industriales
etsii

Escuela Técnica Superior de Ingeniería Industrial

UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Escuela Técnica Superior de Ingeniería Industrial

Proyecto FastCam: Control y Adquisición del instrumento de Lucky Imaging para los observatorios astronómicos de Canarias

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Autor: **MªCRUZ RAMÍREZ TRUJILLO**
Director: ANTONIO PÉREZ GARRIDO
Codirector: ISIDRO PÉREZ VILLO
ROBERTO LÓPEZ LÓPEZ

Cartagena, a 11 de mayo de 2020

Universidad Politécnica de Cartagena

Índice

1.-Estado del Arte

Las turbulencias atmosféricas limitan la resolución espacial de los telescopios terrestres. Para paliar esta dificultad se desarrolló el método de "*Lucky Imaging"*, el cuál es capaz de llevar los telescopios al límite de difracción, coincidente con el límite de resolución teórico.

La técnica del "*Lucky Imaging"*, también conocida como la técnica de las exposiciones afortunadas, es una forma de "*Speckle Imaging*" usada inicialmente en astrofotografía. Se tomarán series de miles de imágenes dentro del tiempo de exposición inferior a los 30 ms, sabiendo que en algunas de esas imágenes tomadas la perturbación atmosférica no será significativa (lo que se conoce como una exposición afortunada). Este será un factor de selección para obtener un lote de imágenes de la mayor calidad posible. Este lote se utilizará para la combinación de cada una de las imágenes obtenidas, mediante un método conocido como *Shift-and-Add*. Se obtendrá, finalmente, una imagen donde se ha logrado reconstruir el objeto de la adquisición minimizando al máximo posible las perturbaciones atmosféricas.

El *Instituto de Astrofísica de Canarias* crea, en colaboración con la *Universidad Politécnica de Cartagena*, un proyecto basado en la técnica anteriormente descrita para obtener imágenes de muy alta resolución espacial en telescopios terrestres. Este proyecto es el llamado ***FastCam***.

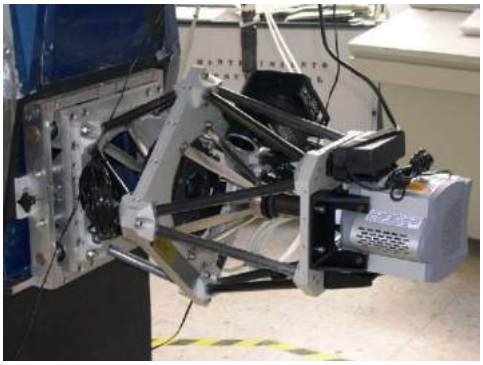


Figura 1.- FastCam instalado en el foco del Telescopio Carlos Sánchez (1.5m)
en el Observatorio del Teide.

El instrumento consiste en una cámara EMCCD de bajo ruido y altas prestaciones en cuanto eficiencia cuántica y velocidad de trabajo, que con el sistema óptico diseñado es capaz de llegar al límite de difracción en telescopios de tamaño medio.

El software que se diseñó especialmente para este proyecto fue pensado para extraer imágenes del detector, en tamaños de decenas de miles. Esto se debe a que la atmósfera tiene tiempos de variabilidad de microsegundos, entonces el sistema debe de ser capaz de responder a esa velocidad de cambio lo más rápido posible. Estamos hablando de tiempos inferiores a treinta milisegundos.

El motivo de este trabajo fin de grado parte de la necesidad de optimizar y renovar este proyecto y los componentes que lo rodean, para facilitar el estudio y trabajo de aquellos que investigan el cielo nocturno en los *Observatorios de Canarias*, dentro de los requisitos que impone el *Instituto de Astrofísica de Canarias*.

1.2.-Antecedentes

Hay que concebir en nuestra mente, para entender el propósito de este proyecto, la dificultad de alcanzar con detectores terrestres los objetos menos luminosos, ya sea porque son muy débiles o porque se encuentran muy alejados. Hoy en día disponemos de grandes telescopios ópticos terrestres, pero tenemos un gran enemigo en el proceso de adquisición de imágenes: la atmósfera. Las ondas de luz que intentamos captar con nuestros detectores se ven distorsionadas cuando la onda atraviesa la capa atmosférica.

Cuando un frente de ondas atraviesa una zona turbulenta se deforma. La atmósfera terrestre se caracteriza por su inestabilidad, ya que cambia su estado en intervalos muy cortos de tiempo. Por lo tanto, no solo tenemos el impedimento del tamaño del detector y la luz, sino nuestra propia atmósfera. El conjunto de efectos que provoca la atmósfera a la hora de adquirir imágenes se refiere al término de *seeing*. De hecho, un detector que enfoca un punto luminoso en el cielo nocturno, sin el procesado adecuado ni la debida corrección óptica, adquirirá una serie de imágenes que se pueden apreciar en la figura 2. El movimiento de la atmósfera hace que los rayos de luz refractados, procedentes de un punto luminoso, incidan en diferentes posiciones de la superficie del colector, por lo que, cuando se procesa en conjunto la acumulación de imágenes, este punto parece que está continuamente moviéndose en el firmamento.

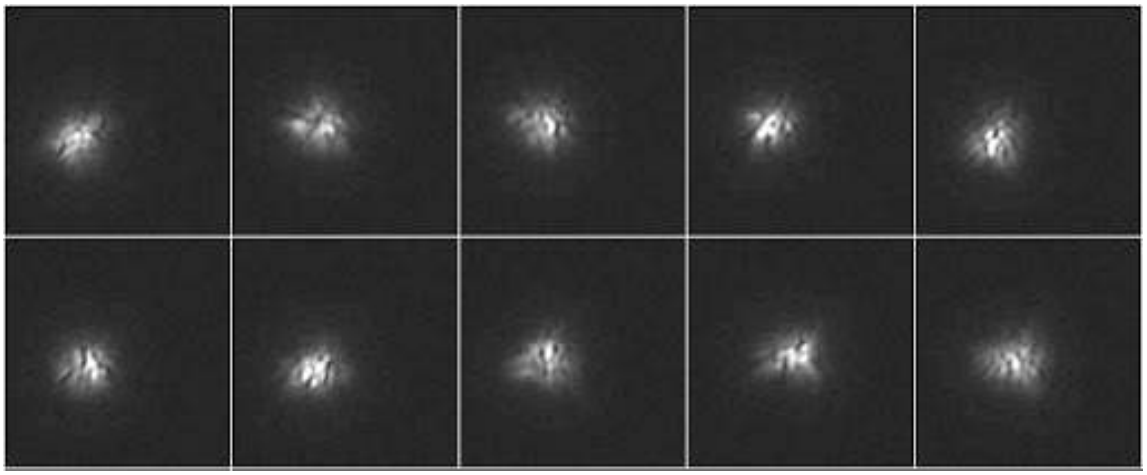


Figura 2.- El efecto de las turbulencias atmosféricas al observar una estrella del firmamento sin corrección en el campo óptico

Por último, cabe mencionar que el índice de refracción depende de la longitud de onda, y para determinadas bandas espectrales se propicia al defecto que se conoce como Dispersión Cromática Atmosférica.

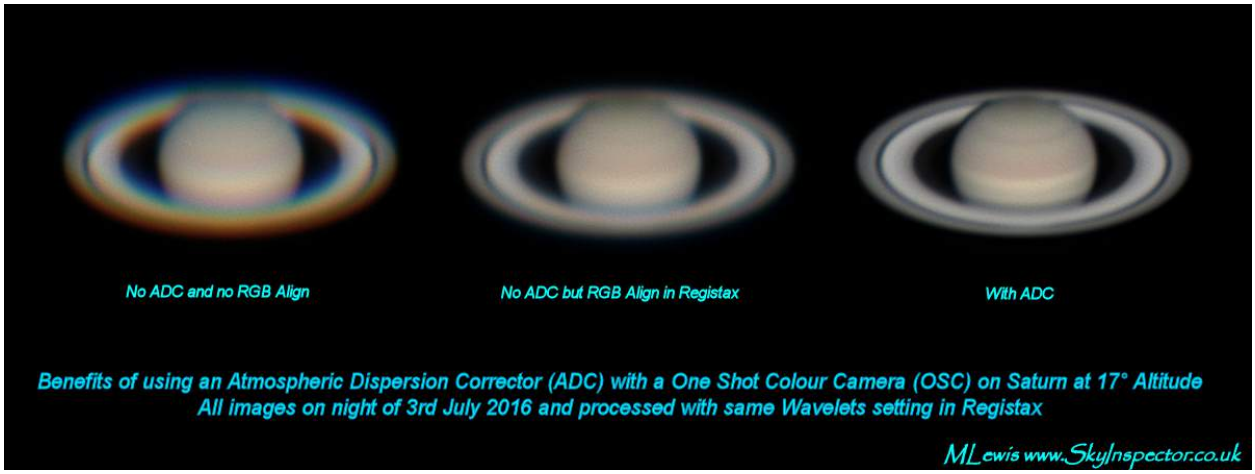


Figura 3.-Ejemplo de defecto de Dispersión Cromática Atmosférica
sobre Saturno y su corrección con ADC

El proyecto dispondrá de todo el material necesario para conseguir, a través de diversas técnicas que se irán explicando a lo largo de este documento, subsanar todos los defectos anteriormente expuestos para obtener imágenes de gran calidad desde telescopios terrestres.

1.3.- Análisis de proyecto

Se le llama FastCam al conjunto de los sistemas que intervienen en el proceso de adquisición de imágenes con una cámara EMCCD de Andor Technologies, junto con un software propio desarrollado por la Universidad Politécnica de Cartagena para el procesado de imágenes y la manipulación de la cámara en modo remoto.

En su versión anterior, el conjunto Andor Access y FastCam lograban su comunicación en un sistema Linux basada en conexión TCP/UDP. Estos protocolos de comunicación servían para que el dispositivo maestro pudiese transferir comandos a través de la línea TCP y recibir, por ejemplo, las imágenes de la cámara a través del canal UDP.

Este software intentó actualizarse y optimizarse para poder utilizarlo en nuevas versiones o diferentes implementaciones del sistema operativo. Desafortunadamente, por problemas de comunicación y desarrollo, quedó obsoleto. Se llegó a la conclusión de que habría que desarrollarlo enteramente desde cero para poder llegar a una actualización exitosa.

La base de esta nueva implementación es la modularización de los diferentes procesos que el software realiza:

- Control y acceso al detector
- Adquisición de datos
- Control para Instrumentación Óptica
- Post-procesado de datos aplicando el algoritmo base de Lucky Imaging

1.3.1.-Protocolo TCP

TCP (que significa Protocolo de Control de Transmisión) es uno de los principales protocolos en la capa de transporte del modelo TCP/IP. En el nivel de aplicación, posibilita la administración de datos que vienen del nivel más bajo del modelo, o van hacia él, (es decir, el protocolo IP). Cuando se proporcionan los datos al protocolo IP, los agrupa en datagramas IP. TCP es un protocolo orientado a conexión, es decir, que permite que dos máquinas que están comunicadas controlen el estado de la transmisión.

En el TCP se establecen las conexiones usando el protocolo de acuerdo a tres vías ("*three-way handshake"*). Para establecer una conexión, el servidor espera pasivamente una conexión entrante ejecutando las primitivas LISTEN y ACCEPT y especificando cierto origen, o bien, nadie en particular.

Desde el lado del cliente, se ejecuta una primitiva CONNECT especificando la dirección y el puerto IP con el que se desea conectar, el tamaño máximo de segmento TCP que está dispuesto aceptar y opcionalmente algunos datos de usuario (ejemplo: contraseña). La primitiva CONNECT envía un segmento TCP con el bit SYN encendido y el bit BACK apagado, y espera una respuesta.

Al llegar el segmento al destino, la entidad TCP ahí revisa si hay un proceso que haya ejecutado un LISTEN en el puerto indicado en el campo de puerto de destino. Si no lo hay, envía una contestación con el bit RST encendido para rechazar la conexión. Si algún proceso está escuchando en el puerto, este proceso recibe el segmento TCP entrante y puede entonces aceptar o rechazar la conexión; si la acepta, se devuelve un segmento de recibo.

Dentro de la interfaz de desarrollo de Qt Creator encontramos una librería creada esencialmente para este tipo de comunicación, llamada QTcpSocket. Esta permite implementar protocolos de redes estándares como POP3, SMTP y NNTP.

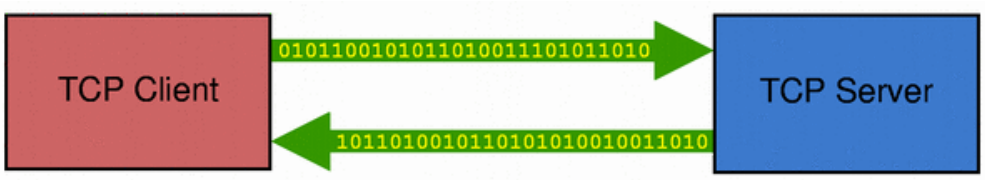


Figura 4.- Protocolo de Comunicación TCP

1.3.2.-Protocolo UDP

El protocolo UDP *("User Data Protocol")*, ofrece a las aplicaciones un mecanismo para enviar datagramas IP en bruto encapsulados sin tener que establecer una conexión. Muchas aplicaciones cliente-servidor que tienen una solicitud y una respuesta usan el UDP en lugar de tomarse la molestia de establecer y luego liberar una conexión.

UDP no admite numeración de los datagramas, factor que, sumando a que tampoco utiliza señales de entrega, hace que la garantía de que un paquete llegue a su destino sea mucho menor que si se usa TCP. Esto también origina que los datagramas pueden llegar duplicados y/o desordenados a su destino. Por estos motivos el control de envío de datagramas, si existe, debe de ser implementado por las aplicaciones que usan UDP como medio de transporte de datos, al igual que el reensamble de los mensajes entrantes.

Es por ello por lo que un protocolo del tipo *best-effort* hace lo que puede para transmitir los datagramas hacia la aplicación, pero no puede garantizar que la aplicación los reciba. Tampoco utiliza mecanismos de detección de errores. Cuando se detecta un error en un datagrama, en lugar de entregarlo a la aplicación de destino se descarta.

Cuando una aplicación envía datos a través de UDP, éstos llegan al otro extremo como una unidad. Por ejemplo, si una aplicación escribe 5 veces en el puerto UDP, la aplicación al otro extremo hará 5 lecturas del puerto UDP. Además, el tamaño de cada escritura será igual que el tamaño de las lecturas.

1.3.3.-UDP vs TCP

La parte problemática para nuestro caso es la poca seguridad para la transmisión completa de datos que la transmisión UDP ofrece. Por ejemplo, UDP es un canal de comunicación utilizado comúnmente en *online gaming*, y numerosos de los glitches que ocurren en estos videojuegos online son propios de problemas de transmisión UDP, ya que algunos paquetes se pierden en la transmisión. Para ese caso no importa, ya que lo que les importa a los jugadores es el tiempo real.

¿Pero qué ocurre si la importancia radica en el conjunto y no se pueden permitir pérdidas de transmisión, como es para el caso de transmisión de imágenes científicas? La poca seguridad que ofrece para este fin hace pensar que, aunque sea una comunicación más rápida y sencilla la UDP, hace pensar que tal vez merezca la pena plantearse la conexión TCP, más segura y eficaz. Esta es la base, por ejemplo, para grandes compañías de comunicaciones online. Pero vamos a ver estas diferencias paso a paso.

- **Conexión**

  TCP es un protocolo orientado a la conexión y el protocolo UDP es sin conexión. Esto significa que antes de enviar paquetes TCP, una conexión es establecida entre el servidor y el cliente. Este proceso es el llamado el "*handshaking"*. La transmisión de paquetes se realiza a través de dicha conexión.

En UDP los paquetes son enviados individualmente y direccionado por el remitente al recipiente sin necesidad de un canal seguro de datos.

- **Secuenciación**

  TCP es un protocolo fiable que añade un número a los paquetes de datos a la vez que los envía a través del canal de salida de datos. Esto ayuda a que el recipiente monte correctamente el mensaje al recibirlo. UDP no añade tal número a su cabecera, por lo que el recipiente no tiene ninguna manera de saber si ha recibido todos los paquetes y en el orden correcto.

- **Velocidad**

  Ya que UDP no tiene muchos requisitos, ofrece una conexión mucho más rápida. TCP, por el otro lado, es más lento, pero más fiable. Si se necesita más fiabilidad que velocidad se debería usar TCP en vez de UDP.

- **Fiabilidad**

  El protocolo TCP posee recursos para la secuenciación de paquetes, detección de errores y corrección. Esto lo hace también mucho más seguro y fiable frente a una transmisión UDP, ya que carece de esto.

- **Tamaño de encabezado**

  Debido a que el protocolo UDP carece de encabezado, la consecuencia inmediata es que es mucho más ligero frente a un paquete TCP. Esto es lo que realentiza el envío de información por TCP.

- **Detección de error/corrección**

  Esta es una capacidad característica del protocolo TCP. Cuando un paquete es identificado como corrupto, TCP ordena al servidor a reenviarlo. De esta forma, el mensaje completo es enviado sin errores.

  En el caso de UDP, se puede detectar un error, pero se descartan los datos; carece de sistema de corrección.

Son por estas características en las que he razonado en cambiar el sistema de comunicación de FastCam a un solo protocolo de comunicación. No interesa que cualquier ordenador que pueda identificar el ordenador tenga acceso a la retransmisión de datos, y sobretodo, tenemos que asegurarnos de que todos los datos son capaces de llegar correctamente a su destino.

1.3.4.-Corrector de Dispersión Atmosférica

Cuando observamos un objeto brillante como una estrella o un planeta a través de un telescopio, muchas veces observamos efectos cromáticos por culpa de la dispersión atmosférica. Podemos ver bordes rojizos y azules en un planeta como Júpiter, y además tener una imagen un poco distorsionada. Este efecto es provocado por la refracción de la luz proveniente de Júpiter interactuando con las moléculas de las capas altas de la atmósfera que se comportan como un prisma. Estos efectos varían ligeramente en función de la posición del objeto, pues si están cerca del horizonte, la luz realiza un recorrido más largo por la atmósfera, y los efectos serán más evidentes que si el objeto se encuentra en una posición cercana al cénit.

Los efectos ópticos se tratan, tradicionalmente, por separado. La dispersión atmosférica se corrige con Correctores de Dispersión Atmosférica (en inglés, ADC). Se trata de sistemas ópticos que van compensando en cada posición la refracción diferencial de las diferentes longitudes de onda. El sistema Corrector de Dispersión Atmosférica que posee el Departamento de Óptica, dentro del Instituto de Astrofísica de Canarias (al cual refiere este proyecto) se puede apreciar en la figura 5.

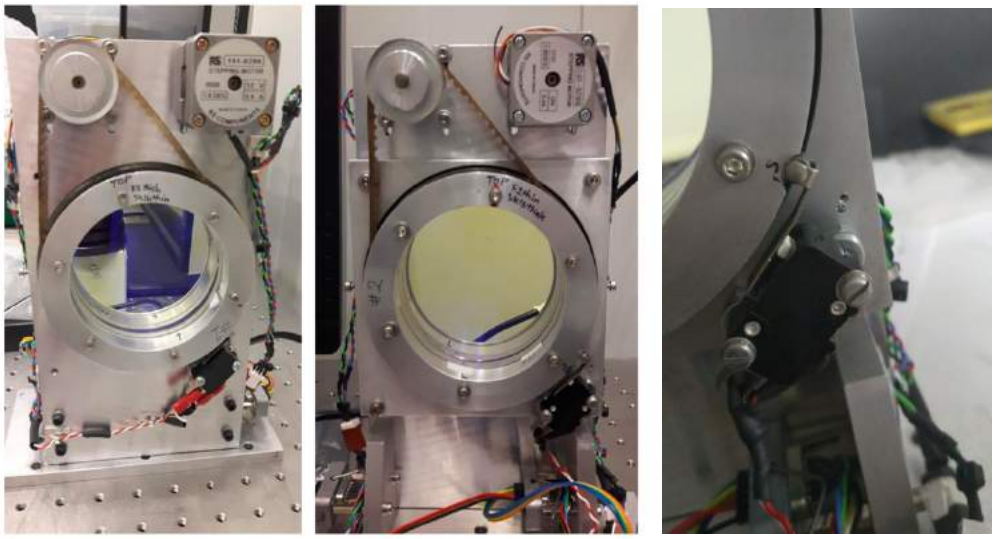


Figura 5.- (Izquierda y centro) Corrector de Dispersión Atmosférica
Figura 6.-(Derecha) Final de carrera utilizado para cada motor

El sistema corrector está formado por dos prismas, montados de forma opuesta, con capacidad de rotación individual para cada par. El objetivo de la capacidad de rotación se debe a que los prismas deben ser capaces de compensar la dispersión atmosférica a lo largo del proceso de adquisición de imágenes, ya que, las estrellas no se mantienen fijas en una misma posición a lo largo de la noche y adquirirán distintas orientaciones.

Para localizar el punto de referencia de cada prisma se utiliza un detector de final de carrera, el cual se puede apreciar en la figura 6, se activa cuando detecta una pequeña hendidura formada en los discos. La palanca del final de carrera se mantiene encima de la superficie de cada disco.

El movimiento rotativo se logra con dos motores paso a paso bipolares de RS Components (figura 7), cuyas especificaciones se podrán consultar en los anexos de este documento.

Figura 7.- Motor híbrido bipolares paso a paso de *RS Components*

Entre las especificaciones básicas, se encuentran:

- Motor híbrido unipolar
- 1.8 º por paso
- 0.10 Nm
- Soporta una alimentación de hasta 12 V en continua, con una intensidad de hasta 400 mA

La necesidad de trabajo sobre este sistema radica en que se desea poder utilizar y controlar este sistema corrector de modo independiente del fabricante. Es decir, este ADC posee una salida con conector DSub-HD (15 pines), con su propio programa de control. Debido a la poca versatilidad de este, se propone crear un sistema controlador basado en Arduino para lograr desarrollar algoritmos e interfaces de control propias para adaptar el ADC a las necesidades del científico. Se detallará más sobre su resolución en el apartado correspondiente.

Los objetivos a lograr son:

- Diseñar un entorno Arduino que permita controlar el motor Paso a Paso a través del Conector DSUB, estableciendo qué electrónica Arduino usar.

- Dentro del algoritmo que se diseñe, que el sistema sea capaz de encontrar el punto de referencia dado por el final de carrera.

- Dotar de una interfaz gráfica al algoritmo que se diseñe usando el entorno Qt.

- Realizar un algoritmo capaz de rotar, una sola vez, al cabo de todo el periodo de observación. Por ejemplo, si se realizan 6h de observación, que no se vuelva a detectar el HOME (final de carrera) hasta el fin del periodo de observación.

1.3.5.-Rueda de Filtros

La única manera de asegurar un cambio de filtros sin complicaciones es con una Rueda de Filtros, que facilite el cambio del filtro delante de la cámara gracias a un mecanismo giratorio. En este caso, la fuente del movimiento será el mismo motor paso a paso descrito en el apartado anterior, pero con una pequeña peculiaridad: la posición de referencia será detectado mediante sensores de Efecto Hall.

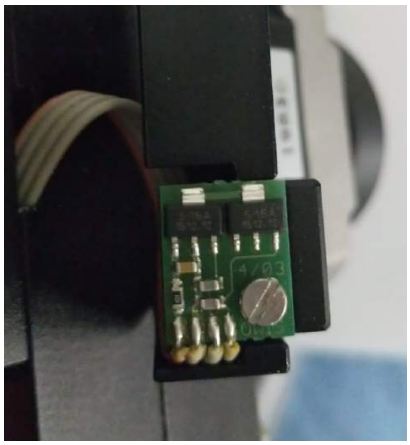


Figura 8.- Placa base de sensorización de la Rueda de Filtros

Esta electrónica se encontraba oculta en la parte superior del motor, separada por escasos milímetros del soporte de los filtros. Allí se encuentra incrustrado un pequeño imán, gracias al cual podemos marcar la referencia al ser detectado por los sensores. Merece la pena comentar que, pese a que se dispone de dos sensores, solo uno se aprovecha realmente.

Figura 9.-Composición estructural del sistema de referencia

En la figura 9 podemos apreciar como sobresale el imán sensiblemente de la carcasa de filtros. Este pasa justo por encima de los detectores de Efecto Hall. Todo lo anteriormente descrito se puede controlar a través del conector DSub-HD-15 que contiene la rueda de filtros. Los detalles de dicho conector se podrán apreciar en la figura 10.

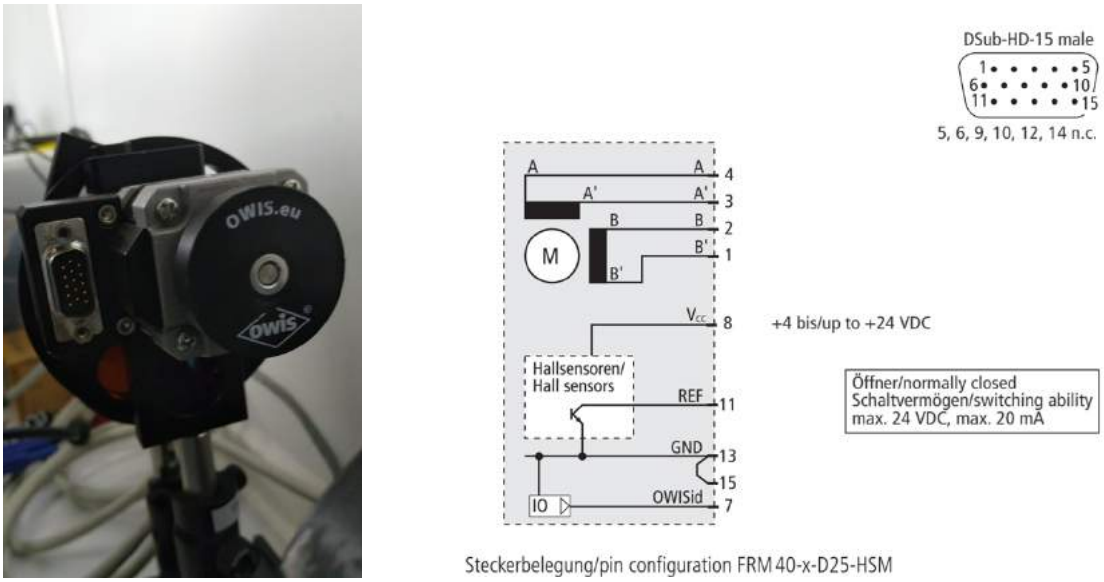


Figura 10.-Rueda de Filtros y esquemático de conexiones

Como propuesta de control adaptada para que funcione de manera independiente a la propuesta por el fabricante, se plantea una solución basada en Arduino, pero esta vez utilizando un Arduino Shield diseñado específicamente para controlar este tipo de estructuras de OWIS, que proporciona no solo el control del motor bifásico, sino tiene una solución adaptada para la adquisición de la señal del Efecto Hall. Se debe mencionar que la solución presentada no solo está adaptada para la rueda de filtros FRM 40 de OWIS, sino es una solución que sirve a una amplia gama de ruedas de filtros de dicho fabricante (a los que contengan el mismo conector). Se ha comprobado que distintas ruedas de filtros de OWIS poseen iguales características y esquemático del conector.

Los objetivos son, por lo tanto, que se disponga:

- Placa Arduino Shield personalizada a la Rueda de filtros, de bajo coste
- Software de adquisición y control propio para la Arduino Shield diseñada
- Interfaz gráfica propia

2.-Desarrollo de Software

2.1.-Nuevo Software de Control

El nuevo software de control se ha planteado para que sea rápido de manejar, intuitivo y eficaz. Dando un salto a que la nueva versión esté adaptada no solo para Linux (como era el caso de su anterior versión) sino para que funcione en Windows y en Mac.

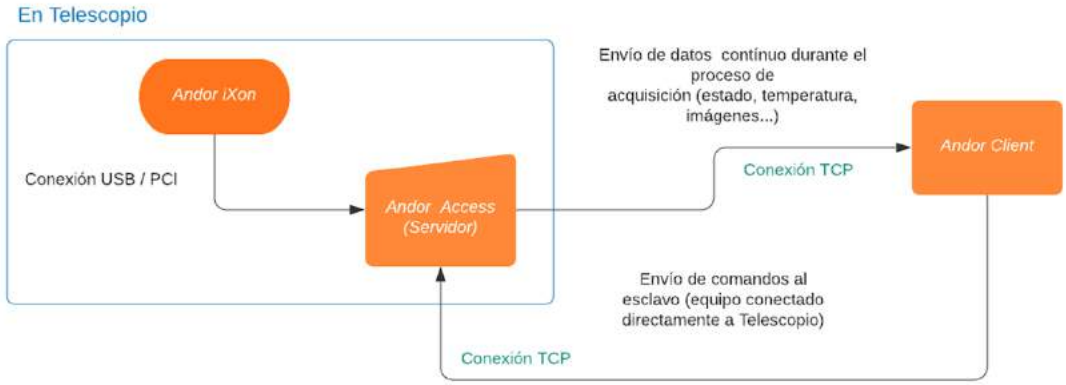


Figura 11.-Diagrama de funcionamiento del sistema Maestro / Esclavo

En la figura 11 se refleja el funcionamiento básico entre equipos. Se ha logrado trabajar, en remoto, con la cámara que funciona en telescopio: desde Cartagena se logra adquirir imágenes de la cámara conectada por PCI a un equipo dentro del Instituto de Astrofísica de Canarias. Esto ha sido posible gracias a la tecnología de redes VPN.

Si el cliente posee acceso a la red interna de la institución, solamente necesita la dirección IP y puerto del equipo que tiene acceso al detector para poder empezar a controlarlo. Esto alberga mayor importancia de la que pueda parecer, ya que capacita al científico de avanzar en su investigación desde su propio hogar, si alberga los permisos requeridos por la institución.

2.1.1.- Conexión VPN

Una VPN (Red Privada Virtual) es un servicio que permite conectarse a la red privada de un lugar aun no estando conectado físicamente a la red interna. Esta tecnología se desarrolló para permitir que usuarios remotos pudieran acceder a las redes internas de las instituciones u empresas.

Para mantener la seguridad, la conexión a la red privada es establecida usando una capa encriptada basada en el protocolo de *tunneling,* donde los usuarios VPN utilizan métodos de autentificación, como contraseñas y certificados, para ganar acceso al VPN.

Se conoce como túnel o *tunneling* a la técnica que consiste en encapsular un protocolo de red sobre otro creando un túnel de información dentro de una red de computadoras. Un canal VPN es creado estableciendo una conexión virtual punto a punto usando el método de *tunneling* sobre redes internas existentes.

El establecimiento de dicho túnel se implementa incluyendo una PDU (unidad de datos de protocolo) determinada dentro de otra PDU con el objetivo de transmitirla desde un extremo al otro del túnel sin que sea necesaria una interpretación intermedia de la PDU encapsulada. De esta manera se encaminan los paquetes de datos sobre nodos intermedios que son incapaces de ver en claro el contenido de dichos paquetes. El túnel queda definido por los puntos extremos y el protocolo de comunicación empleado.

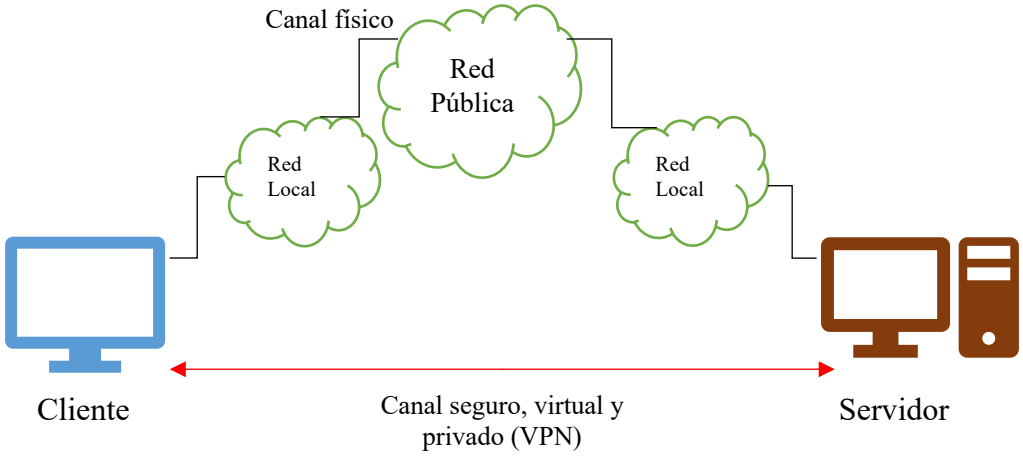


Figura 12.- Diagrama de conexiones para la comunicación entre cliente y servidor

En este caso, se utilizará la propia VPN abierta por el Instituto de Astrofísica de Canarias para poder realizar este trabajo.

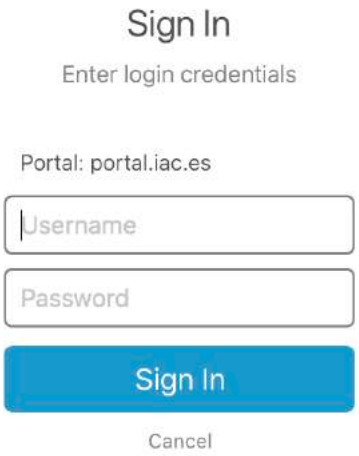


Figura 13.- Portal para la conexión VPN ofrecido por GlobalProtect

Una vez accedido a la VPN con el nombre de usuario y contraseñas concedidas por el servicio de computación de la institución, se podrá establecer la conexión segura con el detector y empezar las comunicaciones y el control del mismo.

2.1.2.-Andor SDK

La propia empresa Andor Technologies ofrece un kit de desarrollado que permite al cliente crear sus propios algoritmos para controlar directamente el detector. Dicho kit abastece de librerías, ficheros de inicialización de los detectores, drivers, ejemplos…todo lo necesario para que el ingeniero tenga una amplia gama de posibilidades de desarrollo de proyectos.

El Kit de Desarrollo de Software (*Software Development Kit*) proporciona al programador acceso a toda la gama de cámaras CCD que tienen disponibles. La parte clave del SDK es el *Dynamic Link Library* (DLL) el cual puede ser usado por una amplia variedad de entornos de desarrollo, incluidos, C++, C, C#, Visual Basic y LabVIEW. Ofrece una amplia variedad de funciones las cuales permiten configurar y adaptar el proceso de adquisición de datos de diferentes formas. Se puede controlar numerosos parámetros de la cámara, como es la temperatura, el tiempo de exposición o la ganancia electrónica.

Para usar el SDK eficientemente, se deben tener en cuenta las siguientes condiciones:

- Configurar apropiadamente el modo de funcionamiento de la CCD
- Hacer un uso eficiente de la memoria computacional
- Crear la interfaz de usuario

En la carpeta de proyecto creada dentro del entorno de desarrollo de Qt, será indispensable que se encuentren los siguientes archivos:

- **Detector.INI:** Es un archivo de texto que proporciona a la cámara de unas órdenes de precompilación para su correcta inicialización. Este archivo se encuentra dentro de las carpetas de instalación de Andor Software.
- **Atmcd32d.dll:** Librería del SDK en Windows.
- **Atmcd32d.h:** Header File a incluir entre los ficheros del programa para poder acceder a las funciones. Contiene todas las declaraciones y macro-definiciones que se comparten entre diferentes archivos fuentes. Esto quiere decir que si, por ejemplo, desde el constructor principal se desea acceder a ciertos parámetros de la cámara, se debe incluir en el encabezado **#include "atcmd32d.h"**

2.1.3.-Servidor

El Servidor es el programa encargado de acceder y controlar la cámara que se encuentra en telescopio. En este apartado se describirá su funcionamiento desde el arranque hasta el modo de operación continuo, detalladamente. Debe de cumplir con las siguientes condiciones:

- Arranque inmediato al iniciarse el sistema operativo
- Al iniciarse debe establecer inmediatamente un servidor TCP
- Debe mantenerse siempre a la escucha de conexiones
- Tiene que ser capaz de comunicarse eficazmente con el conector utilizando las funciones propias dadas por el SDK
- Se deben establecer rutinas de detección de errores a través del modo debug
- Tiene que haber un compromiso entre velocidad de transmisión de datos y de adquisición
- Deben de escogerse los modos de funcionamientos más eficientes sin comprometer grandes cantidades de memoria dinámica y velocidad de ejecución

Grosso modo, podemos en la figura 14 una idea previa de trabajo.

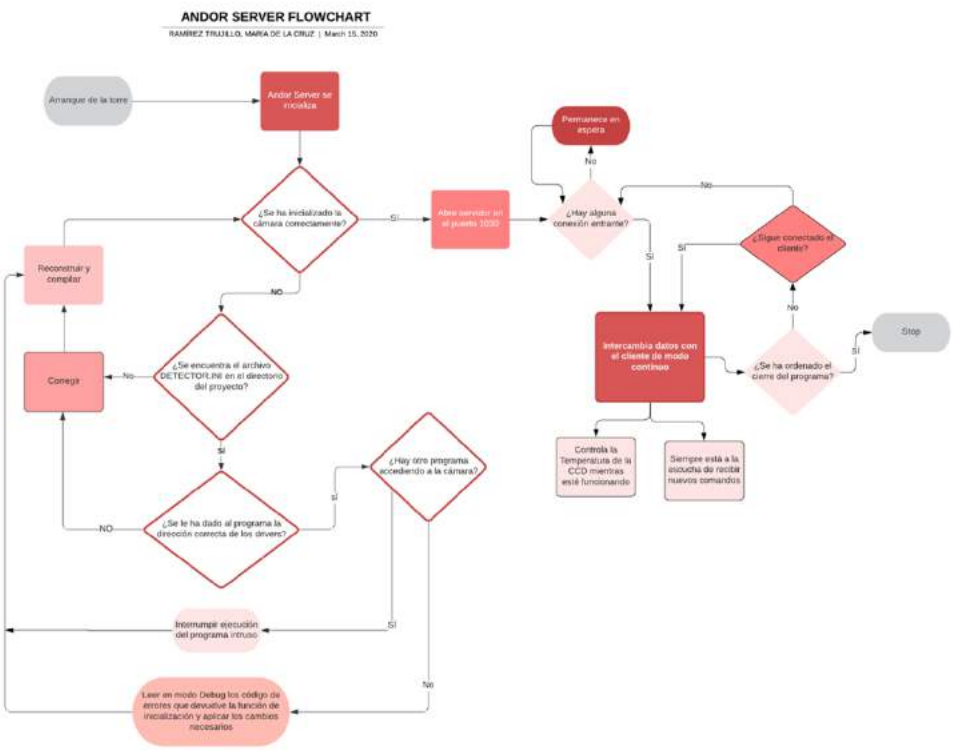


Figura 14.- Flujograma de Andor Server

Al arrancar el programa se inicializa el procedimiento de inicialización de la cámara automáticamente. Para esto, accede a la dirección en memoria del ordenador donde se encuentran los drivers. Aparte, no va a compilar correctamente si no se encuentra el dispositivo de configuración de la cámara llamado DETECTOR.INI.

Una vez detectada, se verá en el programa principal el nombre de esta. Es capaz de trabajar con dos tipos de conexión: USB3 o PCI. Seguidamente inicia el servidor en el puerto 1030, para que el cliente pueda acceder al control remoto del detector. Se queda a la escucha, y ante una conexión válida entrante, empieza a transmitir datos y esperando, a su vez, la llegada de comandos.

