UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA

La Universidad Católica de Loja

ÁREA TÉCNICA

TÍTULO DE INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

Desarrollo de algoritmos de aprendizaje automático basados en plataformas open-hardware y open-software, aplicados a un módulo de educación básica inclusiva.

TRABAJO DE TITULACIÓN

AUTOR: Sarmiento Sinche, Santiago Javier

DIRECTOR: Calderón Córdova, Carlos Alberto, Mgtr.

LOJA – ECUADOR

2019

2019

APROBACIÓN DEL DIRECTOR DE TRABAJO DE TITULACIÓN

Mgtr.

Carlos Alberto Calderón Córdova.

DOCENTE DE LA TITULACIÓN

De mi consideración:

El presente trabajo de titulación: **Desarrollo de algoritmos de aprendizaje automático basados en plataformas open-hardware y open-software, aplicados a un módulo de educación básica inclusiva**, realizado por **Sarmiento Sinche Santiago Javier**, ha sido orientado y revisado durante su ejecución, por cuanto se aprueba la presentación del mismo.

Loja, Junio de 2019

f) ……

DECLARACIÓN DE AUTORÍA Y CESIÓN DE DERECHOS

"Yo **Santiago Javier Sarmiento Sinche** declaro ser autor del presente trabajo de titulación: Desarrollo De Algoritmos De Aprendizaje Automático Basados En Plataformas Open-Hardware Y Open-Software, Aplicados A Un Módulo De Educación Básica Inclusiva, de la Titulación de Electrónica y Telecomunicaciones, siendo Carlos Alberto Calderón Córdova director del presente trabajo; y eximo expresamente a la Universidad Técnica Particular de Loja y a sus representantes legales de posibles reclamos o acciones legales. Además, certifico que las ideas, conceptos, procedimientos y resultados vertidos en el presente trabajo investigativo, son de mi exclusiva responsabilidad.

Adicionalmente declaro conocer y aceptar la disposición del Art. 88 del Estatuto Orgánico de la Universidad Técnica Particular de Loja que en su parte pertinente textualmente dice: "Forman parte del patrimonio de la Universidad la propiedad intelectual de investigaciones, trabajos científicos o técnicos y tesis de grado o trabajos de titulación que se realicen con el apoyo financiero, académico o institucional (operativo) de la Universidad".

f. ……
Autor. Santiago Javier Sarmiento Sinche
Cédula. 1105029456

DEDICATORIA

A mi querida familia y amigos.

Santiago Javier Sarmiento Sinche

AGRADECIMIENTOS

Un agradecimiento especial a mi familia por haberme apoyado en el transcurso de la carrera, a todos mis amigos y compañeros que formaron parte de mi día a día. A mis profesores, tutores y director de trabajo de titulación por todas las enseñanzas.

Santiago Javier Sarmiento Sinche

ÍNDICE DE CONTENIDOS

LISTA DE FIGURAS

LISTA DE TABLAS

RESUMEN

El concepto de inclusión educativa se basa en la premisa de vincular y contener la mayor cantidad de personas en el proceso de aprendizaje, ante la posible diversidad existente en el espacio educativo, ya sea referida a capacidades especiales, aspectos étnicos y entornos culturales.

La visión por computadora ha desarrollado una serie de herramientas y soluciones, que hoy en día vuelcan su esfuerzo a sistemas de asistencia para un número cada vez más creciente de usuarios en distintas áreas de aplicación práctica.

Tomando en consideración lo antes mencionado, se ha visto la necesidad de desarrollar un sistema basado en visión por computador para niños con discapacidad visual como una alternativa a la enseñanza y aprendizaje de matemáticas, mediante un juego de mesa integrado en una placa reducida de bajo costo, que supervise y corrija las actividades realizadas por el usuario.

Se describe a continuación el desarrollo, diseño de software-hardware, la implementación física del prototipo, y finalmente, la evaluación del desempeño del prototipo en un entorno real.

PALABRAS CLAVES: Educación básica inclusiva, Raspberry pi, reconocimiento óptico de caracteres, OpenCV, procesamiento de imágenes, Python, visión por computadora.

ABSTRACT

The concept of educational inclusion is based on the premise of linking and containing as many people as possible in the learning process, given the possible diversity in the educational space, whether it concerns special abilities, ethnic aspects and cultural environments.

Computer vision has developed a number of tools and solutions, which today turn their efforts to support systems for an increasing number of users in different areas of practical application.

Taking into consideration the above, it has been seen the need to develop a computer vision based system for children with visual impairment as an alternative to teaching and learning mathematics, by means of a board set integrated in a low-cost single-board computer, that supervises and corrects the activities carried out by the user.

The development, software-hardware design, physical implementation of the prototype, and finally the evaluation of prototype performance in a real environment are described below.

KEYWORDS: Basic and inclusive education, Raspberry pi, optical character recognition, Opencv, image processing, Python, computer vision.

INTRODUCCIÓN

Según fuente del INEC (Instituto Nacional de Estadísticas y Censos); en el Ecuador existen alrededor de 186.000 personas con discapacidad visual. De las cuales 17.000 son niños con discapacidad visual en edad escolaridad (menores de 12 años). Pero solo el 77 % acceden a formación escolar (INEC (Instituto Nacional de Estadisticas y Censos), 2018). Y las Instituciones de educación especial carecen de material dirigido a niños con discapacidades visuales y de personal docente especial.

El Estado ecuatoriano, a través del artículo 47 de la Ley Orgánica de Educación Intercultural (Ministerio de Educación del Ecuador, 2017), manifiesta que se garantizará la inclusión e integración de personas con discapacidad en los establecimientos educativos, eliminando las barreras de aprendizaje. Para complementar, en los artículos 229 y 230 se indica que profesionales especializados deberán definir cuál es la modalidad más adecuada para cada estudiante y así adaptar los estándares de aprendizaje de acuerdo con las necesidades de cada alumno.

La idea de brindar educación de calidad y con recursos educativos apropiados para todo tipo de personas, llega a ser algo utópico, pues, aunque las leyes estipulen que se debe incluir y educar a todos los grupos, la realidad es que en gran parte de las instituciones de educación primaria no existe el material educativo necesario.

En el contexto de la enseñanza y la educación cognitiva, los juegos interactivos constituyen una herramienta indispensable para el aprendizaje desde edades tempranas, estimulando así la creatividad y su razonamiento lógico.

En base a lo comentado los párrafos anteriores, se formula así un nuevo concepto en materia de enseñanza y aprendizaje. Un prototipo que permita automatizar la educación de estudiantes con discapacidad visual, en donde el dispositivo guie, corrija y supervise las actividades desarrolladas por el usuario final. Integrando conceptos de visión por computadora y programación, basada en una filosofía de Open Source que no restrinja el acceso al código, y que permita modificaciones por parte de otros desarrolladores.

El trabajo de fin de titulación presentado a continuación tiene como finalidad mejorar el proceso de aprendizaje mediante juegos educativos e interactivos, entre personas con

discapacidad visual y un ordenador, o smartphone, usando herramientas de visión por computadora, para adquirir y procesar imágenes y videos del juego en tiempo real.

El diseño físico de este proyecto consta de los elementos siguientes: tarjeta de juego en el que consta el número y su representación en braille, tablero y su respectivo soporte para una cámara, y finalmente una tarjeta electrónica.

En cuanto al diseño de software, se usa la herramienta de programación Python que facilita y favorece el acceso a diversidad de dispositivos que puedan soportar con el programa, así mismo se hace uso de librerías como OpenCV que permite desarrollo de procesamiento de imágenes, Tesseract que posibilita el reconocimiento de caracteres.

OBJETIVOS

Objetivo General

- Desarrollar algoritmos de aprendizaje automático basados en plataformas open-hardware y open software aplicados, a un módulo de educación básica inclusiva.

Objetivos Específicos

- Diseñar una arquitectura hardware/software/firmware aplicada al prototipo de educación básica inclusiva.
- Diseñar y desarrollar el conjunto de algoritmos de: aprendizaje automático, interfaz con periféricos, y, algoritmo del juego educativo.
- Implementar el prototipo hardware y firmware para el módulo de educación básica inclusiva.
- Evaluar el desempeño del prototipo en un entorno real.

ESTRUCTURA DEL TRABAJO DE TITULACIÓN

En el presente trabajo de titulación realiza y desarrolla algoritmos de aprendizaje aplicados a un módulo de educación básica inclusiva. A continuación, se detalla su estructura:

Capítulo 1: Establece los fundamentos y conceptos matemáticos del procesamiento de imágenes y el estado del arte.

Capítulo 2: Diseña, desarrolla e implementa el conjunto de algoritmos de: aprendizaje automático, interfaz con periféricos, y, algoritmos del juego educativo.

Capítulo 3: Evalúa el desempeño del prototipo en un entorno real.

CAPÍTULO I: MARCO TEÓRICO

1.1 Trabajos Relacionados

En el mercado y en la academia se encuentran desarrolladas plataformas innovadoras aplicadas en el ámbito educativo, que sirven de ejemplo para el fin por el cual se está desarrollando el dispositivo en cuestión. A continuación, se presenta un breve resumen de los trabajos y dispositivos relacionados, los autores y el enfoque de la herramienta creada:

La empresa desarrolladora de juegos y aplicaciones: OSMO, es una empresa que crea juegos educativos diseñados para niños de edades de 5 a 11 años, en base a estos juegos fomenta el aprendizaje en áreas clave como: resolución creativa de problemas, arte, ciencia, tecnología, ingeniería y matemáticas (OSMO, 2018).

En (Caporusso, Mkrtchyan, & Badia, 2010) las interfaces de computadora para los juegos de mesa se basan generalmente en el canal visual. El artículo discute un sistema multipropósito que permite especialmente a las personas ciegas y sordociegas jugar al ajedrez u otros juegos de mesa a través de una red, reduciendo así su barrera de discapacidad.

El dispositivo propuesto es capaz de dar una representación no secuencial de la información; Por lo tanto, los jugadores con problemas visuales pueden navegar libremente por las plazas y aprender sobre la configuración del juego como si estuvieran tocando las piezas reales. Incrusta una pantalla Braille y un sensor óptico junto con un sistema de información táctil. La entrada se adquiere detectando la presión del dedo con una rejilla de 64 electrodos, mientras que la salida utiliza corriente eléctrica de bajo voltaje como estímulo; los usuarios son capaces de percibir la configuración real del entorno del tablero.

En (Ohene-Djan & Fernando, 2016) presenta un conjunto de herramientas de gramática que pueden ser utilizadas por personas ciegas para encontrar la orientación espacial, así como el conocimiento y la representación del entorno de dibujo interactivo, para mejorar la comprensión de un concepto o un tema. documento demuestra una arquitectura en la que las personas ciegas manipular gráficos a través de un sistema informático. La gramática se introduce de una manera que se puede personalizar y reutilizar en diferentes contextos y otras lenguas habladas.

La capacidad del software para reflejar claramente la imagen mental de un usuario ciego mediante un mapa de imágenes generado por ordenador. La arquitectura permite a los usuarios definir objetos y reutilizarlos cuando sea necesario, buscar un punto específico en la pantalla mediante un seguimiento de ubicación relativo y especificar el tamaño por dimensiones relativas de la pantalla.

En (Sribunruangrit, Marque, Lenay, & Gapenne, 2004) permite a las personas con discapacidad visual acceder a la información gráfica en la pantalla del ordenador mediante la percepción táctil. Aplica la "caja braille" en la clase de matemáticas enfocada en gráficas lineales, con niños con discapacidad visual. El resultado muestra que los usuarios pueden realizar tareas como determinar la pendiente, la intersección y las coordenadas de la intersección de dos líneas. El sistema de ayuda "caja braille", compuesto por un lápiz óptico y una tableta, está interconectado con un ordenador mediante un software específico: TACTOS, desarrollado por el grupo Perceptive Supplementation. Puede ayudar a las personas con discapacidad visual a analizar gráficos lineales y permite al usuario explorar la forma y recibir la percepción táctil en la misma mano.

En (Oliveira, Quek, Cowan, & Fang, 2012) Se utiliza información sensorial y visión por computadora para reemplazar el material visual. Implementa el uso de guantes hápticos con el rastreo basado en la visión por computadora ayuda a mantener el foco de lectura en una representación en línea elevada de un gráfico impreso al que el instructor señala mientras habla, para lograr un sistema completo para enseñanza inclusiva.

En (Zhang, 2010) describe un sistema interactivo dirigido a la navegación de mapas por un usuario ciego, es un sistema háptico con diferentes niveles de información, incluye generación automática de modelos a partir de colecciones de imágenes y video.

En (Gaudina, Zappi, Brogni, & Caldwell, 2012) se presenta un sistema de juegos interactivo para personas con discapacidad visual, en el cual combina métodos de háptica, audio y visión por computadora. Es un juego musical donde se usan interfaces graficas como OpenCV, OpenGL, y métodos auditivos para guiar en el juego a la persona con discapacidad visual.

En (Matsuo, 2016) diseña un juego de laberinto basado en sistemas táctiles y con interfaz gráfica, permite a los las personas con discapacidad visual crear escenarios para jugar junto a más personas.

En (Park et al., 2016) presenta el diseño de un robot de resolución de matemáticas que tiene diversas aplicaciones en robótica educativa. Escanea y reconoce un problema de matemáticas impreso en un papel y luego escribe una respuesta en él, soluciona los problemas lógico-matemáticos y realiza cálculos simples.

En (Ferreira & Cavaco, 2015) se desarrolló un juego matemático apto para niños y jóvenes con discapacidad visual, en nivel de escolaridad primaria y secundaria, sin necesidad de gráficos, las características se complementan con audio y el juego utiliza audio espacial 2D. Este fue probado en dos escuelas con niños ciegos y con baja visión obteniendo gran aceptación.

1.2 Introducción

El desarrollo y avance de tecnologías permitió mejorar sustancialmente la calidad de vida de las personas, tanto en las ramas de la medicina, transporte, telecomunicaciones, etc. Ocurre de igual forma en la calidad de educación. Mejoras que van desde el desarrollo de plataformas interactivas hasta aulas virtuales. Pero a medida que nuevas herramientas surgen, el proceso de adaptación del usuario toma más tiempo.

Pese a esto hay un sector en la educación que en el Ecuador y a nivel global ha estado marginado, la educación inicial para personas con capacidades especiales, Para el niño con discapacidad, la integración temprana en un ámbito "normalizado" significa también una mejor inserción en la sociedad. Se debe buscar que estos infantes se adapten a los nuevos niveles de enseñanza y que alcancen los mismos objetivos pedagógicos en el mismo periodo en el que lo harían en una escuela ordinaria.

Cambiar esta situación significa realizar una completa reestructuración del sistema educativo, maestros, y la manera en la cual se imparte la enseñanza, ya que deben introducirse, nuevos métodos que permitan a todos los niños con capacidades especiales alcanzar las metas establecidas. La discapacidad no debería ser, por lo tanto, el eje de la problemática, sino que debería evaluarse las capacidades que sí poseen estos niños y encontrar la manera de adaptar el método educativo a estas necesidades.

A continuación, se presenta el desarrollo e implementación de un sistema que permita mejorar la enseñanza-aprendizaje de niños con discapacidades visuales. Se redacta el proceso que llevo a cabo este proyecto, la arquitectura, software y diseño.

1.3 Fundamentos Procesamiento de imágenes

1.3.1 Visión por computadora

Cosiste en la adquisición, procesamiento, análisis, clasificación y reconocimiento de imágenes digitales con el fin de producir información numérica o simbólica para que puedan ser tratados por un ordenador, busca automatizar las tareas que puede realizar el sistema visual humano (Díaz, 2012).

1.3.2 Imagen

Es una arreglo bidimensional de pixeles con diferente intensidad luminosa (Esqueda Elizondo, 2002).Una imagen puede ser representada por la ecuación 1.1:

$$r = f(x, y) \tag{1. 1}$$

Donde r representa la intensidad luminosa del píxel y cuyas coordenadas son: (x, y)

1.3.3 Píxel

Elemento básico de una imagen. la intensidad luminosa de cada píxel se representa por n bits (Esqueda Elizondo, 2002). Ver Figura 1.1.

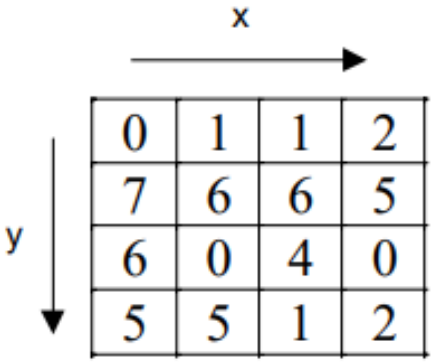


Figura 1. 1 Representación de bits
Fuente: Tomado de (Esqueda Elizondo, 2002)
Elaboración: Esqueda José.

1.3.4 Kernel

Un kernel o matriz de convolución es una matriz numérica que se usa para técnicas de procesamiento de imágenes como desenfoque, difuminado, erosión entre otros, esto se logra haciendo una convolución entre el kernel y una imagen (Ludwig, 2013). La expresión general de una convolución está dada por la ecuación 1.2:

$$g(x,y) = \omega * f(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} \omega(s,t) f(x-s, y-t) \quad (1.\ 2)$$

Donde $g(x,y)$ es la imagen filtrada, $f(x,y)$ es la imagen original, ω es el kernel del filtro. Cada elemento del kernel de filtro está considerado por: $-a \le s \le a$, $-b \le s \le b$.

Dependiendo de los valores de los elementos, un kernel puede causar una amplia gama de efectos, algunos de los más usados en este proyecto son:

Kernel 3x3: para procesos de desenfoque gaussiano, ecuación 1.3:

$$\omega = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \quad (1.\ 3)$$

Kernel 5x5: para procesos de desenfoque gaussiano, ecuación 1.4:

$$\omega = \frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix} \quad (1.\ 4)$$

Kernel normalizado: para procesos de normalización, ecuación 1.5:

$$\omega = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad (1.\ 5)$$

1.3.5 Color

Se forma mediante la combinación de los tres colores básicos rojo, azul y verde (RGB en inglés) la cual resulta de la combinación de tres señales de luminancia distinta. Un color específico es determinado por la cantidad de estos tres colores que se requiere para obtener el color deseado(Esqueda Elizondo, 2002). Ver Figura 1.2.

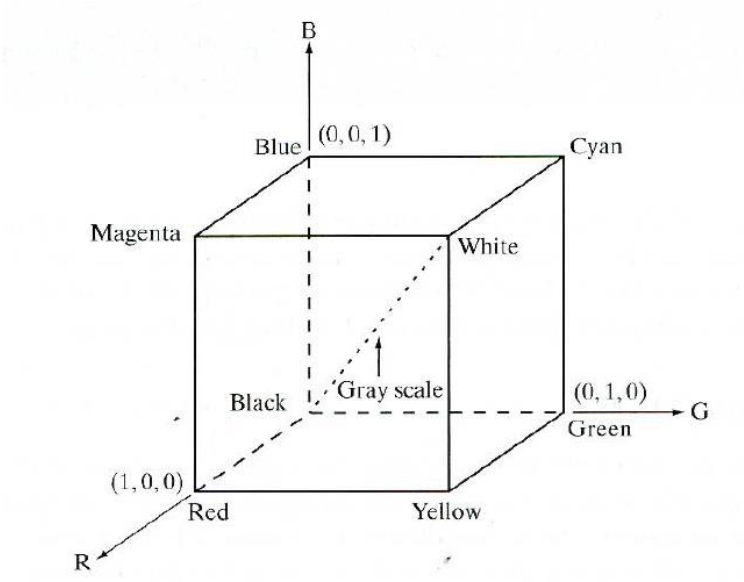


Figura 1. 2 Espacio de colores RGB.
Fuente: Tomado de (Esqueda Elizondo, 2002).
Elaboración: (Esqueda Elizondo, 2002).

1.3.6 Escala de grises

Resultado de distintos todos de gris, en imágenes de 8 bits, puede llegar a tener hasta 256 tonalidades, estos valores de bits están comprendido entre 255 y 0 (Esqueda Elizondo, 2002).

1.3.7 Ruido

Es definido como pixeles aislados que toman un nivel de gris diferente al de sus pixeles vecinos (Díaz, 2012). Todas las imágenes tienen cierta cantidad de ruido, debido a la cámara utilizada o al medio de transmisión de la señal (Esqueda Elizondo, 2002). El ruido puede clasificarse en los siguientes tipos:

1.3.7.1 Ruido Gaussiano

Este tipo de ruido produce pequeñas variaciones sobre la imagen; generalmente se debe a diferentes ganancias en la cámara, ruido en los digitalizadores, perturbaciones en la transmisión entre otros. Se considera que el valor final del píxel sería el ideal más una cantidad correspondiente al error que puede describirse como una variable aleatoria gaussiana (Esqueda Elizondo, 2002).

1.3.7.2 Ruido Impulsional

O también llamado ruido sal y pimienta, es un tipo de ruido en el cual el valor que toma el píxel no tiene relación con el valor ideal, sino con el valor del ruido que toma valores muy altos o bajos, puntos blancos y/o negros.(Esqueda Elizondo, 2002).

1.3.7.3 Ruido Multiplicativo

La imagen obtenida es el resultado de la multiplicación de dos señales (Esqueda Elizondo, 2002).

1.3.8 Filtrado

La acción de filtrado de una imagen se la realiza para lograr dos aspectos: realce de la imagen y realce de su morfología. Es una operación de vecindario, lo que quiere decir que el valor de un píxel dado en la imagen procesado se calcula mediante algoritmos que toman en cuenta valores de los pixeles de la vecindad de la imagen original (Esqueda Elizondo, 2002).

