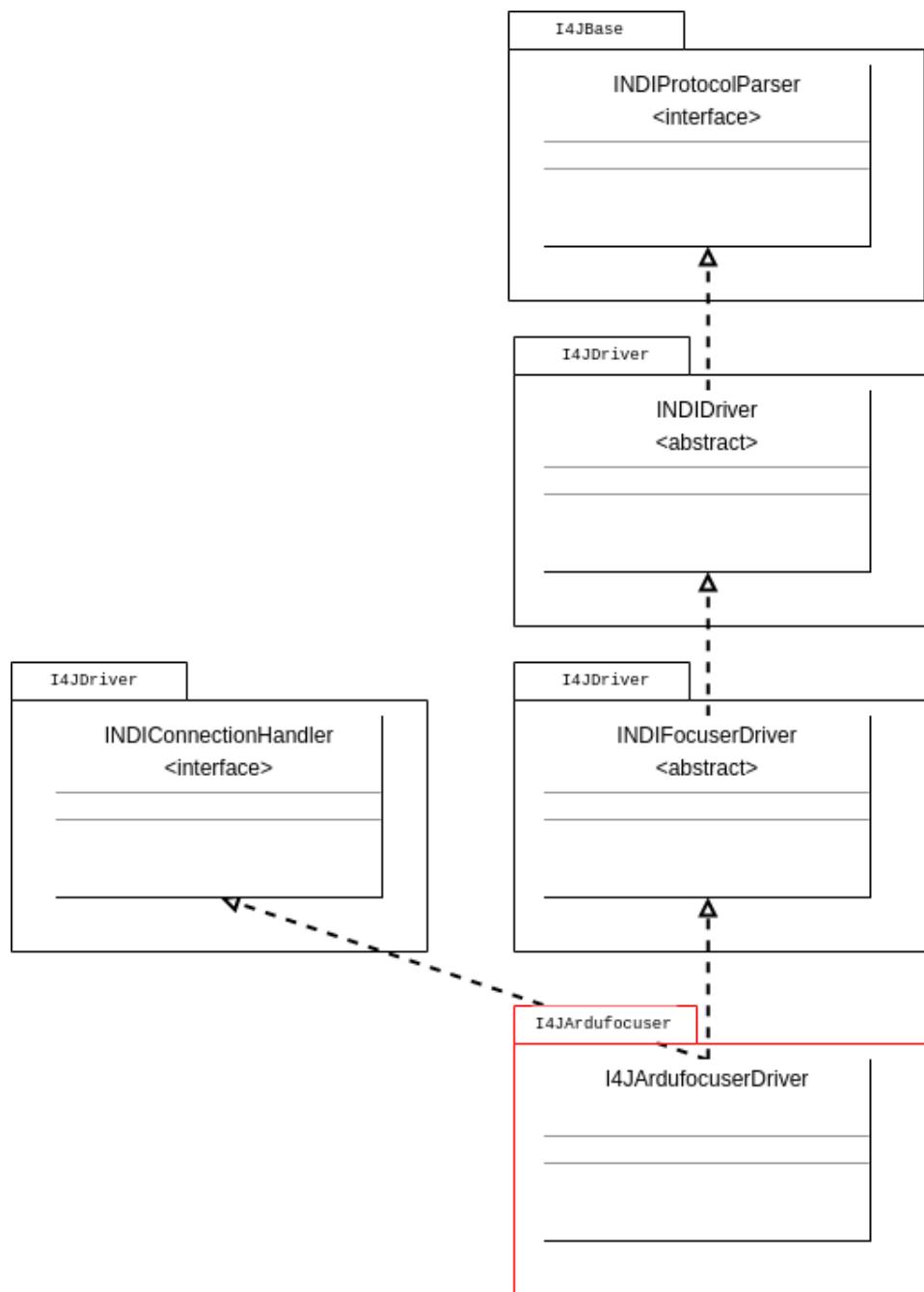


Figura 7.4: Diagrama de clases Driver INDI

### 7.3. Implementación

**Gestionar conexión serie** La comunicación directa con **Ardufocuser** se basa en el protocolo serie. Por tanto el primer paso es hacer que nuestra rutina Java se conecte al puerto serie de Arduino. Para ello se hace uso de la biblioteca **RXTX library**, junto con un conector específico para Arduino, **JavaDuino** [44]. En el fragmento de código 7.1 podemos ver un ejemplo de su uso.

```

1 import gnu.io.SerialPortEvent;
2 import gnu.io.SerialPortEventListener;
3 public class Main {
4
5     public static void main(String[] args) {
6         // Abrir la conexión Arduino.
7         ArduinoConnection ac = new ArduinoConnection();
8         boolean connected = ac.connectToBoard();
9
10        if (connected) {
11            System.out.println("Conectado!");
12        } else {
13            System.out.println("No se puede
14                conectar con Arduino :-(");
15            return;
16        }
17
18        // Añadimos escuchador, que responde a un
19        // evento serie.
20        ac.addListener(new SampleListener(ac));
21        // Enviar mensajes por puerto serie.
22        ac.sendString("Arduino!");
23        // Cerrar la conexión.
24        ac.close();
25    }
26
27    private static class SampleListener implements
28        SerialPortEventListener {
29        private ArduinoConnection ac;
30        public SampleListener(ArduinoConnection
31            ac) {
32            this.ac = ac;
33        }
34
35        @Override
36        public void serialEvent(SerialPortEvent
37            serialPortEvent) {
38            // Callback que se ejecuta con un
39            // evento serie.
40            if (serialPortEvent.getEventType() ==
41                SerialPortEvent.DATA_AVAILABLE) {
42                String inLine = ac.readLine();
43            }
44        }
45    }
46}
```

```

36                     System.out.println("GOT: " +
37                         inLine);
38                 }
39             }
40 }
```

Fragmento de Código 7.1: Ejemplo biblioteca SerialCommand

`ArduinoConnection` cuenta con los siguiente métodos:

- `ArduinoConnection()` : Constructor principal.
- `connectToBoard()` : Se conecta a la placa Arduino.
- `addListener(SerialPortEventListener)` : Añade un escuchador, permite ejecutar una función callback dado un evento serie, sin bloquear el flujo de la rutina main.
- `sendString(String)` : Envía una cadena e texto a Arduino.

Otro aspecto importante en la implementación del driver INDI es la creación e inicialización de las propiedades que hemos comentado.

Para ello se debe crear una instancia `INDIProperty`, que es el objeto que representa a la propiedad en concreto, indicando el grupo de propiedades al que pertenece, el estado inicial así como si es de lectura, escritura o lectura escritura.

Una propiedad puede tener varios elementos: de manera informal podemos decir que los elementos son las diferentes variables que componen una propiedad. Cuando instanciamos un elemento debemos pasar como parámetro la propiedad a la que pertenece. En el fragmento de código 7.2 podemos ver un ejemplo de inicialización de una propiedad. Entre otras se muestra la propiedad definida en el ejemplo anterior.

```

1 // Propiedad numérica para informar de la temperatura.
2 private INDINumberProperty temperatureP;
3
4 // Elemento numérico para informar de la temperatura.
5 private INDINumberElement temperatureE;
6
7 // Inicializador propiedad para manejar temperatura.
8 private void initializeTempertureProperty() {
9     if (temperatureP == null) {
10         // Se crea la propiedad, con el estado inicial,
11         // indicando el conjunto al que pertenece y si
12         // es escritura/lectura.
13         temperatureP = new INDINumberProperty(this, "temperature", "Temperature", "Control",
14                                         PropertyStates.IDLE, PropertyPermissions.RO);
```

```

12         temperatureE = temperatureP.getElement( "temperature_value");
13         if (temperatureE == null) {
14             // La propiedad
15             temperatureE = new INDINumberElement(
16                 temperatureP, "temperature", "Temperature", "1", "1", "99", "1", "%f");
17         }
18     }

```

Fragmento de Código 7.2: Ejemplo iniciar una propieda INDI Numérica

## 7.4. Instalación servidor

Para llevar a cabo todas las pruebas necesarias durante el desarrollo el servidor INDI se ha instalado en un mini-pc Raspberry-Pi, haciendo uso de la distribución Raspbian. El propósito de esta máquina es doble, su uso principal como **servidor astronómico INDI** (figura 7.5) y además al contar con todo el entorno de Arduino, facilita la tarea de **actualización del firmware**. En la figura 7.6 encontramos un diagrama completo del sistema Ardufocuser contando con el servidor INDI en la Raspberry Pi.

Para ello se han instalado las siguientes herramientas:

- **Java**, entorno de ejecución Java.
- **Netbeans** Entorno de desarrollo Java.
- **Arduino IDE**, permite cargar el firmware en la placa.
- **KStars**, cliente INDI.
- **Servidor INDI + Driver INDI**.

La biblioteca necesaria para que Raspberry y Arduino se puedan comunicar mediante puerto serie es **RXTX** en su versión para Java.

```
>sudo apt-get install librxtx-java
```

En los fragmentos de código 7.3 y 7.4 se muestran los scripts que se han creado / modificado en la distribución Raspbian para ejecutar automáticamente el servidor INDI.

Fragmento de Código 7.3: Script de inicio del servidor INDI

```

1 #! /bin/sh
2 # /etc/init.d/indiserver-init
3

```

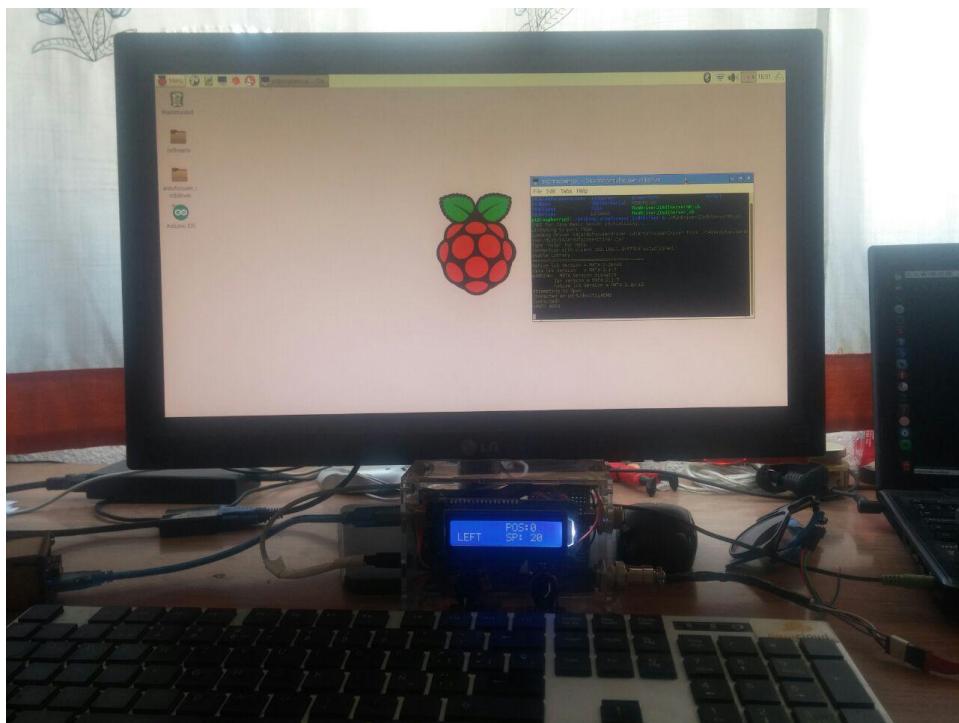


Figura 7.5: Servidor INDI funcionando en Raspberry Pi

```
4 ##### BEGIN INIT INFO
5 # Provides:                      indiserver-init
6 # Required-Start:                $all
7 # Required-Stop:                 $remote_fs $syslog
8 # Default-Start:                2 3 4 5
9 # Default-Stop:                 0 1 6
10 # Short-Description:           Run Indi Server.
11 # Description:                  Run Indi Server with Driver for Ardufocuser.
12 ##### END INIT INFO
13
14 case "$1" in
15 start)
16 echo "IndiServer Runing"
17 /home/pi/Desktop/ardufocuser_inidriver/RunDriver2IndiServerRPBeep.sh
18 ;;
19 stop)
20 echo "IndiServer Stoping"
21 ;;
22 *)
23 echo "Modo de uso: /etc/init.d/indiserver-init {start|stop}"
24 exit 1
25 ;;
26 esac
27
28 exit 0
```

Fragmento de Código 7.4: Script de inicio del servidor INDI (2)

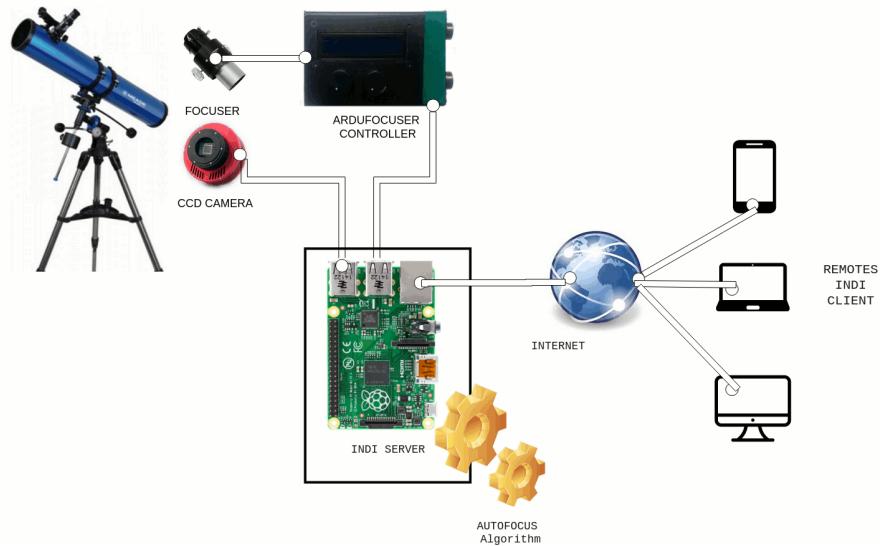


Figura 7.6: Diagrama completo del sistema Ardufocuser

```

1 #!/bin/bash
2
3 # Emite un beep por un zumbador.
4 python /home/pi/Desktop/ardufocuser_inidriver/beepOK.py
5
6 # Ejecuta Servidor INDI con el correspondiente Driver como módulo.
7 java -Djava.library.path=/usr/lib/jni -cp /usr/share/java/RXTXcomm.
     jar -jar /home/pi/Desktop/ardufocuser_inidriver/I4JServer/dist/
           I4JServer.jar -add=/home/pi/Desktop/ardufocuser_inidriver/
           I4JArdufocuserDriver/dist/I4JArdufocuserDriver.jar

```

#### 7.4.1. Crear imagen Rasbindi

Para facilitar la instalación del sistema en la Raspberry se ha creado una imagen de todo el entorno. La imagen la podemos descargar directamente del siguiente enlace:

Enlace para descargar **Rasbindi**, imagen personalizada con un servidor INDI configurado.

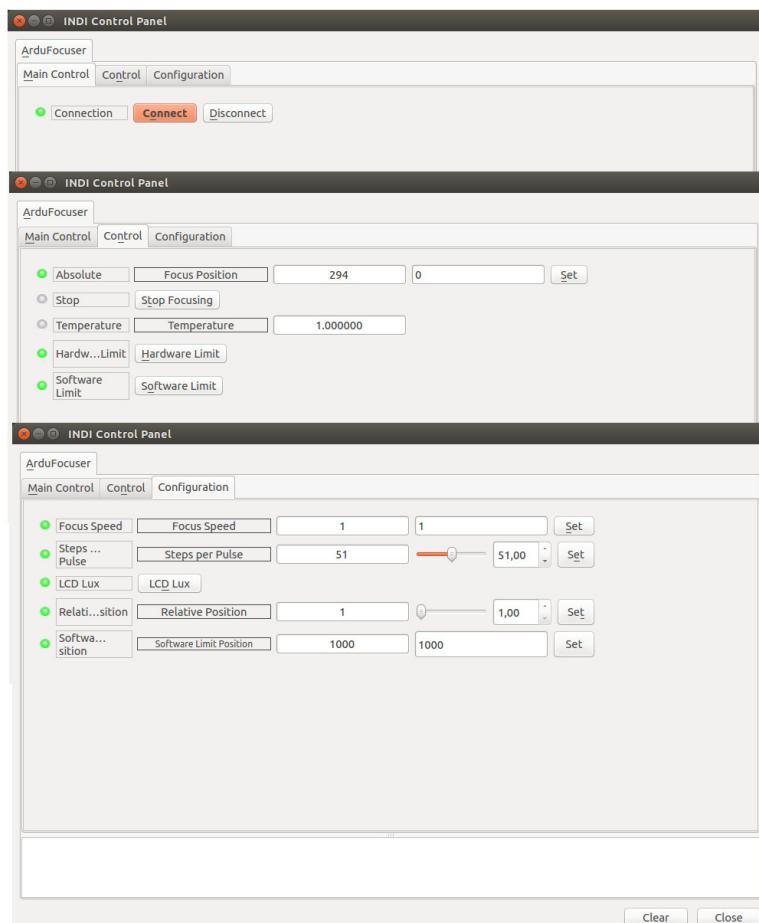


Figura 7.7: Se realizan pruebas desde KStars (Cliente de Escritorio)

## 7.5. Pruebas

Para este módulo se han realizado pruebas de caja negra, es decir, sin conocer detalles internos del dispositivo tratamos de reproducir posibles casos de uso, y se comprueba si las salidas corresponden con lo esperado.

Para las pruebas se han usado dos clientes INDI diferentes, anteriormente comentados, KStars (Linux) 7.7 y Observatorio Remoto (Android) 7.8.

En las tablas 7.2, 7.3, 7.4 y 7.5 se pueden ver ejemplos de dichos casos de prueba. Además, en las figuras 7.9 y 7.10 podemos ver los resultados visuales de la prueba de conexión (con el cliente kstars) al driver y encendido del mismo.