Hay que señalar que no puede trabajar con dos detectores a la vez, pero sí detectarlos a la vez antes de arrancar la hebra principal. Entonces se ha añadido la opción de trabajo que, al ser arrancado y antes de iniciar la hebra principal, que si se encuentran dos cámaras detectadas que el propio usuario sea capaz de escoger con cuál trabajar.

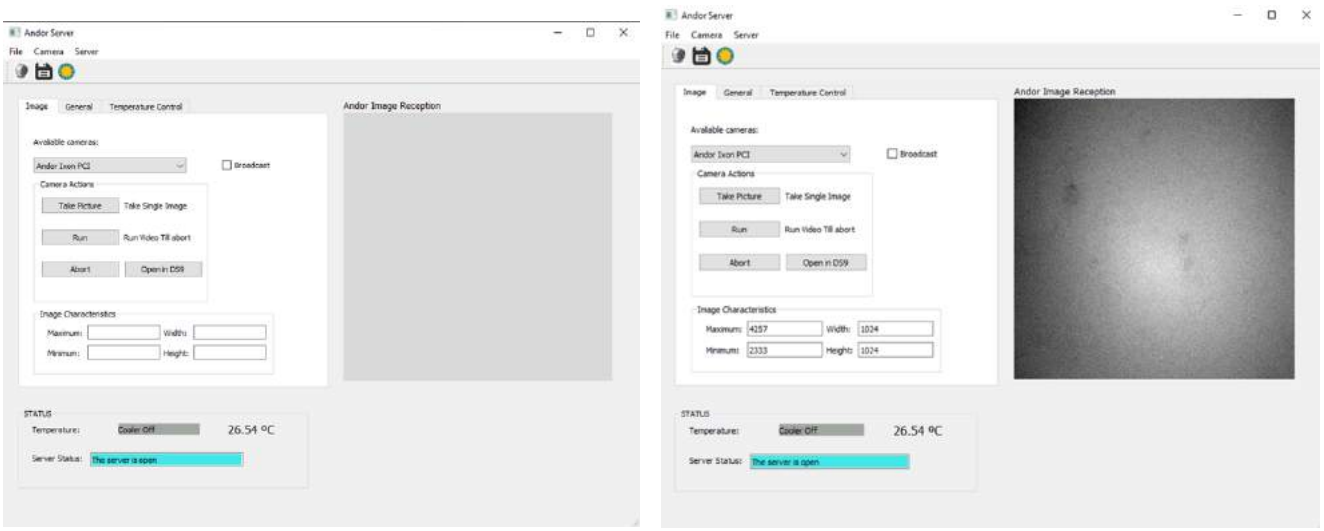


Figura 15.-Pantalla principal de Andor Server

En la figura 15 se aprecia la pantalla principal del programa del servidor. Esta es meramente informativa en modo desarrollador; carece de utilidad cuando se está operando en telescopio en un modo convencional (no se encontrará accesible).

En modo desarrollo sirve para verificar su funcionamiento, así como para certificar con seguridad de que está comportando debidamente. Cuando se realiza la conexión con el cliente, todo lo que ocurra en la pantalla debe de tener su consecuencia en la pantalla del servidor (que tenga que ver con operaciones de adquisición). Por ejemplo, si el cliente ejecuta la acción de tomar imagen, el servidor debe de recoger dicho comando y ejecutar la acción de tomar foto como si se hubiera clicado el correspondiente botón.

Estos eventos son registrados con un temporizador secundario. En Qt hay una clase que se llama QTimer, la cual es capaz de crear temporizadores que, al cabo de un tiempo determinado, ejecuta una función. En este caso lo que se pretende que cada 2ms, el temporizador ejecute la función de escuchar comandos entrantes cuando un cliente se encuentre conectado.

2.1.3.1.-Características y eventos de la pantalla principal

- **Cámaras disponibles:** Señala la cámara con la que el usuario se encuentra trabajando, pero da la realimentación de cuáles se encuentran conectadas.

- **Broadcast:** El servidor entra en modo Broadcast cuando se recibe el comando de adquisición de video del cliente. Esto quiere decir que tan pronto empiece a adquirir imágenes las empezará a mandar por el puerto TCP al cliente. Su máxima velocidad de transmisión es de 7 imágenes por segundo (con exposiciones de 10 ms), lo cual es ligeramente rápido y adecuado para procesos de alineado.

- **Take Picture:** Como su propio nombre indica, es un botón que registra el evento de querer tomar una sola imagen. Esto lo hace preparando el detector de una manera especial para ello, pero se detallará más adelante.

- **Run Till Abort:** Activa el proceso de adquisición de video en la cámara

- **Open in DS9:** Envía al programa de procesado DS9 el fichero .fit creado más recientemente con la cámara tras la captura de la imagen. Solo se utiliza en modo desarrollo, para comprobar que no hay errores en la creación de estos tipos de ficheros. También se ha dejado para poder comparar que no hay errores entre los datos que se mandan al cliente con los que se adquieren directamente en la cámara.

- **Lectura de Temperatura:** Se realiza cíclicamente cada segundo. Si hay un cambio en el dato, se produce el cambio en pantalla. Viene en grados centígrados.

- **Estado del enfriador:** A la vez que se realiza la lectura de la temperatura, se lee el estado del enfriador. Este consta de tres estados: apagado, enfriando y estabilizado. Este dato será enviado, junto con el de temperatura, al cliente.

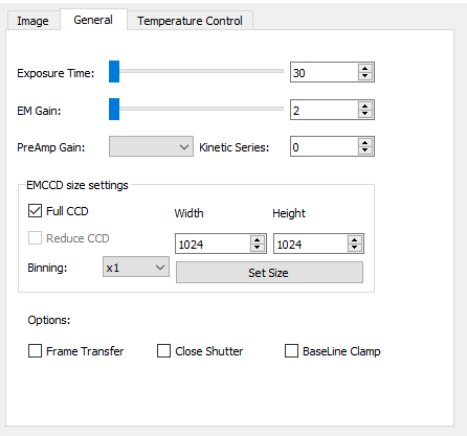


Figura 16.-Pestaña de edición de parámetros para la adquisición

- **Tiempo de Exposición (barra deslizadora):** Permite al usuario alterar, al instante, el tiempo de exposición. Por defecto, a modo de desarrollo, se ha dejado el rango entre 1ms a 30ms, pero se puede cambiar fácilmente dependiendo de las circunstancias. Habrá una opción seleccionable que ejecuta dicho cambio.

  El tiempo de exposición se define como la duración que el detector de la cámara se encuentra expuesto a la luz. La cantidad de luz colectada por la cámara es proporcional al tiempo de exposición; si hay mucha luz ambiental se utilizarán tiempos de exposición muy bajos, pero, si se carece de esta, se deberán utilizar tiempos de exposición más altos, llegando incluso a las horas.

- **Ganancia electrónica**: es el factor que convierte el número de fotoelectrones capturados al realizar una fotografía en el valor numérico que toma cada píxel de tu imagen y que determina su nivel de brillo. Su valor se ajustará dependiendo del brillo aparente del objeto a observar.

- **Variación del tamaño de la imagen a tomar:** se puede variar el tamaño de la imagen a obtener de la cámara. Se puede dejar el tamaño por defecto, que coincide con el tamaño máximo del detector (1024x1024) o disminuirlo.

- **CCD Binning:** Es el proceso de combinar los píxeles vecinos en un detector CCD para formar un "super píxel". Este súper píxel representa el área de todos los píxeles individuales que contribuyen a la carga de fotoelectrones.

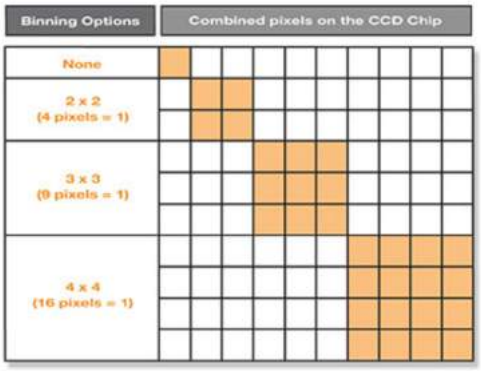


Figura 17.-Funcionamiento del *binning*

- **Frame transfer:** Es un modo de operación del chip que solamente está disponible si tu sistema contiene una *Frame Transfer CCD.* Se puede utilizar para cualquier modo de adquisición.

Una FT CCD difiere de una CCD estándar de dos formas:

1. Primero, una FT CCD contiene dos áreas, de aproximadamente igual tamaño.

   - La primera es el **Área de la imagen**, esta área está por encima y lo más lejos posible del registro de lectura de salida. Este es el área de la CCD sensible a la luz.

   - La segunda es el **Área de almacenamiento**, que se encuentre entre el Área de la imagen y la de los registros de lectura de salida. Esta área se encuentra cubierta por una máscara opaca, típicamente una capa metálica, para insensibilizarla a la luz.

2. La segunda diferencia radica en que las áreas de la imagen y la de almacenamiento son intercambiables independientemente de ambas.

Estas diferencias permiten a una FT CCD ser operada de un modo único donde una imagen puede ser leída mientras la siguiente imagen está siendo adquirida. También permite ser usada sin un obturador.

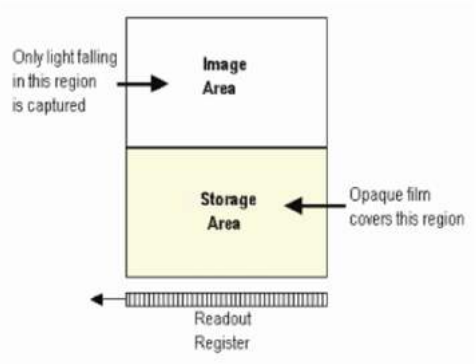


Figura 18.-Funcionamiento interno del Frame Transfer

- **Baseline Clamp:** Se trata de un offset electrónico añadido a la señal de salida del sensor EMCCD para asegurar que nivel enseñado de la señal es siempre un número positivo de cuentas. No se asocia ningún ruido a este valor positivo de cuentas, además, es importante reconocer que no afecta a la sensibilidad. Sin embargo, se debe recordar que hay que substraer el valor baseline offset a la intensidad de la señal cuando se realizan cálculos de ruido/señal.

  Con la función de baseline clamp que se implementa desde el Andor SDK kit, se puede corregir cada imagen con diferentes desviaciones del baseline sustrayendo la media de la señal del bias desde cada píxel de la imagen, luego sumando un valor fijado para asegurar siempre los valores positivos de cuentas.

- **Close Shutter:** Permite abrir (quitar "*check*") o cerrar (*check*) el obturador de la cámara.

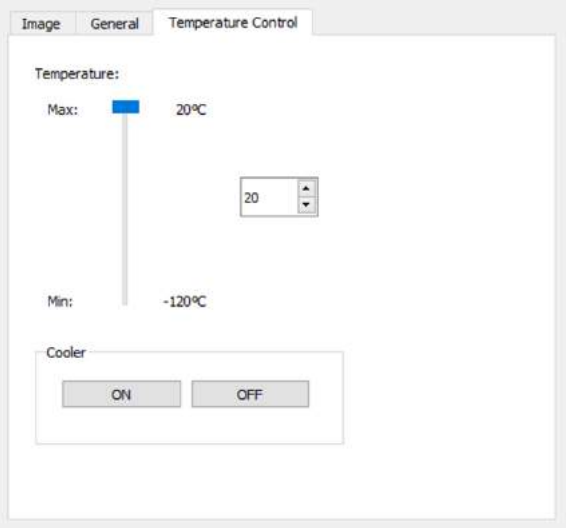


Figura 19.-Pestaña de control de temperatura

- **Control de temperatura:** La temperatura de enfriamiento de la cámara entra en un rango desde los 20ºC (como máximo) hasta un mínimo de -120ºC. El usuario puede modificar la temperatura del detector dependiendo de sus necesidades, con tan solo modificar la barra de selección de temperatura y accionando el enfriador.

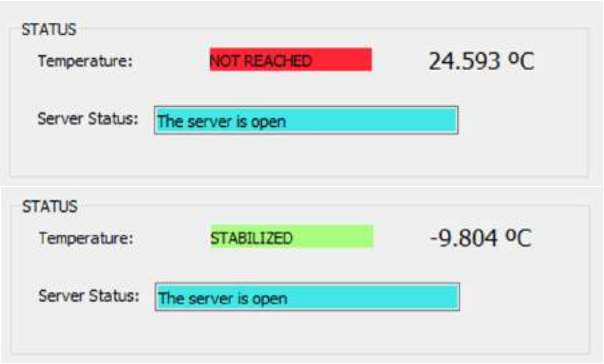


Figura 20.- Cambios de estado de la temperatura en la pantalla del servidor

La cámara Andor incorpora una CCD, la cual es fabricada usando un proceso conocido como "*Multi-Pin Phasing*" (MPP). Como resultado, la corriente de oscuridad es reducida por un factor de, aproximadamente, del 100% comparada con otros dispositivos más estándares bajo mismas condiciones de temperatura.

2.1.3.2.-Eventos y flujogramas

Toma de Imagen y vídeo

Antes de inicializar el algoritmo que ejecuta la acción en cámara de toma de imagen, hay que prepararla. Para esto, se necesitan dos cosas:

- Localizar los drivers a través de la función *initialize*, la cual recibe como argumento la dirección en el equipo donde se encuentran instalados. En el caso de Windows 10, se trata de la siguiente dirección:

  "C:/Windows/System32/drivers"

- Que el fichero de configuración llamado *Detector.ini* se encuentre dentro del proyecto

Una vez que las partes anteriores han funcionado, se empiezan a ejecutar en la hebra principal el proceso de captura de imágenes del detector.

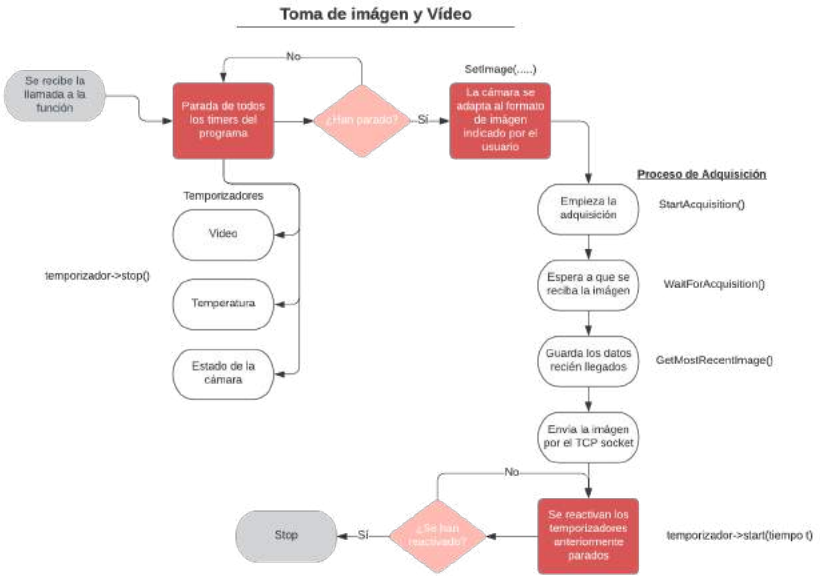


Figura 20.- Flujograma de funcionamiento de la adquisición de imágenes

La diferencia entre el proceso de captura de imagen o de vídeo radica en el uso iterativo de un temporizador. Cuando se trata de una imagen, no se activa el temporizador; ya que al ser solo una única captura solamente debe ejecutarse la función *getImage* una única vez.

Sin embargo, si se trata del proceso de adquisición de vídeo, debemos utilizar un temporizador que llame, dentro de un periodo de tiempo, a esa función. No debe de pararse hasta que el usuario lanza la señal de *AbortAcquisition*: una vez recibida, debe pararse obligatoriamente el temporizador de adquisición de vídeo y reanudarse los de temperatura y envío del estado de la cámara.

Esto se logra haciendo uso de una función que implementa Qt llamada *Connect*. Dicha función se encarga de conectar señales y slots. Por ejemplo, en el caso del temporizador, interesa que cada vez que termina la cuenta de tiempo, se ejecute la señal de *getImage*. Esto se logra escribiendo en el código lo siguiente:

```
connect( temporizador , SIGNAL(timeout()) , this , SLOT(getImage() ) );
```

En definitiva, su función es enlazar eventos. '*this'* hace referencia al objeto donde se encuentra getImage; en el constructor donde se ejecuta el evento.

Procesado de imágenes

En este apartado se va a detallar como se convierte una matriz de bytes a una imagen compuesta en escala de grises.

```
void MainWindow::getImage(){
    temporizadorTemperatura->stop();
    statusTimer->stop();
if(AcquisitionInProcess==true) temporizadorAndor->stop();

    SetAcquisitionMode(1);

   int Width=WidthCCD/binning;
   int Height=HeightCCD/binning;
   int settingImage;
   settingImage=SetImage(binning,binning,initialX,initialX+WidthCCD-1,initialY,initialY+HeightCCD-1);
   if(settingImage!=DRV_SUCCESS){
       qDebug()<<"Error when setting image";
       switch(settingImage){
           case DRV_P1INVALID:
           case DRV_P2INVALID:
              qDebug()<<"------Bad binning parameters";
               break;
           case DRV_P3INVALID:
           case DRV_P4INVALID:
           case DRV_P5INVALID:
           case DRV_P6INVALID:
               qDebug()<<"------Bad subarea";
           break;
           }
   }
```

Figura 21.- Adquisición de imágenes (I)

En esta primera parte de la función se trabaja la preparación de la adquisición. Se paran los temporizadores, por si había procesos anteriores que la cámara no mezcle estados.

Luego, se tiene que comunicar el modo de trabajo con la función *SetAcquisitionMode*. Los modos de trabajo se pueden apreciar en la figura 22.

1	Single Scan
2	Accumulate
3	Kinetics
4	Fast Kinetics
5	Run till abort

Figura 22.- Modos de Adquisición

Como estamos llamando a la función para coger una única imagen, se le indicará que se va a realizar un único escaneo. Posteriormente, se le indica al detector, con detalle, como será la imagen en cuanto a tamaño y *binning* que debe adquirir. Si hay algún error se registra con el *int settingImage*, el cual puede indicar si el error se ha producido por un error de área o de *binning*.

```
    int ui_error;

    ui_error = StartAcquisition();

     ui_error = WaitForAcquisition();


     unsigned long ul_size = Height*Width;
     long * lp_data = new long[ul_size];

     ui_error=GetMostRecentImage(lp_data,ul_size);
     QImage img;
     QVector<long> imgMat;

     for(int i =0; i<ul_size+1;i++){
         imgMat.push_back(lp_data[i]);
     }
    img= paintImage(imgMat,Height,Width);
    QPixmap pixImg(QPixmap::fromImage(img));


    ui->label->setPixmap(pixImg.scaled(ui->label->width(),ui->label->height(),Qt::KeepAspectRatio));

         AbortAcquisition();
         //MUY IMPORTANTE. SI NO SE ABORTA LA ADQUISICION LA CAMARA NO CAMBIARÁ DE ESTADO Y POR LO TANTO NO PODRÁ LEER
         //LA TEMPERATURA CORRECTAMENTE
         if(AcquisitionInProcess==false){

             temporizadorTemperatura->start(1000);
             statusTimer->start(1000);}
         else temporizadorAndor->start((1000/12)*ui->exposureTime_SpinBox->value());

          delete [] lp_data;

}
```

Figura 23.- Adquisición de imágenes (II)

Se llama a la función *GetMostRecentImage* para adquirir la última imagen registrada por el detector, almacenando los valores en una matriz de tipo *long*. Se puede observar que se almacena en un vector tipo *long*. Esto se hace para facilitar luego el envío posterior de imágenes, como se detallará en la función *paintImage*.

Posteriormente, dependiendo de si se ha llamado a la función desde el proceso de adquisición de vídeo (si *Acquisition in process==true*) se reanudará los temporizadores de temperatura, o de lo contrario, se reanudará el proceso de adquisición de vídeo.

```
QImage MainWindow::paintImage(QVector<long> lp_data, int Height,int Width)
{
     QImage img(Width,Height,QImage::Format_RGB32);

    //  ui_error = GetBitDepth(0, &i_depth);
      unsigned long ul_size;
      ul_size=Width*Height;
      long max=-1000000,min=100000;
   // long maxArr=-1000000,minArr=100000;
      int pixels=Width*Height;
      dataSize = new unsigned long;
      dataSize=&ul_size;
      dataArr= new long[ul_size];

      for(int i=0;i<pixels;i++){
          dataArr[i]=lp_data[i];
      }
    /*  for(int i=0;i<pixels;i++){
          if(dataArr[i]>maxArr)maxArr=dataArr[i];
          if(dataArr[i]<minArr)minArr=dataArr[i];
      }*/

           for(int i=0;i<pixels;i++){
               if(lp_data[i]>max)max=lp_data[i];
               if(lp_data[i]<min)min=lp_data[i];
           }

           ui->lineEdit_Max->setText(QString::number(max));
           ui->lineEdit_Min->setText(QString::number(min));
           ui->lineEdit_H->setText(QString::number(HeightCCD));
           ui->lineEdit_w->setText(QString::number(WidthCCD));

          long difference=max-min;

          for(int j=0;j<Height;j++){

              int J=j*Width;
              long color;

          for(int i=0;i<Width;i++){
                      if(difference>0)
                        color=(lp_data[J+i]-min)*255/difference;
                      else color=0;

                      uint k=QColor(color,color,color).rgb();

                      img.setPixel(i,j,k);

                  }
              }

            lp_data.clear();
          return img;

}
```

Figura 24.- Proceso de conversión de matriz a imagen

Primero se crea un elemento tipo imagen, que servirá como plantilla para la transformación de matriz de bytes a imagen. Si se dibujara ahora en pantalla solo veríamos una imagen en negro. Para poder escalar las cuentas de la matriz primero tendremos que extraer el máximo y el mínimo valor. Por condiciones de ruido, se han establecido unos valores altos de comparación. La matriz se copia a un puntero llamado *dataArr*, que será el encargado de guardar la matriz para enviarla por el socket.

Luego, para poder escalar de 16 bits a 8 bits, se necesitará saber cual es el máximo valor de pixel y el mínimo en la imagen. Esto se consigue con la función dentro del bucle for, donde se va extrayendo el color de cada píxel. Se pinta posteriormente en el fichero *img* obteniéndose, finalmente, una imagen coloreada de los datos adquiridos con la cámara.

Este fichero *img* será enseñado en pantalla posteriormente.

La función *getImage* hace un pre-set de la cámara que es distinto al que tendríamos que adaptar en caso de querer adquirir una señal de vídeo.

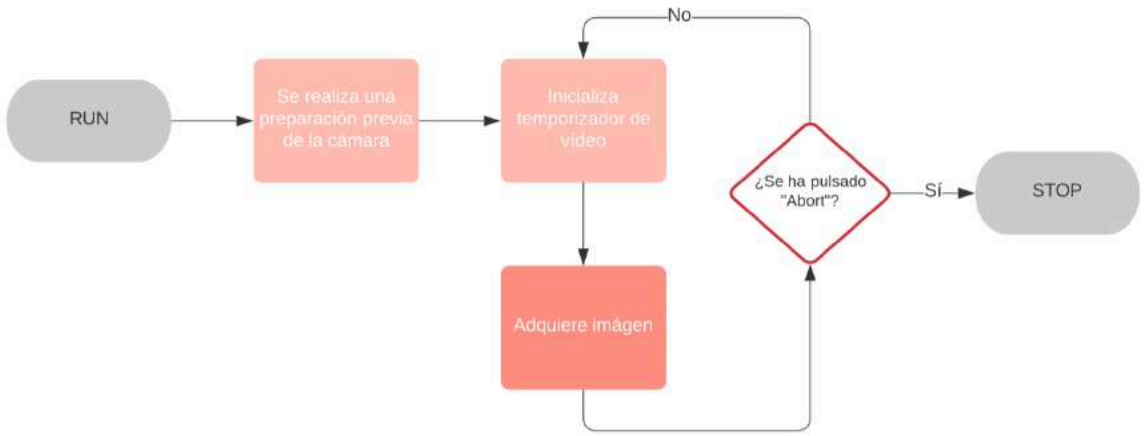


Figura 25.- Flujograma de adquisición de vídeo

Lo que se muestra en la anterior figura es el funcionamiento básico de la operación de vídeo. Cuando se recibe el evento de adquirir vídeo, se prepara inmediatamente la cámara para ello.

```
void MainWindow::on_RunVideo_clicked()

{   AcquisitionInProcess=true;

  SetAcquisitionMode(5);

  int vel;
  float speed;
  int aDChannel=0;
  SetADChannel(aDChannel);
   temporizadorTemperatura->stop();
   //statusTimer->stop();
  GetNumberHSSpeeds(aDChannel,0,&vel);
  int canal=0;
  canal=vel-1;
  GetFastestRecommendedVSSpeed(&vel,&speed);
  qDebug()<<"Vel [msec per pixel]: "<<vel<<" Vertical speed: "<<speed;
  SetVSSpeed(vel);
  for(int i=0;i<vel;i++){
  GetHSSpeed(aDChannel,0,i,&speed);
  qDebug()<<QString::number(i)<<" "<<QString::number(speed)<<" MHz";
  //consolaAndor->append(QString::number(i)+"  "+QString::number(speed)+" MHz");
  }
  //SetADChannel(2);
  SetShutter(1,ShutterMode,50,50);

  int binning,er;
  binning=1;
  er=SetImage(1,1,1,1024,1,1024);
  qDebug()<<"Er setting image: "<<er;

SetNumberKinetics(50);
   statusTimer->stop();
 temporizadorAndor->start((1000/12)*(ui->ExposureSlider->value()));

}
```

Figura 26.- Preparación del detector para la adquisición de señal de vídeo

En primer lugar, hay que indicarle al detector el modo de funcionamiento. En *SetAcquisitionMode* le indicamos con el número cinco que vamos a trabajar en modo "*Run Till Abort*", especialmente diseñado para la adquisición de vídeo. Este modo realiza escaneos continuos de la CCD a la velocidad programada por el usuario, hasta que la adquisición es interrumpida llamando a la función *AbortAcquisition*. En SetNumberKinetics le indicamos a la cámara las acumulaciones por serie. Una vez detallados el canal de vídeo, la velocidad de adquisición, el modo de imagen y obturador se procede a llamar al temporizador de adquisición de imagen. Este, dependiendo del tiempo de exposición, comprobará la adquisición conforme llegue el dato de la cámara. Llamará a la función *getImage*, y se realizará el proceso descrito anteriormente.

Servidores TCP

Esta es la funcionalidad más importante del Programa "Andor Server". Primero hay que abrir dos servidores: uno para recibir comandos y enviar las imágenes y otro para el estado de la temperatura y la cámara. Qt creator dispone de las librerías necesarias para ello, pero tienen la diferencia que estas deben declararse en el fichero de proyecto para que las incluya, tal que:

```
1 QT       += core gui network multimedia
2
3 greaterThan(QT_MAJOR_VERSION, 4): QT += widgets
4
5 CONFIG += c++11
6
```

Figura 27.- Edición de módulos en el fichero .pro

Habrá que reconstruir el programa para que se actualicen los cambios una vez añadidos los nuevos módulos. Una vez validado, se tendrán a disposición todas las librerías relacionadas con "*Network*", tales como *QTcpServer, QTcpSocket…*

Para poder realizar el servidor se necesitará crear una nueva clase, con un constructor y un fichero de encabezado independiente del *MainWindow* (fichero de la ventana principal). Esto va a ser de gran utilidad para crear los dos servidores necesarios para este programa. Como norma general, se compondrán de la siguiente forma:

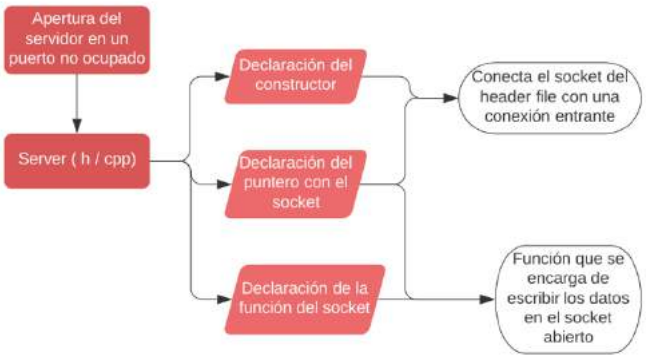


Figura 28.- Flujograma de la clase "Server"

Para ayudar a la comprensión del lector para saber como funciona el servidor se empezará por describir el más sencillo de los casos: el servidor de estado de la cámara. Posteriormente se detallará el servidor principal.

Servidor de Estado de la Cámara

- Header file

```
class statusServerAndor : public QTcpServer
{
public:
    statusServerAndor();
      explicit statusServerAndor(QObject *parent = 0);
    void sendString(QByteArray state);
    QTcpSocket *m_socket;
    bool socketConnected = false;
};
```

Figura 29.- Declaración del header file

Como se puede comprobar, si se declara una nueva variable de la clase statusServerAndor va a tener como atributos la creación de un socket que se enlazará al puerto abierto del servidor, una variable tipo bool que ayudará a que no se ejecuten funciones externas si el puerto no está debidamente abierto, y la función por la cual se escribirán los datos.

```
if(!myStatusServer->listen(QHostAddress::Any,1031)){
    qDebug()<<"Error...";
}else{

qDebug()<<"status port ok";
}
```

Figura 30.- Apertura del servidor desde el constructor principal

- Constructor

```
#include "statusserverandor.h"

statusServerAndor::statusServerAndor(QObject *parent)
                                    :QTcpServer(parent)
{
    m_socket = nullptr;

    connect(this,&statusServerAndor::newConnection,[&](){
        m_socket = nextPendingConnection();
        socketConnected = true;
    //text: m_socket = nextPendingConnection();
    });

}

void statusServerAndor::sendString(QByteArray state)
{
    if(m_socket->isWritable()){
       // qDebug()<<"..";
        m_socket->write(state);
    }
}
```

Figura 31.- Constructor del servidor

Una vez abierto el servidor, el constructor habilita el socket para el intercambio de datos, señalándolo con la variable booleana "*socketConnected=true*". Ahora bien, desde la hebra principal se ha declarado un temporizador para que, cada segundo, si está abierto el programa remoto, se envíe el estado de temperatura, tal que:

```
statusTimer = new QTimer;
connect(statusTimer,SIGNAL(timeout()),this,SLOT(SendStatus()));
statusTimer->start(1000);
```

Figura 32.- Temporizador del servidor

Y, a su vez, la función *SendStatus* se encargará de recolectar los datos necesarios de la cámara y los enviará a través del servidor.

```
void MainWindow::SendStatus()
{
    if(myStatusServer->socketConnected){

        float temp=0;

            int coolerState = GetTemperatureF(&temp);
          ;
            QString tempText = QString::number(temp);

            if(coolerState==DRV_TEMP_OFF){


               tempText.push_front("O");
            }
            if(coolerState==DRV_TEMP_STABILIZED){
                tempText.push_front("S");
            }
            if(coolerState==DRV_TEMP_NOT_REACHED){
              tempText.push_front("N");
            }


            QByteArray com = tempText.toLatin1();
        myStatusServer->sendString(com);
            }

}
```

Figura 33.-Envío de estado

Cuando el reloj da la señal de timeout cada segundo, esta función comprueba, primeramente, que el puerto para mandar el estado tiene un cliente conectado. Si esto es correcto, recoge el valor de la temperatura en grados centígrados en una variable tipo float. Luego, se necesita recoger el estado de la cámara para conocer si el enfriador está funcionando, si está apagado o si la temperatura se encuentra estabilizada. Bastará con enviar un carácter identificativo para cada estado.

Por último, se accede a la función que se encarga de escribir los datos en el socket convirtiendo la cadena tipo string en una de bytes. Llega con una velocidad casi instantánea al cliente, pese a la diferencia de situación geográfica.

La cadena de estado estará compuesta como "***S-C-T***" (sin los guiones), siendo cada carácter:

- **Size:** Se ha habilitado dos tamaños por defecto de adquisición de imagen; 512x512 o 1024x1024. En el primer caso, se añadirá un 5 como primer carácter de la cadena de estado, y en el segundo caso, un 1.

- **Camera:** Se podrá hacer distinción entre cooler off (con el carácter "*O*"), enfriando (con el carácter "*N*") o temperatura estabilizada (con el carácter "*S*").

- **Temperature:** El resto de la cadena será enteramente para el número de la temperatura en *float*.

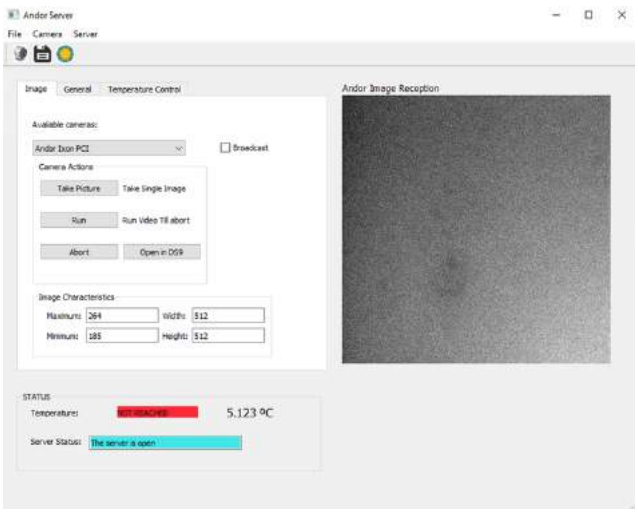


Figura 33.- Ejemplo en la pantalla principal

Por ejemplo, en el caso de la figura 33 la cadena de estado que enviaría el servidor sería "5N5.123", queriendo decir que el tamaño de la imagen es de 512x512, que la temperatura no está estabilizada y que la temperatura medida es de 5.123ºC.

Servidor Principal

En este apartado se explicará las dos funciones del servidor principal: la escucha de comandos y el envío de imágenes. No se detallará, sin embargo, la forma del constructor ni del fichero de encabezado, ya que son iguales al anteriormente explicado.

El modo de envío de datos para las imágenes ya no se hace a través del *array* de bytes, sino de otra variable llamada *QTextStream*, y puede funcionar tanto como un *QString* o como un *QByteArray*. Esto se asemeja al empleo del operador de salida de *strings* llamado *cout* en C++. Esto hace que no se tenga que emplear un bucle for que acceda a cada elemento de la matriz de datos de la imagen para proceder a su envío. Esto ha logrado que la fluidez de envío de imágenes sea notoriamente muchísimo mayor a utilizar bucles iterados.

```
void server::enviaImagen(long *dataAr, unsigned long dl_size)
{
QStringList arrayString;

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


     arrayString.push_back(QString::number(dataAr[i]));

  }
  arrayString.push_back("f");


if(m_socket->isWritable()){
    QTextStream T(m_socket);

    QString dataList = arrayString.join('.');
       T<<dataList;


       m_socket->flush();

   }
}
```

Figura 34.- Codificación de las imágenes

Cada imagen está compuesta por una matriz que almacena valores tipo *long*. Debido a que la cámara recoge valores de intensidad de imagen de hasta 16 bits, es más conveniente utilizar este tipo de variable (sino obligatorio). Se necesitará pasar esa matriz a una variable tipo String antes de poder enviarla a través del socket. El procedimiento que se ha seguido es pasar, primero, la matriz de tipo long a una de tipo StringList. La diferencia es que ahora en vez de ser una matriz de *MxN*, siendo M el ancho de la imagen y siendo N el alto de esta. StringList devolverá un vector. Solo serán válidas las imágenes cuadradas, es decir, de M=N. En la variable dl_size se encontrará almacenado el valor del área de la imagen.

Luego, se puede observar que al final de este vector se añade un carácter que señaliza el final de este. Esto es de gran importancia: para poder descodificar la cadena entrante de imágenes desde el punto de vista del cliente se necesita saber cuando termina cada una, sobre todo cuando está en modo vídeo. Así se garantiza que ha llegado completamente la imagen sin pérdida de datos.

Como cada elemento de la matriz puede tener un tamaño distinto, se agrega un punto para diferenciar cada píxel dentro de la cadena String, ya que al pasar de StringList a List se pierde el dato de vector y se solaparían los números entre sí. Este tipo de codificación ha ayudado tanto a que el tamaño de la imagen a como el valor de cada píxel puedan variar sin romper la ejecución del programa: es capaz de adaptarse perfectamente a diferencias de tamaños. Por ejemplo, es capaz de enviar tanto una imagen de 512x512 como una de 1024x1024 con valores registrados de 16 bits.

Para la escucha de peticiones del cliente se ha implementado otro temporizador en la hebra principal. Se ejecuta cada 100ms una vez abierto el servidor principal.

```
mySocketTimer = new QTimer(this);
connect(mySocketTimer,SIGNAL(timeout()),this,SLOT(getRead()));
mySocketTimer->start(100);
```

Figura 35.- Temporizador para la adquisición de comandos

Cada vez que emite la señal de término de tiempo, se activa la función de getRead(), encargada de comprobar en el TcpSocket del Servidor principal si se ha enviado peticiones desde el cliente. Esto no colapsa el socket, ya que si, por ejemplo, el cliente ha activado la señal de vídeo pero quiere cambiar la temperatura, no se va a realizar la acción hasta que no finalice la adquisición, por las propias propiedades de funcionamiento del mismo detector.

Es por ello que el funcionamiento de este servidor es estructurado y, ya que se sigue esta jerarquía de trabajo, no se da el colapso. Es también para evitarlo el que se ha construido un servidor diferente para el envío reiterado de información del estado del detector. Pero como se ha descrito anteriormente, si está la cámara en proceso de adquisición no va a poder enviar mientras tanto ni estado del detector ni se podrá variar la temperatura, por el propio funcionamiento de la máquina.

La función getRead comprueba que se ha escrito en el socket desde el cliente y es capaz de distinguir los comandos entrantes. Estos son:

- **T*n* :** Siendo n un valor entero, este comando registra el set de temperatura. Acepta tanto valores positivos como negativos. Cuando llega este comando la función elimina el carácter T y envía este valor a la barra de selección de temperatura, afectando de inmediato a la temperatura objetivo del enfriador.

- ***CoolerOn/CoolerOff***: Llama a la acción de encender o apagar el enfriador según el comando entrante.

- **GetImage:** Solo captura una imagen de la cámara y la envía, en cuanto se completa la adquisición, como respuesta por el socket.

- **Run/Abort:** Empieza la adquisición de vídeo como serie de imágenes si el comando es “Run”, y aborta la adquisición si el comando recibido es “Abort”. Cada vez que una nueva imagen es tomada se envía de inmediato por el socket, como es el caso de GetImage, pero este envío es continuo hasta que no se aborte el proceso.

- **EXP/GAIN**n: Siendo n un valor entero, este comando registra el valor entrante de exposición que se ha alterado desde el cliente. Llama como consecuencia a la barra del mismo, y al afectar el valor del mismo se actualiza el tiempo de exposición del detector. El mismo caso se aplica para la ganancia.

- **No/FrameTransfer, Open/CloseShutter, No/BaseLine :** Estos comandos se encargan de registrar si el cliente desea aplicar las propiedades que los mismos comandos indican. Tienen efecto inmediato en las “*checkbox*” declaradas en la interfaz, con consecuencia directa en el detector.

