Universidad Zaragoza
1542

Trabajo Fin de Grado

Sistema Automático de reconocimiento de monedas sobre una tarjeta Raspberry Pi

Autor/es

Iván Martínez Lahuerta

Director/es

Ana María López Torres

Alfonso Blesa Gascón

Escuela Universitaria Politécnica de Teruel

2018

“Sistema Automático de reconocimiento de monedas sobre una tarjeta Raspberry Pi”

Resumen

Trabajo de Fin de Grado cuyo objetivo es reconocer una serie de monedas mediante una tarjeta Raspberry Pi que actúa como servidor, ejecutando un programa de reconocimiento de patrones basado en el uso de la librería OpenCV y un Arduino, encargado de ejecutar distintos actuadores por los cuales podremos ver los resultados obtenidos.

El sistema toma una foto mediante una PiCamera a un conjunto de monedas de euro, las identifica y clasifica, mostrando el resultado (las monedas que hay, el valor total de ellas mismas y el número de monedas) en una pantalla LCD cuyo contenido podremos recorrer con la ayuda de un Joystick.

Palabras Clave: Reconocimiento, Clasificación, Monedas, PiCamera, Raspberry Pi, Arduino.

"Automatic system for recognizing coins on a Raspberry Pi card"

Abstract

Final Degree Project whose objective is to recognize a series of coins by means of a Raspberry Pi card that acts as a server executing a pattern recognition program based on the use of the OpenCV library and an Arduino Mega.This card is responsible for executing different actuators used to visualize the results obtained.
The system takes a picture by means of a PiCamera to a set of euro coins, identifies and classifies them, showing the result (which coins are present, their total value and number) on an LCD screen whose content we can cover with the help of a Joystick.

Keywords: Recognition, Classification, Coins, PiCamera, Raspberry Pi, Arduino.

Índice:

Lista de Ilustraciones:

Lista de tablas:

1. Introducción:

Tradicionalmente el hombre ha podido utilizar únicamente sus propios ojos y su cerebro para ver mundo que le rodea y procesar toda la información que a través de ellos recibe. [1]

Sin embargo, las nuevas tecnologías nos han permitido el desarrollo de la visión artificial, o visión por computador, la cual es una disciplina científica que estudia cómo reconstruir, interpretar y entender una escena en tres dimensiones a partir de imágenes en dos dimensiones, siendo el objetivo último de esto crear un modelo que reproduzca las capacidades humanas usando un software y hardware determinado. Esta disciplina está basada en la extracción automática, el análisis y la comprensión de información útil que pueda ser proporcionada por una imagen o una secuencia de ellas. Aunque no hemos de olvidar que la visión por computador es muy inferior a la visión humana.

Las aplicaciones de la visión por computador son muy numerosas y comprenden ámbitos tan diversos como pueden ser la agricultura, controles de calidad en industria, realidad aumentada, análisis de imágenes médicas, seguridad y vigilancia, transporte, robótica, etc.

La visión artificial es por tanto, un medio que permite complementar la visión del ojo humano dotándola de características que se escapan de su naturaleza. La visión humana consiste en dos procesos, la captura de imágenes y su posterior interpretación. Hoy en día la captura de imágenes está muy evolucionada, pues existen numerosos tipos de cámaras que pueden obtener imágenes basándose en diferentes espectros de luz, con mayor o menor resolución, el problema aparece con la interpretación de las imágenes captadas.

Ilustración 1[2]: Imagen de Visión por Computación

Para una persona la interpretación de una imagen es algo natural ya que el cerebro posee unas capacidades de procesado de imágenes excelentes, sin embargo, para un sistema artificial es muy difícil desarrollar todas estas capacidades, por ello es conveniente desarrollar sistemas que realicen funcionalidades independientes de interpretación de imágenes, como puede ser la visión tridimensional, reconocimiento de objetos o la identificación y seguimiento de objetos.

A continuación, en la Ilustración 2 podremos ver de forma general todas las etapas que debe seguir un sistema de visión artificial, sea cual sea el propósito para el que está diseñado. En este trabajo se seguirá el mismo esquema aunque personalizado para cada una de las etapas con el fin de conseguir el objetivo buscado.

Para llevar a cabo un proceso de visión artificial se deben cumplir unos pasos fundamentales comunes a cualquier desarrollo.

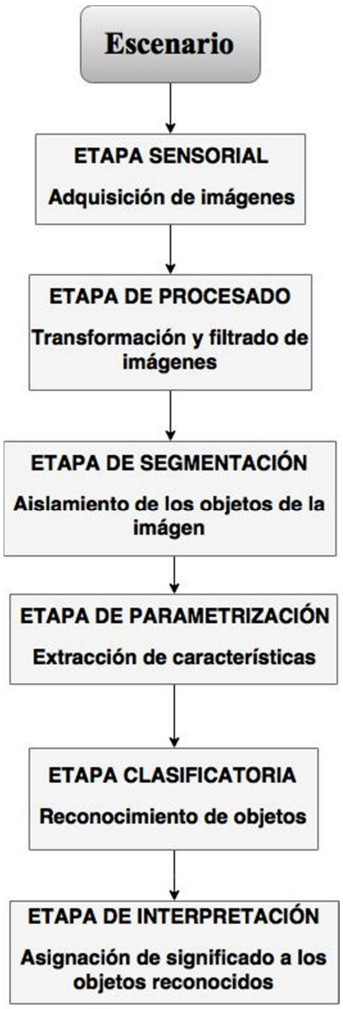


Ilustración 2: Diagrama de flujo general de un proceso de Visión Artificial

- Etapa Sensorial: Consiste en tomar la foto y transformar la imagen real en una digital.
- Etapa de Procesado: Algoritmos necesarios para tratar la imagen y que su tratamiento posterior se vea así facilitado, como podría ser la eliminación de ruido o el incremento del contraste.
- Etapa de Segmentación: Seleccionar los diferentes elementos de la imagen en función de niveles de intensidad o mediante la detención de contornos.
- Etapa de Parametrización: Obtención de diferentes medidas de los objetos extraídos.
- Etapa Clasificatoria: Permite incluir los objetos con las mismas características o rasgos dentro de un mismo grupo, con el fin de agrupar los que sean iguales y separar los que sean diferentes.
- Etapa de Interpretación: Dotar de significado la imagen estudiada. Algunos ejemplos serían el seguimiento de puntos de interés o distinguir entre varias clases de objetos como sería nuestro caso.

1.1 Objetivos

1.1.1 Objetivo Principal

El objetivo principal de este Trabajo de Fin de Grado es la creación de un sistema automático capaz de reconocer los diferentes tipos de monedas existentes de euro, diferenciándolas y haciendo la suma total del valor que representan. La tarjeta Raspberry Pi tomará fotos mediante la PiCamera, también actuará como servidor donde se ejecuta el programa y nos proporciona la extracción de información necesaria para poder realizar el objetivo principal. El Arduino Mega hace que los diferentes actuadores nos muestren la información requerida.

Se utiliza la librería de funciones OpenCv para la interpretación de las principales acciones de procesado de imagen y extracción de la información.

1.1.2 Objetivos Secundarios

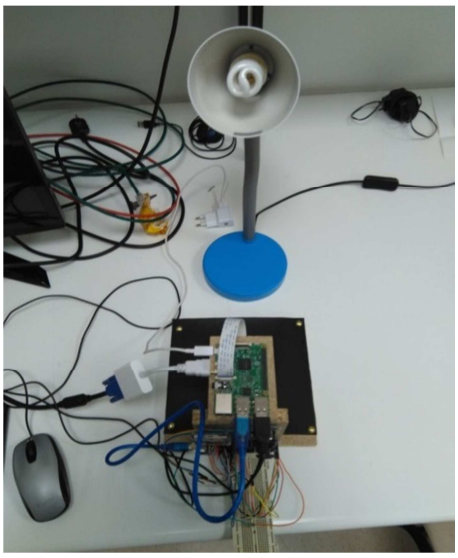
1. Construcción de la maqueta donde se va a llevar a cabo el proyecto.
2. Búsqueda de las mejores condiciones de iluminación para que el reconocimiento de monedas sea óptimo.
3. Puesta en marcha de la tarjeta Raspberry Pi y de la PiCamera.
4. Instalación de las librerías necesarias para el correcto funcionamiento de la Raspberry Pi y de la PiCamera.
5. Instalación de la librería OpenCV en la tarjeta Raspberry Pi.
6. Puesta en marcha del Arduino Mega.
7. Puesta en marcha de la pantalla Nokia 5110, Joystick y Display 7 segmentos.
8. Programación en código Python para el reconocimiento de las monedas.
9. Programación en código C para el uso de los actuadores.
10. Programación en código C la comunicación entre Raspberry Pi y Arduino Mega.
11. Evaluación del funcionamiento de la aplicación.

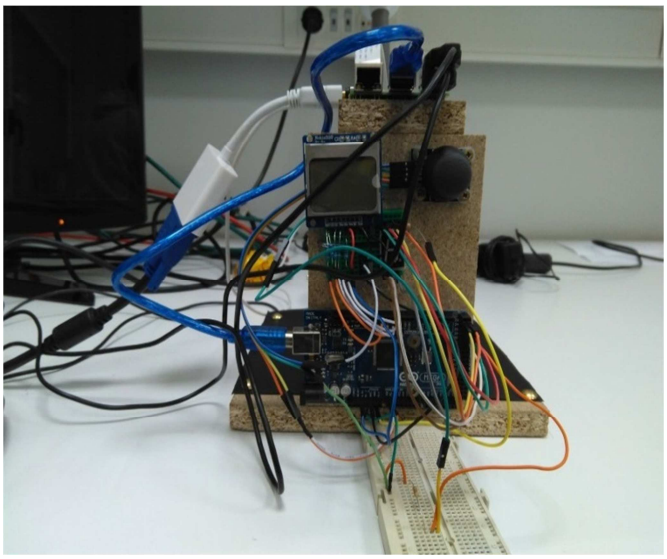
2. Diseño Hardware:

Para llevar a cabo el desarrollo del Trabajo de Fin de Grado se ha construido una maqueta de aglomerado mediante 3 piezas (20 x20 cm, 10 x 15 cm, 10 x 7 cm). Se ha usado este material debido a su bajo coste y fácil manejo. En la pieza superior por la parte de arriba se colocará la Raspberry Pi y por la parte de debajo de la misma, la PiCamera, ya que estará así orientada hacia la base, que es donde se dejarán las monedas.

En la parte central se colocará el Arduino junto a todos los actuadores, y en la base se pondrá una cartulina negra donde se depositarán las monedas. Se ha elegido una cartulina negra debido a que si se hacía directamente sobre el aglomerado se reconocían más objetos de los debidos, ya que la función de reconocimiento usada nos señalaba como objetos las astillas del aglomerado, y si se hacía sobre un folio blanco aparecían sombras interfiriendo de manera negativa en el reconocimiento de monedas.

La maqueta será iluminada mediante un flexo que facilitará la iluminación justa para evitar el brillo de las monedas y poder realizar un correcto reconocimiento.





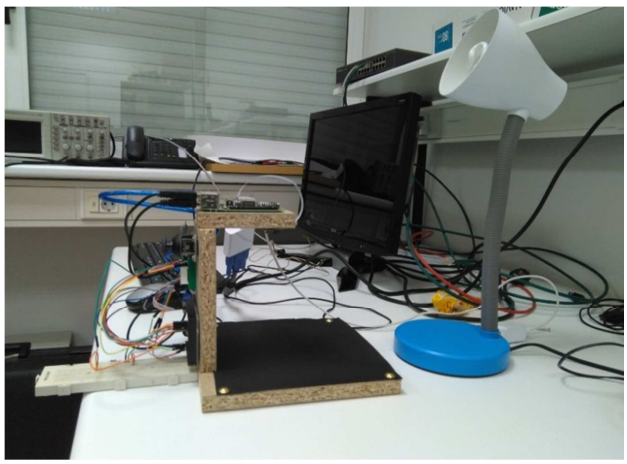


Ilustración 3: Distintos puntos de vista del proyecto

2.1 Diagrama de bloques del sistema completo:

A continuación, como podemos ver en la Ilustración 4, se muestra un diagrama de bloques en el que aparecen todos los elementos que forman el proyecto y como están comunicados. En los siguientes apartados se irán explicando cada uno de ellos de forma más detallada.

La Rasberry Pi, que es donde se ejecuta el programa principal, se conecta por banda CSI con la PiCamera, encargada de tomar las fotos. También se conecta mediante cable USB con Arduino Mega, el teclado del ordenador y el ratón. A la pantalla del ordenador se conecta a través de cable HDMI.

A Arduino están conectados tres actuadores, el Joystick mediante 2 entradas Analógicas y 1 Digital, la pantalla Nokia que se conecta mediante I2C y el Display conectado a través de 7 salidas Digitales. Cada uno de los elementos será explicado a continuación:

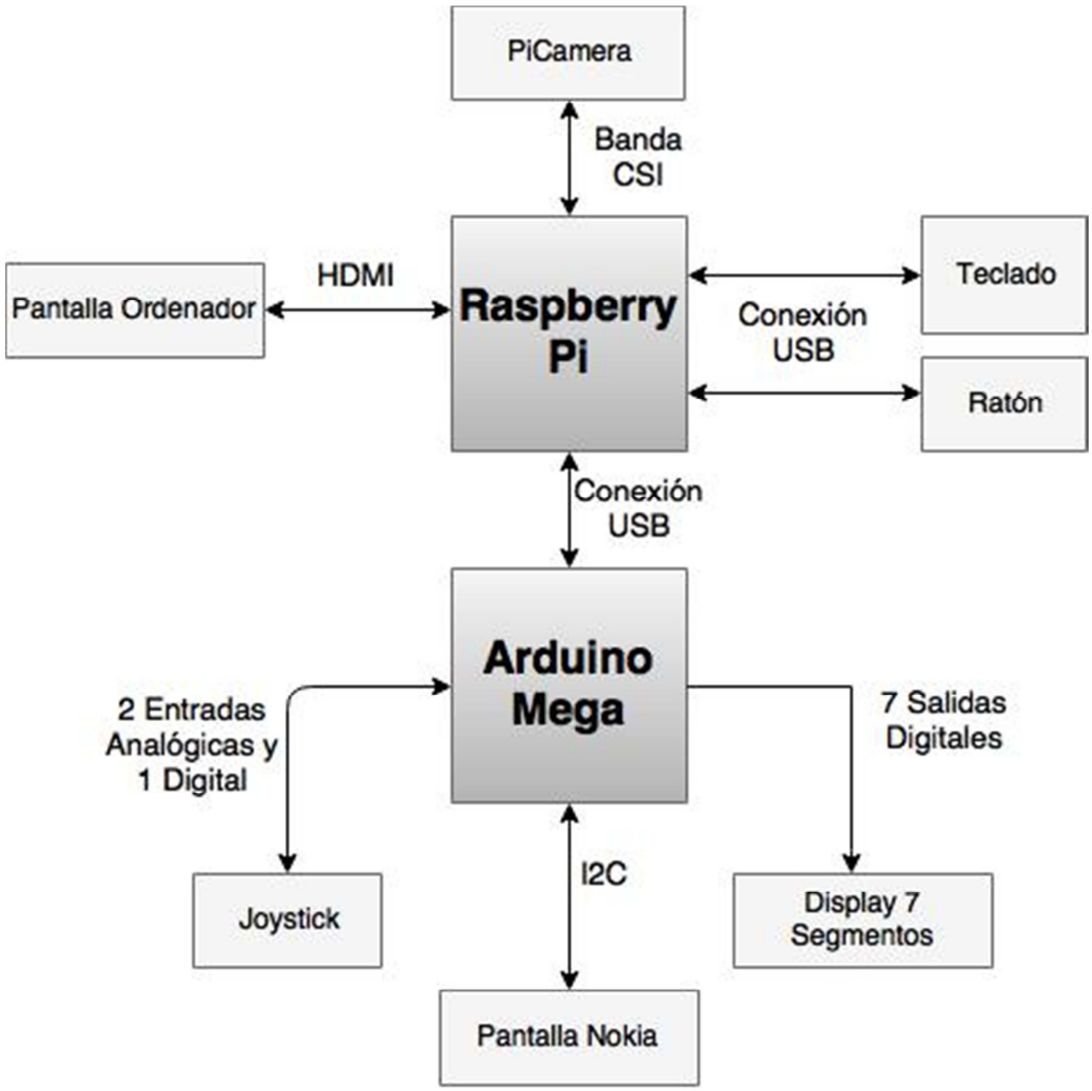


Ilustración 4: Diagrama de bloques del sistema completo

2.2 Raspberry Pi:

La Raspberry Pi[3] es una placa computadora o SBC (del inglés: Single Board Computer) de bajo coste desarrollada en Reino Unido con el objetivo de facilitar el acceso a la tecnología de la computación a un público más amplio y potenciar la enseñanza de esta ciencia en las escuelas.