1.3.8.1 *Filtrado lineal espacia*l

Este tipo de filtrado opera sobre un grupo de pixeles dentro de una determinada área, los pixeles vecinos dan información sobre el brillo en el área que se va a tratar, es usada para atenuar o resaltar detalles dentro de una imagen (Esqueda Elizondo, 2002).

1.3.8.2 *Filtro pasa bajas espacial*

Disminuye la ganancia de los componentes de alta frecuencia para minimizar el ruido, con el objetivo de suavizar contrastes espaciales en la imagen (Esqueda Elizondo, 2002).

Tomando como ejemplo a (Esqueda Elizondo, 2002) el filtro más sencillo es aquel que tiene coeficientes unitarios en todos los elementos , tal como se observa en la Figura 1.3.

1	1	1
1	1	1
1	1	1

Figura 1. 3 Máscara de pixeles unitarios.
Fuente: Tomado de (Esqueda Elizondo, 2002).
Elaboración: Esqueda José.

Como se muestra en la Figura 1.4, si al resultado se le multiplica por un noveno, se obtiene la media de todos los píxeles, por lo que el ruido disminuye.

$1/9$	$1/9$	$1/9$
$1/9$	$1/9$	$1/9$
$1/9$	$1/9$	$1/9$

Figura 1. 4 Coeficientes de la máscara.
Fuente: Tomado de (Esqueda Elizondo, 2002)
Elaboración: Esqueda José.

Sin embargo, este filtro presupone que la influencia de todos los píxeles es igual. Otra consideración es que cuanto más alejado esté el píxel del central, su valor será menor y se obtiene la siguiente máscara, ver Figura 1.5:

1	2	1
2	4	2
1	2	1

Figura 1. 5 Pixeles centrales frente a pixeles vecinos.
Fuente: Tomado de (Esqueda Elizondo, 2002)
Elaboración: Esqueda José.

En general, se tiene (ver Figura 1.6):

1	b	1
b	b^2	b
1	b	1

Figura 1. 6 Ganancia de cada píxel.
Fuente: Tomado de (Esqueda Elizondo, 2002)
Elaboración: Esqueda José.

Al hacer un promedio de los valores vecinos aparece una difuminación de los bordes y otros detalles de contraste.

1.3.8.3 *Filtrado promedio*

El filtrado promedio elimina puntos blancos y negros (ruido tipo sal y pimienta) presentes en la imagen. Dada una imagen $f(x,y)$ de tamaño $N \times N$, el valor del nivel de gris de la imagen suavizada $g(x,y)$ en el punto (x,y) se obtiene promediando los valores de nivel de gris de los puntos f contenidos en una cierta vecindad de (x,y), correspondiente a la ecuación 1.6 (Esqueda Elizondo, 2002).

$$g(x,y) = \frac{1}{M} \sum_{(n,m) \in S} f(n,m) \tag{1. 6}$$

Donde $x, y = 0,1,...,N-1$. S es el conjunto de coordenadas de los puntos vecinos a (x,y) incluyendo el propio (x,y), y M es el número de puntos de la vecindad.

1.3.9 Difuminado

O desenfoque se utiliza para definir una escala de imagen para trabajar, para interpolación, para calcular puntos de interés y en muchas más aplicaciones. La imagen en escala de grises es convolucionada con un kernel gaussiano, como se observa en la ecuación 1.7, para crear una imagen difuminada (Solem, 2012).

$$I_\sigma = I * G_\sigma \tag{1. 7}$$

Donde el operador * indica la convolución y G_σ es un kernel gaussiano con la desviación estándar definida por la ecuación 1.8:

$$G_\sigma = \frac{1}{2\pi\sigma} e^{-(x^2+y^2)/2\sigma^2} \quad (1.\ 8)$$

1.3.10 Umbralización

Es el valor adecuado de un nivel de gris, a partir del cual los pixeles se clasifican en dos zonas: los que poseen un valor de gris mayor que el umbral y los pixeles que tienen un valor de gris menor que el umbral (Díaz, 2012).

1.3.11 Binarización

es una operación que permite transformar una imagen en tonos de gris a una imagen en blanco y negro solamente (Díaz, 2012).

1.3.11.1 *Binarización Otsu*

Creado por Nobuyuki Otsu en 1979, es un procedimiento de binarización que selecciona el umbral óptimo maximizando la varianza entre clases. La varianza entre clases se define como una suma ponderada de las varianzas. El método Otsu no precisa información previa de la imagen antes de su procesamiento, ni supervisión humana para el cálculo de los umbrales (Magro, 2013).

La probabilidad de ocurrencia del nivel de gris en una imagen según Otsu es la que se muestra en la ecuación 1.9:

$$P_i = \frac{f_i}{N} \quad (1.\ 9)$$

Donde: *N* son los pixeles cuyos niveles de gris se encuentran entre 1 y *L*. El número de píxeles con nivel gris *i* se denota como f_i.

Luego de determinar la probabilidad de ocurrencia de nivel de gris, es posible determinar la probabilidad de paso de un píxel a un nivel de umbral a través de la ecuación 1.10:

$$C_1 = \frac{p_t}{\omega_1(t)} \quad ; \quad C_2 = \frac{p_{t+i}}{\omega_2(t)} \quad (1.\ 10)$$

La distribución de probabilidad para los dos niveles de gris es equivalente a la ecuación 1.11, donde ω_1 representa los valores menores o iguales al umbral T y ω_2 representa los valores mayores.

$$\omega_1 = \sum_{i=0}^{T} p_i \quad ; \quad \omega_1 = \sum_{i=T+1}^{L=255} p_i \tag{1. 11}$$

Donde: $\omega_1 = (i \leq T)$; $\omega_2 = (i > T)$. Se determina la media existente en cada grupo. La ecuación 1.12 determina los valores medios de los grupos μ_1 y μ_2 .

$$\mu_1 = \sum_{i=0}^{T} \frac{ip_i}{\omega_1} \quad ; \quad \mu_2 = \sum_{i=T+1}^{L=255} \frac{ip_i}{\omega_2} \tag{1. 12}$$

Otsu definió la variancia entre clases de una imagen umbralizada en la ecuación 1.13.

$$\sigma_1^2 = \sum_{0}^{T} \frac{(i-\mu_1)^2 p_i}{\omega_1} \quad ; \quad \sigma_2^2 = \sum_{T+1}^{255} \frac{(i-\mu_2)^2 p_i}{\omega_2} \tag{1. 13}$$

Una vez determinada se puede determinar la variancia total ψ de la imagen en la ecuación 1.14:

$$\psi = \sqrt{\omega_1 \sigma_1^2 + \omega_2 \sigma_2^2} \tag{1. 14}$$

Los píxeles se dividen en dos clases con diferentes niveles de gris respectivamente. La importancia de este método es que es automático y no se necesita ingresar ninguna información (Magro, 2013).

1.3.12 Erosión

Esta operación encoge los objetos oscuros o luminosos. Consiste en reemplazar el píxel con el menor valor (para objetos luminosos) o el mayor valor (para objetos oscuros) del vecindario (Díaz, 2012).

1.3.13 Dilatación

Se reemplaza el píxel con el mayor valor para objetos luminosos o el menor valor para objetos oscuros del vecindario. Esta operación ensancha los objetos oscuros o luminosos según sea la operación(Díaz, 2012).

1.3.14 Ecualización

Es una transformación que obtiene un histograma con una distribución uniforme para una imagen. Es decir, que exista el mismo número de pixeles para cada nivel de gris del histograma de una imagen (ver Figura 1.7), mejora así el contraste de la imagen (Bradski & Kaehler, 2008).

Figura 1. 7 Ecualización de una imagen.
Fuente: Tomado de (Bradski & Kaehler, 2008)
Elaboración: Bradski Gary, Kaehler Adrian.

En una imagen de escala de grises, donde n_j es el número de apariciones de nivel de gris *j*. La probabilidad de que ocurra un píxel de nivel *j* en la imagen es dada por la ecuación 1.15:

$$p_x(i) = p(x = i) = \frac{n_j}{n} \tag{1. 15}$$

En una imagen *MxN*, con n_k pixeles para cada nivel r_k , donde la intensidad de los pixeles va de 0 a L-1. L es el número de valores de intensidad posibles, T es una función acumulativa. El histograma de la imagen está presente en la ecuación 1.16:

$$S_k = T(r_k) = (L-1)\sum_{j=0}^{k} p_r(r_j) = \frac{L-1}{MN}\sum_{j=0}^{k} n_j \tag{1. 16}$$

1.3.15 Normalización

Es un proceso que cambia el rango de valores de intensidad de pixeles, con esta técnica la imagen original puede cubrir los 256 niveles de intensidad disponibles (Gonzalez & Woods, 2008). Se realiza de acuerdo con la ecuación 1.17:

$$N_{x,y} = \frac{N_{\max} - N_{\min}}{O_{\max} - O_{\min}} * (O_{x,y} - O_{\min}) + N_{\min} \tag{1. 17}$$

Donde:

$O_{\max}$ = Brillo máximo de la imagen de entrada.

$O_{\min}$ = Brillo mínimo de la Imagen de entrada.

$N_{x,y}$ = Nueva imagen.

$N_{\max}$ = Brillo máximo de la nueva imagen

$N_{\min}$ = Brillo mínimo de la nueva imagen

$O_{x,y}$ =imagen de entrada

1.3.16 Contornos y Segmentación

Segmentación es el proceso de dividir una imagen en regiones significativas u objetos de interés (Díaz, 2012).

Un contorno se define por un cambio en el nivel de gris, un operador que sea sensible a este cambio operara como un detector de contorno, lo cual la realiza un operador de derivada, como se presenta en la ecuación 1.18. La detección de contorno es parte de un proceso de aislamiento o segmentación, consiste en la identificación de objetos dentro de una imagen. (University of Toronto, 2019).

$$C = \sqrt{\left(\frac{dA}{dx}\right)^2 + \left(\frac{dA}{dy}\right)^2} \quad (1.\ 18)$$

1.3.17 Reconocimiento óptico de caracteres (OCR)

Tal y como lo define (Navarro, 2016) el OCR (optical caracter recognition), consiste en la identificación automatizada de símbolos o caracteres pertenecientes a un determinado alfabeto, a partir de una imagen recogida mediante la lectura óptica de un texto grabado en un apoyo real.

1.4 Recursos de Software

1.4.1 Python

Es un lenguaje de programación de alto nivel creado a finales de la década de los ochenta por Guido van Rossum. Sus estructuras de datos integradas de alto nivel lo hacen óptimo para el desarrollo de aplicaciones, así como para conectar componentes existentes entre sí. Posee una licencia de código abierto, capaz de analizar y ejecutar otros programas, usa tipado dinámico, soporta orientación a objetos, y además es multiplataforma. Posee una variedad de librerías capaces de dotar a este lenguaje de programación de multitud de funciones adicionales(Python, 2019a).

1.4.2 OpenCV

Open Source Computer Vision Library es una biblioteca de visión por computadora y de aprendizaje automático de código abierto. Fue construido para proporcionar una infraestructura común para las aplicaciones de visión artificial. Tiene interfaces C++, Python,

Java y MATLAB y es compatible con Windows, Linux, Android y Mac OS. OpenCV se inclina principalmente hacia aplicaciones de visión en tiempo real (OpenCV, 2019).

1.4.3 Tesseract

Tesseract es un motor de reconocimiento óptico de caracteres, fue desarrollado originalmente en Hewlett-Packard Laboratories Bristol y en Hewlett-Packard Co, Greeley Colorado entre 1985 y 1994. En 2005 Tesseract libero sus recursos. Y desde 2006 continúa siendo desarrollado por Google. Posee soporte para Unicode (UTF-8) y puede reconocer más de 100 idiomas. Su última versión es la 4.0.0 que agrega un nuevo motor de OCR basado en redes neuronales. Admite varios formatos de salida: texto plano, HTML, PDF, TSV, XML. Puede ser entrenado para reconocer otros idiomas (Tesseract, 2019).

1.4.4 Pyttsx3

Python Text to Speech (Pyttsx3) es una librería de Python que convierte una entrada de texto en voz hablada, con ayuda del motor “espeak” el cual es un sintetizador de voz. Disponible en sistemas operativos como Windows, Mac Os y Linux. Es capaz de leer cualquier cadena de caracteres de texto para formar oraciones compuestas. Su principal ventaja es que prescinde de conexión a internet para realizar todas sus conversiones. Disponible en español. (Python, 2019b).

1.5 Descripción de Juego educativo

El prototipo de juego educativo se basa en el trabajo inicial de (Guajala Michay, 2016) el cual elabora una propuesta de material didáctico que se adapte a niños con discapacidad visual en nivel de escolaridad inicial (ver Figura 1.8), y que a su vez sea usado por un amplio espectro de público objetivo. En su posterior etapa de desarrollo de la propuesta, los autores (Calderon-Cordova, Guajala-Michay, Barba-Guaman, & Quezada-Sarmiento, 2016) y (Lanchi, 2018) investigaron acerca del uso de juegos y juguetes como estrategia pedagógica con el fin de propiciar el desarrollo psicomotor, lógico-matemático, y sensorial de los niños.

El juego educativo compuesto de un tablero en el que se coloca en la parte superior la información necesaria, basado en un sistema numérico que van desde las unidades, decenas, centenas y unidades de mil, que permite realizar las tres operaciones básicas: suma, resta y multiplicación. Contiene espacios en los que se insertan las cantidades correspondientes y el signo según las operaciones a realizar, estos fueron colocados en el centro del tablero. Se identifica cada ficha con código Braille para orientar a los niños con discapacidad visual, y del mismo modo ha sido grabada en relieve los números arábigos.

El objetivo del juego es que el niño pueda:

- Familiarizarse con el código Braille.
- Reconocer números de hasta 4 cifras.
- Realizar operaciones básicas (+, -, x).
- Mejorar su desarrollo mental y psicomotriz.

Figura 1. 8 Prototipo juego educativo.
Fuente: (Calderon-Cordova, Guajala-Michay, Barba-Guaman, & Quezada-Sarmiento, 2016)
Elaboración: (Calderon-Cordova, Guajala-Michay, Barba-Guaman, & Quezada-Sarmiento, 2016)

CAPÍTULO II: DISEÑO Y DESARROLLO DE ALGORITMOS

2 DISEÑO Y DESARROLLO DE ALGORITMOS

2.1 Arquitectura software del prototipo

Se presenta en detalle los recursos de software para la realización del programa, de la misma manera se muestra un diagrama de flujo de su arquitectura (ver Figura 2.1). Estas funciones son el núcleo del proyecto y en el cual se basa toda la lógica de programación.

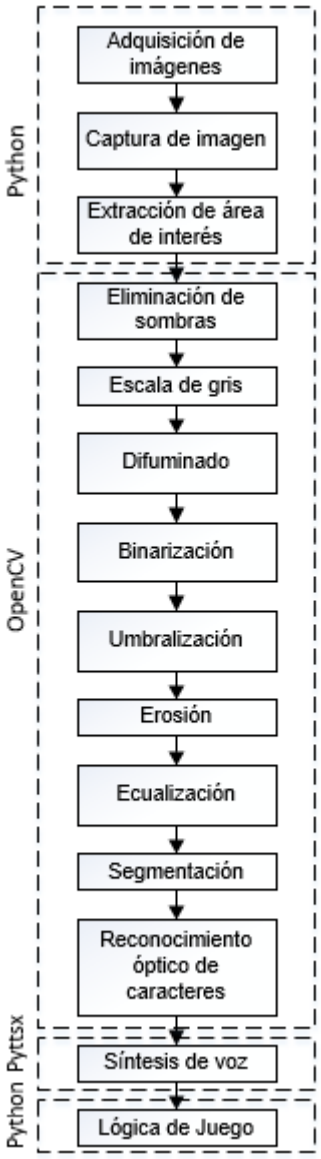


Figura 2. 1 Diagrama de flujo de software del prototipo.
Fuente: Autor.
Elaboración: Autor.

Se ha escogido trabajar con Python como lenguaje de programación ya que trabaja con software libre y es de amplio uso en aplicaciones de visión por computadora. Del mismo modo, se empleó las librerías OpenCV, Tesseract, Numpy y Pyttsx, ver tabla 1.

Tabla 1 Programas y librerías.

Nombre	Versión
Python	3.5.0
OpenCV	3.4.3
Numpy	1.12.1
Tesseract	0.2.6
Pyttsx3	2.71

Fuente: Autor.
Elaboración: Autor.

2.1.1 Adquisición y captura de imagen

Se lee la entrada de video y se convierte en una matriz, de esta manera se puede manejar una amplia gama de formatos de imagen. Se coloca un número entero, este es el identificador del dispositivo de video, luego se redimensiona para de esta manera la ventana que se muestra en pantalla sea siempre del mismo tamaño. El video captado se muestra en la Figura 2.2, a la espera del comando para capturar la imagen que se realiza por medio de entrada de teclado.

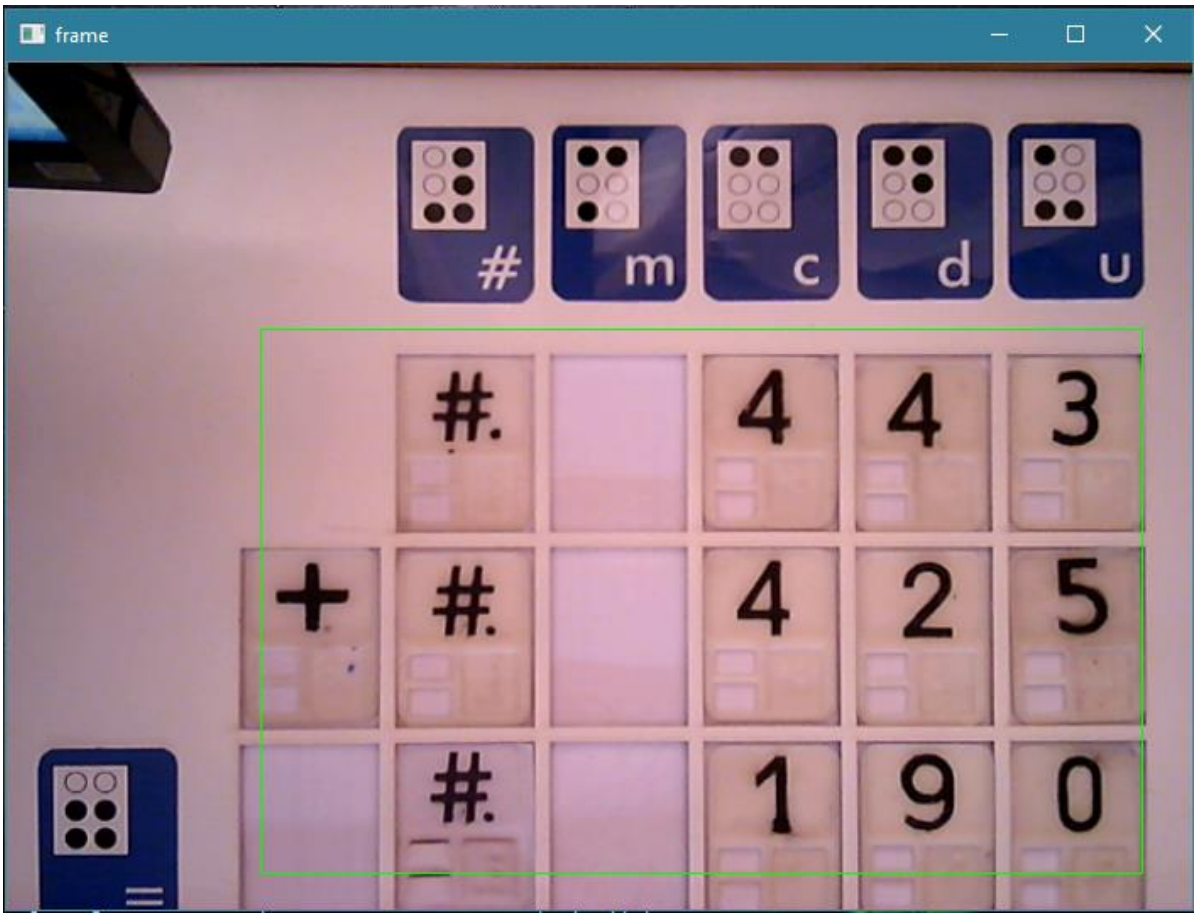


Figura 2. 2 Adquisición de video.
Fuente: Autor.
Elaboración: Autor.

2.1.2 Extracción de área de interés

El tablero es dividido en 18 celdas, distribuidas en 6 columnas por 3 filas y cuyas dimensiones están dadas por (α, β), que representa el ancho y alto de cada celda (ver Figura 2.3), se determina los límites del tablero, definidos por X_0, X_6, Y_0, Y_3 estos son insertados mediante las coordenadas de la imagen en Python.

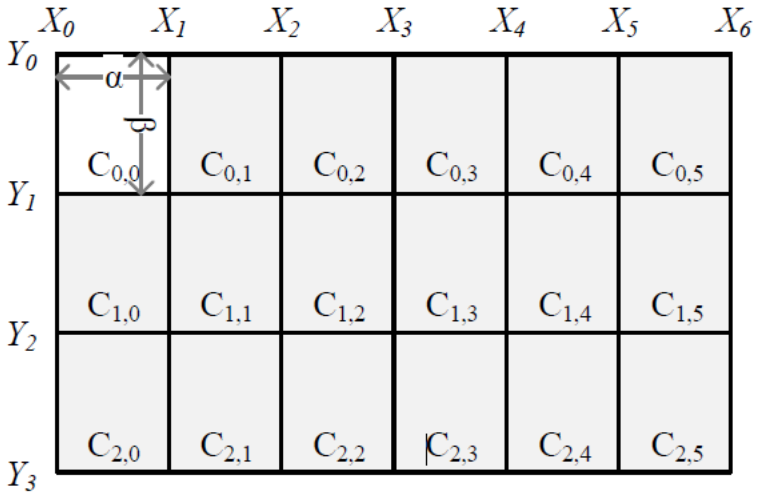


Figura 2. 3 Segmentación de áreas de interés de tablero.
Fuente: (Lanchi, 2018).
Elaboración: (Lanchi, 2018).