2.1.4.-Cliente

El modo cliente del software desarrollado en este trabajo va a ser el programa con el que trabaje el astrónomo directamente para controlar el detector. Se ha desarrollado para que, cuando el trabajador esté conectado debidamente a la red interna del Instituto de Astrofísica de Canarias, ya sea desde su casa a través de una cuenta VPN o estando en la sala de control del telescopio, sea capaz de conectarse al servidor y controlar en su totalidad el equipo de adquisición.

A modo de resumen, sus características principales se compondrán de:

- Conexión al servidor de imágenes de la cámara Andor, mediante IP y Puerto. Automáticamente se conecta al puerto de Andor Status, ya que se ha diseñado para que este se abra en una unidad superior al puerto general abierto. Por ejemplo, el puerto principal se declara en 1030, y se abre por defecto el secundario en el 1031.

- Relación de eventos cliente-servidor mediante comandos

- Hebra secundaria que durante la adquisición de imágenes va haciendo cubos de ficheros fits. Estos cubos contienen 20 imágenes de 1024x1024 cada uno

- Hebra secundaria para el procesado en *shift and add (opcional)*

2.1.4.1.-Sistema de Pantallas

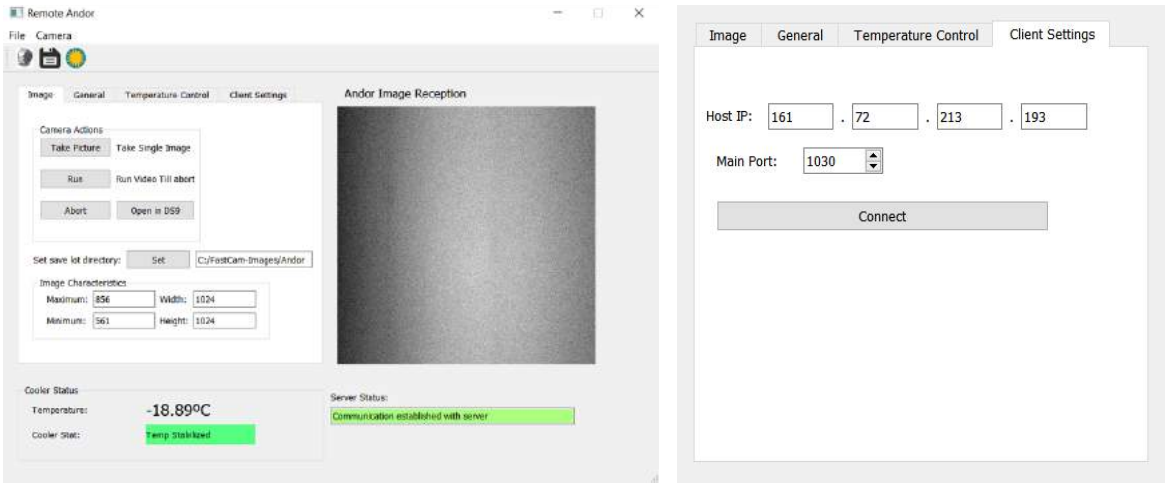


Figura 36.- Pantalla principal del cliente (izquierda), junto con la pestaña de client settings (derecha)

En la figura 36 se muestra la pantalla principal del software a remoto. Se ha diseñado con la idea que se asemeje lo máximo posible a la GUI diseñada para el servidor. Las pestañas de "*General*" y "*Temperature Control*" son exactamente iguales a las mostradas anteriormente. Solamente cambia que ahora nos encontramos con una nueva pestaña: "*Client Settings*", donde se pueden alterar los parámetros para conectarse al servidor.

Funcionamiento

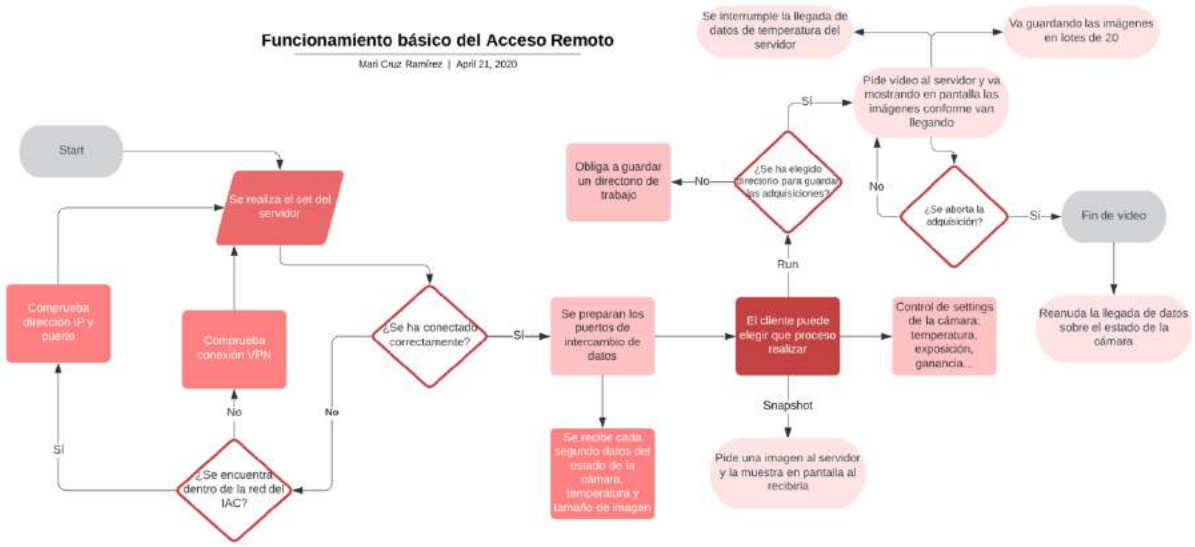


Figura 37.- Flujograma del acceso remoto

Al abrir el sistema remoto la pantalla principal encuentra todo desconectado, tal como se refleja en la figura 38.

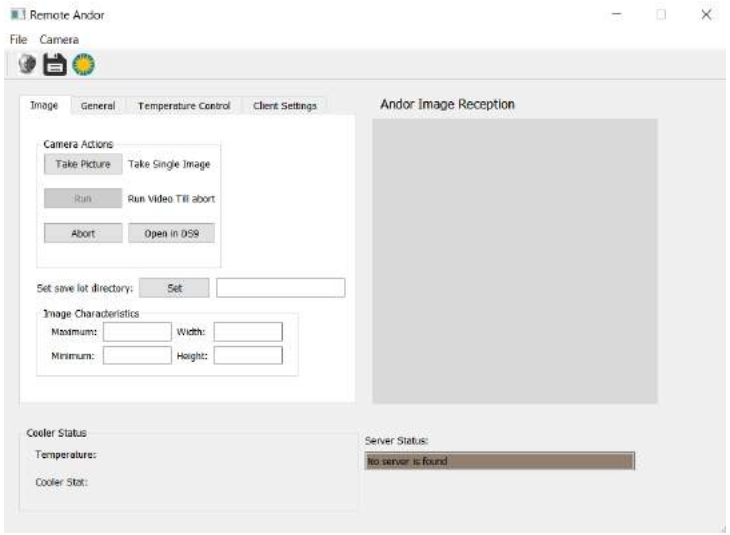


Figura 38.- Acceso remoto sin conexión previa al servidor

Como se aprecia no se refleja aún ningún dato en pantalla. Esto no va a cambiar hasta que no se conecte adecuadamente al servidor. Una vez ajustados los parámetros de IP y puerto en *Client Settings*, empieza a recibirse de inmediato los datos de temperatura y estado de la cámara.

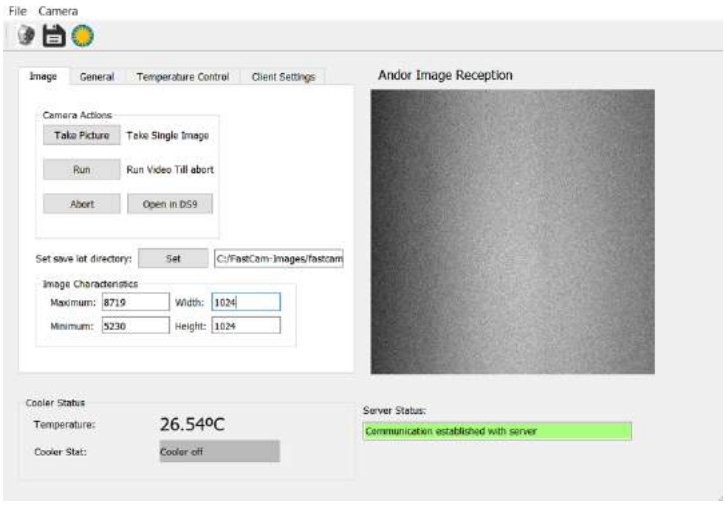


Figura 39.- Comunicación establecida con el servidor

Si se aprecia el detalle que el botón de "Run" ha cambiado a estar disponible de una imagen a otra, esto ha ocurrido gracias a que se ha señalado un directorio donde guardar los lotes de imágenes durante el proceso de adquisición.

Luego, tomar una imagen o vídeo se realiza de la misma forma como si se estuviera utilizando el software de servidor en laboratorio. La velocidad máxima de adquisición de imágenes en vídeo depende del tiempo de exposición: el socket podrá enviar sin dificultad videos de entre 7-9 fps (una exposición de 6-10 ms).

Posteriormente, el software irá midiendo el número de imágenes que se van adquiriendo, acumulándolas en una memoria temporal del programa. Cuando se llega a un cúmulo de 20 imágenes, guarda en un único fichero .FITS el lote. El porqué de 20 como máximo radica en la capacidad de la librería CFITSIO: se ha comprobado, experimentalmente, que lotes superiores a 20 para el máximo tamaño de imagen se malograban. Si se trabajase, por ejemplo, con un tamaño de 128x128, se puede llegar a grabar hasta 1000 imágenes.

Hay un detalle en el que difiere del servidor: la capacidad de abrir los *snapshots* directamente en un programa profesional de procesado, como es el DS9. Esto se ha añadido para comprobar la validez de las imágenes, como se explicará a continuación.

Archivos .FITS

Los archivos tipo "*Flexible Image Transport System*" (*FITS*) es el formato estándar de datos utilizado en astronomía. FITS es a menudo utilizado no solo para transportar imágenes, sino también desde espectros electromagnéticos hasta catálogos de datos. Un fichero FITS puede contener varias extensiones, cada una de ellas variando en dato y en objeto.

La mayor ventaja que supone la utilización de este formato es que la información de las cabeceras es legible en ASCII, de modo que un usuario puede examinar dichas cabeceras para obtener información desde el lugar donde se adquirió la imagen, el nombre del astrónomo y datos de filtros utilizados y mucho más.

Desde dichas cabeceras se podrá verificar que la imagen se ha guardado correctamente. Se ha podido utilizar este formato utilizando la librería que dispone gratuitamente la NASA, llamada CFITSIO. Es una librería multiplataforma de rutinas para leer y escribir datos en el formato .FITS. Está escrita en ANSI C y proporciona una poderosa y simple interfaz para acceder a estos archivos.

2.1.4.2.-Rutinas y eventos

Los eventos desde el software de acceso remoto son más sencillos de explicar, ya que al no tener que acceder directamente al detector se limita al envío de comandos y a recibir datos del servidor. Los eventos principales serán, por lo tanto: recibimiento de estado de la cámara, recibimiento de imágenes (ya sean snapshots o vídeo), el envío de comandos y el guardado de imágenes en la extensión .FITS.

Conexión al servidor

La conexión al servidor va a ser relativamente sencilla. Desde el cliente no se tendrá que crear una clase a parte del *mainwindow,* como ocurría en el caso del servidor. Bastará con punteros de la clase QTcpSocket para cada server port (uno para el principal y otro para el del estado de la cámara).

Este evento es accionado cuando se pulsa el botón de "*Connect*" desde la pestaña "*Client Settings*".

```
void MainWindow::on_connectToHostButton_clicked()
{

        QString HostIp = ui->ip->text()+"."+ui->ip_2->text()+"."+ui->ip_3->text()+"."+ui->ip_4->text();
        int HostPort = ui->CLientPortspinBox->value();

        mySocket->connectToHost(HostIp,HostPort);
        statusSocket->connectToHost(HostIp,HostPort+1);
        if(mySocket->waitForConnected(3000)){

            qDebug() <<"Connected!";
            dataArrival = new QString;
           //| datAr = new QVector<QString>;
            //sizeData = new unsigned long;
            QMessageBox::information(this,"Connection Information","Communication established with server succesfully");
            count=0;
            ui->serverStatusLineEdit->clear();
            ui->serverStatusLineEdit->setText("Communication established with server");
            ui->serverStatusLineEdit->setStyleSheet("QLineEdit{background: rgb(170, 255, 127);}");

        }else{
            qDebug() << "Not Connected,sorry ;c";

        }
        if(statusSocket->waitForConnected(3000)){
            qDebug()<<"status socket connected";
        }else{qDebug()<<"status socket failed to connect to host";}

}
```

Figura 40.- Conexión con los servidores

Primero, se realiza la conversión a una sola cadena el conjunto de parámetros de la IP que el usuario a dispuesto.

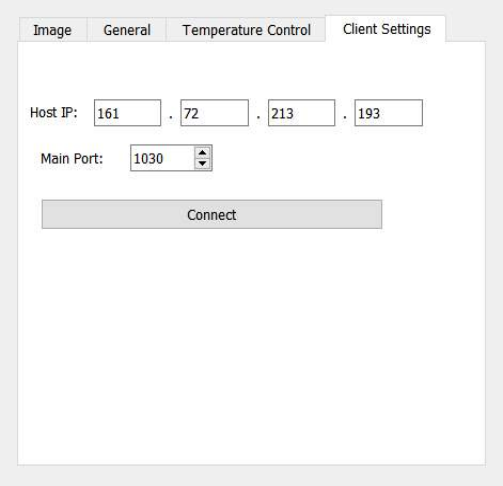


Figura 41.- Pestaña de client settings

Se guarda el valor del Main Port en un entero llamado HostPort. Como se ha programado en el servidor que el puerto secundario (el del estado de la cámara) sea una unidad mayor de este valor, se conectará automáticamente, en este caso, el puerto del estado al 1031. Como es lógico, aunque sean puertos distintos, compartirán la misma IP.

Si la conexión se realiza correctamente usando la función *ConnectToHost*, el principal comunicará por pantalla que la conexión con el servidor se ha realizado correctamente, y el secundario lo comunicará por el compilador del modo debug.

```
connect(mySocket,&QTcpSocket::readyRead,[&](){
    showRecievedImage();
});
connect(statusSocket,&QTcpSocket::readyRead,[&](){
    getState();
});
```

Figura 42.- Conexión a los servidores

Se conectará cada socket a su correspondiente función cuando detecten datos entrantes.

Estado del detector

```
void MainWindow::getState()
{
    QByteArray message = statusSocket->readAll();
    QString command;
    command.push_back(message);
if(command.startsWith("5")){

    HeightCCD=512;
    WidthCCD=512;
    ui->lineEdit_H->setText("512");
     ui->lineEdit_w->setText("512");
}
else if(command.startsWith("1")){
    HeightCCD=1024;
    WidthCCD=1024;
    command.remove(0,1);
    ui->lineEdit_H->setText("1024");
     ui->lineEdit_w->setText("1024");
}
if(command.startsWith("O")){
 command.remove(0,1);
 command=command+"°C";
ui->tempState->setText(command);
ui->CoolerStat->setText("Cooler off");
ui->CoolerStat->setStyleSheet("QLabel{background: rgb(186, 186, 186); }");
//Pinta en Gris
}

else if(command.startsWith("N")){
    command.remove(0,1);
    command=command+"°C";
   ui->tempState->setText(command);
   ui->CoolerStat->setText("Temp Not Reached");
   ui->CoolerStat->setStyleSheet("QLabel{background: rgb(255, 41, 26); }");
   //Pinta en Rojo
   }

else if(command.startsWith("S")){
    command.remove(0,1);
    command=command+"°C";
   ui->tempState->setText(command);
   ui->CoolerStat->setText("Temp Stabilized");
   ui->CoolerStat->setStyleSheet("QLabel{background: rgb(85, 255, 127); }");
   //Pinta en verde
   }
}
```

Figura 43.- Descodificación estado del detector

Sabiendo qué significa el orden de los caracteres entrantes, solo habrá que identificar de cuales se tratan, como se puede averiguar en la función. Recordando lo que se explicó en el servidor, tenemos que esta cadena se compone de la siguiente forma:

La cadena de estado estará compuesta como "S-C-T" (sin los guiones), siendo cada carácter:

- *Size: Se ha habilitado dos tamaños por defecto de adquisición de imagen; 512x512 o 1024x1024. Esto se debe a que son los tamaños estándares de los detectores con los que se va a trabajar. En el primer caso, se añadirá un 5 como primer carácter de la cadena de estado, y en el segundo caso, un 1.*

- *Camera: Se podrá hacer distinción entre cooler off (con el carácter "O"), enfriando (con el carácter "N") o temperatura estabilizada (con el carácter "S").*

- *Temperature: El resto de la cadena será enteramente para el número de la temperatura en float.*

Lo que va haciendo el algoritmo es leer un carácter desde el principio de la cadena, identificarlo, y borrarlo, para leer el siguiente. De esta manera se hace la identificación de una manera mucho más eficaz y rápida.

Entrada de imágenes

```
void MainWindow::showRecievedImage()
{

    QByteArray resp;
    resp=mySocket->readAll();

if(!resp.endsWith(".f"))dataArrival->push_back(resp);
  else if(resp.endsWith(".f")){


  dataArrival->push_back(resp);


  list = new QStringList;


 *list= dataArrival->split('.',QString::SkipEmptyParts);
  list->removeAt(list->length());
  delete dataArrival;

  getImage(*list);
  delete list;


}
}
```

Figura 44.- Descodificación de imágenes

La matriz de datos de la imagen se ha codificado de la siguiente forma:

$$X_0.X_1.X_2.\ldots X_{height\cdot Width}.f$$

Siendo X_i cada pixel de la imagen codificado en caracteres tipo string. Entonces, cuando la función lea que se ha recibido el carácter "f" que señaliza el final de la cadena, se romperá la misma en los puntos con la función *Split*, convirtiendo la cadena en una matriz. Esta matriz se guardará en el puntero *list*.

Se deben deslocalizar los punteros para no abusar de la memoria del programa: esto es de gran importancia, ya que se puede ver interrumpido si la memoria se agota. Luego, se llamará a la función getImage, encargada de convertir la matriz en una imagen que mostrar en pantalla, con la misma función explicada en el apartado del servidor.

Guardado de ficheros .FITS

Para realizar el guardado simultáneo de ficheros .FITS en paralelo con el funcionamiento de la hebra principal se abrirá una secundaria, liberando así el procesador. Si se realizara desde la hebra principal se podría colapsar el funcionamiento del programa. Para esto, se deberá crear una nueva clase, que se llamará *Thread.* Esta se declarará desde un fichero de extensión *header file*, tal que:

```
#ifndef THREAD_H
#define THREAD_H
#include <QThread>
#include <QString>
#include <QDebug>
#include <QMutex>
#include <QMutexLocker>
#include <QVector>
class Thread : public QThread
{
public:
    Thread(QVector<QVector<long>> Arr,QString myDir);
    void stopThread();
    bool stopped=false;
protected:
    void run();

private:

QMutex mutex;
int count=0;
QVector<QVector<long>> myArr;
QString Dir;


};

#endif // THREAD_H
```

Figura 46.-Header file para la hebra

Para crear la hebra de guardado de imágenes se necesitará pasar como argumento la matriz que contiene el lote de imágenes y el directorio a guardarlas. Se utilizará programación con mutex para impedir que mientras esté ejecutándose se pueda sobreescribir por un nuevo lote entrante de imágenes.

```
void Thread::run()
{

    int time = QDateTime::currentDateTime().time().second();
    QString dateTime = QString::number(time);
     qDebug()<<"date:"<<dateTime;


     unsigned long sizeMat=1024*1024*20;
        long *mat= new long[sizeMat];

             QMutexLocker locker(&mutex);
        for(int i=0;i<20;i++){

        for(int j=0;j<1024*1024;j++){


            mat[i*1024*1024+j]=myArr[i][j];
        }
        }


       fitsfile *fptr;
       QString filename = Dir+dateTime+".fits";
       qDebug()<<"filename: "<<filename;
       QByteArray FileName = filename.toLatin1();
       int status, ii, jj;
       long fpixel = 1, naxis = 3, exposure;
       long naxes[3];
        LONGLONG nelements=1024*1024*20;
       status = 0;
       fits_create_file(&fptr,FileName,&status);
           naxes[0]=naxes[1]=1024;
       naxes[2]=20;
       int er2=10;

       fits_create_img(fptr,ULONG_IMG,naxis,naxes,&status);
       while(er2!=0){
       er2=fits_write_img(fptr,TLONG,naxis,nelements,mat,&status);}
       qDebug()<<"er2:"<<er2;
       fits_close_file(fptr,&status);
       qDebug()<<"finished saving package";

   }
```

Figura 47.- Constructor de la hebra

La hebra deberá convertir el QVector<QVector>> long a una cadena tipo long del tamaño apropiado, ya que la librería CFITSIO es independiente de QT y no va a comprender este tipo de formato. Sabrá descomponer las imágenes gracias a los valores guardados en *naxes*. Este es un vector de tres espacios, donde la librería interpreta que:

long naxes[3]={Ancho, Alto, Número de imágenes}

De esta manera identifica en la cadena cada imagen: sabe que debe partir dicha cadena en lotes de 20 sabiendo el tamaño que le pertenece a cada imagen. Para identificar los diferentes lotes se ha añadido la hora a la que el sistema los crea, añadiendo el nombre identificador que previamente el usuario ha especificado al crear el directorio.

Cuando se ha guardado el fichero, la hebra comunicará el evento por el modo debug en pantalla. Esta hebra será llamada cada vez que se guarden 20 imágenes en la dirección *Arr,* el cual consiste en un vector de vectores tipo long, que almacena en cada posición una imagen.

```
Arr->push_back(dat);
 if(Arr->size()==20){

   myThread = new Thread(*Arr,myDir);
   if(myThread->isFinished()){myThread->start();}
   else if(!myThread->isFinished()){
       myThread->quit();
       myThread->start();}


   delete Arr;
   Arr = new QVector<QVector<long>>;
 }
```

Figura 48.- Declaración de hebra

Si la hebra ha terminado se iniciará el proceso de la misma, sin embargo, si se da el caso de que aun está en modo de trabajo (aunque ya haya grabado la imagen), se obligará a que termine para empezar de nuevo. Se borrará la matriz contenedora de imágenes y se reabrirá el puntero para reiniciar el proceso.

Este algoritmo está introducido en la función getImage, la cual difiere en este detalle con respecto al descrito en el servidor.

2.1.5.-FastCam

Se completará el sistema de adquisición adaptado para trabajar en remoto con un soporte de procesamiento para *Lucky Imaging*. Este programa está pensado para poder procesar cubos de imágenes adquiridos en formato .FITS.

El objetivo consistirá en que dado, supuesto el caso de estudio sobre una estrella (por ejemplo), dentro de un cúmulo de 1000 imágenes, obtener el 5% de imágenes que garantice la mayor calidad de las mismas con máxima intensidad. Luego, los defectos propios de la atmósfera se corregirán utilizando el método de *Shift and Add*, el cual será explicado posteriormente.

Interfaz gráfica

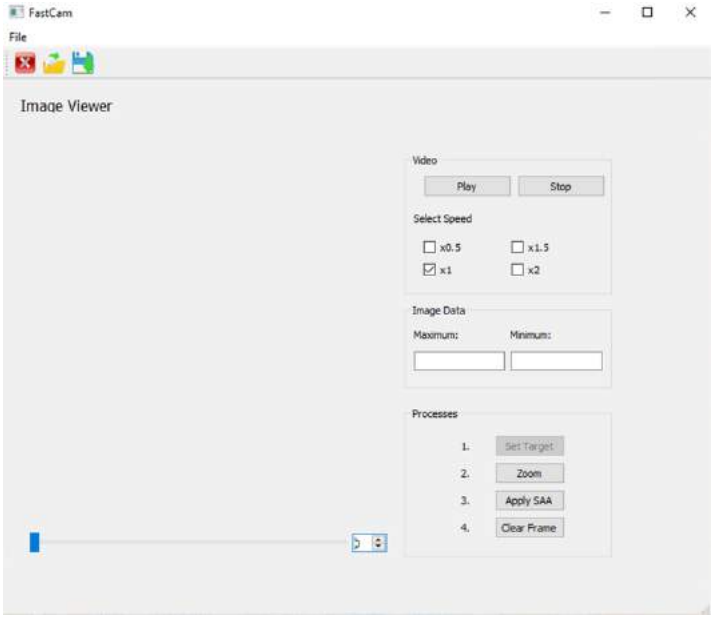


Figura 49.-Pantalla principal de FastCam

En la figura 49 se muestra el visor sin cargar lotes. Desde su estado inicial espera a que el usuario introduzca la serie de imágenes a tratar. Seleccionando un lote de 1000 imágenes como ejemplo, da el resultado en pantalla que se aprecia en la figura 50.

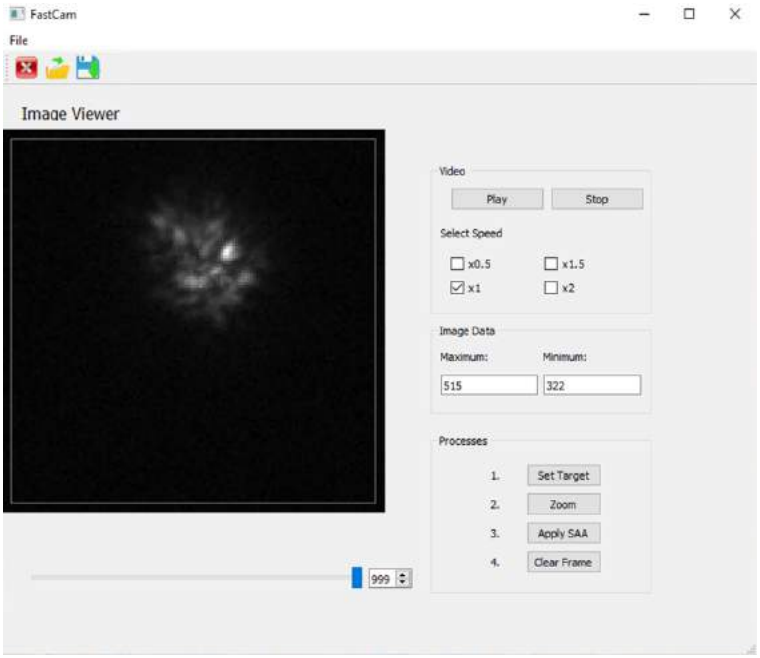


Figura 50-Pantalla después de seleccionar un lote

Se puede visualizar como pseudo-vídeo el lote completo a distintas velocidades, pero las funciones principales son:

- Seleccionar las imágenes de mayor calidad para, posteriormente, aplicar el procesado de shift and add (Apply SAA)
- Poder recortar y disminuir de tamaño las imágenes para poder recentrar en una zona en concreto del lote (set target y zoom)

Funcionamiento

Open Lot

Se ha de recurrir a las funciones contenidas en la librería proporcionada por CFITSIO para poder adquirir la imagen en una matriz utilizable para poder visualizarla en pantalla. Teniendo en cuenta que se encargará de abrir lotes, habrá que trabajar con matrices tridimensionales.

Set Target

Se debe permitir que un usuario cree un rectángulo en el visor para que pueda centrar la parte a copiar de la imagen. La posición y medida de este se guarda en memoria dinámica del programa.

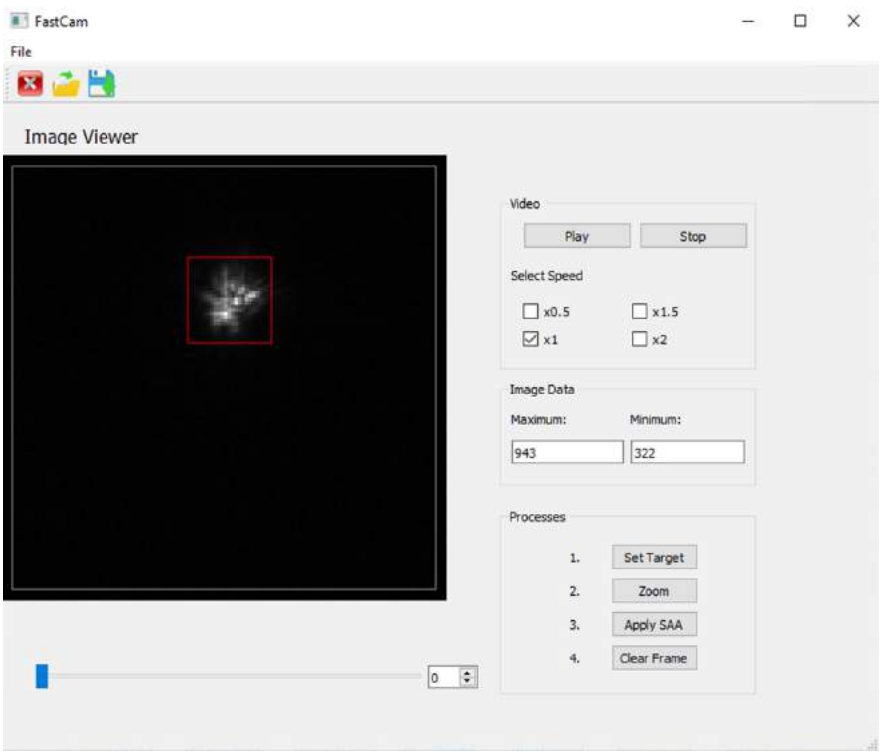


Figura 51.- Selección de objetivo

Para realizar esto se debe utilizar un módulo en Qt conocido como *QPaintEvent*. Es un tipo de máquina virtual que se encarga de dibujar en la pantalla por eventos. Por ejemplo, en este caso queremos que, una vez hayamos pulsado el botón, de "Set Target", se empiece a dibujar un cuadrado en la pantalla cuando clicamos en una esquina y se termine de dibujar cuando hayamos desplazado el puntero a otra.

QPaintEvent y *QMouseEvent* hacen posible que, habiendo detectado que el usuario ha pulsado el botón izquierdo (por ejemplo) del ratón, si se dan las condiciones empiece a dibujar lo establecido en el algoritmo.

```
void MainWindow::mousePressEvent(QMouseEvent *event)
{ if(mEnabled){
        if(event->button()==Qt::LeftButton){

            Begin = event->pos();

        }
    }

}

void MainWindow::mouseReleaseEvent(QMouseEvent *event)
{
      mEnabled= false;
      event->accept();
      translateOk = true;
       entran2(myPixmap);

}
void MainWindow::mouseMoveEvent(QMouseEvent *event)
{if(mEnabled){
        End = event->pos();
        paint(Begin,End);

        update();
 }

}
```

Figura 52.- Eventos de dibujo en pantalla

En la figura 52 se pueden distinguir las funciones que registran los eventos del ratón que van a pintar el recuadro propuesto para localizar o enfocar una estrella. Estas son: *mousePressEvent*, que registra cuando el usuario hace *click* con el ratón, *mouseReleaseEvent*, que se utilizará para registrar la posición final del puntero, y *mouseMoveEvent*, para ir guardando la variación de posición e ir pintando el cuadro en pantalla.

Las funciones anteriores ayudan a capturar el punto de partida del cuadrado, el desplazamiento y el punto final. El funcionamiento para dibujar el cuadrado es el siguiente:

1.- Se hace click derecho en la pantalla cuando se ha abierto, anteriormente, un lote de imágenes.

2.-Se arrastra el ratón, manteniendo el click, hacia un lado conveniente de la pantalla. Mientras esto sucede se va dibujando el cuadrado. Se llama a la función Paint para ir dibujándolo. Luego, la función entran2 dibuja el cuadrado en un frame superior a la pantalla donde se dibuja la imagen, sino ocurren distorsiones.

3.-Una vez decidido el tamaño final, se suelta el click y se queda dibujado permanentemente.

```
void MainWindow::paint(QPoint mBegin,QPoint mEnd)
{       translateOk = false;
    QImage img(myPainterLabel->size(),QImage::Format_ARGB32);
    QPixmap myPix(QPixmap::fromImage(img));


    QPainter painter(&myPix);
    painter.setRenderHint(QPainter::Antialiasing,true);
    painter.setPen(Qt::red);

    QRect rect(mBegin.x(),mBegin.y(),mEnd.x()-mBegin.x(),mEnd.y()-mBegin.y());
    painter.drawRect(rect);


    mRect = rect;

    myPainterLabel->setPixmap(myPix.scaled(myPainterLabel->size(),Qt::KeepAspectRat
        if(!myPix.isNull()){
            myPixmap = myPix;

        }
}
```

```
void MainWindow::entran2(QPixmap pix)
{
  //ui->label_2->setPixmap(pix.scaled(ui->label_2->size(),Qt::KeepAspectRatio));
   dragitem *item = new dragitem(this);
   item->setMyPixmap(&pix);

   myframe = new QFrame(this);
   myframe->setGeometry(0,100,384,384);
   myPainterLabel->clear();
   myPainterLabel->update();
   myframe->raise();
   myframe->setVisible(true);
   QHBoxLayout *lay2 = new QHBoxLayout(myframe);
   lay2->addWidget(item);

   // ui->framePainter->layout()->addWidget(item);
}
```

Figura 53.-Algoritmo de dibujo en pantalla

Zoom

Set Target carecerá de valor sin el algoritmo de "Zoom". Este se encargará de recortar, a lo largo del lote de imágenes, dentro de un área y coordenadas especificados por el cuadrado y su situación dentro de la imagen decidido por el usuario.

```
QVector<QPixmap> croppedCube;
int n = FitsCube.length();
qDebug()<<abs(m_x-abs(mRect.height()))*1.25;
QVector<QImage> croppedImage;

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

   QPixmap pix;
   int hR=0;

   ui->horizontalSlider->setValue(i);
   pix = * pixFitsView->pixmap();
   if(mRect.height()<(FitsCube.first().length()/2)){
    hR=mRect.height();
   }else hR=(FitsCube.first().length()/2);

   QPixmap crop = pix.copy((m_x-abs(mRect.height()))*1.25,m_y-10,abs(hR),abs(hR));
   croppedCube.push_back(crop);
   croppedImage.push_back(croppedCube.at(i).toImage());

}
QVector<QVector<int>> imgVV;

for(int i=0;i<n;i++){
imgVV.push_back(addToVectMat(croppedImage.at(i)));
}
```

Figura 54.- Algoritmo de zoom

Utilizando la librería propia de Pixmap de qt, se puede "copiar y pegar" una región de la imagen dentro de otra variable. Esto es lo que se hace en la línea donde se define QPixmap crop, dentro del bucle for. Se puede observar que se ha ajustado un offset con las coordenadas a pasar para el recorte: esto viene porque las coordenadas vienen en relación con la pantalla general, y no donde se dibuja el cuadrado.

También se puede aclarar que si el cuadrado supera en dimensiones a la altura real de la imagen sin modificar, directamente el código devolverá la imagen sin recortar. Esto se hace debido a que la imagen mostrada en pantalla es mayor en dimensiones que el fichero real.

El bucle for irá recorriendo el cubo original de imágenes, realizando el recorte y guardando cada nueva imagen en un vector de almacenamiento. Posteriormente, se realizará el guardado del fichero en .fits como se ha descrito anteriormente.

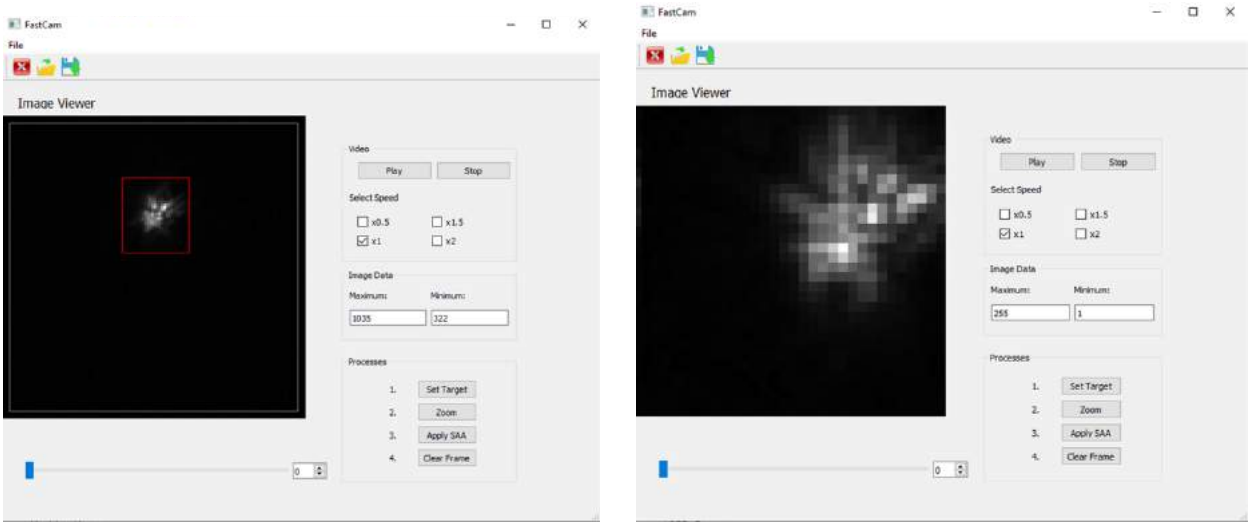


Figura 55.- Resultado de zoom

Shift and Add y el Método de Lucky Imaging

El filtro de procesado de imágenes astronómicas conocido como Shift and Add consiste, mediante el recentrado contínuo de las imágenes de cortas exposiciones, en corregir la distorsión que estas presentan. Las imágenes que se utilizan para esto provienen de la técnica del Lucky imaging, pero solo utilizará aquellas donde se garantiza la mejor calidad. Así se obtendrán mejores resultados en la reconstrucción. El procedimiento será el siguiente:

1.-Selección de las mejores imágenes del cubo

```
int N = FitsCube.length();

QVector<QVector<long>> LuckyCube;
int maxC=1000;
bool okMax=false;

for(int i=0;i<N;i++){
    for(int j=0;j<iW*iH;j++){
    if(FitsCube[i][j]>maxC){
      maxC=FitsCube[i][j];
      okMax=true;
    }

    }
    if(okMax){
        LuckyCube.push_front(FitsCube.at(i));
        okMax=false;
    }


}
```

Figura 56.- Selección de imágenes

La selección de imágenes se hace en base de la magnitud de brillo. Si la imagen supera en brillo a la anterior leída, se guardará en el vector de imágenes para procesado. El resto de las imágenes del lote no se aprovecharán: solo interesarán las de mayor calidad.

2.-Conversión de cadena a Matriz 2D

```
QVector<long> mX,mY,minV;
  for (int k =0; k<LuckyCube.length();k++){

     int max, j0,i0;
     max=0;
     int min = 100000;

     long array[iW][iH];

     for(int i=0;i<iW;i++){
         for(int j=0;j<iH;j++){


             array[i][j]=LuckyCube[k][i*iH+j];
         }
     }
```

Figura 57.- Conversión de cadena a matriz

Una vez obtenido un vector de imágenes, como se guardan como cadena habrá que convertir el vector 1D en una matriz 2D, para lograr sacar las coordenadas X e Y del píxel más brillante.