Como vemos en la Ilustración 5, para el desarrollo de este trabajo se dispone de la Raspberry Pi 3, modelo B. En la cual, la PiCamera ha sido conectada al conector CSI. Además se han utilizados 3 puertos USB (ratón, teclado y comunicación Raspy-Arduino) y una salida HDMI para poder ver el entorno gráfico a través de la pantalla del ordenador y poder ejecutar el programa de reconocimiento de las monedas. Se instaló el sistema operativo Raspbian que es de software libre y se describe más adelante.

Podríamos haber utilizado el programa ”Putty” para conectarnos de forma remota a nuestra Raspberry Pi y haber ejecutado el programa desde allí, tan solo habríamos necesitado instalar en un portátil externo el programa “Putty”, la aplicación informática “Samba” (programa para compartir archivos) que se instalaría en nuestra Raspberry Pi y conexión Ethernet. Finalmente se decidió no implementarlo porque para una aplicación real no creemos que fuera necesario ejecutar nuestro programa de reconocimiento desde una conexión remota, sino desde una conexión local.

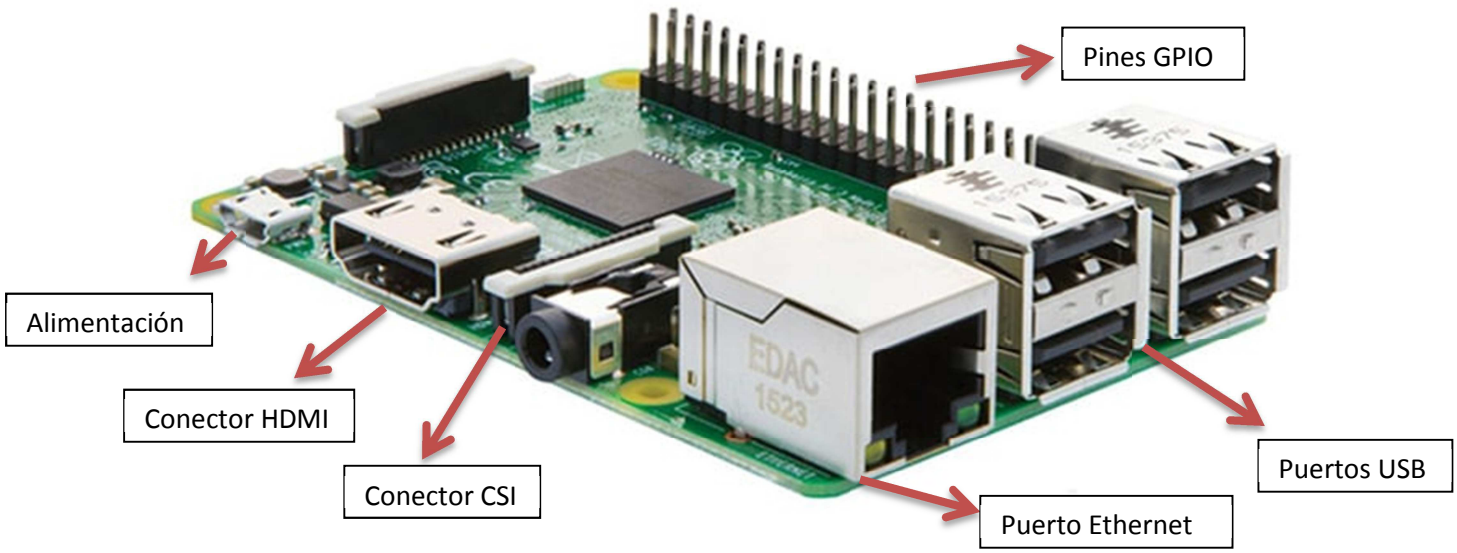


Ilustración 5: Partes de la Raspberry Pi 3

2.3 PiCamera:

La cámara oficial de Raspberry Pi, Figura 6, será la plataforma utilizada para la adquisición de imágenes. Se trata de una cámara V2[4] de 8 MegaPíxeles y con un sensor de imagen Sony IMX219, que se conecta directamente con el conector CSI de la Raspberry Pi.

El conector CSI (del inglés: Camera Serial Interface, o Interfaz de Cámara Serie) es un estándar definido por la MIPI (Mobile Industry Processor Interface)que permite la conexión serie directa de una cámara con el procesador, proporcionando velocidades de transmisión mayores al evitar interfaces intermedias. Tiene las siguientes dimensiones: 25 x 23 x 9 mm

La ventaja de utilizar la cámara oficial es la compatibilidad y la facilidad que presenta respecto a otras cámaras USB. La desventaja más llamativa es el precio, ya que es más elevado.

Ilustración 6: PiCamera

2.3.1 Configuración:

Tras varios ensayos y para mejorar la calidad de las imágenes adquiridas por la PiCamera se han configurado los siguientes parámetros recogidos en la Tabla 1:

- Resolución: cantidad de pixeles efectivos que aparecen en la fotografía.
- Contraste[5]: diferencia del tono de color con intensidad entre un punto de una imagen. Se expresa como un número entero de -100 a 100.
- Saturación[6]: es la intensidad de un matiz específico y se basa en la pureza del color; un color muy saturado tiene un color vivo e intenso, mientras que un color menos saturado parece más descolorido y gris. Se expresa como un número entero de -100 a 100.
- Brillo[6]: capacidad de un color para reflejar la luz blanca que incide en él. Alude a la claridad u oscuridad de un tono.
- Velocidad de fotogramas: indica la tasa de fotogramas por segundo que puede capturar la cámara.

Parámetros	Valores
Resolución	(1000,720)px
Contraste	50%
Saturación	0%
Brillo	75%
Velocidad de fotograma	50 fps

Tabla 1: Configuración de los parámetros de nuestra PiCamera

2.4 Arduino:

Arduino[7] es un micro controlador que será utilizado principalmente para el funcionamiento de los actuadores (pantalla Nokia, Joystick y Display 7 segmentos). Se ha elegido está plataforma para este fin porque es un dispositivo de fácil configuración y las características que posee son más que suficientes para la tarea que se va a desarrollar. Se dispone de Arduino Mega 2560 por todos los pines que dispone y la facilidad a la hora de programar el correcto funcionamiento de los distintos actuadores.

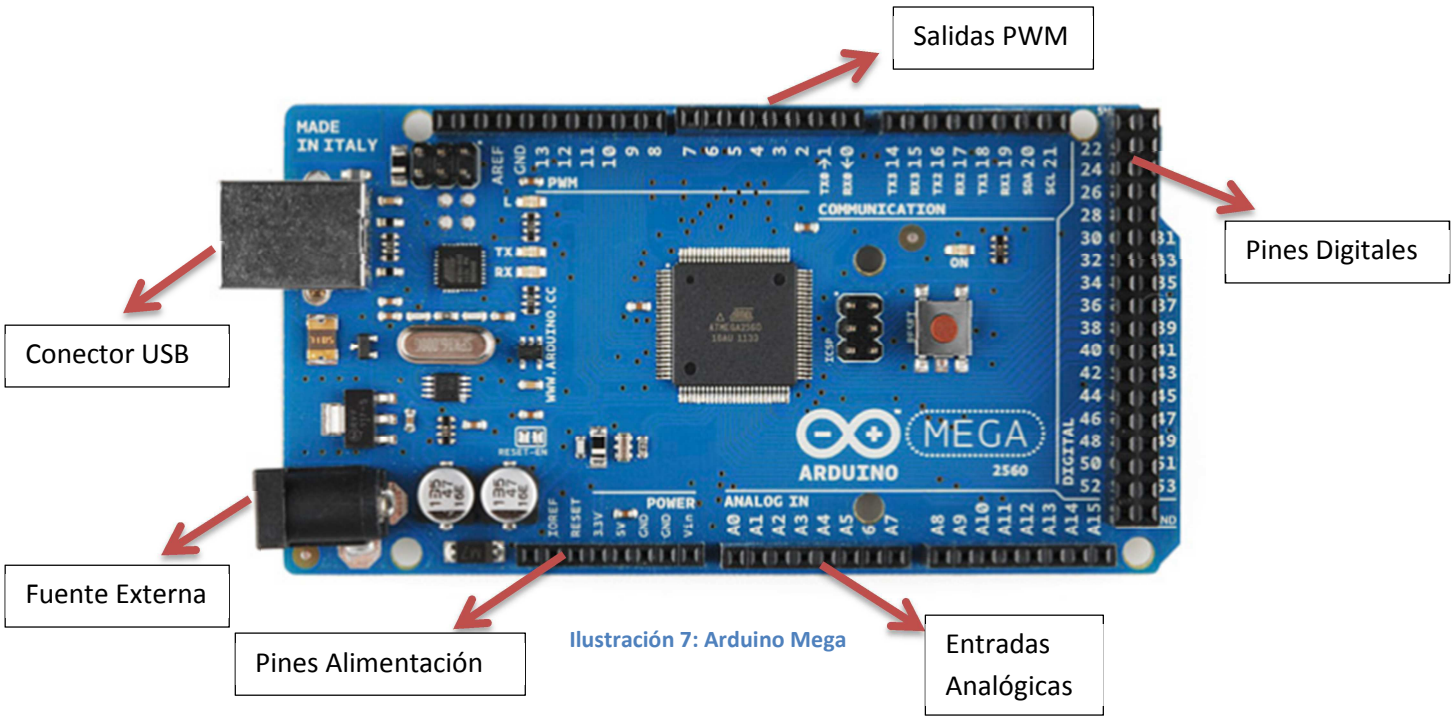


Ilustración 7: Arduino Mega

Arduino Mega como vemos en la Ilustración 7, tiene 16 entradas analógicas de las cuales para este Trabajo de Fin de Grado serán usados dos (A0 y A1) para el control del Joystick. También tiene 54 pines digitales siendo utilizados los pines del 3 al 7 para el control de la pantalla Nokia, del pin 22 al 28 para el manejo del Display 7 segmentos y el pin 29 para el selector del Joystick.

Se podrían haber conectado todos los actuadores a los diversos GPIOS de la Raspberry Pi en vez de a los pines de Arduino y mediante la librería WiringPi haber accedido y controlado todos los GPIOS y a su vez los diferentes actuadores. Esta situación nos habría hecho prescindir de la tarjeta Arduino, pero para este Trabajo de Fin de Grado se deseaba incluir las dos tarjetas para probar la comunicación entre ambas placas y la posibilidad de que trabajaran juntas, por eso las dos fueron incluidas.

2.5 Pantalla Nokia:

El LCD Gráfico de Nokia 5110[8], es una pequeña pantalla gráfica de 1.5", montada sobre un PCB de 45mm x 45mm, con una resolución de 84 x 48 píxeles. Se Puede controlar cada pixel individualmente por lo que es posible realizar gráficos e incluso mostrar imágenes. En este proyecto ha sido usada para representar los datos obtenidos y se eligió debido a su bajo coste. En la siguiente Tabla 2 vemos los pines de cada componente, cables y resistencias utilizadas para su montaje.

Nokia 5110	Color Cables	Resistencias	Arduino (Pines)
RST	Naranja	Resistencia 10 kΩ	7
CE	Naranja	Resistencia 10 kΩ	6
DC	Blanco	Resistencia 10 kΩ	5
Din	Blanco	Resistencia 10 kΩ	4
Clk	Blanco	Resistencia 10 kΩ	3
Vcc	Rojo	-	3.3V
BL	Negro	-	3.3V
GND	Negro	-	GND

Tabla 2: Conexiones de la pantalla Nokia

Imagen del montaje realizado y sus conexiones:

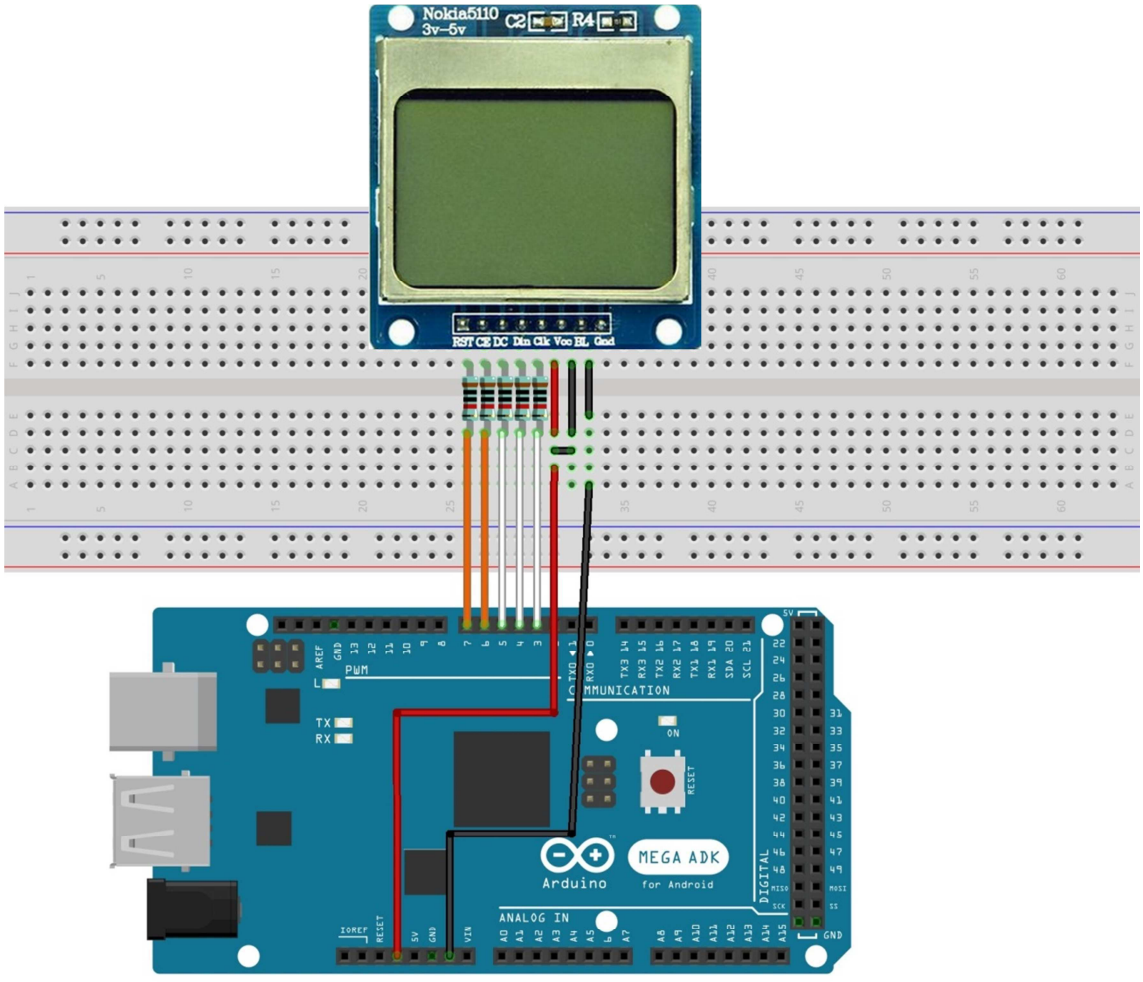


Ilustración 8: Esquema Protoboard del montaje de la pantalla Nokia

2.6 Joystick:

Un joystick[9] suele estar formado por dos potenciómetros a 90º que transforman el movimiento en X e Y del mando en una señal eléctrica proporcional a su posición y que además suele incluir un botón selector. Así pues, suelen tener 5 pines: X, Y, botón selector y 5V más GND como se puede ver en la Ilustración 9.

En nuestro prototipo el joystick será utilizado para poder inicializar o terminar el programa, movernos por los dos menús, si giramos hacía la izquierda nos sacará la lista de monedas identificadas y si lo movemos hacia la derecha nos mostrará la suma total. En la Tabla 3 podemos ver las conexiones del Joystick.