La región de cada celda está definida por la ecuación 2.1:

$$\alpha = \frac{X_0 - X_6}{6}\ ;\ \beta = \frac{Y_3 - Y_0}{3} \tag{2. 1}$$

Los límites de cada celda están basados en la ecuación 2.2:

$$X_i = X_0 + i\alpha\ ; \qquad Y_i = Y_0 + j\beta \tag{2. 2}$$

$$i = 0...6\ ; \qquad j = 0...3$$

La región de interés de cada celda está determinada por la ecuación 2.3:

$$ROI\_C_{j,i} = [X_i, Y_j, X_{i+1}, Y_{j+1}] \tag{2. 3}$$

Se determina el ROI numérico cuyas coordenadas están expresadas en (x, y) inferiores y (x, y) superiores, las celdas $C_{0,0}$ a $C_{0,5}$ se calculan a través de la ecuación 2.4, para las celdas $C_{1,0}$ a $C_{1,5}$ a través de ecuación 2.5 y para las celdas $C_{2,0}$ a $C_{2,5}$ mediante la ecuación 2.6.

$$ROI\_Nm_{0,i} = \left[X_i + \frac{\alpha}{10}, Y_0, X_i + \frac{9\alpha}{10}, Y_0 + \frac{3\beta}{5} \right] \quad (2.4)$$

$$ROI\_Nm_{1,i} = \left[X_i + \frac{\alpha}{10}, Y_1, X_i + \frac{9\alpha}{10}, Y_1 + \frac{3\beta}{5} \right] \quad (2.5)$$

$$ROI\_Nm_{2,i} = \left[X_i + \frac{\alpha}{10}, Y_2, X_i + \frac{9\alpha}{10}, Y_2 + \frac{3\beta}{5} \right] \quad (2.6)$$

Una vez obtenido estas tres regiones de interés, se puede proceder a las fases de filtrado y procesamiento de imágenes.

2.1.3 Eliminación de sombras

Debido a condiciones de intensidad, dirección de luz, así mismo el tablero y las fichas pueden generar sombras o regiones de oscuridad en la imagen a procesar (ver Figura 2.4), esto puede generar inconvenientes en etapas posteriores. Para ello se tiene una etapa previa la cual tiene por objeto la eliminación o reducción de estas sombras.

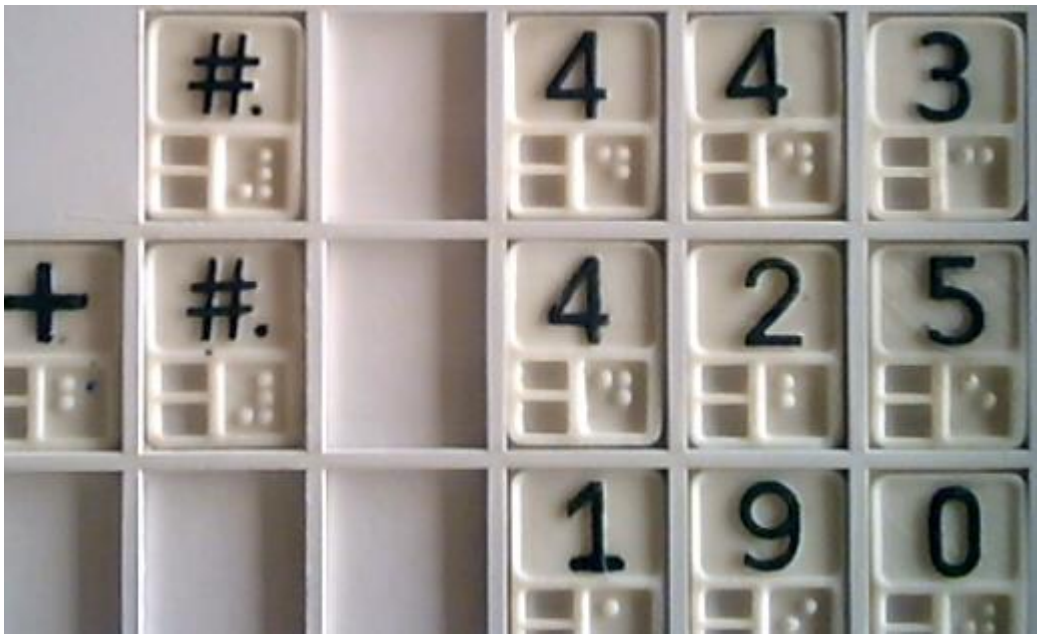


Figura 2. 4 Imagen de tablero y fichas con presencia de sombras.
Fuente: Autor.
Elaboración: Autor.

Se dilata la imagen para de esta manera preservar las regiones con más intensidad que son los números de las fichas ingresadas, ver Figura 2.5.

Figura 2. 5 Dilatación de imagen con sombras.
Fuente: Autor.
Elaboración: Autor.

Luego se aplica un desenfoque gaussiano con un kernel de dimensiones 3x3 para reducir el ruido de la imagen y reducir los detalles, esto resalta todas las sombras o decoloraciones presentes en la imagen, ver Figura 2.6.

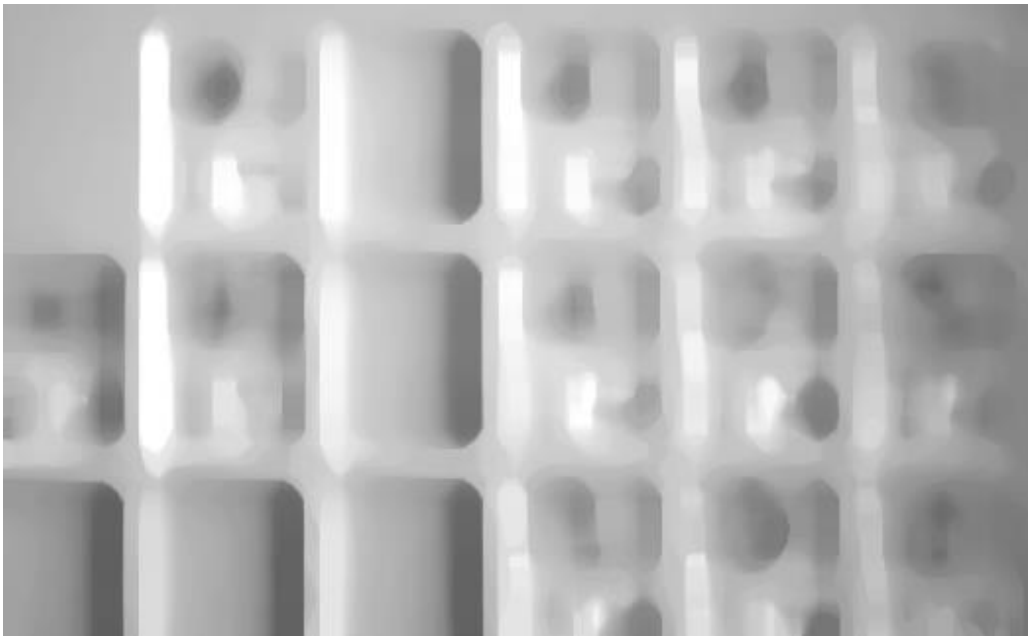


Figura 2. 6 Filtro de desenfoque gaussiano a una imagen sombreada y dilatada.
Fuente: Autor.
Elaboración: Autor.

Finalmente, se calcula la diferencia entre la imagen original y la imagen que se acaba de obtener en el paso anterior, y el resultado se normaliza, ver Figura 2.7.

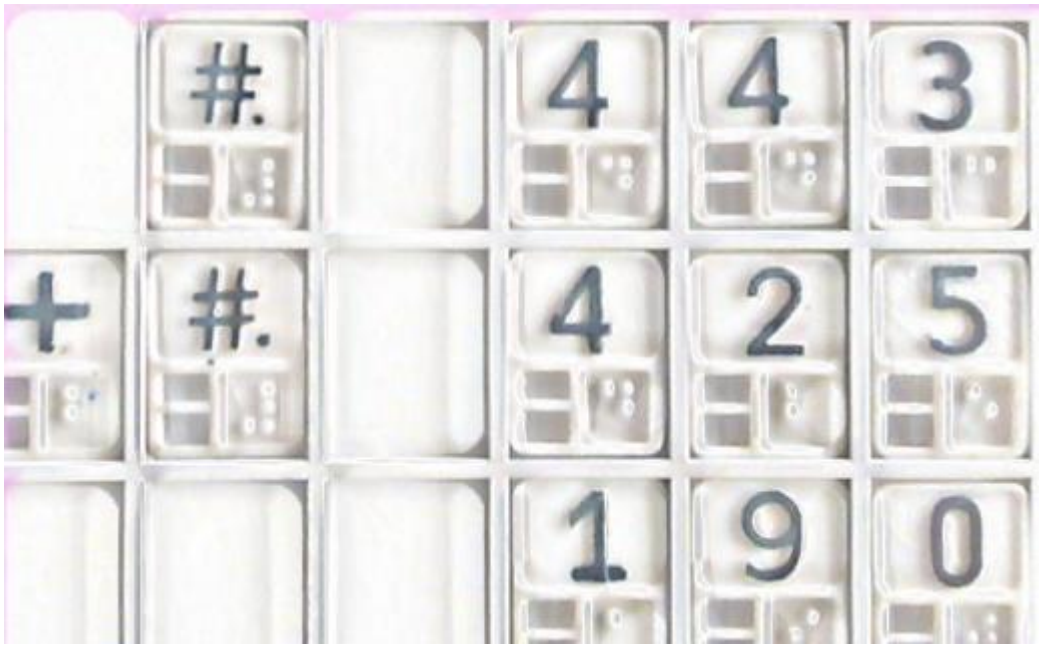


Figura 2. 7 Imagen filtrada y normalizada.
Fuente: Autor.
Elaboración: Autor.

2.1.4 Escala de Gris

La siguiente etapa es convertir la imagen resultante a escala de grises, OpenCV dispone de una función que transforma una imagen RGB a gris, ver Figura 2.8.

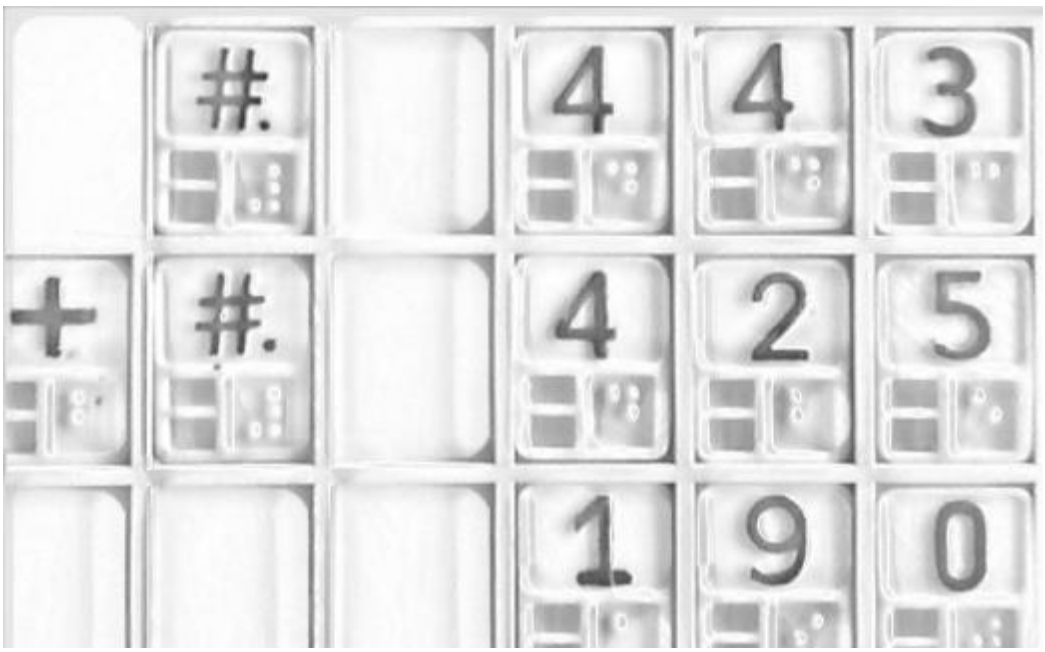


Figura 2. 8 Imagen a escala de grises.
Fuente: Autor.
Elaboración: Autor.

2.1.5 Difuminado

Luego de la etapa de difuminado medio, aplicando un kernel de 5x5, de esta manera logramos hacer realzar los caracteres ingresados y eliminar el ruido que contenga la imagen. Ver Figura 2.9.

Figura 2. 9 Difuminado de la imagen.
Fuente: Autor.
Elaboración: Autor.

2.1.6 Umbralización y Binarización

A la imagen previamente filtrada pasa por la etapa la binarización y el umbral de Otsu. Con esto se consigue la reducción de la escala de grises a dos únicos valores. Ver Figura 2.10.

Figura 2. 10 Resultado de la umbralización y binarización.
Fuente: Autor.
Elaboración: Autor.

2.1.7 Erosión

Se erosiona la imagen de esta manera se consigue que los caracteres tengan una contextura más fina y nítida, y se reduce los bordes presentes alrededor estos. Ver Figura 2.11.

Figura 2. 11 Resultado de la etapa de erosión.
Fuente: Autor.
Elaboración: Autor.

2.1.8 Ecualización

En la etapa de ecualización, la imagen resultante es la de una imagen más nítida, en blanco y negro, logrando de esta manera asemejarse a una hoja de papel impresa. Ver Figura 2.12.

Figura 2. 12 Resultado de la etapa de Ecualización.
Fuente: Autor.
Elaboración: Autor.

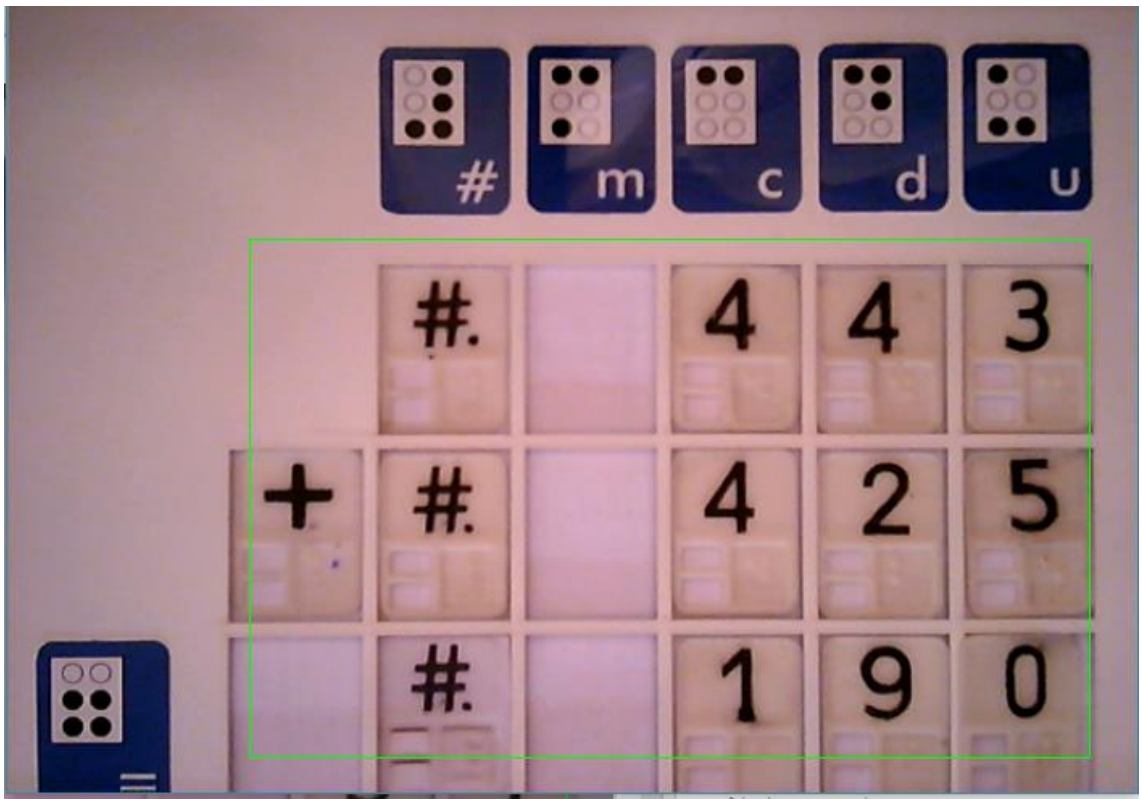
2.1.9 Contornos y Segmentación

Esta etapa permite realizar un recorte a la imagen para que de esta manera se ajuste a cada carácter y poder separar cada imagen y aplicar el reconocimiento de caracteres de manera individual. La segmentación se realiza mediante la detección de contornos y fijando un área mínima. Ver Figura 2.13.

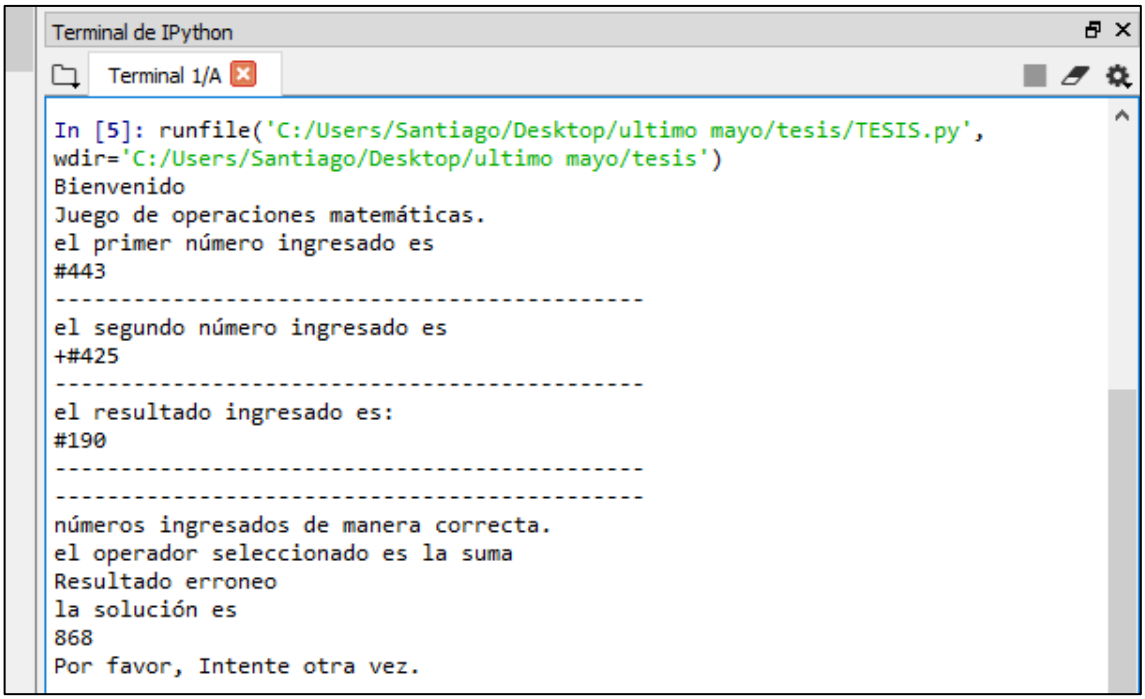
Figura 2. 13 Resultado de segmentación de carácter.
Fuente: Autor.
Elaboración: Autor

2.1.10 Reconocimiento Óptico de Caracteres

Una vez terminada la fase de filtrado y procesamiento, la imagen es apta para aplicar el reconocimiento de caracteres, para ello se utiliza la librería Tesseract. Como se presenta en la Figura 2.14a y 2.14b la salida del OCR se muestra en formato de texto en consola.



a)



b)

Figura 2. 14 a) Imagen de entrada. b) Reconocimiento Óptico de caracteres aplicado sobre imagen.
Fuente: Autor.
Elaboración: Autor.

Se ha modificado la configuración de la librería para adaptarla al reconocimiento de caracteres simples que se guardan durante la ejecución del programa. En la tabla 2, se detalla la configuración utilizada.

Tabla 2 Configuración de motor OCR Tesseract.

Lenguaje	Inglés (Por defecto)
Modo segmentación	Carácter Simple
Motor OCR	Basado en redes Neuronales
Salida	Texto (.txt)

Fuente: Autor.
Elaboración: Autor

2.1.11 Síntesis de Voz

Una vez realizado el reconocimiento de caracteres, la siguiente y última etapa es la de la síntesis de voz. Se hace uso de la librería Pyttsx. Replica mediante audio el estado del juego, los números ingresados y el operador seleccionado.

2.1.12 Lógica de juego

El juego valida operaciones matemáticas (suma, resta, multiplicación) de dos cantidades de hasta cuatro cifras y su resultado. El usuario debe ingresar los números anteponiendo el signo numeral “#”, así como el operador (+, -, x).