3.-Sacar coordenadas del píxel más brillante

```
      for(int i=0;i<iW;i++){
          for(int j=0;j<iH;j++){


              if(array[i][j]>max){
                  max=array[i][j];
                  i0=i;
                  j0=j;
              }
              if(array[i][j]<min){
                  min=array[i][j];
              }
          }
      }

      mX.push_back(i0);
      mY.push_back(j0);

      minV.push_back(min);


}
```

Figura 58.- Coordenadas de los máximos de cada imagen

Solamente habrá que localizar los máximos de la matriz y guardar las coordenadas del absoluto. Las coordenadas del máximo se van actualizando cada vez que el bucle if detecta un nuevo máximo, pero al final de la imagen se guardará este absoluto. Este máximo se va actualizando por imagen; no se arrastra para la siguiente.

Se obtendrán dos vectores, mX y mY, que guardarán en cada posición las coordenadas X e Y de los máximos absolutos del cubo de imágenes (del cubo creado con las que se garantiza la mayor calidad con respecto a la intensidad de luz).

4.-Acumulación

```
int iA,jA;
iA=mX.first();
jA=mY.first();

int nim=LuckyCube.length();
int total =1;
long accumulador[iW][iH];

int dx,dy;
for (int k =0; k<total;k++){
for(int i=0;i<iW;i++){
    for(int j=0; j<iH; j++){
        accumulador[i][j]=0;
    }
}
}
qDebug()<<"Fase 2.1";

qDebug()<<LuckyCube.length();

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

    dx = iA - mX.at(i);
    dy = jA - mY.at(i);
    for(int w =0; w<iW;w++){
        for(int h=0;h<iH;h++){

            int absX,absY;
            absX = dx+w;
            absY = dy+h;
            if(absX>=0 && absX<iW &&absY>=0 && absY< iH){

                accumulador[w][h]+=(LuckyCube[i][absX+absY*iH])/nim;

            }else{
                accumulador[w][h]+=0;
            }
        }
    }
```

Figura 59.- Algoritmo de acumulación entorno a un centro

En la anterior figura se muestra el procedimiento seguido para el recentrado y acumulación del lote de imágenes en una sola matriz, que devolverá como resultado una única imagen. Primero, debe crearse esta matriz contenedora con las mismas dimensiones que el resto (en cuanto altura y ancho), y debe ponerse todos los píxeles con un valor inicial cero. Luego, deberá calcularse los diferenciales de posición de los máximos y mínimos de cada imagen con respecto al centro absoluto establecido. Este se tomará como la posición del máximo de la primera imagen del lote. Este diferencial deberá situarse dentro de la imagen. Para ello se le sumará el ancho o el alto dependiendo de si se trata el diferencial en X o en Y.

Si al centrar en la imagen el diferencial está dentro del rango de la misma, se pasará a situar este valor en el acumulador, pero aplicando el valor con la media del numero total de imágenes (sino se podría llegar a una saturación). Si está fuera del rango, directamente se le pasará un cero al acumulador.

Resultados del procesado

Para la comprobación del correcto funcionamiento del algoritmo se ha trabajado con un cubo de imágenes de la estrella HR511, una enana naranja situada en la constelación de Casiopea.

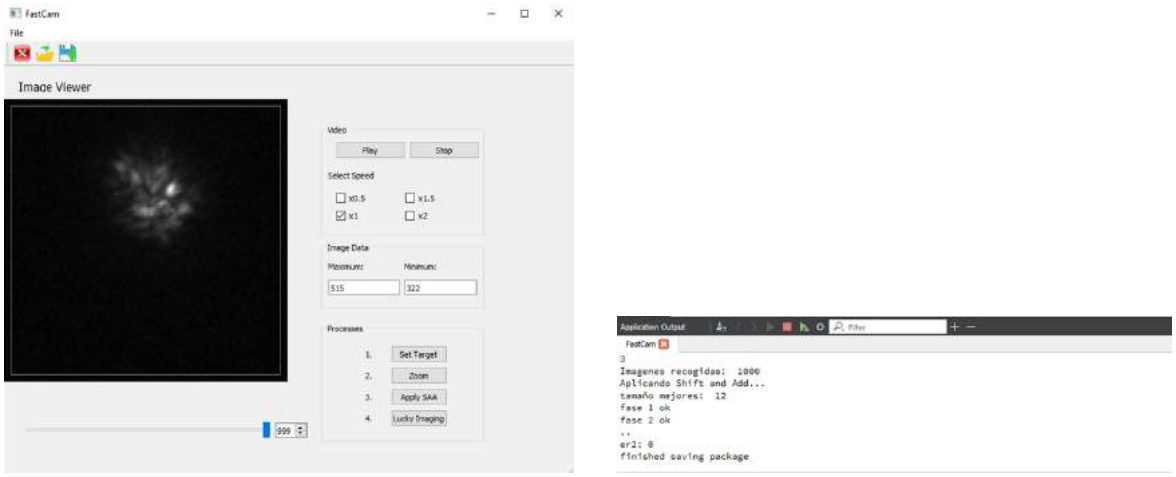


Figura 60.- Antes del procesado

Como se puede apreciar en la anterior figura, la imagen que se recibe de la estrella se encuentra muy distorsionada por culpa de la atmósfera. Una vez que el procesado ha creado el archivo .FITS, lo abriremos en el programa de visualización profesional DS9 para evaluar su correcta creación y, evidentemente, visualizar si ha corregido el efecto de la atmósfera. Como se podrá ver en la siguiente figura, la mejoría es notable; se ha logrado recomponer la imagen de la estrella eficientemente.

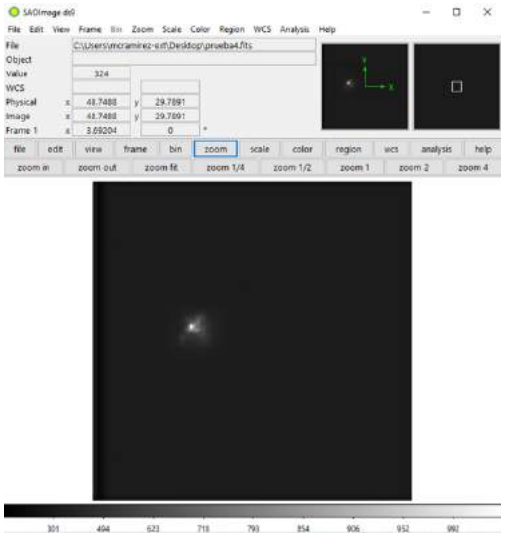


Figura 61.-Resultado del procesado *Shift and Add*

3.-Control de Instrumentación Óptica

En el proceso de adquisición de imágenes astronómicas no solo va a participar el detector EMCCD, sino que también intervienen elementos que actuarán como filtros de imagen. Estos elementos van ayudar a garantizar que se obtiene una imagen de la mejor calidad óptica posible como , a su vez , en distintas bandas para diferentes estudios posteriores. Estos elementos serán un Corrector de Dispersión Atmosférica y una rueda de filtros.

Esta instrumentación ha sido controlada tradicionalmente con elementos predispuestos por el fabricante, pero no se lograba con facilidad su máxima adaptibilidad para distintos proyectos astrofísicos. Esto es un inconveniente, ya que distintos proyectos requerirán cualidades distintas aunque se basen en la misma instrumentación.

Para solventar este problema, se ha pensado en interrumpir la dependencia de la instrumentación con la ayuda de las herramientas del fabricante. Para ello, se ha estudiado la electrónica base controlable de cada objeto, y se ha diseñado su respectivo control basado en Arduino. Gracias a esto, se ha independizado la instrumentación para demostrar que, el científico, puede modelar su instrumentación y adaptarla a sus necesidades.

Facilitando la utilización de instrumentación óptica desde su control con Arduino, se ha dispuesto en este trabajo de unas sencillas interfaces gráficas que se comunican con dicha placa de control. El usuario podrá utilizar, al mismo tiempo, el software de adquisición de imágenes como al mismo tiempo controlar el resto de instrumentación óptica que interviene en el proceso.

3.1.-Interfaz gráfica para el Corrector de Distorsión Atmosférica

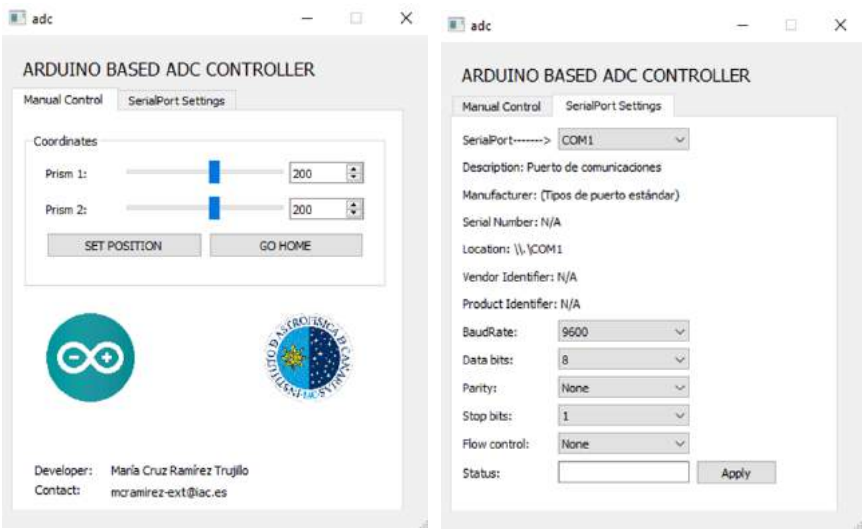


Figura 62.-GUI para el ADC

Se muestra en la anterior figura la interfaz gráfica diseñada en Qt Creator para controlar el corrector de distorsión atmosférica. Su funcionamiento es relativamente sencillo: una vez escogido el puerto desde la pestaña SerialPort Settings, y haciendo los ajustes necesarios para la correcta comunicación con el puerto, empieza a comunicarse con el programa ya guardado en el microcontrolador como si de un socket se tratase. Es decir, es capaz de leer la respuesta del serial del arduino así como escribir comandos para que el microcontrolador los reciba y ejecute las acciones correspondientes.

A modo de prueba de laboratorio se ha dispuesto que el científico pueda controlar las rotaciones de los prismas que componen el corrector. Por supuesto, también puede indicarle al microcontrolador que vuelvan los prismas a la posición inicial con el botón "Go Home".

3.1.1.-Eventos

Reconocimiento de Puertos Disponibles

Esta parte es esencial para la correcta comunicación con el microcontrolador Arduino. La facilidad que contendrá es que no será necesario crear una clase aparte del diálogo de la hebra principal para ello; bastará con incluir al fichero header principal una estructura que detalle las propiedades de un puerto serie.

```
struct Settings {

    QString name;
    qint32 baudRate;
    QString stringBaudRate;
    QSerialPort::DataBits dataBits;
    QString stringDataBits;
    QSerialPort::Parity parity;
    QString stringParity;
    QSerialPort::StopBits stopBits;
    QString stringStopBits;
    QSerialPort::FlowControl flowControl;
    QString stringFlowControl;
    QString FrameTransfer;
    QString Shutter;


};
```

Figura 63.-Estructura para las propiedades de los puertos serie

Los parámetros más importantes para establecer la correcta comunicación son:

- **Baud rate:** Velocidad de transferencia de la información a través de un canal de comunicación. Si esta velocidad no coincide con la velocidad de transferencia asignada al microcontrolador, la comunicación no será establecida.

- **Tamaño de datos:** Designar el tamaño de datos que se va a establecer para la transferencia de comandos o respuesta.

Inciso

El protocolo de comunicación del puerto serie rige la siguiente forma de datos:

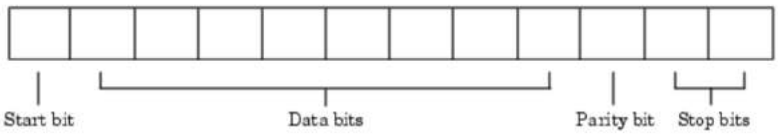


Figura 64.-Cadena de transmisión de datos por puerto serie

Este formato incluye un bit que indica el inicio de la cadena, seguido de entre 5 y 8 bits de datos, un bit de paridad adicional (opcional) y uno de final de cadena. Es importante que al indicar estos parámetros coincida con los que nos indica del puerto de lectura la interfaz de Arduino.

El ajuste de parámetros dependerá de las propiedades del puerto serie utilizado para controlar el arduino. Como se puede comprobar desde el administrador de dispositivos de Windows, los parámetros detallados anteriormente se encuentran especificados en la forma que se aprecia en la figura 65.

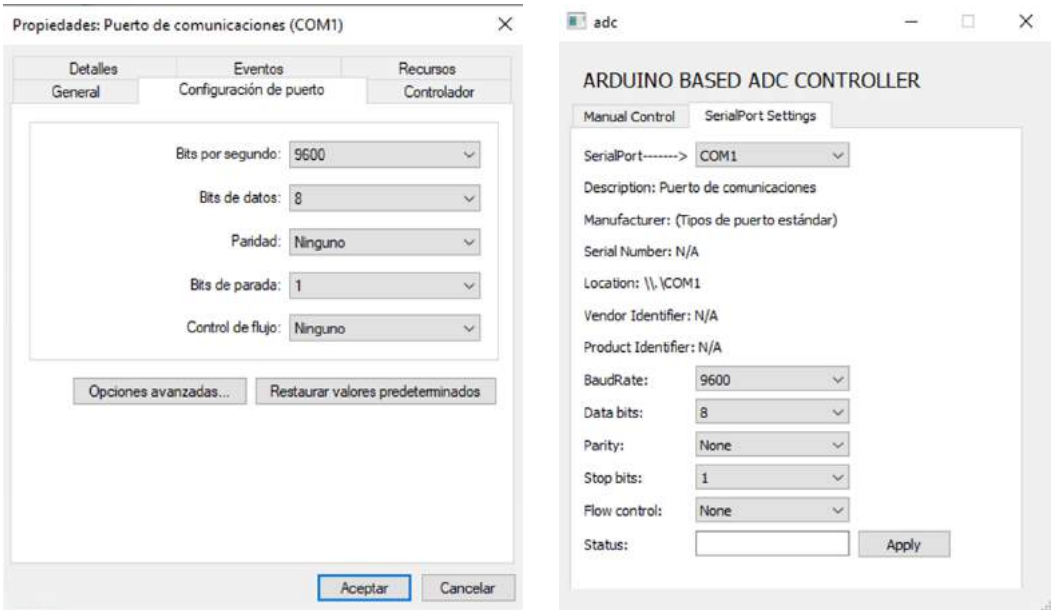


Figura 65.- Comparación de propiedades designadas al puerto utilizado por el microcontrolador dese el adminstrador de dispositivos de Windows 10 entre el módulo de propiedades del puerto serie configurable del software desarrollado en Qt

La configuración del puerto es posible dentro de Qt 5 gracias a la librería QSerialPort. Esta permite acceder a los parámetros de una sucesión de puertos disponibles en el equipo y reconfigurarlos sin dificultad a través de funciones construidas dentro de la propia librería.

```
void adc::apply()
{
    QMessageBox msgBox;
    msgBox.setText("Change Alert");
    msgBox.setInformativeText("You are about to change the SerialPort settings. Are you sure?");
    msgBox.setStandardButtons(QMessageBox::Yes |  QMessageBox::Cancel);

    int ret = msgBox.exec();
    switch (ret){
    case QMessageBox::Yes:
         updateSettings();
         serialPort->setPortName(m_currentSettings.name);
         serialPort->setBaudRate(m_currentSettings.baudRate);
            serialPort->setParity(m_currentSettings.parity);
            serialPort->setDataBits(m_currentSettings.dataBits);
            serialPort->setStopBits(m_currentSettings.stopBits);
            serialPort->open(QSerialPort::OpenModeFlag::ReadWrite);
         QMessageBox::information(this,"Information","Changes had been applied to SerialPort "+m_currentSettings.name);
       qDebug()<<serialPort->portName();
       if(serialPort->portName()==arduino_port_name){
       m_ui->lineEdit_status->setText("ARDUINO CONNECTED");
       m_ui->lineEdit_status->setStyleSheet("QLineEdit { background: rgb(77, 232, 113); selection-background-color: rgb(77, 232, 113); }");
       }else{
           m_ui->lineEdit_status->setText("CONNECTED");
           m_ui->lineEdit_status->setStyleSheet("QLineEdit { background: rgb(50, 232, 113); selection-background-color: rgb(77, 232, 113); }");

       }
        break;
    case QMessageBox::Cancel:
        break;


    }
```

Figura 66.-Aplicación de cambios en el puerto serie escogido

Cada vez que el usuario pulsa el botón de “Apply”, esta función se ejecuta. Primero recoge los nuevos parámetros a aplicar llamando a la función updateSettings, para actualizar la información contenida en los punteros de name, baudRate, dataBits y stopBits.

```
void adc::updateSettings()
{
    m_currentSettings.name = m_ui->serialPortInfoListBox->currentText();

    if ( m_ui->baudRateBox->currentIndex()==4){
        m_currentSettings.baudRate = m_ui->baudRateBox->currentText().toInt();
    }else{
        m_currentSettings.baudRate = static_cast<QSerialPort::BaudRate>(
        m_ui->baudRateBox->itemData(m_ui->baudRateBox->currentIndex()).toInt());

    }

    m_currentSettings.stringBaudRate = QString::number(m_currentSettings.baudRate);

    m_currentSettings.dataBits =static_cast<QSerialPort::DataBits>(
                m_ui->dataBitsBox->itemData(m_ui->dataBitsBox->currentIndex()).toInt());
    m_currentSettings.stringDataBits = m_ui->dataBitsBox->currentText();

    m_currentSettings.parity =static_cast<QSerialPort::Parity>(
                m_ui->parityBox->itemData(m_ui->parityBox->currentIndex()).toInt());
    m_currentSettings.stringParity = m_ui->parityBox->currentText();

    m_currentSettings.stopBits =static_cast<QSerialPort::StopBits>(
                m_ui->stopBitsBox->itemData(m_ui->stopBitsBox->currentIndex()).toInt());
    m_currentSettings.stringStopBits = m_ui->stopBitsBox->currentText();

    m_currentSettings.flowControl =static_cast<QSerialPort::FlowControl>(
                m_ui->flowControlBox->itemData(m_ui->flowControlBox->currentIndex()).toInt());
    m_currentSettings.stringFlowControl = m_ui->flowControlBox->currentText();

   // m_currentSettings.localEchoEnabled = m_ui->localEchoCheckBox->isChecked();

}
```

Figura 67.- Actualización de parámetros para el puerto serie

Como se muestra en la figura anterior, la función updateSettings se encarga de recoger la información dentro de los cajetines editables de la interfaz gráfica y guardarla en los punteros correspondientes.

Si un arduino se encuentra conectado al puerto serie, al aplicar los cambios se notifica en la interfaz como “Arduino Connected”

Figura 68.-GUI cuando se han realizado la adaptación de parámetros para la comunicación con Arduino

Nota: La imagen que se muestra en la figura 68 fue tomada desde un Mac como muestra que el proyecto se puede compilar desde distintas plataformas

Detección de arduino disponible

```
void adc::fillPortsInfo()
{
    m_ui->serialPortInfoListBox->clear();
 QString description, manufacturer,serialNumber;
    const auto infos = QSerialPortInfo::availablePorts();
    for ( const QSerialPortInfo &info :  infos){
        QStringList list;
        description = info.description();
        manufacturer = info.manufacturer();

        serialNumber = info.serialNumber();

        list << info.portName()
             << (!description.isEmpty() ? description : blankString)
             << (!manufacturer.isEmpty() ? manufacturer : blankString)
             << (!serialNumber.isEmpty() ? serialNumber : blankString)
             << info.systemLocation()
             << (info.vendorIdentifier() ? QString::number(info.vendorIdentifier(), 16) :  blankString)
             << (info.productIdentifier() ? QString::number(info.productIdentifier(), 16) :  blankString);

        m_ui->serialPortInfoListBox->addItem(list.first(),list);
    }

    m_ui->serialPortInfoListBox->addItem(tr("Custom"));


    foreach(const QSerialPortInfo &serialPortInfo, QSerialPortInfo::availablePorts()){
        if(serialPortInfo.hasVendorIdentifier() && serialPortInfo.hasProductIdentifier()){
            if(serialPortInfo.vendorIdentifier() == arduino_uno_vendor_id){
                if(serialPortInfo.productIdentifier() == arduino_uno_product_id){
                    arduino_port_name = serialPortInfo.portName();


                    arduino_is_available = true;
                }
            }
        }
    }
```

Figura 69.-Algoritmo que recorre los puertos disponibles y encuentra si hay un arduino conectado

Teniendo registrado en el programa las propiedades identificadoras del arduino, como es el ID del vendedor y el ID del producto, solo habrá que recorrer las propiedades de los puertos detectados hasta que estas propiedades coincidan. Cuando esto ocurra, la variable booleana "*arduino_is_available*" pasará al estado TRUE. Esto será declarado en el archivo header del programa como:

```
QSerialPort *arduino;
static const quint16 arduino_uno_vendor_id = 9025;
static const quint16 arduino_uno_product_id = 67;
QString arduino_port_name;
bool arduino_is_available;
QString descriptionArduino;
QString ArduinoPort;
```

Figura 70.-Parámetros para la detección de arduino

Comunicación con Arduino

La comunicación con el Arduino se hará a través del serial. Se comportará de una manera parecida al QTcpSocket: el Arduino estará continuamente leyendo el serial, a la espera de determinados comandos. Entonces, lo que se tendrá que hacer desde Qt es escribir estos mismos como si de un socket se tratase.

Para esta aplicación solo habrá dos comandos base:

- **Set de una posición particular:** Se hará enviando una cadena de caracteres que empieza con el carácter "_" (debe de empezar con cualquier carácter excepto por una h). De esta forma, el Arduino entenderá que van a llegar dos posiciones separadas por una coma: la primera correspondiente al prisma 1 y la segunda correspondiente al prisma dos. Esto se entenderá más fácilmente cuando se explique a continuación el algoritmo en Arduino.

```
void adc::on_pushButton_clicked()
{
    //Envia posición

    int posPrism1 = m_ui->prism1Slider->value();
    int posPrism2 = m_ui->prism2Slider->value();
    QString pos1,pos2;
    pos1=QString::number(posPrism1);
    pos2=QString::number(posPrism2);
    QString commandMov = "_"+pos1+","+pos2;
    QByteArray ba = commandMov.toLocal8Bit();
    const char *myCommand = ba.data();
    serialPort->write(myCommand);
    serialPort->waitForBytesWritten(3000);


}
```

Figura 71.-Envío de comandos de movimiento al Arduino

- **Set home:** Gracias a los dos finales de carrera que posee cada prisma, solo habrá que esperar a un alto en la salida de ambos para saber que se ha llegado a la posición HOME. Bastará con mandar una h al arduino para que entienda que tiene que buscar esta posición de referencia.

```
void adc::on_pushButton_2_clicked()
{
    //Envia al inicio

    serialPort->write("h");
    serialPort->waitForBytesWritten(3000);

}
```

Figura 72.-Comando set home

3.2.- Control del Corrector de Distorsión Atmosférica en Arduino

El funcionamiento del programa va a ser relativamente sencillo. Nada más arrancarse, prisma por prisma se buscará la posición HOME (por orden, es decir, primero el primer prisma y luego el segundo).

Luego, el programa se quedará a la espera de la introducción de una posición para los dos prismas a través del serial. Introduciendo un valor para cada prisma, cada motor rota la cantidad de número de pasos indicada a través del serial. Esto se puede realizar en un sentido positivo o negativo.

Durante el movimiento se puede introducir una nueva combinación de pasos, interrumpiendo el desplazamiento anterior y reanudando el movimiento hasta llegar a la nueva posición indicada.

Por último, si llega al serial el carácter “h”, se reseteará cada prisma a la posición home.

```
void loop(){

  if (Serial.available()>0){                              // Mira si se ha escrito algo en el serial

   if (Serial.read() == 'h'){

    stepperHome(1);
    stepperHome(2);
   }

   else{

    Travel1= Serial.parseInt();
    Serial.print(Travel1);

    Travel2= Serial.parseInt();
    Serial.print(Travel2);

    stepper1.moveTo(Travel1);
    stepper2.moveTo(Travel2);

    while ((stepper1.distanceToGo() != 0) || (stepper2.distanceToGo() !=0)) {              // Continua moviendose hasta alcancar posición deseada

     stepper1.run();
     stepper2.run();
     Serial.println(stepper1.currentPosition());
     Serial.println(stepper2.currentPosition());

     if (Serial.available()>0){                           // Interrumpe order anterior para ir a la nueva posición

       Travel1= Serial.parseInt();
       Serial.print(Travel1);

       Travel2= Serial.parseInt();
       Serial.print(Travel2);

       stepper1.moveTo(Travel1);
       stepper2.moveTo(Travel2);

       while ((stepper1.distanceToGo() != 0) || (stepper2.distanceToGo() !=0)) {  //continua el movimiento hasta que se alcance la posición deseada

         stepper1.run();
         stepper2.run();
         Serial.println(stepper1.currentPosition());
         Serial.println(stepper2.currentPosition());
       }
     }
    }
   }
  }
}
```

Figura 73.-Loop de funcionamiento en Arduino. Dependiendo del comando entrante buscará la referencia o se moverá a una posición.

La función setHome (llamada *stepperHome*) funcionará de la forma mostrada en la figura 74.

```
void stepperHome(int mot){

  if (mot == 1){

    hBval1 = digitalRead(homeButton1);
    while (hBval1 == HIGH){                  //Retrocede lenamente hasta que no se detecte un LOW en el switch

      stepper1.moveTo(-10000);
      stepper1.run();
    //  digitalWrite(ledPin, LOW);
      hBval1 = digitalRead(homeButton1);
    }

   // digitalWrite(ledPin, HIGH);              //indicates it's doing something
    stepper1.setCurrentPosition(0);          //Sale de la rutina
  }

  if (mot == 2){

    hBval2 = digitalRead(homeButton2);
    while (hBval2 == HIGH){                  //Retrocede lenamente hasta que no se detecte un LOW en el switch

      stepper2.moveTo(-10000);
      stepper2.run();
     // digitalWrite(ledPin, LOW);
      hBval2 = digitalRead(homeButton2);
    }

  //  digitalWrite(ledPin, HIGH);
    stepper2.setCurrentPosition(0);          //Sale de la rutina
  }
}
```

Figura 74.-Función de encontrar las posiciones de referencia para cada motor

Según el motor al que se le ha convocado para ir al home, se iniciará uno de los dos bucles if. Igualmente, el comportamiento es el mismo para ambos. Mientras que el valor del switch sea alto, se irá retrocediendo hasta que esto cambie. Se saldrá de la función cuando se haya recibido el LOW, haciendo que la posición actual sea la de referencia.

Al inicio del programa, en el bucle de *setup*, se establece la rutina de home indicada anteriormente, tal que:

```
void setup()
{
    Serial.begin(9600);
    stepper1.setMaxSpeed(100.0);
    stepper1.setAcceleration(50.0);

    stepper2.setMaxSpeed(100.0);
    stepper2.setAcceleration(50.0);

   stepperHome(1);
    stepperHome(2);
}
```

Figura 75.-Setup previo a la ejecución principal

Se indica velocidad máxima y aceleración. Luego, se llama a la rutina de setHome, y el programa estará listo para funcionar.

3.2.1.-Montaje

A continuación se va a detallar como hacer las conexiones entre los distintos elementos del sistema para hacer funcionar el Corrector de Dispersión Atmosférica en arduino.

- **Arduino Uno:** contiene el microcontrolador que va a contener y ejecutar el programa creado en el Arduino IDE. Gracias a su propiedad multi-placa, podremos trabajar con las llamadas “Arduino shield” para añadir puertos adicionales de control.

Figura 76.-Arduino Uno

- **Arduino Sensor Shield:** Proporciona de puertos adicionales para los finales de carrera de cada prisma. Este piso es realmente opcional, ya que se podrían soldar los pines de los finales de carrera al *arduino dc motor shield*.

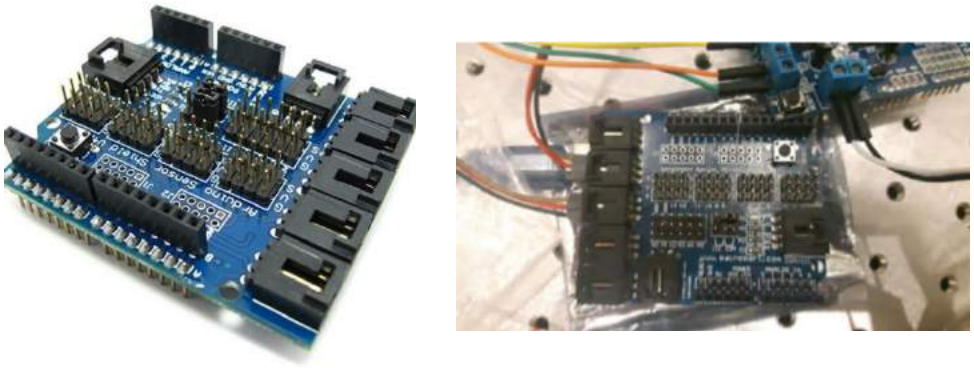


Figura 77.- A la izquierda se puede ver la placa Arduino Sensor Shield. A la derecha, cuando se han realizado las debidas conexiones provenientes de los finales de carrera.

- **Arduino DC motor shield:** Placa encargada de las conexiones a los dos motores bifásicos controladores del movimiento de los prismas.

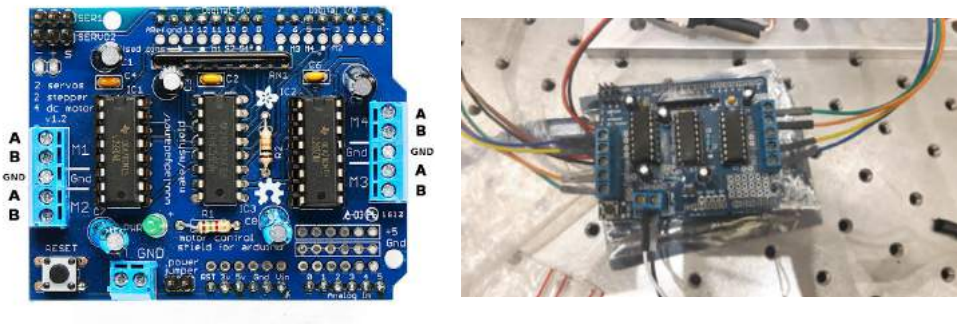


Figura 78.- A la izquierda se puede ver la placa Arduino DC motor Shield. A la derecha, cuando se han realizado las debidas conexiones provenientes de los motores del ADC

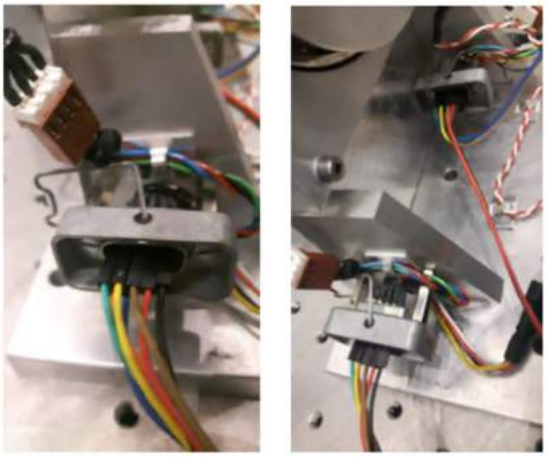


Figura 79.- En esta imagen se muestra la salida del conector DSUB de cada prisma del ADC. El cable, rojo y marrón corresponden al final de carrera. El resto de cables corresponden al motor.

Por último, se conectará a la arduino motor shield una fuente de tensión de 12 V. Con todas las conexiones debidamente realizadas el sistema está listo para funcionar.

3.4.- Interfaz gráfica para la Rueda de Filtros

Para este programa la interfaz será fácilmente la más sencilla de las mostradas en este documento. El único objeto que tiene que cumplir es la selección correcta de los filtros que introducir en el campo óptico del detector. Por lo tanto, las partes principales de este programa serán:

- Comunicación con arduino
- Posición de referencia
- Selección de filtros

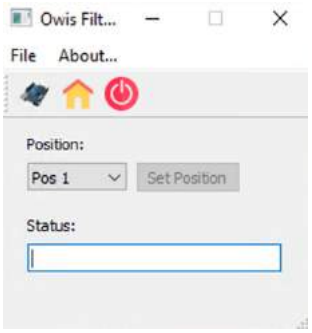


Figura 80.- Interfaz antes de arrancar el funcionamiento

Funcionamiento

1.-Primero, al abrirse la interfaz, tendremos que indicarle que **detecte** por los puertos usb si hay **un arduino conectado**. Si encuentra uno, lo notifica en pantalla, tal que:

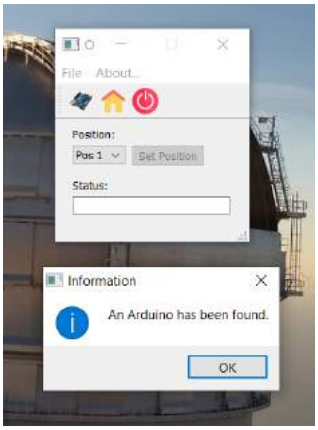


Figura 81.- Interfaz cuando detecta un arduino conectado al puerto serie

El algoritmo para conseguir esto ya se desarrolló en el apartado correspondiente del Corrector de Dispersión Atmosférica (dentro del apartado 3.1.1).

2.-Posteriormente, habrá que seleccionar que busque la **posición de referencia**; sería un error dejar que se seleccionase posición antes de tener la referencia, ya que el algoritmo no sabría situarse correctamente dentro de la rueda de filtros. El programa simplemente se encarga de enviar un comando a través del serial que se encarga de esto.

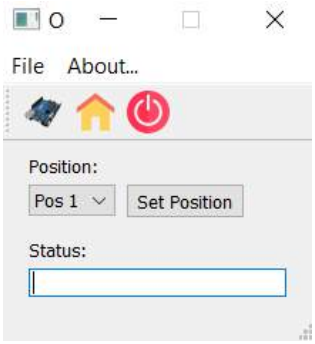


Figura 82.- Interfaz una vez detectada la posición de referencia

```
void MainWindow::on_actionFind_Home_triggered()
{
    if(arduino_is_available==false){
        QMessageBox::information(this,"Alert","No arduino has been found yet. "
                                              "Please, check if Arduino is connected properly,"
                                              " otherwise run Find Arduino.");
    }
    else{
    serialPort->write("FindHome");
    serialPort->waitForBytesWritten(3000);
    ui->setPosButton->setEnabled(true);
    }
}
```

```
if(command.equals("FindHome")){
 while(analogRead(A0)<1000){
   //Serial.println(analogRead(A0));
    myStepper.step(1);
    count+=1;
   delay(50);}
   if(analogRead(A0)>1000) Serial.println("Ya estoy en home");
   Serial.println("Pasos hasta home:");
   Serial.println(count);
   myStepper.step(10);
   actualPos=1;
}
```

Figura 83.- Arriba: Código en QT para la petición de situar el motor en la posición de referencia. Abajo: Algoritmo en arduino para encontrar dicha posición.

En la figura 83 se puede apreciar que para detectar la referencia compara el valor recibido por la entrada analógica A0. La referencia de la rueda de filtros, como se mencionó anteriormente, se compone por un sensor de efecto Hall situado en la parte superior de la rueda de filtros. Su rango de valores de lectura cuando no detecta presencia de imán cierne en las centenas, en cambio, cuando detecta la presencia de este, pasa a las miles de unidades. De ahí que hasta que no detecte esta cifra por la entrada A0 no señalice que se ha encontrado el home. El llevarlo diez pasos más lejos una vez detectado el imán es por offset a la hora de situar correctamente los filtros en el objetivo del detector.

Figura 84.-Mecanismo de detección de la referencia en la rueda de filtros

3.-Una vez completados correctamente los pasos anteriores, el programa ya permite **seleccionar los filtros.** Notifica que ha llegado a la nueva posición con un mensaje en la barra de estado.

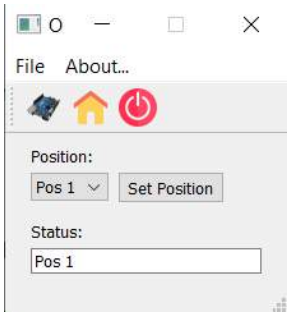


Figura 85.- Interfaz en funcionamiento

Dependiendo del filtro seleccionado, enviará un comando distinto en el serial, pero el funcionamiento no difiere más de lo explicado.

```
void MainWindow::on_setPosButton_clicked()
{
    if(ui->Pos_comboBox->currentText()=="Pos 1"){
        serialPort->write("F1");
        serialPort->waitForBytesWritten(3000);
        serialPort->waitForReadyRead(500);
        ui->status_LineEdit->setText("Pos 1");
    }else if(ui->Pos_comboBox->currentText()=="Pos 2"){
        serialPort->write("F2");
        serialPort->waitForBytesWritten(3000);
        serialPort->waitForReadyRead(500);
        ui->status_LineEdit->setText("Pos 2");
    }
    if(ui->Pos_comboBox->currentText()=="Pos 3"){
        serialPort->write("F3");
        serialPort->waitForBytesWritten(3000);
        serialPort->waitForReadyRead(500);
        ui->status_LineEdit->setText("Pos 3");
    }
    if(ui->Pos_comboBox->currentText()=="Pos 4"){
        serialPort->write("F4");
        serialPort->waitForBytesWritten(3000);
        serialPort->waitForReadyRead(500);
        ui->status_LineEdit->setText("Pos 4");
    }
    if(ui->Pos_comboBox->currentText()=="Pos 5"){
        serialPort->write("F5");
        serialPort->waitForBytesWritten(3000);
        serialPort->waitForReadyRead(500);
        ui->status_LineEdit->setText("Pos 5");
    }
    if(ui->Pos_comboBox->currentText()=="Pos 6"){
        serialPort->write("F6");
        serialPort->waitForBytesWritten(3000);
        serialPort->waitForReadyRead(500);
        ui->status_LineEdit->setText("Pos 6");
    }
}
```

Figura 86.- Posiciones disponibles y su comunicación con el Arduino

Las distintas posiciones se basan en el mismo procedimiento dentro del algoritmo desarrollado en Arduino. A modo de simplificación se explicará a continuación qué ocurre cuando el comando recibido es que se mueva la rueda de filtros a la posición 1.

```
else if(command.equals("F1")){
  if(actualPos==1){Serial.println("Already at F1");}
    else if(actualPos==2){
    myStepper.step(-67);
    actualPos=1;
    delay(500);
    }
  else if(actualPos==3){
    myStepper.step(-134);
    actualPos=1;
    delay(500);
    }
    else if(actualPos==4){
    myStepper.step(-201);
    actualPos=1;
    delay(500);
    }
    else if(actualPos==5){
    myStepper.step(-268);
    actualPos=1;
    delay(500);
    }
    else if(actualPos==6){
    myStepper.step(67);
    actualPos=1;
    delay(500);
    }
```

Figura 87.- Algoritmo en Arduino para situar la rueda de filtros en la posición adecuada, en este caso, en el filtro número seis

Los números de pasos que da el motor va en relación con los grados por pasos. Si se sabe la apertura en ángulo entre filtro y filtro, se multiplica por dicha relación para obtener los pasos, tal que:

$$n_{pasos} = 1.8 \cdot \theta$$

Siendo 1.8 los pasos por grado del motor utilizado en la rueda de filtros, y θ la posición en grados desde la referencia al filtro que se desea seleccionar.