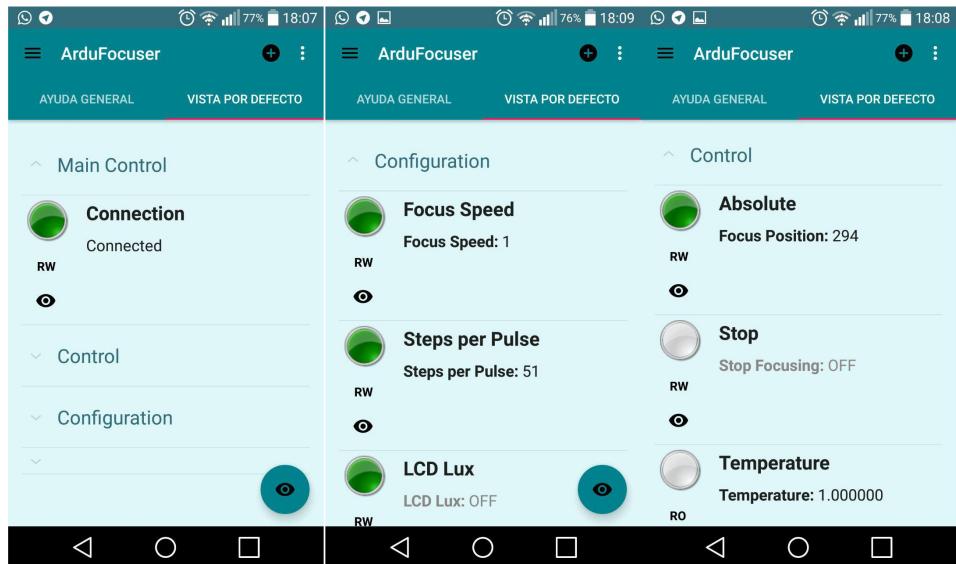


Figura 7.8: Se realizan pruebas desde Observatorio Remoto (Cliente Android)

ID caso de prueba	1
Nombre prueba	Conectar a servidor INDI con driver Ardufocuser
Autor de la prueba	José Miguel López
Responsable diseño	José Miguel López
Pasos y condiciones ejecución	<ul style="list-style-type: none"> <li>- Ejecutar cliente INDI.</li> <li>- Ir a Device Manager en KStars y pulsar en Add.</li> <li>- Introducir IP y puerto del servidor INDI.</li> <li>- Pulsar botón Connect.</li> </ul>
Resultado deseado	Debe aparecer una nueva ventana donde se muestran las propiedades y dispositivos INDI, organizados en pestañas.
Resultado obtenido	Aparece una nueva ventana donde se muestran las propiedades de los dispositivos INDI.
Estado caso de prueba	Éxito
Errores asociados	Ninguno
Comentario	

Tabla 7.2: Caso de prueba, conectar con servidor INDI

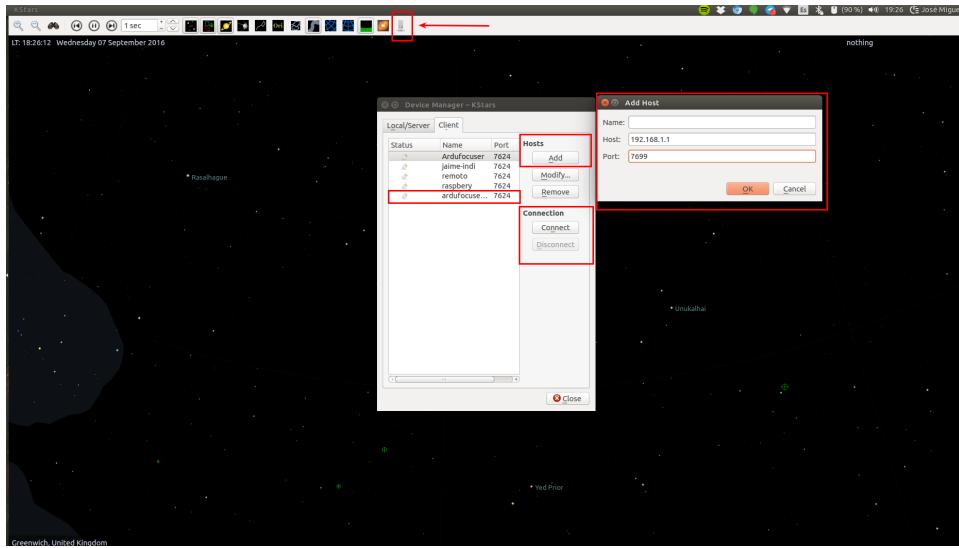


Figura 7.9: Conectando el servidor INDI con el driver Ardufocuser

ID caso de prueba	2
Nombre prueba	Conectar Ardufocuser
Autor de la prueba	José Miguel López
Responsable diseño	José Miguel López
Pasos y condiciones ejecución	<ul style="list-style-type: none"> <li>- Partimos del resultado obtenido en el caso de uso anterior.</li> <li>- Ir a la pestaña <b>Ardufocuser</b></li> <li>- Ir a la pestaña <b>Main Control</b></li> <li>- Pulsar <b>Connect</b></li> </ul>
Resultado deseado	Debe cambiar la luz de la propiedad a verde. Se activan dos pestañas adicionales <b>Control</b> y <b>Configuration</b> .
Resultado obtenido	Cambiar la luz de la propiedad a verde. Se activan dos pestañas adicionales <b>Control</b> y <b>Configuration</b> .
Estado caso de prueba	Éxito
Errores asociados	Ninguno
Comentario	

Tabla 7.3: Caso de prueba, conectar Ardufocuser

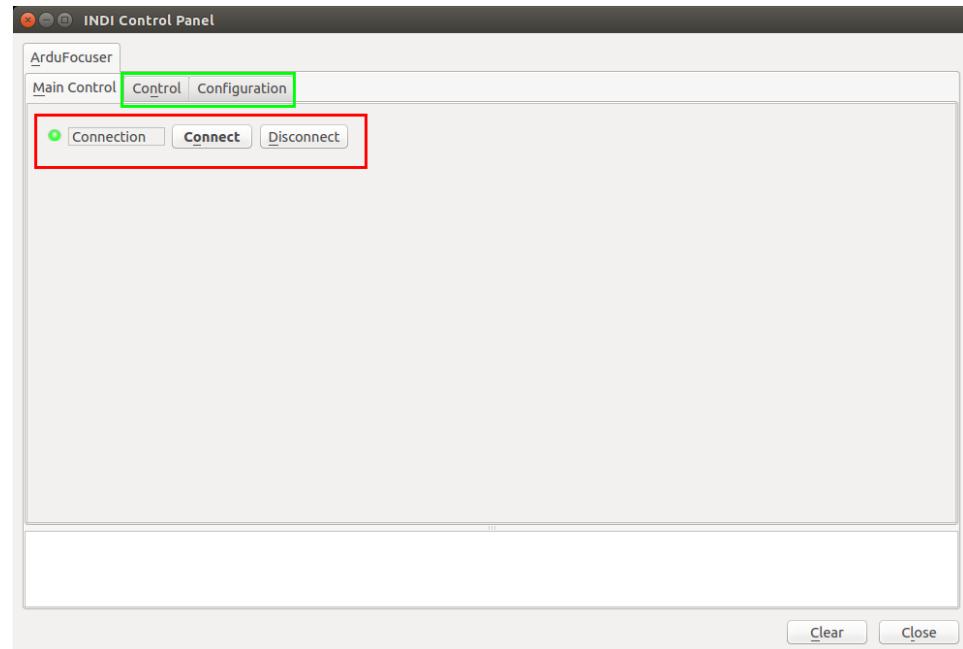


Figura 7.10: Conectando el Ardufocuser a través de Kstars

ID caso de prueba	3
Nombre prueba	Modificar velocidad del motor
Autor de la prueba	José Miguel López
Responsable diseño	José Miguel López
Pasos y condiciones ejecución	<ul style="list-style-type: none"> <li>- Partimos del resultado obtenido en el caso de uso anterior.</li> <li>- Ir a la pestaña Ardufocuser.</li> <li>- Ir a la pestaña Configuration.</li> <li>- Modificar valor Focus Speed y pulsar en Set.</li> </ul>
Resultado deseado	Debe cambiar la luz de la propiedad a verde. Debe cambiar el valor SP en la pantalla LCD
Resultado obtenido	Cambiar la luz de la propiedad a verde. Cambiar el valor SP en la pantalla LCD
Estado caso de prueba	Éxito
Errores asociados	Ninguno
Comentario	

Tabla 7.4: Caso de prueba, modificar velocidad

ID caso de prueba	4
Nombre prueba	Establecer posición del enfocador.
Autor de la prueba	José Miguel López
Responsable diseño	José Miguel López
Pasos y condiciones ejecución	<ul style="list-style-type: none"> <li>- Partimos del resultado obtenido en el caso de uso anterior.</li> <li>- Ir a la pestaña <b>Ardufocuser</b>.</li> <li>- Ir a la pestaña <b>Control</b>.</li> <li>- Modificar valor <b>Absolute</b> y pulsar en <b>Set</b>.</li> </ul>
Resultado deseado	<p>Debe cambiar la luz a rojo.</p> <p>El motor del enfocador comienza a girar</p> <p>Se debe ver el avance del motor en tiempo real</p> <p>Al pararse la luz vuelve a color verde.</p>
Resultado obtenido	<p>Cambiar la luz a rojo.</p> <p>El motor del enfocador comienza a girar</p> <p>Se ve el avance del motor en tiempo real</p> <p>Al pararse la luz vuelve a color verde.</p>
Estado caso de prueba	Éxito
Errores asociados	Ninguno
Comentario	

Tabla 7.5: Caso de prueba, establecer posición del enfocador



# Capítulo 8

## Módulo Software

Una vez creado el dispositivo nos centramos en la implementación de algoritmos para el enfoque. Resumidamente, estos algoritmos consisten en tomar una imagen, analizarla y evaluar su calidad, determinar si el enfoque es óptimo, en caso contrario movemos la posición del enfocador y reiniciamos el ciclo hasta encontrar el punto óptimo de enfoque.

### 8.1. Motivación

Actualmente el problema del autofocus esta resuelto en prácticamente todas las cámaras de fotos, incluidas las integradas en los teléfonos móviles. ¿Por qué es un problema particularmente interesante en astronomía?

- Imágenes de baja luminosidad.
- Un único plano focal, al contrario de las imágenes fotográficas convencionales donde contamos con profundidad, los objetos celestes se localizan en un único plano muy alejado al observador.
- Poco contraste, al ser imágenes poco iluminadas el contraste suele ser bajo.
- Muchos de los objetos celestes son puntuales y no tienen bordes ni detalles significantes.
- Efecto **seeing** [83], turbulencias producidas por la atmósfera, que distorsiona las imágenes.

Es por ello que el enfoque en astronomía tiene sus particularidades y hay que trabajarla de forma diferente.

## 8.2. Adquisición de Imágenes

El primer paso para trabajar con imágenes astronómicas es ser capaces de capturar dichas imágenes. Usualmente esto se realiza mediante un sensor CCD que obtiene un conjunto de imágenes en formato FITS.

Estas imágenes deben contener objetos de diferentes características y condiciones puesto que nos valen de entrenamiento a las que nos encontraremos en un escenario real donde tiene que actuar la rutina de enfoque astronómico.

## 8.3. Herramienta de visualización y Análisis

En este paso diseñamos una herramienta que permite analizar las características de la imagen, así como realizar diferentes operaciones sobre ella.

Previamente se estudian algunas plataformas sobre las que nos podemos apoyar.

- **Maplab:** Contamos con la biblioteca *fitsread*, que permite trabajar con imágenes FITS.
- **ImageJ:** Programa de procesamiento de imágenes digitales, ampliamente extensible a través de plugins y utilizado en numerosos ámbitos científicos como la medicina.
- **AstroPy** [75] (figura 8.1): Paquete Python que unifica herramientas y utilidades básicas necesarias en astronomía y astrofísica. Incluye los paquetes usados para las tareas más básicas (manejo de imágenes FITS, tablas, coordenadas, ...). Hace uso del paquete matemático **numpy**.



Figura 8.1: **Astropy**, framework procesamiento imágenes en python, con gran cantidad de rutinas ya implementadas.

- **Ginga: Image Viewer and Toolkit** (figura 8.2): Es un conjunto de herramientas diseñadas para visualizar los datos de imágenes científicas en Python.
- **MaxIm DL** [60], es una solución integral de análisis de imágenes astronómicas, con rutinas de enfoque automático. Es comercial, no es software

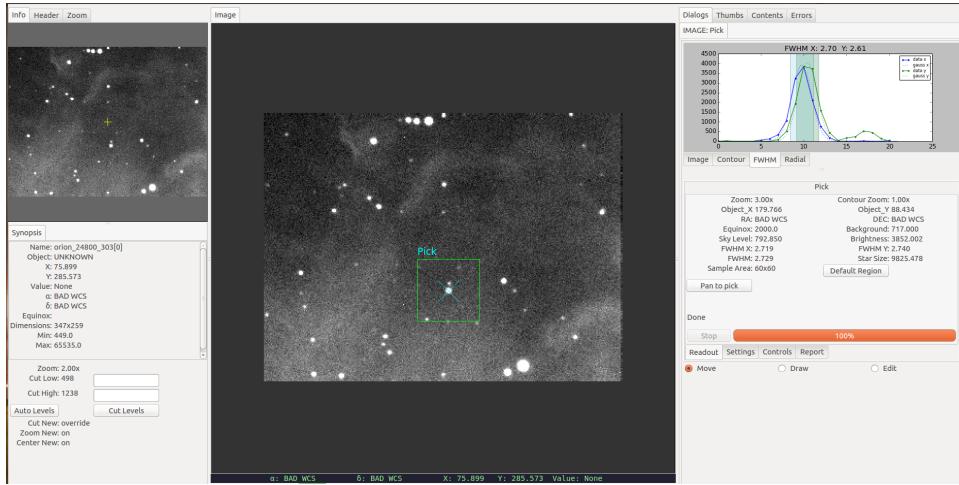


Figura 8.2: Ginga: Image Viewer and Toolkit

libre y su precio mínimo es de 200\$.

Tras este repaso de herramientas existentes (y tras probarlas todas) tenemos una visión más o menos amplia de lo que existe en el mercado y podemos estudiar si se pueden adaptar alguna de ellas a nuestro propósito.

Se determina que las soluciones anteriores no se adaptan exactamente a nuestro problema por diferentes causas, bien necesitan gran cantidad de software adicional para desplegarse, porque no se pueden integrar fácilmente con el driver INDI (recordemos que se encuentra implementado en Java) o son comerciales y privativas y tienen un elevado precio.

### 8.3.1. Planificación y temporización

Estimación de tiempos

- Planteamiento del problema
  - Reunión con cliente **2 horas**.
  - Análisis inicial **5 horas**.
  - Estudio del dominio **5 horas**.
  - Estudio de herramientas **5 horas**.
  - Fundamentos matemáticos curva de luz de las estrellas **8 horas**.
  - Cálculo del FWHM [37] **5 horas**.
- Análisis y diseño

- Diagramas **4 horas**.
- Diagrama algoritmo de detección **6 horas**.
- Implementación
  - Biblioteca FITS **5 horas**.
  - Módulo de visualización **8 horas**.
  - Resumen de la imagen **4 horas**.
  - Algoritmos de detección de estrellas **10 horas**.
  - Visualización de resultados gráficos **8 horas**.
  - Visualización de resultados avanzados **5 horas**.
- Pruebas **5 horas**
- Documentación **10 horas**

En total se estiman **95 horas** para el desarrollo de esta herramienta.

### 8.3.2. Análisis de requisitos

En este punto inicial del desarrollo software, paso a identificar los requisitos funcionales y no funcionales. El producto final debe cumplir cada uno de ellos para adaptarse a producto que nos describe el cliente mediante entrevistas.

Señalar que la finalidad última de esta aplicación es servir como herramienta para conseguir extraer unos parámetros de ajuste de los algoritmos de enfoque, y hacer comparaciones, aunque como objetivo secundario también puede ser muy útil para visualizar imágenes FITS.

**Requisitos funcionales** Los requisitos funcionales nos informan de los problemas que va a tratar de solucionar el producto que estamos desarrollando.

- **RF-1.:** Abrir ficheros FITS: desde un explorador de archivos debe permitir seleccionar y abrir un archivo fits.
- **RF-2.:** Representar los píxeles de la imagen permitiendo así una visualización de la imagen.
- **RF-3.:** Mostrar resumen de la imagen:
  - **RF-3.1.:** Anchura y altura en píxeles.
  - **RF-3.2.:** Valor del pixel más iluminado y menos iluminado.

- **RF-3.3.**: Media de iluminación de píxeles.
- **RF-4.**: Operaciones sencillas con las imágenes FITS.
  - **RF-4.1.**: Hacer zoom.
  - **RF-4.2.**: Configurar gamma.
  - **RF-4.3.**: Configurar máximos y mínimos del histograma.
  - **RF-4.4.**: Negativo.
- **RF-5.**: Visualizar de forma gráfica la ampliación de una región de la imagen.
  - **RF-5.1.**: Gráfico matricial.
  - **RF-5.2.**: Gráfico líneas.
- **RF-6.**: Algoritmos de detección y filtrado.
  - **RF-5.1.**: Detección de objetos.
  - **RF-5.2.**: Filtrado de objetos estelares.
- **RF-7.**: Representar resultado de los filtros de detección, coloreando las estrellas detectadas por cada uno de los filtros. También se debe permitir seleccionar mediante doble click los objetos interesantes.
- **RF-8.**: Resumen avanzado, con medidas como el número de objetos detectados o el valor FWHM del objeto seleccionado.

**Requisitos no funcionales** Los requisitos no funcionales son restricciones sobre la forma de completar los requisitos funcionales propuestos.

- **RNF1.**: Usabilidad para un usuario intermedio.
- **RNF2.**: Diseño modular, extensible con nuevas rutinas de detección de objetos, nuevas medidas, así como tratamiento con objetos planetarios.
- **RNF3.**: Eficiencia en los algoritmos de detección, para conseguir una rutina de enfoque rápida.

### 8.3.3. Casos de uso

Un caso de uso es una descripción de los pasos o las actividades que deber realizarse para llevar a cabo algún proceso. Los personajes o entidades que participarán en un caso de uso se denominan actores.

### Descripción de actores

- **Ac-1.** Usuario.
  - Descripción: Persona que utiliza la aplicación.
  - Características: Es un usuario medio/avanzado, familiarizado con la astronomía.  
Desarrolladores de rutinas de enfoque y detección de objetos.
  - Relaciones: Ninguna
  - Atributos: Ninguno
  - Comentarios: Ninguno.

### Descripción casos de uso

- **CU-1.** Abrir imagen (tabla 8.1).
  - **Actores:** Usuario.
  - **Tipo:** Primario, esencial.
  - **Referencias:**
  - **Precondición:**
  - **Postcondición:** El sistema carga un archivo de imagen y se visualiza en la pantalla principal.
  - **Autor:** José Miguel López.
  - **Versión:** 1.0.
  - **Propósito:** Introducir en el sistema un fichero de imagen.
  - **Resumen:** El usuario selecciona un archivo de imagen del sistema de ficheros de la computadora, al aceptar el sistema carga la imagen y se visualiza.
- **CU-2.** Modificar gamma (tabla 8.2).
  - **Actores:** Usuario.
  - **Tipo:** Primario, esencial.
  - **Referencias:**
  - **Precondición:** El usuario ha de haber completado el CU-1
  - **Postcondición:** Se actualiza la imagen con la nueva configuración de gamma.

Curso normal		
	Actor	Sistema
1	Usuario: Pulsa el botón para abrir una imagen.	
		2 El sistema muestra un explorador con los ficheros y directorios de la máquina.
3	Usuario: Selecciona un archivo con formato <b>fits</b> y pulsa en el botón de aceptar.	
		4 Se visualiza la imagen en pantalla Se muestra resumen de la imagen Anchura, altura, valor píxel más oscuro valor píxel más iluminado, media luminosidad de los píxeles.

Tabla 8.1: CU-1. Abrir imagen FITS.

- **Autor:** José Miguel López.
- **Versión:** 1.0.
- **Propósito:** Modificar visualización de la imagen.
- **Resumen:** El usuario modificar el valor del campo gamma. Para entender bien el efecto consultar la referencia Gamma [24].

Curso normal		
	Actor	Sistema
1	Usuario: Selecciona un valor de compensación de gamma entre 0.05 y 10	
		2 El sistema computa el valor de los píxeles y muestra la imagen con la corrección de gamma aplicada

Tabla 8.2: CU-2. Modificar Gamma.

- **CU-3.** Modificar ecuación del histograma (tabla 8.3).
  - **Actores:** Usuario.
  - **Tipo:** Primario, esencial.
  - **Referencias:**
  - **Precondición:** El usuario has de haber completado el CU-1

- **Postcondición:** Se actualiza la imagen con la nueva configuración del histograma.
- **Autor:** José Miguel López.
- **Versión:** 1.0.
- **Propósito:** Modificar visualización de la imagen.
- **Resumen:** El usuario modificar el valor mínimo o máximo del histograma.