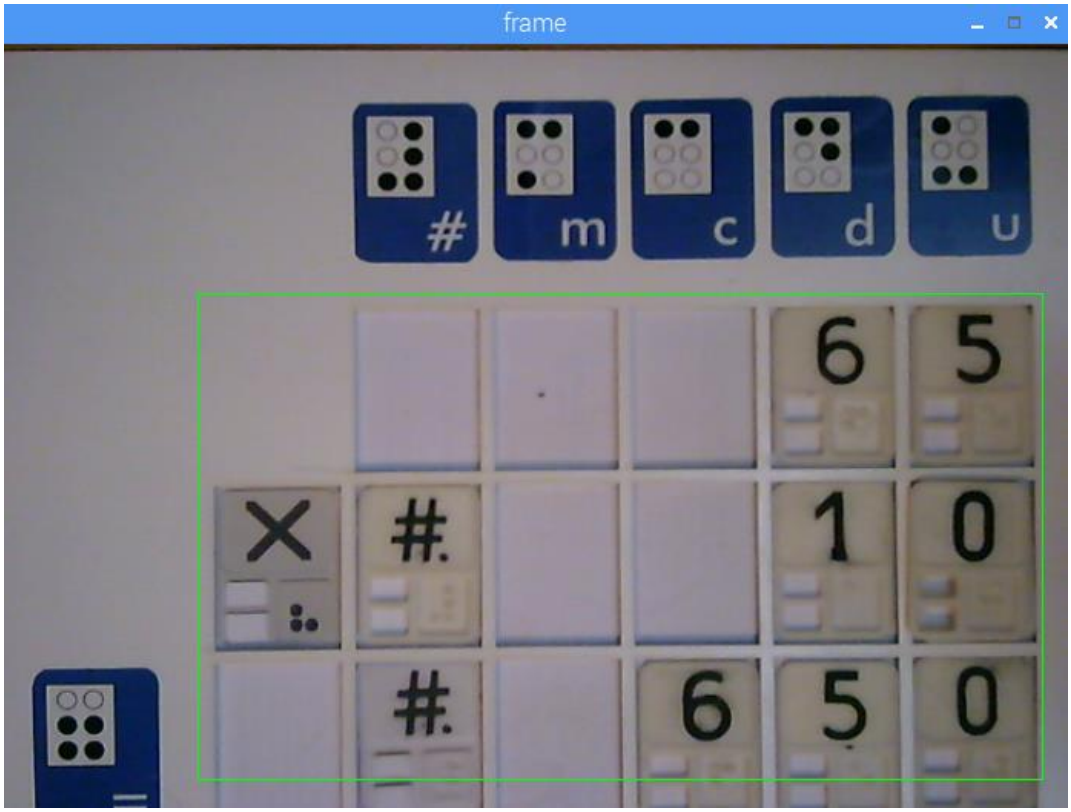
Una vez ingresados los números, el usuario pulsa el botón de enviar números y validar resultado, el cual indica los tres números ingresados, la operación seleccionada, y el resultado, mensajes y errores, a través de texto y voz. Los errores y mensajes se detallan en la tabla 3.

Tabla 3 Listado de mensajes y errores.

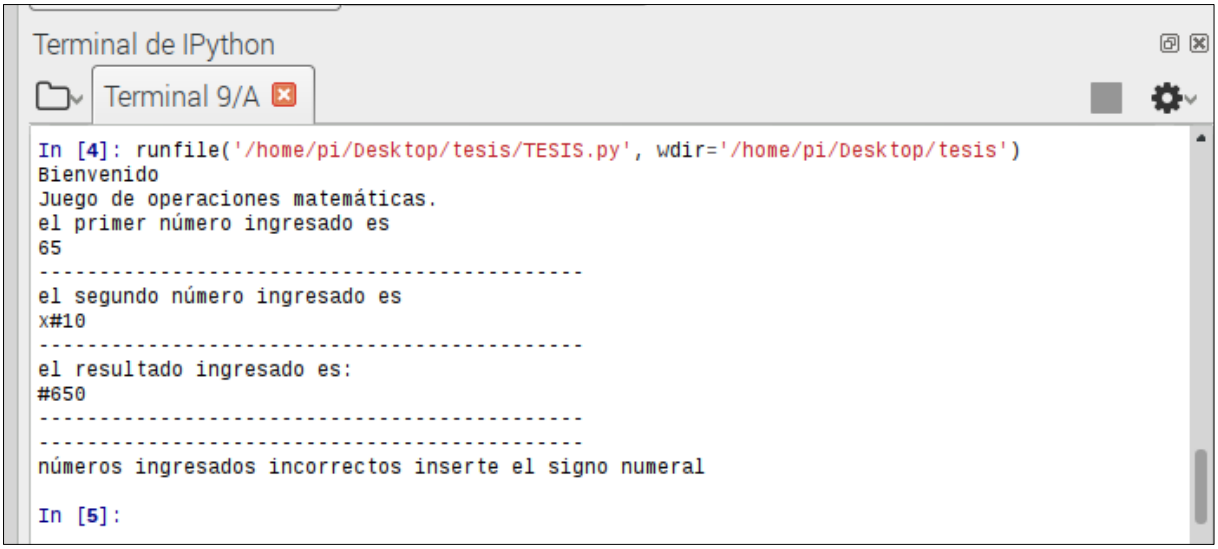
Tipo	**Detalle**
Error 1	Inserte signo numeral.
Error 2	Números ingresados de manera incorrecta.
Error 3	Resultado incorrecto. El resultado es: xxxx. Intentar otra vez.
Mensaje 1	Bienvenida
Mensaje 2	El primer número ingresado es: xxxx.
Mensaje 3	El segundo número ingresado es: xxxx.
Mensaje 4	El tercer número ingresado es: xxxx.
Mensaje 5	El operador seleccionado es: xxxx.
Mensaje 6	¡Felicitaciones! Respuesta correcta.

Fuente: (Calderon, Guajala, Lanchi, Barba-guaman, & Bermeo, 2018).
Elaboración: Autor.

En la Figura 2.15a se ingresan los números omitiendo el signo numeral en una de las cifras, el programa reconoce este error y lanza un mensaje en consola y en voz (ver Figura 2.15b), por lo cual el usuario deberá corregir el error e intentar nuevamente.



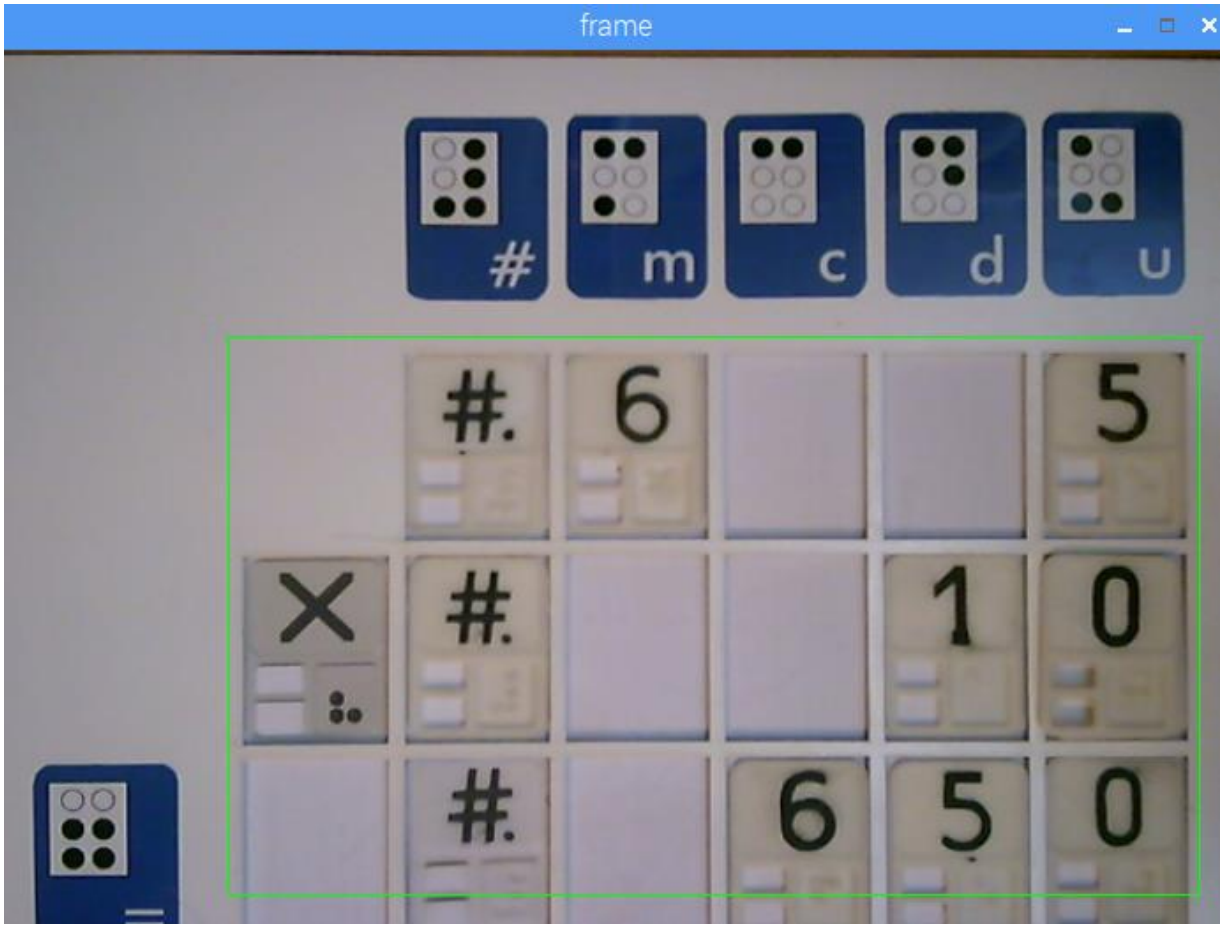
a



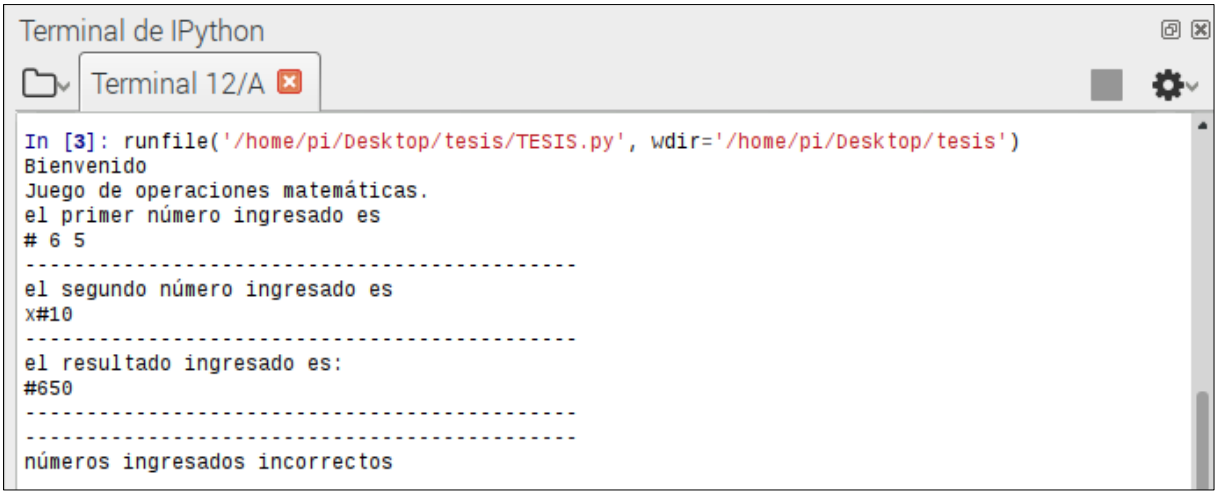
b

Figura 2. 15 a) Imagen de entrada. b) Detección de error.
Fuente: Autor.
Elaboración: Autor.

En la Figura 2.16a se ingresan, colocando de manera errónea la cifra, el programa reconoce este error y lanza un mensaje en consola y en voz (ver Figura 2.16b), por lo cual el usuario deberá corregir el error e intentar nuevamente.



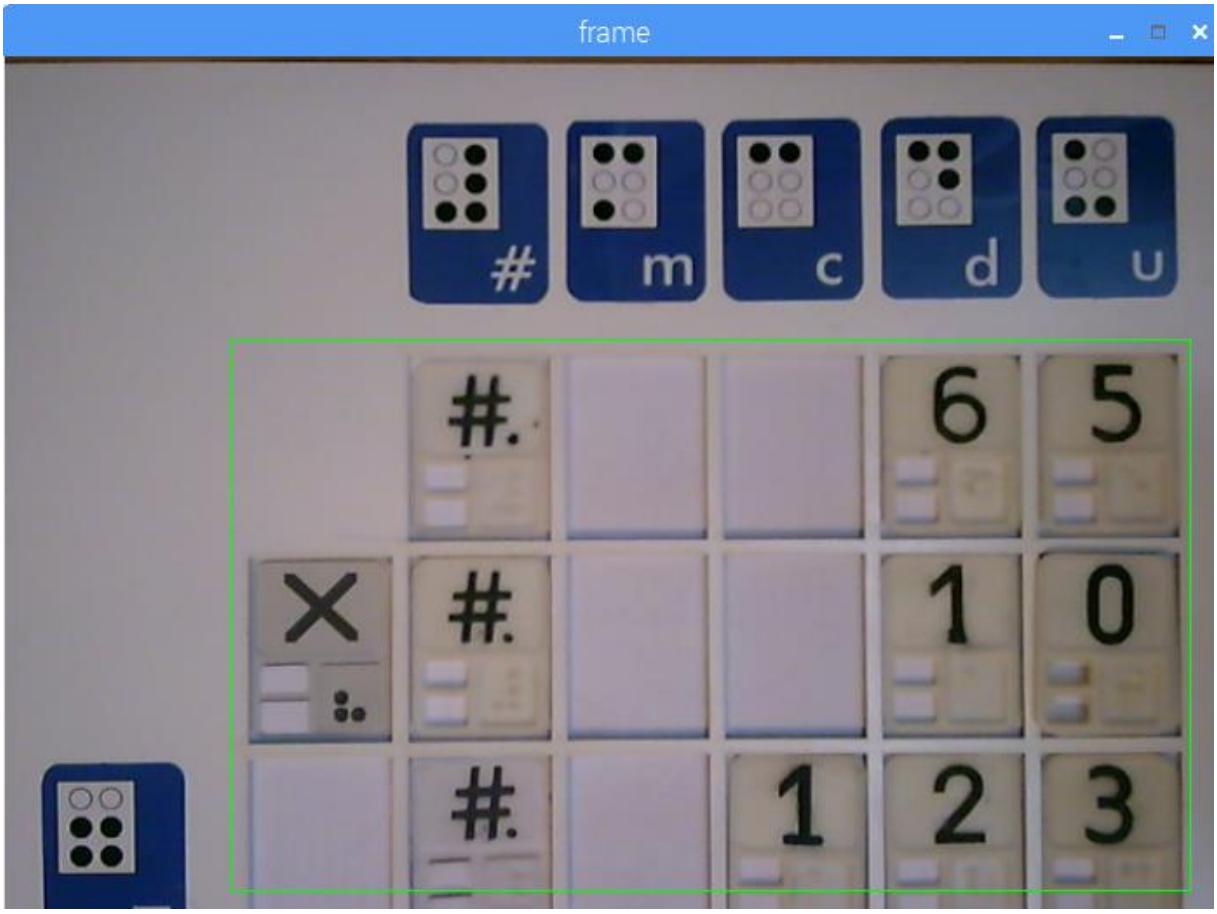
a



b

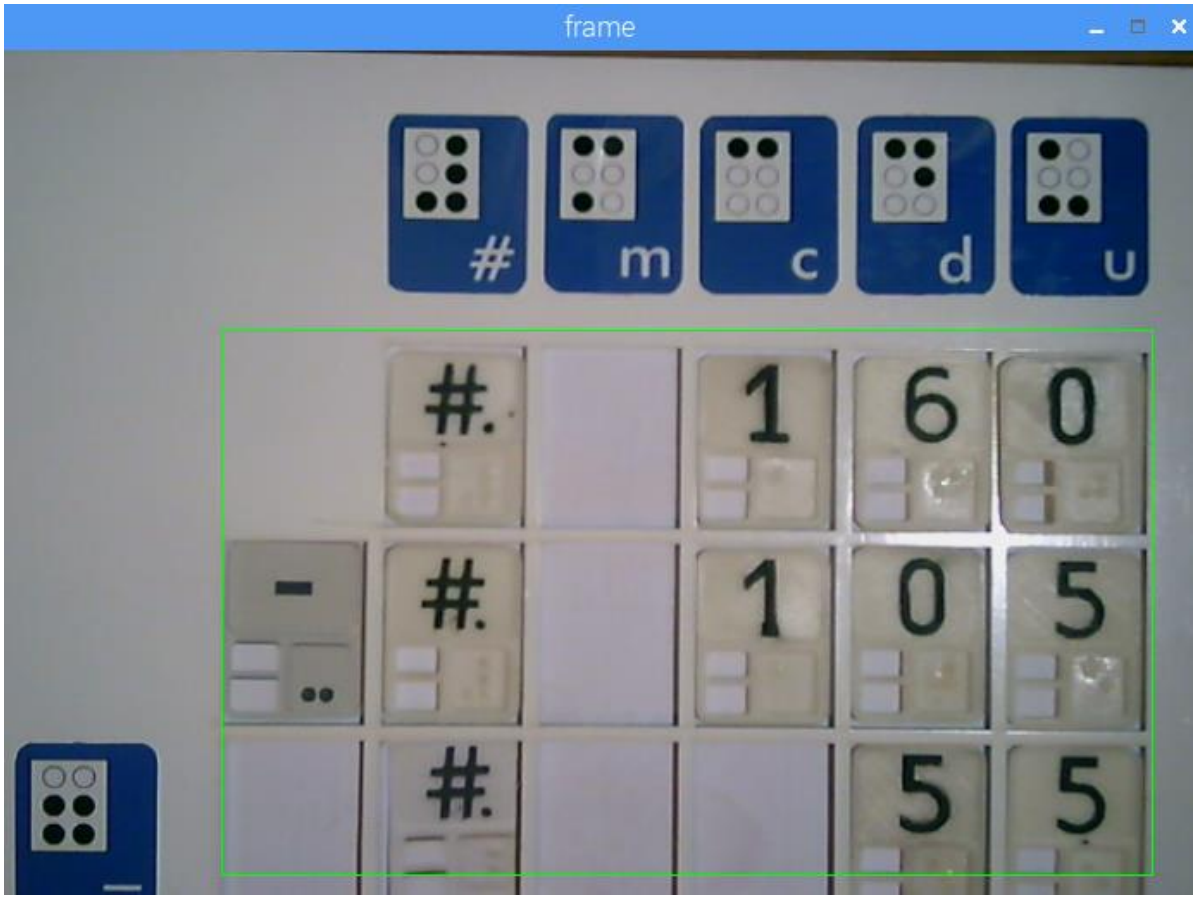
Figura 2. 16 a) Imagen de entrada. b) Detección de error.
Fuente: Autor.
Elaboración: Autor.

En la Figura 2.17a se ingresan los números de manera correcta, sin embargo, el resultado es erróneo, el programa reconoce este error y lanza un mensaje en consola y en voz (ver Figura 2.17b), por lo cual el usuario deberá corregir el error e intentar nuevamente.



a

b



a

Figura 2. 17 a) Imagen de entrada. b) Detección de error.
Fuente: Autor.
Elaboración: Autor.

En la Figura 2.18a se ingresan tres cifras de manera correcta, el programa muestra en consola que el resultado es el correcto y envía un mensaje de felicitación (ver Figura 2.18b).

b

Figura 2. 18 a) Imagen de entrada. b) Mensaje en consola.
Fuente: Autor.
Elaboración: Autor.

2.2 Arquitectura hardware del prototipo

En este apartado se detalla los elementos de hardware que serán implementados para el desarrollo y funcionamiento del prototipo. En la Figura 2.19 se presenta la arquitectura.

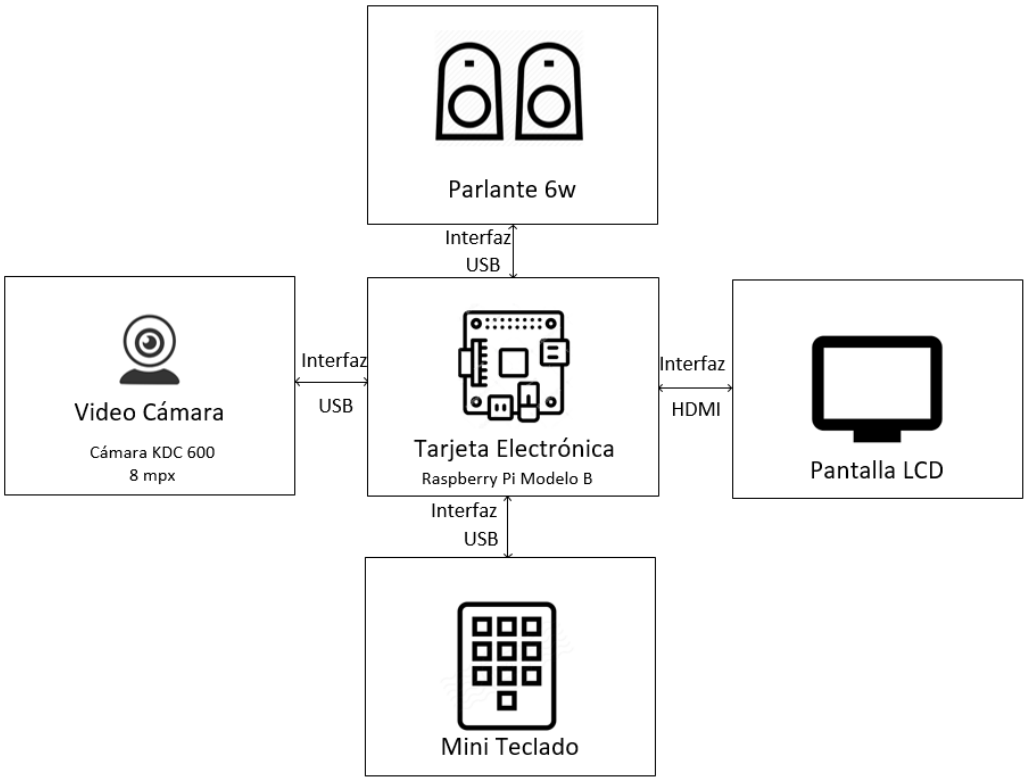


Figura 2. 19 Arquitectura hardware del prototipo.
Fuente: Autor.
Elaboración: Autor.