Se comprobó empíricamente en laboratorio que dicha distancia entre filtro y filtro era de 37.22º. Por lo tanto, si quiero ir desde la posición de referencia al primer filtro sé que tendré que girar -37.22º (en sentido horario). Utilizando una fórmula anterior, se llega al resultado de que tendrán que realizarse -67 pasos para llegar al primer filtro.

3.5.-Desarrollo de Electrónica

3.5.1.-Planteamiento del Problema

Se necesita realizar una placa compatible con Arduino que sea capaz de controlar un motor unipolar híbrido y un sensor de Efecto Hall, con su correspondiente adaptación para la señal de lectura. Todas las señales de control pasarán a través de un conector DSUB de 15 pines.

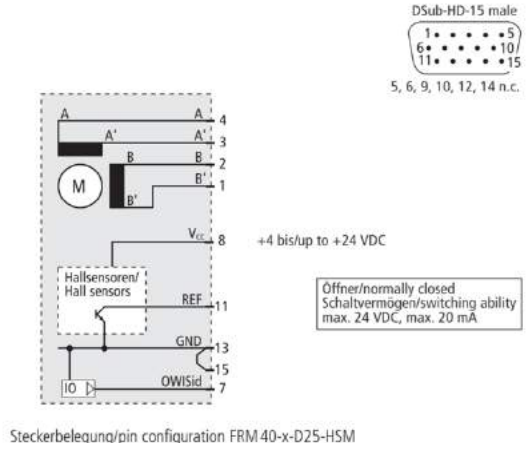


Figura 88.-Conexionado del conector DSUB de la rueda de filtros de OWIS

Sensor de Efecto Hall

Se conoce como **Efecto Hall** a la aparición de un campo eléctrico por separación de cargas de distinto signo en el interior de un conductor por el que circula una corriente en presencia de un campo magnético con componente perpendicular al movimiento de las cargas. Cuando se da estas circunstancias, aparece un voltaje medible entre los extremos del conductor.

Es en este principio físico en el que se basan los Sensores de Efecto Hall; se activan con la presencia de un campo magnético. Las dos características más importantes de este fenómeno físico son la densidad de flujo (B) y la polaridad. La señal de salida de un Sensor de Efecto Hall, como norma general, es la función de la densidad de flujo de un campo magnético alrededor del dispositivo. Cuando dicha densidad excede un límite predispuesto por el dispositivo, el sensor logra detectar el campo magnético y genera un voltaje de salida V_o .

El sensor HAL516A es el más sensible de los interruptores unipolares de salida invertida de la familia de *MICRONAS*. La salida da paso a un estado HIGH cuando detecta un campo magnético y vuelve a un estado LOW cuando este desaparece. Entre sus características se encuentra:

- Tipo de interruptor: unipolar invertido
- Alta sensibilidad
- Flujo Magnético estándar para señal HIGH: B_{ON} = 3.5 mT a temperatura ambiente (25ºC)
- Flujo Magnético estándar para señal LOW: B_{Off} = 5.5 mT a temperatura ambiente (25ºC)
- Opera tanto con campo magnéticos estáticos y dinámicos hasta los 10kHz

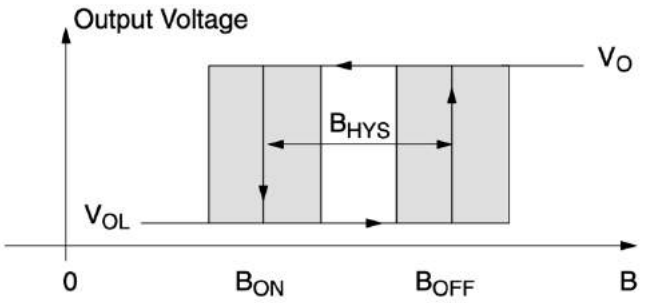


Figura 89.-Definición de los cambios de estado para el Sensor de Efecto Hall 516A de *Micronas*

Figura 90.- 516A utilizado por *OWIS* en su rueda de filtros

Para poder leer la señal que proporciona de salida el sensor de efecto hall, habrá que añadir una etapa de acondicionamiento para poder realizar la lectura. Para este caso es relativamente sencillo: bastará con un acondicionamiento resistivo para crear una diferencia de voltaje. Para realizar la lectura en Arduino se necesitará seguir el esquemático presentado en la figura 91.

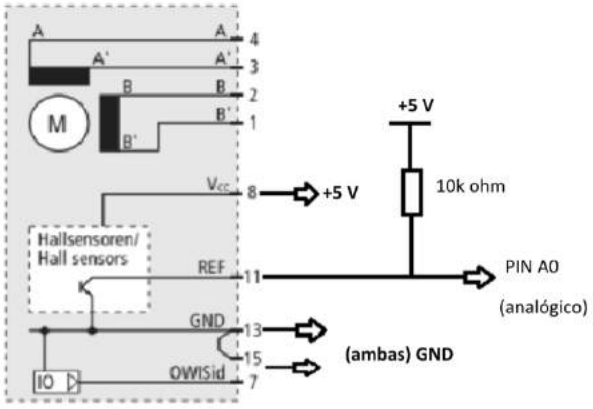


Figura 91.-Esquema de conexiones para la lectura en ARDUINO

Dual Full bridge driver

Un puente en H es un circuito electrónico que generalmente se usa para permitir a un motor eléctrico DC girar en ambos sentidos (en avance y retroceso). Son ampliamente usados en robótica y como convertidores de potencia. Los puentes en H están disponibles como circuitos integrados, pero también se pueden construir a partir de componentes discretos

El término "Puente H" proviene de la representación gráfica del circuito. Un puente H se construye con cuatro interruptores (típicamente con transistores), que se accionan en parejas. Por ejemplo, nunca se podrán accionar a la vez los interruptores de la misma columna, sino que solo podrán cerrarse al mismo tiempo los interruptores T1-T2, estando abiertos T3-T4. O de lo contrario, estando cerrados T3 y T4, deberán estar abiertos T1 y T2.

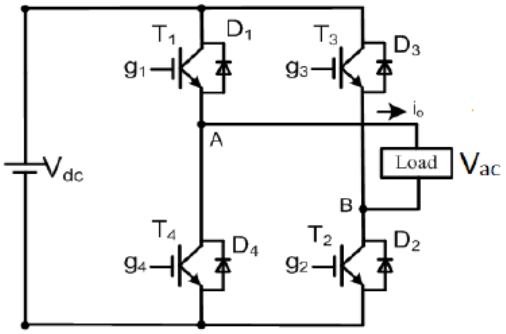


Figura 92.- Circuito de un Puente H

Para nuestro caso, lo que se señala en la figura como "Load" (carga), sería el motor híbrido bipolar que se describió en apartados anteriores. Para dicho motor se ha escogido uno de los puentes H más comunes para controlar esta clase de motores: el L298P (La "P" señala que es el modelo SMD). Es un puente H capaz de soportar altos voltajes y corrientes, diseñado para trabajar bajo los estándares de la lógica TTL y controlar cargas inductivas como relés, solenoides y, por supuesto, motores paso a paso y motores DC. Se dan dos entradas para habilitar o deshabilitar el dispositivo independientemente de las señales de entrada.

Los emisores de cada uno de los transistores inferiores de cada puente se encuentran interconectados y el correspondiente terminal externo se puede utilizar para la conexión de un sensor resistivo.

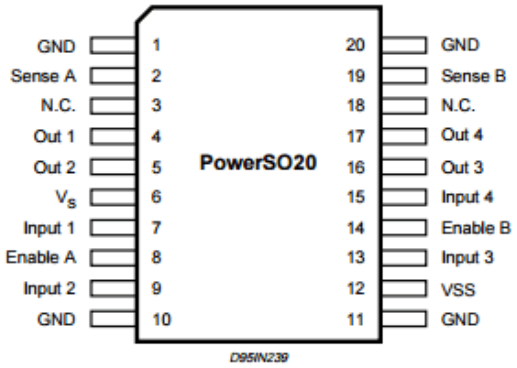


Figura 93.-Circuito integrado L298P

Características:

- Soporta un voltaje de entrada de hasta 46 V
- Corriente continua de hasta 4 A
- Voltaje de baja saturación
- Protección para elevadas temperaturas
- Inmunidad a alto ruido. Acepta como un “0” lógico de entrada hasta los 1.5V

3.5.2.-Esquemático

<u>Conexiones de Entrada</u>

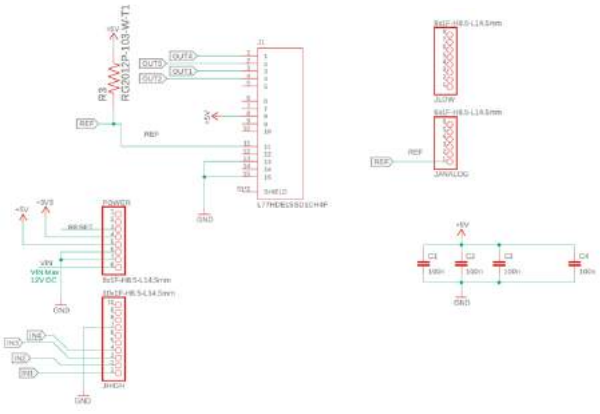


Figura 94.-Parte del esquemático donde se muestra las entradas y salidas de la placa

- **L77HDE15SD1CH4F:** Se trata del conector DSUB que comunicará el arduino con la rueda de filtros. Las cuatro primeras entradas corresponden con el control del motor, y el resto se utilizan para la lectura del sensor de Efecto Hall. Como se ve en la siguiente figura, se corresponderá con lo explicado anteriormente, tal que:

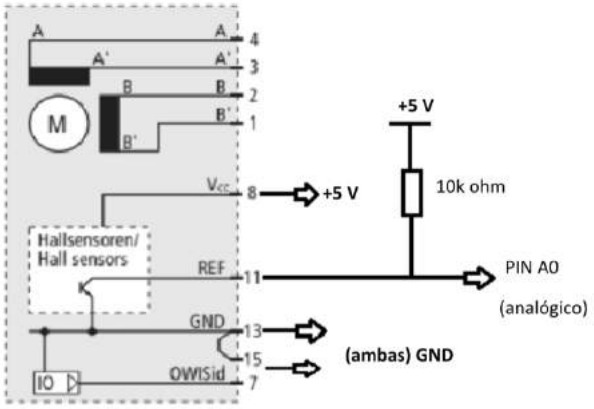


Figura 95.-Esquema de conexiones para la lectura en ARDUINO

Para poder realizar la lectura de la salida del sensor de Efecto Hall, se empleará a su salida una resistencia *pull-up,* que determinará variaciones respecto a un voltaje de referencia, en este caso de 5 Voltios. Aunque es un sensor de salida digital, se ha trabajado con una entrada analógica por su precisión de lectura.

- **Power, JAnalog, JLow y JHigh:** Se corresponden con los pines de entrada/salida del arduino.

- **Condensadores:** Actuarán como filtro de entrada para el motor. De esta manera se amortigua el pico de voltaje producido en el arranque.

Alimentación del Motor

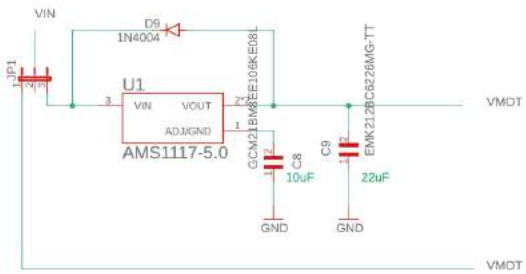


Figura 96.-Circuito de alimentación del motor

Según el fabricante del motor de la rueda de filtros, este es capaz de soportar hasta 75 W por fase. Sin embargo, presentaba dificultades en el control cuando en el laboratorio se aplicaba una potencia mayor a 7 W (con una alimentación de 5V a 1.4 A). Es por esto por lo que se dejará a elección del usuario, dependiendo de la fuente conectada, si regular el voltaje de entrada a 5 Voltios o dejar pasar por completo la potencia de la fuente. Se ha escogido el regulador que utiliza la placa de Arduino Uno, con el circuito aconsejado por el fabricante (ver *datasheet* en Anexos).

Motor Driver

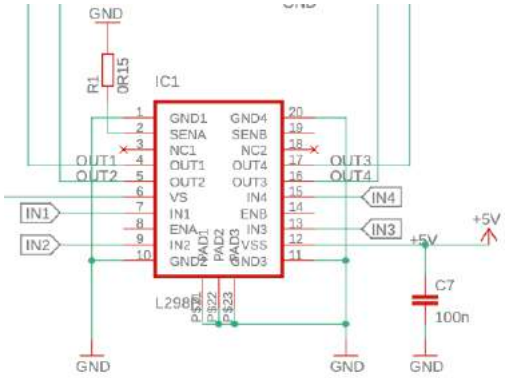


Figura 97.-Esquemático de conexiones del Dual Full Bridge utilizado

El driver utilizado para controlar el motor unipolar paso a paso será el L298, en su versión SMD. Los pines que controlarán cada fase del motor serán los indicados con las etiquetas IN1…IN4. Esto se traducirá en una salida de voltaje a través de los pines de OUT1…OUT4 para el control de la polaridad de las bobinas.

Posee dos entradas de enable por si se desea regular la velocidad del motor, que deberían estar conectados a 5V, pero como se ve en la figura no están conectados. Este error no se vio hasta que no se hicieron pruebas con la placa, pero tuvo una solución relativamente sencilla: se conectaron dos resistencias de 10 ohmios (a modo de jumper) entre la salida de voltaje del Arduino y las entradas 8 y 14 del controlador.

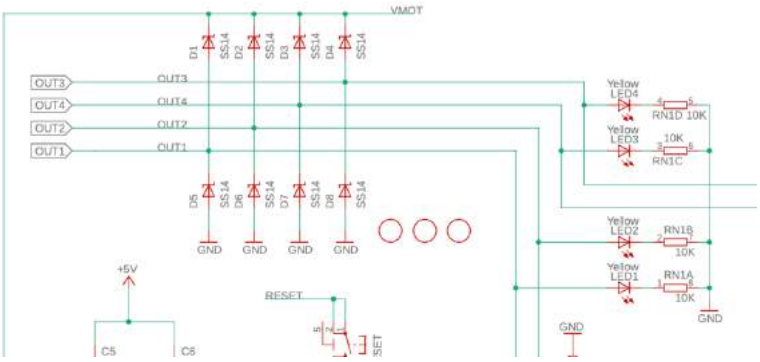


Figura 98.-Circuito de protección de las fases del motor

En la figura 98 se detalla la parte de protección del motor. Se utilizará un puente de diodos por fase, con el objetivo de proteger al driver de los picos de tensión producidos en la bobina al cambiar de polaridad. Los leds que se aprecian sirven para señalar la fase activa y su polaridad.

Motor

El motor que utiliza la rueda de filtros se trata de un motor paso a paso bipolar. Los motores bipolares están constituidos por un único embobinado por fase. El controlador necesitará ser capaz de contrarrestar el polo magnético, consiguiéndose mediante la inversión de la corriente en la bobina. Se utilizará un puente H (por bobina) para este objetivo, en concreto, el ofrecido con el modelo L298N de STMicroelectronics, driver constituido por un doble puente H (*Dual Full Bridge Driver).*

Al contrario que los motores unipolares, el motor paso a paso bipolar tiene dos cables por fase no comunes entre si.

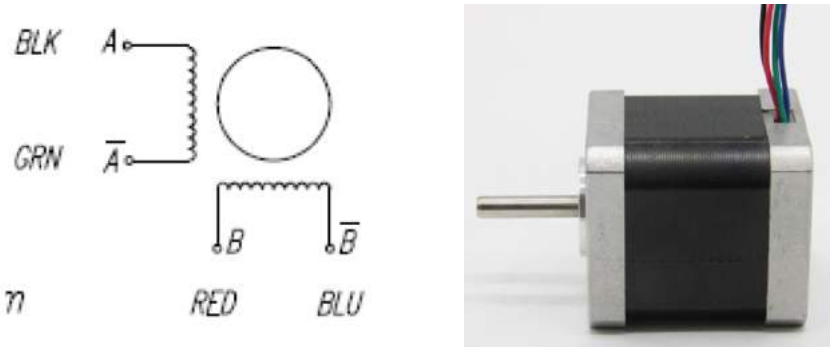


Figura 99.-Estructura interna de un motor paso a paso bipolar

Es de gran importancia que se respete la secuencia de activación de las fases para que se complete un paso del motor, sino puede dar lugar a vibraciones sin movimiento. La librería Stepper.h en arduino ya se encarga de traducir el número de pasos en la secuencia apropiada para el motor.

In_1	In_2	In_3	In_4	Motor A+	Motor A-	Motor B+	Motor B-
1	0	0	0	+	-	-	-
0	1	0	0	-	+	-	-
0	0	1	0	-	-	+	-
0	0	0	1	-	-	-	+

Figura 100.-Secuencia de activación de un motor paso a paso tipo bipolar

Esta secuencia, si va en orden creciente, hará que el motor gire en el sentido horario. De otro modo, empezará a girar en sentido antihorario.

PCB layout

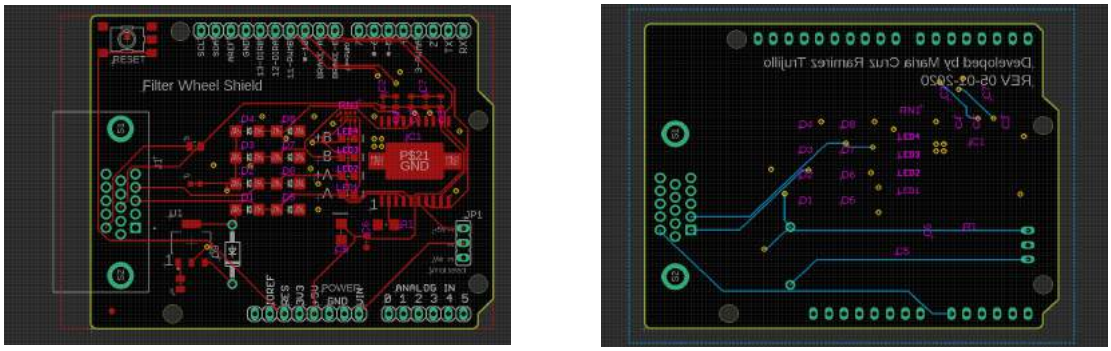


Figura 101.-Top (izquierda) y Bottom (derecha) de la PCB diseñada

Se presenta en este apartado la vista de la PCB (figura 101) una vez hecha la conversión en el programa EAGLE. La placa, una vez hecho el ensamblaje, se verá de la siguiente forma:

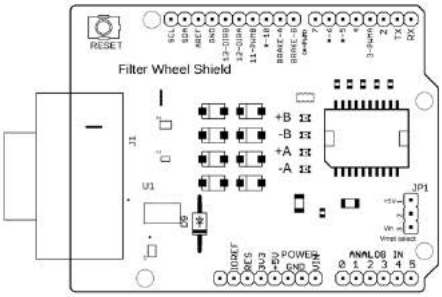


Figura 102.-Placa representada desde EAGLE

Una vez realizada la PCB, se prepara los archivos GERBERS para enviarle la documentación al fabricante. Entre los archivos más importantes que se adjuntaron, fueron:

- Bill of Materials
- Esquemático
- PCB
- Coordenadas para Pick and Place

El resultado final se podrá apreciar en la figura 103.

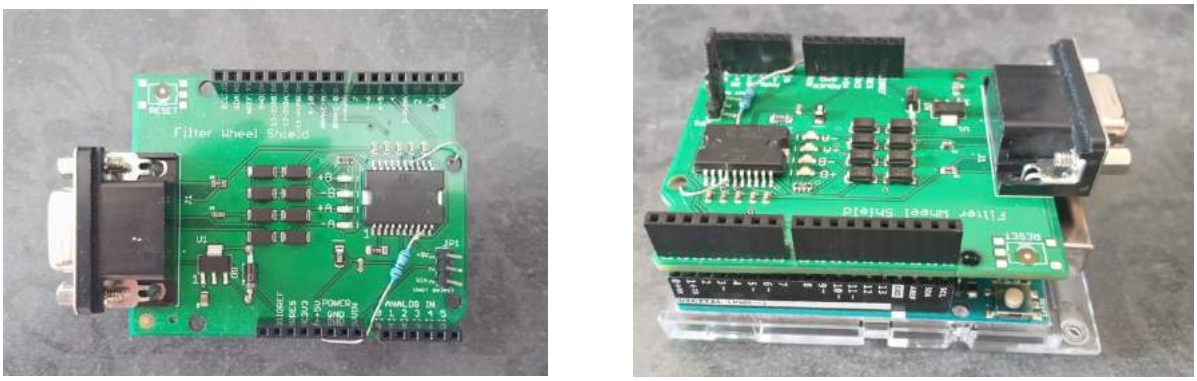


Figura 104.-Montaje Final

4.-Conclusión

Este proyecto ha logrado satisfacer las expectativas propuestas, y superarlas. Se ha convertido, por consiguiente, en realidad el hecho de poder realizar observaciones astronómicas desde un lugar de control remoto, incluso su casa, en tiempo real. Es de vital importancia, en los tiempos que transcurren, el garantizar la disponibilidad de tales herramientas para hacer al científico independiente del lugar donde se halle para poder trabajar.

Surgió la idea del nuevo modo de trabajo mientras se desarrollaba la nueva versión del antiguo programa de FastCam. Conociendo las disposiciones que ofrecía el entorno de desarrollo de Qt 5, más las herramientas de comunicaciones que ofrece el Instituto de Astrofísica de Canarias (refiriéndome aquí a la red VPN), me planteé la idea de convertirlo independiente de la red de internet en donde se halle el ordenador cliente.

Se pueden enumerar las ocasiones donde esta simple característica sería beneficiosa para la investigación. Es un hecho que cada vez más instituciones colaboran entre ellas, como ocurre con el Instituto de Astrofísica de Canarias y la Universidad de Canterbury, universidad situada en Nueva Zelanda. Los científicos de dicha institución podrán beneficiarse, gracias a las ventajas que ofrece este proyecto, de poder realizar observaciones del hemisferio norte sin tener que desplazarse del hemisferio sur.

Las prestaciones de velocidad de transmisión de datos se han comprobado durante el desarrollo del mismo programa. Ya que carecía de posibilidad de trabajar desde el laboratorio de óptica en Tenerife, este software de adquisición se ha terminado de desarrollar desde Cartagena, Murcia. Las prestaciones de vídeo mencionadas están basadas en la realidad de no estar, tan siquiera, cerca de la institución.

No solamente cabe hablar de la única parte del software del detector, ya que se ha logrado, por otro lado, hacer realidad la modulación de instrumentación óptica con Arduino y Qt Creator. Es de gran importancia para el desarrollo propio de proyectos el poder independizarse siempre de los fabricantes, ya que podría ser poco flexible a la hora de adaptarse a ciertas necesidades del proyecto.

Cabe la posibilidad de poder controlarse los elementos ópticos restantes también por internet, como se ha hecho con el detector, pero ya no tenía cabida para este proyecto. Se cierra, así, un proyecto que antes era rígido, inaccesible desde el exterior del observatorio, uno nuevo totalmente modularizado y abierto, extraíble del entorno de trabajo para así, adaptarse a las nuevas necesidades.

Con respecto a mis estudios de grado, este ha sido un proyecto que combina, de forma eficaz, las distintas disciplinas de las que se compone el Grado de Ingeniería Electrónica Industrial y Automática. Gracias a su complejidad, he podido demostrar mis conocimientos en cuanto comunicaciones, diseño de electrónica, desarrollo de software y control instrumental.

Por ejemplo, para poder crear las interfaces gráficas y el desarrollo de algoritmos capaces de trabajar con datos en tiempo real, me ha sido necesario aplicar los conocimientos adquiridos en la asignatura de Programación en Tiempo real. Luego, las asignaturas como Informática Industrial y Microrobótica han sido las principales aportadoras de conocimientos para las comunicaciones entre sistemas electrónicos. Por último, Instrumentación Electrónica y Fundamentos de Electrónica han sido las bases para el desarrollo de electrónica de este proyecto.

Este proyecto, puso a prueba desde el primer momento, mis capacidades para poder abastecer al Instituto de Astrofísica de Canarias de un innovador sistema de control en remoto para los detectores actuales. Al término de este trabajo final de grado dio tiempo de establecer el sistema de adquisición en todos los sistemas operativos disponibles: Linux, Windows y Mac, permitiéndose así su operación en telescopio.

Un segundo prototipo de la placa presentada en este trabajo se encuentra en desarrollo para su segunda versión.

Este proyecto seguirá siendo desarrollado durante este año, y posteriormente, durante mis estudios de máster.

5.-Anexos

5.1.-Corriente de Oscuridad

Un píxel de un detector CCD está formado por una unión semiconductora P-N. Es sabido que un semiconductor tiene la capacidad de generar electrones libres como consecuencia de la agitación térmica de su red cristalina. Estos electrones térmicos se generan de forma espontánea y están presentes siempre, incluso en ausencia de luz. Los electrones térmicos no se pueden distinguir de los fotoelectrones y constituyen una fuente de error.

Denotando con i el número de electrones térmicos almacenados en un píxel por unidad de tiempo, denominándose **corriente de oscuridad** o **corriente térmica**, creciendo su valor exponencialmente con la temperatura. Para un tiempo de exposición t, el número total de electrones térmicos n_{ij} almacenados en el píxel ij vendrá dado por

$$n_{ij} = i_{ij}(T) \cdot t$$

En una aproximación de primer orden se puede considerar correcta esta dependencia lineal con el tiempo. No obstante, fijada la temperatura, los valores i_{ij} oscilarán con una determinada dispersión respecto a un valor medio que detonaremos como $i(T)$. Puede demostrarse que dicha dispersión, que llamaremos **ruido térmico,** es poissoniana. En consecuencia, el ruido térmico vendrá dado por:

$$\sigma_{dark} = \sqrt{i(T) \cdot t} \quad [e^{-}rms]$$

La tabla inferior muestra un ejemplo típico de rango de temperaturas alcanzable dependiendo del modo de enfriamiento utilizado. El modelo de Andor iXon con el que se ha basado este trabajo fin de grado utiliza el modo de enfriamiento "***Ultra-High Cooling***", con aire. Pero es capaz de llegar a -120ºC con ciertas mejoras térmicas.

Moderate Cooling		High Cooling		Ultra-High Cooling	
Air	Water	Air	Water	Air	Water
-5°C	-25°C	-30°C	-55°C	-75°C	-90°C

Figura 105.-Mínimos absolutos de temperatura alcanzables según el tipo de enfriador disponible

Notas:

- Debido a que el enfriamiento rápido o el calentamiento puede ocasionar estrés térmico en la CCD el ratio de enfriamiento y de calentamiento está regulado para que sea menos de 10ºC por minuto en algunos sistemas

- Mientras que se está haciendo control sobre la temperatura, NO se puede empezar el proceso de adquisición, ya que se toman errores de lectura. La cámara entiende que no pueden ocurrir dos estados al mismo tiempo, por lo que siempre se debe cambiar de un estado a otro antes de inicializar un proceso. Es decir, si estás adquiriendo imágenes en vídeo y quieres hacer control de temperatura, debes interrumpir la adquisición. Una vez terminada la adquisición, el sistema retorna al control de temperatura.

5.2.-Presupuesto

JLCPCB

Invoice No: 2921967E20200511231145
Invoice Date: 15/02/2020
Batch No: W202002152338255

JiaLiChuang (HongKong) Co., Limited

JLCPCB
C5DONG3LOU FA HUO BU TONG FU YU GONG YE
YUAN PING DE FU DE GANG.LONG GANG DIST.
SHENZHEN 518000
The People's Republic of China

Telephone: +86 755 23919769
support@jlcpcb.com
https://jlcpcb.com

Total Value:
$58.25

Ship To:

Mari Cruz Ramirez Trujillo

[redacted]

maricruz_ram96@outlook.es
+34651375432
VAT No:

Billing Address:

Mari Cruz Ramirez Trujillo

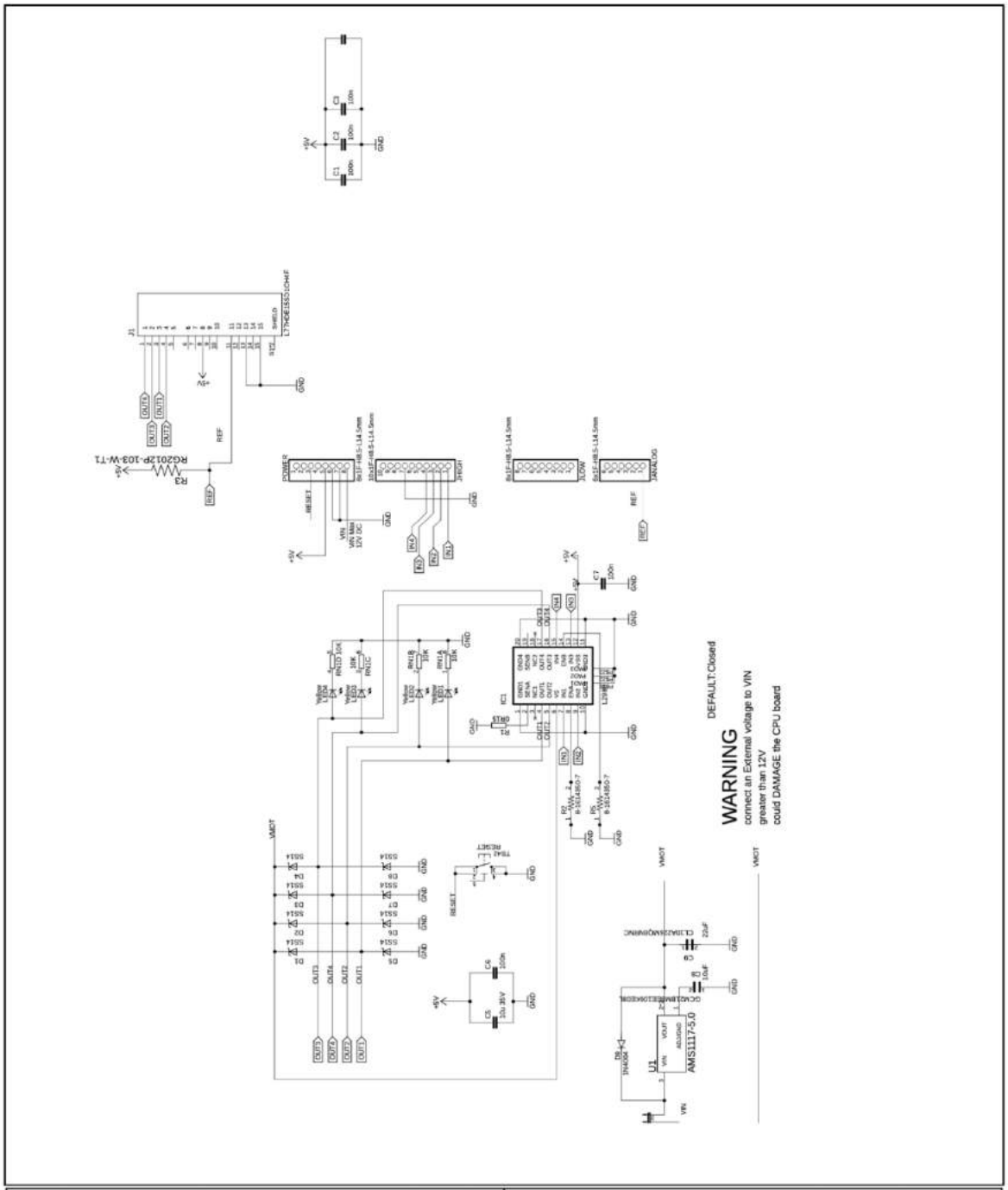
maricruz_ram96@outlook.es
+34651375432

Description of Goods	HS Code	Order Number.	Qty.	Unit Value	Total Value
PCB Assembly	7314500000	SMT020021522184	5	$7.5500	$37.75
PCB Samples	8534009000	Y1	5	$0.4000	$2.00
				SUB-TOTAL	$39.75
				FREIGHT COST	$25.50
				DISCOUNT	-$7.00
				INVOICE TOTAL	$58.25

Attachment

Remarks

5.3.-Esquemático completo



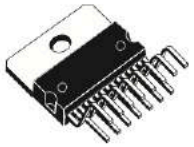
Realizado por: María de la Cruz Ramírez Trujillo Fecha: 12 de Mayo de 2020		Título: Esquemático de Arduino Shield adaptado a ruedas de filtros	
Tamaño: A4	Número de Revisión: 0	Número de hojas: 1	Hoja: 1 de 1
Plano propiedad del Proyecto FastCam			

5.4.-Datashets

L298

DUAL FULL-BRIDGE DRIVER

- OPERATING SUPPLY VOLTAGE UP TO 46 V
- TOTAL DC CURRENT UP TO 4 A
- LOW SATURATION VOLTAGE
- OVERTEMPERATURE PROTECTION
- LOGICAL "0" INPUT VOLTAGE UP TO 1.5 V (HIGH NOISE IMMUNITY)



Multiwatt15

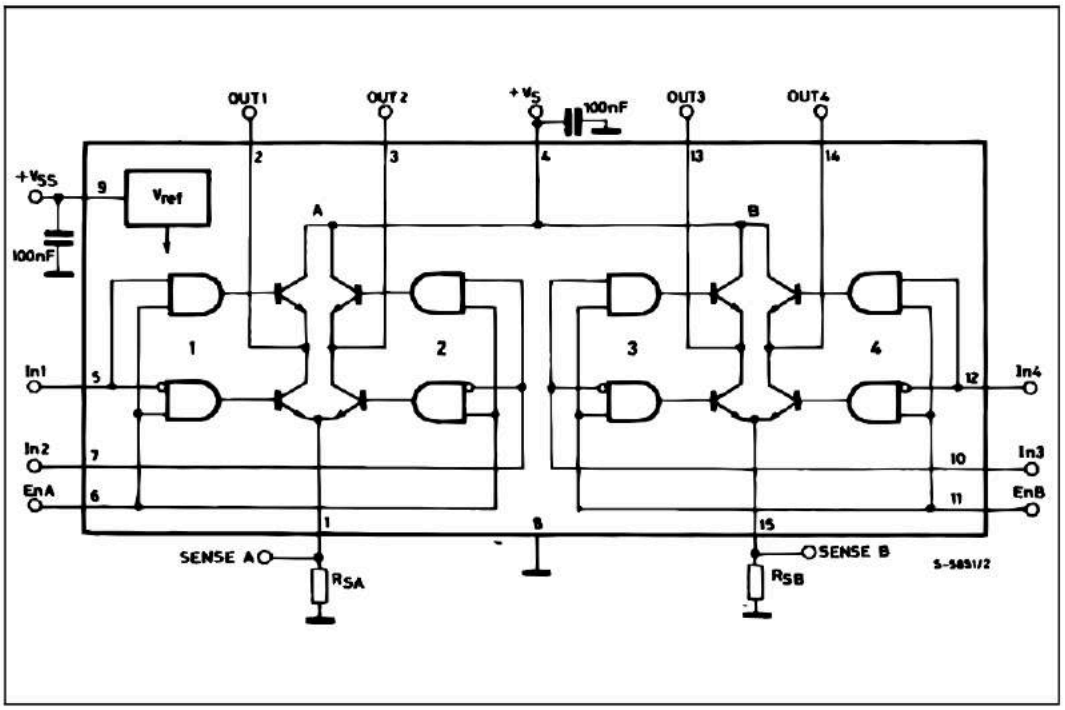
PowerSO20

ORDERING NUMBERS : L298N (Multiwatt Vert.)
L298HN (Multiwatt Horiz.)
L298P (PowerSO20)

DESCRIPTION

The L298 is an integrated monolithic circuit in a 15-lead Multiwatt and PowerSO20 packages. It is a high voltage, high current dual full-bridge driver designed to accept standard TTL logic levels and drive inductive loads such as relays, solenoids, DC and stepping motors. Two enable inputs are provided to enable or disable the device independently of the input signals. The emitters of the lower transistors of each bridge are connected together and the corresponding external terminal can be used for the connection of an external sensing resistor. An additional supply input is provided so that the logic works at a lower voltage.