Curso normal		
	Actor	Sistema
1	Usuario: Selecciona un valor de min y max entre el 0 y el valor del píxel más luminoso	
		2 El sistema muestra la imagen según una nueva ecuación de histograma, ello repercute en el brillo y contraste.

Tabla 8.3: CU-3.Modificar ecuación Histograma de la Imagen.

- **CU-4.** Modificar zoom (tabla 8.4).
  - **Actores:** Usuario.
  - **Tipo:** Primario, esencial.
  - **Referencias:**
  - **Precondición:** El usuario has de haber completado el CU-1
  - **Postcondición:** Se amplia o reduce la visualización de la imagen.
  - **Autor:** José Miguel López.
  - **Versión:** 1.0.
  - **Propósito:** Ampliar o reducir la visualización de la imagen.
  - **Resumen:** El usuario modificar el valor zoom y la imagen se amplia o reduce.
- **CU-5.** Generar gráfico de zona (tabla 8.5).
  - **Actores:** Usuario.
  - **Tipo:** Primario, esencial.
  - **Referencias:**

Curso normal		
	Actor	Sistema
1	Usuario: Selecciona un valor de zoom entre el 25 % y el 100 %	
		2 El sistema muestra la imagen con el nuevo escalado

Tabla 8.4: CU-4.Modificar Zoom.

- **Precondición:** El usuario has de haber completado el CU-1 y hace click en una zona de la imagen.
- **Postcondición:** Genera un gráfico en forma matriz (mapa de calor) y líneas con una ampliación de una zona de la imagen.
- **Autor:** José Miguel López.
- **Versión:** 1.0.
- **Propósito:** Generar gráfico de detalle de una zona.
- **Resumen:** El usuario hace click en una zona de la imagen y se muestran gráficos de luz de la zona con tales coordenadas.

Curso normal		
	Actor	Sistema
1	Usuario: Hace click en un punto de la imagen.	
		2 El sistema genera un gráfico de luz del entorno del punto seleccionado.

Tabla 8.5: CU-5.Generar Gráfico luz de zona.

- **CU-6.** Activar detección de objetos (tabla 8.6).
  - **Actores:** Usuario.
  - **Tipo:** Primario, esencial.
  - **Referencias:**
  - **Precondición:** El usuario has de haber completado el CU-1.
  - **Postcondición:** Se hace una búsqueda inicial de objetos, siguiendo el algoritmos de máximo locales y marcan con puntos de color.
  - **Autor:** José Miguel López.
  - **Versión:** 1.0.

- **Propósito:** Hacer una búsqueda inicial de objetos estelares.
- **Resumen:** El usuario activa la casilla de detección de objetos, y se pintan en pantalla puntos donde pueden existir objetos.

Curso normal		
	Actor	Sistema
1	Usuario: Activa función “Detect Stars” y establece valor “Radius surrounding peak” entre 1 y 20.	
		2 El sistema busca objetos estelares que respondan a un pico de luz en el radio seleccionado. Los objetos detectados son marcados en color.

Tabla 8.6: CU-6. Activar detección de objetos..

- **CU-7.** Generar gráfico de objeto (tabla 8.7).
  - Actores: Usuario.
  - Tipo: Primario, esencial.
  - Referencias:
  - Precondición: El usuario ha de haber completado el CU-1 y el CU-5
  - Postcondición: Genera un gráfico en forma matriz (mapa de calor) y líneas con una ampliación del objeto.
  - Autor: José Miguel López.
  - Versión: 1.0.
  - Propósito: Generar gráficos de detalle del objeto
  - Resumen: El usuario hace doble click en un objeto y se muestran gráficos de luz del objeto.
- **CU-8.** Aplicar filtro margen (tabla 8.8).
  - Actores: Usuario.
  - Tipo: Primario, esencial.
  - Referencias:

Curso normal		
Actor		Sistema
1	Usuario: Hace doble clic en un objeto detectado.	
		2 El sistema genera un gráfico de luz del entorno al objeto seleccionado.

Tabla 8.7: CU-7. Generar Gráfico luz de objeto.

- Precondición: El usuario ha de haber completado el CU-1 y el CU-5
- Postcondición: Se descartan los objetos situados a una distancia del borde de la imagen.
- Autor: José Miguel López.
- Versión: 1.0.
- Propósito: Descartar objetos cerca del borde de la imagen.
- Resumen:

Curso normal		
Actor		Sistema
1	Usuario: Marca la casilla “Filter Margin Stars” y establece un valor “Margin Size” entre 1 y 40.	
		2 El sistema marca las estrellas a distancia “Margin Size” píxeles como descartadas.

Tabla 8.8: CU-8. Aplicar filtro margen.

- **CU-9.** Aplicar filtro contraste (tabla 8.9).
  - Actores: Usuario.
  - Tipo: Primario, esencial.
  - Referencias:
  - Precondición: El usuario ha de haber completado el CU-1 y el CU-5
  - Postcondición: Se descartan objetos estelares que no cumplen con un perfil de contraste dado.
  - Autor: José Miguel López.

- Versión: 1.0.
- Propósito: Eliminar objetos que no cumplan un perfil de contraste dado, descartando así falsos positivos en la detección inicial.
- Resumen:

Curso normal			
	Actor	Sistema	
1	Usuario: Marca la casilla “Contrast Filter” y establece un valor para “Radius” entre 1 y 20 y “Minimun Quotient” entre 1 y 10.		
		2	El sistema marca las estrellas que no responda al perfil de contraste, como rechazas (Cociente entre punto más luminosos del objeto y mínimo del entorno)

Tabla 8.9: CU-9.Aplicar filtro contraste.

- **CU-10.** Aplicar filtro distancia entre objetos (tabla 8.10).
  - Actores: Usuario.
  - Tipo: Primario, esencial.
  - Referencias:
  - Precondición: El usuario has de haber completado el CU-1 y el CU-5
  - Postcondición: Se descartan objetos que estén próximos los unos a los otros.
  - Autor: José Miguel López.
  - Versión: 1.0.
  - Propósito: Eliminar objetos que se encuentren muy próximos entre ellos.
  - Resumen:
- **CU-11.** Aplicar filtro FWHM (tabla 8.11).
  - Actores: Usuario.
  - Tipo: Primario, esencial.

Curso normal		
	Actor	Sistema
1	Usuario: Marca la casilla “Distance Filter” y establece un valor para “Distance“ entre 1 y 500	
		2 El sistema marca las estrellas próximas unas a otras a una distancia inferior a “Distance“ como rechazadas.

Tabla 8.10: CU-10.Aplicar filtro distancia.

- Referencias:
- Precondición: El usuario ha de haber completado el CU-1 y el CU-5.
- Postcondición: Se descartan objetos que no respondan a un perfil FWHM dado.
- Autor: José Miguel López.
- Versión: 1.0.
- Propósito: Eliminar objetos que no respondan a un perfil **FWHM**.
- Resumen:

Curso normal		
	Actor	Sistema
1	Usuario: Marca la casilla “FWHM Filter” y establece un valor para “Minimun value“ entre 1 y 40 y “Radius“ entre 1 y 100.	
		2 El sistema marca las estrellas que no respondan al perfil FWHM como rechazadas.

Tabla 8.11: CU-11.Aplicar filtro FWHM.

- **CU-12.** Aplicar número de objetos (tabla 8.12).
  - Actores: Usuario.
  - Tipo: Primario, esencial.

- Referencias:
- Precondición: El usuario ha de haber completado el CU-1 y el CU-5
- Postcondición: Se descargan todos los objetos menos los más luminosos.
- Autor: José Miguel López.
- Versión: 1.0.
- Propósito: Descartar los objetos menos luminosos de la imagen.
- Resumen:

Curso normal		
	Actor	Sistema
1	Usuario: Marca la casilla “Number Filter” y establece un valor para “Number Star”.	
		2 El sistema marca las X “Number Star” estrellas como válidas y descarta el resto.

Tabla 8.12: CU-12. Aplicar número de objetos.

■ **CU-13.** Aplicar filtro ajuste Gauss (tabla 8.13).

- Actores: Usuario.
- Tipo: Primario, esencial.
- Referencias:
- Precondición: El usuario ha de haber completado el CU-1 y el CU-5
- Postcondición: Se descartan los objetos cuya curva de luz no responda a una curva de Gauss con los parámetros dados.
- Autor: José Miguel López.
- Versión: 1.0.
- Propósito: Descartar objetos que no respondan a una curva de Gauss dada por parámetros.
- Resumen:

Curso normal		
	Actor	Sistema
1	Usuario: Marca la casilla “Gaussian Filter” y establece un valor para “Radius“, “Normal“, “Mean“ y “Sigma“.	
		2 El sistema marca las estrellas que no respondan cuyo curva de luz no responda a una curva de Gauss

Tabla 8.13: CU-13. Aplicar filtro ajuste Gauss..

### 8.3.4. Diagrama de casos de uso

Los diagramas de casos de uso sirven para representar las relaciones que existen entre los diferentes actores y el sistema. Dado que en nuestro caso solo hay un actor, todos los casos de uso son iniciados por él (figura 8.3).

## 8.4. Diseño e implementación

El diseño de la aplicación puede dividirse en las siguientes partes:

- **Diseño de clases.**
- **Diseño de algoritmos de detección.**
- **Diseño de interfaz de usuario.**

### 8.4.1. Diseño de clases

El diseño de las clases define la arquitectura de la aplicación. Llevar a cabo una buena arquitectura determina la calidad, facilidad ante cambios o ampliaciones futuras, así como la propia implementación, testeo y corrección de posibles errores. En la figura 8.4 se muestra un pequeño diagrama de clases de la aplicación.

Las clases principales con las que cuenta nuestra aplicación son las siguientes:

- **FitsImage:** Realiza una abstracción de las propiedades y operaciones con las imágenes.

Las imágenes se encuentran en formato FITS. Para poder trabajar con ellas desde Java hacemos uso de la biblioteca **jfits** [47]. Con ella

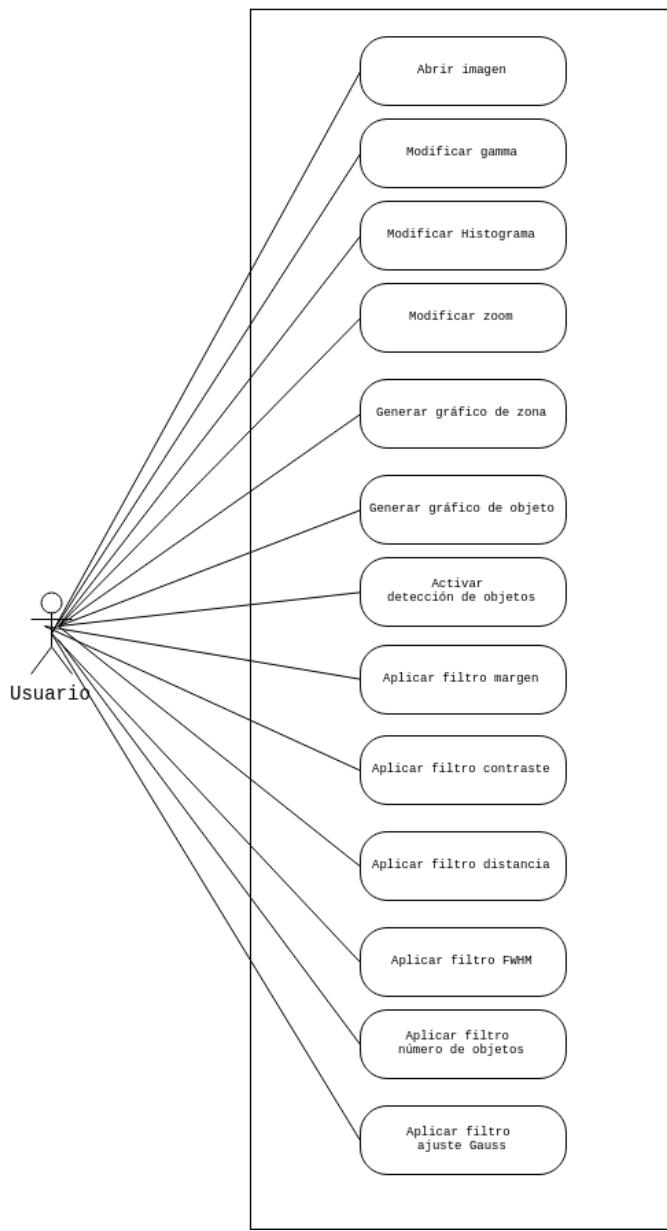


Figura 8.3: Diagrama de casos de uso.

tenemos acceso a los datos (píxeles de la matriz), y a los metadatos que incorporan la cabecera.

Para facilitar la programación se implementa una clase que encapsula los atributos y métodos para manejar imágenes FITS: `FitsImage`.

- **Star**: Realiza una abstracción de las propiedades y operaciones con los

## Diagrama Clases procesamiento Imágenes

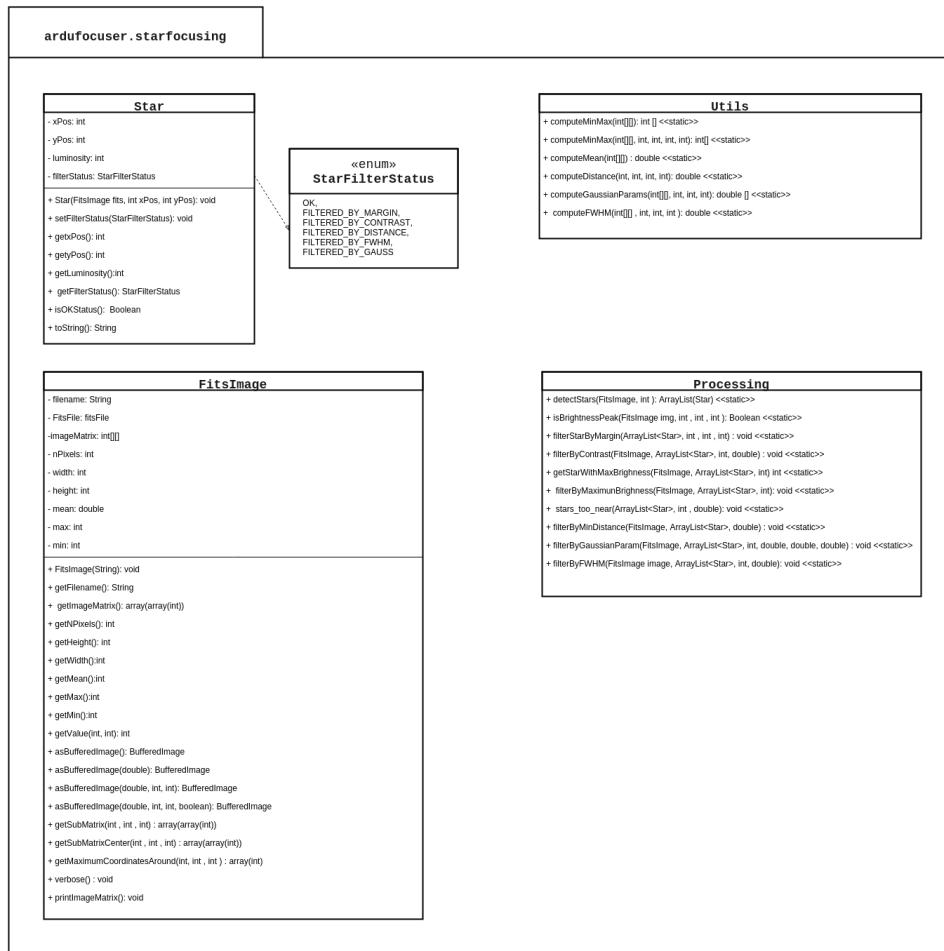


Figura 8.4: Diagrama de clases

objetos estelares (estrellas).

- **Processing:** Encapsula funciones y operaciones avanzadas de procesamiento y búsqueda de objetos.
- **Utils:** Operaciones de uso común en la aplicación, principalmente manejo de matrices y matemáticas.

#### 8.4.2. Diseño de interfaz de usuario

En la figura 8.5 se muestra un **mockup** aproximado donde se puede ver como ser organizan visualmente los elementos dentro de la interfaz de la aplicación.

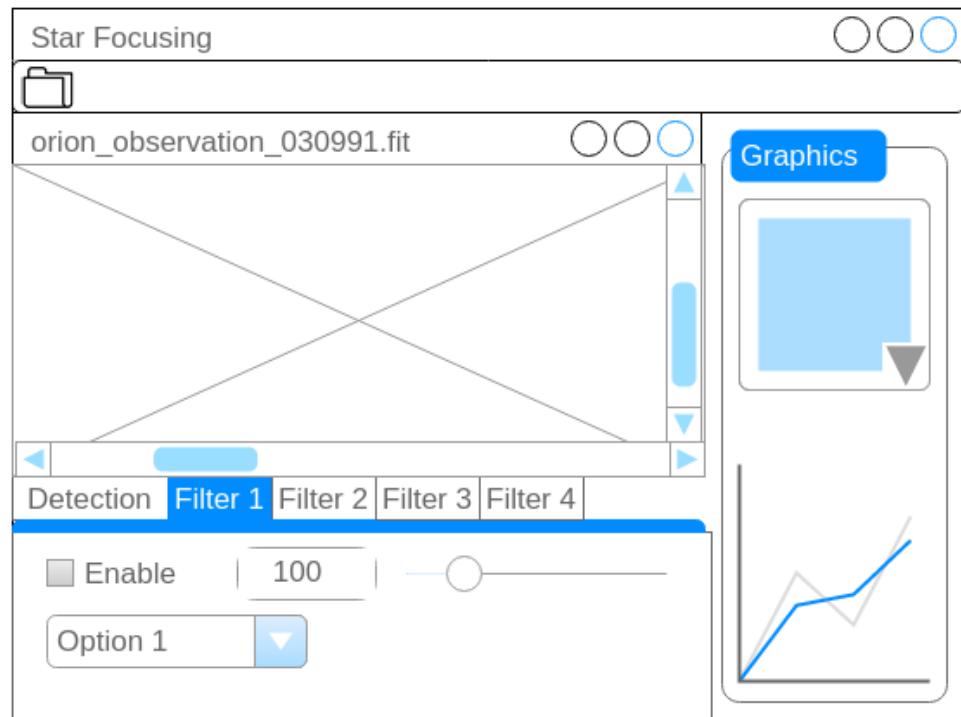


Figura 8.5: Mockup interfaz

En la figura 8.6 se muestra el aspecto de la aplicación terminada.

#### 8.5. Algoritmos de detección

Las imágenes que analizamos cuentan con gran cantidad de objetos estelares, que debemos ser capaces de detectar y filtrar según los patrones que nos

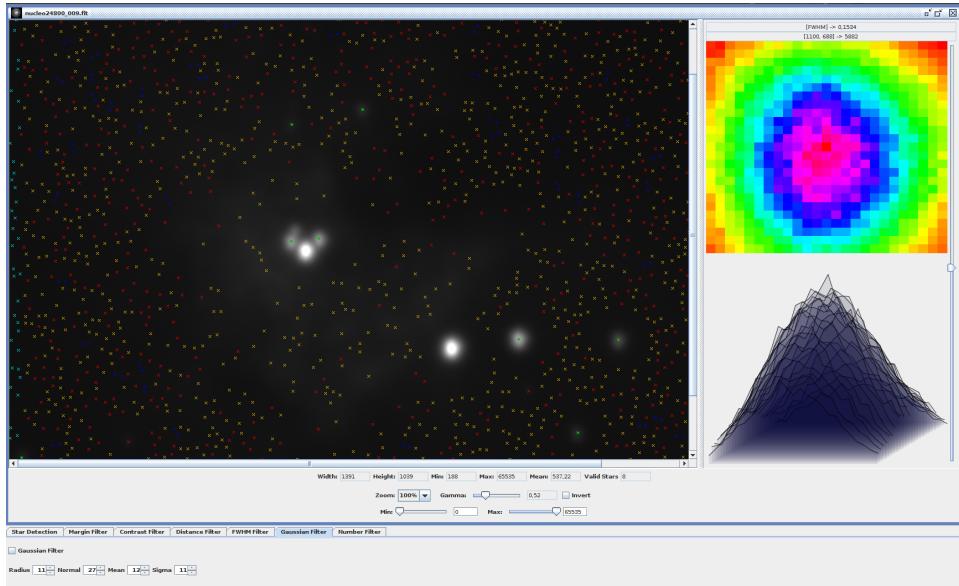


Figura 8.6: Aplicación Star Focusing

interesen. El objetivo de aplicar detección de objetos es poder detectar la posición de los objetos e la imagen que queremos enfocar: en este caso, estrellas (figura 8.7).

