



# Trabajo de Fin de Grado

Grado en Ingeniería Informática

# Visión en Sistemas Empotrados

*Vision in Embedded Systems*

Alejandro Delgado Martel

La Laguna, 1 de julio de 2017

D. **Jonay Tomás Toledo Carrillo**, con N.I.F. 78.698.554-Y profesor contratado Doctor adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor

Dña. **Sara González Pérez**, con N.I.F. 78.705.691-J profesora CLI adscrita al Departamento de Ingeniería Industrial de la Universidad de La Laguna, como cotutora

**C E R T I F I C A (N)**

Que la presente memoria titulada:

*“Visión en sistemas empotrados”*

ha sido realizada bajo su dirección por D. **Alejandro Delgado Martel**,

con N.I.F. 45.941.757-Q.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 1 de julio de 2017

# Agradecimientos

Unas veces tenemos que comenzar nuevas etapas y otras veces finalizarlas. En este caso me toca cerrar una de la que estoy particularmente agradecido, porque me ha hecho conocer personas maravillosas y conocerme a mi mismo.

Quería darle las gracias a mis compañeros, de los que me llevo ya muchos como amigos, personas excepcionales que me han ayudado a superarme día tras día y que han echo posible que llegue este día.

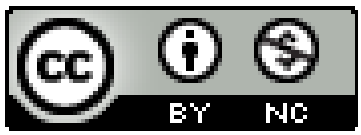
Agradecer también a los profesores de la carrera de los que he aprendido muchísimo y de seguro sin ellos este trabajo no sería posible.

Agradecer especialmente a dos de ellos, en concreto a Jonay Tomás Toledo Carrillo y a Sara González Pérez por haberme dado la oportunidad de haber realizado este proyecto y por todo el apoyo brindado.

Por último, a mi familia, aquellos que empezaron conmigo esta lucha, por los que continúan en ella y por los que ya no están, gracias por todos estos años apoyándome y alentándome a seguir adelante, sin ustedes nada de esto habría sido posible.

A todos, GRACIAS DE CORAZÓN.

# Licencia



## Resumen

*El objetivo de este trabajo es desarrollar un sistema que usa imágenes del cielo y pequeños algoritmos de visión por computador para estimar y predecir la irradiación solar en un área de estudio dada según el estado del entorno.*

*Para alcanzar esta tarea se han utilizado varios sistemas empotrados como los conocidos Raspberry Pi y Arduino. Además de un ordenador encargado de procesar las imágenes.*

*La predicción se realizará gracias a imágenes capturadas con una cámara de lente ojo de pez unida a la Raspberry Pi, extrayéndose datos meteorológicos a partir de cálculos obtenidos mediante algoritmos de visión por computador. Todo el trabajo se ha desarrollado en sistemas Linux tanto el ordenador encargado de procesar las imágenes como la Raspberry Pi. El desarrollo se ha basado en el procesado de imágenes gracias a la librería de software libre OpenCV que provee de los principales algoritmos y procedimientos necesarios para la manipulación de las imágenes.*

*Se ha diseñado un sistema fácilmente utilizable, capaz de procesar imágenes del cielo en tiempo real y que realiza una predicción meteorología sobre un corto periodo de tiempo.*

*El sistema es capaz de predecir la irradiación sobre la placa PV con 3-5 min de antelación*



**Palabras clave:** Meteorología, Raspberry Pi, Arduino, OpenCV, Linux, Predicción, Cámara.

## Abstract

*The objective of this work is to develop a system that uses Sky images and small computer vision algorithms to forecast solar irradiation in a given area of study according to the state of the enviroment.*

*To achive this task it have been used several embedded system such as the known Raspberry Pi and Arduino. Also a computer has been used to process the images.*

*The prediction is posible thanks to images captured with a fisheye lens camera attached to the Raspberry Pi and meteorological data will be obtained using computer vision algotithms for prediction purposes. This project has been developed on Linux systems, both the computer in charge of processing the images as the Raspberry Pi. The development has been based on the processing of images thanks to the free software library called OpenCV that provides the main algorithms and procedures necessary for the manupulation of the images.*

*An easily and usable system have been designed, capable of processing Sky images in real time to give a weather prediction over a short period of time.*

*The system is capable of predict the irradiation over the photoelectric plate with 3-5 minutes in advance.*



**Keywords:** Meteorology, Raspberry Pi, Arduino, OpenCV, Linux, Prediction, Camera

# Índice general

**Capítulo 1 Introducción.....1**
1.1 Sistemas empotrados.....1
1.2 Visión por Computador.....1
1.3 Predicción meteorológica.....2
**Capítulo 2 Antecedentes y estado actual.....3**
2.1 Pares-FV y Sunoracle.....3
**Capítulo 3 Objetivos del trabajo.....4**
3.1 Sistema para la previsión de imágenes meteorológicas.....4
**Capítulo 4 Recursos utilizados.....6**
4.1 Recursos.....6
4.1.1 Raspberry Pi 3 model B.....6
4.1.2 Raspberry Pi Camera (Fisheye lens).....7
4.1.3 Arduino UNO.....8
4.1.4 Power over Ethernet.....9
4.1.5 Sensor DHT22.....10
4.1.6 Célula fotoeléctrica.....10
**Capítulo 5 Desarrollo.....11**
5.1 Introducción.....11
5.2 Instalación y configuración de la Raspberry Pi 3 model B.....11
5.2.1 Instalación y configuración Raspbian Jessie.....12
5.2.2 Descarga y compilación de la librería OpenCV.....12
5.2.3 Descarga y compilación de la librería Raspicam.....14
5.2.4 Instalación del gestor de base de datos SQLite.....15
5.2.5 Instalación librerías boost.....15

5.2.6 Configuración del cliente NFS....16
5.2.7 Instalación Arduino....16
5.3 Configuración equipo de procesamiento....17
5.3.1 Configuración OpenCV, SQLite y boost....17
5.3.2 Configuración Servidor NFS....17
5.4 Captura y almacenamiento de imágenes....18
5.4.1 Captura secuencia imágenes con diferente ISO....18
5.4.2 Generación de imágenes fusionadas....19
5.5 Recogida de datos....20
5.5.1 Programa Arduino para la detección de parámetros....20
5.5.2 Creación de la base de datos....21
5.5.3 Paso de los datos....22
5.6 Procesamiento de imágenes....22
5.6.1 Detección del Sol....22
1 Detección mediante contornos....23
2 Detección por brillo....24
5.6.2 Cálculo vectores de movimiento....25
5.6.3 Algoritmo de predicción fotovoltaica....28
5.7 Evaluación de resultados....32
**Capítulo 6 Problemática y Limitaciones del Sistema....38**
6.1 Capacidad de computo y temperatura de la Raspberry Pi 3. 38
6.2 Captura en la secuencia de imágenes. Librería Raspicam....38
6.3 Calidad de la cámara....39
6.4 Tiempo atmosférico....39
**Capítulo 7 Conclusiones y líneas futuras....40**
**Capítulo 8 Summary and Conclusions....41**
**Capítulo 9 Presupuesto....42**
9.1 Presupuesto de materiales....42
9.2 Presupuesto de trabajo realizado....43
9.3 Presupuesto total del proyecto....43
**Apéndice A. Código desarrollado....44**
9.4 Algoritmo Arduino....44
9.5 Código del proyecto....46

**Apéndice B.....47**
9.6 Raspberry Pi 3 model B. Características.....47
9.7 Arduino Uno.....48
9.8 DHT22 Sensor.....49
**Bibliografía.....50**

# Índice de figuras

Figura 1: Sistema de captura.....5
Figura 2: Raspberry Pi 3 model B.....7
Figura 3: Pi Camera OV5647 FishEye.....8
Figura 4: Arduino UNO rev3.....9
Figura 5: PoE Emisor.....9
Figura 6: PoE Receptor.....10
Figura 7: Sensor DHT22.....10
Figura 8: CMake gráfico para la compilación de OpenCV.....13
Figura 9: Gestor de base de datos SQLite.....16
Figura 10: Esquemático Arduino y Sensores.....17
Figura 11: Fichero /etc/exports.....18
Figura 12: Generación de imágenes. Diagrama flujo.....19
Figura 13: Ejemplo de los datos recogidos.....21
Figura 14: Detección por contornos.....23
Figura 15: Detección por color (brillo).....24
Figura 16: Puntos de interés de la imagen.....26
Figura 17: Múltiples vectores de movimiento.....27
Figura 18: Vectores de movimiento por cada zona.....28
Figura 19: Distancia entre puntos.....29
Figura 20: Eje de coordenadas cartesianas (2D).....29
Figura 21: Ángulo del vector (Grados).....29
Figura 22: Distancia entre el vector y la zona Solar.....31
Figura 23: Variación en la insolación.....33

Figura 24: Área de captura de la cámara.....34
Figura 25: Secuencia 1.....35
Figura 26: Secuencia 2.....36
Figura 27: Datos extraídos de la predicción.....37

# Índice de tablas

Tabla 9.1: Resumen de materiales.....42
Tabla 9.2: Resumen de recursos.....43
Tabla 9.3: Presupuesto total del proyecto.....43

# Capítulo 1
# Introducción

En este capitulo de introducción se tratarán los conceptos más importantes en los que se basará todo el trabajo, como son: los sistemas empotrados, la visión por ordenador o la predicción meteorológica.

## 1.1 Sistemas empotrados

Los sistemas empotrados son sistemas informáticos diseñados para resolver una o un conjunto de tareas concreto. Es por ello que podemos decir que un sistema empotrado es un sistema de propósito específico y como todo sistema informático, tiene que tener un procesador (o microcontrolador) encargado de gestionar las tareas a realizar.

En la actualidad, los sistemas empotrados se encuentran en nuestro entorno dentro de aparatos tan cotidianos como relojes, móviles, mandos de televisor, detectores de humo…

Cada vez son más utilizados debido a su bajo coste, su facilidad de producción y por su reducido tamaño. Haciendo de los sistemas empotrados una gran opción a tener en cuenta para la realización de tareas de procesamiento en tiempo real.

Hoy en día existen numerosas organizaciones encargadas del diseño de sistemas empotrados de las que destacaremos Arduino, mbed, Upboard, BeagleBone o la propia Raspberry Pi.

## 1.2 Visión por Computador

La visión artificial es uno de los campos que mayor crecimiento ha experimentado en los últimos años y que actualmente está en auge sobre todo en lo relacionado a la realidad virtual. Sin embargo existen innumerables aplicaciones que se basan en la visión por computador como por ejemplo el reconocimiento de huellas

dactilares para un escáner biométrico, la detección de rostros, el reconocimiento de objetos en la conducción o en un entorno como trataremos en este trabajo.

Esta ciencia trata de procesar imágenes obteniendo información que es capaz de ser analizada e interpretada por un sistema informático.

En la actualidad todavía existe una gran diferencia entre la visión humana y la artificial sin embargo ésta última se encuentra en pleno avance y en un futuro no muy alejado me atrevo a decir que podrán realizarse tareas que actualmente parecen impensables.

## 1.3 Predicción meteorológica

Otro de los temas importantes a tratar en este trabajo, es la predicción meteorológica, en este caso, la meteorología es la ciencia que estudia es estado del tiempo, el tiempo atmosférico, los fenómenos que se producen y sus leyes.

En el caso de este proyecto, principalmente nos centraremos en el movimiento y comportamiento de las nubes ya que serán las encargadas de ocultar o no el Sol y por consiguiente limitar la intensidad de los rayos solares y así disminuir la cantidad de energía solar recogida.

Para la medición utilizaremos sensores gestionados por sistemas empotrados que harán el papel de higrómetro y termómetro, además de captar los rayos solares con una célula fotoeléctrica.

# Capítulo 2
# Antecedentes y estado actual

En este apartado trataremos dos de los proyectos que más se asemeja a este tipo de sistemas y que se han desarrollado en nuestro país.

## 2.1 Pares-FV y Sunoracle

Hace ya algunos años, se planteó el desarrollo de sistemas que pretenden mejorar la eficiencia de las plantas solares (fotovoltaicas o termosolares) y se crearon dos proyectos denominados Pares-FV y Sunoracle.

El primero de ellos enfocado a la energía solar fotovoltaica ha desarrollado un sistema que permite predecir la producción de energía registrando datos de las instalaciones fotovoltaicas como la intensidad, tensión, potencia y temperatura de los módulos. Con ello se intenta detectar averías  incluso antes de que sucedan para optimizar el rendimiento y la rentabilidad de las instalaciones fotovoltaicas.

Por otro lado, Sunoracle está enfocado en instalaciones termosolares y se digitaliza el paso de las nubes obteniendo información del movimiento, forma o altura. Todo ello con el objetivo de minimizar gastos de mantenimiento y optimizando así el rendimiento de las centrales.

Según expertos, alrededor de un 20% de los beneficios de las centrales se destinan al mantenimiento y averías. De ese 20%, la mitad se destina a averías (12%), sin embargo si se implantaran este tipo de sistemas se cree que podrían reducirse estas pérdidas a un 5%.

Los proyectos mencionados anteriormente, se desarrollaron gracias al grupo andaluz de ingeniería y medioambiente (MAGTEL) respaldados por la Corporación Tecnológica de Andalucía (CTA).

# Capítulo 3
# Objetivos del trabajo

En este capitulo, se describirán los objetivos que se han propuesto para la realización del proyecto.