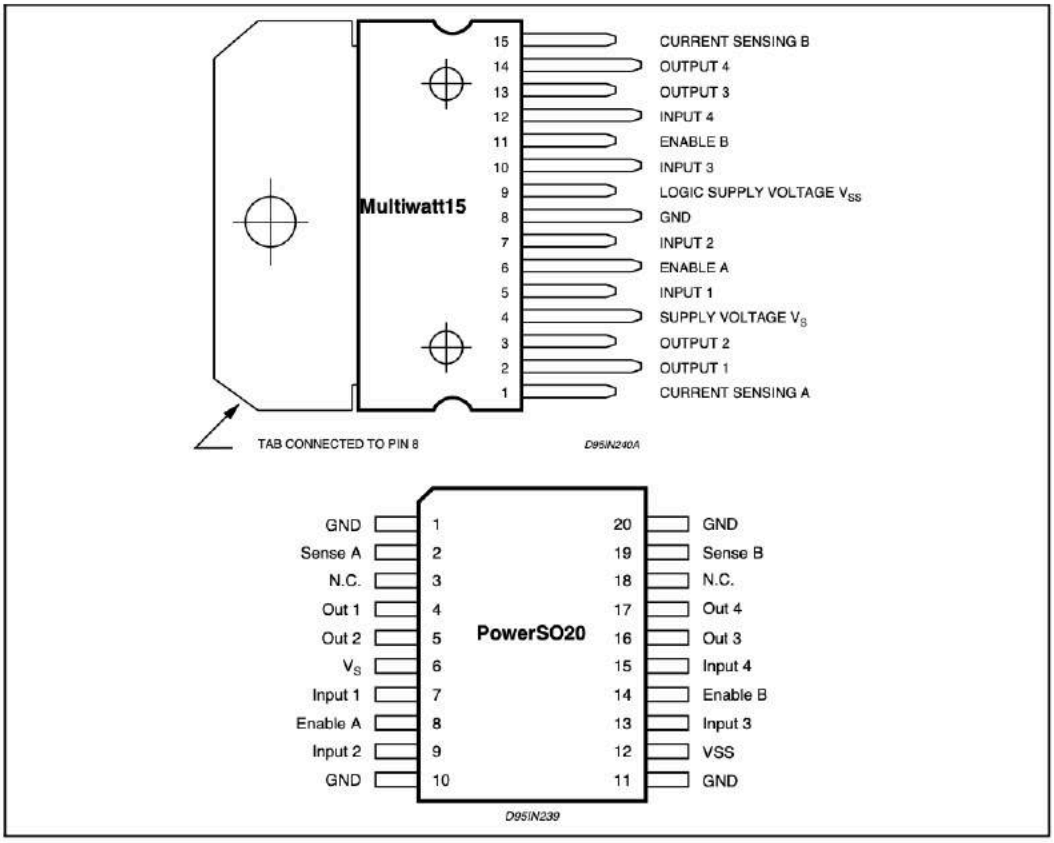
BLOCK DIAGRAM



ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_S	Power Supply	50	V
V_{SS}	Logic Supply Voltage	7	V
V_I, V_{en}	Input and Enable Voltage	–0.3 to 7	V
I_O	Peak Output Current (each Channel) – Non Repetitive (t = 100µs) –Repetitive (80% on –20% off; t_{on} = 10ms) –DC Operation	 3 2.5 2	 A A A
V_{sens}	Sensing Voltage	–1 to 2.3	V
P_{tot}	Total Power Dissipation (T_{case} = 75°C)	25	W
T_{op}	Junction Operating Temperature	–25 to 130	°C
T_{stg}, T_j	Storage and Junction Temperature	–40 to 150	°C

PIN CONNECTIONS (top view)



THERMAL DATA

Symbol	Parameter		PowerSO20	Multiwatt15	Unit
$R_{th\ j\text{-}case}$	Thermal Resistance Junction-case	Max.	–	3	°C/W
$R_{th\ j\text{-}amb}$	Thermal Resistance Junction-ambient	Max.	13 (*)	35	°C/W

(*) Mounted on aluminum substrate

PIN FUNCTIONS (refer to the block diagram)

MW.15	PowerSO	Name	Function
1;15	2;19	Sense A; Sense B	Between this pin and ground is connected the sense resistor to control the current of the load.
2;3	4;5	Out 1; Out 2	Outputs of the Bridge A; the current that flows through the load connected between these two pins is monitored at pin 1.
4	6	V_S	Supply Voltage for the Power Output Stages. A non-inductive 100nF capacitor must be connected between this pin and ground.
5;7	7;9	Input 1; Input 2	TTL Compatible Inputs of the Bridge A.
6;11	8;14	Enable A; Enable B	TTL Compatible Enable Input: the L state disables the bridge A (enable A) and/or the bridge B (enable B).
8	1,10,11,20	GND	Ground.
9	12	VSS	Supply Voltage for the Logic Blocks. A100nF capacitor must be connected between this pin and ground.
10; 12	13;15	Input 3; Input 4	TTL Compatible Inputs of the Bridge B.
13; 14	16;17	Out 3; Out 4	Outputs of the Bridge B. The current that flows through the load connected between these two pins is monitored at pin 15.
–	3;18	N.C.	Not Connected

ELECTRICAL CHARACTERISTICS (V_S = 42V; V_{SS} = 5V, T_j = 25°C; unless otherwise specified)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_S	Supply Voltage (pin 4)	Operative Condition	V_{IH} +2.5		46	V
V_{SS}	Logic Supply Voltage (pin 9)		4.5	5	7	V
I_S	Quiescent Supply Current (pin 4)	V_{en} = H; I_L = 0 V_i = L V_i = H		13 50	22 70	mA mA
		V_{en} = L V_i = X			4	mA
I_{SS}	Quiescent Current from V_{SS} (pin 9)	V_{en} = H; I_L = 0 V_i = L V_i = H		24 7	36 12	mA mA
		V_{en} = L V_i = X			6	mA
V_{iL}	Input Low Voltage (pins 5, 7, 10, 12)		–0.3		1.5	V
V_{iH}	Input High Voltage (pins 5, 7, 10, 12)		2.3		VSS	V
I_{iL}	Low Voltage Input Current (pins 5, 7, 10, 12)	V_i = L			–10	µA
I_{iH}	High Voltage Input Current (pins 5, 7, 10, 12)	V_i = H ≤ V_{SS} –0.6V		30	100	µA
V_{en} = L	Enable Low Voltage (pins 6, 11)		–0.3		1.5	V
V_{en} = H	Enable High Voltage (pins 6, 11)		2.3		V_{SS}	V
I_{en} = L	Low Voltage Enable Current (pins 6, 11)	V_{en} = L			–10	µA
I_{en} = H	High Voltage Enable Current (pins 6, 11)	V_{en} = H ≤ V_{SS} –0.6V		30	100	µA
$V_{CEsat\ (H)}$	Source Saturation Voltage	I_L = 1A I_L = 2A	0.95	1.35 2	1.7 2.7	V V
$V_{CEsat\ (L)}$	Sink Saturation Voltage	I_L = 1A (5) I_L = 2A (5)	0.85	1.2 1.7	1.6 2.3	V V
V_{CEsat}	Total Drop	I_L = 1A (5) I_L = 2A (5)	1.80		3.2 4.9	V V
V_{sens}	Sensing Voltage (pins 1, 15)		–1 (1)		2	V

ELECTRICAL CHARACTERISTICS (continued)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
T_1 (V_i)	Source Current Turn-off Delay	0.5 V_i to 0.9 I_L (2); (4)		1.5		µs
T_2 (V_i)	Source Current Fall Time	0.9 I_L to 0.1 I_L (2); (4)		0.2		µs
T_3 (V_i)	Source Current Turn-on Delay	0.5 V_i to 0.1 I_L (2); (4)		2		µs
T_4 (V_i)	Source Current Rise Time	0.1 I_L to 0.9 I_L (2); (4)		0.7		µs
T_5 (V_i)	Sink Current Turn-off Delay	0.5 V_i to 0.9 I_L (3); (4)		0.7		µs
T_6 (V_i)	Sink Current Fall Time	0.9 I_L to 0.1 I_L (3); (4)		0.25		µs
T_7 (V_i)	Sink Current Turn-on Delay	0.5 V_i to 0.9 I_L (3); (4)		1.6		µs
T_8 (V_i)	Sink Current Rise Time	0.1 I_L to 0.9 I_L (3); (4)		0.2		µs
fc (V_i)	Commutation Frequency	I_L = 2A		25	40	KHz
T_1 (V_{en})	Source Current Turn-off Delay	0.5 V_{en} to 0.9 I_L (2); (4)		3		µs
T_2 (V_{en})	Source Current Fall Time	0.9 I_L to 0.1 I_L (2); (4)		1		µs
T_3 (V_{en})	Source Current Turn-on Delay	0.5 V_{en} to 0.1 I_L (2); (4)		0.3		µs
T_4 (V_{en})	Source Current Rise Time	0.1 I_L to 0.9 I_L (2); (4)		0.4		µs
T_5 (V_{en})	Sink Current Turn-off Delay	0.5 V_{en} to 0.9 I_L (3); (4)		2.2		µs
T_6 (V_{en})	Sink Current Fall Time	0.9 I_L to 0.1 I_L (3); (4)		0.35		µs
T_7 (V_{en})	Sink Current Turn-on Delay	0.5 V_{en} to 0.9 I_L (3); (4)		0.25		µs
T_8 (V_{en})	Sink Current Rise Time	0.1 I_L to 0.9 I_L (3); (4)		0.1		µs

1) 1)Sensing voltage can be –1 V for t ≤ 50 µsec; in steady state V_{sens} min ≥ – 0.5 V.
2) See fig. 2.
3) See fig. 4.
4) The load must be a pure resistor.

Figure 1 : Typical Saturation Voltage vs. Output Current.

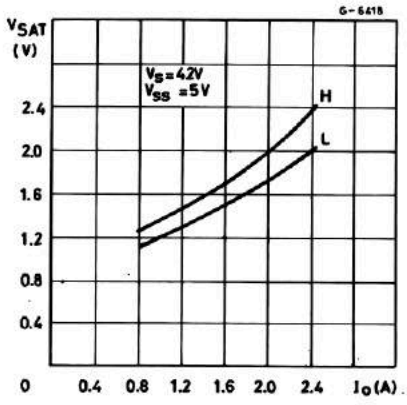
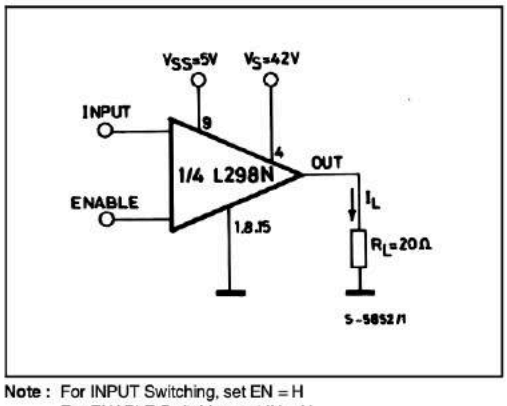


Figure 2 : Switching Times Test Circuits.



Note : For INPUT Switching, set EN = H
For ENABLE Switching, set IN = H

Figure 3 : Source Current Delay Times vs. Input or Enable Switching.

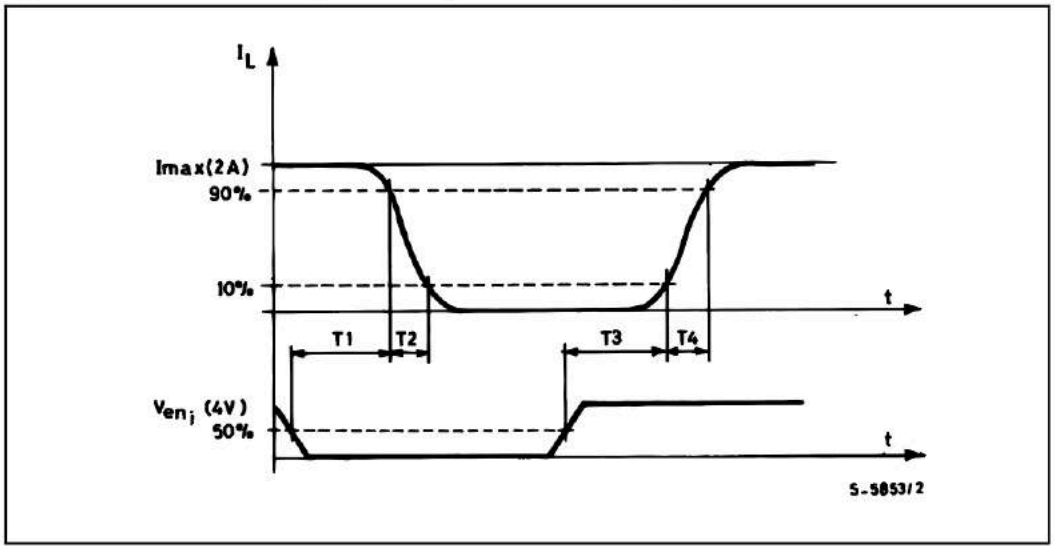
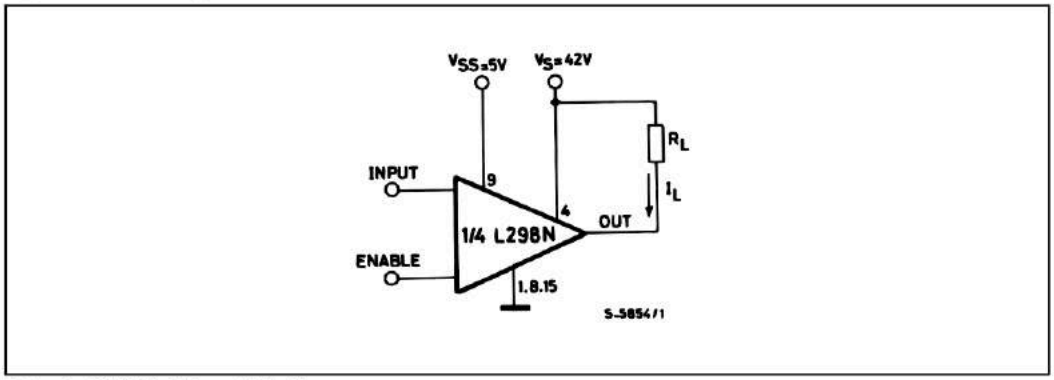


Figure 4 : Switching Times Test Circuits.



Note : For INPUT Switching, set EN = H
For ENABLE Switching, set IN = L

Figure 5 : Sink Current Delay Times vs. Input 0 V Enable Switching.

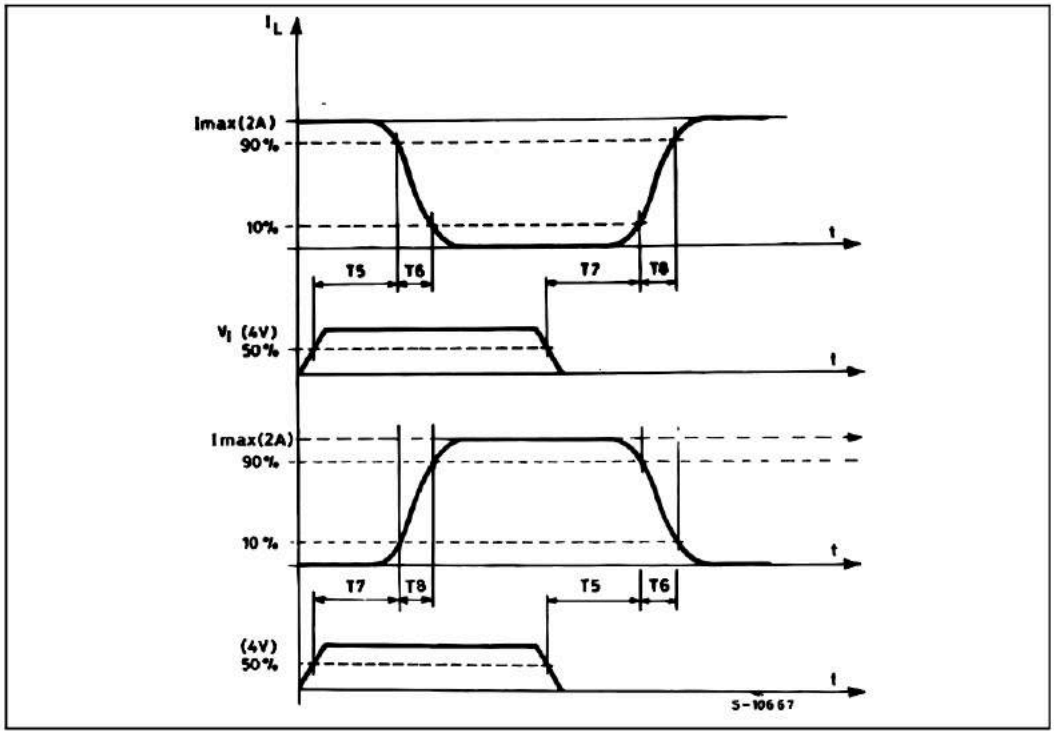
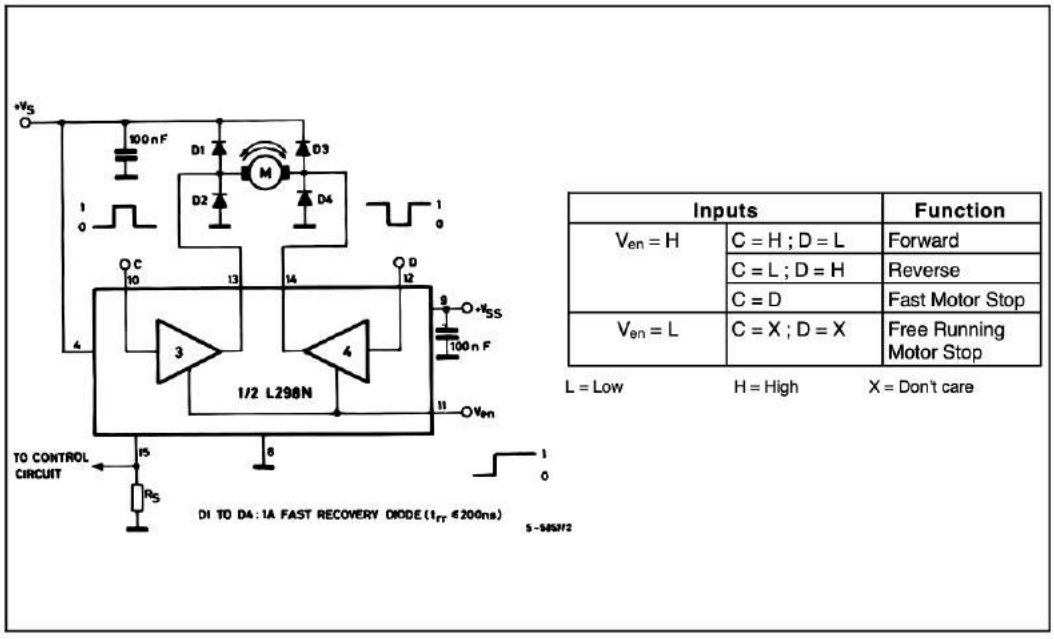


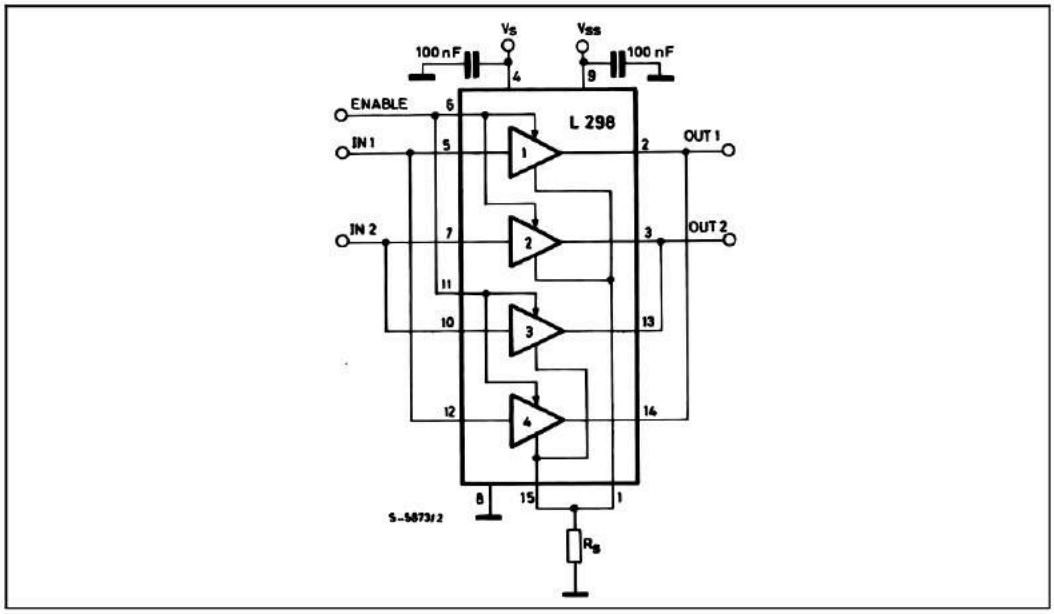
Figure 6 : Bidirectional DC Motor Control.



Inputs		Function
V_{en} = H	C = H ; D = L	Forward
	C = L ; D = H	Reverse
	C = D	Fast Motor Stop
V_{en} = L	C = X ; D = X	Free Running Motor Stop

L = Low H = High X = Don't care

Figure 7 : For higher currents, outputs can be paralleled. Take care to parallel channel 1 with channel 4 and channel 2 with channel 3.



APPLICATION INFORMATION (Refer to the block diagram)

1.1. POWER OUTPUT STAGE

The L298 integrates two power output stages (A ; B). The power output stage is a bridge configuration and its outputs can drive an inductive load in common or differenzial mode, depending on the state of the inputs. The current that flows through the load comes out from the bridge at the sense output : an external resistor (R_{SA} ; R_{SB}.) allows to detect the intensity of this current.

1.2. INPUT STAGE

Each bridge is driven by means of four gates the input of which are In1 ; In2 ; EnA and In3 ; In4 ; EnB. The In inputs set the bridge state when The En input is high ; a low state of the En input inhibits the bridge. All the inputs are TTL compatible.

2. SUGGESTIONS

A non inductive capacitor, usually of 100 nF, must be foreseen between both Vs and Vss, to ground, as near as possible to GND pin. When the large capacitor of the power supply is too far from the IC, a second smaller one must be foreseen near the L298.

The sense resistor, not of a wire wound type, must be grounded near the negative pole of Vs that must be near the GND pin of the I.C.

Each input must be connected to the source of the driving signals by means of a very short path.

Turn-On and Turn-Off : Before to Turn-ON the Supply Voltage and before to Turn it OFF, the Enable input must be driven to the Low state.

3. APPLICATIONS

Fig 6 shows a bidirectional DC motor control Schematic Diagram for which only one bridge is needed. The external bridge of diodes D1 to D4 is made by four fast recovery elements (trr ≤ 200 nsec) that must be chosen of a VF as low as possible at the worst case of the load current.

The sense output voltage can be used to control the current amplitude by chopping the inputs, or to provide overcurrent protection by switching low the enable input.

The brake function (Fast motor stop) requires that the Absolute Maximum Rating of 2 Amps must never be overcome.

When the repetitive peak current needed from the load is higher than 2 Amps, a paralleled configuration can be chosen (See Fig.7).

An external bridge of diodes are required when inductive loads are driven and when the inputs of the IC are chopped ; Shottky diodes would be preferred.

This solution can drive until 3 Amps In DC operation and until 3.5 Amps of a repetitive peak current.

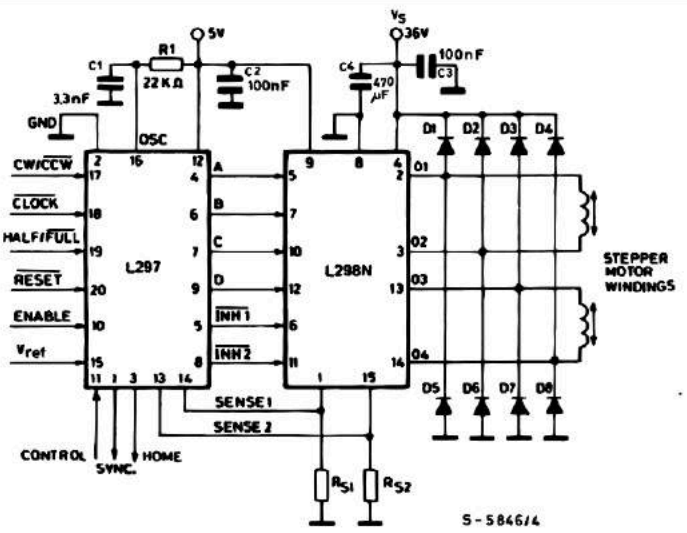
On Fig 8 it is shown the driving of a two phase bipolar stepper motor ; the needed signals to drive the inputs of the L298 are generated, in this example, from the IC L297.

Fig 9 shows an example of P.C.B. designed for the application of Fig 8.

Fig 10 shows a second two phase bipolar stepper motor control circuit where the current is controlled by the I.C. L6506.

Figure 8 : Two Phase Bipolar Stepper Motor Circuit.

This circuit drives bipolar stepper motors with winding currents up to 2 A. The diodes are fast 2 A types.



$R_{S1} = R_{S2} = 0.5\ \Omega$

D1 to D8 = 2 A Fast diodes { $V_F \leq 1.2$ V @ I = 2 A; trr ≤ 200 ns

Figure 9 : Suggested Printed Circuit Board Layout for the Circuit of fig. 8 (1:1 scale).

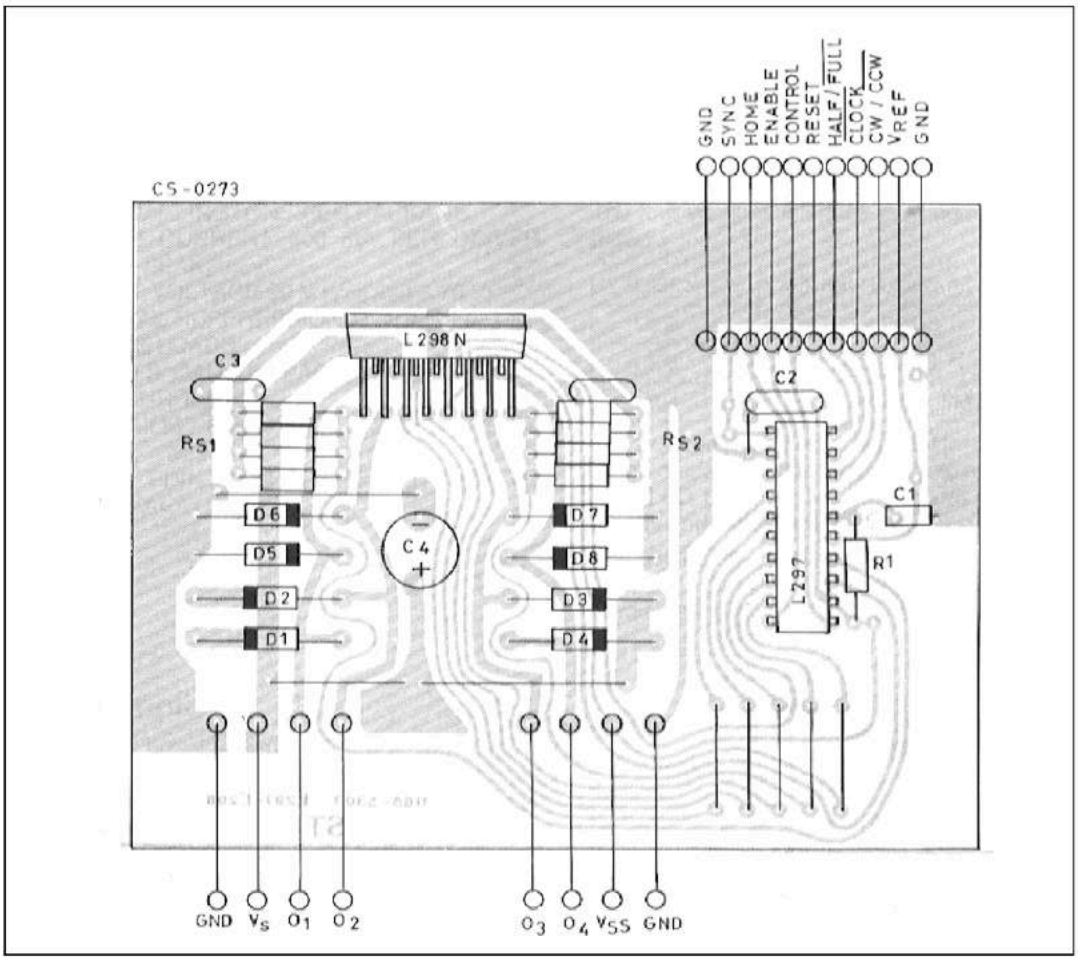
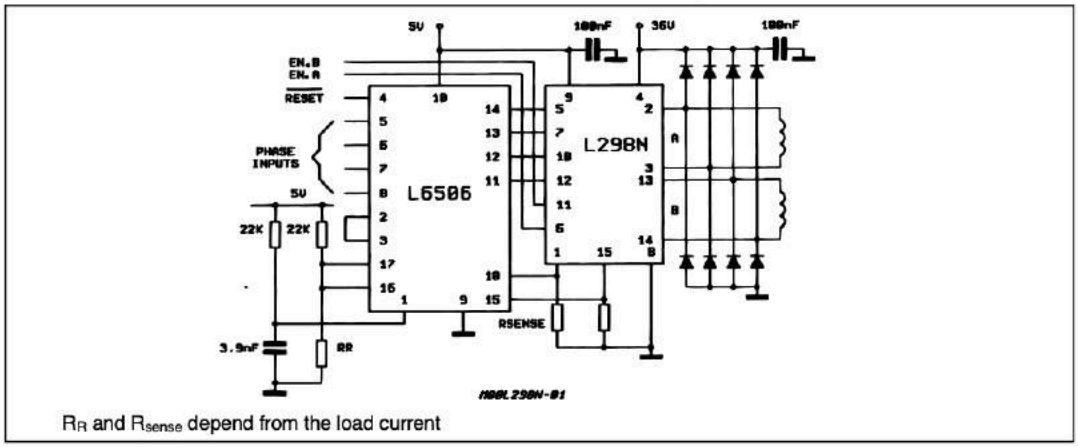


Figure 10 : Two Phase Bipolar Stepper Motor Control Circuit by Using the Current Controller L6506.

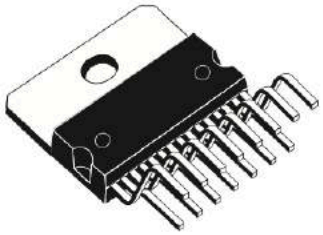


R_R and R_{sense} depend from the load current

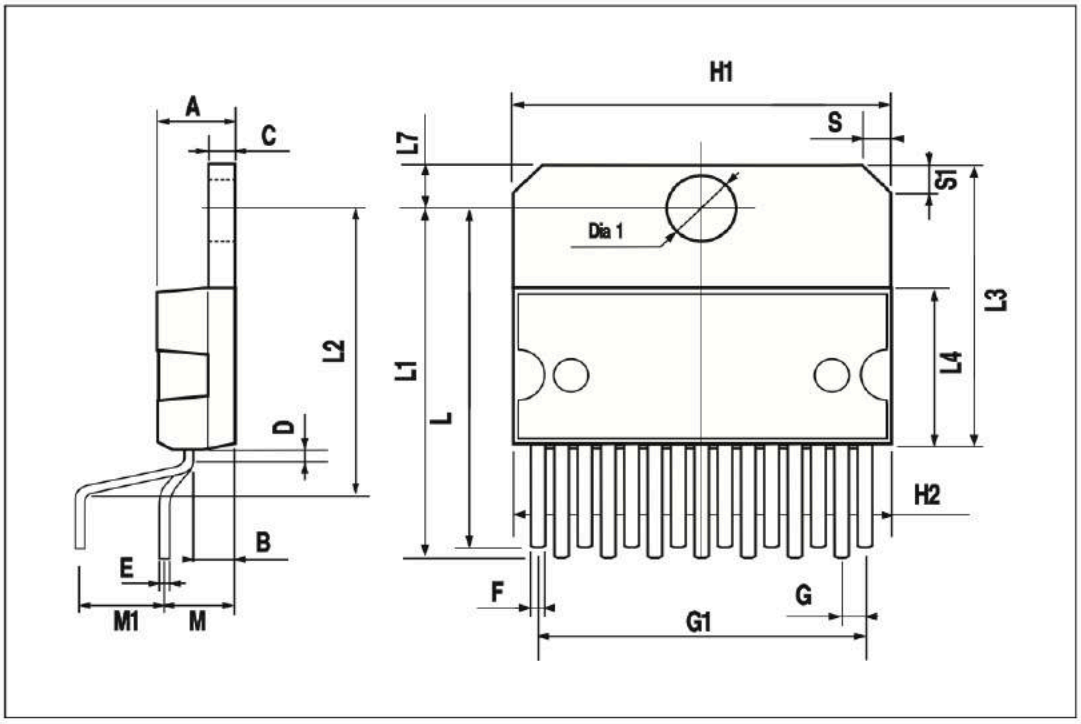
L298

DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
A			5			0.197
B			2.65			0.104
C			1.6			0.063
D		1			0.039	
E	0.49		0.55	0.019		0.022
F	0.66		0.75	0.026		0.030
G	1.02	1.27	1.52	0.040	0.050	0.060
G1	17.53	17.78	18.03	0.690	0.700	0.710
H1	19.6			0.772		
H2			20.2			0.795
L	21.9	22.2	22.5	0.862	0.874	0.886
L1	21.7	22.1	22.5	0.854	0.870	0.886
L2	17.65		18.1	0.695		0.713
L3	17.25	17.5	17.75	0.679	0.689	0.699
L4	10.3	10.7	10.9	0.406	0.421	0.429
L7	2.65		2.9	0.104		0.114
M	4.25	4.55	4.85	0.167	0.179	0.191
M1	4.63	5.08	5.53	0.182	0.200	0.218
S	1.9		2.6	0.075		0.102
S1	1.9		2.6	0.075		0.102
Dia1	3.65		3.85	0.144		0.152

OUTLINE AND MECHANICAL DATA

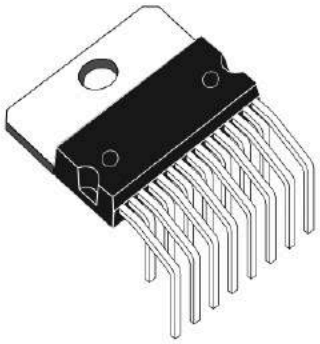


Multiwatt15 V

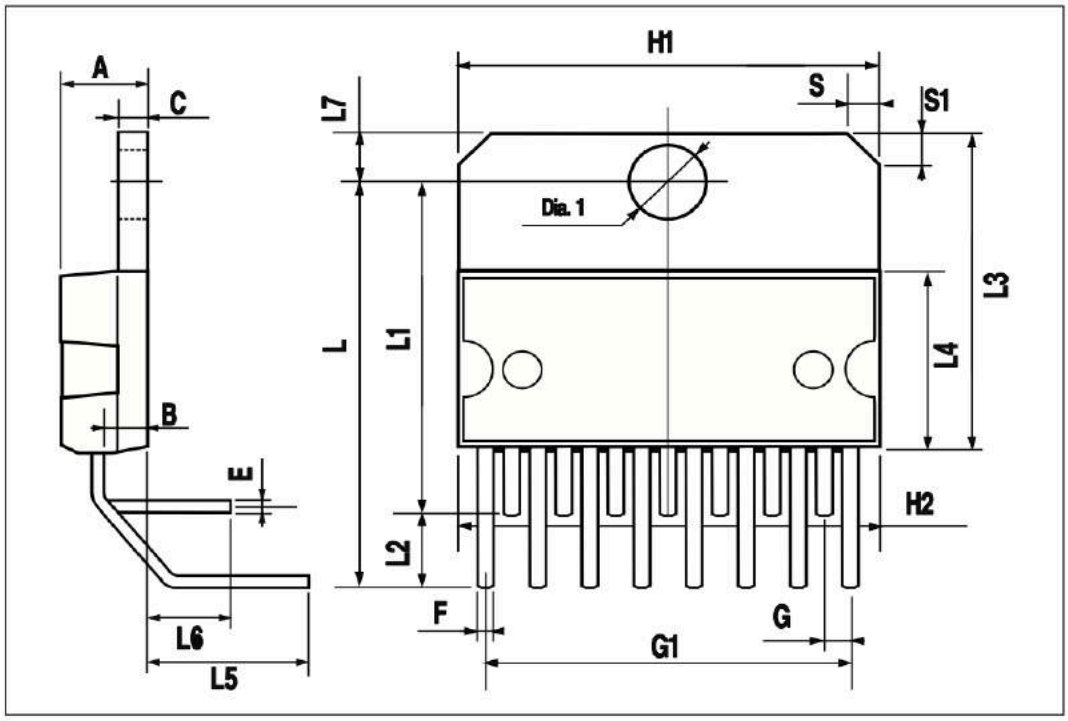


DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
A			5			0.197
B			2.65			0.104
C			1.6			0.063
E	0.49		0.55	0.019		0.022
F	0.66		0.75	0.026		0.030
G	1.14	1.27	1.4	0.045	0.050	0.055
G1	17.57	17.78	17.91	0.692	0.700	0.705
H1	19.6			0.772		
H2			20.2			0.795
L		20.57			0.810	
L1		18.03			0.710	
L2		2.54			0.100	
L3	17.25	17.5	17.75	0.679	0.689	0.699
L4	10.3	10.7	10.9	0.406	0.421	0.429
L5		5.28			0.208	
L6		2.38			0.094	
L7	2.65		2.9	0.104		0.114
S	1.9		2.6	0.075		0.102
S1	1.9		2.6	0.075		0.102
Dia1	3.65		3.85	0.144		0.152

OUTLINE AND MECHANICAL DATA



Multiwatt15 H

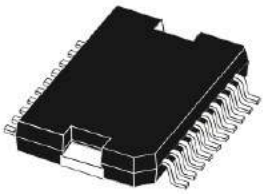


L298

DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
A			3.6			0.142
a1	0.1		0.3	0.004		0.012
a2			3.3			0.130
a3	0		0.1	0.000		0.004
b	0.4		0.53	0.016		0.021
c	0.23		0.32	0.009		0.013
D (1)	15.8		16	0.622		0.630
D1	9.4		9.8	0.370		0.386
E	13.9		14.5	0.547		0.570
e		1.27			0.050	
e3		11.43			0.450	
E1 (1)	10.9		11.1	0.429		0.437
E2			2.9			0.114
E3	5.8		6.2	0.228		0.244
G	0		0.1	0.000		0.004
H	15.5		15.9	0.610		0.626
h			1.1			0.043
L	0.8		1.1	0.031		0.043
N	10° (max.)					
S	8° (max.)					
T		10			0.394	

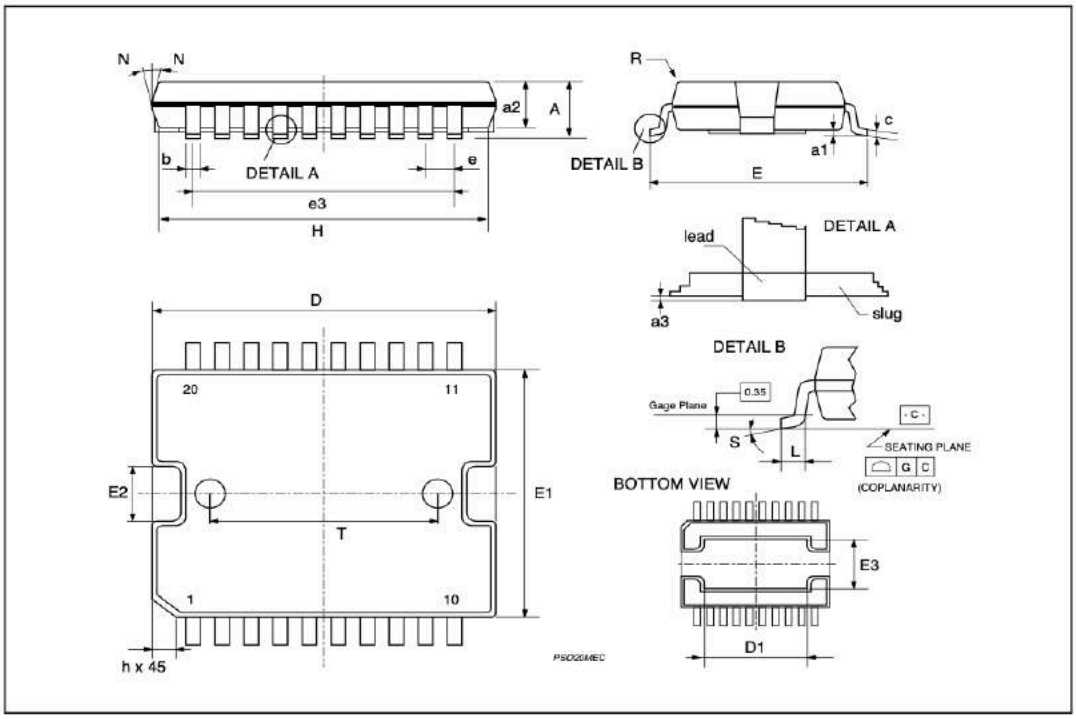
(1) "D and F" do not include mold flash or protrusions.
- Mold flash or protrusions shall not exceed 0.15 mm (0.006").
- Critical dimensions: "E", "G" and "a3"

OUTLINE AND MECHANICAL DATA



JEDEC MO-166

PowerSO20

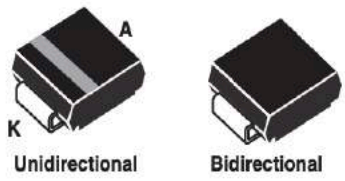


life.augmented

SMA6J

High junction temperature Transil™

Datasheet - production data



Unidirectional Bidirectional

SMA
(JEDEC DO-214AC)

Features

- Peak pulse power:
  - 600 W (10/1000 µs)
  - 4 kW (8/20 µs)
- Stand off voltage range: from 5 V to 188 V
- Unidirectional and bidirectional types
- Low clamping voltage versus standard series
- Low leakage current:
  - 0.2 µA at 25 °C
  - 1 µA at 85 °C
- Operating T_j max: 175 °C
- JEDEC registered package outline

Complies with the following standards

- IEC 61000-4-2 level 4:
  - 15 kV (air discharge)
  - 8 kV (contact discharge)
- MIL STD 883G-Method 3015-7: class3B
  - 25 kV (human body model)

Description

The SMA6J Transil series has been designed to protect sensitive equipment against electro-static discharges according to IEC 61000-4-2, MIL STD 883 Method 3015, and electrical overstress such as IEC 61000-4-4 and 5. They are generally for surges below 600 W 10/1000 µs.