Ilustración 9: Joystick

Joystick	Cables	Resistencia	Arduino (Pines)
GND	Negro	-	GND
5V	Rojo	-	5V
VRx	Amarillo	-	A0
VRy	Naranja	-	A1
Sw	Verde	330 Ω	29

Tabla 3: Conexiones del Joystick

En la Ilustración 10 vemos el montaje realizado y sus conexiones:

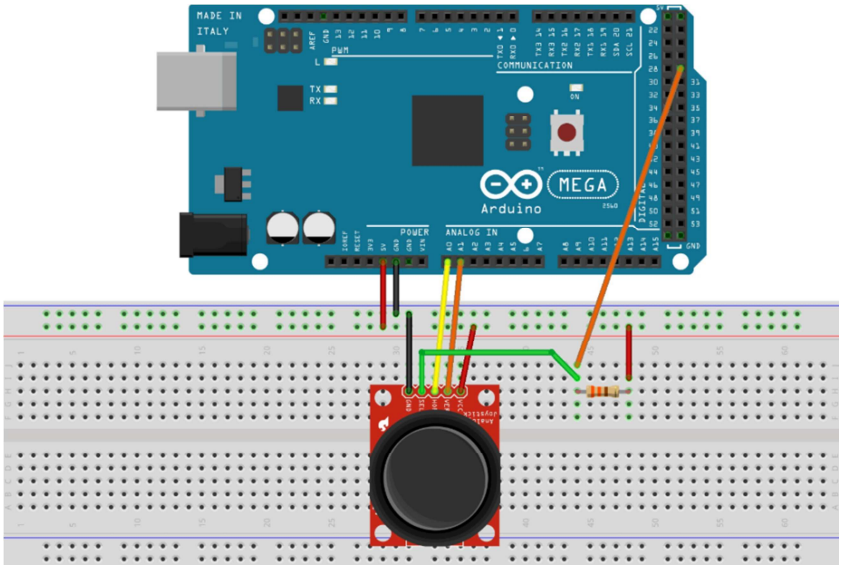


Ilustración 10: Esquema Protoboard del montaje del Joystick

2.7 Display 7 segmentos (1 dígito):

Un Display[10] de segmentos (o visualizador) es un componente electrónico que se utiliza para representar números. Como su propio nombre indica y, como se puede observar en la imagen siguiente, el display está compuesto por 7 segmentos, los cuales se encenderán y/o apagarán en función del número a representar. Ha sido usado para representar el número total de monedas que hay. En la siguiente Ilustración vemos los pines del Display 7 segmentos.

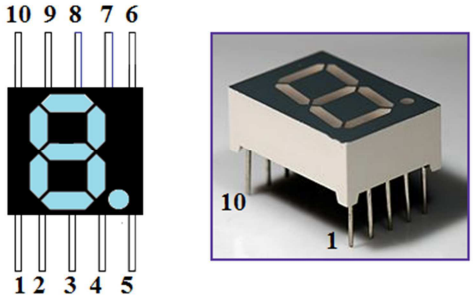


Ilustración 11: Display 7 segmentos físico y posicionamiento de sus pines

En la siguiente Tabla 4 se muestran los pines del Display utilizado, el color de los cables y a que pines se han conectado a Arduino. Y a continuación en la Ilustración 13 podemos ver las conexiones realizadas sobre un esquema en una Protoboard.

Display 7 Segmentos (Pines y Cables)	Arduino (Pines)
1- Cable Marrón	28
2- Cable Blanco	26
-	-
4- Cable Gris	24
-	-
6- Cable Naranja	23
7- Cable Amarillo	22
8- Resistencia 330Ω Cable Rosa	GND
9- Cable Rojo	25
10- Cable Verde	27

Tabla 4: Conexiones del Display

Imagen del montaje realizado y sus conexiones del Display en la Ilustración 12 :

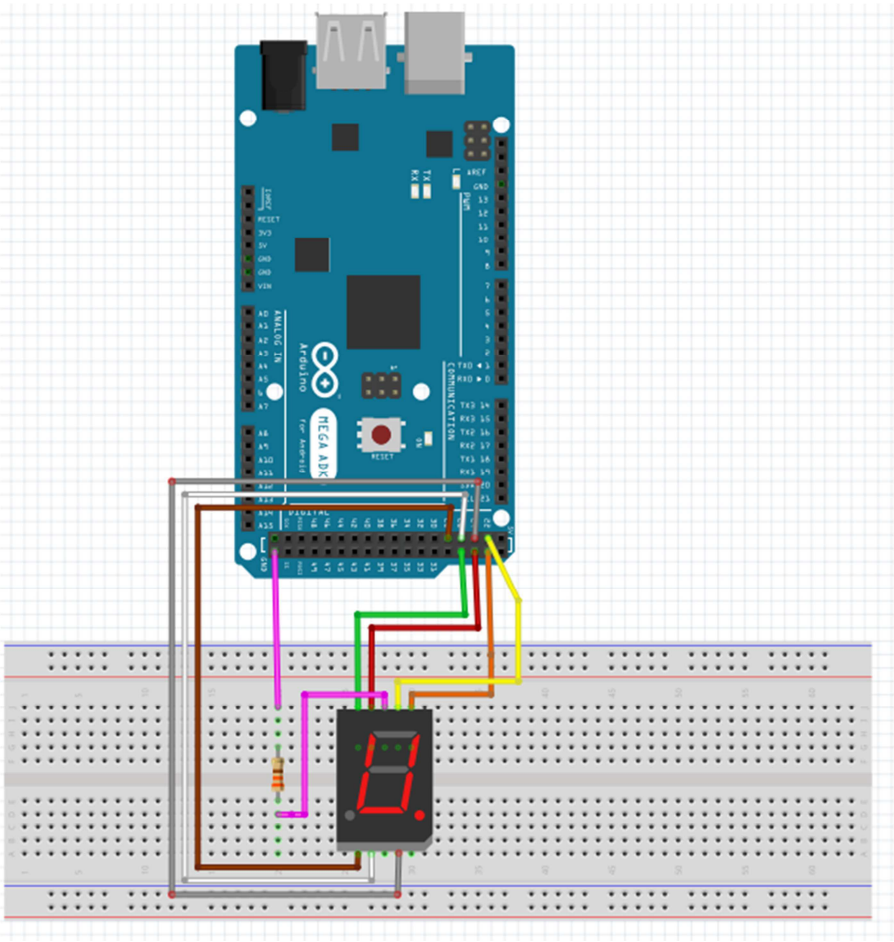


Ilustración 12: Esquema Protoboard del montaje del Display

En la Ilustración 13 podemos ver lo que sería el montaje de todas las conexiones eléctricas en un esquema Protoboard de nuestros tres actuadores (Display 7 segmentos, Joystick y Pantalla)

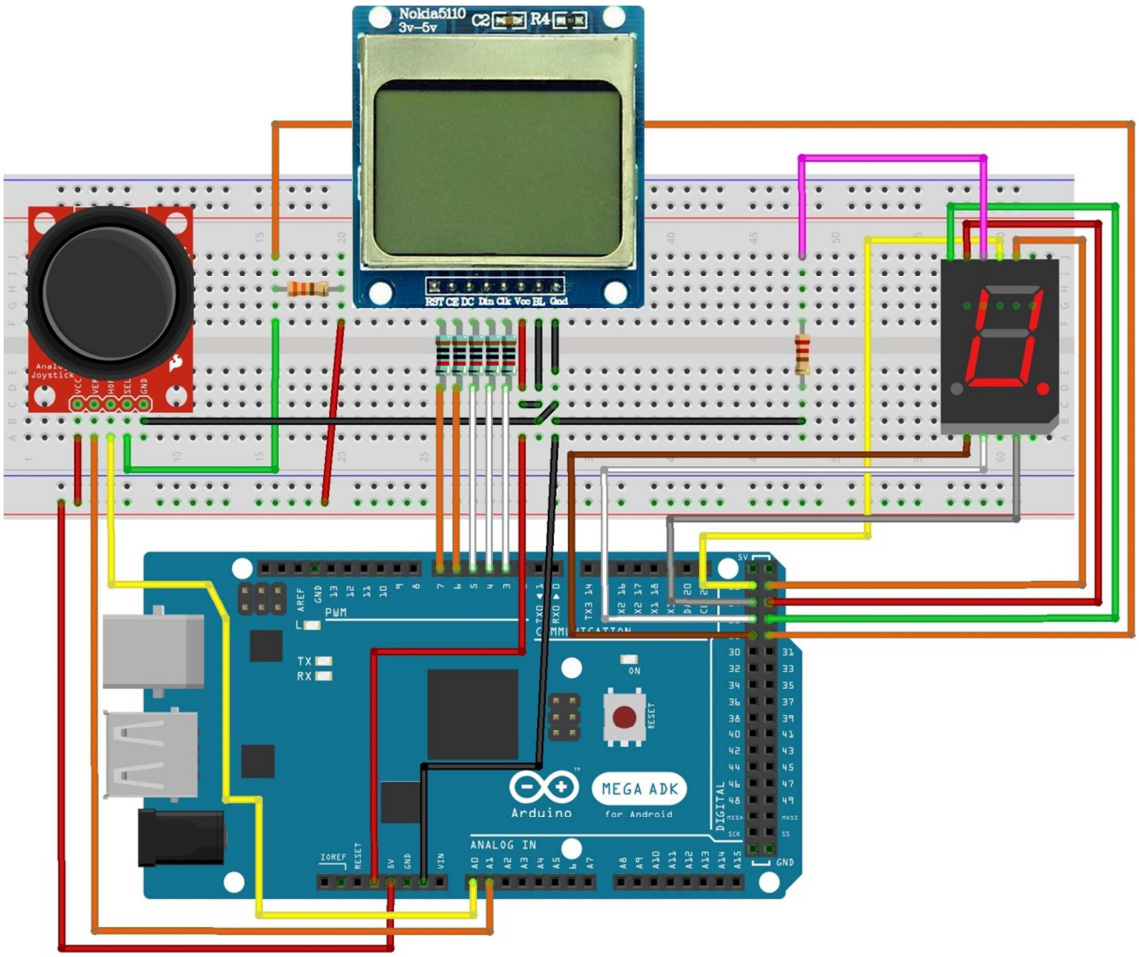


Ilustración 13: Conexiones de todos los actuadores

3. Diseño Software:

El programa principal que se ejecuta en la tarjeta Raspberry Pi se ha llevado a cabo en el lenguaje de programación Python y se encarga de todo el proceso de Visión por Computador, es el responsable del reconocimiento y clasificación de las monedas. Para la tarjeta Arduino el código se hizo en lenguaje C++ y lleva la comunicación entre la tarjeta Raspberry Pi con Arduino y el control de los diversos actuadores que nos mostraran la información requerida.

El programa hecho en C++ se pasa a la tarjeta Arduino una sola vez, desde el ordenador a través del cable USB y él está constantemente ejecutándose, siempre que reciba alimentación a través del cable USB de la Raspberry Pi. El programa de la Raspberry Pi lo hemos de ejecutar nosotros de forma manual desde el ordenador, una vez ejecutado, mediante el selector del Joystick decidiremos si comenzamos el programa (tomar foto) o salimos de él como se muestra en las imágenes 14 y 15.

Esto sería lo que nuestra pantalla Nokia nos mostraría al ejecutar nuestro programa de Python. El usuario, a través del Joystick, elige si comenzamos el reconocimiento y la clasificación de las monedas.

Ilustración 14: Pantalla Nokia con marca en el Inicio

O si decide salir del programa de Python.

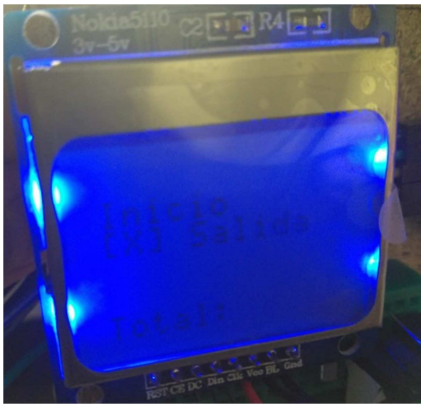


Ilustración 15: Pantalla Nokia con marca en Salida

3.1 Lenguaje Python:

Python es un lenguaje de alto nivel y fácil de aprender, ya que posee mucha legibilidad de su código. Su naturaleza interpretada hace de éste un lenguaje ideal para scripting y desarrollo rápido de aplicaciones en diversas áreas y sobre la mayoría de las plataformas. También puede ahorrar mucho tiempo durante el desarrollo ya que no es necesario compilar ni enlazar. Por contra, el rendimiento de la velocidad de procesamiento es inferior a otros lenguajes, aunque para este trabajo no es importante este inconveniente.

Siguiendo las recomendaciones de los repositorios oficiales de Raspberry Pi respecto a la PiCamera y dado que OpenCv es compatible con Python, se ha decidido hacer el programa principal en Python con versión 2.7.6. Este código será ejecutado en la Raspberry Pi desde la terminal del ordenador y de forma manual.

3.2 Lenguaje C++:

El lenguaje C++[11] tiene la intención de extender al exitoso lenguaje de programación C con mecanismos que permitieran la manipulación de objetos. En ese sentido, desde el punto de vista de los lenguajes orientados a objetos, el C++ es un lenguaje híbrido. Posteriormente se añadieron facilidades de programación genérica, que se sumó a los otros dos paradigmas que ya estaban admitidos (programación estructurada y la programación orientada a objetos).

Las ventajas que presenta es que es un lenguaje muy potente en lo que se refiere a creación de sistemas complejos, además de ser muy didáctico, por el contrario es más difícil que otros lenguajes de programación.

Se ha elegido C++ para el control de los actuadores y para la comunicación de la Raspberry Pi con Arduino debido a que la Interfaz de desarrollo Arduino IDE, que posteriormente se hablará de ella, nos lo recomienda. Este código se pasara mediante cable USB desde Arduino IDE a nuestro Arduino y allí estará ejecutándose constantemente siempre que reciba tensión como se ha comentado anteriormente.

3.3 Interfaz de Desarrollo:

3.3.1 Arduino IDE:

Para la creación del programa de la tarjeta Arduino Mega se ha utilizado el entorno de desarrollo propio de Arduino.

Para desarrollar el programa de conexión entre Arduino y Raspberry Pi es necesario utilizar Arduino IDE para implementar un programa, compilarlo y subirlo al micro controlador.

La función de Arduino IDE es simplificar el código en una sola función “Main” quitando las funciones del programa “setup y loop”, compilándolo en hexadecimal y enviándolo al micro controlador Arduino.

La plataforma de desarrollo de Arduino IDE se puede descargar de forma gratuita desde la página oficial y cuenta con numerosos tutoriales y foros de ayuda, lo que proporciona un valor añadido a este dispositivo.

3.3.2 Python IDE:

El programa de Python se ha desarrollado en un bloc de notas bajo la extensión .py

3.4 Sistemas Operativo:

3.4.1 Raspbian:

Raspbian es una distribución del sistema operativo Linux y por lo tanto de software libre basado en Debian Jessie para la placa computadora Raspberry Pi, orientado a la enseñanza de informática. Para nuestra Raspberry este sistema operativo se ha instalado en una tarjeta SD de 16 GB y entre las ventajas que nos ofrece encontramos un alto rendimiento y una interfaz gráfica sencilla.

3.5 Librerías:

Se han utilizado diferentes librerías para el desarrollo del código Python, se explican a continuación:

3.5.1 NumPy

NumPy[12] es el encargado de añadir toda la capacidad matemática y vectorial a Python haciendo posible operar con cualquier dato numérico o matriz. Esto es imprescindible ya que vamos a trabajar con imágenes y no dejan de ser matrices.

Función	Explicación
Ones	Devuelve una nueva matriz de forma y tipo determinados

Tabla 5: Función de la librería NumPy

3.5.2 Picamera

Es la librería recomendada por Raspberry Pi para la captura de imágenes y videos. En la Tabla 6 se detallan las funciones utilizadas.

Función	Explicación
Resolución	Ajusta la resolución de la cámara
Contrast	Ajusta el contraste en la cámara
Saturation	Ajusta la saturación en la cámara
Brihtness	Ajusta el brillo en la cámara
Framerate	Ajusta los fotogramas por segundo de la cámara

Tabla 6: Funciones de la librería PiCamera

3.5.3 Time

Librería que permiten entre varias opciones, manipular y dar formato a fechas y horas, obtener fechas actuales, rangos y hacer cálculos con estas.

Función	Explicación
Time.sleep	Detiene el programa la cantidad de segundos que le ponemos. Hacemos esto para que la cámara, al cambiarle la resolución, se acostumbre a la luz que recibe

Tabla 7: Función de la librería Time

3.5.4 Math

Se incluye esta librería para el uso de operaciones matemáticas.

Función	Explicación
Math.sqrt	Hacer una raíz cuadrada

Tabla 8: Función de la librería Math

3.5.5 Serial

En esta librería encontramos los comandos para interactuar con el puerto USB, ya sea bien para leer o escribir. En la siguiente Tabla vamos a ver las funciones utilizadas.

Función	Explicación
Serial	Conexión con el puerto USB
Readline	Lee información del puerto USB
Write	Envía información al puerto USB

Tabla 9: Funciones de la librería Serial

3.5.6 OpenCV

OpenCV[13] es una librería de visión por computador de código abierto. Está escrita en los lenguajes C y C++ y es compatible con Linux, Windows y Mac OS X. Cuenta con un desarrollo activo en interfaces para Python, Ruby, Matlab y otros lenguajes. OpenCV ha sido diseñado para ser eficiente en cuanto a gasto de recursos computacionales y con un enfoque hacia las aplicaciones de tiempo real.