## 3.1 Sistema para la previsión de imágenes meteorológicas

El principal objetivo del proyecto es crear un sistema capaz de llevar a cabo una previsión o estimación del comportamiento de las nubes para realizar predicciones meteorológicas. Este tipo de aplicación es fundamental para optimizar el rendimiento y la integración de la energía generada por las  por las plantas fotovoltaicas, y así ser capaces de estimar cuanta energía serían capaces de generar o si  deberían minimizar el gasto de energía para el mantenimiento de la planta.

El sistema hará uso de algoritmos de visión por computador y de sistemas empotrados ya que pueden ser una gran alternativa en base a costos y flexibilidad. Se utilizará una Raspberry Pi 3 model B y de una cámara integrada equipada con una lente de ojo de pez que fotografiará el cielo.

Para el cumplimiento del objetivo principal serán necesario establecer otra serie de metas:

1. **Captura de imágenes:** Para poder realizar la predicción obviamente deberemos capturar imágenes del cielo para el posterior estudio del movimiento relativo entre las nubes y el Sol. Este cometido lo realizaremos con la Raspberry utilizando la cámara ojo de pez.
2. **Detección de parámetros del entorno:** Otro de los objetivos que deberemos cumplir será el de capturar parámetros como la temperatura, humedad o insolación mediante el uso de sensores gestionados por otro sistema

empotrado (Arduino Uno).

3. **Algoritmo de predicción:** Por último, una vez hayamos capturado imágenes y tomado los datos proporcionados por los sensores, debemos ser capaces de procesarlos y generar una salida. Esta salida constará de una estimación que del movimiento de las nubes respecto a la posición del Sol detectando cuando lo va a ocultar y por tanto la variación del índice de insolación para un determinado instante de tiempo posterior.

Figura 1: Sistema de captura

# Capítulo 4
# Recursos utilizados

En este capítulo se listarán y detallarán los principales recursos que hemos utilizado para el desarrollo del proyecto.

## 4.1 Recursos

### 4.1.1 Raspberry Pi 3 model B

Pilar fundamental de este trabajo, la Raspberry Pi 3 model B es un ordenador de bolsillo de tercera generación fabricado por la Fundación Raspberry Pi y que salió al mercado en febrero de 2016.

La Raspberry Pi 3 cuenta con las siguientes características técnicas:

- CPU
  - Procesador ARM Cortex-A53 (Arquitectura ARMv8) a 1,2 GHz
  - 64 bit quad-core
- Memoria
  - 512 KB Caché L2 Compartida
  - 1 GB RAM DDR2
- Juego de instrucciones RISC de 32 bits
- GPU
  - Broadcom VideoCore IV
  - OpenGL ES 2.0, MPEG-2, VC-1
  - 1080p30 H.264/MPEG-4 AVC
- Almacenamiento
  - MicroSD
- Video: Entrada CSI

- Salida Vídeo
  - HDMI
  - RCA
  - DSI
- Puertos USB 2.0 Tipo A (4)
- Conectividad de red
  - 10/100 Ethernet
  - Wifi 802.11n
  - Bluetooth 4.1
- Puertos GPIO (General Purpose Input/Output) (17)

Figura 2: Raspberry Pi 3 model B

### 4.1.2 Raspberry Pi Camera (Fisheye lens)

Otro de los componentes necesarios para el buen desarrollo del proyecto es la cámara.

En este caso hemos utilizado una cámara diseñada precisamente para la Raspberry, se trata de la OV5647. La cámara soporta las últimas versiones de Raspberry Pi y Raspbian.

Algunas de las características de esta cámara son las siguientes:

- Resolución video HD 1080p
- Frecuencia máxima fotogramas 30
- Sensor Omnivision 5647

- 5MPixel sensor
- Filtro IR (Infrarrojos)
- Tamaño: 36 x 36 mm
- Interfaz 15 pin MIPI CSI (Camera Serial Interface)
- Ls-40180 FishEye Lens

Figura 3: Pi Camera
OV5647 FishEye

### 4.1.3 Arduino UNO

Arduino UNO es la placa que se basa en el microcontrolador ATmega328 y que se utilizará para recoger datos del entorno debido a su fácil uso y la eficacia de los datos obtenidos.

Arduino UNO cuenta con 14 pines de entrada/salida de los cuales 6 pueden ser salidas PWM (Modulación de ancho de pulsos) y otras 6 como entradas de tipo analógico.

Las características más importantes de esta placa son las siguientes:

- Microcontrolador ATmega328
- Puertos I/O Digital 14 (6 PWM)
- Puertos Entrada Analógica 6
- Memoria flash 32KB
- SRAM 2KB
- EEPROM 1KB
- Velocidad del reloj: 16 MHz

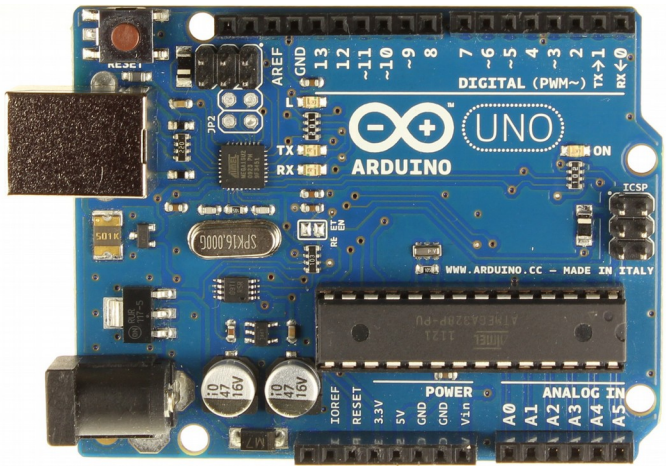


Figura 4: Arduino UNO rev3

### 4.1.4 Power over Ethernet

Al tratarse de un sistema de exteriores, será más que recomendable la utilización de un sistema Power over Ethernet para alimentar tanto la Raspberry como el Arduino, es por ello que hemos decidido utilizar esta tecnología, que permite alimentar equipos y dispositivos con corriente que circularía a través de los cables de Ethernet, el Powe over Ethernet (PoE) se regula con la norma IEEE 802.3af.

Estos sistemas de alimentación son fáciles de utilizar, y se componen en nuestro caso de un emisor y un receptor. El emisor se conectará a la corriente y tiene dos puertos RJ45 uno como entrada que irá conectado a nuestro router o switch de la red local y otro puerto de salida que a su vez será la entrada del receptor, por donde circularán datos y corriente eléctrica. En el receptor, se cuenta también con dos puertos RJ45 la entrada corresponderá con la salida del emisor y la salida se conectará al puerto Ethernet de la Raspberry, además el receptor se encargará de separar los datos de la corriente y esta última saldrá por un cable (alimentación y tierra) que se conectará también a la Raspberry con un conector del tipo micro-usb.

Para el proyecto se utiliza un PoE PHIHONG POE31U-1AT de 30W

Figura 5: PoE Emisor

El modelo del receptor es el POE14-050

Figura 6: PoE Receptor

### 4.1.5 Sensor DHT22

El DHT22 es un sensor de temperatura y humedad mucho más preciso y fiable que su hermano menor el DHT11 que podía llegar a tener un gran margen de error.

Las principales características son las siguientes:

- Alimentación: 3,3 o 5 voltios (5 recomendado)
- Tiempo de muestreo: 2 segundos
- Rango temperatura: -40ºC – 80ºC
- Precisión temperatura: ±0,5ºC
- Rango humedad: 0 – 99,9%
- Precisión humedad: ±2%

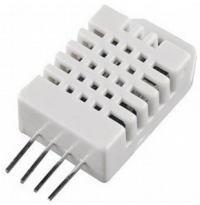


Figura 7: Sensor DHT22

### 4.1.6 Célula fotoeléctrica

La medición de la irradiación solar se realiza gracias a una célula fotovoltaica que nos arroja un voltaje determinado según la cantidad de luz solar incidente.

La célula va conectada a unas resistencias de 45 ohmios y de 1W de potencia tal que el voltaje de salida puede variar entre 0 y 4 voltios. El valor de 0 voltios lo obtenemos si la irradiación solar es baja y el valor de 4 voltios o un valor muy próximo a éste lo obtenemos cuando los rayos solares inciden directamente y con fuerza en el panel.

# Capítulo 5
# Desarrollo

En este capitulo se detallarán las diferentes tareas que hemos realizado durante el periodo de desarrollo. Estas tareas consisten en la configuración inicial de los sistemas que hemos utilizado, las pruebas realizadas, el desarrollo del código y los resultados obtenidos.

## 5.1 Introducción

El desarrollo de nuestro proyecto se ha basado principalmente en tres tareas fundamentales. Una primera tarea de configuración de nuestro sistema de bolsillo (Raspberry Pi 3 model B). En segundo lugar la investigación y estudio de las librerías utilizadas. Seguido por el tratamiento de imágenes. En una segunda fase se realizó una segunda tarea de instalación de la cámara y célula fotovoltaica con la consiguiente captura de imágenes para tener como referencia y por último las pruebas en tiempo real. Todo ello teniendo en cuenta que trabajamos con un sistema de bajo rendimiento y que por norma general, el procesado de imágenes, conlleva un alto coste computacional.

## 5.2 Instalación y configuración de la Raspberry Pi 3 model B

Para poder llevar a cabo nuestro proyecto, se ha decidido utilizar un sistema operativo basado en Linux como es Raspbian. A continuación se detallarán los pasos llevados a cabo para la puesta a punto de la Raspberry, la instalación y compilación de las librerías necesarias para el procesamiento de imágenes, así como el uso de gestores de bases de datos y librerías como ***boost*** que nos facilitará la tarea de comunicación serial y la puesta a punto de Arduino para la recopilación de información del entorno.

### 5.2.1 Instalación y configuración Raspbian Jessie

Si asumimos que nuestro sistema (Raspberry Pi 3) viene de fábrica sin ningún tipo de sistema operativo, deberemos descargarlo e instalarlo en la Raspberry.

En nuestro caso, utilizaremos como se ha mencionado anteriormente Raspbian Jessie, que lo descargaremos como imagen y con ayuda de un sistema Linux como por ejemplo Ubuntu, en Utilidades de disco restauramos la imagen descargada desde la web oficial de Raspberry y simplemente con eso tendremos nuestro sistema listo para el uso.

Por defecto, el sistema utiliza como usuario "**Pi**" y **sin contraseña**. En un primer momento deberíamos ejecutar los comandos encargados de actualizar los paquetes a nuevas versiones (si hiciera falta). Los comandos son los siguiente:

```
$ sudo apt-get update
$ sudo apt-get upgrade
```

Enlace de descarga Raspbian Jessie:
https://www.raspberrypi.org/downloads/raspbian/

### 5.2.2 Descarga y compilación de la librería OpenCV

Para la realización del proyecto, es necesario como se ha comentado la utilización de la librería llamada OpenCV, que nos facilitará en gran medida la manipulación de las imágenes, en nuestro caso utilizamos la última versión hasta la fecha, la 3.2.0.

De igual manera, dado que en las últimas versiones de la librería, se han hecho reestructuraciones en módulos y puesto que son necesarias la utilización de funciones incluidas en esos módulos extra, deberemos descargar la librería **opencv_contrib** en su versión 3.2, la misma que hemos utilizado anteriormente.

En un primer momento, será necesario descargar la librería para posteriormente su compilación. Para ello podemos ir directamente a la web (http://opencv.org/releases.html) y descargar el código fuente para sistemas Linux o visitando y descargando el repositorio desde GitHub (https://github.com/opencv/opencv).

Una vez hecho esto, lo que nos queda es compilar las librerías. Para ello nos dirigimos al directorio de OpenCV y creamos una

carpeta que podríamos llamar build.

```
$ cd opencv-3.2.0/
$ mkdir build
```

Para la compilación nos usaremos de una herramienta denominada Cmake que se utiliza para la generación y automatización de código. En este caso, utilizaremos la versión gráfica que puede resultar mucho más fácil de utilizar.

```
$ cmake-gui
```

Como resultado se nos abrirá la aplicación y tendremos que elegir el directorio que contendrá el código de OpenCV y los binarios se crearán en la carpeta “build” recientemente creada. Así mismo, una vez puesto las dos rutas y configurada la ruta para los módulos extra, presionaremos el botón “Configure” y una vez terminado el “Generate”. Con ello ya tendremos listo los Makefile necesarios para compilar la librería.

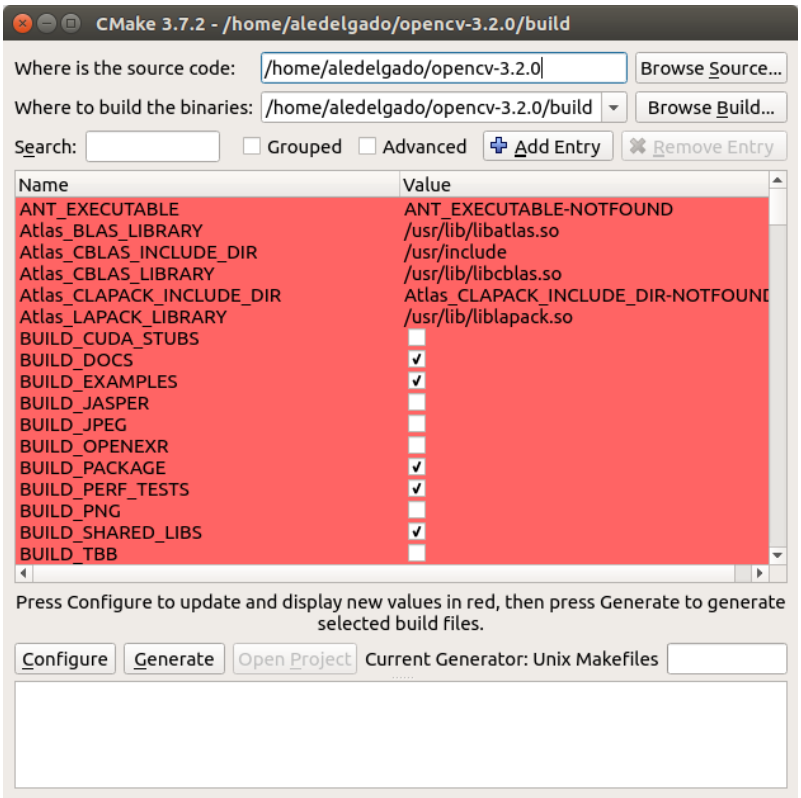



Figura 8: CMake gráfico para la compilación de OpenCV

Para finalizar con la compilación, situados en el directorio “build” ejecutamos los siguiente comandos:

```
$ make
$ sudo make install
```

Una vez terminados tendremos OpenCV listo para utilizar.