This planar technology makes it compatible with high-end equipment and SMPS where low leakage current and high junction temperature are required to provide reliability and stability over time. Their low clamping voltages provides a better safety margin to protect sensitive circuits with extended life time expectancy.

Packaged in SMA, this minimizes PCB space consumption (SMA footprint in accordance with IPC 7531 standard).

TM: Transil is a trademark of STMicroelectronics

1 Characteristics

Table 1. Absolute ratings (T_{amb} = 25 °C)

Symbol	Parameter		Value	Unit
P_{PP}	Peak pulse power dissipation (1)	T_j initial = T_{amb}	600	W
P	Power dissipation on infinite heatsink	T_{amb} = 55 °C	4	W
I_{FSM}	Non repetitive surge peak forward current for unidirectional types	t_p = 10 ms T_j initial = T_{amb}	60	A
T_{stg}	Storage temperature range		-65 to +175	°C
T_j	Operating junction temperature range		-55 to +175	°C
T_L	Maximum lead temperature for soldering during 10 s		260	°C

1. For a surge greater than the maximum values, the diode will fail in short-circuit.

Table 2. Thermal resistances

Symbol	Parameter	Value	Unit
$R_{th\ (j-l)}$	Junction to leads	30	°C/W
$R_{th\ (j-a)}$	Junction to ambient on printed circuit on recommended pad layout	120	°C/W

Table 3. Electrical characteristics - definitions (T_{amb} = 25 °C)

Symbol	Parameter
V_{RM}	Stand-off voltage
V_{BR}	Breakdown voltage
V_{CL}	Clamping voltage
I_{RM}	Leakage current @ V_{RM}
I_{PP}	Peak pulse current
αT	Voltage temperature coefficient
V_F	Forward voltage drop
R_D	Dynamic resistance

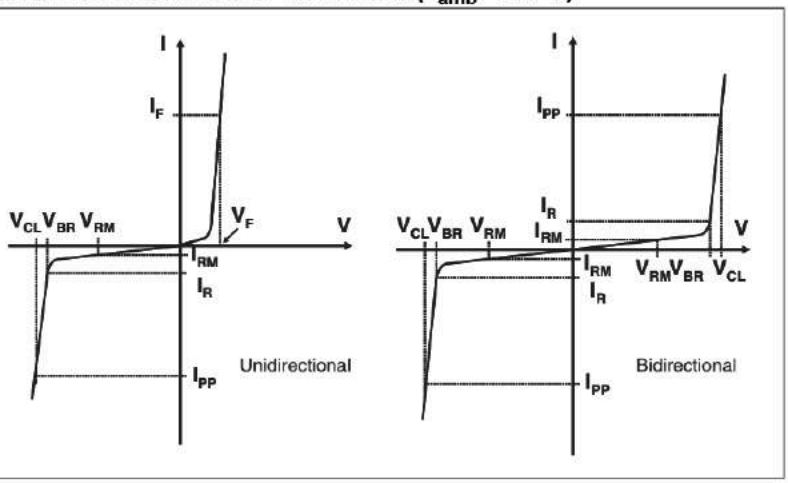


Table 4. Electrical characteristics - values (T_{amb} = 25 °C)

Type	I_{RM} max@V_{RM}			V_{BR} @I_R(1)				V_{CL} @I_{PP} 10/1000 µs		R_D(2) 10/1000 µs	V_{CL} @I_{PP} 8/20 µs		R_D(2) 8/20 µs	αT(3)
	25 °C	85 °C		min	typ	max		max			max			max
	µA		V	V			mA	V	A	Ω	V	A	Ω	10-4/°C
SMA6J5.0A/CA	20	50	5.0	6.40	6.74	7.07	10	9.1	68	0.029	13.4	298	0.021	5.7
SMA6J6.0A/CA	20	50	6.0	6.70	7.05	7.41	10	9.5	61	0.034	13.7	290	0.022	5.9
SMA6J6.5A/CA	20	50	6.5	7.20	7.58	7.96	10	10.2	56	0.040	14.5	276	0.024	6.1
SMA6J8.5A/CA	20	50	8.5	9.4	9.9	10.4	1	13.3	41.7	0.070	19.5	205	0.044	7.3
SMA6J10A/CA	0.2	1	10	11.1	11.7	12.3	1	15.7	37	0.093	21.7	184	0.051	7.8
SMA6J12A/CA	0.2	1	12	13.3	14.0	14.7	1	18.8	31	0.133	25.3	157	0.068	8.3
SMA6J13A/CA	0.2	1	13	14.4	15.2	15.9	1	20.4	29	0.154	27.2	147	0.076	8.4
SMA6J15A/CA	0.2	1	15	16.7	17.6	18.5	1	23.6	25.1	0.206	32.5	123	0.114	8.8
SMA6J18A/CA	0.2	1	18	20.0	21.1	22.1	1	28.3	21.5	0.288	39.3	102	0.168	9.2
SMA6J20A/CA	0.2	1	20	22.2	23.4	24.5	1	31.4	19.4	0.354	42.8	93	0.196	9.4
SMA6J24A/CA	0.2	1	24	26.7	28.1	29.5	1	37.8	16	0.516	50	80	0.256	9.6
SMA6J26A/CA	0.2	1	26	28.9	30.4	31.9	1	40.9	14.9	0.600	53.5	75	0.288	9.7
SMA6J28A/CA	0.2	1	28	31.1	32.7	34.4	1	44.0	13.8	0.697	59	68	0.363	9.8
SMA6J33A/CA	0.2	1	33	36.7	38.6	40.6	1	51.9	11.8	0.963	69	57	0.512	10.0
SMA6J40A/CA	0.2	1	40	44.4	46.7	49.1	1	62.8	9.7	1.42	84	48	0.728	10.1
SMA6J48A/CA	0.2	1	48	53.3	56.1	58.9	1	75.4	8.1	2.04	100	40	1.03	10.3
SMA6J58A/CA	0.2	1	58	64.4	67.8	71.2	1	91.1	6.7	2.97	121	33	1.51	10.4
SMA6J70A/CA	0.2	1	70	77.8	81.9	86.0	1	110	5.5	4.38	146	27	2.22	10.5
SMA6J85A/CA	0.2	1	85	94	99	104	1	134	4.6	6.45	178	22.5	3.29	10.6
SMA6J100A/CA	0.2	1	100	111	117	123	1	157	3.8	9.03	212	19	4.69	10.7
SMA6J130A/CA	0.2	1	130	144	152	159	1	204	3	14.9	265	15	7.03	10.8
SMA6J154A/CA	0.2	1	154	171	180	189	1	242	2.4	22.1	317	12.6	10.2	10.8
SMA6J170A/CA	0.2	1	170	189	199	209	1	275	2.2	30.0	353	11.3	12.7	10.8
SMA6J188A/CA	0.2	1	188	209	220	231	1	328	2	48.5	388	10.3	15.2	10.8

1. Pulse test: t_p <50ms.
2. To calculate maximum clamping voltage at other surge currents, use the following formula

   $V_{CLmax} = R_D \times I_{PP} + V_{BRmax}$
3. To calculate V_{BR} versus junction temperature, use the following formula:

   $V_{BR} @ T_j = V_{BR} @ 25\ °C \times (1 + \alpha T \times (T_j - 25))$

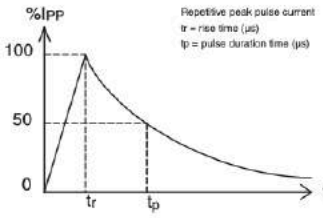


Figure 1. Peak power dissipation versus initial junction temperature

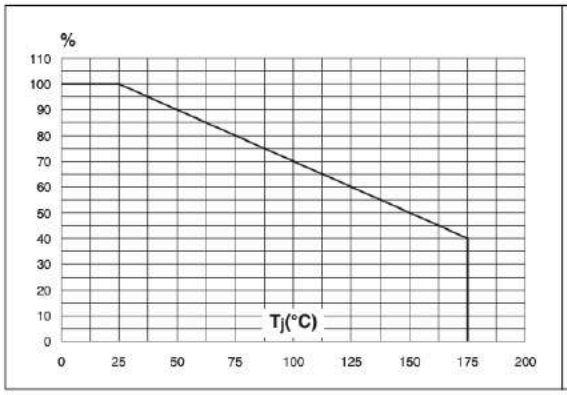


Figure 2. Peak pulse power versus exponential pulse duration (T_j initial = 25 °C)

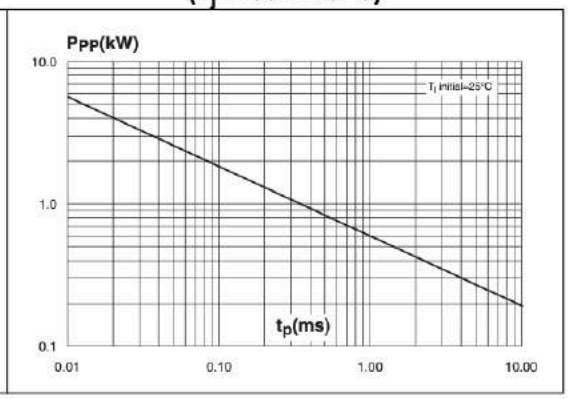


Figure 3. Clamping voltage versus peak pulse current (exponential waveform, maximum values)

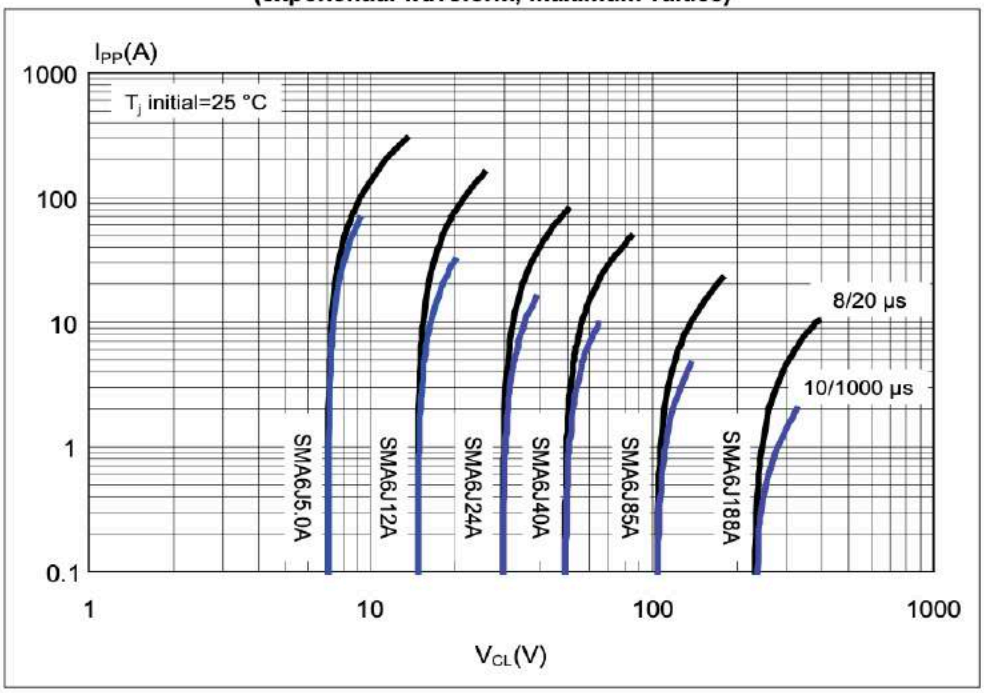


Figure 4. Junction capacitance versus reverse applied voltage (typical values) (SMA6JxxA)

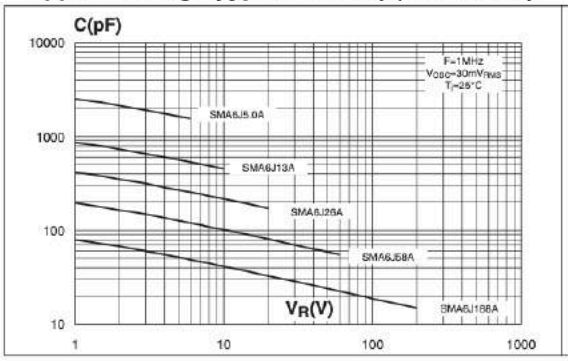


Figure 5. Junction capacitance versus reverse applied voltage (typical values) (SMA6JxxCA)

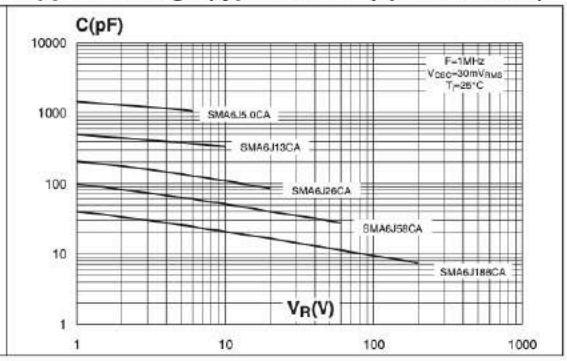


Figure 6. Peak forward voltage drop versus peak forward current (typical values)

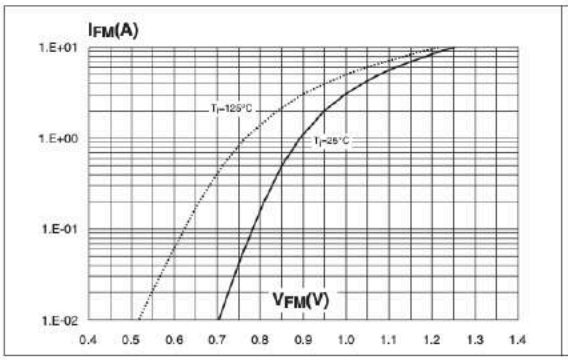


Figure 7. Relative variation of thermal impedance junction to ambient versus pulse duration (printed ciruit board FR4, S_{Cu} = 1 cm^2)

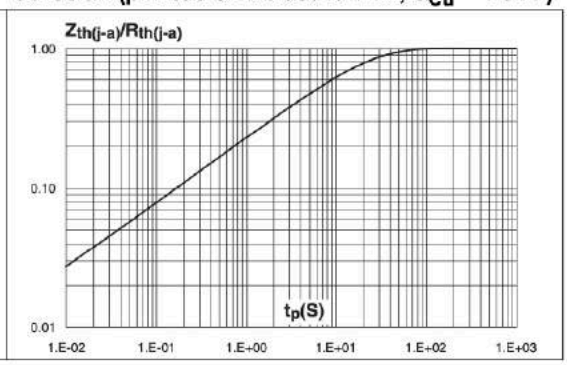


Figure 8. Thermal resistance junction to ambient versus copper surface under each lead (printed circuit board FR4, e_{Cu} = 35 µm)

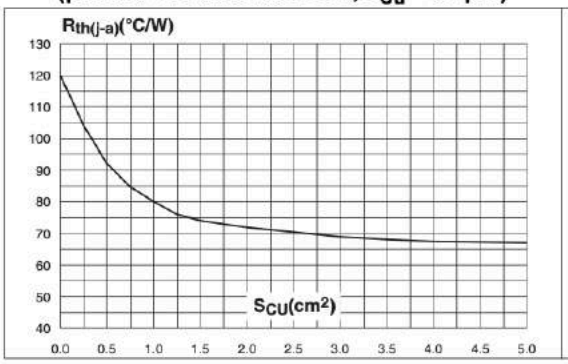
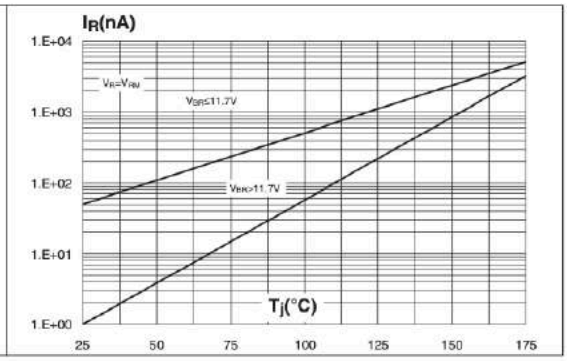
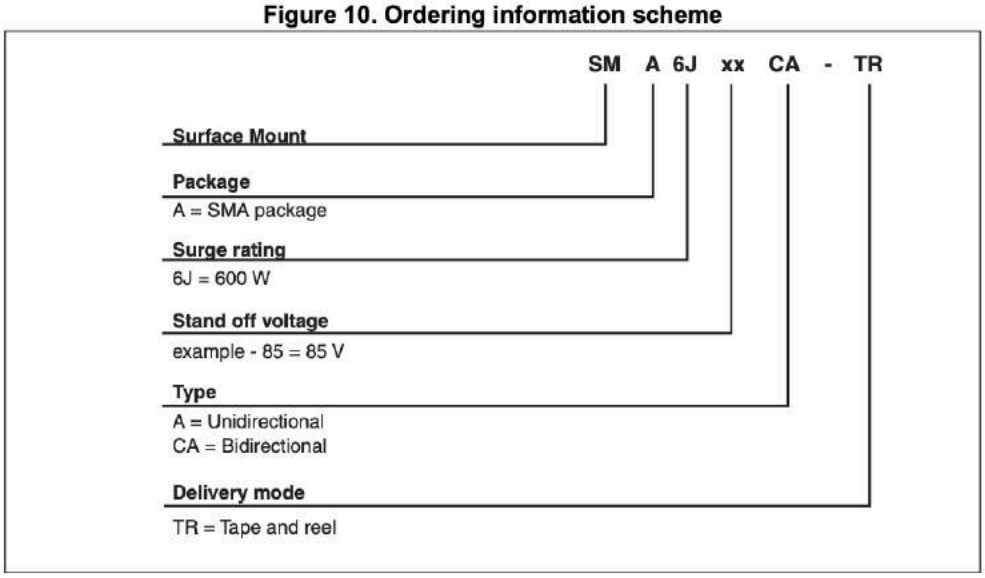


Figure 9. Leakage current versus junction temperature (typical values)



2 Ordering information scheme

Figure 10. Ordering information scheme

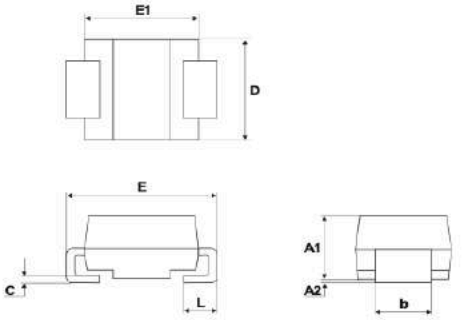


3 Package information

- Case: JEDEC DO-214AC molded plastic over Planar junction
- Terminals: Solder plated, solderable per MIL-STD-750, Method 2026
- Polarity: For unidirectional types the band indicates cathode.
- Flammability: Epoxy is rated UL94V-0
- RoHS package

In order to meet environmental requirements, ST offers these devices in ECOPACK® packages. These packages have a Lead-free second level interconnect . The category of second level interconnect is marked on the package and on the inner box label, in compliance with JEDEC Standard JESD97. The maximum ratings related to soldering conditions are also marked on the inner box label. ECOPACK is an ST trademark. ECOPACK specifications are available at: www.st.com.

Table 5. SMA dimensions



Ref.	Dimensions			
	Millimeters		Inches	
	Min.	Max.	Min.	Max.
A1	1.90	2.03	0.075	0.08
A2	0.05	0.20	0.002	0.008
b	1.25	1.65	0.049	0.065
c	0.15	0.40	0.006	0.016
D	2.25	2.90	0.089	0.114
E	4.80	5.35	0.189	0.211
E1	3.95	4.60	0.156	0.181
L	0.75	1.50	0.030	0.059

Figure 11. SMA footprint dimensions

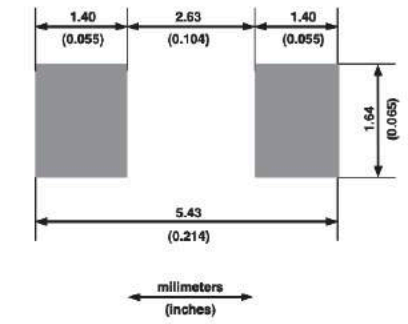


Figure 12. Marking information

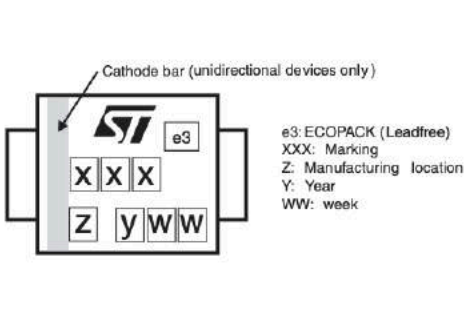


Table 6. Marking

Type	Marking	Type	Marking
SMA6J5.0A-TR	6UA	SMA6J5.0CA-TR	6BA
SMA6J6.0A-TR	6UB	SMA6J6.0CA-TR	6BB
SMA6J6.5A-TR	6UC	SMA6J6.5CA-TR	6BC
SMA6J8.5A-TR	6UD	SMA6J8.5CA-TR	6BD
SMA6J10A-TR	6UE	SMA6J10CA-TR	6BE
SMA6J12A-TR	6UF	SMA6J12CA-TR	6BF
SMA6J13A-TR	6UG	SMA6J13CA-TR	6BG
SMA6J15A-TR	6UH	SMA6J15CA-TR	6BH
SMA6J18A-TR	6UJ	SMA6J18CA-TR	6BJ
SMA6J20A-TR	6UK	SMA6J20CA-TR	6BK
SMA6J24A-TR	6UM	SMA6J24CA-TR	6BM
SMA6J26A-TR	6UN	SMA6J26CA-TR	6BN
SMA6J28A-TR	6UO	SMA6J28CA-TR	6BO
SMA6J33A-TR	6UQ	SMA6J33CA-TR	6BQ
SMA6J40A-TR	6UR	SMA6J40CA-TR	6BR
SMA6J48A-TR	6US	SMA6J48CA-TR	6BS
SMA6J58A-TR	6UT	SMA6J58CA-TR	6BT
SMA6J70A-TR	6UU	SMA6J70CA-TR	6BU
SMA6J85A-TR	6UV	SMA6J85CA-TR	6BV
SMA6J100A-TR	6UW	SMA6J100CA-TR	6BW
SMA6J130A-TR	6UX	SMA6J130CA-TR	6BX
SMA6J154A-TR	6UY	SMA6J154CA-TR	6BY
SMA6J170A-TR	6UZ	SMA6J170CA-TR	6BZ
SMA6J188A-TR	6UAA	SMA6J188CA-TR	6BAA

4 Ordering information

Table 7. Ordering information

Order code[1]	Marking	Package	Weight	Base qty	Delivery mode
SMA6JxxA-TR	See *Table 6.*	SMA	0.072 g	5000	Tape and reel
SMA6JxxCA-TR	See *Table 6.*	SMA	0.072 g	5000	Tape and reel

1. xx indicates stand-off voltage

5 Revision history

Table 8. Document revision history

Date	Revision	Changes
21-Feb-2007	1	First issue.
07-Nov-2007	2	Updated Description. Improved readability of Ordering information scheme. Reformatted to current standards.
04-Aug-2014	3	Updated weight in Table 7.
28-Oct-2015	4	Updated Table 4 and Figure 3.
4-Jul-2017	5	Updated *Table 4*.

IMPORTANT NOTICE – PLEASE READ CAREFULLY

STMicroelectronics NV and its subsidiaries ("ST") reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST's terms and conditions of sale in place at the time of order acknowledgement.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of Purchasers' products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

昆 山 厚 聲 電 子 工 業 有 限 公 司
UNIROYAL ELECTRONICS INDUSTRY (KUNSHAN) CO., LTD.

ISO14001 ISO/TS16949 244546 245468 REG.-Nr.A759 CQC04001010668

UniOhm

Specification for Approval

Customer : 深圳市嘉立創科技發展有限公司

Product Name: LEAD-FREE THICK FILM CHIP RESISTORS

Part Name : CHIP SERIES ±0.5%, ±1%, ±2%, ±5% & 0Ω

21 XIAJIA NORTH ROAD, ECONOMIC AND TECHNICAL DEVELOPMENT ZONE, KUNSHAN CITY, JIANGSU, CHINA 215334
TEL: 86 512 57631411 / 22 / 33
FAX: 86 512 57631431
E-mail: globalsales@uniohm.com localsales@uniohm.com

Approved	Checked	Prepared	File NO.	Edition	Date	Page
William Zhao	Apple Liu	Tang chengxia	JL-01-002	1	2014.2.11	1/14

昆 山 厚 聲 電 子 工 業 有 限 公 司
UNIROYAL ELECTRONICS INDUSTRY (KUNSHAN) CO., LTD.

ISO14001 ISO/TS16949 244546 245468 REG.-Nr.A759 CQC04001010658

UniOhm
We Need it!

Contents

Approved	Checked	Prepared	File NO.	Edition	Date	Page
William Zhao	Apple Liu	Tang chengxia	JL-01-002	1	2014.2.11	2/14

昆 山 厚 聲 電 子 工 業 有 限 公 司
UNIROYAL ELECTRONICS INDUSTRY (KUNSHAN) CO., LTD.

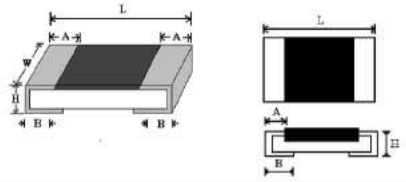
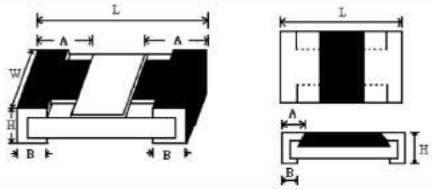
ISO14001 ISO/TS16949 244546 245468 REG.-Nr.A759 CQC04001010656

UniOhm
We Need it!

1.0 Scope:

This specification for approve relates to the Lead-Free Thick Film Chip Resistors manufactured by UNIOHM.

2.0 Ratings & Dimension:

01005、0201、0402	0603、0805、1206、1210、1812、2010、2512
	

2.1 Dimension & Resistance Range：

Type	70℃ Power	Dimension(mm)					Resistance Range			
		L	W	H	A	B	0.5%	1.0%	2.0%	5.0%
01005	1/32W	0.40±0.02	0.20±0.02	0.13±0.02	0.10±0.05	0.10±0.03	--	10Ω-10MΩ	10Ω-10MΩ	10Ω-10MΩ
0201	1/20W	0.60±0.03	0.30±0.03	0.23±0.03	0.10±0.05	0.15±0.05	--	1Ω-10MΩ	1Ω-10MΩ	1Ω-10MΩ
0402	1/16W	1.00±0.10	0.50±0.05	0.35±0.05	0.20±0.10	0.25±0.10	1Ω-10MΩ	0.2Ω~22MΩ	0.2Ω~22MΩ	0.2Ω~22MΩ
0603	1/10W	1.60±0.10	0.80±0.10	0.45±0.10	0.30±0.20	0.30±0.20	1Ω-10MΩ	0.1Ω~33MΩ	0.1Ω~33MΩ	0.1Ω~100MΩ
0805	1/8W	2.00±0.15	1.25 +0.15 -0.10	0.55±0.10	0.40±0.20	0.40±0.20	1Ω-10MΩ	0.1Ω~33MΩ	0.1Ω~33MΩ	0.1Ω~100MΩ
1206	1/4W	3.10±0.15	1.55 +0.15 -0.10	0.55±0.10	0.45±0.20	0.45±0.20	1Ω-10MΩ	0.1Ω~33MΩ	0.1Ω~33MΩ	0.1Ω~100MΩ
1210	1/3W 1/2W	3.10±0.10	2.60±0.20	0.55±0.10	0.50±0.25	0.50±0.20	1Ω-10MΩ	0.1Ω~10MΩ	0.1Ω~22MΩ	0.1Ω~100MΩ
1812	1/2W 3/4W-S	4.50±0.20	3.20±0.20	0.55±0.20	0.50±0.20	0.50±0.20	1Ω-10MΩ	0.11Ω-10MΩ	0.1Ω-10MΩ	0.1Ω-10MΩ
2010	1/2W 3/4W-S	5.00±0.10	2.50±0.20	0.55±0.10	0.60±0.25	0.50±0.20	1Ω-10MΩ	0.1Ω~22MΩ	0.1Ω~22MΩ	0.1Ω~22MΩ
2512	1W	6.35±0.10	3.20±0.20	0.55±0.10	0.60±0.25	0.50±0.20	1Ω-10MΩ	0.1Ω~33MΩ	0.1Ω~33MΩ	0.1Ω~33MΩ

昆 山 厚 聲 電 子 工 業 有 限 公 司
UNIROYAL ELECTRONICS INDUSTRY (KUNSHAN) CO., LTD.

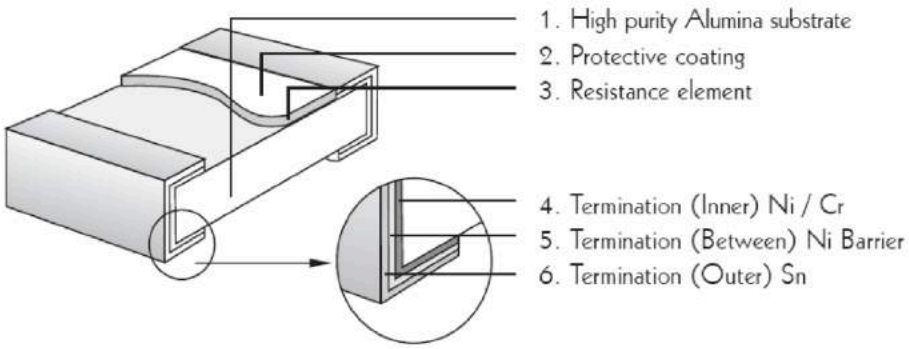
ISO14001 ISO/TS16949 244546 245468 REG.-Nr.A759 CQC04001010658

UniOhm

2.2 Ratings

Type	70℃ Power	Max. Working Voltage	Max. Overload Voltage	Dielectric withstanding Voltage	Resistance Value of Jumper	Rated Current of Jumper	Max. Rated Current of Jumper	Operating Temperature
01005	1/32W	15V	30V	--	<50mΩ	--	--	-55℃~155℃
0201	1/20W	25V	50V	--	<50mΩ	0.5A	1A	-55℃~155℃
0402	1/16W	50V	100V	100V	<50mΩ	1A	2A	-55℃~155℃
0603	1/10W	50V	100V	300V	<50mΩ	1A	2A	-55℃~155℃
0805	1/8W	150V	300V	500V	<50mΩ	2A	5A	-55℃~155℃
1206	1/4W	200V	400V	500V	<50mΩ	2A	10A	-55℃~155℃
1210	1/3W 1/2W	200V	500V	500V	<50mΩ	2A	10A	-55℃~155℃
1812	1/2W 3/4W-S	200V	500V	500V	<50mΩ	2A	10A	-55℃~155℃
2010	1/2W 3/4W-S	200V	500V	500V	<50mΩ	2A	10A	-55℃~155℃
2512	1W	200V	500V	500V	<50mΩ	2A	10A	-55℃~155℃

3.0 Structure:



Approved	Checked	Prepared	File NO.	Edition	Date	Page
William Zhao	Apple Liu	Tang chengxia	JL-01-002	1	2014.2.11	5/14

4.0 Marking:

(1) For 01005、0201 and 0402 size. Due to the very small size of the resistor's body, there is no marking on the body.

Example:

01005、0201、0402

(2) ±2%,±5%Tolerance:The first two digits are significant figures of resistance and the third denotes number of zeros following

Example:

33000 → 33KΩ

(3) ±2%、±5%Tolerance: Below 10Ω show as following, read alphabet"R" as decimal point.

Example:

2R2 → 2.2Ω

(4) ±0.5%、±1% Tolerance: 4 digits, first three digits are significant; forth digit is number of zeros. Letter r is decimal point.

2701 → 2.7KΩ

10R0 → 10Ω

(5) standard E-24 and not belong to E-96 series values(in ±0.5%、±1%tolerance)of 0603 size the marking is the same as 5% tolerance but marking as underline

333=33000→33KΩ

680→68Ω

(6) Product below 1Ω,show as following, the first digit is "R" which as decimal point.

R30→0.3Ω

Approved	Checked	Prepared	File NO.	Edition	Date	Page
William Zhao	Apple Liu	Tang chengxia	JL-01-002	1	2014.2.11	6/14

(7) Standard E-96 series values (±0.5% 、±1% tolerance) of 0603 size. Due the small size of the resistor's body, 3 digits marking will be used to indicate the accurate resistance value by using the following multiplier & resistance code.

Multiplier code:

Code	A	B	C	D	E	F	G	H	X	Y	Z
Multiplier	10^0	10^1	10^2	10^3	10^4	10^5	10^6	10^7	10^{-1}	10^{-2}	10^{-3}

Coding formula

First two digits----------Resistance code Third digit----------Multiplier code

EXAMPLE: 1.96KΩ=196×10^1Ω------29B 12.4Ω=124×10^{-1}Ω-----10X

STANDARD E-96 VALUES AND 0603 RESISTANCE CODE

Ω VALUE	CODE	Ω VALUE	CODE	Ω VALUE	CODE	Ω VALUE	CODE
100	01	178	25	316	49	562	73
102	02	182	26	324	50	576	74
105	03	187	27	332	51	590	75
107	04	191	28	340	52	604	76
110	05	196	29	348	53	619	77
113	06	200	30	357	54	634	78
115	07	205	31	365	55	649	79
118	08	210	32	374	56	665	80
121	09	215	33	383	57	681	81
124	10	221	34	392	58	698	82
127	11	226	35	402	59	715	83
130	12	232	36	412	60	732	84
133	13	237	37	422	61	750	85
137	14	243	38	432	62	768	86
140	15	249	39	442	63	787	87
143	16	255	40	453	64	806	88
147	17	261	41	464	65	825	89
150	18	267	42	475	66	845	90
154	19	274	43	487	67	866	91
158	20	280	44	499	68	887	92
162	21	287	45	511	69	909	93
165	22	294	46	523	70	931	94
169	23	301	47	536	71	953	95
174	24	309	48	549	72	976	96

(8) 0Ω Marking:

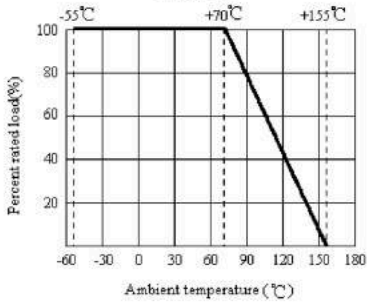
Normally for 01005、0201 and 0402 size, no marking on the body:

Normally, the making of 0Ω 0603, 0Ω 0805, 0Ω 1206, 0Ω 1210, 0Ω 1812, 0Ω 2010, 0Ω 2512 resistors as following

5.0 Derating Curve:

Resistors shall have a power rating based on continuous load operation at an ambient temperature from -55℃ to 70℃. For temperature in excess of 70℃, the load shall be derate as shown in figure 1

Figure 1



5.1 Voltage rating:

Resistors shall have a rated direct-current (DC) continuous working Voltage or an approximate sine-wave root-mean-square (RMS) alternating-current (AC) continuous working voltage at commercial-line frequency and waveform corresponding to the power rating, as determined from the following formula:

$$RCWV = \sqrt{P \times R}$$

Where: RCWV commercial-line frequency and waveform (Volt.)
P = power rating (WATT.) R = nominal resistance (OHM)

In no case shall the rated DC or RMS AC continuous working voltage be greater than the applicable maximum value.

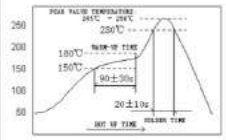
The overload voltage is 2.5 times RCWV or Max. Overload voltage whichever is less.

昆 山 厚 聲 電 子 工 業 有 限 公 司
UNIROYAL ELECTRONICS INDUSTRY (KUNSHAN) CO., LTD.

ISO14001 ISO/TS16949 244546 245468 REG.-Nr.A759 CQC04001016858

UniOhm

6.0 Performance Specification:

Characteristic	Limits		Test Method (JIS-C-5201& JIS-C-5202)
◎Temperature Coefficient	**01005:** 10Ω≦ R ≤100Ω: ±400PPM/℃ >100Ω: ±250PPM/℃ **0201:** 1Ω≦ R ≤10Ω: ±400PPM/℃ >10Ω:±200PPM/℃ **0402~2512 :** <1Ω ≤±800PPM/°C 1Ω≦ R ≤10Ω ≤±400PPM/°C 10Ω< R ≦100Ω ≤±200PPM/°C 100Ω<R<10MΩ ≤±100PPM/°C 10MΩ ≦R<100MΩ≤±200PPM/°C		4.8 Natural resistance changes per temp. Degree centigrade $\frac{R_2-R_1}{R_1(T_2-T_1)} \times 10^6$ (PPM/℃) R_1: resistance value at room temp. (T_1) R_2: resistance value at room temp. +100℃ (T_2) Test pattern: room temp. (T_1), room temp. +100℃(T_2)
◎ *Short-time overload	±0.5%,±1%	±(1%+0.1Ω) Max.	4.13 Permanent resistance change after the application of a potential of 2.5 times RCWV or Max. Overload Voltage whichever less for 5 seconds..
	±2%,±5%	±(2%+0.1Ω) Max.	
	01005	±(2%+0.1Ω) Max	
	* <50mΩ		Apply max Overload current for 0Ω
* Dielectric withstanding voltage	No evidence of flashover mechanical damage, arcing or insulation breaks down.		4.7 Resistors shall be clamped in the trough of a 90℃ metallic v-block and shall be tested at ac potential respectively specified in the given list of each product type for 60-70 seconds.
◎ *Solderability	95% coverage Min.		Wave solder: Test temperature of solder: 245℃±3℃ dipping time in solder: 2-3 seconds.
	Go up tin rate bigger than half of end pole		Reflow: 
◎ Temperature cycling	±0.5%,±1%	±(0.5%+0.05 Ω)Max	4.19Resistance change after continuous five cycles for duty cycle specified below:
	±2%,±5%	±(1.0%+0.05 Ω) Max	
	01005	±(1%+0.05Ω) Max	
◎ Soldering heat	Resistance change rate is: ±(1%+0.05Ω) Max		4.18 Dip the resistor into a solder bath having a temperature of 260℃±5℃ and hold it for 10±1 seconds.