Uno de los objetivos de OpenCV es proveer una infraestructura de Visión por Computador fácil de utilizar. Se ha instalado la versión 2.4.9. En la Tabla 10 se explican las funciones utilizadas.

Función	Explicación
Imread	Lee una imagen. La imagen debe estar en el directorio de trabajo o se debe dar una ruta completa de la imagen.
CvtColor	Convierte imágenes de un espacio de color a otro. Pueden pasar de BGR ↔HSV o BGR↔GRAY
Threshold	Se utiliza para binarizar una imagen y consiste en que si el valor del pixel es mayor que el valor umbral, se le asigna un nuevo valor que puede ser blanco o negro.
MorphologyEx	Son algunas operaciones simples basadas en la forma de la imagen. Normalmente se realiza en imágenes binarias.
GaussianBlur	Elimina el ruido Gaussiano de la imagen
Erode	Erosiona los límites del objeto en primer plano y hace que todos los píxeles cercanos al límite se descarten dependiendo del tamaño del kernel. Es útil para eliminar pequeños ruidos blancos
Canny	Algoritmo de detección de bordes.
FindContours	Localiza los contornos de la imagen y produce otra modificada. Cada contorno individual es una matriz NumPy de coordenadas (x, y) de puntos límite del objeto.
DrawContours	Se utiliza para dibujar cualquier forma siempre que se tengan sus puntos límites.
Moments	Lista de todos los valores de momento calculados
ContourArea	Nos proporciona el área del contorno.
BoundingRect	Guarda en unas variables las coordenadas del rectángulo mínimo que encierra un contorno
Rectangle	Dibuja un rectángulo mediante dos puntos, el primer punto corresponde a la esquina superior izquierda y el segundo a la esquina inferior derecha del rectángulo
InRange	Se devuelve una máscara binaria, donde los píxeles blancos (255) representan píxeles que caen dentro del rango de límite superior e inferior y los píxeles negros (0) no.

Tabla 10: Funciones de la librería OpenCV

3.6 Diagramas de flujo:

A continuación en la Ilustración 16 se muestra el diagrama de flujo total de ambos programas. El diagrama de la izquierda correspondería al de la Raspberry Pi y el de la derecha a Arduino.

3.6.1 Diagrama de flujo Raspberry Pi-Arduino:

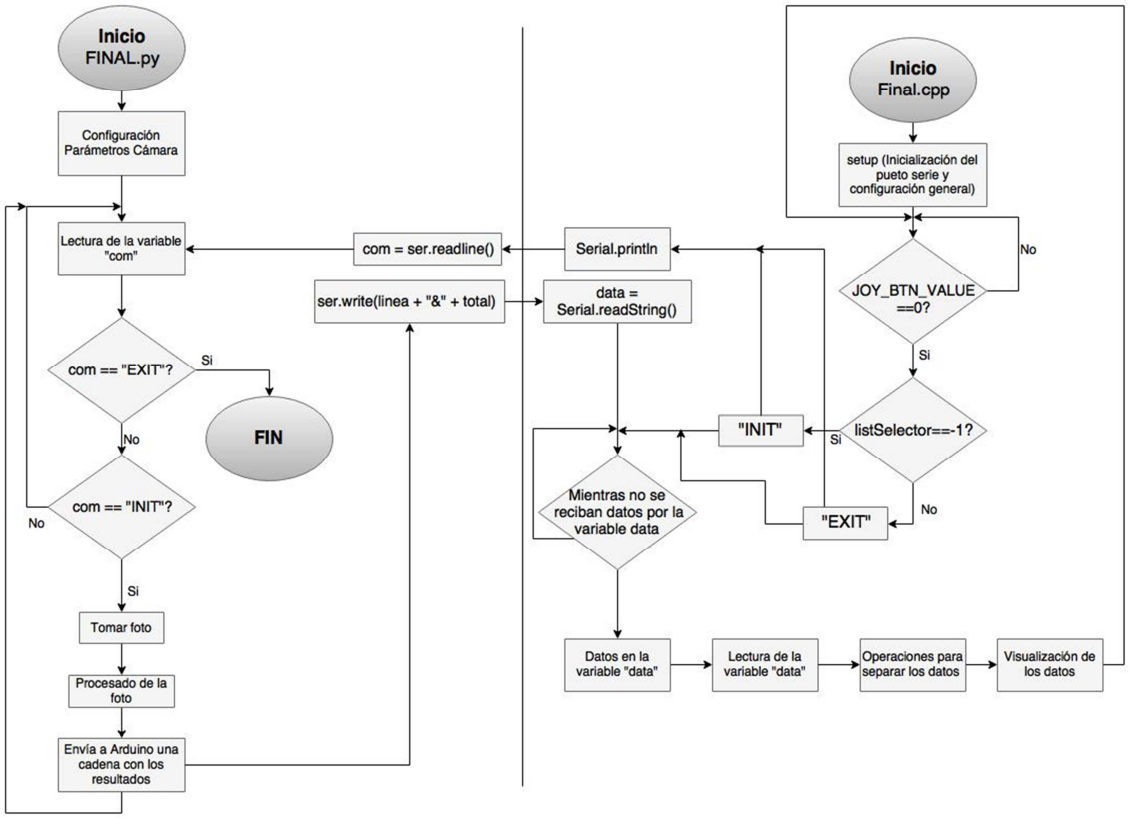


Ilustración 16: Diagrama de flujo total Raspberry Pi - Arduino

Como se ha comentado anteriormente, nuestro programa en Arduino (Final.cpp) está ejecutándose y esperando a que se ejecute el programa de Python (FINAL.py). Lo ejecutaremos de forma manual desde la terminal de Linux de nuestro ordenador.

Una vez ejecutado el programa de Arduino le envía a nuestra Raspberry Pi una cadena que se muestra por pantalla en la que pone “Arduino Conectado”, ya tenemos nuestro Arduino listo para ser usado. El programa espera a que seleccionemos mediante el Joystick si queremos tomar foto o salir. Cuando el selector del Joystick es pulsado, JOY_BTN_VALUE == 0 y dependiendo de la opción que elijamos saldremos del programa Python (se envía “EXIT”) o se tomará la foto de las monedas (“INIT”), monedas que previamente hemos de haber depositado en la base. El programa de Arduino pasará a un estado de espera hasta que vuelva a recibir la variable “data”, que es donde tenemos recogida toda la información.

Una vez tomada la foto se procesa la imagen y con la información obtenida la enviamos a Arduino, donde tras un proceso de operaciones para separar la información recibida, nos mostrará todos los datos de interés (Nº de monedas, Tipo de moneda y Suma Total) a través de los diferentes actuadores. El programa de Python volverá al bucle de espera de inicialización o salida del programa.

3.7 Explicación de los datos de comunicación:

Una vez que Raspberry Pi ha procesado la imagen y ha obtenido unos datos, crea una cadena para enviarla por USB a Arduino.

- Formato de cadena:

Estructura	Nº de moneda;Moneda1;Moneda2;Moneda3...&SumaTotal
Ejemplo	3;1 cent;2 cent; 1 euro&1.03

Tabla 11: Estructura de los datos de comunicación

Una vez que Arduino recoge la cadena primero hace un Split (dividir la cadena delimitada por un carácter) cortando por el carácter "&" para separar toda la línea en dos partes. La primera parte que es el Nº de monedas y el tipo y la segunda que es el valor total.

La primera parte es dividida una segunda vez (En este caso el delimitador es el ";") el Nº de monedas lo pasamos a la función MostrarNum para que el Diplay de 7 segmentos lo muestre y el tipo de las monedas las guardamos en un Array (Matriz con una longitud igual al número de monedas). Una vez que los datos han sido guardados los mostramos por la pantalla Nokia.

- Función para dividir la cadena:

```
String getValue(String data, char separator, int index) ;
```

Data es la cadena total que recibimos de la Raspberry y queremos dividir
Separator es el delimitador que puede ser "&" o ";"
Index es el índice de la cadena que queremos recoger del tipo de moneda

Esta función internamente es un bucle que reconoce la posición de cada delimitador y nos devuelve la subcadena (Tipo de moneda) comprendida entre dos delimitadores ";" según el index dado.

Mediante las siguientes imágenes vamos a comprender mejor el ejemplo anterior de la Tabla 11:

Pondríamos en nuestra base las monedas, en este caso 1 Euro, 1 Céntimo y 2 Céntimos

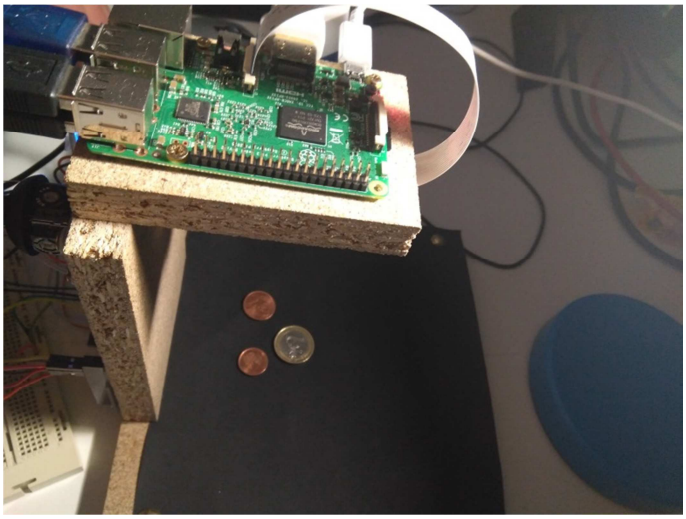


Ilustración 17: Ejemplo de colocación de monedas en la base

El Display 7 segmentos nos mostraría el número de monedas que hay

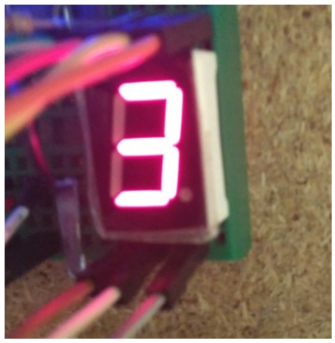


Ilustración 18: Ejemplo del número de monedas en el Display

En la pantalla Nokia vemos la suma total 1.03 y también las dos opciones de volver a iniciar el reconocimiento o por el contrario salir del programa.

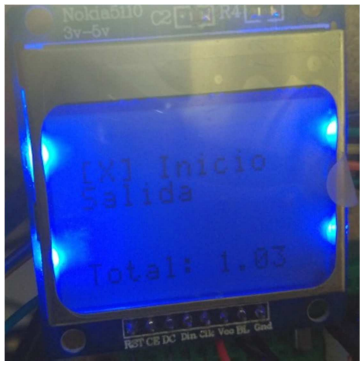


Ilustración 19: Ejemplo de la suma total de nuestras monedas

Si desplazamos el Joystick hacia la izquierda no mostraría este menú en el que nos aparece el tipo de monedas que son (1 euro, 1 cent, 2 cent)

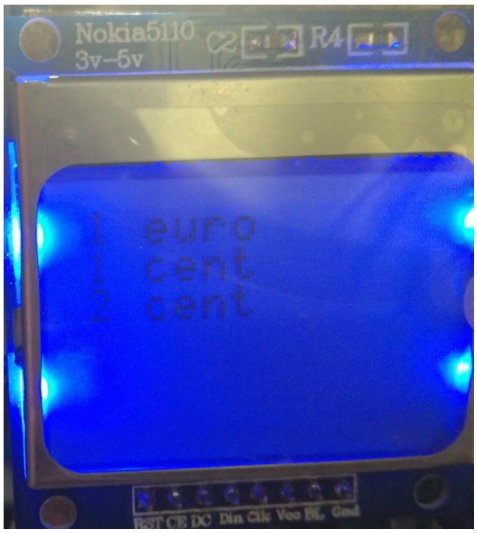


Ilustración 20: Tipos de monedas de nuestro ejemplo

4. Visión por Computador:

Para el cumplimiento de los objetivos marcados de este Trabajo de Fin de Grado se han tenido que procesar las imágenes tomadas para obtener de ellas toda la información posible. Recordemos que nuestro espacio de trabajo está formado por todas las monedas de euro de curso legal independientemente del país en el que hayan sido emitidas.

8 Monedas
1 Céntimo
2 Céntimos
5 Céntimos
10 Céntimos
20 Céntimos
50 Céntimos
1 Euro
2 Euros

Tabla 12: Tipos de Monedas a Identificar

Es importante destacar que esto supone una gran cantidad de monedas y que limita las propiedades que se pueden utilizar para reconocerlas. No se pueden usar parámetros asociados al grabado de las monedas porque hay muchas diferentes, por lo que se ha utilizado el diámetro y el color. La siguiente tabla nos muestra las características determinantes para la clasificación de cada moneda:

Monedas	Diámetro(pixeles)	Color BGR	Color HSV	Parejas Conflictivas
1 Céntimo	88-95	No se aplica	No se aplica	No tiene
2 Céntimos	100-114	No se aplica	Se aplica ([0,70,70]) ([14,255,255])	10 Céntimos
5 Céntimos	114.01-129	No se aplica	Se aplica ([0,70,70]) ([14,255,255])	20 Céntimos
10 Céntimos	100-114	No se aplica	Se aplica. Pero se identifica si no entra en este rango	2 Céntimos
20 Céntimos	114.01-129	No se aplica	Se aplica. Pero se identifica si no entra en este rango	5 Céntimos
50 Céntimos	129.01-142	Se aplica. Pero se identifica si no entra en este rango	No se aplica	1 Euro
1 Euro	129.01-142	Se aplica ([137,137,137]) ([214,214,214])	No se aplica	50 Céntimos
2 Euros	142.01-150	No se aplica	No se aplica	No tiene

Tabla 13: Parámetros de la clasificación de las Monedas

Mediante la Tabla 13 vemos que la moneda de 1 Céntimo y 2 Euros son clasificadas tan solo por su diámetro, mientras que la de 50 Céntimos y 1 Euro al tener un diámetro parecido se ha clasificado por Color BGR. Lo mismo ocurre con las monedas de 2 y 10 Céntimos y 5 y 20 Céntimos, solo que en estos casos han sido clasificadas por HSV.

Se ha implementado que el usuario mediante el selector del Joystick decida si tomar una foto de las monedas, comenzando así la clasificación o por el contrario si sale del programa.

Si elige tomar la foto previamente habrá colocado en la base las monedas a identificar y nuestro programa se encargará de mostrarnos toda la información a través de la pantalla Nokia.

La iluminación tiene un papel clave para el correcto reconocimiento de las monedas. En este caso, tras varios ensayos, se ha optado por un ambiente oscuro y tan solo la iluminación de un flexo que no incide de forma directa sobre las monedas; ya que si hubiera demasiada luz el brillo nos impediría reconocer los contornos de las monedas y si hubiera poca iluminación no se podrían apreciar los colores de ellas mismas y no se podrían diferenciar.

En primer lugar el programa obtiene el número total de monedas que se han dejado en la base, después obtiene el diámetro y las monedas de 1 céntimo (rango entre 88-95 pixeles) y 2 euros (rango entre 142.01-150 pixeles) ya pueden ser identificados. Los valores de los diámetros son valores relativos aptos para esta aplicación, los valores absolutos no nos interesan. El problema viene a continuación ya que las monedas de 2 y 10 céntimos son de un diámetro similar (rango 100-114 entre pixeles), y lo mismo ocurre con las de 5 y 20 céntimos (rango entre 114.01-129 pixeles). Estas monedas las diferenciaremos por el color, hemos establecido un rango de rojo en HSV y si el programa identifica algún pixel en ese rango será de 2 o 5 céntimos y sino pues de 10 o 20 céntimos.

Otro problema es diferenciar las monedas de 50 céntimos y la de 1 euro (rango entre 129.01-142 pixeles). En este caso ambas poseen amarillo, por lo que se decidió coger un pequeño rectángulo creado a partir del centro de la moneda, así en la moneda de euro será todo gris y en la de 50 céntimos será amarillo. Definimos un rango de grises en BGR y si encuentra algún pixel gris será de euro y sino de 50 céntimos.

La clasificación de las monedas está hecha en diferentes espacios de color (HSV y BGR) debido a que se consiguió un rango bastante bueno de gris pero pésimo de rojo en BGR y al contrario en HSV; se obtuvo un buen rango de rojo pero malísimo en gris. Debido a estos problemas, se decidió hacerlo de esta manera. También comentar que el uso habitual del espacio de color es RGB pero en la librería OpenCv se implementa como BGR.