**Nota 1:** Los comandos anteriormente citados pueden tardar alrededor de dos horas en terminar de ejecutarse ya que nos encontramos en un sistema de bajo rendimiento como es la Raspberry Pi 3.

### 5.2.3 Descarga y compilación de la librería Raspicam

Otra de las librerías necesarias para la captura de las imágenes y la configuración de los parámetros de la cámara es Raspicam ya que nos provee de las clases necesarias para el control de la cámara además de contar con funciones especialmente implementadas para trabajar con OpenCV.

La instalación y compilación se realiza de la siguiente manera:

Primero descargamos el código desde SourceForge, en concreto la versión más actual, la 0.1.6 desde el siguiente enlace

https://sourceforge.net/projects/raspicam/files/raspicam-0.1.6.zip/download

Seguidamente, extraemos la carpeta desde el zip y como hemos realizado anteriormente con la compilación de OpenCV creamos una carpeta “build” y nos situamos en ella.

```
$ mkdir build
$ cd build
```

Ahora ejecutamos el comando:

```
$ cmake ..
```

Una vez terminado, si queremos comprobar que el módulo de OpenCV para Raspicam se compiló, en la ejecución del comando deberá aparecer lo siguiente:

```
CREATE OPENCV MODULE=1
```

Finalmente lo que quedará por hacer es ejecutar los siguientes comandos para terminar de configurar Raspicam

```
$ make
```

```
$ sudo make install
$ sudo ldconfig
```

**Nota 1:** El comando “ldconfig” creará los vínculos y caché necesarios para las bibliotecas compartidas más recientes.

### 5.2.4 Instalación del gestor de base de datos SQLite

Para el desarrollo del trabajo, es necesario la utilización de base de datos para el almacenamiento de los datos recogidos a través de los sensores, como pueden ser la temperatura, humedad, voltaje obtenido de la placa fotoeléctrica y la fecha y la hora de realización de la captura.

Utilizaremos un gestor de base de datos relacional y libre denominado SQLite, para ello únicamente deberemos instalar los paquetes mediante terminal y posteriormente incluir la librería en nuestro código cuando vayamos a utilizar la base de datos.

Para instalar SQLite y sus librerías ejecutamos los siguientes comandos por consola:

- `$ sudo apt-get install sqlite3`
- `$ sudo apt-get install libsqlite3-dev`

### 5.2.5 Instalación librerías boost

Boost es una librería desarrollada para C++, bajo software libre que ofrece multitud de funcionalidades.

Se trata de una biblioteca que usaremos principalmente por la buena gestión de la entrada/salida, concretamente para la comunicación mediante un puerto serial entre la Raspberry y Arduino que se ocupará del paso de información. Otro de los usos que se ha dado en este proyecto, es la interpretación de los datos pasados por la línea de comando.

Para su instalación, se ejecutará el siguiente comando en la terminal:

- `$ sudo apt-get install libboost-all-dev`


Figura 9: Gestor de base de datos SQLite

### 5.2.6 Configuración del cliente NFS

El uso de instalar un servicio NFS era imperativo, dado que la capacidad de la Raspberry puede llegar a ser muy limitada y ya que se trabajará en el procesamiento de imágenes, el tamaño y los datos que recogemos pueden sobrepasar el límite, es por ello que se decidió guardar todas las imágenes capturadas en un ordenador que hará de centro de procesamiento mediante la compartición de directorios en red gracias a NFS.

Para ello en la Raspberry se instalará un cliente nfs mediante el siguiente comando:

- `$ sudo apt-get install nfs-common`

Una vez hecho esto y teniendo configurado el servidor (ordenador de procesamiento), montaremos la carpeta compartida en nuestro ordenador de bolsillo:

- `sudo mount ip_servidor_nfs:ruta_directorio_a_montar ruta_directorio_raspberry`

### 5.2.7 Instalación Arduino

La instalación de Arduino es muy sencilla, únicamente tendremos que instalar el entorno de desarrollo como se ha venido haciendo en casos anteriores. En una terminal ejecutamos:

- `$ sudo apt-get install arduino`

Una vez instalado el entrono de desarrollo de Arduino, podemos diseñar el circuito con los diferentes sensores, este es un circuito muy sencillo y quedaría una vez montado de la siguiente manera:

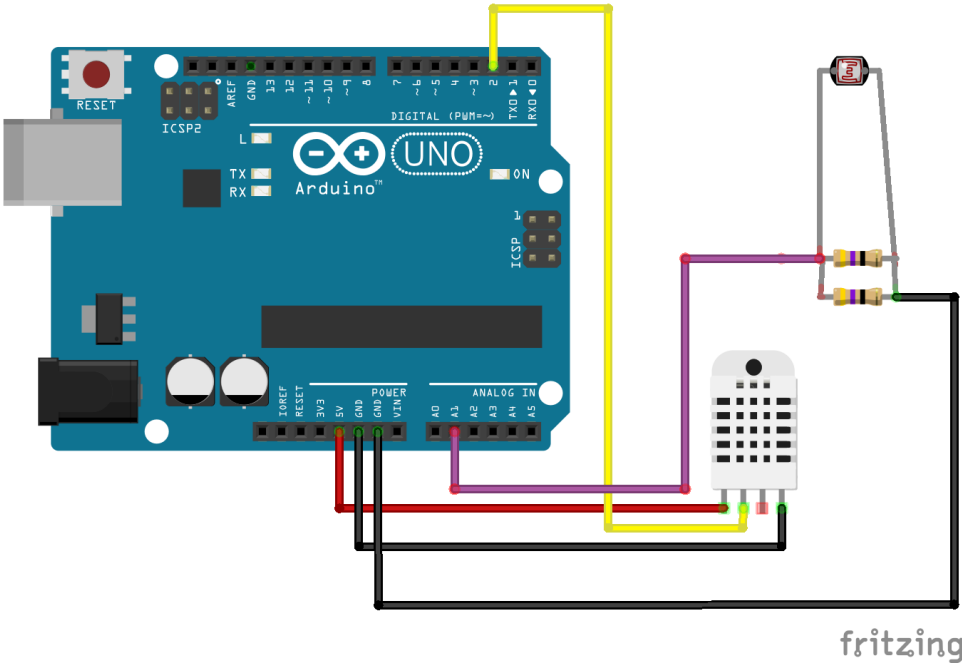



Figura 10: Esquemático Arduino y Sensores

## 5.3 Configuración equipo de procesamiento

En esta parte se detallará todo el proceso que se ha llevado a cabo para la configuración del equipo que desempeñará la tarea que tiene mayor coste computacional, el procesado de las imágenes, ya que como se ha comentado en apartados anteriores, la Raspberry no tiene la suficiente potencia para ofrecer resultados en periodos cortos de tiempo y de manera eficiente.

### 5.3.1 Configuración OpenCV, SQLite y boost

El proceso a llevar a cabo para la instalación y configuración de las diferentes librerías y el gestor de bases de datos, será en mismo que se ha realizado en la Raspberry, por ello omitiremos este paso .

### 5.3.2 Configuración Servidor NFS

En este caso, como se ha mencionado, el ordenador hará de servidor NFS donde se almacenarán las imágenes. Para ello instalaremos el servicio nfs-server:

- `$ sudo apt-get install nfs-server`

Posteriormente comprobaremos que el servicio está habilitado y configuraremos el fichero */etc/exports* con las carpetas que estamos

dispuestos a compartir, en este caso a todo el rango de direcciones IP privadas de nuestra red, con permisos de lectura y escritura.

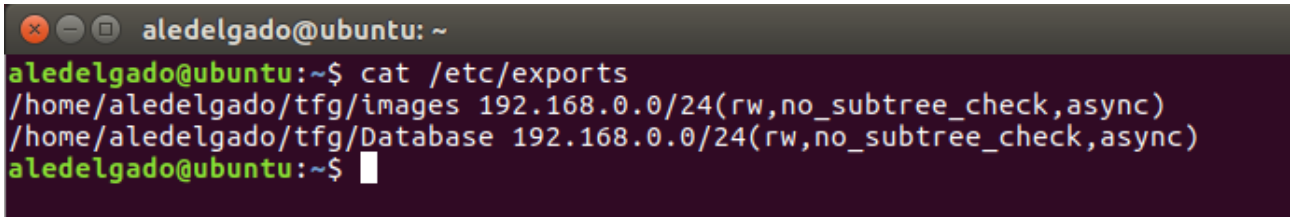



Figura 11: Fichero /etc/exports

## 5.4 Captura y almacenamiento de imágenes

Para testear los algoritmos utilizados tanto para la predicción del movimiento y comportamiento de las nubes como para la predicción fotovoltaica, se han recopilado una serie de imágenes, a lo largo de un periodo de tiempo y que servirán para ajustar los algoritmos antes de probarlos en tiempo real.

### 5.4.1 Captura secuencia imágenes con diferente ISO

El principal objetivo de este paso es la creación de una fusión imágenes a partir de una secuencia de tres imágenes capturadas aplicando en cada una de ellas diferentes tipo de ISO (Sensibilidad fotográfica que permite recoger mayor o menor cantidad de luz e información de la escena, aumentando o disminuyendo la corriente eléctrica que circula por las celdas fotosensibles del sensor de nuestra cámara).

El algoritmo utilizado, funciona de la siguiente manera.

Hemos creado un programa en C++ que utiliza dos threads, uno de ellos se encargará de ir grabando continuamente el entorno y extrayendo la fecha y hora del sistema. A la par, otro hilo se encargará de capturar las secuencias de imágenes, dicho hilo estará inactivo el tiempo “t” en segundos que se haya pasado por linea de comandos y una vez reactivado capturará las imágenes cambiando las ISO de cada una de ellas y las almacenará en el destino seleccionado (directorio compartido ubicado en el ordenador de procesamiento) cuyo nombre se tratará de la hora capturada seguido del índice de la imagen (1,2 o 3).

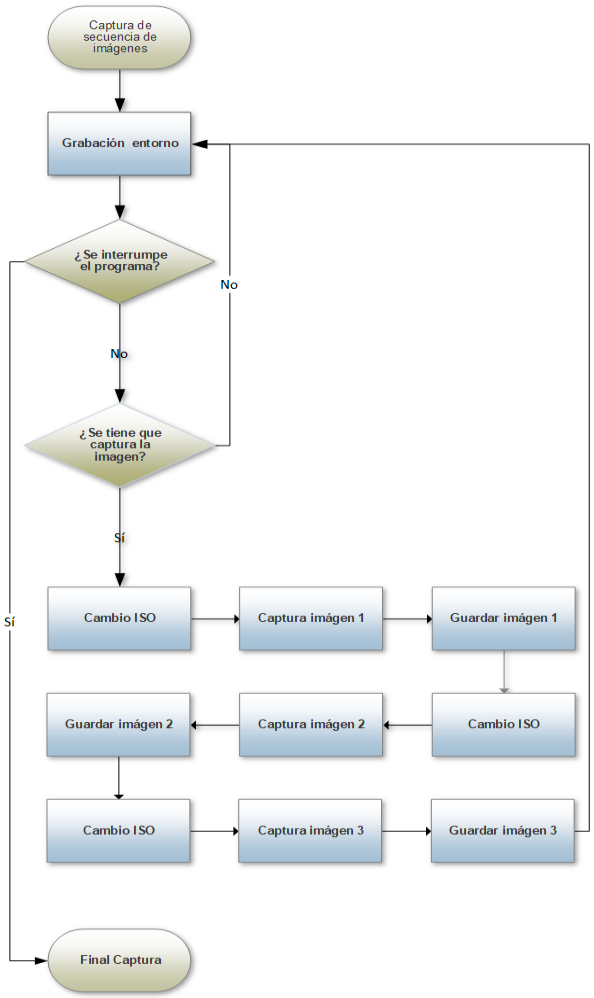



Figura 12: Generación de imágenes.
Diagrama flujo

### 5.4.2 Generación de imágenes fusionadas

La parte de generación de imágenes en alto rango dinámico se realiza en el ordenador o máquina destinada a procesar las imágenes ya que la fusión de las tres imágenes capturadas anteriormente puede llegar a ralentizar la propia Raspberry impidiendo así capturar y salvar correctamente imágenes en un rango de tiempo relativamente corto (20-30 segundos).

La creación de las imágenes resultantes se realiza en tiempo real, con un programa encargado de comprobar continuamente el directorio de destino de las imágenes mientras la Raspberry almacena las imágenes en dicho directorio, cuando se llega a un número de imágenes múltiplo de 3, se espera un cierto tiempo para que se escriban completamente y se reordenan todas las imágenes por el identificador del principio, seguidamente se realiza el merge (fusión) de las 3 últimas almacenando la imagen resultante en otro

directorio.