Step	Temperature	Time
1	-55℃±3℃	30 mins
2	Room temp.	10 --- 15 mins
3	+155℃±2℃	30 mins
4	Room temp.	10 --- 15 mins

Terminal bending	±(1%+0.05Ω) Max		4.33 Twist of test board: Y/x = 3/90 mm for 60Seconds
* Insulation resistance	1,000 MΩ or more		4.6 the measuring voltage shall be ,measured with a direct voltage of (100±15)V or a voltage equal to the dielectric withstanding voltage., and apply for 1min
◎ Humidity (steady state)	±0.5%,±1%	±(0.5%+0.1Ω) Max.	4.24Temporary resistance change after 240 hours exposure in a humidity test chamber controlled at 40±2℃ and 90-95% relative humidity,
	±2%,±5%	±(3.0%+0.1Ω) Max.	
	01005	±(3.0%+0.05Ω) Max.	
◎ *Load life in humidity	±0.5%,±1%	±(1%+0.1Ω)max.	7.9 Resistance change after 1,000 hours (1.5 hours "ON",0.5 hour "OFF") at RCWV in a humidity chamber controlled at 40℃±2℃ and 90 to 95% relative humidity.
	±2%,±5%	±(3%+0.1Ω)Max.	
	01005	±(3.0%+0.05Ω) Max.	
	* <50mΩ		Apply to rated current for 0Ω
◎ *Load life	±0.5%,±1%	±(1%+0.1Ω)Max.	4.25.1 Permanent resistance change after 1,000 hours operating at RCWV with duty cycle 1.5 hours "ON", 0.5 hour "OFF" at 70℃±2℃ ambient.
	±2%,±5%	±(3%+0.1Ω)Max	
	01005	±(3.0%+0.05Ω) Max.	
	* <50mΩ		Apply to rated current for 0Ω
The resistors of 0Ω only can do the characteristic noted of * The resistors of 01005 & 0201 only can do the characteristic noted of ◎			

7.0 Explanation of Part No. System:

The standard Part No. includes 14 digits with the following explanation:

7.1 1st~4th digits

This is to indicate the Chip Resistor.

Example: 01005, 0201, 0402, 0603,0805,1206,1210,2010,1812,2512

7.2 5th~6th digits:

7.2.1 This is to indicate the wattage or power rating. To dieting the size and the numbers, The following codes are used; and please refer to the following chart for detail:

W=Normal Size; S=Small Size; U= Ultra Small Size; "1" ~ "G" to denotes "1" ~ "16" as Hexadecimal:

1/16W~1W:

Wattage	1/32	3/4W	1/2	1/3	1/4	1/8	1/10	1/16	1/20W	1
Normal Size	WH	07	W2	W3	W4	W8	WA	WG	WM	1W
Small Size	/	07	S2	S3	S4	S8	SA	SG	/	1S
Ultra Small Size	/	/	U2	U3	U4	U8	UA	UG	/	1U

7.2.2 For power rating less or equal to 1 watt, the 5th digit will be the letters W or S to represent the size required & the 6th digit will be a number or a letter code.

Example: WA=1/10W; S4=1/4W-S

7.3 The 7th digit is to denote the Resistance Tolerance. The following letter code is to be used for indicating the standard Resistance Tolerance.

D=±0.5% F=±1% G=±2% J=±5% K= ±10%

7.4 The 8th to 11th digits is to denote the Resistance Value.

7.4.1 For the standard resistance values of 5%&10% series, the 8th digit is "0" ,the 9th & 10th digits are to denote the significant figures of the resistance and the 11th digit is the number of zeros following;

For the standard resistance values of ≤2% series in, the 8th digit to the 10th digits is to denote the significant figures of the resistance and the 11th digit is the zeros following.

7.4.2 The following number s and the letter codes are to be used to indicate the number of zeros in the 11th digit:

$0=10^0$ $1=10^1$ $2=10^2$ $3=10^3$ $4=10^4$ $5=10^5$ $6=10^6$ $J=10^{-1}$ $K=10^{-2}$ $L=10^{-3}$ $M=10^{-4}$

7.4.3 The 12th, 13th & 14th digits.

The 12th digit is to denote the Packaging Type with the following codes:

C=Bulk in (Chip Product) T=Tape/Reel

7.4.4 The 13th digit is normally to indicate the Packing Quantity of Tape/Reel packaging types. The following letter code is to be used for some packing quantities:

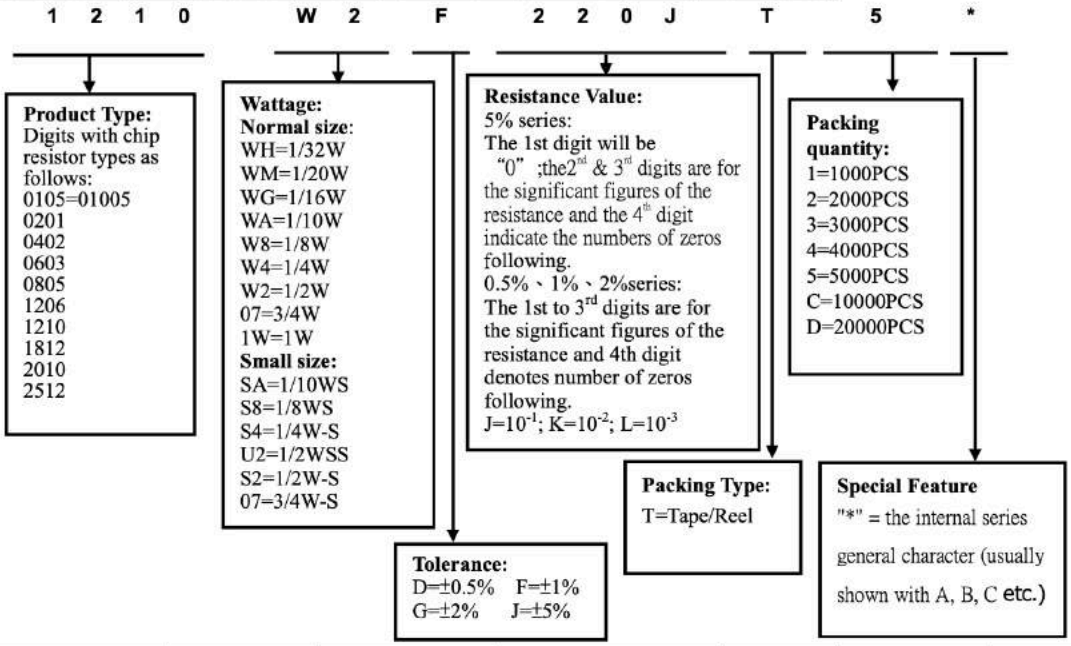
4=4000pcs 5=5000pcs C=10000pcs D=20000pcs E=15000pcs

Chip Product: BD=B/B-20000pcs TC=T/R-10000pcs

7.4.5 For some items, the 14th digit alone can use to denote special features of additional information with the following codes:

"*" = the internal series general character (usually shown with A, B, C etc.)

8.0 Ordering Procedure: (Example: 1210 1/2W ±1% 22Ω T/R-5000)

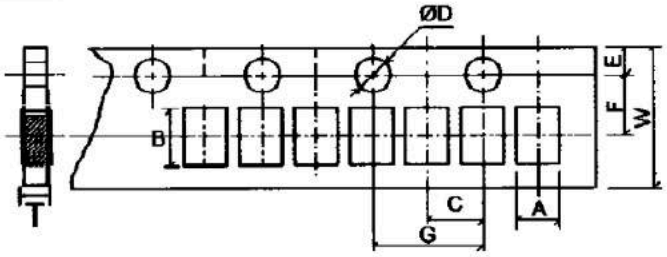


昆 山 厚 聲 電 子 工 業 有 限 公 司
UNIROYAL ELECTRONICS INDUSTRY (KUNSHAN) CO., LTD.

ISO14001 ISO/TS16949 244546 245468 REG.-Nr.A759 CQC04001010656

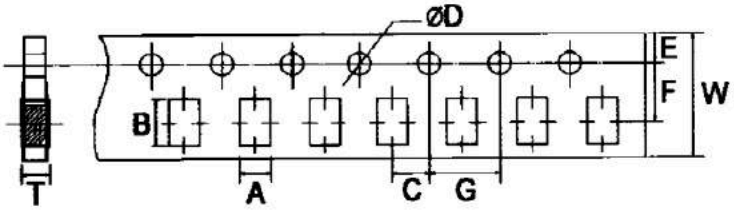
9.0 Packaging:

9.1 Tapping Dimension:



Unit: mm

Type	A	B	C±0.05	ΦD +0.1 -0	E±0.1	F±0.05	G±0.1	W±0.2	T±0.1
01005	0.24±0.05	0.45±0.05	2.00	1.50	1.75	3.50	4.00	8.00	0.40
0201	0.40±0.05	0.70±0.05	2.00	1.50	1.75	3.50	4.00	8.00	0.42
0402	0.65±0.20	1.15±0.20	2.00	1.50	1.75	3.50	4.00	8.00	0.45

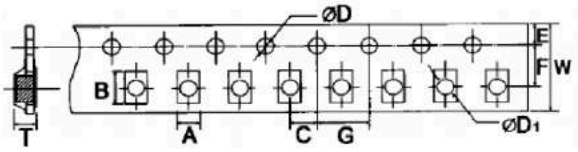


Unit: mm

Type	A ±0.2	B ±0.2	C±0.05	ΦD +0.1 -0	E±0.1	F±0.05	G±0.1	W±0.2	T±0.1
0603	1.10	1.90	2.00	1.50	1.75	3.50	4.00	8.00	0.67
0805	1.65	2.40	2.00	1.50	1.75	3.50	4.00	8.00	0.81
1206	2.00	3.60	2.00	1.50	1.75	3.50	4.00	8.00	0.81
1210	2.80	3.50	2.00	1.50	1.75	3.50	4.00	8.00	0.75
2010	2.80	5.40	2.00	1.50	1.75	5.50	4.00	12.00	0.75

昆 山 厚 聲 電 子 工 業 有 限 公 司
UNIROYAL ELECTRONICS INDUSTRY (KUNSHAN) CO., LTD.

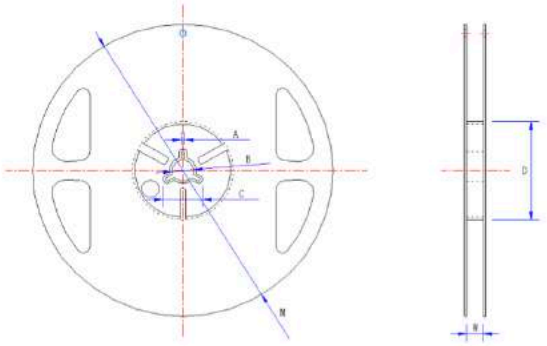
ISO14001 ISO/TS16949 244546 245468 REG.-Nr.A759 CQC04001010656



Unit: mm

Type	A±0.2	B±0.2	C±0.05	+0.1 ϕD -0	+0.25 ϕD1 -0	E±0.1	F±0.05	G±0.1	W±0.2	T±0.1
1812	3.50	4.80	2.00	1.50	1.50	1.75	5.50	4.00	12.00	1.00
2512	3.50	6.70	2.00	1.50	1.50	1.75	5.50	4.00	12.00	1.00

9.2 Dimension:



Unit: mm

Type	Taping	Qty/Reel	A±0.5	B±0.5	C±0.5	D±1	M±2	W±1
01005	Paper	20,000pcs	2.0	13.0	21.0	60.0	178.0	10.0
0201	Paper	10,000pcs	2.0	13.0	21.0	60.0	178.0	10.0
0402	Paper	10,000pcs	2.0	13.0	21.0	60.0	178.0	10.0
0603	Paper	5,000pcs	2.0	13.0	21.0	60.0	178.0	10.0
0805	Paper	5,000pcs	2.0	13.0	21.0	60.0	178.0	10.0
1206	Paper	5,000pcs	2.0	13.0	21.0	60.0	178.0	10.0
1210	Paper	5,000pcs	2.0	13.0	21.0	60.0	178.0	10.0
2010	Paper or Embossed	4,000pcs	2.0	13.0	21.0	60.0	178.0	13.8
1812	Embossed	4,000pcs	2.0	13.0	21.0	60.0	178.0	13.8
2512	Embossed	4,000pcs	2.0	13.0	21.0	60.0	178.0	13.8

10.0: Note Matter：

10.1 UNIOHM recommend the storage condition temperature: 15℃~35℃, humidity :25%~75%.
(Put condition for individual product).
Even under UNIOHM recommended storage condition, solderability of products over 1 year old.
(Put condition for each product) may be degraded.

10.2 Store / transport cartons in the correct direction, which is indicated on a carton as a symbol.
Otherwise bent leads may occur due to excessive stress applied when dropping of a carton.

10.3 Product performance and soldered connections may deteriorate if the products are stored in the following places:

a. Storage in high Electrostatic.

b. Storage in direct sunshine、rain and snow or condensation.

c. Where the products are exposed to sea winds or corrosive gases, including Cl_2, H_2S_3 NH_3, SO_2, NO_2.

10.4 The products are used in circuit board thickness greater than 1.6mm. If customers use less than the thickness of the circuit board that you should confirm with the company, in order to recommend a more suitable product.

Advanced Monolithic Systems

AMS1117

1A LOW DROPOUT VOLTAGE REGULATOR

RoHS compliant

FEATURES

- **Three Terminal Adjustable or Fixed Voltages* 1.5V, 1.8V, 2.5V, 2.85V, 3.3V and 5.0V**
- **Output Current of 1A**
- **Operates Down to 1V Dropout**
- **Line Regulation: 0.2% Max.**
- **Load Regulation: 0.4% Max.**
- **SOT-223, TO-252 and SO-8 package available**

APPLICATIONS

- **High Efficiency Linear Regulators**
- **Post Regulators for Switching Supplies**
- **5V to 3.3V Linear Regulator**
- **Battery Chargers**
- **Active SCSI Terminators**
- **Power Management for Notebook**
- **Battery Powered Instrumentation**

GENERAL DESCRIPTION

The AMS1117 series of adjustable and fixed voltage regulators are designed to provide 1A output current and to operate down to 1V input-to-output differential. The dropout voltage of the device is guaranteed maximum 1.3V at maximum output current, decreasing at lower load currents.

On-chip trimming adjusts the reference voltage to 1%. Current limit is also trimmed, minimizing the stress under overload conditions on both the regulator and power source circuitry.

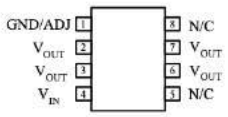
The AMS1117 devices are pin compatible with other three-terminal SCSI regulators and are offered in the low profile surface mount SOT-223 package, in the 8L SOIC package and in the TO-252 (DPAK) plastic package.

ORDERING INFORMATION:

PACKAGE TYPE			OPERATING JUNCTION TEMPERATURE RANGE
TO-252	SOT-223	8L SOIC	
AMS1117CD	AMS1117	AMS1117CS	-40 to 125° C
AMS1117CD-1.5	AMS1117-1.5	AMS1117CS-1.5	-40 to 125° C
AMS1117CD-1.8	AMS1117-1.8	AMS1117CS-1.8	-40 to 125° C
AMS1117CD-2.5	AMS1117-2.5	AMS1117CS-2.5	-40 to 125° C
AMS1117CD-2.85	AMS1117-2.85	AMS1117CS-2.85	-40 to 125° C
AMS1117CD-3.3	AMS1117-3.3	AMS1117CS-3.3	-40 to 125° C
AMS1117CD-5.0	AMS1117-5.0	AMS1117CS-5.0	-40 to 125° C

*For additional available fixed voltages contact factory.

8L SOIC Top View

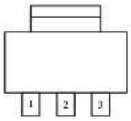


PIN CONNECTIONS

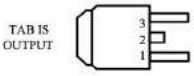
3 PIN FIXED/ADJUSTABLE VERSION

1- Ground/Adjust
2- V_{OUT}
3- V_{IN}

SOT-223 Top View



TO-252 FRONT VIEW



AMS1117

ABSOLUTE MAXIMUM RATINGS (Note 1)

Power Dissipation	Internally limited
Input Voltage	15V
Operating Junction Temperature	
Control Section	0°C to 125°C
Power Transistor	0°C to 150°C
Storage temperature	- 65°C to +150°C

Soldering information	
Lead Temperature (25 sec)	265°C
Thermal Resistance	
SO-8 package	φ_{JA}= 160°C/W
TO-252 package	φ_{JA}= 80°C/W
SOT-223 package	φ_{JA}= 90°C/W*

* With package soldering to copper area over backside ground plane or internal power plane φ_{JA} can vary from 46°C/W to >90°C/W depending on mounting technique and the size of the copper area.

ELECTRICAL CHARACTERISTICS

Electrical Characteristics at I_{OUT} = 0 mA, and T_J = +25°C unless otherwise specified.

Parameter	Device	Conditions	Min	Typ	Max	Units
Reference Voltage (Note 2)	AMS1117	I_{OUT} = 10 mA 10mA ≤ I_{OUT} ≤ 1A, 1.5V≤ (V_{IN} - V_{OUT}) ≤ 12V	1.238 **1.225**	1.250 **1.250**	1.262 **1.270**	V V
Output Voltage (Note 2)	AMS1117-1.5	0 ≤ I_{OUT} ≤ 1A , 3.0V≤ V_{IN} ≤ 12V	1.485 **1.476**	1.500 **1.500**	1.515 **1.524**	V V
	AMS1117-1.8	0 ≤ I_{OUT} ≤ 1A , 3.3V≤ V_{IN} ≤ 12V	1.782 **1.773**	1.800 **1.800**	1.818 **1.827**	V V
	AMS1117-2.5	0 ≤ I_{OUT} ≤ 1A , 4.0V≤ V_{IN} ≤ 12V	2.475 **2.460**	2.500 **2.500**	2.525 **2.560**	V V
	AMS1117-2.85	0 ≤ I_{OUT} ≤ 1A , 4.35V≤ V_{IN} ≤ 12V	2.82 **2.79**	2.850 **2.850**	2.88 **2.91**	V V
	AMS1117-3.3	0 ≤ I_{OUT} ≤ 1A , 4.75V ≤ V_{IN} ≤ 12V	3.267 **3.235**	3.300 **3.300**	3.333 **3.365**	V V
	AMS1117-5.0	0 ≤ I_{OUT} ≤ 1A , 6.5V ≤ V_{IN} ≤ 12V	4.950 **4.900**	5.000 **5.000**	5.050 **5.100**	V V
Line Regulation	AMS1117	I_{LOAD} = 10 mA , 1.5V≤ (V_{IN} - V_{OUT}) ≤ 12V		0.015 **0.035**	0.2 **0.2**	% %
	AMS1117-1.5	3.0V≤ V_{IN} ≤ 12V		0.3 **0.6**	5 **6**	mV mV
	AMS1117-1.8	3.3V≤ V_{IN} ≤ 12V		0.3 **0.6**	5 **6**	mV mV
	AMS1117-2.5	4.0V≤ V_{IN} ≤ 12V		0.3 **0.6**	6 **6**	mV mV
	AMS1117-2.85	4.35V≤ V_{IN} ≤ 12V		0.3 **0.6**	6 **6**	mV mV
	AMS1117-3.3	4.75V≤ V_{IN} ≤ 12V		0.5 **1.0**	10 **10**	mV mV
	AMS1117-5.0	6.5V≤ V_{IN} ≤ 12V		0.5 **1.0**	10 **10**	mV mV
Load Regulation (Notes 2, 3)	AMS1117	(V_{IN} - V_{OUT}) =3V, 10mA ≤ I_{OUT} ≤ 1A		0.1 **0.2**	0.3 **0.4**	% %
	AMS1117-1.5	V_{IN} = 5V, 0 ≤ I_{OUT} ≤ 1A		3 **6**	10 **20**	mV mV
	AMS1117-1.8	V_{IN} = 5V, 0 ≤ I_{OUT} ≤ 1A		3 **6**	10 **20**	mV mV
	AMS1117-2.5	V_{IN} = 5V, 0 ≤ I_{OUT} ≤ 1A		3 **6**	12 **20**	mV mV

ELECTRICAL CHARACTERISTICS

Electrical Characteristics at I_{OUT} = 0 mA, and T_J = +25°C unless otherwise specified.

Parameter	Device	Conditions	Min	Typ	Max	Units
Load Regulation (Notes 2, 3)	AMS1117-2.85	V_{IN} = 5V, 0 ≤ I_{OUT} ≤ 1A		3 **6**	12 **20**	mV mV
	AMS1117-3.3	V_{IN} = 5V, 0 ≤ I_{OUT} ≤ 1A		3 **7**	15 **25**	mV mV
	AMS1117-5.0	V_{IN} = 8V, 0 ≤ I_{OUT} ≤ 1A		5 **10**	20 **35**	mV mV
Dropout Voltage (V_{IN} - V_{OUT})	AMS1117-1.5/-1.8/-2.5/-2.85/-3.3/-5.0	ΔV_{OUT}, ΔV_{REF} = 1%, I_{OUT} = 1A (Note 4)		**1.1**	**1.3**	V
Current Limit	AMS1117-1.5/-1.8/-2.5/-2.85/-3.3/-5.0	(V_{IN} - V_{OUT}) = 5V	**900**	**1,100**	**1,500**	mA
Minimum Load Current	AMS1117	(V_{IN} - V_{OUT}) = 12V (Note 5)		**5**	**10**	mA
Quiescent Current	AMS1117-1.5/-1.8/-2.5/-2.85/-3.3/-5.0	V_{IN} ≤ 12V		**5**	**10**	mA
Ripple Rejection	AMS1117	f =120Hz , C_{OUT} = 22µF Tantalum, I_{OUT} = 1A, (V_{IN}-V_{OUT}) = 3V, C_{ADJ} =10µF	**60**	**75**		dB
	AMS1117-1.5/-1.8/-2.5/-2.85	f =120Hz , C_{OUT} = 22µF Tantalum, I_{OUT} = 1A, V_{IN} = 6V	**60**	**72**		dB
	AMS1117-3.3	f =120Hz , C_{OUT} = 22µF Tantalum, I_{OUT} = 1A V_{IN} = 6.3V	**60**	**72**		dB
	AMS1117-5.0	f =120Hz , C_{OUT} = 22µF Tantalum, I_{OUT} = 1A V_{IN} = 8V	**60**	**68**		dB
Thermal Regulation	AMS1117	T_A = 25°C, 30ms pulse		0.008	0.04	%W
Adjust Pin Current	AMS1117	10mA ≤ I_{OUT} ≤ 1A , 1.5V≤ (V_{IN} - V_{OUT}) ≤ 12V		55	 **120**	µA µA
Adjust Pin Current Change	AMS1117	10mA ≤ I_{OUT} ≤ 1A , 1.5V≤ (V_{IN} - V_{OUT}) ≤ 12V		**0.2**	**5**	µA
Temperature Stability				**0.5**		%
Long Term Stability		T_A =125°C, 1000Hrs		0.3	1	%
RMS Output Noise (% of V_{OUT})		T_A = 25°C , 10Hz ≤ f ≤ 10kHz		0.003		%
Thermal Resistance Junction-to-Case					15	°C/W

Parameters identified with **boldface type** apply over the full operating temperature range.

Note 1: Absolute Maximum Ratings indicate limits beyond which damage to the device may occur. For guaranteed specifications and test conditions, see the Electrical Characteristics. The guaranteed specifications apply only for the test conditions listed.

Note 2: Line and Load regulation are guaranteed up to the maximum power dissipation of 1.2 W. Power dissipation is determined by the input/output differential and the output current. Guaranteed maximum power dissipation will not be available over the full input/output range.

Note 3: See thermal regulation specifications for changes in output voltage due to heating effects. Line and load regulation are measured at a constant junction temperature by low duty cycle pulse testing. Load regulation is measured at the output lead ~1/8" from the package.

Note 4: Dropout voltage is specified over the full output current range of the device.

Note 5: Minimum load current is defined as the minimum output current required to maintain regulation. When 1.5V ≤ (V_{IN} - V_{OUT}) ≤ 12V the device is guaranteed to regulate if the output current is greater than 10mA.

APPLICATION HINTS

The AMS1117 series of adjustable and fixed regulators are easy to use and are protected against short circuit and thermal overloads. Thermal protection circuitry will shut-down the regulator should the junction temperature exceed 165°C at the sense point.
Pin compatible with older three terminal adjustable regulators, these devices offer the advantage of a lower dropout voltage, more precise reference tolerance and improved reference stability with temperature.

Stability

The circuit design used in the AMS1117 series requires the use of an output capacitor as part of the device frequency compensation. The addition of 22µF solid tantalum on the output will ensure stability for all operating conditions.
When the adjustment terminal is bypassed with a capacitor to improve the ripple rejection, the requirement for an output capacitor increases. The value of 22µF tantalum covers all cases of bypassing the adjustment terminal. Without bypassing the adjustment terminal smaller capacitors can be used with equally good results.
To further improve stability and transient response of these devices larger values of output capacitor can be used.

Protection Diodes

Unlike older regulators, the AMS1117 family does not need any protection diodes between the adjustment pin and the output and from the output to the input to prevent over-stressing the die. Internal resistors are limiting the internal current paths on the AMS1117 adjustment pin, therefore even with capacitors on the adjustment pin no protection diode is needed to ensure device safety under short-circuit conditions.
Diodes between the input and output are not usually needed. Microsecond surge currents of 50A to 100A can be handled by the internal diode between the input and output pins of the device. In normal operations it is difficult to get those values of surge currents even with the use of large output capacitances. If high value output capacitors are used, such as 1000µF to 5000µF and the input pin is instantaneously shorted to ground, damage can occur. A diode from output to input is recommended, when a crowbar circuit at the input of the AMS1117 is used (Figure 1).

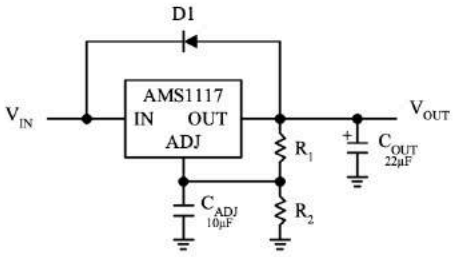
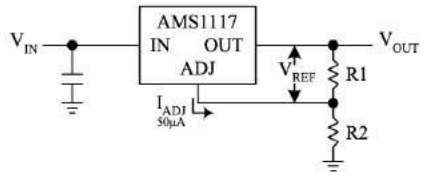


Figure 1.

Output Voltage

The AMS1117 series develops a 1.25V reference voltage between the output and the adjust terminal. Placing a resistor between these two terminals causes a constant current to flow through R1 and down through R2 to set the overall output voltage. This current is normally the specified minimum load current of 10mA. Because I_{ADJ} is very small and constant it represents a small error and it can usually be ignored.



$$V_{OUT} = V_{REF}(1 + R2/R1) + I_{ADJ}R2$$

Figure 2. Basic Adjustable Regulator

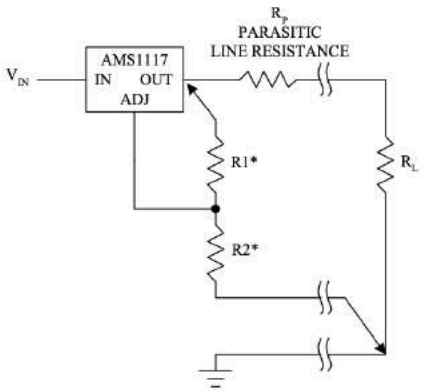
Load Regulation

True remote load sensing it is not possible to provide, because the AMS1117 is a three terminal device. The resistance of the wire connecting the regulator to the load will limit the load regulation. The data sheet specification for load regulation is measured at the bottom of the package. Negative side sensing is a true Kelvin connection, with the bottom of the output divider returned to the negative side of the load.
The best load regulation is obtained when the top of the resistor divider R1 is connected directly to the case not to the load. If R1 were connected to the load, the effective resistance between the regulator and the load would be:

$$R_P \times \left(\frac{R2+R1}{R1}\right), \quad R_P = \text{Parasitic Line Resistance}$$

APPLICATION HINTS

Connected as shown , R_P is not multiplied by the divider ratio



*CONNECT R1 TO CASE
CONNECT R2 TO LOAD

Figure 3. Connections for Best Load Regulation

In the case of fixed voltage devices the top of R1 is connected Kelvin internally, and the ground pin can be used for negative side sensing.

Thermal Considerations

The AMS1117 series have internal power and thermal limiting circuitry designed to protect the device under overload conditions. However maximum junction temperature ratings of 125°C should not be exceeded under continuous normal load conditions.
Careful consideration must be given to all sources of thermal resistance from junction to ambient. For the surface mount package SOT-223 additional heat sources mounted near the device must be considered. The heat dissipation capability of the PC board and its copper traces is used as a heat sink for the device. The thermal resistance from the junction to the tab for the AMS1117 is 15°C/W. Thermal resistance from tab to ambient can be as low as 30°C/W.

The total thermal resistance from junction to ambient can be as low as 45°C/W. This requires a reasonable sized PC board with at least on layer of copper to spread the heat across the board and couple it into the surrounding air.
Experiments have shown that the heat spreading copper layer does not need to be electrically connected to the tab of the device. The PC material can be very effective at transmitting heat between the pad area, attached to the pad of the device, and a ground plane layer either inside or on the opposite side of the board. Although the actual thermal resistance of the PC material is high, the Length/Area ratio of the thermal resistance between layers is small. The data in Table 1, was taken using 1/16" FR-4 board with 1 oz. copper foil, and it can be used as a rough guideline for estimating thermal resistance.
For each application the thermal resistance will be affected by thermal interactions with other components on the board. To determine the actual value some experimentation will be necessary.
The power dissipation of the AMS1117 is equal to:

$P_D = (V_{IN} - V_{OUT})(I_{OUT})$

Maximum junction temperature will be equal to:

$T_J = T_{A(MAX)} + P_D$(Thermal Resistance (junction-to-ambient))

Maximum junction temperature must not exceed 125°C.

Ripple Rejection

The ripple rejection values are measured with the adjustment pin bypassed. The impedance of the adjust pin capacitor at the ripple frequency should be less than the value of R1 (normally 100Ω to 200Ω) for a proper bypassing and ripple rejection approaching the values shown. The size of the required adjust pin capacitor is a function of the input ripple frequency. If R1=100Ω at 120Hz the adjust pin capacitor should be >13µF. At 10kHz only 0.16µF is needed.

The ripple rejection will be a function of output voltage, in circuits without an adjust pin bypass capacitor. The output ripple will increase directly as a ratio of the output voltage to the reference voltage (V_{OUT} / V_{REF}).

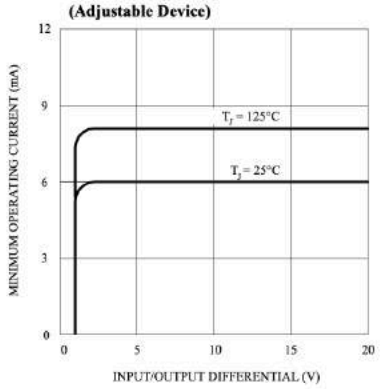
Table 1.

COPPER AREA			THERMAL RESISTANCE
TOP SIDE*	BACK SIDE	BOARD AREA	(JUNCTION-TO-AMBIENT)
2500 Sq. mm	2500 Sq. mm	2500 Sq. mm	45°C/W
1000 Sq. mm	2500 Sq. mm	2500 Sq. mm	45°C/W
225 Sq. mm	2500 Sq. mm	2500 Sq. mm	53°C/W
100 Sq. mm	2500 Sq. mm	2500 Sq. mm	59°C/W
1000 Sq. mm	1000 Sq. mm	1000 Sq. mm	52°C/W
1000 Sq. mm	0	1000 Sq. mm	55°C/W

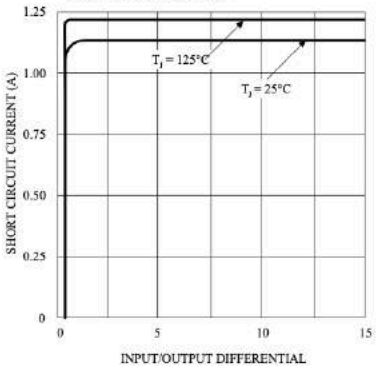
* Tab of device attached to topside copper.

TYPICAL PERFORMANCE CHARACTERISTICS

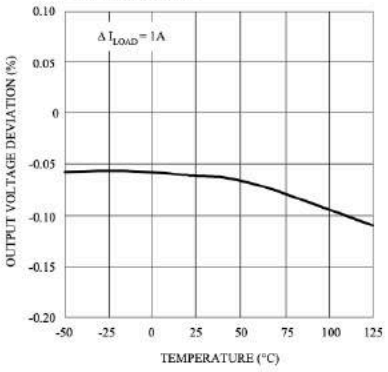
Minimum Operating Current (Adjustable Device)



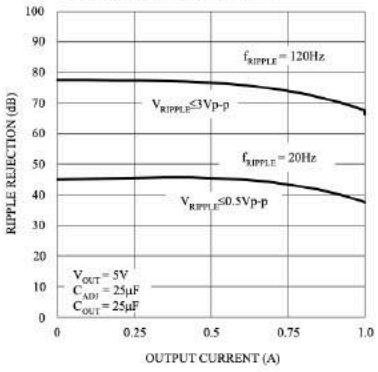
Short-Circuit Current



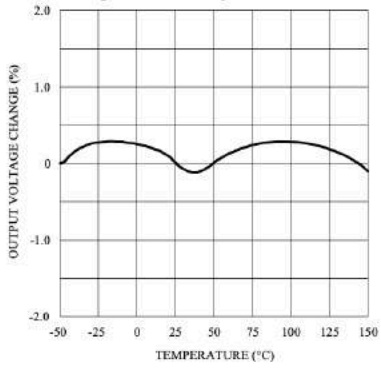
Load Regulation



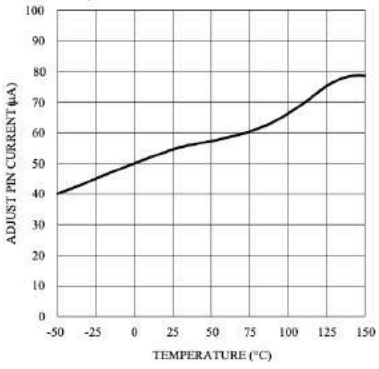
Ripple Rejection vs. Current



Temperature Stability

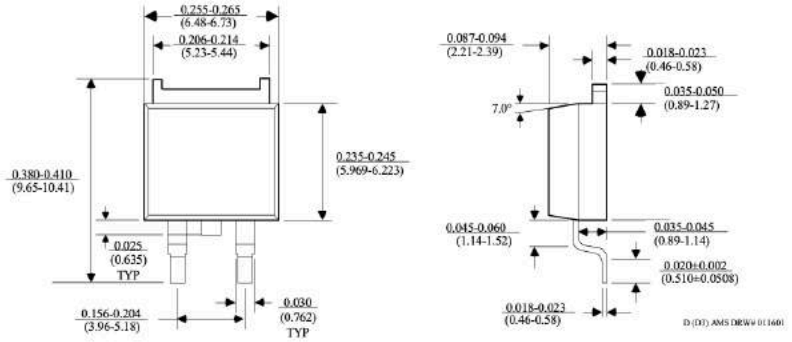


Adjust Pin Current

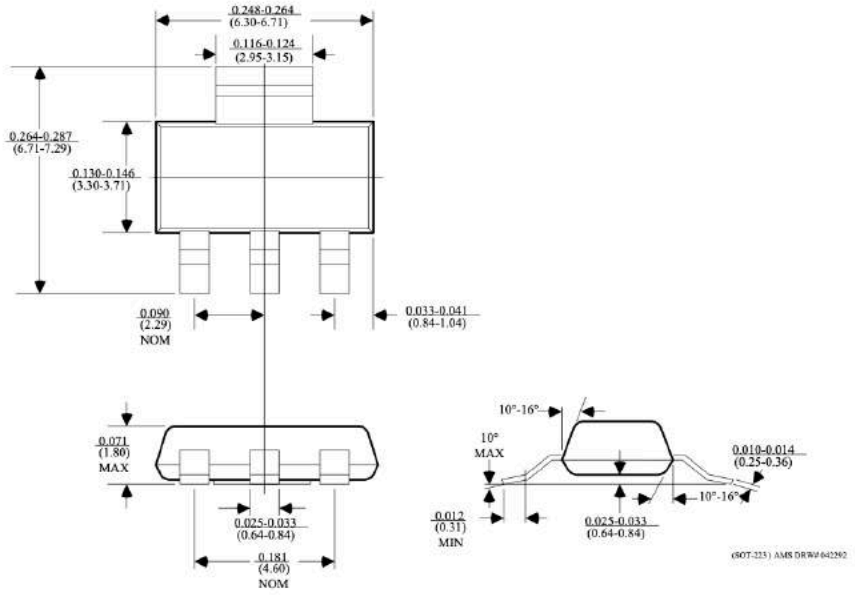


PACKAGE DIMENSIONS inches (millimeters) unless otherwise noted.

TO-252 PLASTIC PACKAGE (D)

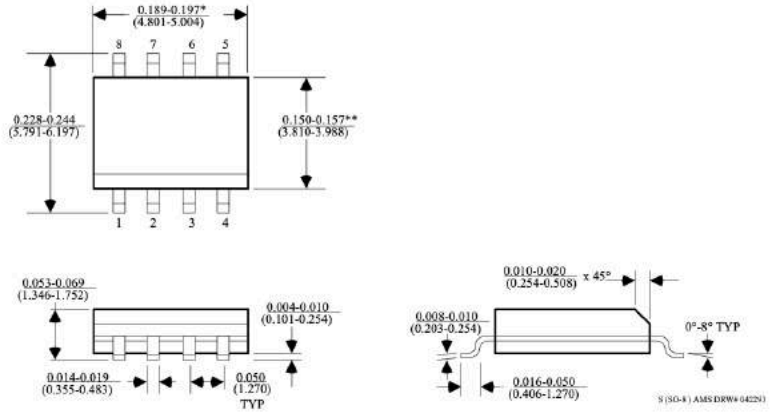


3 LEAD SOT-223 PLASTIC PACKAGE



PACKAGE DIMENSIONS inches (millimeters) unless otherwise noted (Continued).

8 LEAD SOIC PLASTIC PACKAGE (S)



*DIMENSION DOES NOT INCLUDE MOLD FLASH. MOLD FLASH SHALL NOT EXCEED 0.006" (0.152mm) PER SIDE

**DIMENSION DOES NOT INCLUDE INTERLEAD FLASH. INTERLEAD FLASH SHALL NOT EXCEED 0.010" (0.254mm) PER SIDE

6.-Bibliografía

1. Neo.lcc.uma.es [internet]. *Herramientas web para la enseñanza de protocolos de comunicación*. Protocolo TCP/UDP. Disponible en: http://neo.lcc.uma.es/evirtual/cdd/tutorial/Indice.html

2. Northwestern University. McCormick School of Engineering. *Serial Port and Communications*. Disponible en: http://www.ece.northwestern.edu/local-apps/matlabhelp/techdoc/matlab_external/ch_seri8.html

3. Ettrich M, Blanchette J, Summerfield M. *C++ GUI programming with Qt4*. Second Edition. Westford, Massachusetts: Prentice-Hall; 2014.

4. Villó, I. *Sistemas Electrónicos para Astronomía*. Cartagena, Murcia.

5. López, R. *Diseño, construcción y desarrollo de un sistema limitado por difracción para telescopios terrestres: FastCam*. Pontevedra: Universidad de Vigo [Tesis Doctoral]; 2012.

6. Oscoz A, López R, Garrido A. *Commisioning and first observations with Wide FastCam at the Telescopio Carlos Sánchez*. Disponible en: https://personas.upct.es/publicacion/85007179064

7. Law et al, *Lucky Imaging: High Angular Resolution Imaging in the Visible from the Ground*, A&A 446, pp. 739-745.