Para la detección de objetos utilizamos las siguientes rutinas:

1. Algoritmo búsqueda máximo local absolutos, tomando en cuenta puntos máximos en la imagen. Detecta falsos positivos, así como píxeles calientes aislados.
2. Región de puntos con tendencia descendente. Consiste en detectar regiones donde exista una sucesión de radio  $x$ , cuyo valor siga una tendencia clara, esta solución es bastante mejor aproximación que la anterior y elimina el problema de los píxeles calientes. No es óptimo pero se puede usar como un filtro de grano grueso.

Para considerar que un objeto es una estrella es condición necesaria, pero no suficiente que cumpla lo anterior.

Para asegurarnos de que realmente hemos encontrado un objeto estelar debemos aplicar una serie de filtros adicionales al resultado anterior y eliminar progresivamente falsos positivos, así como casos poco interesantes.

1. Filtro por perfil de contraste, resultado de dividir los valores altos y valores bajos, tiene un cociente alto. Esto indica que es esa región de la imagen existe una perturbación importante de luz. Dado que con el algoritmo anterior ya tenemos un conjunto de candidatos, se puede

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	3	5	5	7	7	7	6	6	7	5	5	5	4	3	2
2	5	7	8	9	9	9	9	9	9	7	7	6	6	4	3
3	7	9	11	13	13	13	14	14	13	14	11	9	7	6	5
4	9	12	14	19	20	20	21	18	18	18	16	14	11	9	6
5	11	15	22	25	27	30	28	28	29	26	23	19	15	11	9
6	13	19	24	32	36	40	40	39	36	33	28	25	20	16	12
7	18	23	31	40	46	50	54	56	54	48	42	35	25	17	13
8	20	25	37	45	56	64	71	69	65	60	49	41	31	22	15
9	22	35	43	51	61	68	78	79	80	69	54	45	33	26	18
10	24	36	44	57	74	80	87	87	86	79	62	51	35	26	18
11	24	33	46	66	77	87	96	100	90	86	72	50	36	25	18
12	25	33	47	64	80	85	94	93	88	83	69	52	34	23	16
13	21	28	37	53	66	76	78	83	74	66	59	42	31	21	14
14	18	25	33	40	52	62	65	67	68	55	48	38	26	20	13
15	16	19	25	32	41	49	52	56	50	45	37	31	21	16	11
16	12	17	22	26	29	34	42	39	35	31	28	23	18	13	8
17	10	13	16	19	21	24	25	27	25	22	19	16	13	10	8
18	8	10	12	15	16	18	20	20	18	16	14	13	9	7	6
19	6	7	9	10	11	14	14	14	11	11	9	8	7	5	4
20	4	5	6	7	8	9	9	9	8	7	7	6	5	4	3
21	2	2	3	4	4	6	6	6	5	6	6	5	4	4	3

Figura 8.7: Mapa de calor de una estrella

hacer de forma eficiente solo sobre los puntos previamente encontrados.

2. Filtro gaussiano y FWHM. Consisten en buscar regiones de la imagen donde se de un ajuste de la gaussiana dentro de cierto parámetros, la normal, el sigma o directamente calcular el FWHM. Este proceso es costoso y pesado, por tanto se debe aplicar a un conjunto muy pequeño de regiones, que previamente hayan pasado los filtros anteriores.

El perfil de cada estrella se puede aproximar con una curva Gaussiana, asumiendo que cuanto mejor el ajuste a tal curva, mejor es el enfoque obtenido en la imagen.

El perfil de dicha gaussiana se caracteriza por su ancho en la mitad de su valor máximo o **FWHM** (Full Width at Half Maximum [37]), este valor solo se ve afectado por el seeing [83], o la distorsión de la atmósfera, siendo constante para todos los objetos estelares en una misma imagen. FWHM, es una medida del efecto provocado por la dispersión del haz de luz que provoca que las estrellas no aparezcan como puntos sino como discos.

3. Filtros adicionales. Podemos eliminar objetos cuyo máximo este por encima o por debajo de cierto valor. Estrellas muy cercanas entre sí o estrellas próximas a los límites de la imagen.

Todas estas rutinas de filtrado quedan encapsuladas en la clase **Processing**.

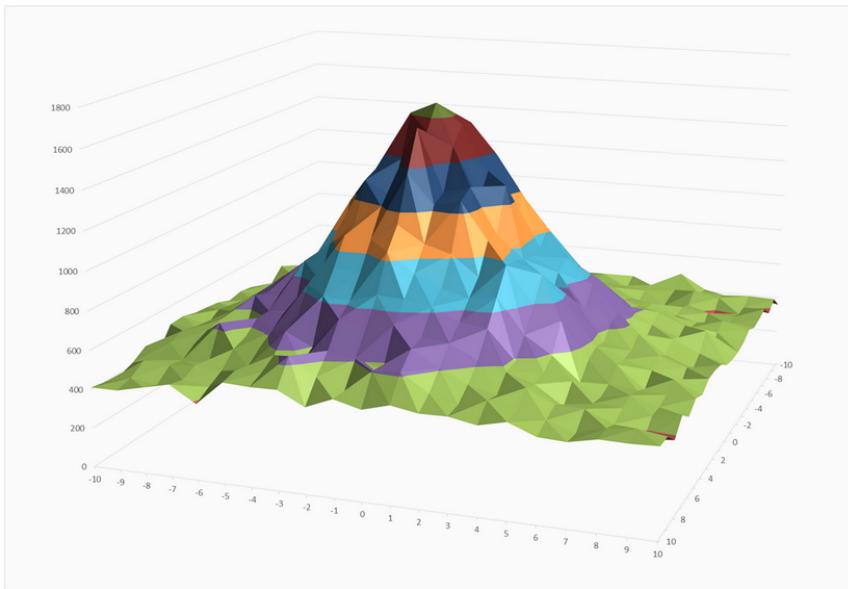


Figura 8.8: Ajuste Gaussiana de una Estrella

## 8.6. Evaluación nivel de enfoque

Una vez detectadas las estrellas, debemos determinar una medida para evaluar la calidad del enfoque.

El perfil en la imagen de cada estrella se puede aproximar con una curva Gaussiana 2D (figura 8.8).

Una característica de la curva de Gauss que nos da información de la calidad del enfoque de un objeto es la medida FWHM (Full width at half maximum, ancho total a mital del máximo [figura 8.9]).

Cuando la función considerada es una distribución normal de la forma:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{(x-x_0)^2}{2\sigma^2}\right]$$

Donde  $\sigma$  es la desviación típica y  $x_0$  puede ser cualquier valor (la anchura de la función no cambia con una traslación). La relación entre FWHM y la desviación típica es [39]:

$$\text{FWHM} = 2\sqrt{2 \ln 2} \sigma \approx 2.35482 \sigma$$

El FWHM puede medirse en diferentes ejes (en el eje X y en el eje Y por ejemplo). Si las estrellas son redondas, como comentábamos en el apartado

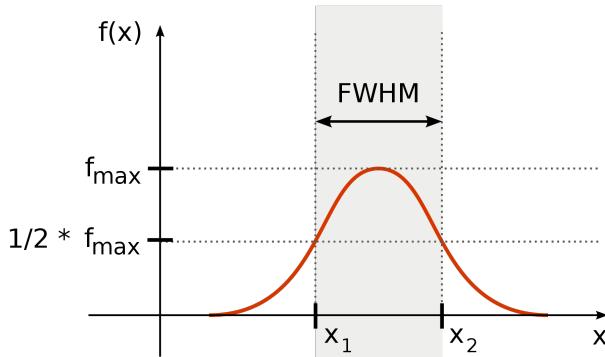


Figura 8.9: Gráfica FWHM. **Fuente:** [37]

anterior, el FWHM debe ser el mismo en cualquier eje. La diferencia entre los ejes X e Y puede utilizarse como medida de la redondez de la estrella.

#### Pasos para el cálculo del FWHM:

- **Ajustar perfil de luz a una Gausiana.**

Se implementa el método `computeGaussianParams`

Que recibe como parámetros:

- Matriz de pixeles de la imagen.
- Coordenadas punto central del objeto.
- Radio del objeto o zona.

Haciendo uso de la clase `GaussianCurveFitter` [2] que incorpora el paquete matemático Apache Commons Math [59] conseguimos hacer el ajuste en a dicha función.

- Extraer parámetro  $\sigma$  La función `computeGaussianParams`, nos devuelve directamente los parámetros de función (normal, media y sigma).
- Aplicar fórmula FWHM, siguiendo la formula descrita en la definición (fragmento de código 8.1).

---

```

29
30 public static double computeFWHM( int [][] image , int
31     starCenterX , int starCenterY , int radius ) {
32
33     double [] gaussparam ;
34     double sigma ;
35     double fwhm ;

```

```

36     gaussparam=computeGaussianParams(image, starCenterX,
37                                         starCenterY, radius);
38     double gfactor=2.0*Math.sqrt(2*Math.log(2));
39     sigma=gaussparam[2];
40     fwhm=gfactor*sigma;
41
42     return fwhm;
43 }
```

Fragmento de Código 8.1: Cálculo FWHM

## 8.7. Enfoque

Una vez contamos con las rutinas para detectar objetos y evaluar la calidad del enfoque de una imagen pasamos a definir posibles algoritmo de autofocus.

Estos algoritmo deben coordinar la obtención de imágenes con la manipulación del ajuste del enfocador.

Para manipular la configuración del enfoque ya contamos con el dispositivo que hemos desarrollado en los capítulos anteriores: **Ardufocuser**.

Además para la parte de control hemos implementado su correspondiente **Driver INDI**.

Para la adquisición de imágenes se ha usa el equipo **CCD ATIK-314L** [100] (figura 8.10) con las siguientes características:

- **Sensor:** CCD - Sony ICX-285AL
- **Resolución:** 1392 x 1040 pixels
- **Tamaño del Pixel:**  $6.45\mu m \times 6.45\mu m$
- **Interfaz:** USB 2.0
- **Máxima longitud de exposición:** (Ilimitada)
- **Mínima longitud de exposición:** (1/1000 s)
- **Refrigeración a  $-27^{\circ}C$**

La cámara cuenta con un driver INDI, para encargarse del control [22] de la misma.

### Acciones enfoque automático:

- **Tomar foto:** Solicitar imagen a CCD.



Figura 8.10: CCD ATIK-314L - Fuente: [19]

- **Evaluar:** Calcular FWHM imagen.
- **Solicitar nuevo foco:** Mover enfocador en una dirección, un determinado número de pasos.

Con las operaciones anteriores pasamos a definir una serie de algoritmos de autoenfoque cuyo funcionamiento general se muestra en la figura 8.11.

### 8.7.1. Algoritmo búsqueda a ciegas

Este algoritmo consiste en hacer una primera aproximación rápida del enfoque, buscando la posición adecuada para que pueda funcionar los siguientes algoritmos. Es por ello que se denomina “a ciegas” puesto que el sistema no tiene ninguna información del estado anterior. La idea fundamental es que detecte cuando se empiezan a detectar estrellas (figura 8.12).

```

1 INICIO [EnfoqueACiegas]
2   Direccion = 0
3   NumeroPasos = 1
4   FocuserPosition = 0
5   FocuserPosition = moverEnfocadorHastaTope(Direccion)
6   Direccion = 1
7   Imagen = PedirImagenCCD()
8   existenEstrellas = DetectarEstrellas(Imagen)
9   WHILE !existenEstrellas :
```

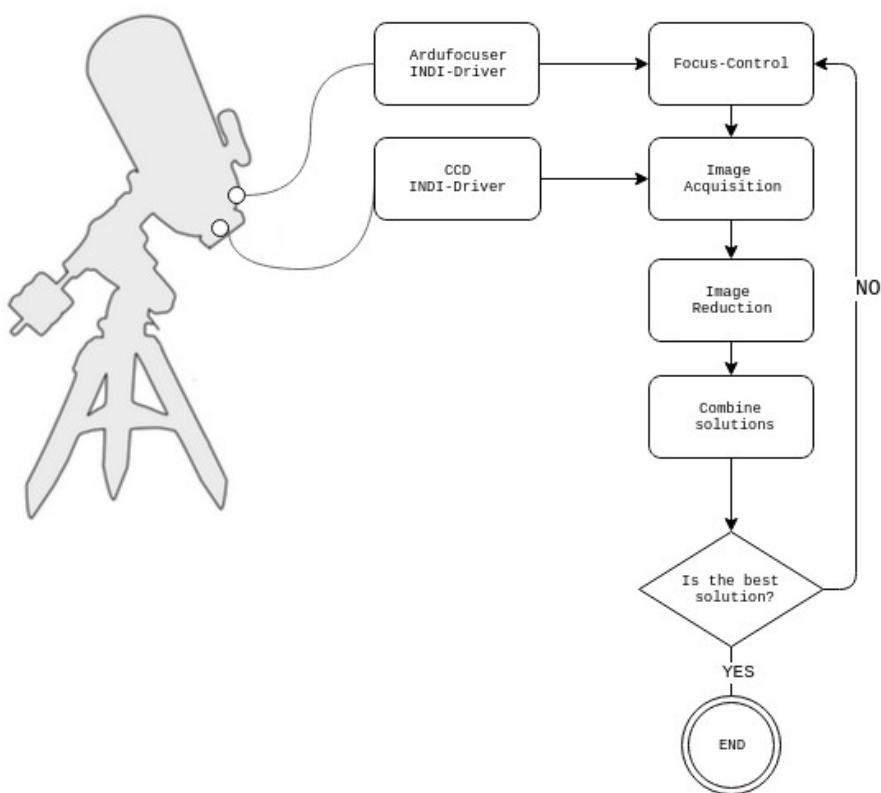


Figura 8.11: Método propuesto

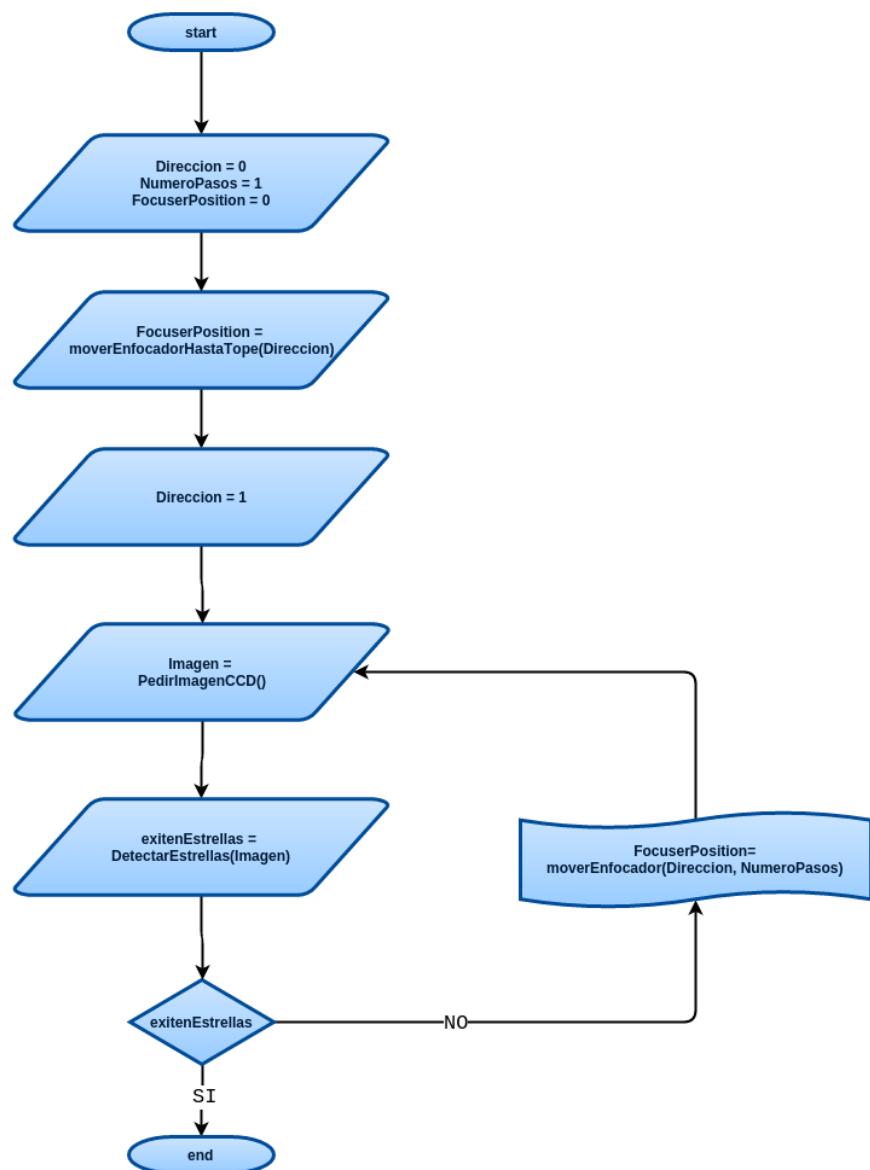


Figura 8.12: Algoritmo - A Ciegas

```

10     FocuserPosition=moverEnfocador(Direccion , NumeroPasos)
11     Imagen = PedirImagenCCD()
12     existenEstrellas = DetectarEstrellas(Imagen)
13
14 FIN : return FocuserPosition

```

### 8.7.2. Algoritmo autofocus básico

Con este algoritmo conseguimos llegar al máximo de enfoque (figura 8.13).

```

1 INICIO [EnfoqueBase]
2   Direccion = 1
3   NumeroPasos = 1
4   FocuserPosition = 0
5   Imagen = PedirImagenCCD()
6   fwhm2 = CalcularFwhm(Imagen)
7   fwhm1 = 99
8
9   WHILE fwhm2 < fwhm1 :
10     fwhm1 = fwhm2
11     FocuserPosition=moverEnfocador(Direccion , NumeroPasos)
12     Imagen = PedirImagenCCD()
13     fwhm2 = CalcularFwhm(Imagen)
14
15 FIN : return FocuserPosition

```

### 8.7.3. Algoritmo autofocus para evitar problemas de seeing

Con este algoritmo conseguimos llegar al máximo de enfoque evitando de manera más eficiente las perturbaciones atmosféricas (figura 8.14).

```

1 DEFINE [TomaFotosYCalcularMediaFWHM()]
2   FOR i=0;i<10; i++ :
3     Imagen = PedirImagenCCD()
4     SumFwhm += CalcularFwhm(Imagen)
5   MeanFwhm = SumFwhm / 10
6 FIN : return MeanFwhm
7
8 INICIO [EnfoqueSeeing]
9   Direccion = 1
10  NumeroPasos = 1
11  FocuserPosition = 0
12  fwhm1 = TomaFotosYCalcularMediaFWHM()
13  fwhm2 = 99
14
15  WHILE fwhm2 < fwhm1 :
16    fwhm2 = fwhm1
17    FocuserPosition=moverEnfocador(Direccion , NumeroPasos)
18    fwhm2 = TomaFotosYCalcularMediaFWHM()
19
20 FIN : return FocuserPosition