Para llevar a cabo dicha fusión, se utiliza principalmente la técnica de Mertens que se basa en la utilización de la técnica de la pirámide de Laplacian contenida en librería OpenCV:

- createMergeMertens()

donde pasamos al proceso un vector con las imágenes a fusionar y nos dará como resultado una imagen con valores de entre [0-1].

Además podemos aplicar un mapa tonal cambiando un poco la saturación y el contraste para obtener una mejor imagen.

Finalmente convertimos la imagen a 8-bits multiplicando por 255 y la almacenamos en nuestro directorio para posteriormente procesarla.

## 5.5 Recogida de datos

Para la recogida de los datos de interés, como ya hemos comentado, nos apoyaremos en una base de datos gestionada por SQLite que se creará en la propia Raspberry y donde se almacenará la información que nos proporcione los sensores controlados con Arduino.

La comunicación entre la Raspberry y Arduino se realizará a través de una conexión por el puerto serie.

### 5.5.1 Programa Arduino para la detección de parámetros

Para la extracción de la información de interés, utilizaremos Arduino, porque nos ofrece de manera sencilla y rápida las funcionalidades necesarias para nuestro cometido.

El principal objetivo es la captura de parámetros como pueden ser la humedad, la temperatura y la información que recojamos a través de una célula fotoeléctrica.

Para ello, se implementó un programa para Arduino que recoja continuamente estos datos ayudados de librerías externas para la mejor manipulación en este caso del sensor de humedad y temperatura DHT22, dicha librería es de software libre y provista por Adafruit.

En el caso de la placa fotoeléctrica, únicamente será necesario hacer alguna conversión y la utilización de los puertos analógicos de Arduino ya que su valor según la cantidad de energía solar que reciba variará entre los 0 y los 4 voltios.

### 5.5.2 Creación de la base de datos

La creación de la base de datos se realizará en la propia Raspberry y simplemente se tratará de una tabla que contendrá los siguientes campos:

- Identificador
- Hora
- Fecha
- Temperatura
- Humedad
- Voltaje recogido por la célula fotoeléctrica
- Estado
  - Despejado
  - Parcialmente nublado
  - Nublado

Tabla: DATOS

| | ID_NOMBRE | HORA | FECHA | TEMPERATURA | HUMEDAD | SOLAR | ESTADO |
|---|---|---|---|---|---|---|---|
| | Filtro | Filtro | Filtro | Filtro | Filtro | Filtro | Filtro |
| 1 | 0 | 14:24:45 | 2017/6/4 | 36 | 27 | 2.07 | NULL |
| 2 | 1 | 14:25:09 | 2017/6/4 | 34 | 27 | 1.86 | NULL |
| 3 | 2 | 14:25:33 | 2017/6/4 | 33 | 27 | 3.7 | NULL |
| 4 | 3 | 14:25:58 | 2017/6/4 | 34 | 27 | 5.0 | NULL |
| 5 | 4 | 14:26:23 | 2017/6/4 | 35 | 27 | 5.0 | NULL |
| 6 | 5 | 14:26:48 | 2017/6/4 | 36 | 27 | 5.0 | NULL |
| 7 | 6 | 14:27:13 | 2017/6/4 | 36 | 26 | 5.0 | NULL |

Figura 13: Ejemplo de los datos recogidos

Cada vez que se capturen nuevos datos asociados a el estado actual del cielo y entorno, se copiarán al ordenador de procesado a través del servicio NFS para posteriormente poder realizar la predicción.

### 5.5.3 Paso de los datos

Mientras se va realizando la captura de imágenes, cuando se termina de capturar la última de las tres imágenes que generan la fusión, se recurre a una función que envía por el serial al Arduino los comandos (caracteres que se procesan en Arduino) para la temperatura, humedad y célula (‘t’,’h’ y ‘s’) y recibe la respuesta con los datos para posteriormente introducirlos en la base de datos.

Toda la comunicación se lleva a cabo gracias a las librerías boost utilizando **serial_port** para la el paso de datos a través del serial. Las funciones que principalmente se utilizan son:

- boost::asio::write(SyncWriteStream& s,
  const ConstBufferSequence& buffer)
- boost::asio::read_until(SyncReadStream& s,
  boost::asio::basic_streambuf<Allocator>& b, char delimitador)

Con la primera se escriben los comandos y con la segunda se almacenan los datos recibidos utilizando como delimitador en nuestro caso los saltos de línea.

## 5.6 Procesamiento de imágenes

En este apartado realizaremos el tratamiento y procesado de las imágenes mediante procedimientos y técnicas contenidas en la librería OpenCV.

Como se ha comentado anteriormente esta parte se realiza íntegramente en el ordenador o máquina de procesamiento y se diferenciarán varios pasos que deberemos tomar para conseguir nuestro objetivo, una estimación del estado del cielo (nubes y Sol) según varía el tiempo y la meteorología.

El primer paso es la detección del propio Sol o una estimación del área en el que se encuentra, seguido del calculo de vectores de desplazamiento de las nubes y finalmente una estimación o predicción fotovoltaica. Para la detección del Sol no será necesaria realizarla en el procesado de cada una de las imágenes puesto que el desplazamiento será mínimo.

### 5.6.1 Detección del Sol

La detección es uno de los pasos fundamentales para el desarrollo del proyecto. Por ello y dependiendo de las condiciones meteorológicas y lumínicas, deberemos aplicar dos técnicas diferentes para conseguir nuestro fin.

Diferenciaremos varios casos que se pueden dar:

- El Sol está despejado y se puede diferencias a simple vista su propia forma.
- El Sol se encuentra parcialmente nublado y su forma circular no se puede distinguir correctamente.
- El Sol está completamente tapado pero se puede intuir su posición gracias al brillo de las nubes.
- El Sol no se diferencia puesto que las nubes son muy densas y no se diferencia el brillo.

Para intentar lograr un buen reconocimiento y aproximación de la posición del Sol, utilizaremos dos técnicas de reconocimiento, la primera es una técnica de detección por contornos buscando un elemento circular, mientras que la segunda es una técnica de detección de color en la que utilizaremos umbrales (colores blancos brillantes) y detectaremos las zonas más luminosas del cielo.

### *1 Detección mediante contornos*

Para llevar a cabo la detección por contornos, primero podemos intentar retocar la imagen para facilitar el trabajo al algoritmo de detección de contornos.

En primer lugar, aplicaremos un filtro bilateral (**bilateralFilter()**) que reducirá el ruido no deseado en la imagen manteniendo los bordes.

Seguidamente, convertimos la imagen una vez aplicado el filtro a escala de grises y generamos una imagen de binaria (negros y blancos) aplicando un umbral con el objetivo de segmentar la imagen y quedarnos con el objeto (en este caso el Sol) que nos interese, para ello hemos utilizado la función **threshold()** y aplicando un umbral (235-255) que hemos detectado mediante ensayo-error para ajustarnos lo máximo posible al contorno del Sol.

Figura 14: Detección por contornos

Una vez tengamos la imagen binaria con el umbral, utilizamos la función **findCongours()** de la que obtendremos un vector de puntos con cada contorno, posteriormente utilizando la función **minEnclosingCircle()** detectamos los círculos y obtenemos el centro y el radio de los mismos. Únicamente tendremos que buscar que radio se aproxima más al tamaño del Sol si existiera más de un contorno circular.

### *2 Detección por brillo*

Si no se han detectado contornos que correspondan al Sol, utilizaremos una técnica de detección por brillo que hace uso de umbrales (colores) para detectar aquella región que tendrá la mayor probabilidad de encontrarse el Sol, en nuestro caso después de estudiar algunas capturas, las regiones y colores próximas al Sol de ellas obtenemos un valor mínimo y máximo de color RGB:

- LOW: RGB(228,230,237) Tonalidad grisácea
- HIGH: RGB(255,255,255) Blanco puro

Una vez tenemos los umbrales utilizamos la función **inRange()** para obtener la máscara binaria que contendrá la imagen del Sol. El siguiente paso a dar, es eliminar ruidos e intentar aproximarnos lo máximo posible a la posición real del Sol puesto que puede darse el caso de que nubes próximas tengan la misma tonalidad.

Para la eliminación de los ruidos y el cálculo del centro, utilizamos los momentos de la imagen binaria con los que calculamos su centro y por ende el centro del cuerpo celeste.

Figura 15: Detección por color (brillo)

### 5.6.2 Cálculo vectores de movimiento

Una vez hayamos detectado el Sol, deberemos trabajar sobre las nubes, intentando determinar su movimiento (dirección y velocidad) para conseguir este objetivo, la idea es intentar detectar puntos de interés en la imagen anterior a la que estamos procesando actualmente y mediante técnicas de flujo óptico, determinar donde estarían esos puntos en la imagen procesada actual obteniendo así un desplazamiento, es decir, el movimiento en nuestro caso de la nube.

Actualmente el calculo de vectores se realiza en las áreas próximas al Sol, intentando así aumentar la precisión de la predicción.

Además la imagen, se divide en áreas determinadas por la persona como si de una matriz se tratase y se procesa cada área de la imagen.

Como ya se ha comentado, para el calculo de los vectores de movimiento, se han utilizado principalmente dos funciones. La primera de ellas trata de calcular los puntos de interés de la imagen, en nuestro caso, los puntos se extraerán de la imagen anterior a la procesada para poder establecer el movimiento. Para ello utilizaremos la función **goodFeaturesToTrack()** de la que obtendremos un vector de puntos (**Point2f**) con los puntos de interés. El cálculo de dichos puntos se realiza de la siguiente manera:

1. Se calcula la calidad de cada píxel para determinar si se trata de una esquina utilizando por ejemplo el algoritmo de Harris-Stephens.
2. Posteriormente se descartan puntos obtenidos si se encuentran fuera del umbral de calidad para el punto establecido.
3. Se ordenan los puntos de manera descendiente según la calidad.
4. Y por último se desechan los puntos si existe una esquina de mayor calidad dentro de un rango de cercanía que se establece. Sin embargo, en nuestro caso no desecharemos ninguna esquina (punto de interés) puesto que establecemos esa distancia a 0.

Una vez obtengamos los puntos de interés de la imagen anterior y los tengamos almacenados en un vector, intentaremos calcular la

posición de ese punto en la imagen actual que estemos procesando para determinar el desplazamiento de las nubes. Para ello aplicamos un algoritmo de estimación de flujo óptico como es el método de Lucas-Kanade piramidal mediante la función implementada en OpenCV **calcOpticalFlowPyrLK()** con la que obtendremos otro vector de puntos que corresponderán a los punto en la imagen actual de los puntos de interés anteriormente extraídos.

Figura 16: Puntos de interés de la imagen

Una vez tenemos los dos vectores con los puntos podremos generar un vector que corresponderá con el movimiento de las nubes. En un primer momento, podemos generar un gran número de vectores sin embargo se ha decidido tratar la imagen como una matriz y dividirla en filas y columnas como si de una maya se tratara para procesar cada región que contenga vectores de movimiento por separado.

Se pueden elegir el numero de filas y columnas que el usuario

desee sin embargo para obtener mejores resultados es recomendable utilizar filas múltiplo de 12 y columnas múltiplo de 8. Una vez decidamos el número de áreas que queremos que tenga la imagen, procesaremos cada área una a una.


Figura 17: Múltiples vectores de movimiento

Para ello, detectamos los vectores que pertenezcan a el área en concreto a procesar ya que nuestro objetivo será el de calcular un vector promedio por cada zona. Los puntos del vector euclidiano (base y punto final) se almacenan en un vector y posteriormente se calcula un punto medio entre todos los puntos que corresponden con las bases de los vectores.

Realizaremos el mismo procedimiento para los puntos finales de los vectores euclidianos y con ello obtendremos un vector que puede representar el sentido y el módulo de la media de todos los vectores correspondientes a esa zona, obteniendo así un único vector.