4.1 Diagrama de bloques del procesado de imagen:

En la Ilustración 21 se puede apreciar el diagrama de bloques en cuanto a la programación de procesado de la imagen llevada a cabo en código Python.

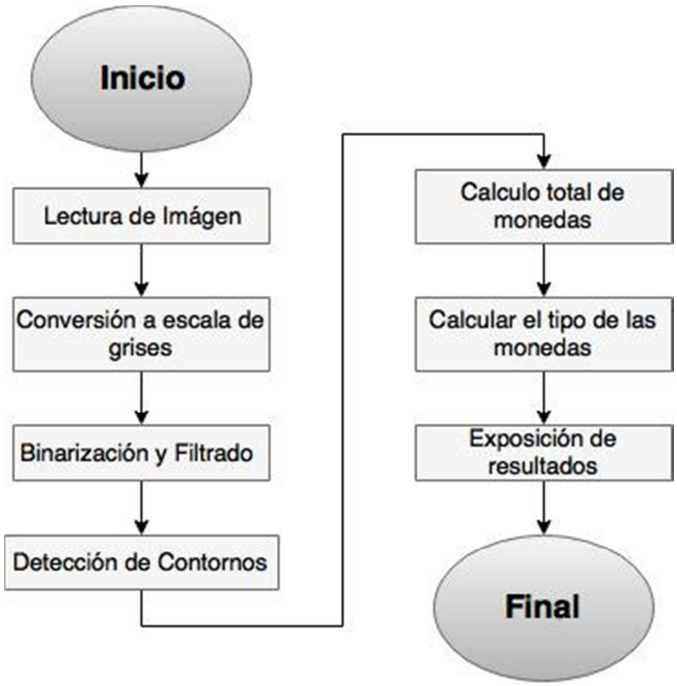


Ilustración 21: Diagrama de bloques del reconocimiento de monedas

A continuación se van a explicar todos los pasos anteriores de forma general.

Lectura de Imagen

Leemos la imagen tomada por nuestra PiCamera del directorio donde le hemos indicado que nos la guarde mediante la función de la librería Open cv "imread".

Conversión a escala de grises

Para realizar la binarización de las imágenes, estas deben ser imágenes en escala de grises. Se pasan a escala de grises mediante la función "cvtColor".

Binarización y Filtrado

Binarizamos las imágenes (threshold) mediante el método OTSU[14] calcula automáticamente un valor de umbral optimo a partir de un histograma para una imagen bimodal (imagen cuyo histograma tiene dos picos distinguidos en niveles diferentes), y aplicamos un par de operaciones morfológicas de Erosión y Cerrar (morphologyEx) para poder obtener más información de la imagen. También aplicamos un filtro gaussiano que nos elimina ruido pero nos distorsiona un poquito la imagen (GaussianBlur).

Detección de contornos

Dejamos en nuestra imagen únicamente los contornos de las monedas, que no son más que círculos. Esto se realiza mediante la función Canny[15] que es uno de los más robustos empleados en la detección y extracción de contornos.

Se basa en la información proporcionada por el gradiente de la imagen, en concreto aplicando las matrices tipo Sobel[16] (operador diferencial discreto que calcula una aproximación al gradiente de la función de intensidad de una imagen).

Además, utiliza información de vecindad para que, después de seleccionar los puntos por su gradiente elevado, localizar aquellos que forman parte efectiva de un contorno.

Calculo total de monedas

Contamos los contornos que aparecen hay en nuestra imagen y así sabemos el número total de monedas que hay. Tenemos el proceso en imágenes en la Ilustración 19.

Calcular el tipo de las monedas

Se han utilizado dos clasificaciones, por diámetro y por color. Por el diámetro serán clasificadas todas las monedas siendo las de 1 céntimo y 2 euros las únicas que puedan ser diferenciadas de una forma inmediata ya que no hay otra moneda con un rango diámetro parecido al suyo.

Cuando las monedas tienen un rango de diámetro parecido, casos de 2 y 10 céntimos, 5 y 20 céntimos, y 50 céntimos y 1 euro, la clasificación se hace a través del color. En la siguiente página se explica este proceso de forma mas detallada.

Exposición de resultados

Se saca por pantalla los resultados y también se envían a Arduino para su exposición en los distintos actuadores.

De los apartados anteriores se va a explicar el del Cálculo del tipo de las monedas de una forma más detallada y acompañada de imágenes para su mejor comprensión:

Como hemos comentado anteriormente empezamos clasificando las monedas por el rango de su diámetro. A través de este criterio se eliminan objetos de menor tamaño que las monedas que podrían ser reconocidos. Así diferenciamos de una manera clara las monedas de 1 céntimo y la de 2 euros.

El segundo criterio de clasificación es el color. Cuando el diámetro de la moneda entra dentro de un rango conflictivo la clasificación se lleva a cabo a través del color, pero se ha implementado mediante dos formas.

En monedas de 2 y 10 céntimos y monedas de 5 y 20 céntimos mediante HSV

En monedas de 50 céntimos y 1 euro mediante BGR

- Clasificación de monedas por HSV: Cuando el programa encuentra una moneda del diámetro conflictivo (monedas de 2 y 10 Céntimos, 5 y 20 Céntimos) recorta la imagen dejando la moneda sola y reduciendo sus dimensiones. Se pasas a formato HSV, que no es más que definir un modelo de color en término del tono[17] (es la característica principal de un color), la saturación[18] (intensidad o grado de pureza de cada color) y el valor[17] (diferentes mezclas que puedes obtener al mezclar un tono con blanco y negro. A mayor cantidad de blanco, mayor luminosidad.). Mediante el rango de color rojo que hemos definido previamente se crea una máscara en que los pixeles toman valores binarios dependiendo si están o no comprendidos en ese rango.

  Aplicando de nuevo las operaciones morfológicas conseguimos que las monedas de 10 y 20 céntimos queden totalmente en negro, como si no hubiera ninguna moneda. Como después tenemos una función de reconocimiento de contornos, en las monedas de 10 y 20 céntimos es 0 (que no encuentra ninguno) y para las de 2 y 5 céntimos es mayor que 0 (encuentra contornos). En la Ilustración 23 posterior se ve en un ejemplo realizado con una moneda de 5 y otra de 20 céntimos el proceso realizado para diferenciarlas.

- Clasificación de monedas por BGR: En este caso creamos un rectángulo pequeño en el centro de la moneda y lo recortamos. Obtenemos un rectángulo gris en caso de que la moneda sea de un euro y un rectángulo amarillo en la de 50 céntimos. Definimos un rango de gris en BGR, que es la composición del color en términos de la intensidad de los colores primarios de la luz, y al igual que en el caso anterior creamos una máscara en que los pixeles toman valores binarios dependiendo si están o no comprendidos en ese rango. Recorremos por filas y columnas todos los pixeles y cuando se encuentra uno blanco se incrementa una variable. Para clasificarlas ponemos la condición de que si esa variable es distinta de 0 la moneda será de euro y si es igual a 0 será de 50 céntimos. Proceso en imágenes en la Ilustración 23.

Respecto a la erosión comentar que si la función identifica nuestros objetos como pixeles blancos o bien como pixeles negros, la erosión puede convertirse en una dilatación, es decir, en vez de erosionarnos un objeto, nos lo dilatará. Efecto que se nos ha producido en la imagen 6 de nuestra Ilustración 22.

Imágenes del proceso para contar el nº total de monedas que hay:

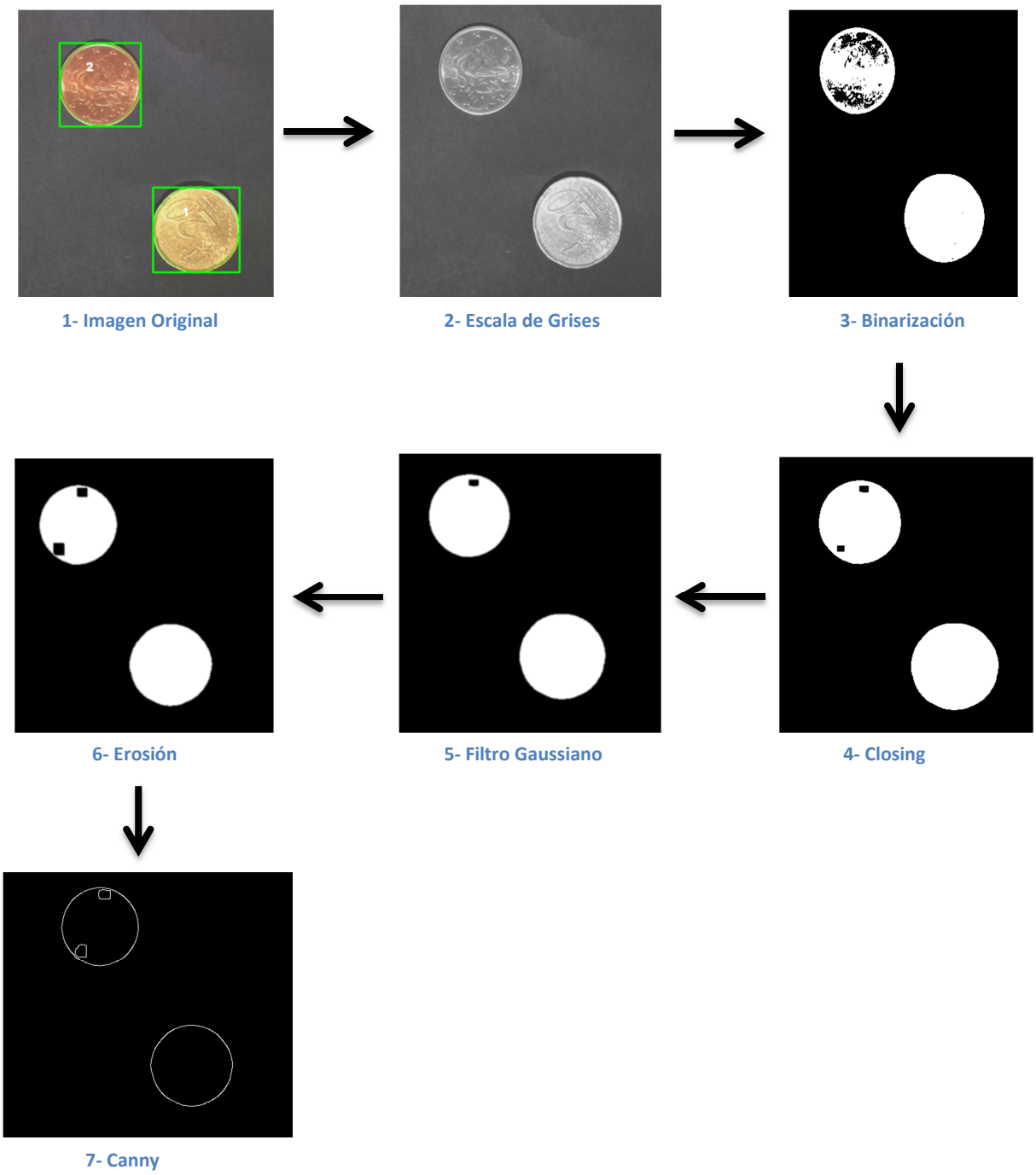


Ilustración 22: Imágenes del proceso para contar monedas

Imagenes del proceso de diferenciación entre una moneda de 5 céntimos y otra de 20 céntimos:

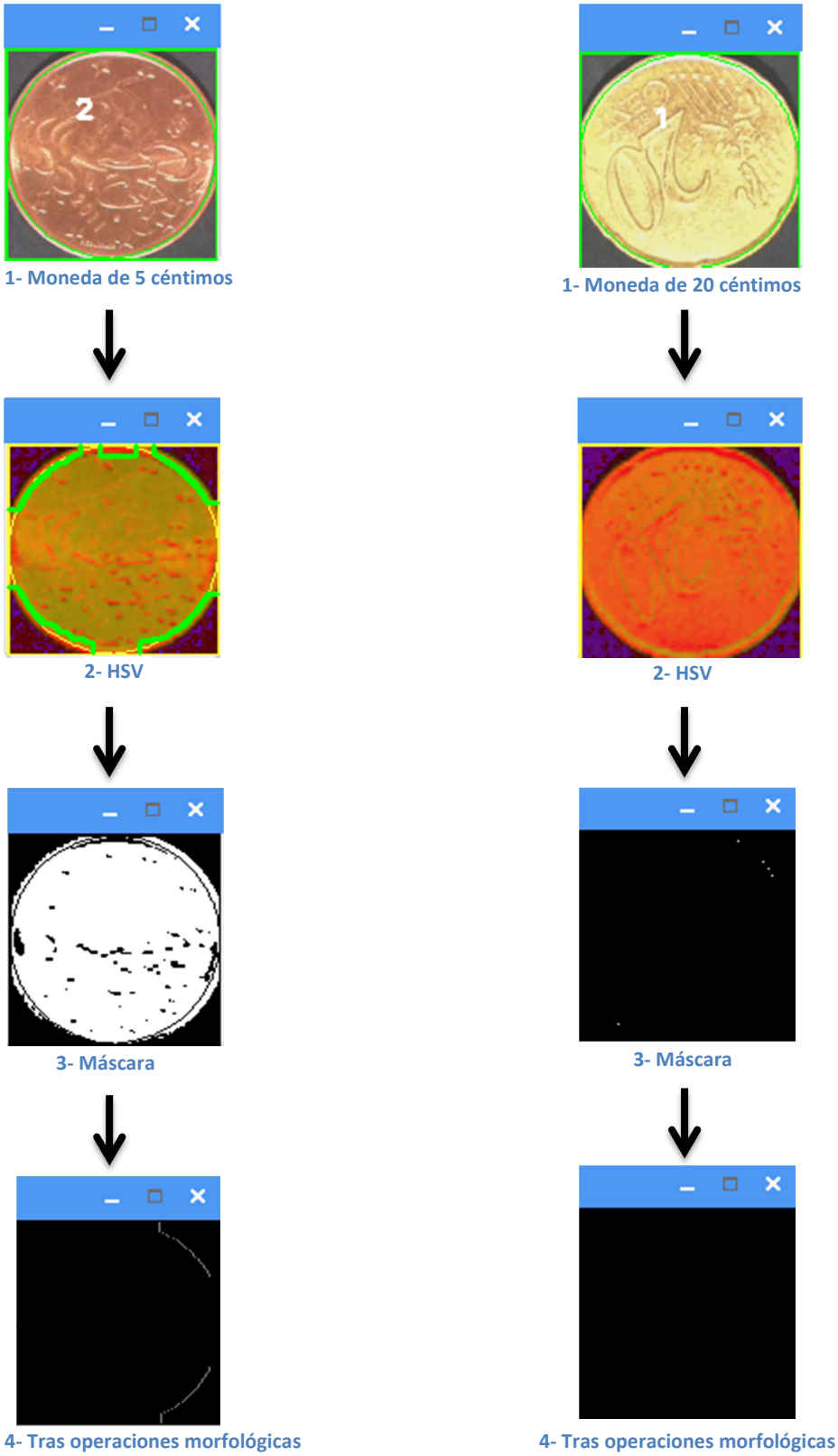


Ilustración 23: Diferenciación entre monedas de 5 céntimos y de 20

Imágenes del proceso de diferenciación entre una moneda de 50 céntimos y otra de un euro:

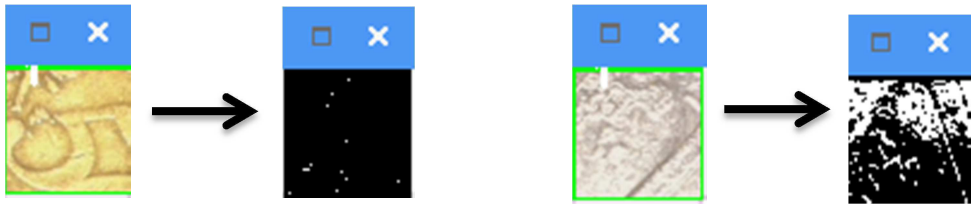


Ilustración 24: Reconocimiento monedas de 50 céntimos y de 1

Se decidió coger un rectángulo partiendo desde el centro de la moneda ya que el euro tiene un borde amarillo y eso podría confundir a nuestro programa porque la moneda de 50 Céntimos también es amarilla. Recortando un cuadrado desde el centro vemos perfectamente el color gris si es una moneda de 1 euro o, por el contrario, amarillo si es una moneda de 50 Céntimos.