```

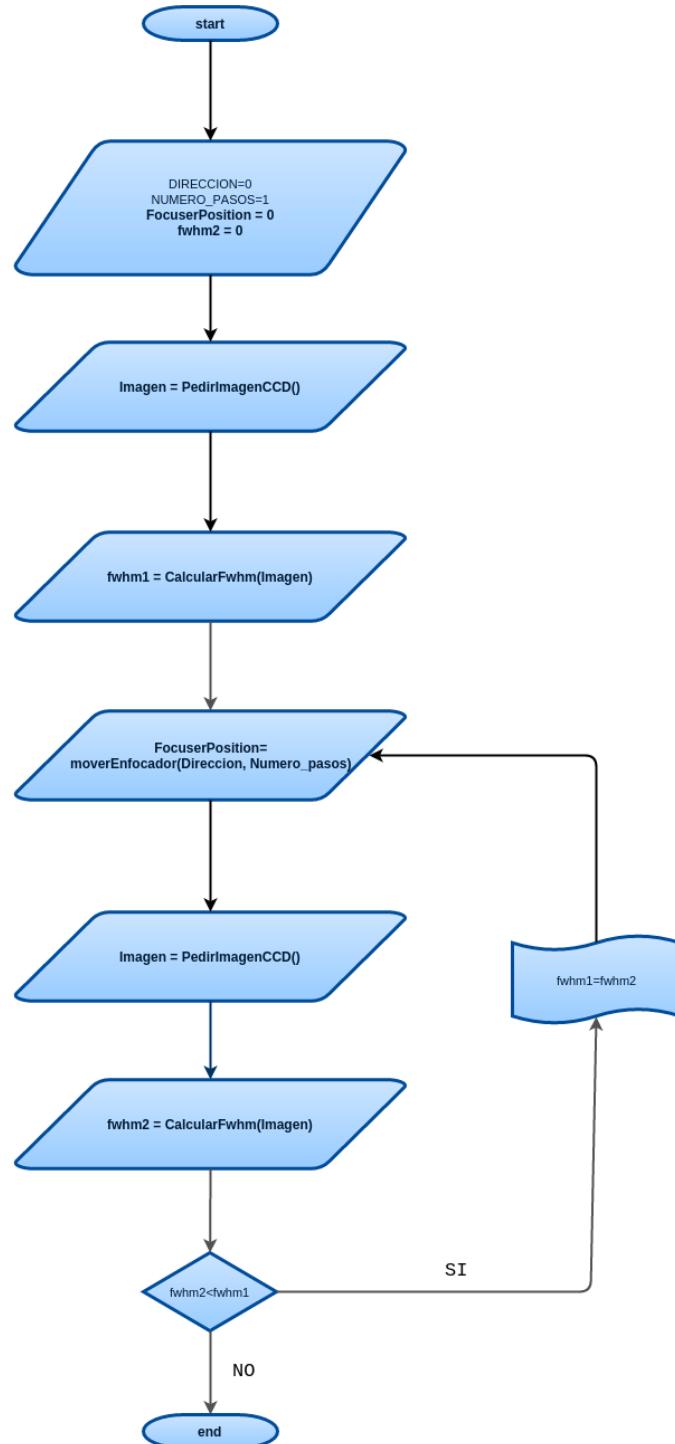


Figura 8.13: Algoritmo de autofocus básico

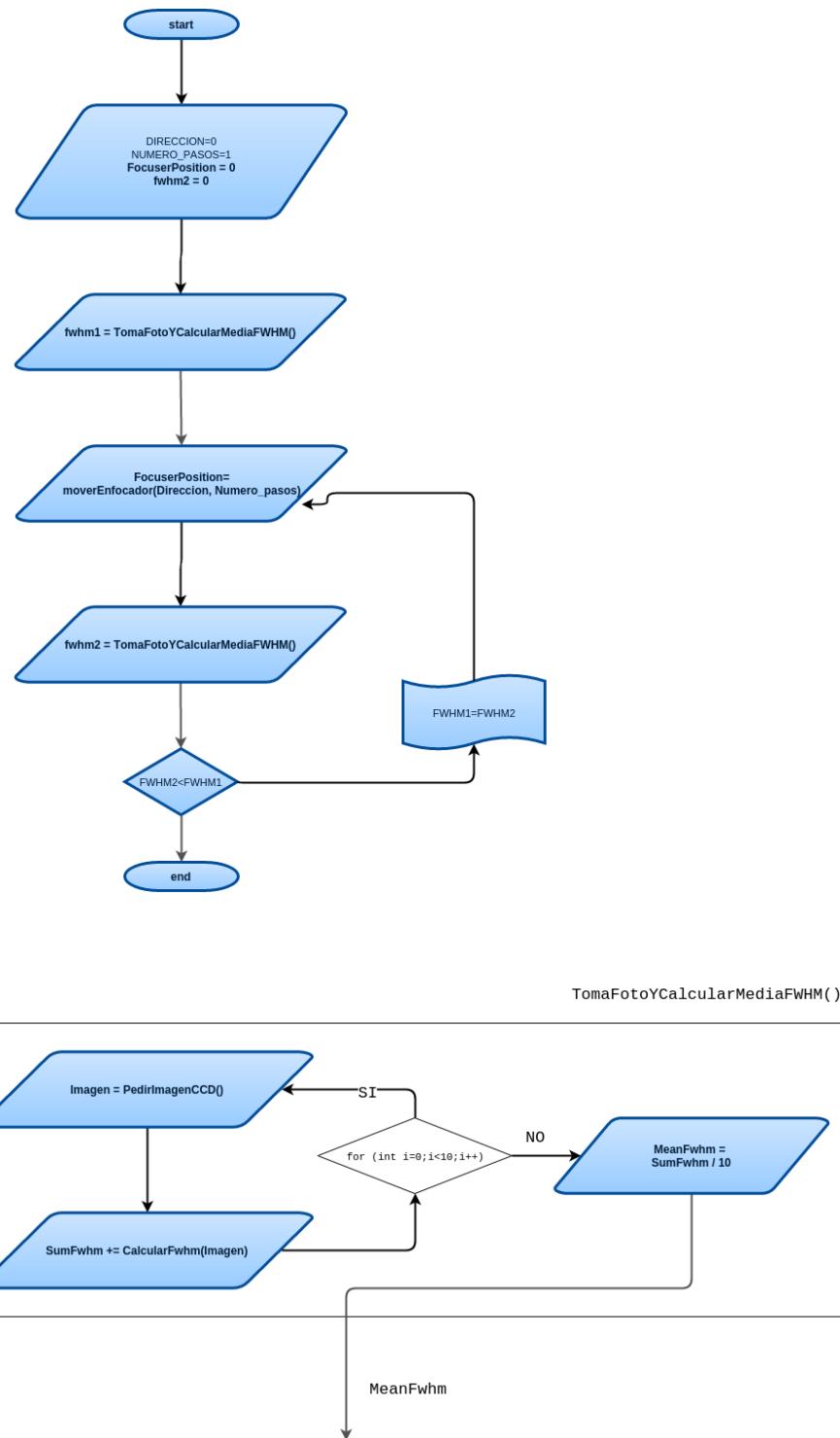


Figura 8.14: Algoritmo de autofocus para evitar problemas de seeing

### 8.7.4. Algoritmo autofocus para evitar problemas de backlash

Con este algoritmo conseguimos llegar al máximo de enfoque evitando problemas de backlash (figura 8.15).

```

1 DEFINE [TomaFotosYCalcularMediaFWHM()]
2   FOR i=0;i<10;i++ :
3     Imagen = PedirImagenCCD()
4     SumFwhm += CalcularFwhm(Imagen)
5     MeanFwhm = SumFwhm / 10
6   FIN : return MeanFwhm
7
8 INICIO [EnfoqueBacklash]
9   Direccion = 1
10  NumeroPasos = 1
11  FocuserPosition = 0
12  fwhm1 = TomaFotosYCalcularMediaFWHM()
13  fwhm2 = 99
14
15  WHILE fwhm2 < fwhm1 :
16    fwhm2 = fwhm1
17    FocuserPosition=moverEnfocador(Direccion, NumeroPasos)
18    fwhm2 = TomaFotosYCalcularMediaFWHM()
19
20    BestFocusPosition=FocuserPosition
21
22    FocuserPosition = moverEnfocador(FocuserPosition-10)
23
24    FocuserPosition = moverEnfocador(BestFocusPosition)
25
26  FIN : return FocuserPosition

```

## 8.8. Métodos de simulación

Aquí se muestra una breve introducción al simulador que se ha diseñado para poder realizar pruebas de enfoque sin tener que usar instrumental en directo ya que realizar la implementación y pruebas en vivo puede llegar a ser tedioso, dado que requiere unas condiciones de observación muy específicas.

Es por ello que se opta por implementar un simulador de CCD y Enfocador.

Este software cuenta con las mismas funciones que el sistema real (tomar foto, avanza enfocador, retrocede), pero se ejecuta en un entorno virtual.

Además para asegurar que es fiel a la realidad, su estado inicial debe ser aleatorio.

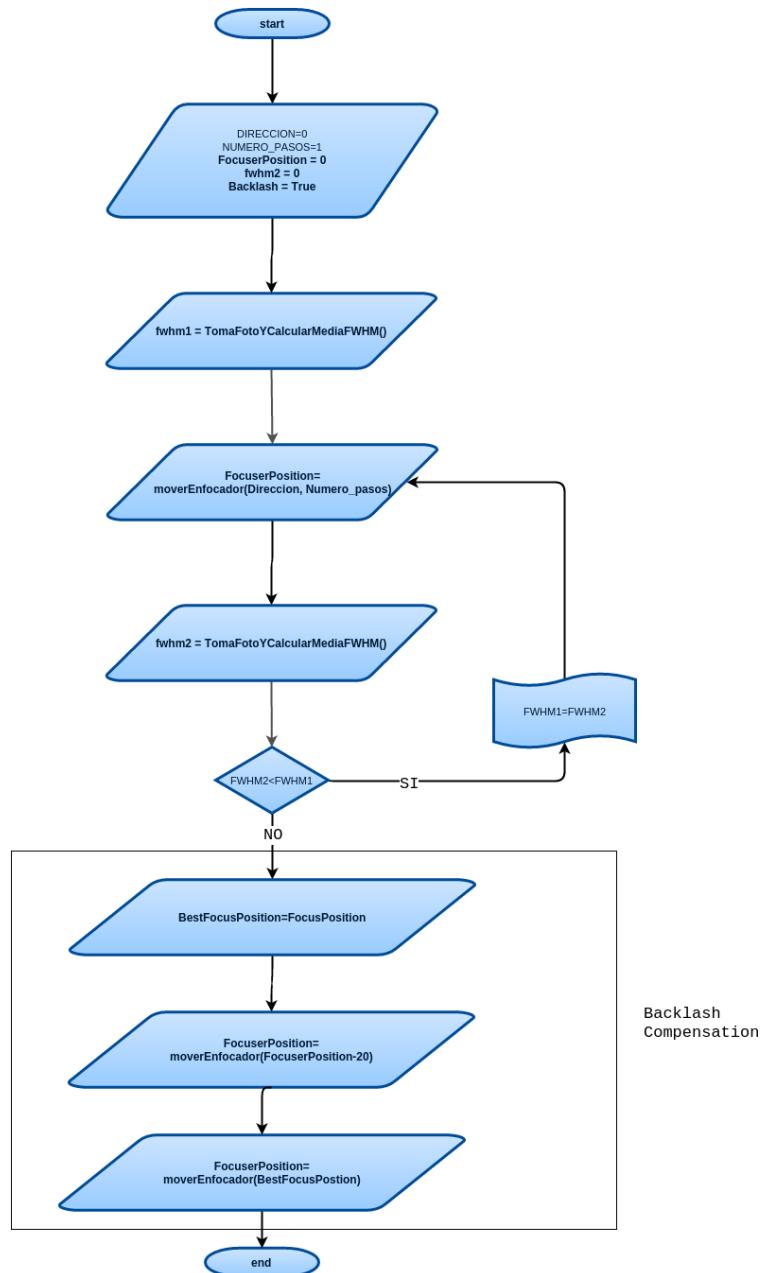


Figura 8.15: Algoritmo de autofocus para evitar problemas de backlash



## Capítulo 9

# Documentación y Difusión

Para favorecer el interés del proyecto entre los aficionados a la astronomía (y al hardware libre en general) se han emprendido diversas actividades divulgativas, como son:

- Creación de una página web del proyecto.
- Creación de canales en redes sociales.
- Asistencia y presentación del proyecto en congresos astroómicos.
- Mantenimiento del repositorio **GitHub** con toda la información relevante del proyecto (documentación, código, diagramas...)

En las siguientes secciones se comenta más en detalle cada una de dichas actividades.

### 9.1. Página web del proyecto

Se diseña una página web a modo de “**landing page**” que resuma toda la información del proyecto de una forma agradable al visitante. Las características más reseñables de dicha página web son:

- Mostrar toda la información del proyecto bien catalogada y organizada en las diferentes secciones. Se incluyen fotos y resúmenes de algunos pasos del proyecto, lista de materiales y enlaces de descarga.
- Seguir diseño visual atractivo y favorecer una buena experiencia de usuario, cuidando el esquema de colores, la disposición de los elementos y su tamaño, permitiendo la navegación por el sitio de una forma cómoda e intuitiva.

- Diseño *responsive*, asegurando que se ven todos los elementos de forma correcta en ordenadores, móviles y tabletas.
- Seguir los últimos estándares HTML5, validando una correcta semántica, utilizar **metadescripciones** [61]. Con ello favorecemos una correcta indexación por los motores de búsqueda, haciendo llegar la página al público objetivo de una forma más rápida y selectiva (uno de los principios básicos de SEO [74]).

Dicha página web puede verse en <http://ardufocuser.info>.

## 9.2. Canales en redes sociales

Al comienzo del desarrollo se creó una cuenta en twitter **@ardufocusindi** con la finalidad de crear una comunidad de gente interesada en el proyecto y en otros temas relacionados.

La idea de este canal era mantener informada a la comunidad de los últimos avances del proyecto y nuevas actualizaciones así como compartir también material interesante para estar al día de los últimos avances.

Sin embargo, hay que reconocer que este intento de difusión no ha funcionado bien, probablemente por la falta de actualización así como la dificultad de publicitar algo tan específico que prácticamente solo interesa a un público objetivo muy concreto (astrónomos) sin ser el propio desarrollador un miembro de esta comunidad.

## 9.3. Congresos Astronómicos

Durante el proyecto he tenido la oportunidad de asistir a eventos y congresos astronómicos donde se ha promocionado el proyecto.

- **Astro-Encuentro La Sagra 2015:** Celebrado del 16 al 18 de Octubre en la localidad de la Puebla de Don Fabrique, dos días diversas actividades en el marco de la astronomía, donde tuve oportunidad de visitar el “Observatorio Astronómico de la Sagra” [31] operado, entre otros, por el Instituto de Astrofísica de Andalucía (IAA).

Entre todas las actividades, por su relación con el presente proyecto, cabe destacar la ponencia de D. Nicolás Morales (investigador del Instituto de Astrofísica de Andalucía) donde realiza una demostración de una observación con telescopios manejados en remoto en tiempo real demostrando como realizan estudios de impactos en la Luna.

- **Astro-Alcala Alcalá la Real 2016:** Celebrado del 8 al 10 de Abril en la localidad de Alcalá la Real. En dicho encuentro se presentó un póster mostrando un resumen del proyecto Ardufocuser (figura 9.1).
- Presentación del proyecto **Ardufocuser** ante la SAG (Sociedad Astronómica Granadina) [15] en Febrero de 2016: En dicha presentación se mencionan a grandes rasgos las ventajas que ofrece en comparación a los productos comerciales y se comparten detalles técnicos del mismo. Varios de los asistentes mostraron mucho interés en el proyecto.

## 9.4. Mantenimiento del repositorio GitHub

Para el desarrollo del proyecto se ha contado con varios repositorios en GitHub, por comodidad uno para cada parte del proyecto.

- **Ardufocuser-firmware:** Contiene todo el código fuente del firmware que ejecuta la placa de Arduino, junto con las bibliotecas correspondientes y los scripts de pruebas.
- **Ardufocuser-indidriver:** Contiene el módulo driver INDI específico para Ardufocuser, implementado en Java a partir de las especificaciones **INDIforJava**.
- **Ardufocuser-image-processing:** Contiene el código fuente del software de análisis de imágenes, detección de objetos y cálculo de FWHM. Además cuenta con la biblioteca de manejo de imágenes FITS implementada para facilitar el trabajo.
- **Ardufocuser-documentations:** Contiene los fuentes **LATeX** de este mismo documento junto con las imágenes y recursos utilizados.

Utilizar GitHub me ha proporcionado grandes ventajas.

- Contar con un sistema de control de versiones distribuido, con toda la potencia de **Git**.
- Publicar los cambios directamente y ser accesibles desde la web (se puede consultar desde un navegador web).
- Conectar servicio de integración continua Travis CI.

En todo momento se ha intentado cuidar el estilo y la calidad del código, añadiendo los pertinentes comentarios y utilizando nombres adecuados para nombrar variables, clases, métodos o funciones con el propósito que cualquier interesado pueda acceder al código, comprender la lógica y realizar cambios de forma fácil.

Cualquier interesado en colaborar queda invitado a realizar **Merge Request**, indicando un resumen de la mejora o el bug que soluciona. En lo más breve se evaluará el cambio, se incorporará a la base de código y aparecerá como colaborador del proyecto.



Figura 9.1: Poster presentando el proyecto en AstroAlcalá 2016.



## Capítulo 10

# Conclusiones y trabajos futuros

Lo primero que me gustaría resaltar es que creo que el resultado de este trabajo ha sido muy satisfactorio tanto por el resultado como por el proceso de desarrollo. Con el desarrollo del proyecto he tenido oportunidad de familiarizarme con multitud de tecnologías y disciplinas para solucionar problemas de diferente naturaleza.

He conseguido implementar una solución de enfoque perfectamente válida, que cumple con gran parte de los objetivos propuestos (no solo los obligatorios, sino también algunos secundarios). No obstante es importante remarcar que **Ardufocuser** aun permanece en fase **beta**.

Otro de los grandes alicientes que ha tenido el presente TFG, ha sido poder participar en jornadas de observación, agradeciendo experiencia al grupo de personas que conforman **La Azotea** [94] ya que con ellos he aprendido concepto básicos de astronomía, así como hacer pruebas del sistema y comprender mejor algunos de los retos que suponen algo tan aparentemente sencillo como es enfocar un telescopio.

También he podido participar en varios congresos astronómicos **AstroAlcalá 2016** (Alcalá la Real) y **AstroEncuentro La Sagra 2015** (Puebla Don Fabrique), eventos muy enriquecedores donde he tenido posibilidad de ver el funcionamiento de observatorios profesionales, las soluciones de control que aplican y discutir con profesionales las posibles limitaciones, inconvenientes de los sistemas actuales y como se podrían hacer una implantación de sistemas de control basados en la tecnologías que hemos presentado en el proyecto, especialmente **INDIforJava** y **Hardware Libre**.

En el presente proyecto nos hemos centrado en el módulo de enfoque, por ser uno de los problemas más interesantes, pero con la base tecnológica

implementada podemos afrontar el control de muchos otros elementos de un observatorio con facilidad.