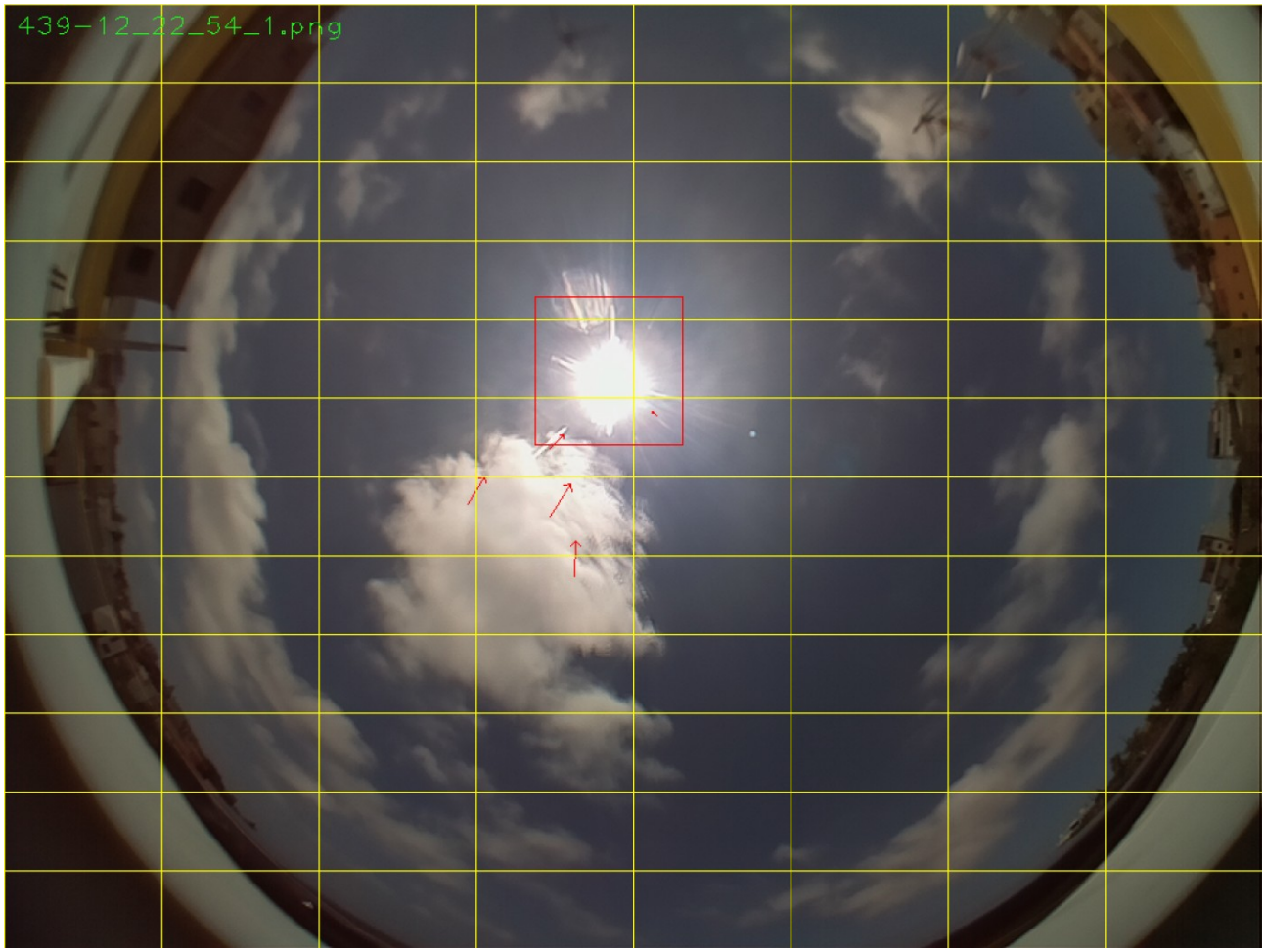


Figura 18: Vectores de movimiento por cada zona

### 5.6.3 Algoritmo de predicción fotovoltaica

Si ya se han detectado la zona donde se encontrará el Sol y se han calculado los vectores del movimiento de las nubes, podremos llevar a cabo una estimación del tiempo que tardará una nube en ocultar el Sol.

Para ello se han tenido en cuenta diversos factores, como puede ser el sentido del vector,que nos determinará si la nube se va a aproximar a la estrella, la velocidad de desplazamiento que lleva la nube, el tiempo o los cambios del tipo de detección del Sol, debido a que si se realiza un detección por brillo por ejemplo, querrá decir que el contorno del Sol no se diferencia y por ende que el Sol está parcialmente oculto y la cantidad de energía recogida por la célula fotovoltaica debería ser menor que en caso de estar incidiendo la luz directamente. O si por el contrario no se detecta el Sol ni por contornos ni por brillo, podremos afirmar que el Sol se encuentra completamente oculto y la cantidad de energía será mínima muy cercana al 0.

Para la predicción, primero deberemos tener claro la distancia en píxeles que tiene el vector de movimiento. Como se trata al fin y al cabo de dos puntos de una imagen, podemos calcular la distancia con la siguiente formula:

$$d=\sqrt{((x_2-x_1)^2+(y_2-y_1)^2)}$$

Figura 19: Distancia entre puntos

Si el vector tiene una distancia considerable será candidato a la predicción, es nuestro caso establecemos esa distancia en vectores mayor o igual a 15 píxeles. Una vez hecho esto, calculamos las componentes X e Y del vector, teniendo en cuenta siempre que en el caso de las imágenes no corresponde con eje de coordenadas cartesianas como el que estamos acostumbrados, sino que se trata de una matriz de tamaño (1024x768) por ello todos los puntos serán positivos. Debido a esto, en el cálculo de la componente Y del vector, deberemos negarla una vez calculada para que el resultado obtenido sea correcto.

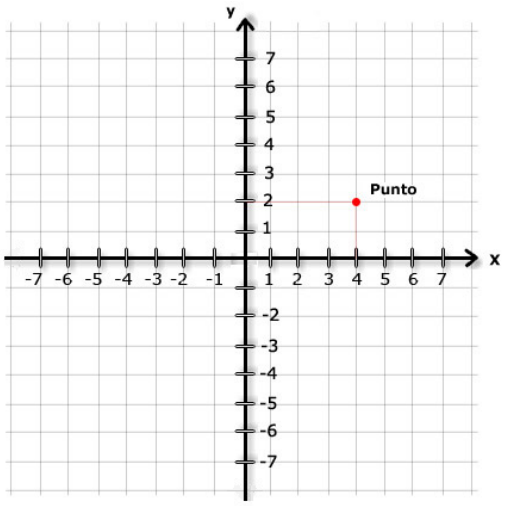



Figura 20: Eje de coordenadas cartesianas (2D)

Lo que quedaría por hacer con los vectores es calcular su ángulo , y esta tarea resulta muy fácil ya que conocemos las componentes X e Y del vector, por ello aplicamos la siguiente fórmula:

$$\alpha=\frac{\arctan((C1_Y-C2_Y)/(C1_X-C2_X))*360}{2*\pi}$$

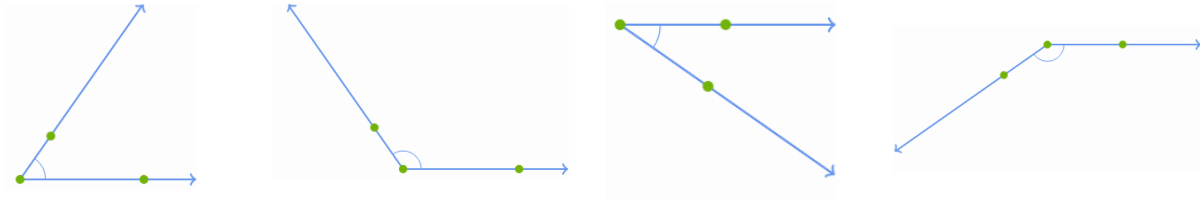
Figura 21: Ángulo del vector (Grados)

Una vez obtenemos tanto el módulo como el ángulo de los vectores, la idea es detectar si el vector se dirige a la región donde se encuentra el Sol, por ello, para cada vector calculamos la trayectoria que tendría hasta el final de la imagen o bien la región del Sol. Si nuestro vector se dirige hacia el cuerpo celeste, procesaremos ese vector para intentar realizar una estimación de donde se encontrará respecto al tiempo, calculando así la llegada de la nube hasta la región deseada.

Para realizar la trayectoria de los vectores, tendremos que tener muy presente el ángulo ya que corresponderá a la dirección que tome nuestro vector y calculamos los nuevos puntos de la trayectoria desde el final del vector a procesar cada uno con una separación de 1/4 parte del módulo del vector. Nuevamente tendremos que tener en cuenta que nos movemos en una matriz y no en un eje cartesiano, así mismo el ángulo podrá tomar diferentes valores:

- α = 0º α = 90º α = 180º α = -90º

- 90º > α > 0º 180º > α > 90º 0º > α > -90º -90º > α > -180º



Si ya tenemos calculadas las trayectorias, procesamos aquellas que interceptarán con la zona que contendrá el Sol.

Primero calculamos la distancia que separa el punto final del vector con el Sol, dado que tenemos el último punto que hemos calculado con el producto del vector por el escalar y que estará muy próximo a la zona del Sol unido al punto final del vector, determinamos con la formula anteriormente expuesta para el cálculo de distancias, la distancia en píxeles que separa al vector de la región de la estrella.

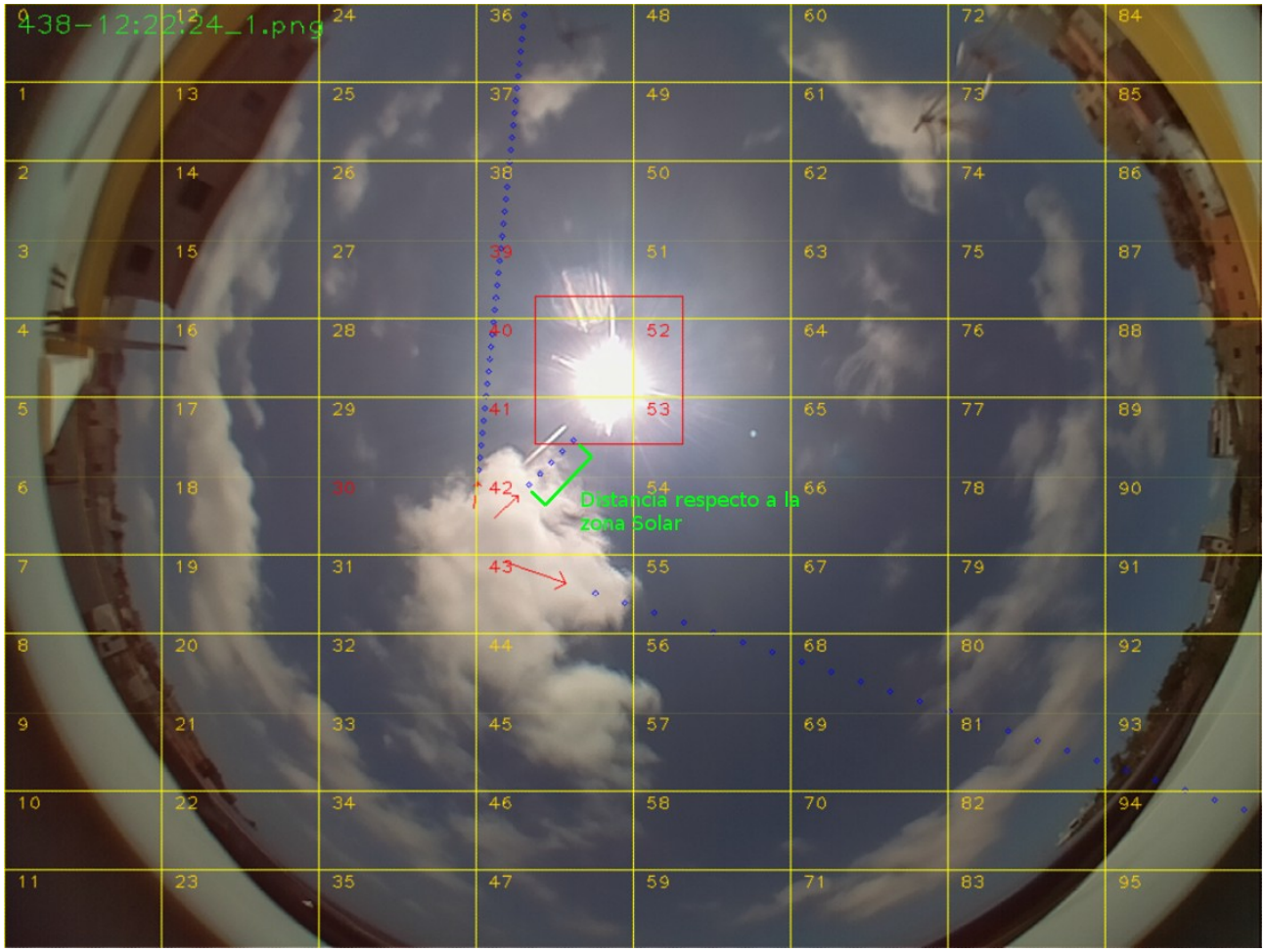


Figura 22: Distancia entre el vector y la zona Solar

Seguidamente, calcularemos el paso del tiempo entre la imagen anterior y la que estamos procesando gracias a la información recogida a la hora de realizar la captura, en el caso que estamos analizando la imagen de la **Figura 21** se ha realizado a las 12:22:24 y la imagen anterior a las 12:21:52, sabiendo esto pasaremos la hora a segundos y hallaremos la diferencia obteniendo así el tiempo que transcurre entre las capturas.

Si tenemos el tiempo y la distancia calcularemos la velocidad en **píxeles/segundos** del desplazamiento de la nube y realizando una regla de tres, podremos establecer una estimación sabiendo la velocidad, la distancia al Sol y el tamaño del vector para obtener el tiempo aproximado en el que la nube ocultará la estrella.

Estos cálculos unido al tipo de detección o si el propio vector se encuentra completamente dentro de la región que acota al Sol, pueden arrojarnos mayor precisión a la hora de realizar la predicción y poder hacer una estimación de la energía recogida.

## 5.7 Evaluación de resultados

Para evaluar los resultados obtenidos, utilizaremos una secuencia de algunas imágenes en las que podemos observar como la nube se dirige hacia el Sol y llega a taparlo parcialmente.

En las pruebas realizadas hemos realizado capturas cada 30 segundos aproximadamente ya que según la experiencia, las nubes varían en forma, tamaño y desplazamiento a gran velocidad y es por ello que las capturas se realizan en un corto periodo de tiempo.

En las **Figuras 25** y **26** se muestran una secuencias de imágenes procesadas donde se calculan los vectores de movimiento y previamente la detección del Sol. Además los resultados que hemos obtenido se generan como salida a un fichero que preferiblemente será del tipo **csv** u **odt** como si de una hoja de cálculo se tratara.