El prototipo físico consta de los siguientes elementos: un tablero fijo en el cual se evaluará el estado del juego, fichas con un valor numérico y alfanumérico que encajan en cada casilla dispuesto en el tablero, un soporte del cual está sujeto la cámara y el tablero, un teclado por el que el usuario envía instrucciones al dispositivo, un altavoz y pantalla.

2.2.1 Raspberry Pi

La tarjeta electrónica es el núcleo del proyecto ya que en este se ejecutarán el programa principal y se conectan todos periféricos de entrada y salida.

La elección de una tarjeta electrónica se hace en base a los requerimientos del programa, ya que debe soportar procesamiento de imágenes y video en tiempo real, así como conexión inalámbrica a otros dispositivos, este deberá ser compatible tanto con Python como con OpenCV.

Se toma en consideración tres modelos de tarjetas electrónicas: Odroid, Raspberry Pi, BeagleBone, de tres fabricantes diferentes. En la tabla 4 se detalla las características más relevantes de estos tres modelos escogidos previamente.

Tabla 4 Especificaciones técnicas de tarjetas electrónicas.

	odroid-c2	**Raspberry Pi 3B**	**BeagleBone Black**
CPU	Amlogic S905 SoC	Broadcom BCM2837	Sitara AM3359AZCZ100
	4 x ARM Cortex-A53 1.5GHz	4 x ARM Cortex-A53 1.2Ghz	1GHz, 2000 MIPS
	64bit ARMv8 Architecture @28nm	64bit ARMv7 Architecture @40nm	
GPU	3 x ARM Mali-450 MP 700MHz	1 x VideoCore IV 250MHz	SGX530 3D, 20M Polygons/S
RAM	2GB 32bit DDR3 912MHz	1GB 32bit LPDDR2 450MHz	512MB DDR3L 606MHZ
USB host	4 Ports	4Ports	2 ports
Ethernet/LAN	10 / 100 / 1000 Mbit/s	10 / 100 Mbit/s	10 / 100 mbit/s
Video Output	HDMI 2.0 4K / 60Hz	HDMI 1.4 / RCA / DSI	16b HDMI
Audio Output	HDMI / I2S	MDMI / 3.5mm Jack / I2S	HDMI Interface, Stereo
Camera Input	USB 720p	MIPI CSI 1080p	
IR Receiver	Yes (on-board IR sensor)		
IO Expansion	40 + 7 pin port	40 pin port	46 pin
	GPIO / UART / I2C / I2S / ADC	GPIO / UART / SPI / I2S	
ADC	10bit SAR 2 channels	no	
Heat sink	Included	opcional	
Wifi	USB IEEE 802.11b/g/n	si	usb host
Bluetooth	si	si	usb host
HDMI	si	si	si
Power supply	5 VDC 2A		5VDC 1A
Battery power supply	si		
microSD	si		1 slot

Fuente: Tomado de (BeagleBoard, 2019; Fundation Raspberry, 2019; ODROID, 2019).
Elaboración: Autor.

Luego de conocer las especificaciones, se opta por elegir la Raspberry Pi3, modelo B, ya que presenta las mejores características y mejor relación calidad/precio disponible, alrededor de 90 dólares, además es compatible tanto con Python como con OpenCV y soporta diferentes sistemas operativos. Ver Figura 2.20.

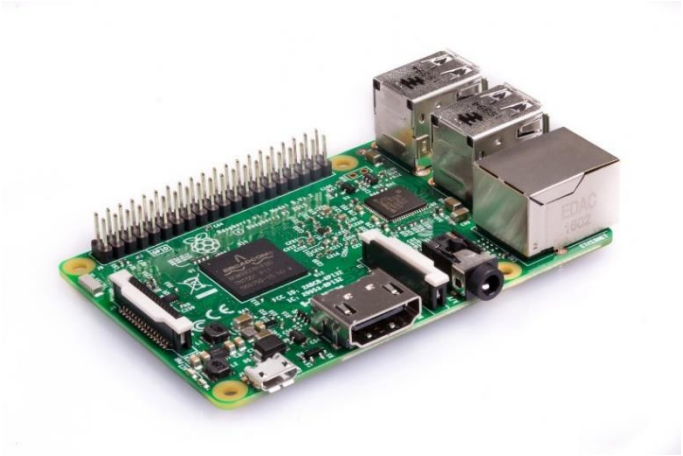


Figura 2. 20 Tarjeta electrónica Raspberry Pi Modelo B.
Fuente: Tomado de (Fundation Raspberry, 2019).
Elaboración: Autor.

2.2.2 Cámara

La cámara utilizada en el prototipo se muestra en la Figura 2.21, se trata de la cámara KDC (Voila HD), el uso de una cámara con buenas características disminuye el error en la etapa de reconocimiento de caracteres y mejora de esta manera el rendimiento del programa. La cámara presenta las siguientes características (ver Tabla 5).

Figura 2. 21 Cámara KDC 600.
Fuente: Tomado de (KlipXtreme, 2019).
Elaboración: Autor.

Tabla 5 Especificaciones de cámara KDC 600.

Resolución de video	1600x1200
Resolución de imagen	12 megapíxeles
Captura de imágenes	30 fps
Distancia focal	30mm ~ VGA
Formato de imagen	RGB24 y YUY2
Interfaz	USB 2.0
Configuración	Plug and play

Fuente: Tomado de (KlipXtreme, 2019)
Elaboración: Autor.

2.2.3 Tablero y Fichas

Para el diseño del tablero se toma como referencia la idea de diseño de tablero de (Lanchi, 2018). Pero se agregan las siguientes modificaciones:

- Le ancho del tablero es menor en el nuevo diseño, logrando así reducir las sombras y el contraste generado al momento de reconocer las fichas.
- Mejora el color del tablero, de esta manera al momento del filtrado no existirán diferencias de tonalidades entre las fichas y tablero.
- Mejora el soporte del tablero, logrando así más estabilidad para la cámara y se logra colocarla en un punto fijo.
- Elimina todo tipo de líneas o figuras que no sirvan al momento del procesamiento de la imagen, eliminando así ruido en el procesamiento.

Así, con estas modificaciones se crea un diseño más simple, pero que cumple la misma función practica que el de la anterior versión. Para el diseño de los indicadores de fichas y el color del tablero se hace uso del programa CorelDraw Ilustrador (ver Figura 2.22), y para las dimensiones del tablero y ranuras para las fichas se hace uso del programa Autocad (ver Figura 2.23) para su posterior corte.

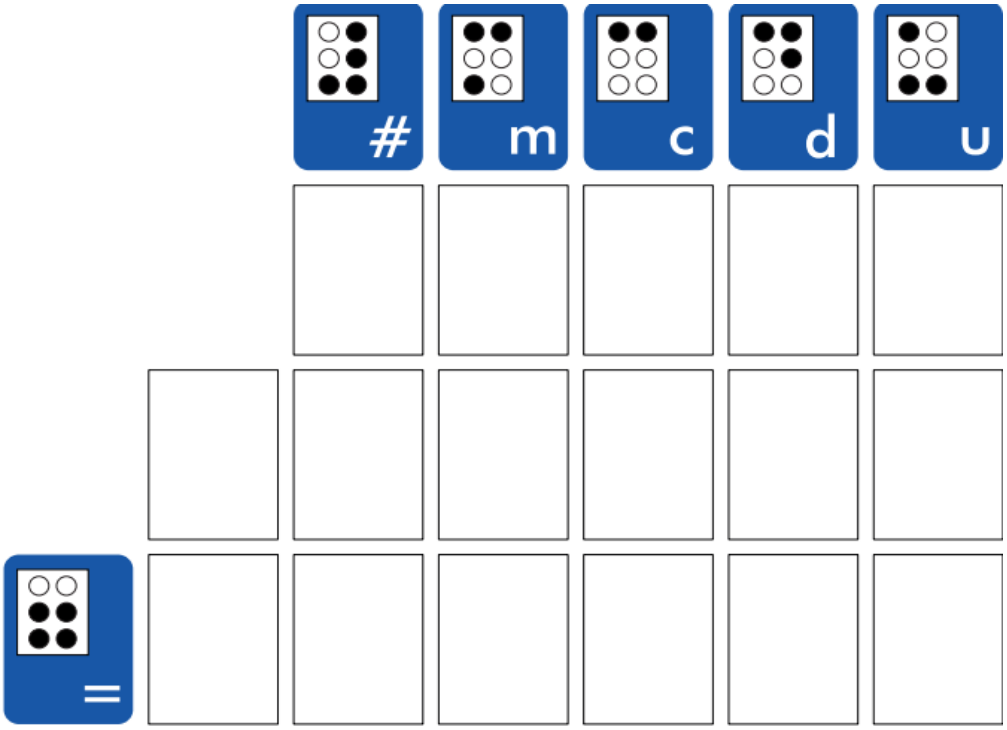


Figura 2. 22 Diseño del Tablero.
Fuente: Tomado de (Lanchi, 2018).
Elaboración: John Lanchi.

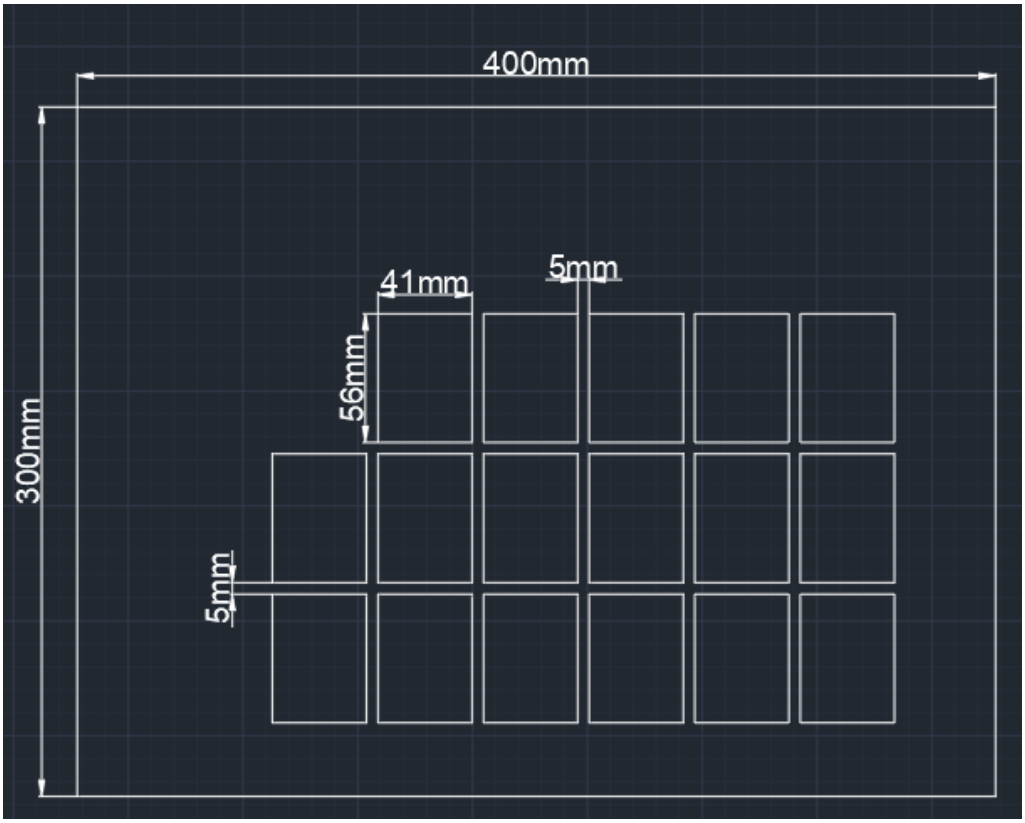


Figura 2. 23 Dimensiones del Tablero.
Fuente: Tomado de (Lanchi, 2018).
Elaboración: John Lanchi.

La fabricación de fichas fue tomado del trabajo de (Lanchi, 2018) que fueron probadas en aplicaciones anteriores. La fuente o tipografía utilizada en la elaboración de la ficha es la fuente de texto NI Vision, la Figura 2.24 ilustra cada ficha utilizada.

Esta fuente es reconocida por Tesseract por lo que no presenta inconvenientes en el rendimiento del programa. En la Figura 2.25 se muestra la idea del diseño que será aplicable para todos los caracteres tanto de números (0-9) como de caracteres alfanuméricos (+, -, x ,#). Para finalmente obtener el diseño en físico impreso en plástico mediante una impresora 3D como se muestra en la Figura 2.26.

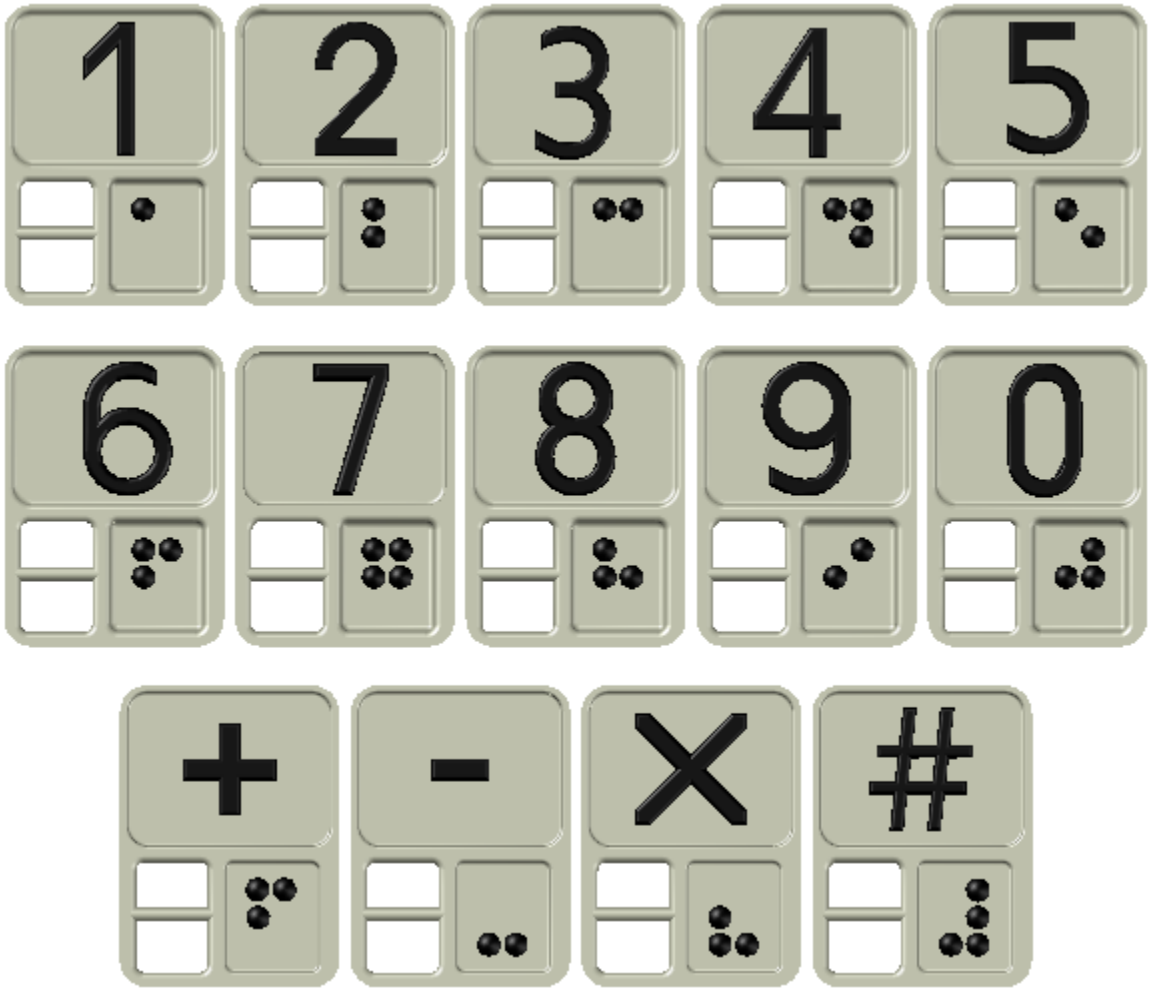


Figura 2. 24 Ilustración de Fichas.
Fuente: (Lanchi, 2018).
Elaboración: John Lanchi.

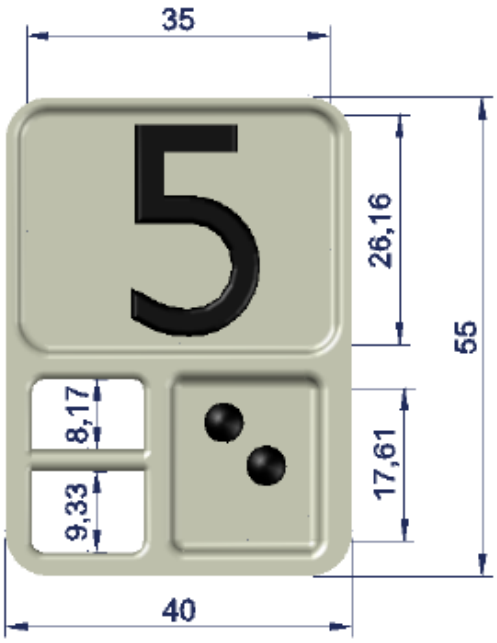


Figura 2. 25 Diseño de fichas.
Fuente: (Lanchi, 2018).
Elaboración: John Lanchi.

Figura 2. 26 Prototipo impreso de fichas.
Fuente: (Lanchi, 2018).
Elaboración: John Lanchi.

El material del soporte es de aluminio, y en él está fijado la cámara en dirección al tablero, presenta una altura total de 60 cm desde la base, lo cual facilita una imagen de todo el tablero. El material de la base es MDF (Tablero de fibra de densidad media) y brinda estabilidad y seguridad al prototipo. El prototipo final se presenta en la Figura 2.27.

Figura 2. 27 Prototipo Final.
Fuente: Autor.
Elaboración: Autor.

CAPÍTULO III: PRUEBAS Y RESULTADOS

3 Pruebas y resultados

3.1 Pruebas de desempeño del prototipo

Se evalúa el desempeño del dispositivo Raspberry Pi al momento de ejecutar el programa principal del prototipo.

3.1.1 Consumo de recursos del sistema embebido

Se realiza pruebas de rendimiento donde se observa el consumo de recursos de la tarjeta electrónica Raspberry Pi durante la ejecución del programa. Se muestra uso de memoria, CPU y RAM.

En la figura 3.1 muestra el porcentaje de memoria y CPU usados luego de ejecutar el programa en el dispositivo Raspberry. Se puede observar que, de un total de 927MB de memoria RAM, actualmente se usan 468MB, esto equivale al 50.48%. Y el uso del CPU alcanza un 48%. La ejecución del programa abarca la mitad de los recursos de la tarjeta, pero el programa se ejecuta con normalidad.

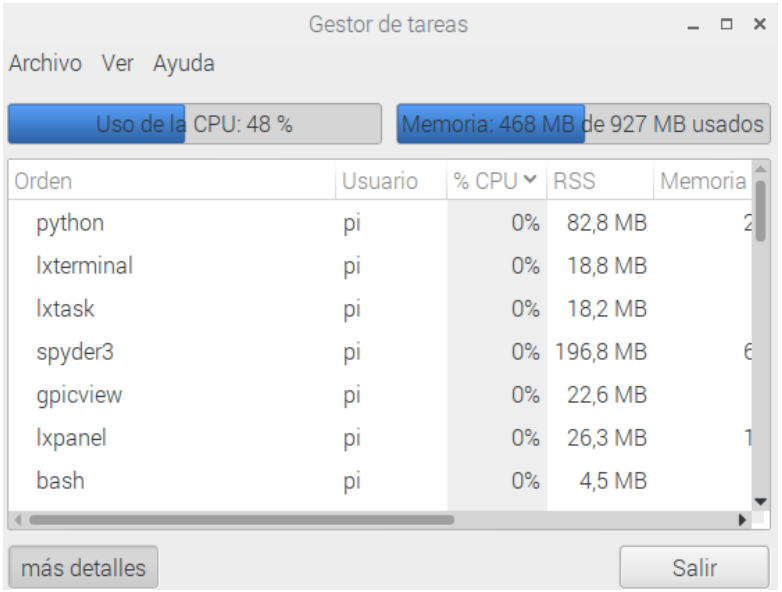


Figura 3. 1 Gestor de tareas y recursos de Raspberry.
Fuente: Autor.
Elaboración: Autor.