4.2 Diagrama de flujo: Proceso de reconocimiento y clasificación de las monedas

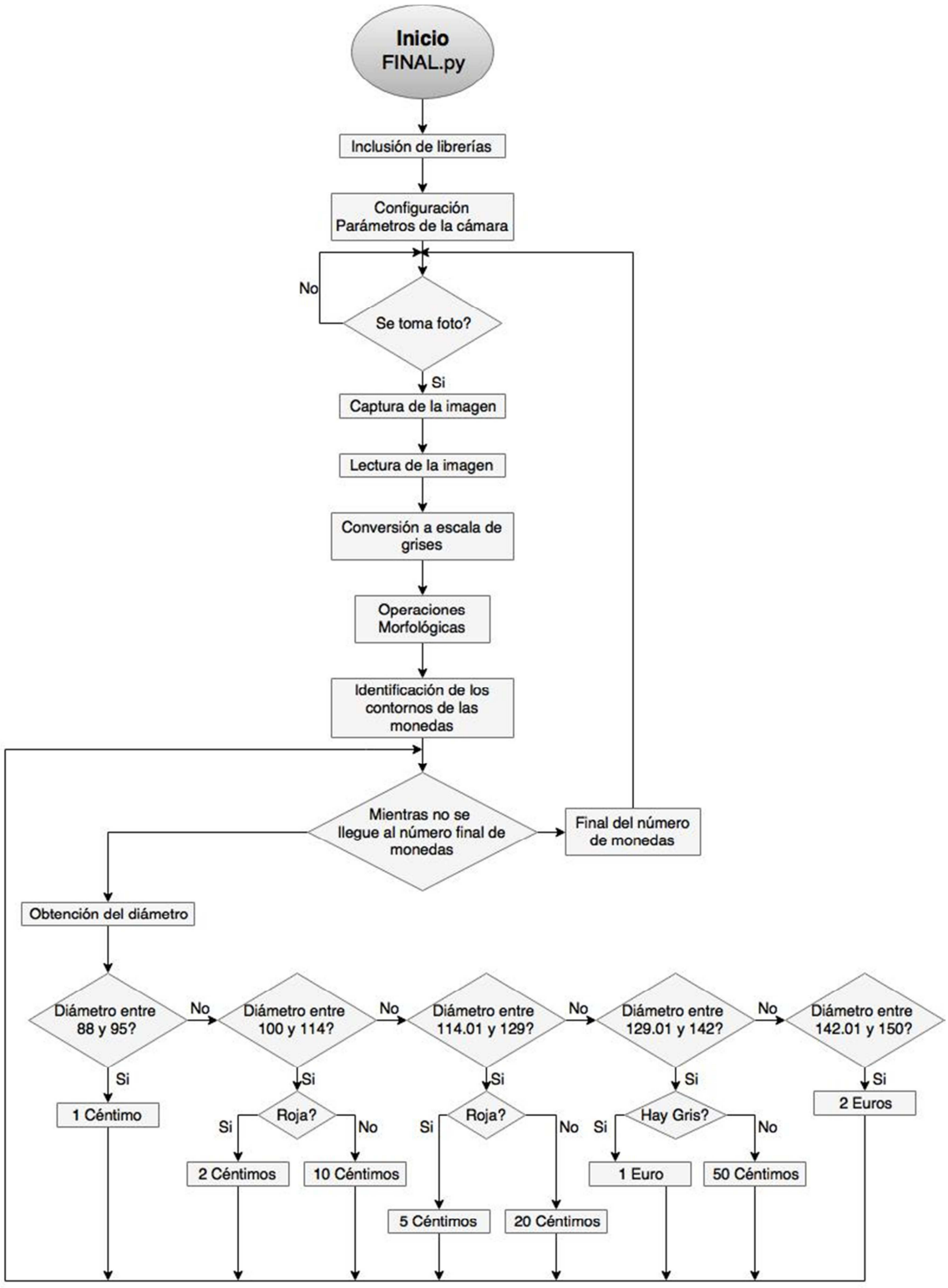


Ilustración 25: Diagrama de flujo proceso de reconocimiento y clasificación de monedas

5- Evaluación del sistema:

En este apartado de la memoria se va a proceder a comprobar que tasa de éxito tiene nuestra aplicación de reconocimiento de monedas. Se van a realizar una serie de pruebas con la iluminación fija, ya que el prototipo está preparado para que funcione así. Si intentáramos realizar otras pruebas cambiando la posición del flexo, todas las pruebas nos darían erróneas ya que si moviéramos el flexo hacia arriba los rangos de intensidad de las monedas serian distintos y ya no podrían ser clasificadas; y si lo moviéramos hacia abajo las monedas recibirían demasiada luz siendo imposible la detención de sus contornos.

Se han situado diferentes combinaciones de monedas para ver si el prototipo las identificaba correctamente. En primer lugar le pasaremos todas las monedas de euro de una en una para que las reconozca, que corresponde de la prueba 1 a la prueba 8 de la Tabla 14. Posteriormente le pasaremos las parejas de monedas que por sus rangos son conflictivas, pruebas de la 9-11. Las demás pruebas han sido hechas al azar.

La tasa de éxito ha sido determinada con la siguiente ecuación:

$$Tasa\ de\ éxito = \frac{nº\ éxitos}{nº\ pruebas} \cdot 100$$

5.1 Evaluación del sistema:

Prueba	Monedas a Identificar	Monedas Identificadas	Suma Total	Éxito
1	1 Céntimo	1 Céntimo	0.01	Bien
2	2 Céntimos	2 Céntimos	0.02	Bien
3	5 Céntimos	5 Céntimos	0.05	Bien
4	10 Céntimos	10 Céntimos	0.1	Bien
5	20 Céntimos	20 Céntimos	0.2	Bien
6	50 Céntimos	50 Céntimos	0.5	Bien
7	1 Euro	1 Euro	1	Bien
8	2 Euros	2 Euros	2	Bien
9	2 y 10 Céntimos	2 y 10 Céntimos	0.12	Bien
10	5 y 20 Céntimos	5 y 20 Céntimos	0.25	Bien
11	50 Céntimos y 1 Euro	50 Céntimos y 1 Euro	1.5	Bien
12	50 Céntimos y 2 Euros	50 Céntimos y 2 Euros	2.5	Bien
13	1, 2, y 5 Céntimos	1, 2, y 5 Céntimos	0.08	Bien
14	10, 20 y 50 Céntimos	10, 20 y 50 Céntimos	0.8	Bien
15	1 y 2 Euros	1 y 2 Euros	3	Bien
16	2 de 5 y 2 de 20 Céntimos	2 de 5 y 2 de 20 Céntimos	0.5	Bien
17	2 de 1 E y 2 de 50 Céntimos	3 de 1 E y 1 de 50 C ✖	3.5	Mal
18	3 de 1 E y 50 Céntimos	3 de 1 E y 50 Céntimos	3.5	Bien
19	10 y 20 Céntimos	10 y 20 Céntimos	0.3	Bien
20	2 y 5 Céntimos	2 y 5 Céntimos	0.07	Bien
21	1 E, 2 C y 2 de 5 C	1 E, 2 C y 2 de 5 C	1.12	Bien
22	1 E y 10 Céntimos	1 E y 10 Céntimos	1.1	Bien
23	2 de 50 Céntimos	2 de 50 Céntimos	1	Bien
24	3 de 1 E y 2 E	3 de 1 E y 2 E	5	Bien
25	4 de 5 C y 20 Céntimos	4 de 5 C y 50 Céntimos ✖	0.7	Mal
26	4 de 5 C, 2 y 1 Céntimos	4 de 5 C, 2 y 1 Céntimos	0.23	Bien
27	3 de 20 C y 2 de 10 C	3 de 20 C y 2 de 10 C	0.8	Bien
28	3 de 20 C y 3 de 10 C	4 de 20 C y 1 de 10 C ✖	1	Mal
29	2 E y 2 de 50 Céntimos	2 E y 2 de 50 Céntimos	3	Bien
30	2 E y 4 de 5 Céntimos	2 E y 4 de 5 Céntimos	2.2	Bien
31	3 de 10 C y 20 Céntimos	3 de 10 C y 20 Céntimos	0.5	Bien
32	3 de 1 E y 3 de 20 Céntimos	3 de 1 E y 3 de 20 Céntimos	3.6	Bien
33	2 de 5 C, 2 de 1E, 2 de 20C	2 de 5 C, 2 de 1E, 2 de 20C	2.5	Bien
34	3 de 10 C y 1 de 1 Céntimos	3 de 10 C y 1 de 1 Céntimos	0.31	Bien
35	3 de 5 C y 3 de 10 Céntimos	3 de 5 C y 3 de 10 Céntimos	0.45	Bien

Tabla 14: Pruebas realizadas

$$Tasa\ de\ éxito = \frac{32\ éxitos}{35\ pruebas} \cdot 100 = 91.4\ \%$$

5.2 Análisis de los datos obtenidos:

Basándonos en los datos anteriores de la Tabla 14, podemos concluir que el sistema es capaz de identificar bastante bien las monedas y hacer su suma correctamente. Las pruebas que han fallado (17, 25,28) es debido a la dificultad de diferenciar los rangos conflictivos.

El sistema tiene un rango de reconocimiento óptimo para las monedas que caen justo debajo del objetivo, ya que entran en su rango de diámetro y color correcto; conforme las monedas se alejan del objetivo, las condiciones de iluminación ya no son las adecuadas y el sistema reconoce peor las monedas involucradas con diámetros similares y también pueden salirse del rango del color estipulado para ellas.

Los casos en los que el prototipo ha fallado son los conflictivos por el rango de diámetro (monedas de 2 y 10 Céntimos, 5 y 20 Céntimos, 50 Céntimos y 1 euro). Estos fallos como hemos comentado son debidos a la iluminación, en cuanto ponemos una moneda en la base y no tiene unas perfectas condiciones de iluminación, el sistema se equivoca.

6- Conclusiones y Mejoras:

Como conclusión se puede decir que el proyecto ha cumplido los objetivos propuestos, ya que es capaz de reconocer las monedas de euro de cualquier país, un alto porcentaje de veces y realizar su suma correctamente de forma automática.

Siempre que coloquemos las monedas lo más cerca posible debajo del objetivo de la PiCamera, el sistema las reconocerá correctamente, conforme nos alejemos las posibilidades de que falle irán aumentando. Como ya hemos comentado debido a la iluminación y a los pares de monedas conflictivos.

Este Trabajo de Fin de Grado ha servido en gran parte para darme cuenta de la gran importancia que tiene la iluminación en la Visión por Computador, ya que es mucho más fructífero y fácil trabajar sobre una imagen bien iluminada que sobre una que no lo está. También adquirir nuevos conocimientos sobre el lenguaje de programación Python y ampliar los que tenía sobre C++.

Comentar que el Trabajo de Fin de Grado corresponde mayormente al área de Visión por Computador y se ha centrado casi toda la atención en ello, por lo que no se ha tenido tanto en cuenta al rendimiento ni a la arquitectura de Raspberry Pi o Arduino, simplemente son elementos que se han utilizado porque nos parecían interesantes y se querían utilizar. Como se ha dicho anteriormente se podrían haber controlado todos los actuadores (Joystick, Display y Pantalla) desde los GPIOS de la Raspberry Pi, sin necesidad de Arduino.

Futuras mejoras posibles para siguientes Trabajos de Fin de Grado:

- Que el sistema sea capaz de reconocer las monedas independientemente de la iluminación que reciba el proyecto, no como en este caso en que la luz que le incide siempre es fija, adaptación a la iluminación en tiempo real.
- Controlar todos los actuadores desde la Rasberry Pi
- En vez de una cartulina negra para la base probar con otros colores o incluso directamente sobre el aglomerado.
- Implementación de otro sistema para clasificar las monedas, como podría ser el peso de las mismas.
- Cerrar en montaje para aislar la iluminación ambiente.

Referencias Bibliográficas:

[1] «What is computer vision?» [En línea]. Disponible en: http://www.bmva.org/visionoverview. [Accedido: 03-feb-2018].
[2] «Los retos de la visión artificial en los Cursos de Verano de la UC | Noticias de Camargo: Maliaño, Muriedas, Revilla, Herrera, Cacicedo, Igollo, Escobedo y Camargo. Noticias de Cantabria». [En línea]. Disponible en: http://noticiasdecamargo.es/2012/07/02/los-retos-de-la-vision-artificial-en-los-cursos-de-verano-de-la-uc/. [Accedido: 26-ene-2018].
[3] «Raspberry Pi - Wikipedia, la enciclopedia libre». [En línea]. Disponible en: https://es.wikipedia.org/wiki/Raspberry_Pi. [Accedido: 03-feb-2018].
[4] «Camera Module V2 - Raspberry Pi». [En línea]. Disponible en: https://www.raspberrypi.org/products/camera-module-v2/. [Accedido: 09-feb-2018].
[5] «Contraste - Wikipedia, la enciclopedia libre». [En línea]. Disponible en: https://es.wikipedia.org/wiki/Contraste. [Accedido: 17-ene-2018].
[6] «El Color: Tono, saturación, brillo e iluminación». [En línea]. Disponible en: http://tonosatubrilloilu.blogspot.com.es/. [Accedido: 17-ene-2018].
[7] «Arduino Mega 2560 R3 ~ Arduino.cl». [En línea]. Disponible en: http://arduino.cl/arduino-mega-2560/. [Accedido: 17-ene-2018].
[8] «Tutorial: LCD Gráfico Nokia 5110 con Arduino • Electronilab». [En línea]. Disponible en: https://electronilab.co/tutoriales/tutorial-lcd-grafico-nokia-5110-con-arduino/. [Accedido: 17-ene-2018].
[9] «Circuito con joystick y servo | Tienda y Tutoriales Arduino». [En línea]. Disponible en: https://www.prometec.net/joystick-servo/. [Accedido: 17-ene-2018].
[10] «Display de 7 segmentos | Tienda y Tutoriales Arduino». [En línea]. Disponible en: https://www.prometec.net/display-7segmentos/. [Accedido: 17-ene-2018].
[11] «LENGUAJE DE PROGRAMACIÓN C++». [En línea]. Disponible en: http://lenguajedeprogramacion21.blogspot.com.es/. [Accedido: 08-feb-2018].
[12] «4.1. Numpy — Computación científica con Python para módulos de evaluación continua en asignaturas de ciencias aplicadas». [En línea]. Disponible en: http://webs.ucm.es/info/aocg/python/modulos_cientificos/numpy/index.html. [Accedido: 17-ene-2018].
[13] «OpenCV: Librería de Visión por Computador - Oficina de Software Libre (OSL)». [En línea]. Disponible en: https://osl.ull.es/software-libre/opencv-libreria-vision-computador/. [Accedido: 09-ene-2018].
[14] «OpenCV: Image Thresholding». [En línea]. Disponible en: https://docs.opencv.org/3.4.0/d7/d4d/tutorial_py_thresholding.html. [Accedido: 06-feb-2018].
[15] «Canny.pptx». .
[16] «Operador Sobel - Wikipedia, la enciclopedia libre». [En línea]. Disponible en: https://es.wikipedia.org/wiki/Operador_Sobel. [Accedido: 16-feb-2018].
[17] «2. Cualidades del color. Tono, saturación, luminosidad. |». [En línea]. Disponible en: http://www.lanubeartistica.es/dibujo_artistico_1/Unidad4/DA1_U4_T2/2_cualidades_del_color_tono_saturacin_luminosidad.html. [Accedido: 16-feb-2018].
[18] «Tono, saturación y luminosidad». [En línea]. Disponible en: http://naturpixel.com/2011/08/17/tono-saturacion-y-luminosidad/. [Accedido: 16-feb-2018].