Los parámetros que se recogen son los siguientes:

- Nombre de la imagen
- Área en la que se encuentra el vector
- Punto base del vectores
- Distancia (Módulo del vector)
- Ángulo del vector
- Aproximación al Sol
- Hora de la captura
- Fecha de la captura
- Tiempo de llegada de la nube al Sol
- Hora de la llegada (HH:MM:SS)
- Velocidad de la nube (pixel/segundos)
- Temperatura (ºC)
- Humedad (%)
- Energia Solar Recogida [0-5]

Estudiando la secuencia de imágenes se puede observar que en el primer procesamiento de la imagen **438** claramente un vector se dirige claramente hacia la nube a una velocidad de **0,87 px/s** y se estima el tiempo de llegada en **1 min y 42 segundos**, sobre las **12:24:04** horas. Si observamos la imagen más próxima a la hora estipulada que sería la captura **442** podemos observar que la nube tapa parcialmente el Sol y el filtro Solar que en un primer momento tenía un valor máximo de **5,0** puesto que los rayos solares incidían directamente, ahora toma un valor de **2,9** evidenciándose que existe un claro descenso de la energía recogida.

A continuación se mostrará como afecta el índice de insolación según el comportamiento de las nubes y la disposición del propio Sol, teniendo el cuenta el ángulo de incidencia de los rayos solares.

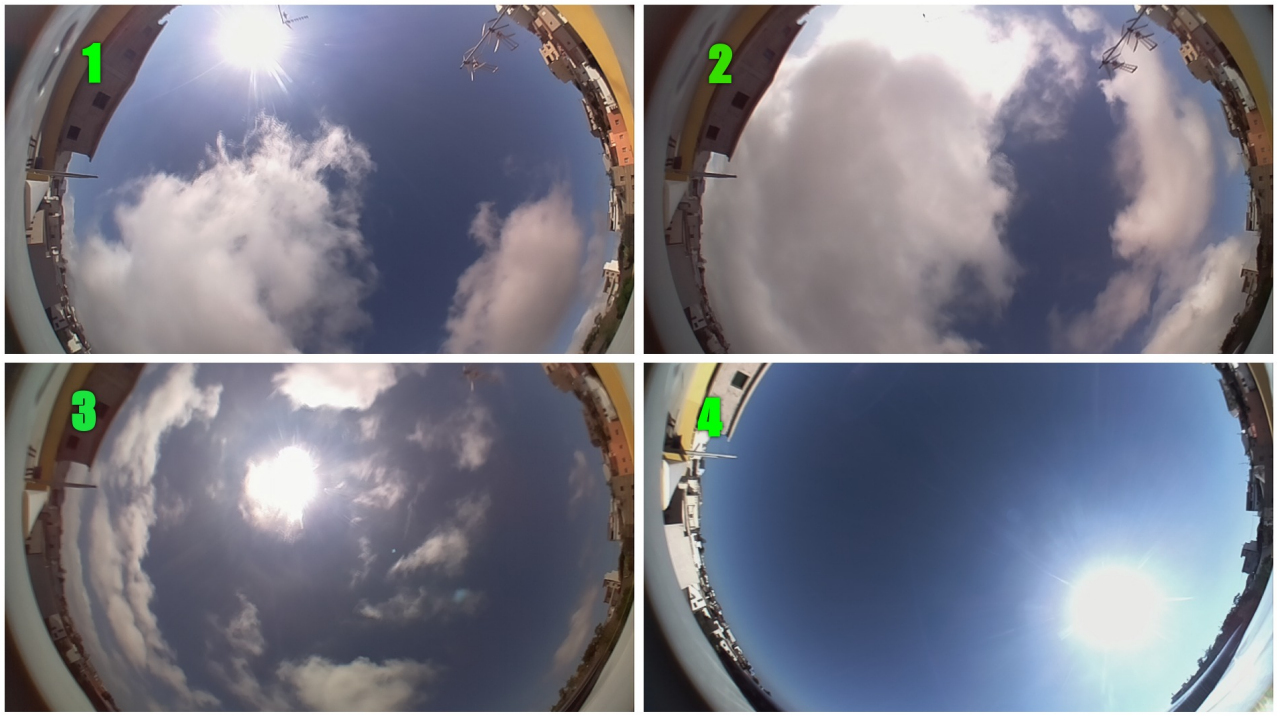



Figura 23: Variación en la insolación

Las imágenes de la figura anterior corresponden todas al mismo día, el 12 de junio de 2017.

La número uno se capturó a las **08:43:53** horas y se registró un valor de insolación de **3,18 sobre 5**.

La segunda imagen fue tomada a las **08:54:22** y claramente se puede ver como la nube tapa de manera significativa al Sol, en este caso la insolación registrada fue de **1,04**.

La tercera imagen corresponde a las **11:47:26** horas y en este caso a penas hay nubes suficientemente densas para ocultar a la estrella y por tanto y más tratándose a dicha hora donde los rayos inciden de forma  casi directa a la célula, se registró una insolación máxima tomando un valor de **5,0**.

Por último, la cuarta fotografía se capturó a las **17:05:16** donde se observa claramente que el Sol no se encuentra alineado con la célula fotovoltaica y aunque no existan nubes que intercedan con lo rayos, se obtuvo una insolación de **2,88**.

Las capturas de las imágenes del cielo se han realizado en Chío, perteneciente al municipio de Guía de Isora y realizando una estimación con la altura a la que se encuentran las nubes en el trópico (400 m) y un  ángulo de visión de aproximadamente unos 140 grados. El área del cielo que es capturada se observa en la siguiente figura.


Figura 24: Área de captura de la cámara

Aproximadamente contará con un radio de 476,70 metros, lo que hacen 713.904,59 m².