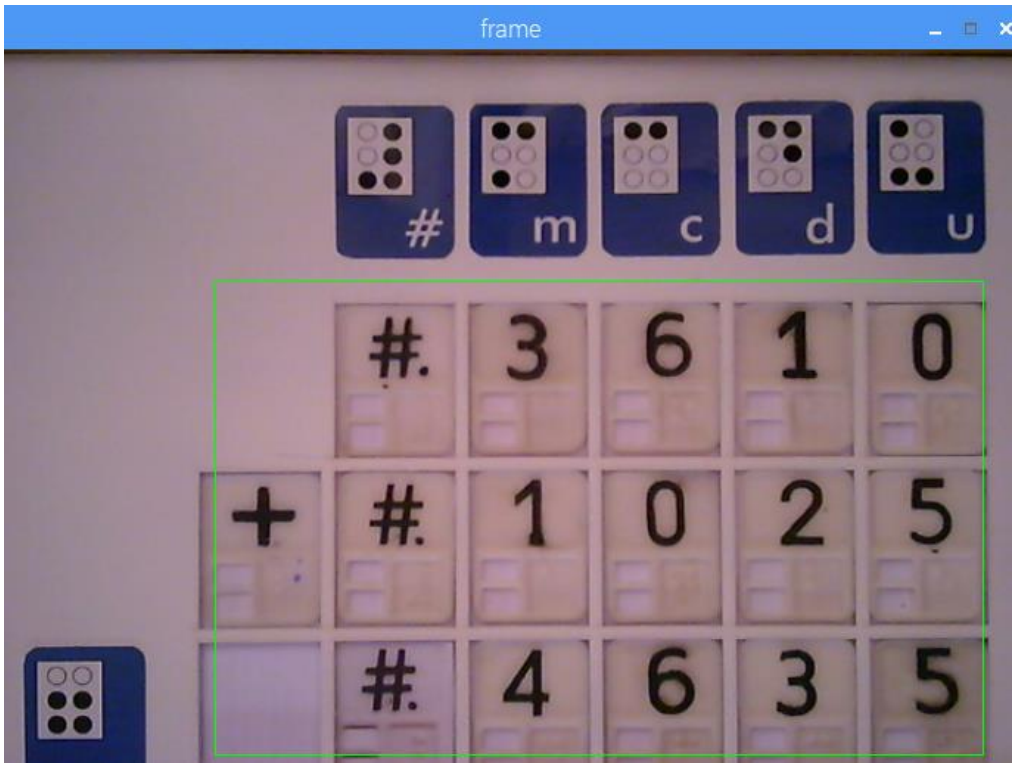
3.1.2 Pruebas de prototipo

3.1.2.1 Pruebas en diferentes ambientes luminiscentes

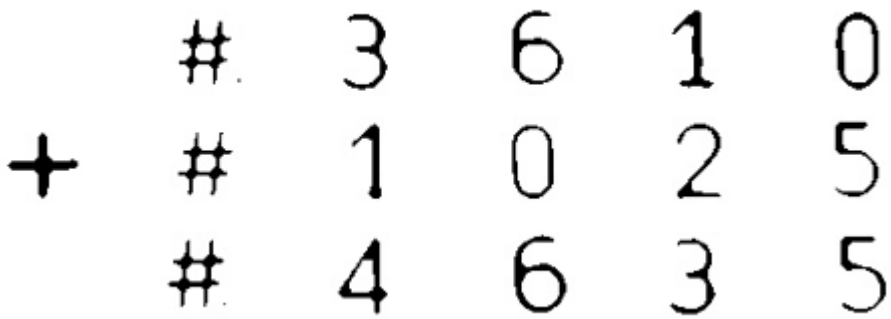
Se realiza las pruebas en diferentes ambientes controlados de iluminación, con la finalidad de validar la eficacia y funcionalidad del prototipo durante la ejecución del juego.

Los niveles de iluminación en las que se realizaron las pruebas fueron registrados usando la aplicación móvil “Lux Light Meter”[1] disponible en PlayStore de Google, la cual funciona utilizando el sensor de luz de un smartphone.

En la Figura 3.2 (a) se presenta la prueba en un ambiente de 26 luxes, en presencia de luz natural. En la Figura 3.2 (b) se muestra los numero ingresados previamente filtrados. En la Figura 3.2 (c) se presenta en consola: los números ingresados, el operador seleccionado y el resultado final del juego y en la Figura 3.2 (d) la cantidad de luz al momento de ejecutar el programa.

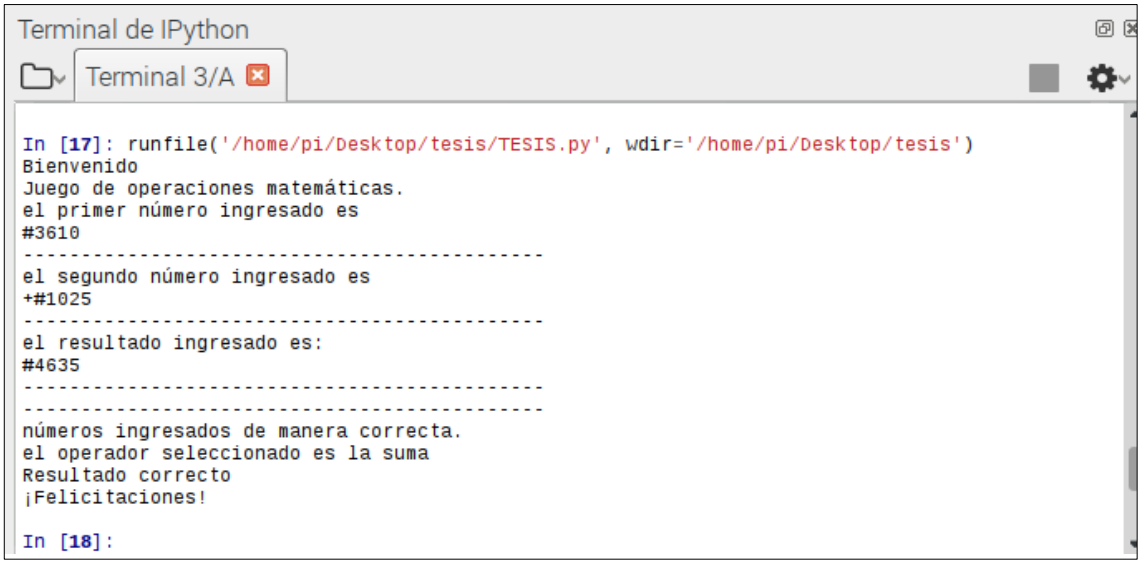


a

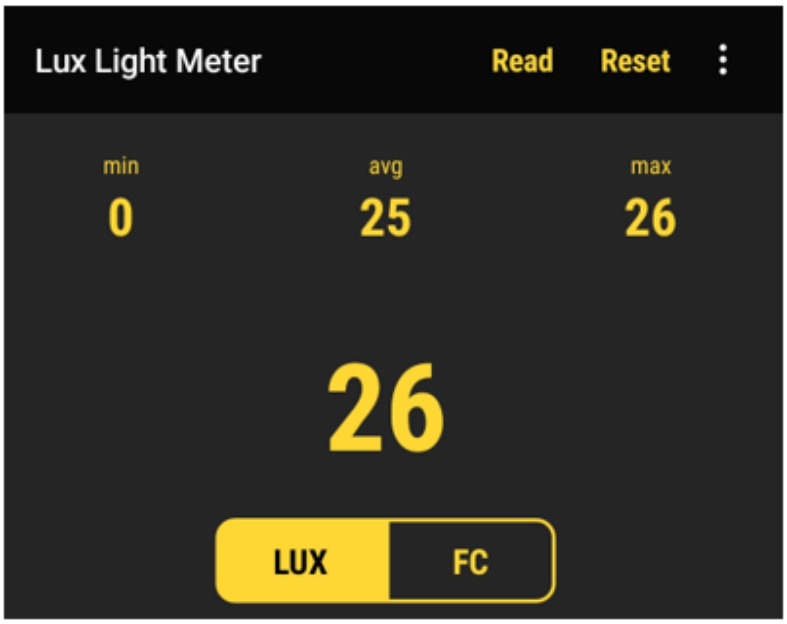


b

[1] Aplicación Lux Light Meter disponible para smartphones Android en Playstore: https://play.google.com/store/apps/details?id=com.doggoapps.luxlight&hl=es



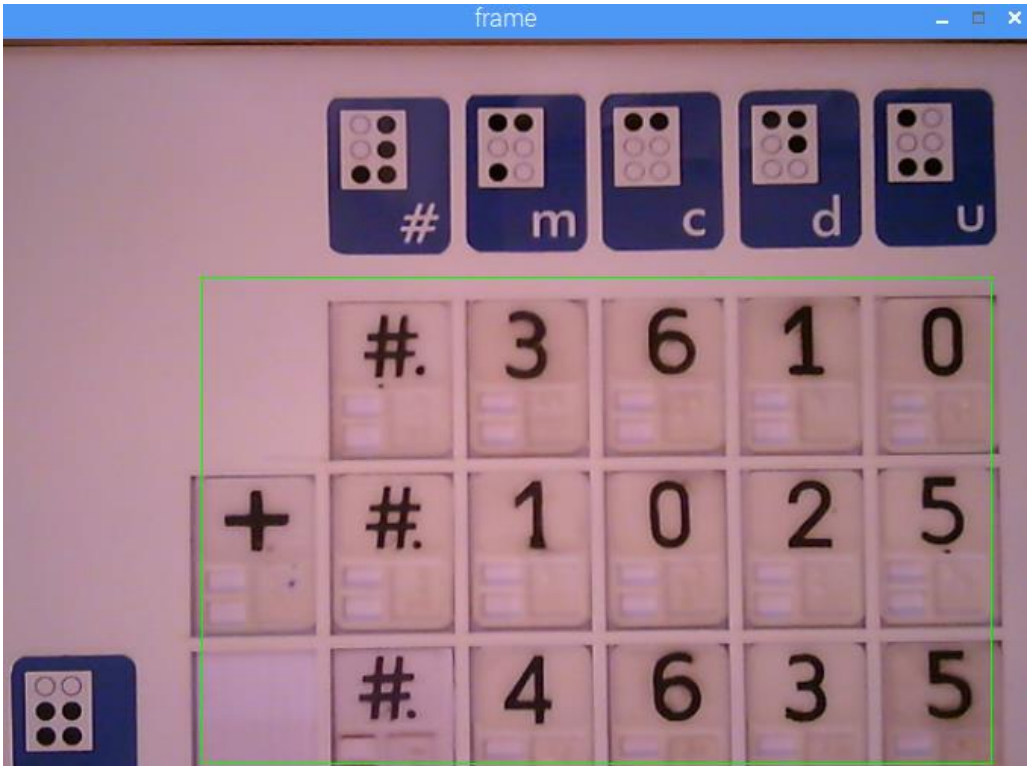
c



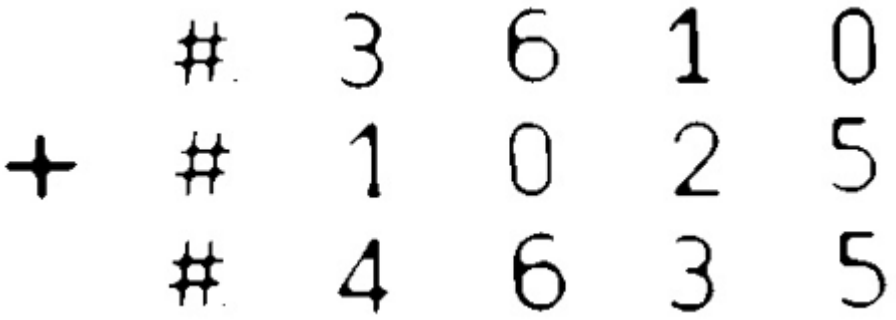
d

Figura 3. 2 a) Prueba de desempeño del prototipo, b) filtrado de números, c) resultado en consola, d) Medición de luz a 26 luxes.
Fuente: Autor.
Elaboración: Autor

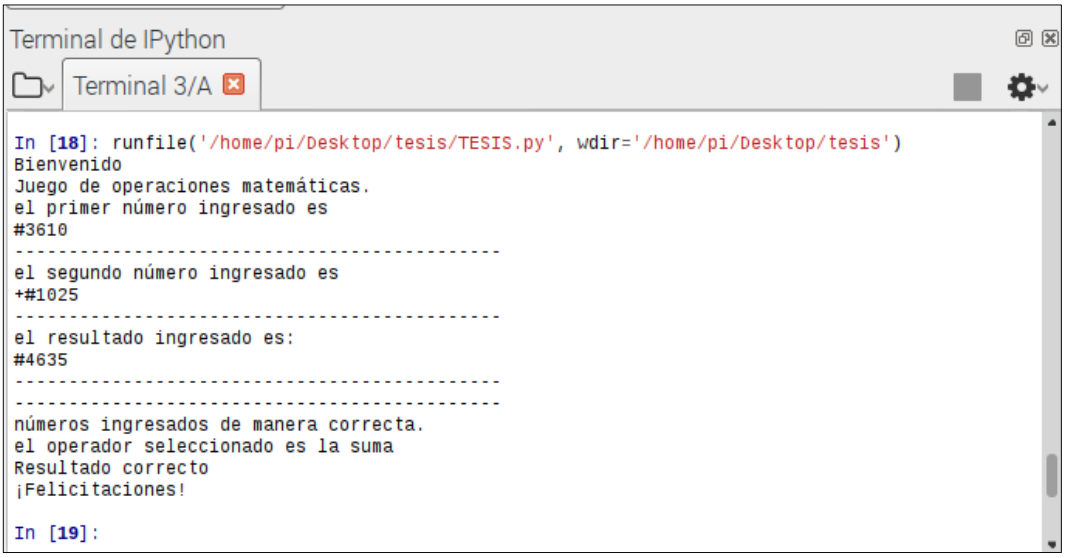
En la Figura 3.3 (a) se presenta la prueba en un ambiente de 37 luxes, en presencia de luz artificial amarilla. En la Figura 3.3 (b) se muestra los numero ingresados previamente filtrados. En la Figura 3.3 (c) se presenta en consola: los números ingresados, el operador seleccionado y el resultado final del juego y en la Figura 3.3 (d) la cantidad de luz al momento de ejecutar el programa.


frame
#
m
c
d
u
#. 3 6 1 0
+ # 1 0 2 5
#. 4 6 3 5

a

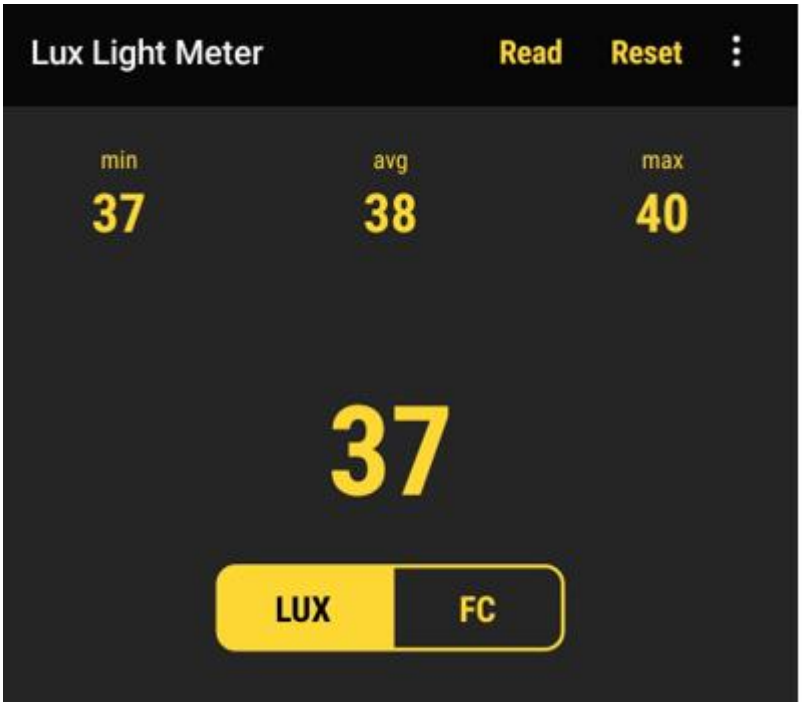

#. 3 6 1 0
+ # 1 0 2 5
#. 4 6 3 5

b


Terminal de IPython
Terminal 3/A
In [18]: runfile('/home/pi/Desktop/tesis/TESIS.py', wdir='/home/pi/Desktop/tesis')
Bienvenido
Juego de operaciones matemáticas.
el primer número ingresado es
#3610
---
el segundo número ingresado es
+#1025
---
el resultado ingresado es:
#4635

---
números ingresados de manera correcta.
el operador seleccionado es la suma
Resultado correcto
¡Felicitaciones!
In [19]:

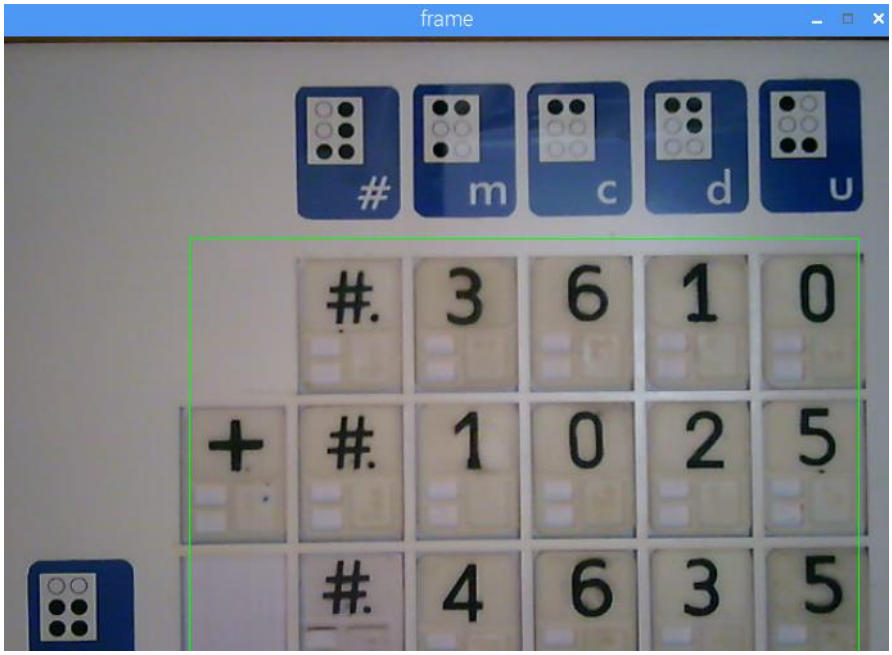
c



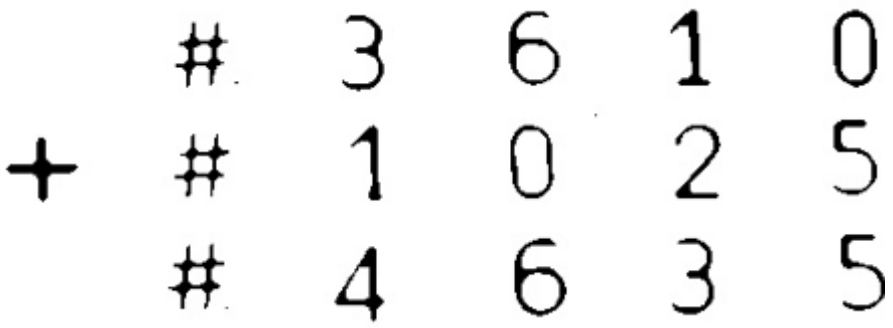
d

Figura 3. 3 a) Prueba de desempeño del prototipo, b) filtrado de números, c) resultado en consola, d) Medición de luz a 37 luxes.
Fuente: Autor.
Elaboración: Autor

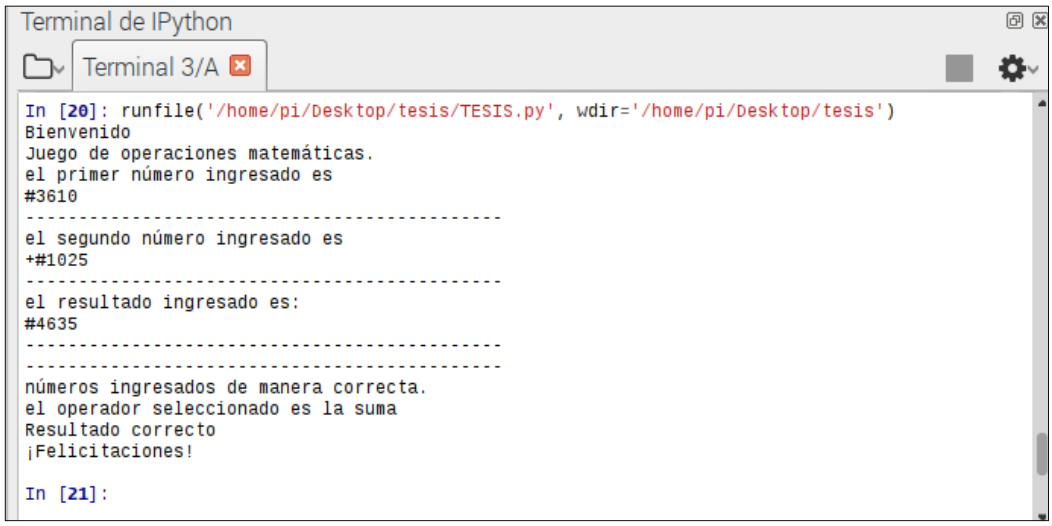
En la Figura 3.4 (a) se presenta la prueba en un ambiente de 94 luxes, en presencia de luz artificial blanca. En la Figura 3.4 (b) se muestra los numero ingresados previamente filtrados. En la Figura 3.4 (c) se presenta en consola: los números ingresados, el operador seleccionado y el resultado final del juego y en la Figura 3.4 (d) la cantidad de luz al momento de ejecutar el programa.