Es también reseñable y digno de destacar que es posible crear soluciones avanzadas con un presupuesto bajo y obteniendo unos resultados más que aceptables. Todo ello gracias a tecnologías libres como Arduino y Raspberry Pi, que además tienen una amplísima comunidad de personas detrás.

El proyecto ha servido para hacerme evolucionar como ingeniero en gran medida y son muchas las lecciones aprendidas:

- Dar la importancia necesaria a las reuniones con los clientes, aprendiendo a escuchar y dirigir la sesión para conseguir profundizar en la necesidad reales.
- Destacar la importancia de las fases de diseño y análisis, pues en base a ello se puede acortar el proceso de implementación de forma considerable, reduciendo el número de iteraciones.
- Aprender a ser más realista realizando estimaciones, especialmente cuando tenemos un alto índice de incertidumbre en muchos puntos. Este punto es de vital importancia de cara a proyectos profesionales donde los proyectos estén limitado en tiempo (existan *deadlines*) y limitados económicamente mediante presupuestos fijos.
- Tener una introducción a las metodológicas ágiles, en este caso la iterrativa basada en prototipos.
- Ser consciente de las capacidades adquiridas durante mi formación y como con el suficiente trabajo soy capaz de trabajar y aprender en muchas otras disciplinas relacionadas.
- Además, el hecho de tener que aprender y aplicar diversos conocimientos de áreas afines (como electrónica o programación a bajo nivel) me ha hecho percatarme de como ciertos desarrollos probablemente pueden hacerse mejor si el equipo de desarrollo no se limita a una única persona sino aarios miembros con experiencia y conocimientos en áreas distintas. Por ejemplo, el presente proyecto podría haberse abordado de manera más sencilla y probablemente más eficiente si en el equipo hubieramos contado con:
  - Un **Astrofísico** con conocimiento en los elementos del observatorio y problemas que pueden darse en las observaciones de los fenómenos.
  - Un **Ingeniero Electrónico** con conocimiento de electrónica y componentes a bajo nivel.

- Un **Ingeniero Informático** con conocimientos de ingeniería del software y procesamiento de imágenes.

También quiero reseñar que la aceptación del proyecto (aún en fases tempranas) ha sido grande y ya hay simpatizantes de diferente formación (astrónomos aficionados, ingenieros informáticos, mecánicos) interesados en participar en el proyecto, aportando también nuevas ideas y habilidades: Dado que es un proyecto abierto y con licencia libre puede participar cualquier interesado. Así mismo me consta que varias personas están esperando a la publicación del proyecto en su actual forma para directamente construir y utilizar Ardufocuser en sus observatorios particulares.

### 10.1. Trabajos futuros

Ardufocuser se encuentra en fase beta, lo que implica que es probable que queden detalles por afinar y ajustes que realizar. Además, existen algunas ideas que aún no se han completado o implementado o simplemente que quedan como posibles mejoras a probar. A continuación se enumeran algunas de estas ideas que podrán incorporarse o desarrollarse en el futuro:

- Reducir el tamaño (y coste) de la electrónica y simplificar las conexiones, utilizando un modelo de Arduino Mini o Micro.
- Incorporar el repertorio de instrucciones **Robofocus** al protocolo serie, permitiendo compatibilidad con cualquier software de enfoque compatible con Robofocus (ASCOM).

Por otra parte a nivel software tenemos los siguientes puntos a completar en el futuro:

- Implementar los algoritmos autofocus de estrellas que se han diseñado y realizar las pertinentes pruebas en un entorno real.
- Corrección automática de foco por compensación por cambio de temperatura.
- Implementar autofocus superficies planetarias, punto muy interesante y que aún no está resuelto de manera satisfactoria por las soluciones comerciales. Para ello se deben estudiar nuevas funciones de evaluación adecuada para superficies planetarias. Esto no es una tarea trivial pues implica conocimientos de procesamiento de imágenes relativamente avanzadas. Además las imágenes planetarias gozan de muy poco contraste (en muchas ocasiones ordenes de magnitud por debajo que fotografías “convencionales”) y sufren de distorsiones generadas por la atmósfera lo que no hace sencilla la adaptación de algoritmos

en enfoque “clásicos” como los que pueden incorporar las cámaras de fotos.

- Implementar test automáticos para las rutinas de evaluación y detección.
- Plantear INDI como un protocolo / plataforma que pueda ser utilizado para otros ámbitos distintos a la astronomía, como podrían ser la **domotización doméstica** o la **sensorización agrícola**.

De hecho, me consta que gracias al buen resultado obtenido en el proyecto pronto otras personas (tanto alumnos como aficionados de la astronomía) tienen pensado continuar con el diseño e implementación de otros módulos de control de instrumental astronómico como pueden ser ruedas portafiltros, cúpulas, techos corredizos, controladoras de relés, etc.

# Glosario

## A

**API.** Application Programming Interface  
**AWT.** Abstract Window Toolkit  
**ASCOM.** AStronomy Common Object Model

## C

**CSS.** Cascading Style Sheets  
**CU.** Caso de uso  
**CCD.** Charge Coupled Device

## D

**DIY.** Do It Yourself - Hágalo usted mismo

## E

**EEG.** Electroencefalografía  
**EDF.** European Data Format

## F

**FITS.** Flexible Image Transport System  
**FWHM.** Full Width at Half Maximum

**G**

**GPS.** Global Positioning System

**GSM.** Global System for Mobile

**GPRS.** General Packet Radio Service

**H**

**Hz.** Hertzios

**I**

**IDE.** Integrated Development Environment

**I2C.** Inter-Integrated Circuit

**INDI.** Instrument Neutral Distributed Interface

**L**

**LCD.** Liquid Crystal Display

**M**

**MVC.** Modelo-Vista-Controlador

**P**

**PC.** Personal Computer

**PWM.** Pulse Width Modulation - Modulación por ancho de pulsos.

**PCB.** Printed Circuit Board - Placa de circuito impreso.

**PHP.** Pre Hypertext - Processor.

**R**

**RF.** Requisito Funcional

**RNF.** Requisito no Funcional

**S**

**SDK.** Software Development Kit  
**SPI.** Serial Peripheral Interface  
**SIM.** Subscriber identity module

**T**

**THT.** Through Hole Technology

**U**

**UBS.** Universal Serial Bus

**X**

**XML.** eXtensible Markup Language



## Apéndice A

### Diagramas circuito

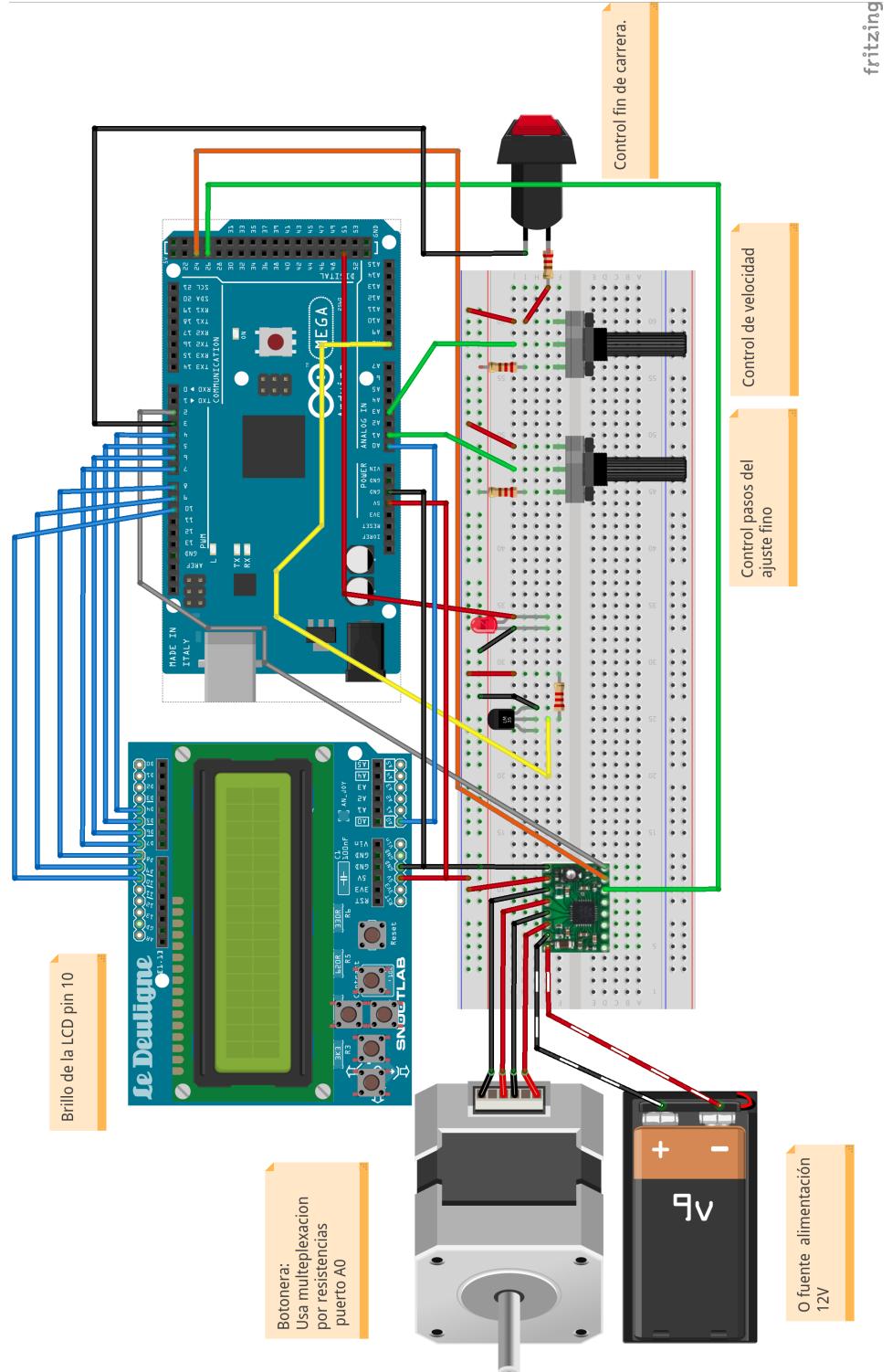


Figura A.1: Versión inicial del dispositivo, montado sobre una placa de prototipado, así como un módulo LCD con botonera y utilizando el controlador Arduino Mega.

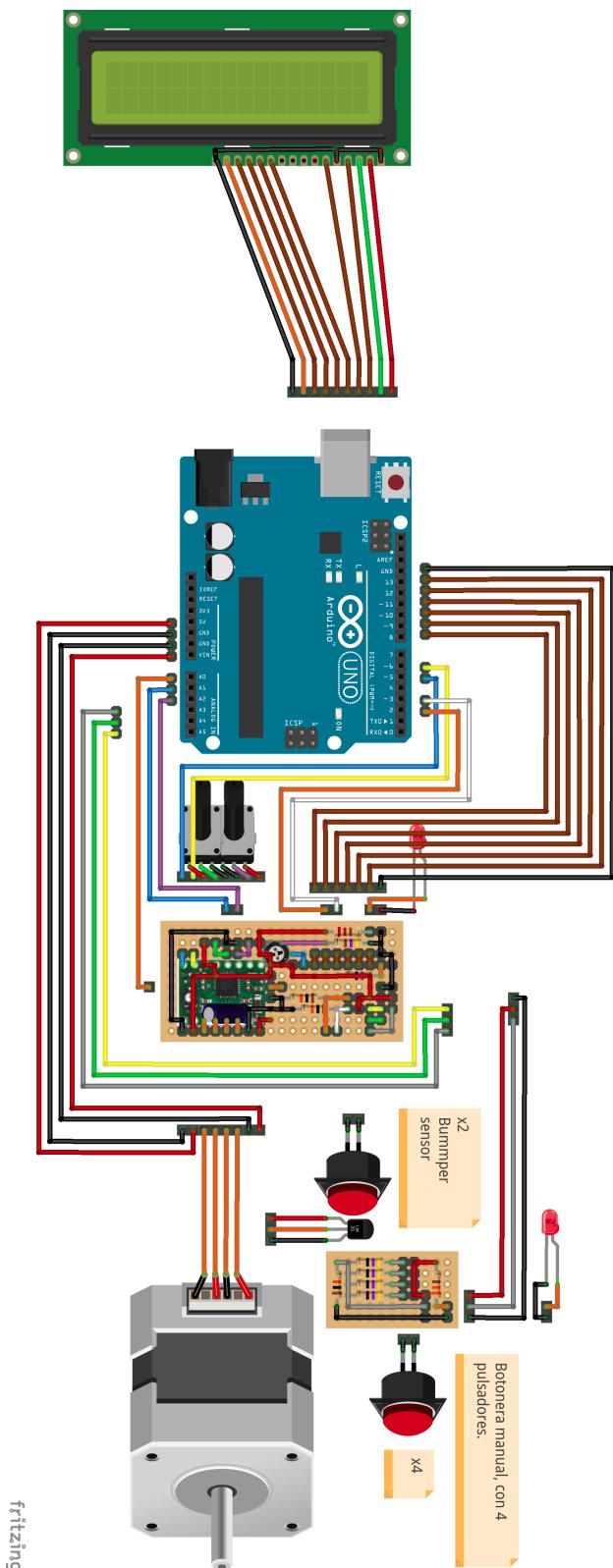


Figura A.2: Versión con pantalla LCD interfaz clásica, botonera analógica y Arduino UNO.

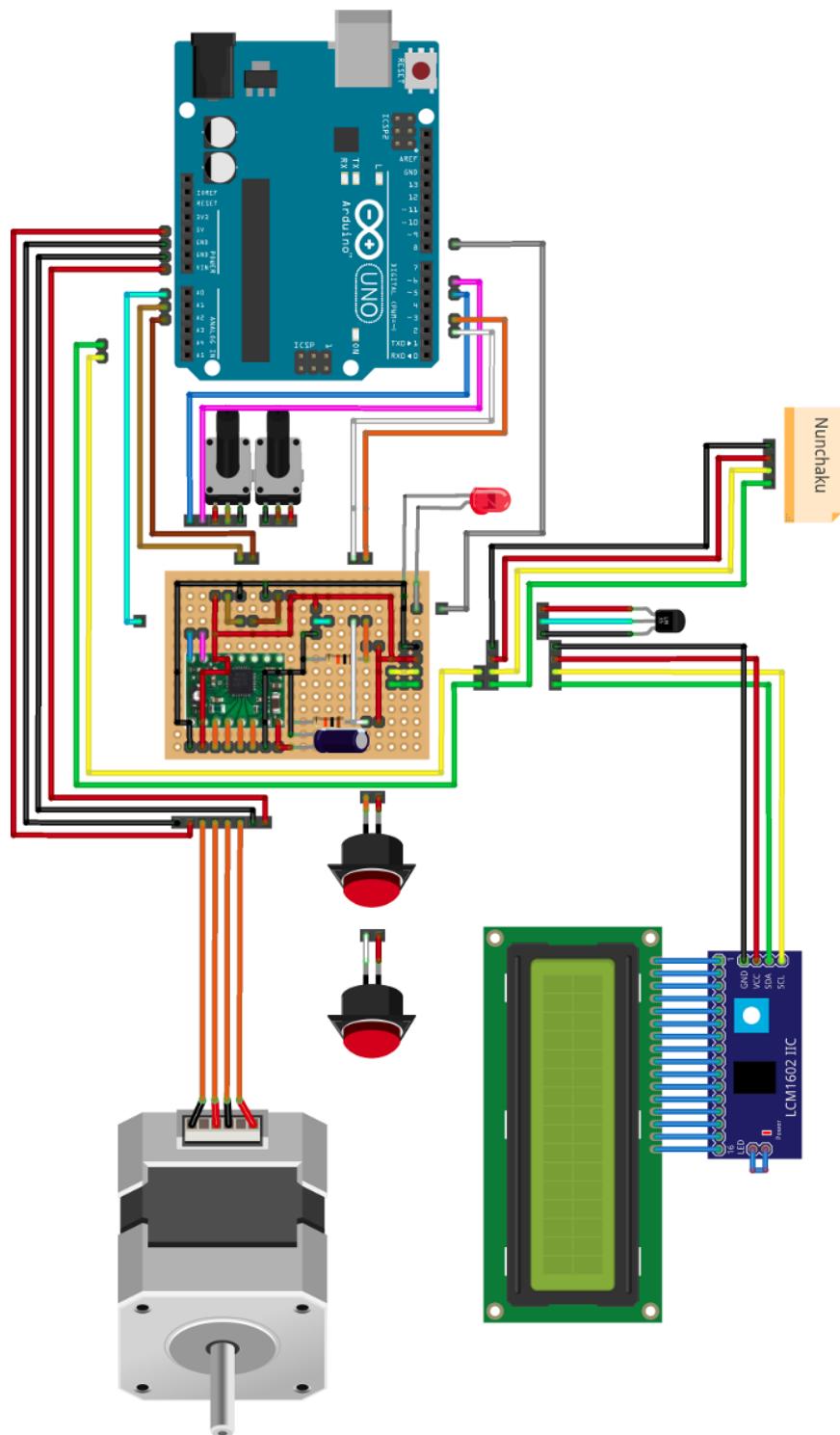


Figura A.3: Versión con pantalla LCD con interfaz I2C, mando Nunchuck, sensor de temperatura LM35 (interfaz analógico) y con controlador Arduino UNO.

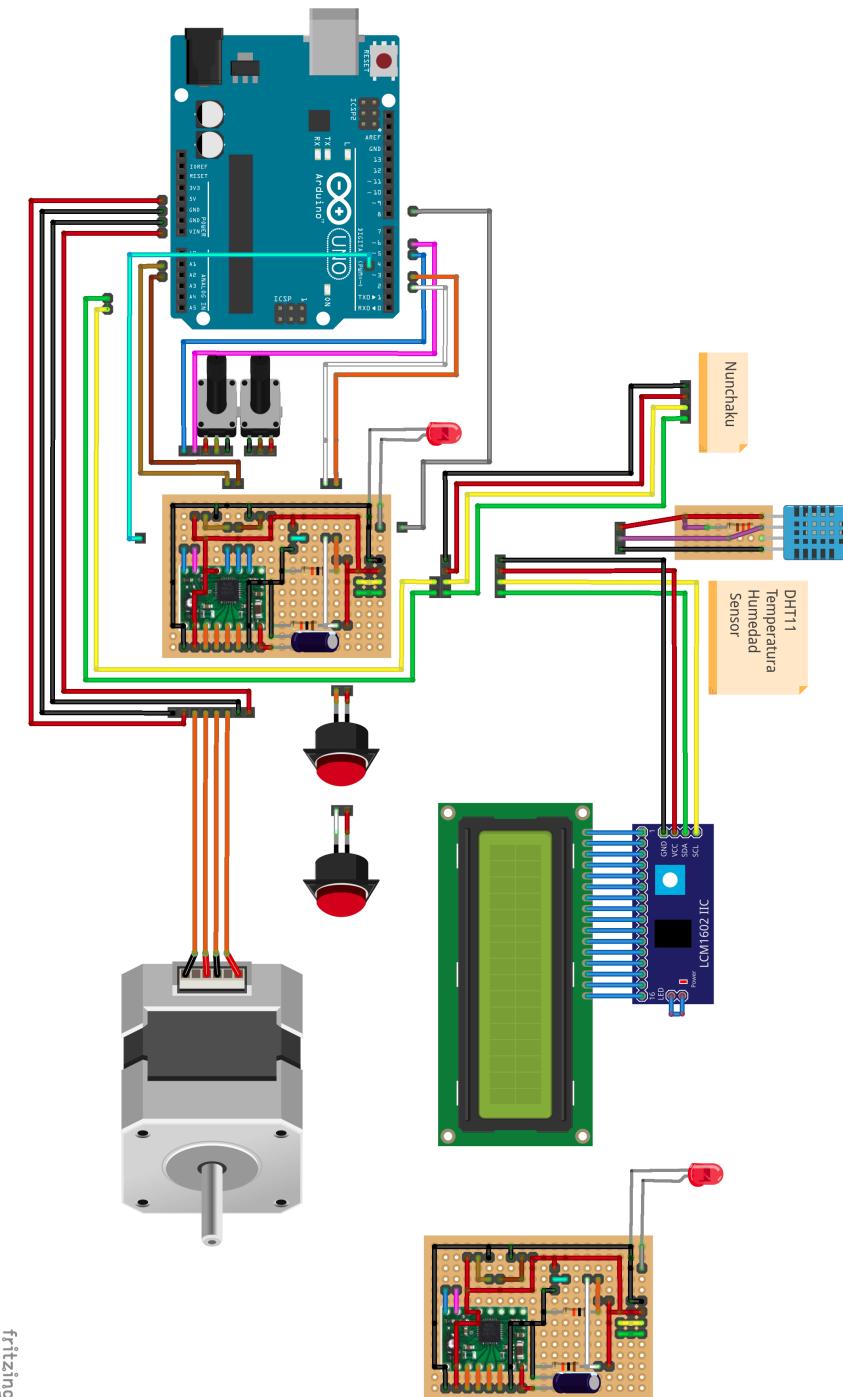


Figura A.4: Versión con pantalla LCD con interfaz I2C, mando Nunchuck, sensor de temperatura DHT11 (interfaz digital) y con controlador Arduino UNO.