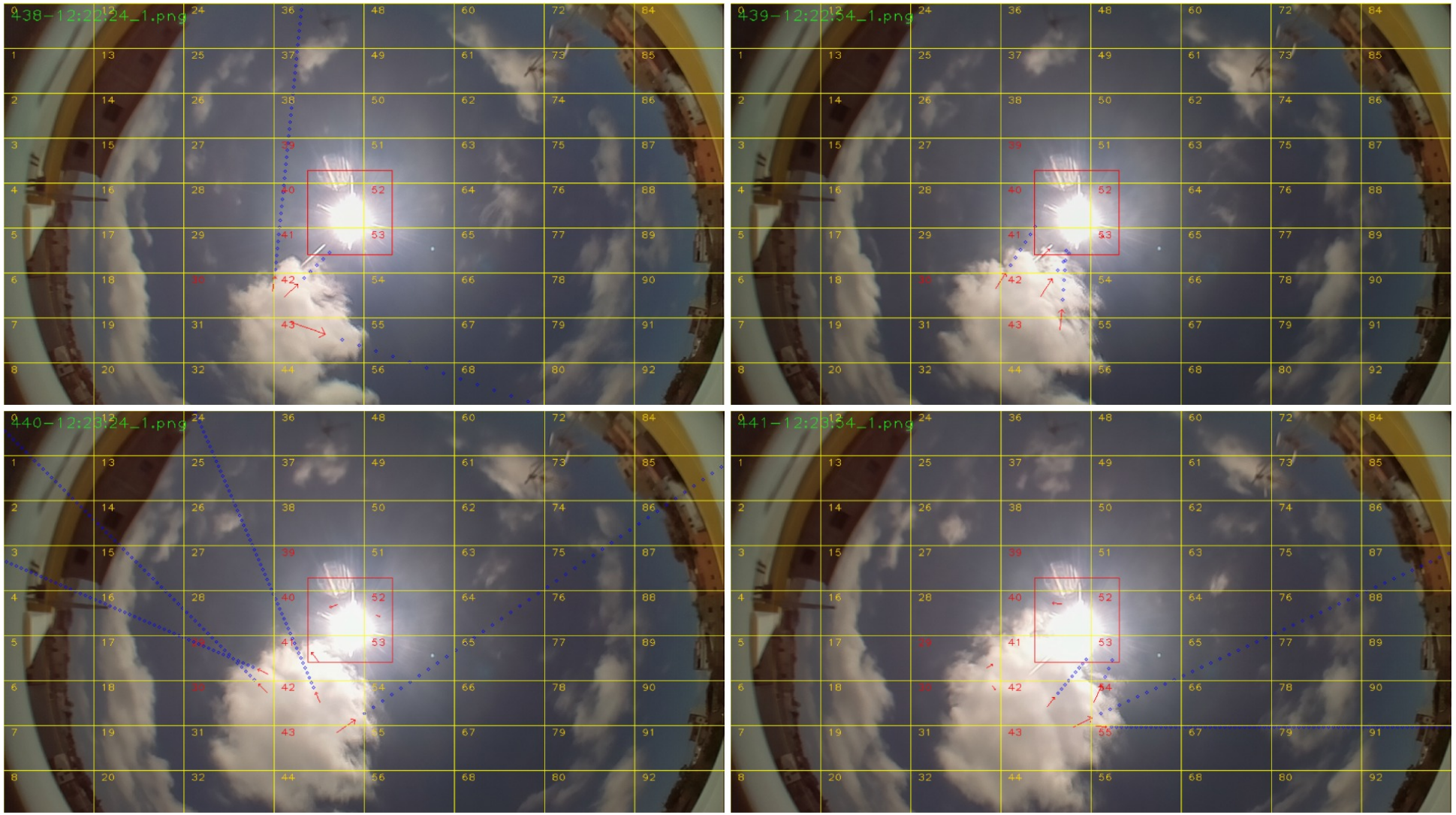



Figura 25: Secuencia 1

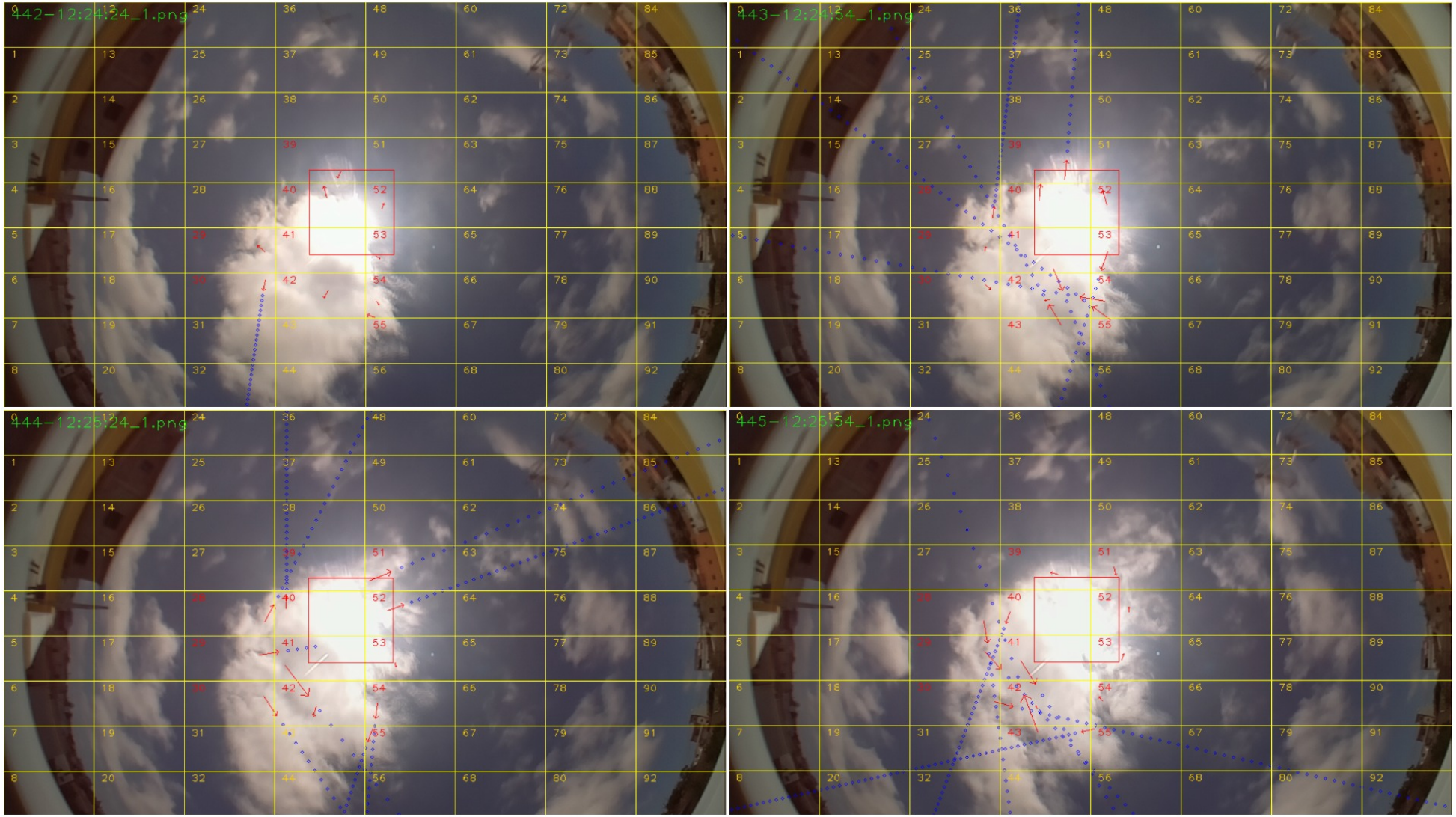



Figura 26: Secuencia 2

| Nombre_Imagen | Región | Base_vector | Distancia | Ángulo | Aproxima_Sol?? | Hora | Fecha | Llegada_aproximada | Hora_llegada | Velocidad (pixel/sec) | Temperatura | Humedad | Solar Actual | Solar Imagen Siguiente |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 438-12:22:24_1.png | 30 | [382.000000 410.000000] | 21.213203 | 81.869898 | No | 12:22:22 | 2017/6/12 | -1 | -1 | 0.000000 | 33ºC | 28% | 5.000000 | 0 |
| 438-12:22:24_1.png | 42 | [399.000000 418.000000] | 26.172505 | 43.451842 | Sí | 12:22:22 | 2017/6/12 | 0 horas 1 minutos y 42 segundos | 12:24:4 | 0.872417 | 33ºC | 28% | 5.000000 | [2.5 - 5] |
| 438-12:22:24_1.png | 43 | [408.000000 454.000000] | 51.865210 | -19.133643 | No | 12:22:22 | 2017/6/12 | -1 | -1 | 0.000000 | 33ºC | 28% | 5.000000 | 0 |
| | | | | | | | | | | | | | | |
| 439-12:22:54_1.png | 30 | [377.000000 406.000000] | 26.076810 | 57.528808 | Sí | 12:22:52 | 2017/6/12 | 0 horas 2 minutos y 0 segundos | 12:24:52 | 0.869227 | 33ºC | 28% | 5.000000 | [2.5 - 3.5] |
| 439-12:22:54_1.png | 41 | [442.000000 361.000000] | 15.556349 | 45.000000 | Sí | 12:22:52 | 2017/6/12 | 0 horas 0 minutos y 30 segundos | 12:23:22 | 0.518545 | 33ºC | 28% | 5.000000 | [2.5 - 3.5] |
| 439-12:22:54_1.png | 42 | [441.000000 417.000000] | 29.681644 | 57.380757 | Sí | 12:22:52 | 2017/6/12 | 0 horas 1 minutos y 13 segundos | 12:24:5 | 0.989388 | 33ºC | 28% | 5.000000 | [2.5 - 3.5] |
| 439-12:22:54_1.png | 43 | [468.000000 465.000000] | 29.154759 | 84.093859 | Sí | 12:22:52 | 2017/6/12 | 0 horas 1 minutos y 56 segundos | 12:24:48 | 0.971825 | 33ºC | 28% | 5.000000 | [1.5 - 3.0] |
| | | | | | | | | | | | | | | |
| 440-12:23:24_1.png | 29 | [375.000000 375.000000] | 15.652476 | 153.434949 | No | 12:23:22 | 2017/6/12 | -1 | -1 | 0.000000 | 33ºC | 28% | 5.000000 | 0 |
| 440-12:23:24_1.png | 30 | [374.000000 401.000000] | 16.970563 | 135.000000 | No | 12:23:22 | 2017/6/12 | -1 | -1 | 0.000000 | 33ºC | 28% | 5.000000 | 0 |
| 440-12:23:24_1.png | 41 | [447.000000 357.000000] | 16.401219 | 127.568592 | Sí | 12:23:22 | 2017/6/12 | Nube tapa parcialmente el Sol | Actualmente | 0.546707 | 33ºC | 28% | 5.000000 | [3.5 - 5] |
| 440-12:23:24_1.png | 42 | [449.000000 415.000000] | 15.231546 | 113.198591 | No | 12:23:22 | 2017/6/12 | -1 | -1 | 0.000000 | 33ºC | 28% | 5.000000 | 0 |
| 440-12:23:24_1.png | 43 | [473.000000 458.000000] | 31.622777 | 34.695154 | No | 12:23:22 | 2017/6/12 | -1 | -1 | 0.000000 | 33ºC | 28% | 5.000000 | 0 |
| | | | | | | | | | | | | | | |
| 441-12:23:54_1.png | 42 | [450.000000 421.000000] | 16.401219 | 52.431408 | Sí | 12:23:52 | 2017/6/12 | 0 horas 2 minutos y 38 segundos | 12:26:30 | 0.546707 | 34ºC | 28% | 5.000000 | [2.5 - 5] |
| 441-12:23:54_1.png | 43 | [487.000000 450.000000] | 29.068884 | 26.565051 | No | 12:23:52 | 2017/6/12 | -1 | -1 | 0.000000 | 34ºC | 28% | 5.000000 | 0 |
| 441-12:23:54_1.png | 54 | [516.000000 415.000000] | 26.400758 | 65.376435 | Sí | 12:23:52 | 2017/6/12 | 0 horas 1 minutos y 14 segundos | 12:25:6 | 0.880025 | 34ºC | 28% | 5.000000 | [3.5 - 5] |
| 441-12:23:54_1.png | 55 | [519.000000 449.000000] | 15.033296 | -3.814075 | No | 12:23:52 | 2017/6/12 | -1 | -1 | 0.000000 | 34ºC | 28% | 5.000000 | 0 |
| | | | | | | | | | | | | | | |
| 442-12:24:24_1.png | 30 | [370.000000 394.000000] | 15.297059 | -101.309932 | No | 12:24:22 | 2017/6/12 | -1 | -1 | 0.000000 | 34ºC | 28% | 2.900000 | 0 |
| 442-12:24:24_1.png | 40 | [457.000000 277.000000] | 16.492423 | 104.036243 | Sí | 12:24:22 | 2017/6/12 | Nube tapa parcialmente el Sol | Actualmente | 0.549747 | 34ºC | 28% | 2.900000 | [3.5 - 5] |
| | | | | | | | | | | | | | | |
| 443-12:24:54_1.png | 28 | [373.000000 306.000000] | 16.124515 | 82.874984 | No | 12:24:52 | 2017/6/12 | -1 | -1 | 0.000000 | 32ºC | 28% | 5.000000 | 0 |
| 443-12:24:54_1.png | 39 | [475.000000 251.000000] | 27.166155 | 83.659808 | No | 12:24:52 | 2017/6/12 | -1 | -1 | 0.000000 | 32ºC | 28% | 5.000000 | 0 |
| 443-12:24:54_1.png | 40 | [439.000000 281.000000] | 23.086793 | 85.030259 | Sí | 12:24:52 | 2017/6/12 | Nube tapa parcialmente el Sol | Actualmente | 0.769560 | 32ºC | 28% | 5.000000 | [3.5 - 5] |
| 443-12:24:54_1.png | 41 | [459.000000 378.000000] | 33.615473 | -67.249024 | No | 12:24:52 | 2017/6/12 | -1 | -1 | 0.000000 | 32ºC | 28% | 5.000000 | 0 |
| 443-12:24:54_1.png | 43 | [470.000000 458.000000] | 33.615473 | 120.379126 | No | 12:24:52 | 2017/6/12 | -1 | -1 | 0.000000 | 32ºC | 28% | 5.000000 | 0 |
| 443-12:24:54_1.png | 52 | [535.000000 287.000000] | 20.880613 | 106.699244 | Sí | 12:24:52 | 2017/6/12 | Nube tapa parcialmente el Sol | Actualmente | 0.696020 | 32ºC | 28% | 5.000000 | [3.5 - 5] |
| 443-12:24:54_1.png | 53 | [536.000000 354.000000] | 27.513633 | -109.093492 | No | 12:24:52 | 2017/6/12 | -1 | -1 | 0.000000 | 32ºC | 28% | 5.000000 | 0 |
| 443-12:24:54_1.png | 54 | [532.000000 424.000000] | 35.510562 | 170.272421 | No | 12:24:52 | 2017/6/12 | -1 | -1 | 0.000000 | 32ºC | 28% | 5.000000 | 0 |
| 443-12:24:54_1.png | 55 | [540.000000 451.000000] | 31.400637 | 142.765166 | No | 12:24:52 | 2017/6/12 | -1 | -1 | 0.000000 | 32ºC | 28% | 5.000000 | 0 |
| | | | | | | | | | | | | | | |
| 444-12:25:24_1.png | 28 | [370.000000 300.000000] | 27.294688 | 61.557071 | No | 12:25:22 | 2017/6/12 | -1 | -1 | 0.000000 | 33ºC | 28% | 4.660000 | 0 |
| 444-12:25:24_1.png | 29 | [363.000000 347.000000] | 27.294688 | 8.426969 | Sí | 12:25:22 | 2017/6/12 | 0 horas 1 minutos y 27 segundos | 12:26:49 | 0.909823 | 33ºC | 28% | 4.660000 | [3.5 - 5] |
| 444-12:25:24_1.png | 30 | [369.000000 406.000000] | 32.449961 | -56.309932 | No | 12:25:22 | 2017/6/12 | -1 | -1 | 0.000000 | 33ºC | 28% | 4.660000 | 0 |
| 444-12:25:24_1.png | 40 | [400.000000 281.000000] | 18.027756 | 86.820170 | No | 12:25:22 | 2017/6/12 | -1 | -1 | 0.000000 | 33ºC | 28% | 4.660000 | 0 |
| 444-12:25:24_1.png | 41 | [399.000000 362.000000] | 54.203321 | -52.495858 | No | 12:25:22 | 2017/6/12 | -1 | -1 | 0.000000 | 33ºC | 28% | 4.660000 | 0 |
| 444-12:25:24_1.png | 51 | [518.000000 242.000000] | 33.615473 | 22.750976 | No | 12:25:22 | 2017/6/12 | -1 | -1 | 0.000000 | 33ºC | 28% | 4.660000 | 0 |
| 444-12:25:24_1.png | 52 | [544.000000 284.000000] | 24.351591 | 19.179008 | No | 12:25:22 | 2017/6/12 | -1 | -1 | 0.000000 | 33ºC | 28% | 4.660000 | 0 |
| 444-12:25:24_1.png | 54 | [530.000000 415.000000] | 22.203603 | -97.765166 | No | 12:25:22 | 2017/6/12 | -1 | -1 | 0.000000 | 33ºC | 28% | 4.660000 | 0 |
| 444-12:25:24_1.png | 55 | [523.000000 452.000000] | 19.313208 | -111.250506 | No | 12:25:22 | 2017/6/12 | -1 | -1 | 0.000000 | 33ºC | 28% | 4.660000 | 0 |
| | | | | | | | | | | | | | | |
| 445-12:25:54_1.png | 28 | [361.000000 300.000000] | 31.400637 | -80.837653 | No | 12:25:52 | 2017/6/12 | -1 | -1 | 0.000000 | 33ºC | 28% | 5.000000 | 0 |
| 445-12:25:54_1.png | 29 | [363.000000 344.000000] | 32.557641 | -47.489553 | No | 12:25:52 | 2017/6/12 | -1 | -1 | 0.000000 | 33ºC | 28% | 5.000000 | 0 |
| 445-12:25:54_1.png | 30 | [376.000000 414.000000] | 26.925824 | -15.068488 | No | 12:25:52 | 2017/6/12 | -1 | -1 | 0.000000 | 33ºC | 28% | 5.000000 | 0 |
| 445-12:25:54_1.png | 40 | [398.000000 287.000000] | 17.088007 | -110.556045 | No | 12:25:52 | 2017/6/12 | -1 | -1 | 0.000000 | 33ºC | 28% | 5.000000 | 0 |
| 445-12:25:54_1.png | 41 | [413.000000 356.000000] | 39.115214 | -57.528808 | No | 12:25:52 | 2017/6/12 | -1 | -1 | 0.000000 | 33ºC | 28% | 5.000000 | 0 |
| 445-12:25:54_1.png | 43 | [437.000000 457.000000] | 55.362442 | 110.071526 | No | 12:25:52 | 2017/6/12 | -1 | -1 | 0.000000 | 33ºC | 28% | 5.000000 | 0 |
| 445-12:25:54_1.png | 55 | [517.000000 453.000000] | 18.439089 | -167.471192 | No | 12:25:52 | 2017/6/12 | -1 | -1 | 0.000000 | 33ºC | 28% | 5.000000 | 0 |

Figura 27: Datos extraídos de la predicción

# Capítulo 6
# Problemática y Limitaciones del Sistema

En este apartado trataremos los problemas más significativos que han complicado en gran medida la realización de nuestro proyecto y la capacidad de nuestro prototipo para realizar una mejor estimación.

## 6.1 Capacidad de computo y temperatura de la Raspberry Pi 3

En primer lugar, el proyecto en un primer momento se pensó para realizarse exclusivamente con la Raspberry sin la necesidad de utilizar otra máquina externa para llevar a cabo el trabajo del procesamiento, sin embargo rápidamente se observo que no iba a ser posible ya que únicamente el trabajo de las capturas y su fusión hacia uso de gran parte de la CPU, esto unido al poco almacenamiento y a los problemas de alta temperatura (>85ºC), hicieron necesario del uso de otros dispositivos. Dejando a la Raspberry y Arduino con los sensores que se encargasen de las capturas y la recogida de datos.

## 6.2 Captura en la secuencia de imágenes. Librería Raspicam.

Otro de los problemas más significativos se encontraron a la hora de generar la fusión de las imágenes, el primero de ellos, es que se hace uso de pocas capturas para generar la nueva imagen. El uso de pocas capturas (3) se debe a que la cámara que se utiliza necesita un tiempo de recuperación entre captura y captura, mientras se espera ese tiempo, la meteorología cambia y el estado del cielo es diferente ya que no estamos frente a paisajes estáticos, y si se realizara una fusión entre muchas imágenes además de afectar a la capacidad de computo, se conseguiría un efector de

movimiento no deseado.

Para la generación de la fusión se pretendía crear imágenes en HDR (Alto Rango Dinámico), que se consiguen principalmente realizando capturas con diferente tiempo de exposición, sin embargo el problema surge cuando intentamos cambiar esos tiempos, ya que al estar utilizando la librería raspicam para la captura, no permite asignar un tiempo en segundos o milisegundos sino que se trata de un rango entre 0-100. Como nos encontramos con la cámara muchas veces dirigida directamente hacia el Sol, desde que se asigne un tiempo de exposición en ese rango mayor a 1 o en el mejor de los casos 2, la imagen que obtendremos estará completamente quemada (la imagen se ve en blanco) y es por ello que se optó por cambiar la ISO a las imágenes y generar una fusión rápida con el algoritmo de Mertens.