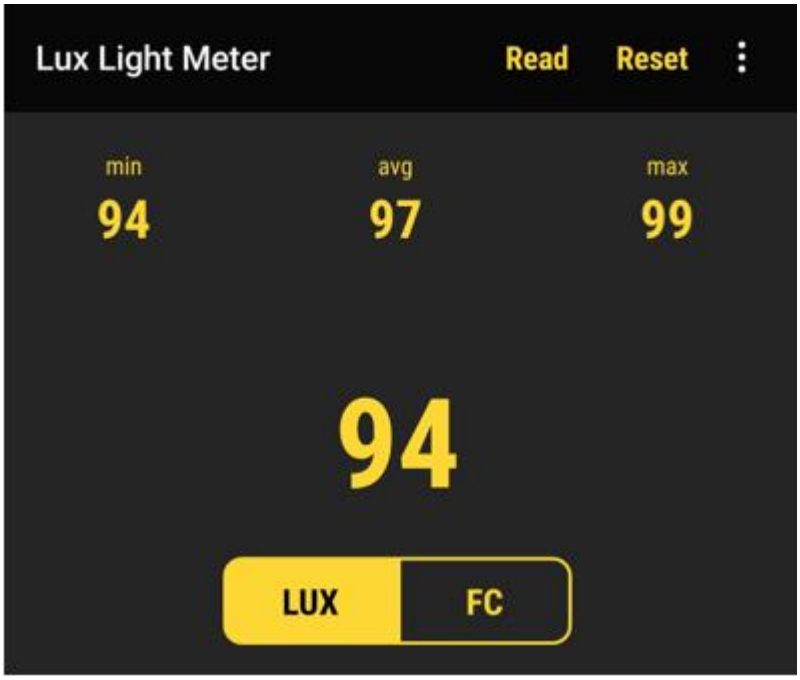
a



b



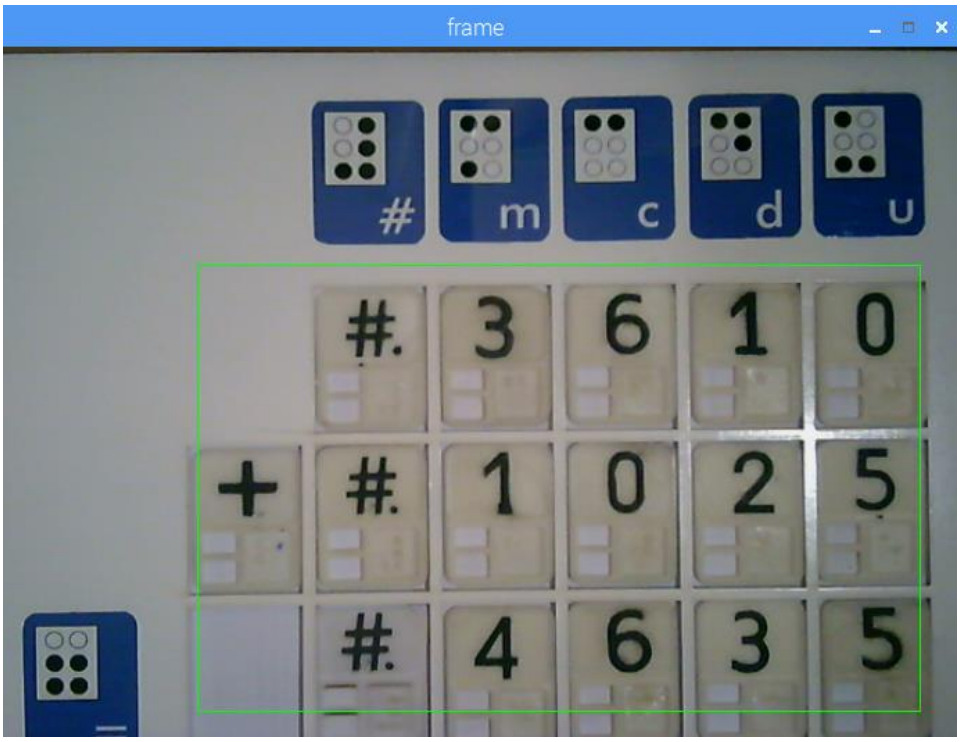
c



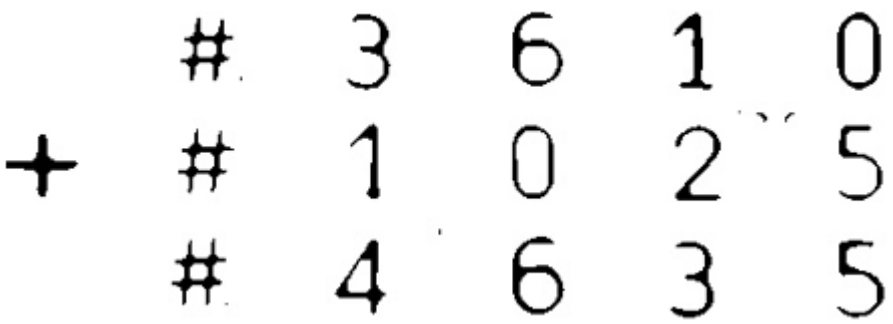
d

Figura 3. 4 a) Prueba de desempeño del prototipo, b) filtrado de números, c) resultado en consola, d) Medición de luz a 94 luxes.
Fuente: Autor.
Elaboración: Autor

En la Figura 3.5 (a) se presenta la prueba en un ambiente de 143 luxes, en presencia de luz artificial blanca. En la Figura 3.5 (b) se muestra los numero ingresados previamente filtrados. En la Figura 3.5 (c) se presenta en consola: los números ingresados, el operador seleccionado y el resultado final del juego y en la Figura 3.5 (d) la cantidad de luz al momento de ejecutar el programa.

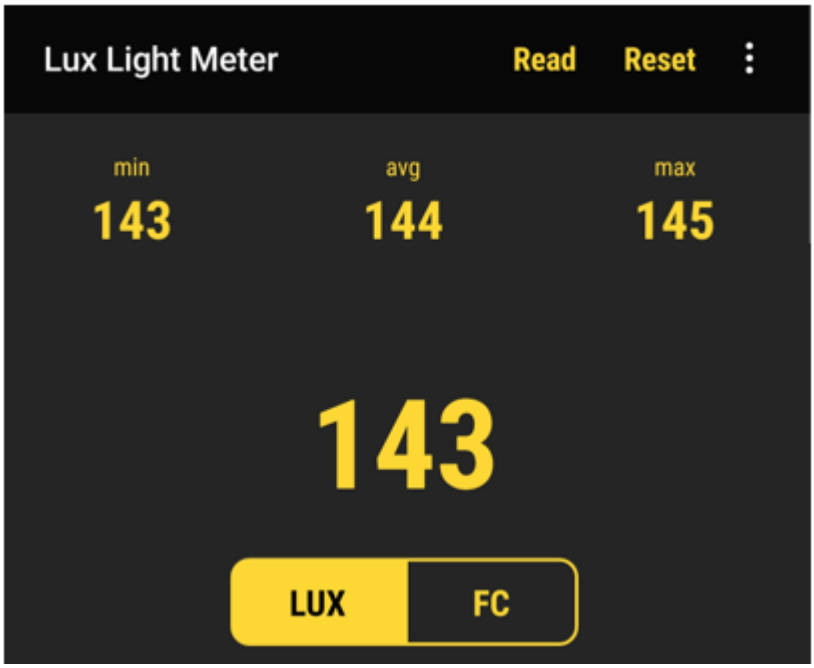


a



b

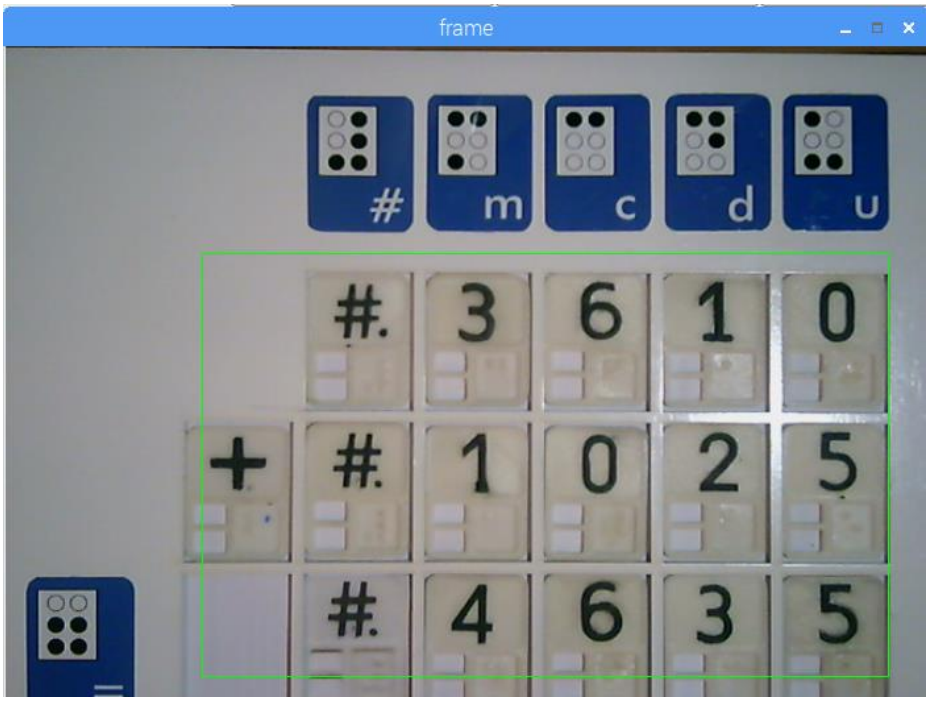
c



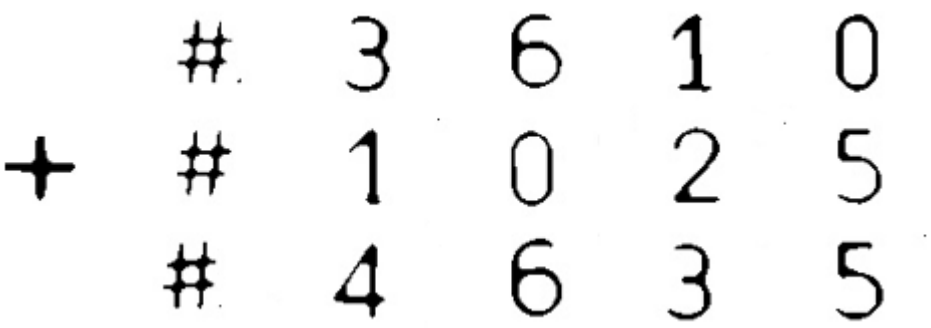
d

Figura 3. 5 a) Prueba de desempeño del prototipo, b) filtrado de números, c) resultado en consola, d) Medición de luz a 143 luxes.
Fuente: Autor.
Elaboración: Autor

En la Figura 3.6 (a) se presenta la prueba en un ambiente de 211 luxes, en presencia de luz artificial blanca. En la Figura 3.6 (b) se muestra los numero ingresados previamente filtrados. En la Figura 3.6 (c) se presenta en consola: los números ingresados, el operador seleccionado y el resultado final del juego y en la Figura 3.6 (d) la cantidad de luz al momento de ejecutar el programa.


frame
#
m
c
d
u
#. 3 6 1 0
+ #. 1 0 2 5
#. 4 6 3 5

a

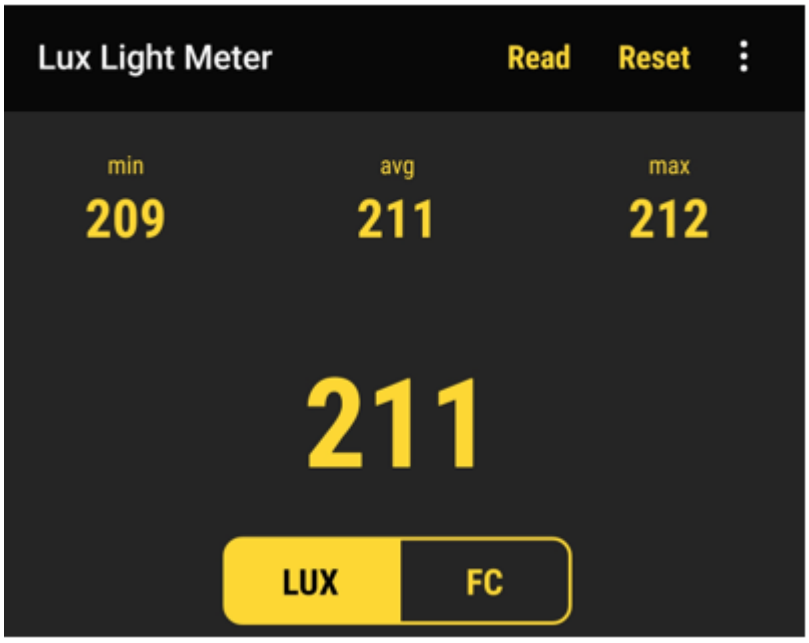

#. 3 6 1 0
+ # 1 0 2 5
#. 4 6 3 5

b


Terminal de IPython
Terminal 3/A
In [28]: runfile('/home/pi/Desktop/tesis/TESIS.py', wdir='/home/pi/Desktop/tesis')
Bienvenido
Juego de operaciones matemáticas.
el primer número ingresado es
#3610
---
el segundo número ingresado es
+#1025
---
el resultado ingresado es:
#4635

---
números ingresados de manera correcta.
el operador seleccionado es la suma
Resultado correcto
¡Felicitaciones!
In [29]:

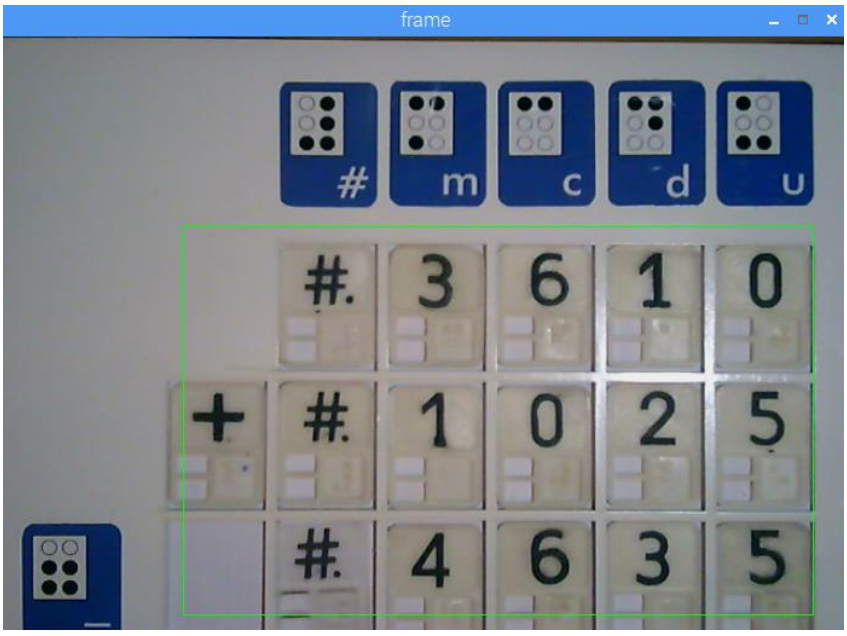
c



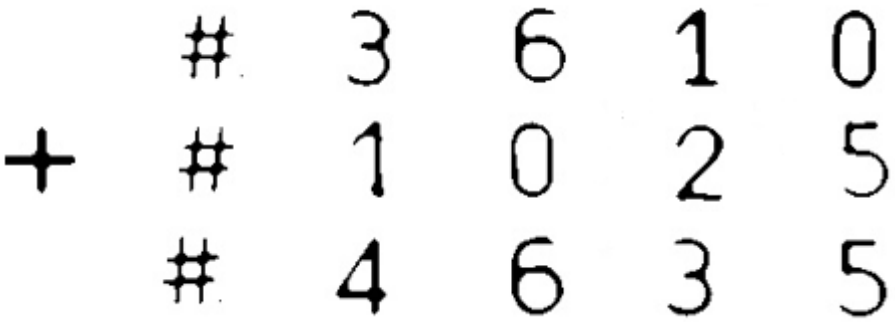
d

Figura 3. 6 a) Prueba de desempeño del prototipo, b) filtrado de números, c) resultado en consola, d) Medición de luz a 211 luxes.
Fuente: Autor.
Elaboración: Autor

En la Figura 3.7 (a) se presenta la prueba en un ambiente de 303 luxes, en presencia de luz artificial blanca. En la Figura 3.7 (b) se muestra los numero ingresados previamente filtrados. En la Figura 3.7 (c) se presenta en consola: los números ingresados, el operador seleccionado y el resultado final del juego y en la Figura 3.7 (d) la cantidad de luz al momento de ejecutar el programa.

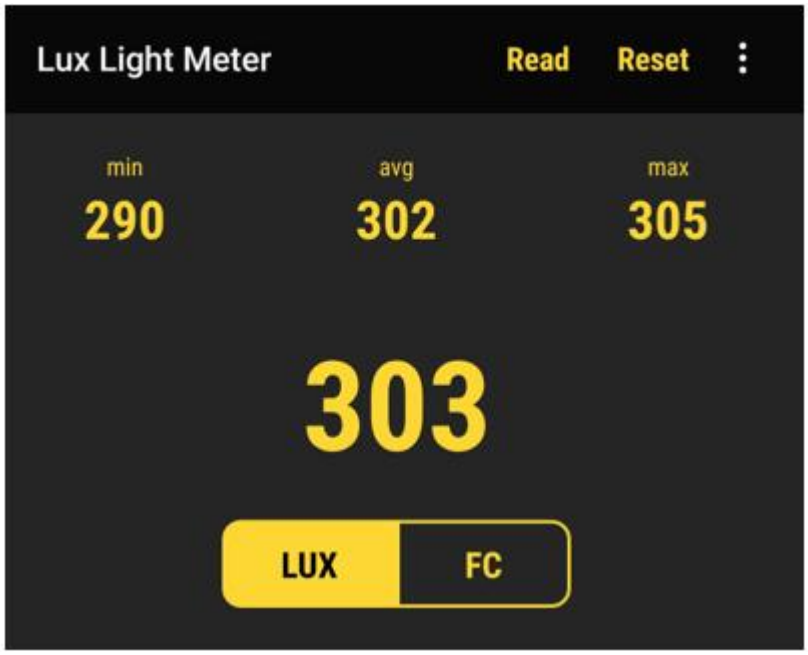


a



b

c



d

Figura 3. 7 a) Prueba de desempeño del prototipo, b) filtrado de números, c) resultado en consola, d) Medición de luz a 303 luxes.
Fuente: Autor.
Elaboración: Autor

Para un óptimo desempeño del prototipo este debe ser ejecutado en un ambiente mayor a 25 luxes. Entre más iluminado este el tablero, el programa tendrá menor probabilidad de fallo en el reconocimiento de los caracteres. Mientas que en ambientes menores a 25 luxes no se garantiza el correcto funcionamiento del mismo.

3.2 Pruebas de detección de caracteres

El experimento realizado para la detección de caracteres consta de 12 pruebas por cada carácter, dando como resultado un total de 168 pruebas, todas ellas realizadas bajo condiciones de luz de 16 luxes y 103 luxes.

La tabla 6 y tabla 7 se especifica el total de pruebas realizadas a cada ficha, el caracter empleado, el número de aciertos y errores y porcentaje de aciertos en el reconocimiento de caracteres. En cada intento existen 3 caracteres idénticos distribuidos en el tablero de forma aleatoria.

En las tablas 8 y 9 se muestran las matrices de confusión de la prueba de caracteres. Para condiciones de luminosidad de 16 luxes el porcentaje de sensibilidad es de 80,35%, de exactitud es 83,35% y precisión general de 96,55%. Mientras que, para condiciones de 103 luxes el porcentaje de sensibilidad es de 97,61%, de exactitud es 99,40% y precisión de 98,20%.

Tabla 6 Detección de caracteres a 16 luxes.

Caracter	Pruebas	Aciertos	Errores	Porcentaje %
0	12	12	0	100
1	12	11	1	91,66
2	12	10	2	83.33
3	12	12	0	100
4	12	10	2	83,33
5	12	9	3	75
6	12	12	0	100
7	12	12	0	100
8	12	12	0	100
9	12	4	8	33,33
#	12	11	1	91,66
+	12	11	1	91,66
-	12	3	9	25
x	12	6	6	50
%TOTAL				80,12

Fuente: Autor.
Elaboración: Autor

Tabla 7 Detección de caracteres a 103 luxes.

Caracter	Pruebas	Aciertos	Errores	Porcentaje %
0	12	12	0	100
1	12	11	1	91,66
2	12	12	0	100
3	12	12	0	100
4	12	12	0	100
5	12	12	0	100
6	12	11	1	91,66
7	12	12	0	100
8	12	12	0	100
9	12	12	0	100
#	12	12	0	100
+	12	12	0	100
-	12	11	1	91,66
x	12	11	1	91,66
%TOTAL				97,61

Fuente: Autor.
Elaboración: Autor.

Tabla 8 Matriz de confusión en detección de caracteres a 16 luxes.