ANEXO 1:
Código Python

```
# Autor : Iván Martínez Lahuerta
# GRADO EN INGENIEÍIA ELECTRÓNICA Y AUTOMÁTICA
# TRABAJO DE FIN DE GRADO
# Fichero : FINAL.py
#ESCUELA UNIVERSITARIA POLITECNICA DE TERUEL
# Breve descripcion: Sistema Automático de Reconocimiento de cualquier tipo de
  Monedas de Euro

#Inclusión de Librerías
import cv2
import numpy as np
import picamera
import time
import sys
import serial
import math

with picamera.PiCamera() as camera:
  #Configuración de los parámetros generales de PiCamera
    camera.framerate = 50                                    #Velocidad de
       fotogramas por segundo
    camera.contrast = 50                                     #Contraste
    camera.saturation = 0                                    #Saturación
    camera.brightness = 75                                   #Brillo

ser=serial.Serial('/dev/ttyACM0',9600)   #Conexión por el puerto serie
exit = 0                                 #Variable de control del bucle infinito
ctrl = -1                                #Variable de control

com = ser.readline()                     #Guarda en la variable com un primer
  string de que Arduino está conectado
print (com)                              #Nos lo escribe en la pantalla. Conexión
  correcta con Arduino

while (exit == 0):                       #Bucle Infinito que se ejecutará hasta que
   salgamos del programa

    while (ctrl == -1):                  #Permanece en este bucle hasta que le
      digamos si empieza o se sale del programa
        com = ser.readline()             #Vuelve a guardar en la variable com lo
          que lee del puerto serie
        if(com.strip() == "EXIT"):       #Si lee exit nos salimos del programa
            sys.exit()
        if(com.strip() == "INIT"):       #Si lee init toma la foto
            ctrl=1

    with picamera.PiCamera() as camera:
      #Configuración de la PiCamera para que tome la foto
        camera.start_preview()                                   #Inicia
          la cámara
        camera.resolution = (1000, 720)
          #Resulución que podemos ver por pantalla
        time.sleep(3)                                            #Tiempo
          que tarda en hacer la imagen
        camera.capture("/home/pi/Desktop/Fotos_camera/foto.jpg")
          #Guarda la imagen en ese directorio y la llama "foto"
```

```
    camera.stop_preview()                                          #Paramos
      la cámara
    camera.close()                                                 #Se
      cierra la cámara


original = cv2.imread('/home/pi/Desktop/Fotos_camera/foto.jpg')
                    #Coge la foto tomada y la guarda en "original"
gris = cv2.cvtColor(original, cv2.COLOR_BGR2GRAY)
                    #Pasamos la imagen a escala de grises
ret, thresh = cv2.threshold(gris,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)
                    #Binarizamos la imagen
kernel = np.ones((3,3), np.uint8)
                    #Máscara de 3x3
closing = cv2.morphologyEx(thresh, cv2.MORPH_CLOSE,kernel, iterations = 4)
                    #Operación morfológica "Cerrar"
gauss = cv2.GaussianBlur(closing, (5, 5), 0)
                    #Filtro Gaussiano
ero = cv2.erode(gauss, kernel, iterations = 2)
                    #Operación morfológica "Erosionar"
canny = cv2.Canny(ero,0,255)
                    #Dibujamos los contornos
(contornos,_) = cv2.findContours(canny.copy(),
  cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)       #Buscamos los
  contornos

#Inicializamos las variables
areas = []
c = 0
cent1 = 0
cent2 = 0
cent5 = 0
cent10 = 0
cent20 = 0
cent50 = 0
e1 = 0
e2 = 0
diametro = 0
total = 0
filas = 0
columnas = 0

linea = str(len(contornos))                                        #Pasamos
   a string el numero de contornos y lo guardamos en "linea"

while (c < len(contornos)):                                        #Bucle
  que se ejecutará tantas veces como mendas haya

    linea = linea + ";"                                            #Se
      añadirá detrás de cada tipo de moneda un ";"
    M = cv2.moments(contornos[c])
      #Calculamos los momentos
    cX  = int(M["m10"] / M["m00"])
    cY  = int(M["m01"] / M["m00"])
    area = cv2.contourArea(contornos[c])
      #Guardamos en "area" el area de las monedas. No es un área real
    diametro = (math.sqrt((area * 4)/3.14))                        #Sacamos
```

```
    el diámetro para ser más precisos al clasificarlas por tamaño

  if(88 < diametro < 95):                                          #Si el
    diámetro está en este rango la moneda será de 1 céntimo
      cent1 += 1
        #Incrementamos el contador de 1 céntimo
      total = total + 0.01                                         #A la
        variable total que es la suma de todas ellas le sumamos 0.01
      linea = linea + "1 cent"
        #Añadimos a la variable "linea" 1 cent, que posteriormente se verá en
        el menú
      c += 1
        #Incrementamos la variable "c" para que pase a la siguiente moenda
  elif(100 < diametro < 114):                                      #Si el
    area está comprendida en ese rango monedas de 2 y 10 céntimos
      x,y,w,h = cv2.boundingRect(contornos[c])
         #Creamos un rectángulo que recuadra la moneda
      cv2.rectangle(original, (x,y), (x+w,y+h), (0,255,0),2)
        #Dibujamos el rectángulo
      imagen =original[y:y+h,x:x+w]
        #Recortamos la imagen quedandonos solo con el rectangulo creado

      hsv = cv2.cvtColor(imagen, cv2.COLOR_BGR2HSV)
         #Pasamos ese recorte a hsv
      lower_red = np.array([0,70,70])                              #Rango
        mínimo del color rojo en hsv
      upper_red = np.array([14,255,255])                           #Rango
        máximo del color rojo en hsv

      mask = cv2.inRange(hsv, lower_red, upper_red)                #Va
        recorriendo la imagen hsv y mirando si los pixeles estan comprendidos
        en ese rango
      kernel_hsv = np.ones((3,3), np.uint8)
      closing_hsv = cv2.morphologyEx(mask,
        cv2.MORPH_CLOSE,kernel_hsv, iterations = 3)
      gauss_hsv = cv2.GaussianBlur(closing_hsv, (1, 1), 0)
      ero_hsv = cv2.erode(gauss_hsv, kernel_hsv, iterations = 3)

      contornos_hsv = []
      (contornos_hsv,_) = cv2.findContours(ero_hsv,
        cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

      cv2.drawContours(hsv, contornos_hsv, -1, (0, 255, 0), 2)

      if((len(contornos_hsv)) != 0):          #Si hay algún píxel rojo se
        incrementa el contador de 2 céntimos
          cent2 +=1
          total = total + 0.02
          linea = linea + "2 cent"
          c += 1
      elif((len(contornos_hsv)) == 0):        #Si no hay ningún píxel rojo se
        incrementa el contador de 10 céntimos
          cent10 +=1
          total = total + 0.10
          linea = linea + "10 cent"
          c +=1
```

```
elif(114.01 < diametro < 129):              #lo mismo que el caso anterior
  pero para las monedas de 5 y 20 céntimos

    x,y,w,h = cv2.boundingRect(contornos[c])
    cv2.rectangle(original, (x,y), (x+w,y+h),
    (0,255,0),2) imagen =original[y:y+h,x:x+w]

    hsv = cv2.cvtColor(imagen, cv2.COLOR_BGR2HSV)
    lower_red = np.array([0,70,70])
    upper_red = np.array([14,255,255])

    mask = cv2.inRange(hsv, lower_red, upper_red)
    kernel_hsv = np.ones((3,3), np.uint8)
    closing_hsv = cv2.morphologyEx(mask,
      cv2.MORPH_CLOSE,kernel_hsv, iterations = 3)
    gauss_hsv = cv2.GaussianBlur(closing_hsv, (1, 1), 0)
    ero_hsv = cv2.erode(gauss_hsv, kernel_hsv, iterations = 3)

    contornos_hsv = []
    (contornos_hsv,_) = cv2.findContours(ero_hsv,
      cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    cv2.drawContours(hsv, contornos_hsv, -1, (0, 255, 0), 2)

    if((len(contornos_hsv)) != 0):
        cent5 +=1
        total = total + 0.05
        linea = linea + "5 cent"
        c += 1
    elif((len(contornos_hsv)) == 0):
        cent20 +=1
        total = total + 0.20
        linea = linea + "20 cent"
        c +=1

elif(129.01 < diametro < 142):
  #Si el area está comprendida en ese rango monedas de 2 y 10 céntimos

    cv2.rectangle(original, (cX-27,cY-27), (cX+27,cY+27), (0,255,0),2)
      #Creamos un rectángulo en el centro de la moneda
    imagen =original[cY-27:cY+27,cX-27:cX+27]
      #Recortamos ese rectángulo

    lower_grey = np.array([137,137,137])
      #Valor mínimo de gris en BGR
    upper_grey = np.array([214,214,214])
      #Valor máximo de gris en BGR

    mask = cv2.inRange(imagen, lower_grey, upper_grey)

    i = 0
    j = 0

    filas, columnas = mask.shape
      #Tomamos el tamaño de la máscara
    punto = 0
```

```
        while (i < filas):
          #Iremos recorriendo cada píxel para ver si hay alguno que entre dentro
           del rango de gris

            while (j < columnas):

                if (mask[i,j] == 255):
                    punto += 1
                  #La variable punto se incrementará

                j += 1
                i += 1

        if(punto != 0):
          #Si punto es distinto de 0, moneda de 1 euro
            e1 +=1
            total = total + 1
            linea = linea + "1 euro"
            c += 1
        elif(punto == 0):
          #Si punto es igual a 0, moneda de 50 céntimos
            cent50 +=1
            total = total + 0.50
            linea = linea + "50 cent"
            c +=1

    elif(142.01 < diametro < 150):
      #Si el diámetro está en este rango la moneda será de 2 euros
        e2 += 1
        total = total + 2
        linea = linea + "2 euros"
        c += 1

    if (88 < diametro < 150):
      #Si los diámetros calculados están en este rango los escribirá por
      pantalla
        print "Objeto {}: {}".format(c,diametro)
        cv2.drawContours(original, contornos, -1, (0, 255, 0), 1)

#Sacar por pantalla los resultados
 print("He encontrado {} monedas".format(len(contornos)))
 print "Monedas 1 centimo: {}".format(cent1)
 print "Monedas 2 centimos: {}".format(cent2)
 print "Monedas 5 centimos: {}".format(cent5)
 print "Monedas 10 centimos: {}".format(cent10)
 print "Monedas 20 centimos: {}".format(cent20)
 print "Monedas 50 centimos: {}".format(cent50)
 print "Monedas 1 euros: {}".format(e1)
 print "Monedas 2 euros: {}".format(e2)

 total = str(total)                  #Pasamos a string la suma total

 ser.write(linea + "&" + total)      #Se envia a Arduino todos los datos
 ctrl = -1                           #Variable de control para volver al bucle
   incial de si queremos tomar otra foto o salir del programa
```