## **Apéndice B**

### **Diagramas de Gantt**

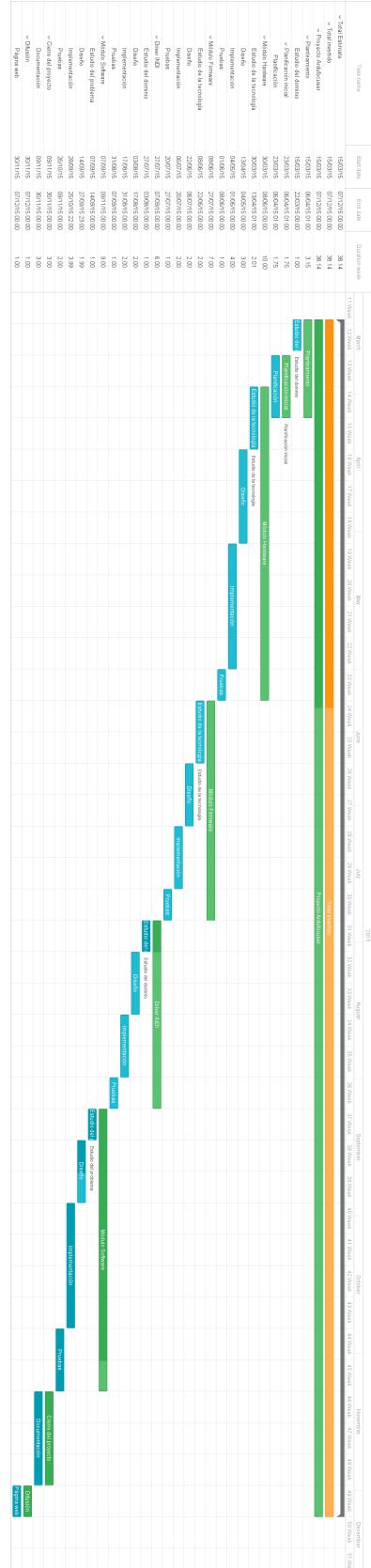


Figura B.1: Diagrama de Gantt Teórico

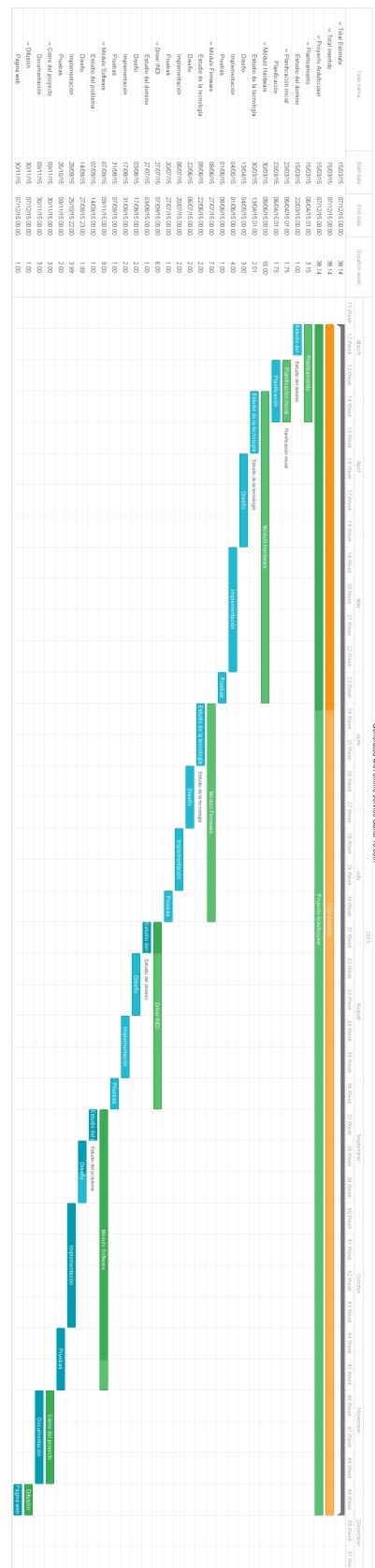


Figura B.2: Diagrama de Gantt Real



## **Apéndice C**

### **Diseño Carcasa**

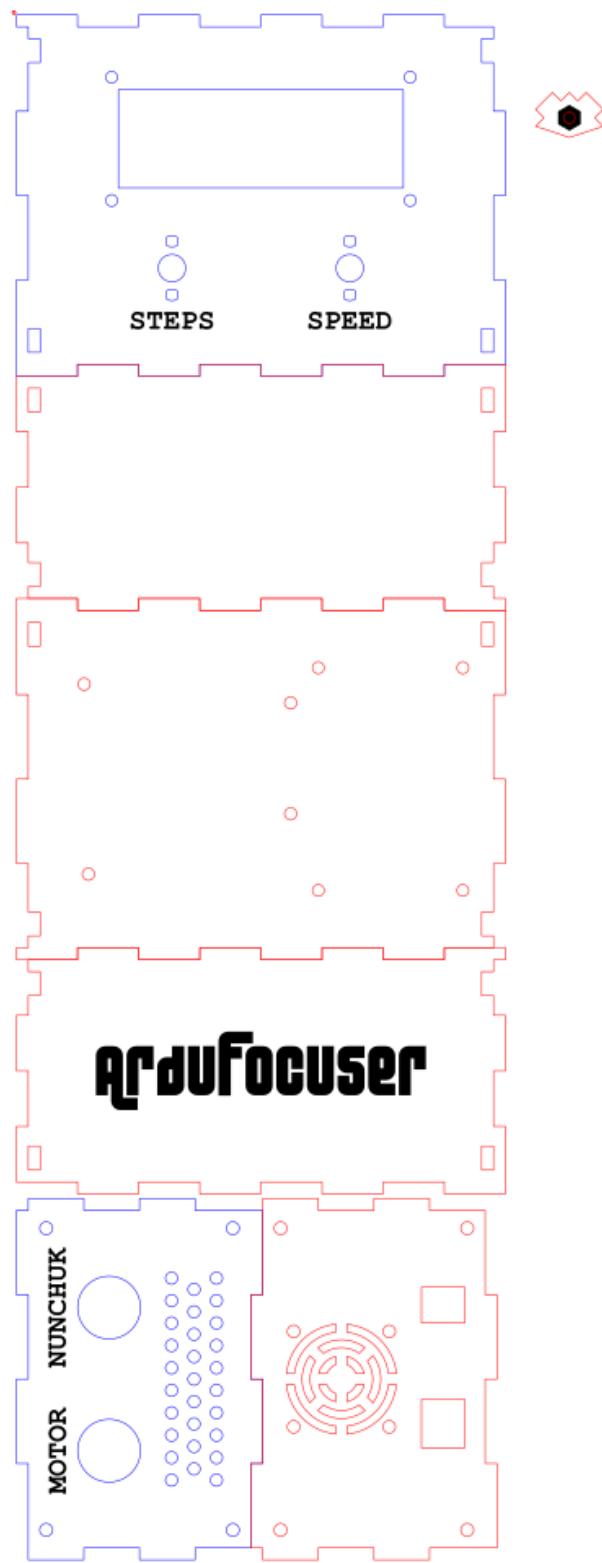


Figura C.1: Plano para cortar la carcasa con láser CNC.

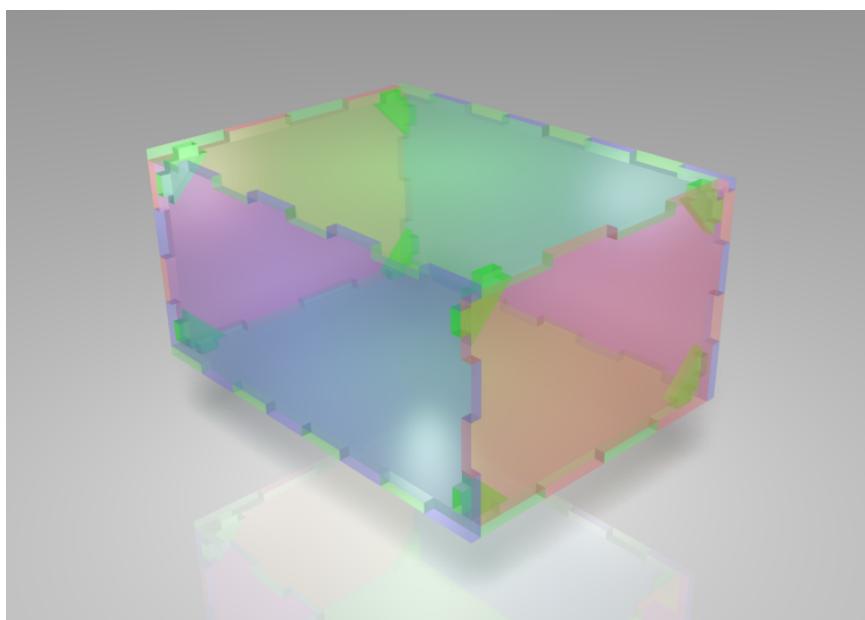


Figura C.2: Vista tridimensional del ensamblado de la carcasa. Visualización creada con Blender.



## Apéndice D

# Pruebas

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

#define MIN 0
#define MAX 100

LiquidCrystal_I2C lcd(0x27, 16, 2);
unsigned long lastTimeUpdate=100;
int lastPulse;

void PotController(){
    int result, result2;
    int pot = analogRead(A0);
    int pot2 = analogRead(A1);
    result = map(pot, 0, 1024, MIN, MAX );
    result2 = map(pot2, 0, 1024, MIN, MAX );
    lcd.clear();
    lcd.print(result); lcd.print(" "); lcd.print(result2);
}

void setup() {
    Serial.begin(9600);
    lcd.begin();
    lcd.backlight();
    lcd.print("RUN!");
}

void loop() {
    if (millis() > lastTimeUpdate) {
        PotController();
        lastTimeUpdate = millis() + 100;
    }
}
```

Fragmento de Código D.1: Pruebas lectura de valores potenciómetros

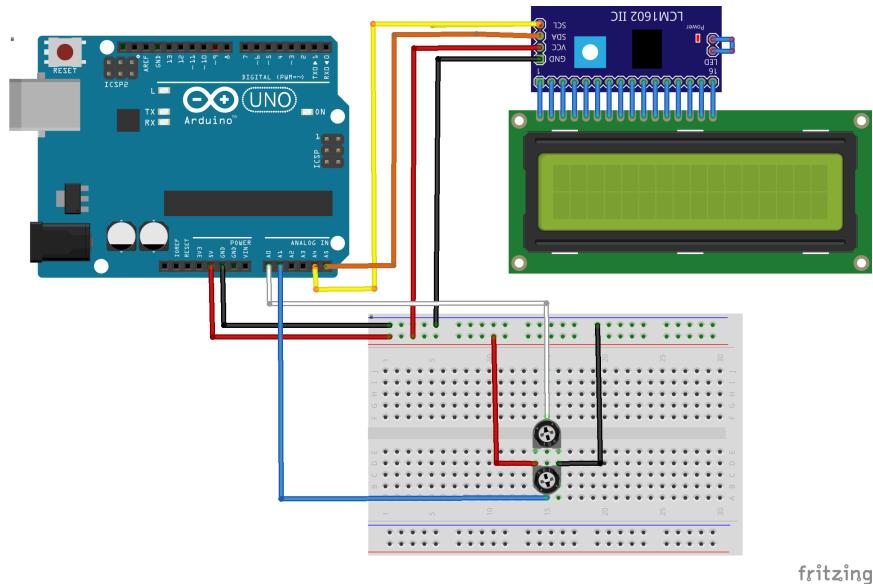


Figura D.1: Circuito para las pruebas lectura de valores potenciómetros

```

1 #include <AccelStepper.h>
2 #define PINDIR 3
3 #define PINSTEP 2
4
5 AccelStepper motor(1, PINSTEP, PINDIR);
6
7 void setup() {
8     Serial.begin(9600);
9     motor.setMaxSpeed(500);
10    motor.setAcceleration(100);
11 }
12
13 void loop() {
14     motor.run();
15     motor.moveTo(-3000);
16 }
```

Fragmento de Código D.2: Prueba mover motor paso a paso

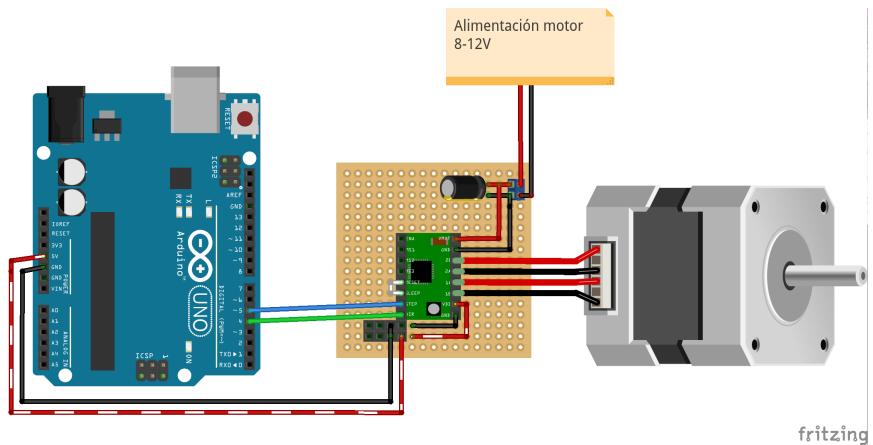
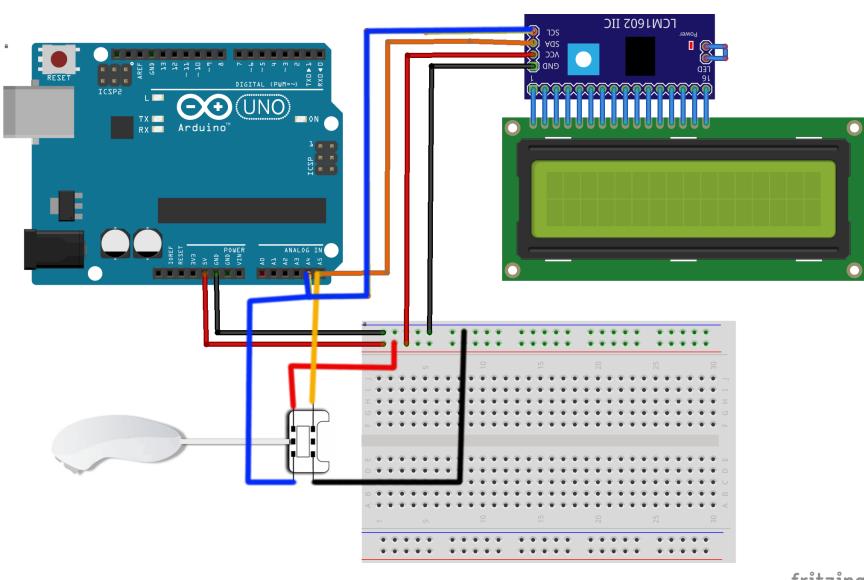


Figura D.2: Circuito para la prueba mover motor paso a paso