## 6.3 Calidad de la cámara

Además de lo comentado anteriormente referente a el tiempo entre capturas, deberemos destacar que otro de los problemas era la baja calidad en la resolución de la cámara que se ha utilizado en el proyecto, concretamente de 5 megapíxeles, unido a los inconvenientes de la lente ojo de pez que pierde mucha información en los extremos de las imágenes y tiene mucha en las zonas centrales.

## 6.4 Tiempo atmosférico

Por último, el principal impedimento con el que nos hemos topado ha sido la meteorología, en concreto el movimiento y comportamiento de las nubes. La topografía de la zona analizada influye en la formación y movimiento de las nubes, tal que su forma nunca queda completamente definida y con una velocidad constante en el tiempo, mostrando por el contrario un comportamiento muy variable. Además, al comprobar los datos de velocidad del tiempo en los días analizados, en la estación meteorológica más cercana con datos accesibles, observamos datos que corresponden a una brisa muy moderada, de velocidad promedio en torno a los 1.8 m/s. En estos casos de baja velocidad, la dirección del viento no suele tener una dirección fija o estable. Lo que hace muy difícil establecer una predicción utilizando este tipo de sistemas.

# Capítulo 7
# Conclusiones y líneas futuras

En este trabajo de fin de grado, se ha diseñado un sistema para la previsión de imágenes meteorológicas mediante algoritmos de visión por computador y con la utilización de sistemas empotrados, concretamente con Raspberry Pi 3 y con Arduino.

Gracias a ello hemos sido capaces de realizar capturas del cielo mediante una cámara con un objetivo gran angular, procesarlas para realizar estimaciones del movimiento de las nubes y predecir cómo este movimiento afectaría a la cantidad de energía solar recogida mediante una célula fotoeléctrica. También se pensó que sería conveniente obtener otros datos del ambiente como podrían ser la temperatura y la humedad relativa.

Como se ha dicho en capítulos anteriores, la idea era utilizar únicamente sistemas como la Raspberry para la realización de todo el proyecto, sin embargo podríamos concluir que para el procesamiento de las imágenes, en la Raspberry se ha visto claramente superada su capacidad de procesamiento y existen claras limitaciones en relación a la temperatura, más tratándose de un sistema que podría verse afectado puesto que estaría en exteriores a altas temperaturas. Optando al final por utilizar un ordenador que realizara las tareas de procesado y liberando así ese trabajo en el sistema empotrado.

Para continuar con el trabajo, se podría, si se desea utilizar únicamente los sistemas empotrados, utilizar otros sistemas con mayor capacidad de computo como por ejemplo las placas upboard que tienen un procesador Intel Atom, aunque se debería estudiar el impacto térmico que tendría sobre el sistema. Además de considerar el uso de cámaras de mejor resolución para obtener mayor nivel de detalles en las fotografías.

En relación a los algoritmos, deberíamos, para conseguir predicciones más acertadas realizar un mejor estudio del comportamiento de las nubes basándonos en un histórico o consultando información a bases de datos externas que nos arrojen mas información en base a la dirección o velocidad del viento.

# Capítulo 8
# Summary and Conclusions

In this end-of-grade work, a system has been designed for the forecasting of meteorological images using computer vision algorithms and the use of embedded systems, specifically with Raspberry Pi 3 and Arduino.

Thanks to this, we have been able to capture sky images through a camera with super wide-angle lens and process them for estimating the movements of clouds and how this move would affect the quantity of solar energy collected by a photocell. It was also thought it would be convenient to obtain other data from the enviroment such as the temperature and relative humidity.

As was said in previous chapters, the idea was to use only Raspberry systems for the realization of the whole proyect, however we could concluded that for the processing of the images, the Raspberry has clearly exceeded it’s processing capacity and there are clear limitations in relation to temperature, more in the case of a system that could be affected by it, as would be found outdoors at high temperatures.

In order to improve this work, you could use, if you want to use only embedded systems, use other systems with greater capacity of computing such as the Upboard boards that have an Intel Atom processor, although the termal impact it would have on the system should be studied. In addition to considering the use of better resolution cameras to obtain grater detail in the photographs.

In relation to the algorithms, we should, in order to obtain better predictions, make a better study of the behavior of the clouds based on a historical or consulting information to external databases that give us more information based on the direction or speed of the wind.

# Capítulo 9
# Presupuesto

En este capitulo se expondrá un presupuesto estimado del coste total del proyecto. Diferenciaremos entre el coste en materiales o componentes y el coste del desarrollo del trabajo.

## 9.1 Presupuesto de materiales

| Material | Cantidad | Precio/Unidad | Total |
|---|---|---|---|
| Raspberry Pi 3 model B | 1 | 37,50€ | 37,50€ |
| Raspberry Camera | 1 | 25,95€ | 25,95€ |
| Arduino Uno | 1 | 21,26€ | 21,26€ |
| Sensor DHT22 | 1 | 9,90€ | 9,90€ |
| Célula Fotoeléctrica | 1 | 6,00€ | 6,00€ |
| PoE Emisor | 1 | 33,43€ | 33,43€ |
| PoE Receptor | 1 | 27,66€ | 27,66€ |
| Resistencias | 2 | 0,02€ | 0,04€ |
| Caja contenedora | 1 | 3,89€ | 3,89€ |
| **TOTAL** | | | **165,63€** |

**Tabla 9.1:** Resumen de materiales

## 9.2 Presupuesto de trabajo realizado

| Recurso | Horas | Precio/Hora | Total |
|---|---|---|---|
| Análisis y documentación. | 40 | 15,00 | 600 |
| Configuración del sistema | 10 | 20,00 | 200 |
| Programación | 150 | 20,00 | 3000 |
| Pruebas | 5 | 15,00 | 75 |
| Documentación | 15 | 20,00 | 300 |
| **TOTAL** | | | **4175€** |

**Tabla 9.2:** Resumen de recursos

## 9.3 Presupuesto total del proyecto

| Descripción | Precio |
|---|---|
| Materiales | 165,63 |
| Desarrollo | 4175,00 |
| **TOTAL** | **4340,63€** |

**Tabla 9.3:** Presupuesto total del proyecto

# Apéndice A. Código desarrollado

## 9.4 Algoritmo Arduino

```
/
**********************************************************************
******
*
* Fichero temperatura.ino
*
**********************************************************************
*******
*
* AUTORES Alejandro Delgado Martel
*
*
* DESCRIPCION
*    Arduino estará continuamente escuchando el puerto serial a 115200
baudios y si recibe algún comando por el serial responderá enviando la
información requerida, extraida bien por la librería para el sensor
DHT22 o por el pin analógico si se trata de la célula fotoeléctrica
*
*
**********************************************************************
*******/


#include <DHT_U.h>
#include <DHT.h>

#include <Adafruit_Sensor.h>


#define DATA_PIN 2
#define DHT_TYPE 22
#define SOLAR_PIN 1
```

```
DHT dht(DATA_PIN, DHT_TYPE);

void setup(){
  Serial.begin(115200);
  dht.begin();

}

void loop(){


  if(Serial.available()>0){

    char option = Serial.read();
    delay(100);

   switch(option){
    case 't':{
       //SE ENVIA LA TEMPERATURA RECOGIDA POR EL SENSOR DHT
       double temperatura = dht.readTemperature();
       delay(100);
       String t = String((int)temperatura);
       Serial.println(t);
    }
       break;
    case 'h':{
       //SE ENVIA LA HUMEDAD RECOGIDA POR EL SENSOR DHT
       double humedad = dht.readHumidity();
       delay(100);
       String h = String((int)humedad);
       Serial.println(h);
    }
    case 's':{
       float value = analogRead(SOLAR_PIN);
       float voltaje = (value * 4) / 819;
       char buffer[6];
       String v =
dtostrf(voltaje,10,2,buffer);//String((analogRead(SOLAR_PIN)*4)/819,
DEC);
       Serial.println(v);

    }
```

```
        break;
     default: break;
    }


  }
}
```

## 9.5 Código del proyecto

El código del proyecto se encuentra disponible en la plataforma GitHub mediante la siguiente dirección: https://github.com/AleDelgado94/TFG-Vision-RaspberryPi3/tree/master/tests

# Apéndice B

## 9.6 Raspberry Pi 3 model B. Características.

Raspberry Pi

### Raspberry Pi 3 Model B

**Specifications**

| | |
|---|---|
| **Processor** | Broadcom BCM2387 chipset.<br>1.2GHz Quad-Core ARM Cortex-A53<br>802.11 b/g/n Wireless LAN and Bluetooth 4.1 (Bluetooth Classic and LE) |
| **GPU** | Dual Core VideoCore IV® Multimedia Co-Processor. Provides Open GL ES 2.0, hardware-accelerated OpenVG, and 1080p30 H.264 high-profile decode.<br>Capable of 1Gpixel/s, 1.5Gtexel/s or 24GFLOPs with texture filtering and DMA infrastructure |
| **Memory** | 1GB LPDDR2 |
| **Operating System** | Boots from Micro SD card, running a version of the Linux operating system or Windows 10 IoT |
| **Dimensions** | 85 x 56 x 17mm |
| **Power** | Micro USB socket 5V1, 2.5A |

**Connectors:**

| | |
|---|---|
| **Ethernet** | 10/100 BaseT Ethernet socket |
| **Video Output** | HDMI (rev 1.3 & 1.4<br>Composite RCA (PAL and NTSC) |
| **Audio Output** | Audio Output 3.5mm jack, HDMI<br>USB 4 x USB 2.0 Connector |
| **GPIO Connector** | 40-pin 2.54 mm (100 mil) expansion header: 2x20 strip<br>Providing 27 GPIO pins as well as +3.3 V, +5 V and GND supply lines |
| **Camera Connector** | 15-pin MIPI Camera Serial Interface (CSI-2) |
| **Display Connector** | Display Serial Interface (DSI) 15 way flat flex cable connector with two data lanes and a clock lane |
| **Memory Card Slot** | Push/pull Micro SDIO |

**Key Benefits**

- Low cost
- Consistent board format
- 10x faster processing
- Added connectivity

**Key Applications**

- Low cost PC/tablet/laptop
- IoT applications
- Media centre
- Robotics
- Industrial/Home automation
- Server/cloud server
- Print server
- Security monitoring
- Web camera
- Gaming
- Wireless access point
- Environmental sensing/monitoring (e.g. weather station)

## 9.7 Arduino Uno

| | |
|---|---|
| Microcontroller | ATmega328P |
| Operating Voltage | 5V |
| Input Voltage (recommended) | 7-12V |
| Input Voltage (limit) | 6-20V |
| Digital I/O Pins | 14 (of which 6 provide PWM output) |
| PWM Digital I/O Pins | 6 |
| Analog Input Pins | 6 |
| DC Current per I/O Pin | 20 mA |
| DC Current for 3.3V Pin | 50 mA |
| Flash Memory | 32 KB (ATmega328P) of which 0.5 KB used by bootloader |
| SRAM | 2 KB (ATmega328P) |
| EEPROM | 1 KB (ATmega328P) |
| Clock Speed | 16 MHz |
| LED_BUILTIN | 13 |
| Length | 68.6 mm |
| Width | 53.4 mm |
| Weight | 25 g |

## 9.8 DHT22 Sensor.

| | |
|---|---|
| Model | DHT22 |
| Power supply | 3.3-6V DC |
| Output signal | digital signal via single-bus |
| Sensing element | Polymer capacitor |
| Operating range | humidity 0-100%RH; temperature -40~80Celsius |
| Accuracy | humidity +-2%RH(Max +-5%RH); temperature <+-0.5Celsius |
| Resolution or sensitivity | humidity 0.1%RH; temperature 0.1Celsius |
| Repeatability | humidity +-1%RH; temperature +-0.2Celsius |
| Humidity hysteresis | +-0.3%RH |
| Long-term Stability | +-0.5%RH/year |
| Sensing period | Average: 2s |
| Interchangeability | fully interchangeable |
| Dimensions | small size 14*18*5.5mm; big size 22*28*5mm |

# Bibliografía


[1] Raspberry Pi, https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

[2] Arduino Uno, https://www.arduino.cc/

[3] OpenCV 3.2.0, http://www.opencv.org/

[4] OpenCV_contrib 3.2, https://github.com/opencv/opencv_contrib/releases

[5] Raspicam, https://www.uco.es/investiga/grupos/ava/node/40

[6] Raspbian Jessie, https://www.raspberrypi.org/downloads/raspbian/

[7] Sistema Operativo Ubuntu, https://www.ubuntu.com/download/desktop

[8] Configuración NFS, https://www.htpcguides.com/configure-nfs-server-and-nfs-client-raspberry-pi/

[9] SqLite3, https://www.sqlite.org/

[10] Librerías boost, http://www.boost.org/

[11] Cmake, https://cmake.org/

[12] FishEye Camera, http://www.ebay.com/itm/ArduCam-OV5647-Camera-Board-w-Fisheye-Lens-M12x0-5-Mount-for-Raspberry-Pi-3-/122498308742

[13] DHT22 Sensor, https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf

[14] Librerías Adafruit, https://github.com/adafruit

[15] Documentación OpenCV, http://docs.opencv.org/

[16] Visión por computador, https://es.wikipedia.org/wiki/Visi%C3%B3n_artificial

[17] Algoritmo detector esquinas, https://es.wikipedia.org/wiki/Detector_de_esquinas