ANEXO 2:
Código C++

```
1   // Autor: Iván Martínez Lahuerta
2   //GRADO EN INGENIEÍIA ELECTRÓNICA Y AUTOMÁTICA
3   //TRABAJO DE FIN DE GRADO
4   //Fichero: Final.cpp
5   //ESCUELA UNIVERSITARIA POLITECNICA DE TERUEL
6   // Breve descripción: Programa para la comunicación Rasberry-Arduino y control de
    los actuadores
7
8   //Pines de la pantalla Nokia
9    const int PIN_RESET = 7;  // LCD1 Reset
10   const int PIN_SCE = 6;    // LCD2 Chip Select
11   const int PIN_DC = 5;     // LCD3 Dat/Command
12   const int PIN_SDIN = 4;   // LCD4 Data in
13   const int PIN_SCLK =3;    // LCD5 Clk
14              // LCD6 Vcc
15              // LCD7 Vled
16              // LCD8 Gnd
17
18  //Botón del Joystick
19  const int PIN_JOY_BTN = 29;
20  int       JOY_BTN_VALUE = -1; //Empieza en -1 para evitar errores
21
22  //Joystick entradas analógicas
23  const int pinJoyX = A0; //Pin del eje X del joystick
24  const int pinJoyY = A1; //Pin del eje Y del joystick
25
26  // Pantalla negro sobre blanco
27  const int LCD_C = LOW;
28  const int LCD_D = HIGH;
29
30  const int LCD_X = 84; //Pixeles en X de la pantalla
31  const int LCD_Y = 48; //Pixeles en Yde la pantalla
32  const int LCD_CMD = 0;
33
34  // Tabla para visualizar caracteres Ascii
35  static const byte ASCII[][5] =
36  {
37     { 0x00, 0x00, 0x00, 0x00, 0x00 } // 20
38     ,{ 0x00, 0x00, 0x5f, 0x00, 0x00 } // 21 !
39     ,{ 0x00, 0x07, 0x00, 0x07, 0x00 } // 22 "
40     ,{ 0x14, 0x7f, 0x14, 0x7f, 0x14 } // 23 #
41     ,{ 0x24, 0x2a, 0x7f, 0x2a, 0x12 } // 24 $
42     ,{ 0x23, 0x13, 0x08, 0x64, 0x62 } // 25 %
43     ,{ 0x36, 0x49, 0x55, 0x22, 0x50 } // 26 &
44     ,{ 0x00, 0x05, 0x03, 0x00, 0x00 } // 27 '
45     ,{ 0x00, 0x1c, 0x22, 0x41, 0x00 } // 28 (
46     ,{ 0x00, 0x41, 0x22, 0x1c, 0x00 } // 29 )
47     ,{ 0x14, 0x08, 0x3e, 0x08, 0x14 } // 2a *
48     ,{ 0x08, 0x08, 0x3e, 0x08, 0x08 } // 2b +
49     ,{ 0x00, 0x50, 0x30, 0x00, 0x00 } // 2c ,
50     ,{ 0x08, 0x08, 0x08, 0x08, 0x08 } // 2d -
51     ,{ 0x00, 0x60, 0x60, 0x00, 0x00 } // 2e .
52     ,{ 0x20, 0x10, 0x08, 0x04, 0x02 } // 2f /
53     ,{ 0x3e, 0x51, 0x49, 0x45, 0x3e } // 30 0
54     ,{ 0x00, 0x42, 0x7f, 0x40, 0x00 } // 31 1
55     ,{ 0x42, 0x61, 0x51, 0x49, 0x46 } // 32 2
56     ,{ 0x21, 0x41, 0x45, 0x4b, 0x31 } // 33 3
57     ,{ 0x18, 0x14, 0x12, 0x7f, 0x10 } // 34 4
58     ,{ 0x27, 0x45, 0x45, 0x45, 0x39 } // 35 5
59     ,{ 0x3c, 0x4a, 0x49, 0x49, 0x30 } // 36 6
60     ,{ 0x01, 0x71, 0x09, 0x05, 0x03 } // 37 7
61     ,{ 0x36, 0x49, 0x49, 0x49, 0x36 } // 38 8
62     ,{ 0x06, 0x49, 0x49, 0x29, 0x1e } // 39 9
63     ,{ 0x00, 0x36, 0x36, 0x00, 0x00 } // 3a :
64     ,{ 0x00, 0x56, 0x36, 0x00, 0x00 } // 3b ;
65     ,{ 0x08, 0x14, 0x22, 0x41, 0x00 } // 3c <
66     ,{ 0x14, 0x14, 0x14, 0x14, 0x14 } // 3d =
```

```
67      ,{ 0x00, 0x41, 0x22, 0x14, 0x08 } // 3e >
 68      ,{ 0x02, 0x01, 0x51, 0x09, 0x06 } // 3f ?
 69      ,{ 0x32, 0x49, 0x79, 0x41, 0x3e } // 40 @
 70      ,{ 0x7e, 0x11, 0x11, 0x11, 0x7e } // 41 A
 71      ,{ 0x7f, 0x49, 0x49, 0x49, 0x36 } // 42 B
 72      ,{ 0x3e, 0x41, 0x41, 0x41, 0x22 } // 43 C
 73      ,{ 0x7f, 0x41, 0x41, 0x22, 0x1c } // 44 D
 74      ,{ 0x7f, 0x49, 0x49, 0x49, 0x41 } // 45 E
 75      ,{ 0x7f, 0x09, 0x09, 0x09, 0x01 } // 46 F
 76      ,{ 0x3e, 0x41, 0x49, 0x49, 0x7a } // 47 G
 77      ,{ 0x7f, 0x08, 0x08, 0x08, 0x7f } // 48 H
 78      ,{ 0x00, 0x41, 0x7f, 0x41, 0x00 } // 49 I
 79      ,{ 0x20, 0x40, 0x41, 0x3f, 0x01 } // 4a J
 80      ,{ 0x7f, 0x08, 0x14, 0x22, 0x41 } // 4b K
 81      ,{ 0x7f, 0x40, 0x40, 0x40, 0x40 } // 4c L
 82      ,{ 0x7f, 0x02, 0x0c, 0x02, 0x7f } // 4d M
 83      ,{ 0x7f, 0x04, 0x08, 0x10, 0x7f } // 4e N
 84      ,{ 0x3e, 0x41, 0x41, 0x41, 0x3e } // 4f O
 85      ,{ 0x7f, 0x09, 0x09, 0x09, 0x06 } // 50 P
 86      ,{ 0x3e, 0x41, 0x51, 0x21, 0x5e } // 51 Q
 87      ,{ 0x7f, 0x09, 0x19, 0x29, 0x46 } // 52 R
 88      ,{ 0x46, 0x49, 0x49, 0x49, 0x31 } // 53 S
 89      ,{ 0x01, 0x01, 0x7f, 0x01, 0x01 } // 54 T
 90      ,{ 0x3f, 0x40, 0x40, 0x40, 0x3f } // 55 U
 91      ,{ 0x1f, 0x20, 0x40, 0x20, 0x1f } // 56 V
 92      ,{ 0x3f, 0x40, 0x38, 0x40, 0x3f } // 57 W
 93      ,{ 0x63, 0x14, 0x08, 0x14, 0x63 } // 58 X
 94      ,{ 0x07, 0x08, 0x70, 0x08, 0x07 } // 59 Y
 95      ,{ 0x61, 0x51, 0x49, 0x45, 0x43 } // 5a Z
 96      ,{ 0x00, 0x7f, 0x41, 0x41, 0x00 } // 5b [
 97      ,{ 0x02, 0x04, 0x08, 0x10, 0x20 } // 5c ¥
 98      ,{ 0x00, 0x41, 0x41, 0x7f, 0x00 } // 5d ]
 99      ,{ 0x04, 0x02, 0x01, 0x02, 0x04 } // 5e ^
100      ,{ 0x40, 0x40, 0x40, 0x40, 0x40 } // 5f _
101      ,{ 0x00, 0x01, 0x02, 0x04, 0x00 } // 60 `
102      ,{ 0x20, 0x54, 0x54, 0x54, 0x78 } // 61 a
103      ,{ 0x7f, 0x48, 0x44, 0x44, 0x38 } // 62 b
104      ,{ 0x38, 0x44, 0x44, 0x44, 0x20 } // 63 c
105      ,{ 0x38, 0x44, 0x44, 0x48, 0x7f } // 64 d
106      ,{ 0x38, 0x54, 0x54, 0x54, 0x18 } // 65 e
107      ,{ 0x08, 0x7e, 0x09, 0x01, 0x02 } // 66 f
108      ,{ 0x0c, 0x52, 0x52, 0x52, 0x3e } // 67 g
109      ,{ 0x7f, 0x08, 0x04, 0x04, 0x78 } // 68 h
110      ,{ 0x00, 0x44, 0x7d, 0x40, 0x00 } // 69 i
111      ,{ 0x20, 0x40, 0x44, 0x3d, 0x00 } // 6a j
112      ,{ 0x7f, 0x10, 0x28, 0x44, 0x00 } // 6b k
113      ,{ 0x00, 0x41, 0x7f, 0x40, 0x00 } // 6c l
114      ,{ 0x7c, 0x04, 0x18, 0x04, 0x78 } // 6d m
115      ,{ 0x7c, 0x08, 0x04, 0x04, 0x78 } // 6e n
116      ,{ 0x38, 0x44, 0x44, 0x44, 0x38 } // 6f o
117      ,{ 0x7c, 0x14, 0x14, 0x14, 0x08 } // 70 p
118      ,{ 0x08, 0x14, 0x14, 0x18, 0x7c } // 71 q
119      ,{ 0x7c, 0x08, 0x04, 0x04, 0x08 } // 72 r
120      ,{ 0x48, 0x54, 0x54, 0x54, 0x20 } // 73 s
121      ,{ 0x04, 0x3f, 0x44, 0x40, 0x20 } // 74 t
122      ,{ 0x3c, 0x40, 0x40, 0x20, 0x7c } // 75 u
123      ,{ 0x1c, 0x20, 0x40, 0x20, 0x1c } // 76 v
124      ,{ 0x3c, 0x40, 0x30, 0x40, 0x3c } // 77 w
125      ,{ 0x44, 0x28, 0x10, 0x28, 0x44 } // 78 x
126      ,{ 0x0c, 0x50, 0x50, 0x50, 0x3c } // 79 y
127      ,{ 0x44, 0x64, 0x54, 0x4c, 0x44 } // 7a z
128      ,{ 0x00, 0x08, 0x36, 0x41, 0x00 } // 7b {
129      ,{ 0x00, 0x00, 0x7f, 0x00, 0x00 } // 7c |
130      ,{ 0x00, 0x41, 0x36, 0x08, 0x00 } // 7d }
131      ,{ 0x10, 0x08, 0x08, 0x10, 0x08 } // 7e ?
132      ,{ 0x00, 0x06, 0x09, 0x09, 0x06 } // 7f ?
133  };
```

```
134
135  // Inicializar la pantalla Nokia
136  void LcdInitialise(void)
137  {
138     pinMode(PIN_SCE, OUTPUT);  //Ponemos los pines de la Nokia como salidas
139     pinMode(PIN_RESET, OUTPUT);
140     pinMode(PIN_DC, OUTPUT);
141     pinMode(PIN_SDIN, OUTPUT);
142     pinMode(PIN_SCLK, OUTPUT);
143
144     digitalWrite(PIN_RESET, LOW); //Reseteo de pantalla
145     digitalWrite(PIN_RESET, HIGH);
147     LcdWrite(LCD_CMD, 0x21);  // Comandos extendidos de la pantalla
148     LcdWrite(LCD_CMD, 0xBf);  // Contraste
149     LcdWrite(LCD_CMD, 0x04);  // Coeficiente temporal. //0x04
150     LcdWrite(LCD_CMD, 0x14);
151     LcdWrite(LCD_CMD, 0x0C);
152     LcdWrite(LCD_C, 0x20);
153     LcdWrite(LCD_C, 0x0C);
154  }
155
156  // Enviar byte a la pantalla
157  void LcdWrite(byte dc, byte data)
158  {
159     digitalWrite(PIN_DC, dc);
160     digitalWrite(PIN_SCE, LOW);
161     shiftOut(PIN_SDIN, PIN_SCLK, MSBFIRST, data);
162     digitalWrite(PIN_SCE, HIGH);
163  }
164
165  // Posicionar cursor en x,y
166  void gotoXY(int x, int y)
167  {
168     LcdWrite(0, 0x80 | x);  // Columna.
169     LcdWrite(0, 0x40 | y);  // Fila.
170  }
171
172  // Borrar pantalla
173  void LcdClear(void)
174  {
175     for (int index = 0; index < LCD_X * LCD_Y / 8; index++)
176     {
177        LcdWrite(LCD_D, 0x00);
178     }
179  }
180
181  // Mostrar caracter por pantalla
182  void LcdCharacter(char character)
183  {
184     LcdWrite(LCD_D, 0x00);
185     for (int index = 0; index < 5; index++)
186     {
187        LcdWrite(LCD_D, ASCII[character - 0x20][index]);
188     }
189     LcdWrite(LCD_D, 0x00);
190  }
191
192  // Mostrar string por pantalla
193  void LcdString(char *characters)
194  {
195     while (*characters)
196     {
197        LcdCharacter(*characters++);
198     }
```

```
199  }
200
201  void setup(){
202     LcdInitialise();
203     LcdClear();
204     Serial.begin(9600); //Abre el puerto serie y fija la velocidad en baudios para la
        transmisión de datos en serie
205     Serial.println("Arduino Conectado");//Mandamos por puerto serie que arduino tiene
        conexion
206
207    pinMode(23, OUTPUT);  // Asignacion de las salidas digitales del Display
208    pinMode(22, OUTPUT);
209    pinMode(24, OUTPUT);
210    pinMode(26, OUTPUT);
211    pinMode(28, OUTPUT);
212    pinMode(25, OUTPUT);
213    pinMode(27, OUTPUT);
214    pinMode(29, INPUT);
215  }
216
217  void display (int a, int b, int c, int d, int e, int f, int g)// Funcion del Display
218  {
219    digitalWrite (22,a);   //Se reciben 7 variables y se asignan a cada una de las
       salidas
220    digitalWrite (23,b);
221    digitalWrite (24,c);
222    digitalWrite (26,d);
223    digitalWrite (28,e);
224    digitalWrite (25,f);
225    digitalWrite (27,g);
226  }
227
228  //VARIABLES DE CONTROL
229
230  String arrayData[25];   //Creamos una cadena de una longuitud de 25, donde se
     guardaran los valores que entren por el puerto serie referidos al tipo de moneda
231  String dataSplit[2];    //Creamos un array en el que guardamos las dos partes de la
     cadena total delimitadas por el &
232  int minShow=0;          //Variable encargada de controlar que no nos pasemos hacia
     lineas superiores y muestre el principio de la lista
233  int maxShow=6;          //Variable encargada de controlar la ultima fila y mostrar
     el ultimo item de la lista que cabe en pantalla
234  int lenght = 0;         //Numero total de monedas(primer valor dado en la cadena
     procedente de Raspberry Pi)
235  int IndexMenu=1;        //Selector del menu a mostrar en pantalla
236  int listSelector = -1;
237
238
239  void loop() {
240
241    JOY_BTN_VALUE = digitalRead(PIN_JOY_BTN);  //Lee el botón del Joystick
242
243    if(JOY_BTN_VALUE==0)              //Si el botón del Joystick está pulsado
244    {
245      if(listSelector==-1)            //Según la opción selecionada enviará a la
       Raspberry Pi que inicie el programa o lo cierre
246        Serial.println("INIT");
247      else if(listSelector==1)
248        Serial.println("EXIT");
249    }
250
251    if(lenght>0 && lenght <10)MostrarNum(lenght);     //Si se cumple está condición se
       muestra el número por el Display

252
253    while(Serial.available() > 0) //Solo funcionará este bucle mientras el puerto
       serie este recibiendo datos
254    {
```

```
255       String data = Serial.readString(); //Pasamos los datos del puerto serie a una
          variable local llamada data
256
257       for(int x = 0; x < 25; x++) arrayData[x] = "";   //Inicializa la variable
          arrayData
258
259       dataSplit[0] = getValue(data,'&',0).c_str();     //Primera parte de la cadena
260       dataSplit[1] = getValue(data,'&',1).c_str();     //Sedunda parte de la cadena
261
262       lenght = getValue(dataSplit[0],';',0).toInt(); //Recibimos el primer valor de la
          cadena y lo pasamos a int para saber el numero total de monedas
263
264          for (int i = 1; i<= lenght; i++)   //Bucle que asigana a arrayData cada
             substring del tipo de moneda
265          {
266           arrayData[i-1]=getValue(dataSplit[0],';',i).c_str();
267          }
268
269          if(IndexMenu==-1)LcdClear(); //Si estamos en el menu de mostrar monedas
             borramos la pantalla para que se actualice porque hemos recibido nuevos datos
270     }
271
272
273     if(lenght<0)return; //Si no se han pasado monedas se sale de la función
274
275     int AxiX = GetJoyX(); //Variable de control temporal del eje x
277     if (IndexMenu!=AxiX && AxiX != 0)   //Condición para cambian de menu
278     {
279       LcdClear();       //Si el eje x cambia, se actualiza el valor de la variable
          global IndexMenu
280       IndexMenu=AxiX;
281     }
282
283     if(IndexMenu==-1) //Si es -1 muestra el menu de lista de monedas
284     {
285           int Ylist = GetJoyY();  //Recibe el valor del eje y para cambiar lo que se
              muestra en la lista
286            gotoXY(0,0);           //Decimos a la pantalla que escriba desde arriba
               izquierda
287
288            if(Ylist==1) {         //Si el valor de y es 1, subimos la lista, siempre
               que sea posible
289               if(maxShow<lenght){
290                 LcdClear();
291                 minShow++; maxShow++;
292               }
293             }
294            if(Ylist==-1) {       //Si el valor de y es -1, bajamos la lista, siempre
               que sea posible
295               if(!minShow<=0){
296                 LcdClear();
297                 minShow--; maxShow--;
298               }
299             }
300
301            int yshower=1;      //Variable para que empiece a escribir desde el
               principio de pantalla
302            for (int i = minShow; i<= maxShow; i++)   //Bucle para escribir la lista
               del tipo de moneda
303            {
304               LcdString(arrayData[i].c_str());
305               gotoXY(0,yshower);
306               yshower++;
307            }
308     }

309     else if(IndexMenu==1) //Si es 1 muestra el menu de selección de inicio del programa
```

```
310     {
311             int t_y = GetJoyY();                                //Variable local donde se
                guarda el valor de GetJoyY
312             listSelector = t_y != 0 ? t_y : listSelector;    //Cambiamos listSelector
                si t_y es distinto de 0
313
314             if(t_y != 0)
315             {
316               LcdClear();              //Si t_y no es 0 limpiamos la pantalla
317               listSelector = t_y;
318             }
319
320               gotoXY(0,1);             //Posicinamos cursor
321             if(listSelector==-1)       //Si el Joystick Y está hacia arriba el valor
                selecionado será inicio
322             {
323               LcdString("[X] Inicio");
324             } else LcdString("Inicio");
325               gotoXY(0,2);            //Posicinamos cursor
326             if(listSelector==1)       //Si el Joystick Y está hacia abajo el valor
                selecionado será salida
327             {
328               LcdString("[X] Salida");
329             } else LcdString("Salida");
330
331             String tempTotal = "Total: " + dataSplit[1] + " ";   //Creamos una
                variable local de tipo string y le pasamos dataSplit[1] que sería el valor
                total de las monedas
332             char charBuf[12]; //Creamos un char array para usarlo como buffer(variable
                temporal para guardar datos que luego se modifican), 13 es la longitudo
                del array, en este caso 13 caracteres; 6 fijos por "Total: " + nuestro
                margen para el total de monedas
334             gotoXY(0,5);           //Posición del cursor línea 5
335             tempTotal += charBuf;  //Concatenamos temptotal con la longuitud
336             tempTotal.toCharArray(charBuf,12); //Coge temptotal y lo pasa a charbuf en
                forma de array
337             LcdString(charBuf);    //Muestra por pantalla el total de las monedas.
338      }
339   }
340
341   int GetJoyX()                     // Aqui cambiamos el valor analogico de 0 a 1024
      por -1, 0 y 1.
342   {
343     int rx = analogRead(pinJoyX);
344     delay(100); //dejamos que ardu descanse
345     if(rx<125)return -1;
346     if(rx>400 && rx<600) return 0;
347     if(rx>800)return 1;
348
349     return 0; //si el Joy fallase retornaria 0 siempre.
350    }
351
352    int GetJoyY()
353    {
354      int ry = analogRead(pinJoyY);
355     delay(100);
356     if(ry<125)return -1;
357     if(ry>400 && ry<600) return 0;
358     if(ry>800)return 1;
359
360     return 0;
361    }
362
```

```
363    String getValue(String data, char separator, int index) //Funcion para dividir la
       cadena y recoger el valor de esta segun el indice que pasamos como argumento.
364   {
365       int found = 0;
366       int strIndex[] = { 0, -1 };
367       int maxIndex = data.length() - 1;
368
369       for (int i = 0; i <= maxIndex && found <= index; i++) { //Bucle para controlar
          el indice segun los separadores.
370           if (data.charAt(i) == separator || i == maxIndex) {
371               found++;
372               strIndex[0] = strIndex[1] + 1;
373               strIndex[1] = (i == maxIndex) ? i+1 : i;
374           }
375       }
376       return found > index ? data.substring(strIndex[0], strIndex[1]) : ""; //retorna
          la subcadena de 'data' que este en ese indice.
377   }
378
379   void MostrarNum(int x) //Recibe el primer dígito de la "string" y lo muestra en el
      Display
380   {
381   if (x==0)       display (1,1,1,1,1,1,0); //escribe 0
382   else if (x==1)  display (0,1,1,0,0,0,0); //escribe 1
383   else if (x==2)  display (1,1,0,1,1,0,1); //escribe 2
384   else if (x==3)  display (1,1,1,1,0,0,1); //escribe 3
385   else if (x==4)  display (0,1,1,0,0,1,1); //escribe 4
386   else if (x==5)  display (1,0,1,1,0,1,1); //escribe 5
387   else if (x==6)  display (1,0,1,1,1,1,1); //escribe 6
388   else if (x==7)  display (1,1,1,0,0,0,0); //escribe 7
389   else if (x==8)  display (1,1,1,1,1,1,1); //escribe 8
390   else if (x==9)  display (1,1,1,0,0,1,1); //escribe 9
391   }
392
393
```

ANEXO 3: Enlaces de páginas web interesantes y Datasheets

-Guía para un implementar un código limpio en Arduino:

https://www.arduino.cc/en/Reference/StyleGuide

-Guía de las funciones de Arduino:

https://www.arduino.cc/reference/en/

-Guía lenguaje de programación en C++:

http://informatica.uv.es/iiguia/AED/laboratorio/Estilocpp.pdf

-Manual de consulta para programar Arduino:

http://www.ardumania.es/wp-content/uploads/2011/10/Arduino_programing_notebook_ES.pdf

-Datasheet PiCamera:

https://media.readthedocs.org/pdf/picamera/latest/picamera.pdf

-Datasheet Arduino Mega:

https://www.robotshop.com/media/files/pdf/arduinomega2560datasheet.pdf

-Datasheet Joystick:

http://www.energiazero.org/arduino_sensori/joystick_module.pdf

-Datasheet pantalla Nokia:

https://cdn-learn.adafruit.com/downloads/pdf/nokia-5110-3310-monochrome-lcd.pdf

-Datasheet Display 7 segmentos:

https://e-radionica.com/productdata/LD3361BS.pdf

-Datasheet Raspberry Pi 3 modelo B:

http://docs-europe.electrocomponents.com/webdocs/14ba/0900766b814ba5fd.pdf

-Guía lenguaje de programación en Python:

http://docs.python.org.ar/tutorial/pdfs/TutorialPython2.pdf