```

1 #include <Wire.h>
2 #include <math.h>
3 #include <nunchuck.h>
4 #include <LiquidCrystal_I2C.h>
5
6 LiquidCrystal_I2C lcd(0x27, 16, 2);
7 WiiChuck chuck = WiiChuck();
8 const int btnRIGHT=0, btnUP=1, btnDOWN=2, btnLEFT=3;
9 const int btnC=4, btnZ=5, btnNONE=6;
10 int lastTimeUpdate=1000;
11 int lastPulse;
12
13 void nunckuckController(){
14     chuck.update();
15     if(chuck.cPressed() && lastPulse!=btnC ){
16         lcd.clear();
17         lcd.print("C");
18         lastPulse=btnC;
19     }if(chuck.zPressed() && lastPulse!=btnZ){
20         lcd.clear();
21         lcd.print("Z");
22         lastPulse=btnZ;
23     }if(chuck.rightJoy() && lastPulse!=btnRIGHT){
24         lcd.clear();
25         lcd.print("RIGHT");
26         lastPulse=btnRIGHT;
27     }if(chuck.leftJoy()&& lastPulse!=btnLEFT){
28         lcd.clear();
29         lcd.print("LEFT");
30         lastPulse=btnLEFT;
31     }else {
32         lastPulse=btnNONE;
33     }
34 }
35
36 void setup() {
37     Serial.begin(9600);
38     lastPulse=btnNONE;
39     lcd.begin();
40     lcd.backlight();
41     chuck.begin();
42     chuck.update();
43 }
44
45 void loop() {
46     if (millis() > lastTimeUpdate) {
47         nunckuckController();
48         lastTimeUpdate = millis();
49     }
50 }
```

Fragmento de Código D.3: Prueba Wii nunchuck



fritzing

Figura D.3: Circuito para la prueba Wii nunchuck



# Bibliografía

- [1] About. <http://indilib.org/about.html>, (Último acceso: 08/30/2016)
- [2] AbstractCurveFitter (Apache Commons Math 3.6.1 API).  
<http://commons.apache.org/proper/commons-math/apidocs/org/apache/commons/math3/fitting/AbstractCurveFitter.html>, (Último acceso: 08/09/2016)
- [3] AccelStepper: AccelStepper library for Arduino. <http://www.airspayce.com/mikem/arduino/AccelStepper/>, (Último acceso: 06/19/2016)
- [4] Ambiente de desarrollo integrado - Wikipedia, la enciclopedia libre. [https://es.wikipedia.org/wiki/Ambiente\\_de\\_desarrollo\\_integrado](https://es.wikipedia.org/wiki/Ambiente_de_desarrollo_integrado), (Último acceso: 04/17/2016)
- [5] Análisis espectral - Wikipedia, la enciclopedia libre. [https://es.wikipedia.org/wiki/An%C3%A1lisis\\_espectral](https://es.wikipedia.org/wiki/An%C3%A1lisis_espectral), (Último acceso: 04/10/2016)
- [6] Arduino - IDE. <https://www.arduino.cc/en/Main/Software>, (Último acceso: 08/20/2016)
- [7] Arduino - Libraries. <https://www.arduino.cc/en/Reference/Libraries>, (Último acceso: 08/30/2016)
- [8] Arduino - Sketch. <https://www.arduino.cc/en/Tutorial/Sketch>, (Último acceso: 08/30/2016)
- [9] Arduino - Wikipedia, la enciclopedia libre. <https://es.wikipedia.org/wiki/Arduino>, (Último acceso: 04/16/2016)
- [10] Aristarco de Samos.  
[https://es.wikipedia.org/wiki/Aristarco\\_de\\_Samos](https://es.wikipedia.org/wiki/Aristarco_de_Samos), (Último acceso: 04/10/2016)

- [11] ASCOM (standard) - Wikipedia, la free encyclopedia. [https://en.wikipedia.org/wiki/ASCOM\\_\(standard\)](https://en.wikipedia.org/wiki/ASCOM_(standard)), (Último acceso: 08/23/2016)
- [12] AstroCiencias Ecuador: Astronomía y Cosmología. <http://astrocienciasecu.blogspot.com.es/p/mucha-de-la-gente-que-se-inicia-en-la.html>, (Último acceso: 08/30/2016)
- [13] Astrofotografía - Wikipedia, la enciclopedia libre. <https://es.wikipedia.org/wiki/Astrofotograf%C3%A1A>, (Último acceso: 04/12/2016)
- [14] Astrofísica - Wikipedia, la enciclopedia libre. <https://es.wikipedia.org/wiki/Astrof%C3%ADsica>, (Último acceso: 04/10/2016)
- [15] astrogranada. <http://www.astrogranada.org/>, (Último acceso: 08/28/2016)
- [16] Astrología. <https://es.wikipedia.org/wiki/Astrolog%C3%A1A>, (Último acceso: 04/10/2016)
- [17] Astronomía - Wikipedia, la enciclopedia libre. <https://es.wikipedia.org/wiki/Astronom%C3%A1A>, (Último acceso: 08/30/2016)
- [18] AstroRED.net - Astronomia Sur. <http://www.astrored.net/astronomiasur/>, (Último acceso: 08/30/2016)
- [19] Atik 314L + - Atik Cameras. <http://www.atik-cameras.com/product/atik-314l-plus/>, (Último acceso: 07/06/2016)
- [20] Atmel AVR 8-bit and 32-bit Microcontrollers. <http://www.atmel.com/products/microcontrollers/avr/>, (Último acceso: 04/17/2016)
- [21] Calendario egipcio - Wikipedia, la enciclopedia libre. [https://es.wikipedia.org/wiki/Calendario\\_egipcio](https://es.wikipedia.org/wiki/Calendario_egipcio), (Último acceso: 08/30/2016)
- [22] CloudMakers. <http://www.cloudmakers.eu/atikdrivers/>, (Último acceso: 08/09/2016)
- [23] CMDann.ca — Install The PiFace Control & Display. <http://cmdann.ca/tutorials/install-piface-lcd-shield/>, (Último acceso: 08/30/2016)

- [24] Corrección gamma - Wikipedia, la enciclopedia libre. [https://es.wikipedia.org/wiki/Correcci%C3%B3n\\_gamma](https://es.wikipedia.org/wiki/Correcci%C3%B3n_gamma), (Último acceso: 08/08/2016)
- [25] Cámara CCD Starlight Xpress Trius-SX9 - Lunatico Astronomia. <http://tienda.lunatico.es/Starlight-Xpress-Trius-SX9-Monocroma-camara-CCD-con-accesorios>, (Último acceso: 08/30/2016)
- [26] Cúpulas - Lunatico Astronomia. <http://tienda.lunatico.es/Cupulas>, (Último acceso: 08/30/2016)
- [27] Diarama conexiones Wii Nunchuck. [https://fr.wikiversity.org/wiki/Micro\\_contr\unhbox\voidb@x\bgroup\let\unhbox\voidb@x\setbox@\tempboxa\hbox{o\global\mathchardef\accent@spacefactor\spacefactor}\spacefactor}\accent94o\egroup\spacefactor\accent@spacefactorleurs AVR](https://fr.wikiversity.org/wiki/Micro_contr\unhbox\voidb@x\bgroup\let\unhbox\voidb@x\setbox@\tempboxa\hbox{o\global\mathchardef\accent@spacefactor\spacefactor}\spacefactor}\accent94o\egroup\spacefactor\accent@spacefactorleurs AVR), (Último acceso: 06/25/2016)
- [28] Diario de un Telescopio Reflector tipo Newton (Pontevedra 2012): El Enfocador. (Focuser). <http://eldiariodeuntelescopio.blogspot.com.es/2012/05/el-enfocador-focuser.html>, (Último acceso: 08/30/2016)
- [29] Dispositivo de carga acoplada. [https://es.wikipedia.org/wiki/Dispositivo\\_de\\_carga\\_acoplada](https://es.wikipedia.org/wiki/Dispositivo_de_carga_acoplada), (Último acceso: 08/23/2016)
- [30] El estudio de la Astronomía por las civilizaciones antiguas. <https://lanaveva.wordpress.com/2011/06/10/el-estudio-de-la-astronomia-por-las-civilizaciones-antiguas/>, (Último acceso: 04/10/2016)
- [31] Encuentro astronómico La Sagra 2015. <http://astroencuentro.wixsite.com/lasagra>, (Último acceso: 08/28/2016)
- [32] Enfocador 2,7" Tecnosky para refractores - Lunatico Astronomía. [http://tienda.lunatico.es/epages/lunatico.sf/es\\_ES/?ObjectPath=/Shops/1010001/Products/CENCRTKrp27](http://tienda.lunatico.es/epages/lunatico.sf/es_ES/?ObjectPath=/Shops/1010001/Products/CENCRTKrp27), (Último acceso: 08/30/2016)
- [33] Escritorio remoto - Wikipedia, la enciclopedia libre. [https://es.wikipedia.org/wiki/Escritorio\\_remoto](https://es.wikipedia.org/wiki/Escritorio_remoto), (Último acceso: 08/23/2016)
- [34] File:Messier 88 galaxy.jpg - Wikimedia Commons. [https://commons.wikimedia.org/wiki/File:Messier\\_88\\_galaxy.jpg](https://commons.wikimedia.org/wiki/File:Messier_88_galaxy.jpg), (Último acceso: 08/30/2016)
- [35] FITS - Wikipedia, la enciclopedia libre. <https://es.wikipedia.org/wiki/FITS>, (Último acceso: 04/25/2016)

- [36] Fritzing Fritzing. <http://fritzing.org/home/>, (Último acceso: 08/23/2016)
- [37] Full Width at Half Maximum. [https://es.wikipedia.org/wiki/Anchura\\_a\\_media\\_altura](https://es.wikipedia.org/wiki/Anchura_a_media_altura), (Último acceso: 07/28/2016)
- [38] Galileo Galilei - Wikipedia, la enciclopedia libre. [https://es.wikipedia.org/wiki/Galileo\\_Galilei#Invenci.C3.B3n\\_del\\_telescopio](https://es.wikipedia.org/wiki/Galileo_Galilei#Invenci.C3.B3n_del_telescopio), (Último acceso: 08/30/2016)
- [39] Gaussian Function – from Wolfram MathWorld. <http://mathworld.wolfram.com/GaussianFunction.html>, (Último acceso: 08/09/2016)
- [40] Ginga: Image Viewer and Toolkit — ginga v2.5.20160810184000. <https://ginga.readthedocs.io/en/latest/>, (Último acceso: 08/23/2016)
- [41] The GNU General Public License v3.0 - GNU Project - Free Software Foundation. <http://www.gnu.org/licenses/gpl-3.0.html>, (Último acceso: 06/12/2016)
- [42] Google Sky. [https://www.google.com/intl/es\\_es/sky/](https://www.google.com/intl/es_es/sky/), (Último acceso: 08/23/2016)
- [43] Hardware libre - Wikipedia, la enciclopedia libre. [https://es.wikipedia.org/wiki/Hardware\\_libre](https://es.wikipedia.org/wiki/Hardware_libre), (Último acceso: 04/16/2016)
- [44] hatboysamJavaDuino: A very, very simple Java starter project for talking to Arduino over Serial. <https://github.com/hatboysam/JavaDuino>, (Último acceso: 06/26/2016)
- [45] Historia de la astronomía - Wikipedia, la enciclopedia libre. [https://es.wikipedia.org/wiki/Historia\\_de\\_la\\_astronom%C3%ADA](https://es.wikipedia.org/wiki/Historia_de_la_astronom%C3%ADA), (Último acceso: 08/30/2016)
- [46] Home. <http://indilib.org/>, (Último acceso: 08/23/2016)
- [47] Homepage of Preben Grosbol. [http://www.eso.org/~pgrosbol/fits\\_java/jfits.html](http://www.eso.org/~pgrosbol/fits_java/jfits.html), (Último acceso: 07/05/2016)
- [48] ImageJ. <https://imagej.nih.gov/ij/>, (Último acceso: 04/25/2016)
- [49] INDI for Java. <http://indilib.org/develop/indiforjava.html>, (Último acceso: 04/25/2016)
- [50] INDI for Java. <http://indilib.org/develop/indiforjava.html>, (Último acceso: 06/25/2016)
- [51] Isaac Newton. [https://es.wikipedia.org/wiki/Isaac\\_Newton](https://es.wikipedia.org/wiki/Isaac_Newton), (Último acceso: 04/10/2016)

- [52] Johannes Kepler.  
[https://es.wikipedia.org/wiki/Johannes\\_Kepler](https://es.wikipedia.org/wiki/Johannes_Kepler),  
(Último acceso: 04/10/2016)
- [53] The KDE Education Project - KStars. <https://edu.kde.org/kstars/>, (Último acceso: 08/20/2016)
- [54] kroimon/Arduino-SerialCommand: A Wiring/Arduino library to tokenize and parse commands received over a serial port. <https://github.com/kroimon/Arduino-SerialCommand>, (Último acceso: 06/25/2016)
- [55] La Astronomía Moderna.  
<http://www.astroelda.com/index.php/16-actividades/27-la-astronomia-moderna>,  
(Último acceso: 04/10/2016)
- [56] Las observaciones del cielo profundo de Charles Messier. <http://www.astronomia-iniciacion.com/las-observaciones-del-cielo-profundo-de-charles-messier.html>, (Último acceso: 04/10/2016)
- [57] LCD Keypad Shield —. <http://www.prometec.net/producto/lcd-keypad/>, (Último acceso: 08/25/2016)
- [58] Lost infinity - Reaching for the stars. <http://www.lost-infinity.com/equipment/>, (Último acceso: 04/23/2016)
- [59] Math – Commons Math: The Apache Commons Mathematics Library. <http://commons.apache.org/proper/commons-math/>, (Último acceso: 08/09/2016)
- [60] Maxim DL. <http://www.cyanogen.com/>, (Último acceso: 08/23/2016)
- [61] Metaetiquetas que Google entiende - Ayuda de Search Console. <https://support.google.com/webmasters/answer/79812?hl=es>, (Último acceso: 08/28/2016)
- [62] MICRODRIVE. [http://www.pfinalfra.com.ar/proyectos/Microstepping/MICRODRIVE.htm#QUE\\_ES\\_MICROSTEPPING](http://www.pfinalfra.com.ar/proyectos/Microstepping/MICRODRIVE.htm#QUE_ES_MICROSTEPPING), (Último acceso: 08/20/2016)
- [63] Montura - Telescopio. <https://es.wikipedia.org/wiki/Telescopio#Monturas>, (Último acceso: 08/23/2016)
- [64] Montura ecuat. SkyWatcher EQ-8 con trípode - Lunático Astronomía. <http://tienda.lunatico.es/Montura-ecuatorial-SkyWatcher-EQ-8-con-tripod>, (Último acceso: 08/30/2016)

- [65] Motor de enfoque ORION AccuFocus. <https://www.valkanik.com/portaoculares-y-sujetaoculares/motor-de-enfoque-orion-accufocus>, (Último acceso: 08/30/2016)
- [66] NetBeans. <https://netbeans.org/>, (Último acceso: 08/20/2016)
- [67] Nicolás Copérnico. [https://es.wikipedia.org/wiki/Nicol%C3%A1s\\_Cop%C3%A9rnico](https://es.wikipedia.org/wiki/Nicol%C3%A1s_Cop%C3%A9rnico), (Último acceso: 04/10/2016)
- [68] Observatorio Remoto. [https://play.google.com/store/apps/details?id=com.jtbenavente.jaime.indiandroidui&hl=es\\_419](https://play.google.com/store/apps/details?id=com.jtbenavente.jaime.indiandroidui&hl=es_419), (Último acceso: 08/20/2016)
- [69] An open source ecosystem for IoT development · PlatformIO. <http://platformio.org/>, (Último acceso: 05/01/2016)
- [70] Optec :: TCF Focuser Series. <https://optecinc.com/astronomy/catalog/tcf/index.htm>, (Último acceso: 08/30/2016)
- [71] Philosophiæ naturalis principia mathematica - Wikipedia, la enciclopedia libre. [https://es.wikipedia.org/wiki/Philosophi%C3%A6\\_naturalis\\_principia\\_mathematica](https://es.wikipedia.org/wiki/Philosophi%C3%A6_naturalis_principia_mathematica), (Último acceso: 08/30/2016)
- [72] Pitágoras. <https://es.wikipedia.org/wiki/Pit%C3%A1goras>, (Último acceso: 04/10/2016)
- [73] Pololu - A4988 Stepper Motor Driver Carrier. <https://www.pololu.com/product/1182>, (Último acceso: 04/30/2016)
- [74] Posicionamiento en buscadores - Wikipedia, la enciclopedia libre. [https://es.wikipedia.org/wiki/Posicionamiento\\_en\\_buscadores](https://es.wikipedia.org/wiki/Posicionamiento_en_buscadores), (Último acceso: 08/28/2016)
- [75] Presentando Astropy: herramientas básicas para Astronomía y Astrofísica en Python — Pybonacci. <http://pybonacci.org/2012/06/20/presentando-astropy-herramientas-basicas-para-astronomia-y-astrofisica-en-python.html>, (Último acceso: 07/05/2016)
- [76] Raspberry Pi - Teach, Learn, and Make with Raspberry Pi. <https://www.raspberrypi.org/>, (Último acceso: 08/23/2016)
- [77] Raspbian. <https://www.raspbian.org/>, (Último acceso: 08/20/2016)
- [78] Raw (formato) - Wikipedia, la enciclopedia libre. [https://es.wikipedia.org/wiki/Raw\\_\(formato\)](https://es.wikipedia.org/wiki/Raw_(formato)), (Último acceso: 04/25/2016)

- [79] Rayos X - Wikipedia, la enciclopedia libre. [https://es.wikipedia.org/wiki/Rayos\\_X#Detectores\\_de\\_rayos\\_X](https://es.wikipedia.org/wiki/Rayos_X#Detectores_de_rayos_X), (Último acceso: 08/30/2016)
- [80] RoboFocus. <http://indilib.org/devices/focusers/robofocus.html>, (Último acceso: 08/30/2016)
- [81] RS-232 - Wikipedia, la enciclopedia libre. <https://es.wikipedia.org/wiki/RS-232>, (Último acceso: 08/25/2016)
- [82] Ruedas de filtros IFW — AstroImagen. <https://astroimagen.wordpress.com/productos/optec/ruedas-de-filtros-ifw/>, (Último acceso: 08/30/2016)
- [83] Seeing (Wikipedia). <https://es.wikipedia.org/wiki/Seeing>, (Último acceso: 08/13/2016)
- [84] Sensor DHT11 (Humedad y Temperatura) con Arduino — Taller Arduino. <https://tallerarduino.com/2012/12/24/sensor-dht11-humedad-y-temperatura-con-arduino/>, (Último acceso: 08/20/2016)
- [85] Shields para Arduino — Aprendiendo Arduino. <https://aprendiendoarduino.wordpress.com/2015/03/23/shields-para-arduino/>, (Último acceso: 04/17/2016)
- [86] Sierra Nevada night sky time lapse 2, In Memory to Iker. <https://vimeo.com/28399458>, (Último acceso: 08/30/2016)
- [87] Stellarium. <http://www.stellarium.org/es/>, (Último acceso: 08/23/2016)
- [88] Sublime Text. <https://www.sublimetext.com/>, (Último acceso: 08/20/2016)
- [89] Tablero de fibra de densidad media - Wikipedia, la enciclopedia libre. [https://es.wikipedia.org/wiki/Tablero\\_de\\_fibra\\_de\\_densidad\\_media](https://es.wikipedia.org/wiki/Tablero_de_fibra_de_densidad_media), (Último acceso: 09/04/2016)
- [90] Tele Vue Optics: Focusmate<sup>TM</sup> Driver. [http://televue.com/engine/TV3b\\_page.asp?id=59&Tab=\\_FMD](http://televue.com/engine/TV3b_page.asp?id=59&Tab=_FMD), (Último acceso: 08/30/2016)
- [91] Telescopios - Características, modelos, partes :: Astronomía Sur. <http://www.astrosurf.com/astronosur/telescopios.htm>, (Último acceso: 04/23/2016)
- [92] Televisión por satélite - Wikipedia, la enciclopedia libre. [https://es.wikipedia.org/wiki/Televisi%C3%B3n\\_por\\_sat%C3%A9lite](https://es.wikipedia.org/wiki/Televisi%C3%B3n_por_sat%C3%A9lite), (Último acceso: 08/30/2016)

- [93] Tycho Brahe.  
[https://es.wikipedia.org/wiki/Tycho\\_Brahe](https://es.wikipedia.org/wiki/Tycho_Brahe),  
(Último acceso: 04/10/2016)
- [94] Urban Astrophotography. <http://laazotea.org/>, (Último acceso: 08/30/2016)
- [95] Welcome. <http://www.robofocus.com/>, (Último acceso: 06/25/2016)
- [96] William Huggins - Wikipedia, la enciclopedia libre. [https://es.wikipedia.org/wiki/William\\_Huggins](https://es.wikipedia.org/wiki/William_Huggins), (Último acceso: 04/10/2016)
- [97] www.carlostapia.es/articulos/archivos/Mascaras.pdf. <http://www.carlostapia.es/articulos/archivos/Mascaras.pdf>,  
(Último acceso: 04/23/2016)
- [98] www.ti.com/lit/ds/symlink/lm35.pdf. <http://www.ti.com/lit/ds/symlink/lm35.pdf>, (Último acceso: 04/30/2016)
- [99] XBee: Connect Devices To The Cloud - Digi International. <http://www.digi.com/lp/xbee>, (Último acceso: 04/17/2016)
- [100] ¡Focus Focus! - Cielos Boreales. <http://www.cielosboreales.com/hacer-foco-telescopio-camara/>, (Último acceso: 04/23/2016)
- [101] ¿Para qué sirve la Astronomía? Usos en medicina — Cuaderno de bitácora estelar. <http://www.madrimasd.org/blogs/astrofisica/2014/10/31/133460>, (Último acceso: 04/10/2016)
- [102] Gartner: Gartner's 2015 Hype Cycle for Emerging Technologies Identifies the Computing Innovations That Organizations Should Monitor. <http://www.gartner.com/newsroom/id/3114217>, (Último acceso: 07/26/2016)
- [103] Hesse, J.M.M.: Los beneficios de la Astronomía — Cuaderno de bitácora estelar. <http://www.madrimasd.org/blogs/astrofisica/2013/03/22/132620>, (Último acceso: 04/10/2016)