	0	1	2	3	4	5	6	7	8	9	#	+	-	x
0	12 100%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%
1	0 0%	11 91.66%	0 0%	0 0%	1 8.33%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%
2	0 0%	0 0%	10 83.33%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%
3	0 0%	0 0%	0 0%	12 100%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%
4	0 0%	0 0%	0 0%	0 0%	10 83.33%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%
5	0 0%	0 0%	0 0%	0 0%	0 0%	9 75%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%
6	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	12 100%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%
7	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	12 100%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%
8	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	12 100%	0 0%	0 0%	0 0%	0 0%	0 0%
9	0 0%	0 0%	0 0%	1 8.33%	0 0%	0 0%	0 0%	0 0%	1 8.33%	4 33.33%	0 0%	0 0%	0 0%	0 0%
#	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	11 91.66%	0 0%	1 8.33%	0 0%
+	0 0%	0 0%	1 8.33%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	11 91.66%	0 0%	0 0%
-	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	3 25%	0 0%
x	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	6 50%

Fuente: Autor.
Elaboración: Autor.

Tabla 9 Matriz de confusión en detección de caracteres a 103 luxes.

	0	1	2	3	4	5	6	7	8	9	#	+	-	x
0	12 100%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%
1	0 0%	11 91.66%	0 0%	0 0%	1 8.33%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%
2	0 0%	0 0%	12 100%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%
3	0 0%	0 0%	0 0%	12 100%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%
4	0 0%	0 0%	0 0%	0 0%	12 100%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%
5	0 0%	0 0%	0 0%	0 0%	0 0%	12 100%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%
6	0 0%	0 0%	0 0%	0 0%	0 0%	1 8.33%	11 91.66%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%
7	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	12 100%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%
8	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	12 100%	0 0%	0 0%	0 0%	0 0%	0 0%
9	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	12 100%	0 0%	0 0%	0 0%	0 0%
#	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	12 100%	0 0%	0 0%	0 0%
+	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	12 100%	0 0%	0 0%
-	0 0%	1 8.33%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	11 91.66%	0 0%
x	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	11 91.66%

Fuente: Autor.
Elaboración: Autor

Queda en evidencia que el porcentaje de aciertos, sensibilidad, exactitud y precisión en la detección de caracteres mejora al superar los 100 luxes de luminosidad en el tablero. Existe una notoria mejoría en el reconocimiento en los caracteres 9, -, x. En ambos escenarios se ha realizado el mismo método de procesamiento de imágenes y detección de caracteres, por lo que se infiere que el nivel de luminosidad mejora el desempeño del prototipo.

CONCLUSIONES

Las prestaciones de la tarjeta electrónica Raspberry Pi 3, han servido de forma notable para la implementación del sistema expuesto, debido a que, integra en un mismo módulo recursos suficientes para la implementación de hardware y de software, la disponibilidad de entradas y salidas la convierte en una potente interfaz de interconexión con el exterior y las prestaciones que posee la hacen óptimo para recrear una máquina de visión artificial. Esto permite la creación de un sistema portable y de bajo costo, por lo que lo convierte en una herramienta de ayuda en la enseñanza de matemáticas en la educación básica inclusiva.

Es posible implementar un prototipo basado en visión por computadora que identifique caracteres e interactúe con el usuario, sin la necesidad de librerias que prescindan conexión a internet o alguna base de datos adicional.

El algoritmo de reconocimiento de caracteres “Tesseract” que se utilizó en el desarrollo del prototipo, es el motor OCR de código abierto más preciso disponible, a diferencia de otros motores de reconocimiento de caracteres, este no necesita de conexión a internet, además soporta una gran variedad de formatos de imágenes.

Se ha implementado como paso adicional en el procesamiento de imágenes, un algoritmo de reducción de sombras, con ayuda de los filtros de dilatación, desenfoque gaussiano y normalización este permite un mejor resultado en imágenes donde existe la presencia de sombras en el tablero, optimizando el desempeño de reconocimiento de caracteres.

Las pruebas llevadas a cabo en ambientes de luminosidad han permitido evidenciar que el sistema cumple con los objetivos planteados, se obtiene así un prototipo capaz de evaluar e interactuar en condiciones de luz superiores a los 25 luxes. La efectividad en el reconocimiento de caracteres del prototipo depende de las condiciones de luminosidad en el espacio realizado.

La disponibilidad de una herramienta de visión por computadora de bajo costo convierte al sistema desarrollado en un producto de interés en ámbitos de educación inclusiva, por lo que en trabajos posteriores se puede establecer la factibilidad de su comercialización y expansión en el sector educativo.

RECOMENDACIONES

En base al tema propuesto: desarrollo de algoritmos de aprendizaje automático basados en plataformas open-hardware y open-software, aplicados a un módulo de educación básica inclusiva, se recomienda:

- Probar diferentes técnicas, tanto de procesamiento de imágenes como de filtrado previo a la etapa de reconocimiento de caracteres, para de esta manera mejorar la imagen y el porcentaje de reconocimiento.
- Probar el prototipo en condiciones de luz normalizadas, no menores a 25 luxes, caso contrario la tasa de reconocimiento de caracteres se verá afectada.
- Probar distintas tipografías, así como distintas fichas para determinar si mejoran o no el resultado del prototipo.
- Para trabajos posteriores se recomienda el uso de mejores y más avanzados motores de reconocimiento de caracteres.
- A futuro se recomienda el uso de tarjetas electrónicas más avanzadas, así como el uso de una tarjeta de video dedicada, para un mejor desempeño del prototipo.

BIBLIOGRAFÍA

BeagleBoard. (2019). BeagleBone Black. Retrieved from https://beagleboard.org/black

Bradski, G., & Kaehler, A. (2008). *Learning OpenCV*.

Calderon-Cordova, C., Guajala-Michay, M., Barba-Guaman, R., & Quezada-Sarmiento, P. (2016). Design of a machine vision applied to educational board game. *2016 6th International Workshop on Computer Science and Engineering, WCSE 2016*, (August), 808–811. https://doi.org/10.4103/1119-3077.122852

Calderon, C. A., Guajala, M., Lanchi, J., Barba-guaman, L., & Bermeo, C. (2018). A machine vision system applied to the teaching of mathematics for blind or visually impaired children, 1–7.

Caporusso, N., Mkrtchyan, L., & Badia, L. (2010). A multimodal interface device for online board games designed for sight-impaired people. *IEEE Transactions on Information Technology in Biomedicine*, *14*(2), 248–254. https://doi.org/10.1109/TITB.2009.2034739

Díaz, A. G. (2012). Visión y procesamiento de imágenes para control de calidad.

Esqueda Elizondo, J. J. (2002). Fundamentos de Procesamiento de Imágenes.

Ferreira, F., & Cavaco, S. (2015). Mathematics for all: A game-based learning environment for visually impaired students. *Proceedings - Frontiers in Education Conference, FIE*, *2015-Febru*(February). https://doi.org/10.1109/FIE.2014.7044493

Fundation Raspberry. (2019). Raspberry pi 3 Model B. Retrieved from https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

Gaudina, M., Zappi, V., Brogni, A., & Caldwell, D. G. (2012). Haptic, audio, and visual: Multimodal distribution for interactive games. *IEEE Transactions on Instrumentation and Measurement*, *61*(11), 3103–3111. https://doi.org/10.1109/TIM.2012.2202071

Gonzalez, R. C., & Woods, R. E. (2008). Digital Image Processing (2nd Edition).

Guajala Michay, M. M. (2016). Diseño de material didáctico inclusivo para la enseñanza de niños ciegos. Retrieved from https://fido.palermo.edu/servicios_dyc/publicacionesdc/vista/detalle_articulo.php?id_libro=735&id_articulo=15430

INEC (Instituto Nacional de Estadisticas y Censos). (2018). Población por condición de discapacidad, según provincia, cantón, parroquia y área de empadronamiento. Retrieved from http://www.ecuadorencifras.gob.ec

KlipXtreme. (2019). Klip Xtreme KDC-600. Retrieved from https://www.klipxtreme.com/us_en/catalogsearch/result/?q=Klip+Xtreme+KDC-600

Lanchi, J. P. O. (2018). Diseñar y desarrollar una máquina de visión portátil aplicado a la supervisión automática de juegos educativos para niños con discapacidad visual.

Ludwig, J. (2013). Image Convolution. In Portland State University. (Ed.) (pp. 1–8). Retrieved from http://web.pdx.edu/~jduh/courses/Archive/geog481w07/Students/Ludwig_ImageConvolution.pdf

Magro, R. (2013). BINARIZACIÓN DE IMÁGENES DIGITALES Y SU ALGORITMIA COMO HERRAMIENTA APLICADA A LA ILUSTRACIÓN ENTOMOLÓGICA, *53*, 443–464.

Matsuo, M. (2016). ShadowRine : Accessible Game for Blind Users , and Accessible Action RPG for Visually Impaired Gamers, 2826–2827.

Ministerio de Educación del Ecuador. (2017). Reglamento general a la ley orgánica de Educación Intercultural. Retrieved from https://educacion.gob.ec/wp-content/uploads/downloads/2017/02/Reglamento-General-a-la-Ley-OrgAnica-de-Educacion-Intercultural.pdf

Navarro, J. A. (2016). Sistemas de Reconocimiento Óptico de Caracteres. *Revista Del Instituto Tecnológico de Informática*, 8–11.

ODROID. (2019). ODROID C2 Specifications. Retrieved from https://www.hardkernel.com/main/products/prdt_info.php?g_code=G145457216438

Ohene-Djan, J., & Fernando, S. (2016). Drawing for blind learners: Assistive technology for graphical design. *Proceedings - IEEE 16th International Conference on Advanced Learning Technologies, ICALT 2016*, 436–440. https://doi.org/10.1109/ICALT.2016.47

Oliveira, F., Quek, F., Cowan, H., & Fang, B. (2012). The haptic deictic systemhds: Bringing blind students to mainstream classrooms. *IEEE Transactions on Haptics*, *5*(2), 172–183. https://doi.org/10.1109/TOH.2011.35

OpenCV. (2019). OpenCV About. Retrieved June 1, 2019, from https://opencv.org/about/

OSMO. (2018). Osmo interactive games. Retrieved from https://www.playosmo.com/es/faq/

Park, Y., Lee, H., Hong, J., Lee, J., Min, T., Min, H. J., … Kim, H. (2016). Design and implementation of math-solving robot: Rank analysis for solving magic square puzzle. *International Conference on Control, Automation and Systems*, *0*(Iccas), 1567–1571. https://doi.org/10.1109/ICCAS.2016.7832510

Python. (2019a). Python. Retrieved from https://www.python.org/doc/essays/blurb/

Python. (2019b). Python text to speech. Retrieved from https://pyttsx3.readthedocs.io/en/latest/

Solem, J. E. (2012). *Programming Computer Vision with Python: Tools and algorithms for analyzing images*.

Sribunruangrit, N., Marque, C., Lenay, C., & Gapenne, O. (2004). Graphic-user-interface system for people with severely impaired vision in mathematics class. *Conf Proc IEEE Eng Med Biol Soc*, *7*, 5145–5148. https://doi.org/10.1109/IEMBS.2004.1404432

Tesseract. (2019). Tesseract OCR. Retrieved from https://github.com/tesseract-ocr/tesseract

University of Toronto, D. of C. S. (2019). Detección de contorno. Retrieved from https://www.cs.toronto.edu/~dmac/images/ProjectFiles/sraf/srafdoc/deteccion.html

Zhang, X. (2010). Adaptive Haptic Exploration of Geometrical Structures in Map Navigation for People with Visual Impairment. https://doi.org/10.1109/HAVE.2010.5623984

ANEXOS

Anexo 1

Código de prototipo

```
#Librerias-------------------------------------------------------------------------
-----------------------------------
import numpy as np
import cv2
from PIL import Image
import pytesseract
import re
import pyttsx3
#-----------------------------------------------------------------------------------
-----------------------------------
#Saludo de bienvenida---------------------------------------------------------------
-----------------------------------
print('Bienvenido')
print('Juego de operaciones matemáticas.')
engine = pyttsx3.init()
engine.say('Bienvenido')
engine.say('Juego de operaciones matemáticas.')
engine.runAndWait()
#-----------------------------------------------------------------------------------
-----------------------------------
#ROI N°1----------------------------------------------------------------------------
-----------------------------------
x1=130
y1=162
x2=680
y2=478
#-----------------------------------------------------------------------------------
-----------------------------------

cap = cv2.VideoCapture(0)
#-----------------------------------------------------------------------------------
-----------------------------------

#Procesamiento de imagen + OCR------------------------------------------------------
-----------------------------------
def proc(img):

    #Tesseract v 4.0
    #-------------------------------------------------------------------------------
-----------------------------------
    #OCR options:
    config = ("-l eng --psm 6 --oem 1")     #--PRIMER NUMERO
    config2 = ("-l eng --psm 6 --oem 1")    #--SEGUNDO NUEMRO
    config3 = ("-l eng --psm 6 --oem 1")    #--TERCER NUMERO

    #lenguaje=default, psm=6:Assume a single uniform block of text., OEM=1: Neural
nets LSTM engine only.--------------
    #-------------------------------------------------------------------------------
-----------------------------------
    i=0, j=0, k=0

    img = cv2.imread('capture.jpg', -1)  # abre imagen
    rgb_planes = cv2.split(img)
```

```
    result_planes = []
    result_norm_planes = []
    for plane in rgb_planes:
        dilated_img = cv2.dilate(plane, np.ones((7,7), np.uint8))
        bg_img = cv2.medianBlur(dilated_img, 11)
        diff_img = 255 - cv2.absdiff(plane, bg_img)
        norm_img = cv2.normalize(diff_img, None ,alpha=0, beta=255,
norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_32F)
        result_planes.append(diff_img)
        result_norm_planes.append(norm_img)

    result = cv2.merge(result_planes)
  #  result_norm = cv2.merge(result_norm_planes)

    cv2.imwrite('shadows_out.png', result)
#########################################################################################
#####################################
#########################################################################################
#####################################

    ROI_1 = cv2.imread("shadows_out.png")
    #-------------------------------------------------------------------------------
-----------------------------------
    #Escala de grises de ROI 1-----------------------------------------------------
-----------------------------------
    gray = cv2.cvtColor(ROI_1, cv2.COLOR_BGR2GRAY)
    #-------------------------------------------------------------------------------
-----------------------------------
    #-------------------------------------------------------------------------------
-----------------------------------
    #ROI 2: PRIMER NUMERO----------------------------------------------------------
-----------------------------------
    x3=89
    y3=7
    x4=523
    y4=73
    #-------------------------------------------------------------------------------
-----------------------------------
    #ROI 3 SEGUNDO NUMERO----------------------------------------------------------
-----------------------------------
    x5=8
    y5=125
    x6=523
    y6=188
    #-------------------------------------------------------------------------------
-----------------------------------
    #ROI 4 RESULTADO---------------------------------------------------------------
-----------------------------------
    x7=2
    y7=240
    x8=526
    y8=300
    #-------------------------------------------------------------------------------
-----------------------------------
    #ROI 2: PROCESAMIENTO DE IMAGEN SOBRE PRIMER NUMERO---------------------------
-----------------------------------
    ROI_2=gray[y3:y4,x3:x4]
    blur = cv2.medianBlur(ROI_2,5)
    _,th = cv2.threshold(blur,0,230,cv2.THRESH_BINARY+cv2.THRESH_OTSU)
```

```
    kernel = np.ones((4,4),np.uint8)
    erosion = cv2.dilate(th,kernel,iterations = 1)
    equ = cv2.equalizeHist(erosion)
    _ , contours, hierarchy =
cv2.findContours(erosion,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
    for cnt in contours:
        if cv2.contourArea(cnt)>52:
            [x,y,w,h] = cv2.boundingRect(cnt)
            if  h>34 and w<47:
                cuadro = equ[y:y+h,x:x+w]
                nombre = './cap%i'%i + '.jpg'
                cv2.imwrite(nombre, cuadro)
                i=i+1
    #cv2.imshow('blurred image 1',equ)
    cv2.imwrite("roi1.jpg", equ)
    #OCR sobre primer numero-------------------------------------------------------
-------------------------------------
    roi1 = Image.open("roi1.jpg")
    t1=pytesseract.image_to_string(roi1, config=config)
    g= ''.join(c for c in t1 if c in '#0123456789')
    print('el primer número ingresado es')
    print(g)
    print('-----------------------------------------------')     #------------------
-------------------------------------------------------------------------------
--------------
    #ROI3 PROCESAMIENTO DE IMAGEN SOBRE SEGUNDO NUMERO---------------------------
-------------------------------------
    valor2=gray[y5:y6,x5:x6]
    blur2 = cv2.medianBlur(valor2,5)
    _,th2 = cv2.threshold(blur2,0,230,cv2.THRESH_BINARY+cv2.THRESH_OTSU)
    erosion2 = cv2.dilate(th2,kernel,iterations = 1)
    equ2 = cv2.equalizeHist(erosion2)
    _ , contours2, hierarchy2 =
cv2.findContours(equ2,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
    for cnt2 in contours2:
        if cv2.contourArea(cnt2)>49:
            [x2,y2,w2,h2] = cv2.boundingRect(cnt2)
            if  h2>34 and w2<47:
                cuadro2 = equ2[y2:y2+h2,x2:x2+w2]
                nombre2 = './cifra%i'%j + '.jpg'
                cv2.imwrite(nombre2, cuadro2)
                j=j+1
   # cv2.imshow('blurred image 2',equ2)
    cv2.imwrite("roi2.jpg", equ2)
    roi2 = Image.open("roi2.jpg")
    t2=pytesseract.image_to_string(roi2, config=config2)
    g2= ''.join(c for c in t2 if c in '-+Xx#0123456789')
    print('el segundo número ingresado es')
    print(g2)
    print('-----------------------------------------------')
    #-------------------------------------------------------------------------------
-------------------------------------
    #ROI4: PROCESAMIENTO DE IMAGEN SOBRE  RESULTADO    ---------------------------
-------------------------------------
    valor3=gray[y7:y8,x7:x8]
    blur3 = cv2.medianBlur(valor3,5)
    _,th3 = cv2.threshold(blur3,0,230,cv2.THRESH_BINARY+cv2.THRESH_OTSU)
    erosion3 = cv2.dilate(th3,kernel,iterations = 1)
    equ3 = cv2.equalizeHist(erosion3)
```

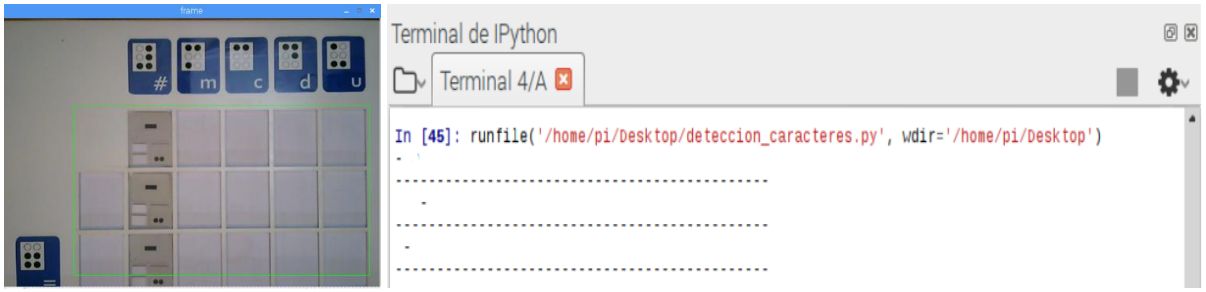
```
    _ , contours3, hierarchy3 =
cv2.findContours(equ3,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
    for cnt3 in contours3:
        if cv2.contourArea(cnt3)>49:
            [x3,y3,w3,h3] = cv2.boundingRect(cnt3)
            if  h3>35 and w3<47:
                cuadro3 = equ3[y3:y3+h3,x3:x3+w3]
                nombre3 = './resul%i'%k + '.jpg'
                cv2.imwrite(nombre3, cuadro3)
                k=k+1
   # cv2.imshow('blurred image 3',equ3)
    cv2.imwrite("roi3.jpg", equ3)
    roi3 = Image.open("roi3.jpg")
    t3=pytesseract.image_to_string(roi3, config=config3)
    g3= ''.join(c for c in t3 if c in '#0123456789')
    print("el resultado ingresado es:")
    print(g3)
    print('-----------------------------------------------')
    print('-----------------------------------------------')
    #-------------------------------------------------------------------------------
-------------------------------------

    #INICIO DE OPERACION MAS COMPARACION DE RESULTADOS---------------------------
-----------------------------------
   #EXTRAE # DE LA CIFRA PARA OPERACION-----------------------------------------
-----------------------------------
    regex = re.compile('#')


    if((regex.search(g) and regex.search(g2) and regex.search(g3)) == None):
        engine.say('números ingresados incorrectos inserte el signo numeral')
        print(('números ingresados incorrectos inserte el signo numeral'))
        engine.runAndWait()
    else:
        engine.say("números ingresados de manera correcta.")
        print("números ingresados de manera correcta.")
    #-------------------------------------------------------------------------------
-----------------------------------
    #MULTIPLICACION-------------------------------------------------------------
-------------------------------------------
        if (g2.find('x') != -1):
            engine.say('el operador seleccionado es la multiplicación')
            print('el operador seleccionado es la multiplicación')
            engine.runAndWait()
            f=g.replace("#")
            f2=g2.replace("x#","")
            f3=g3.replace("#","")
            #print(f,f2)
            cifra1= int(f)
            cifra2= int(f2)
            cifra3= int(f3)
            engine.say('el primer número ingresado es')
            engine.say(cifra1)
            engine.runAndWait()
            engine.say('el segundo número ingresado es')
            engine.say(cifra2)
            engine.runAndWait()
            engine.say('el resultado ingresado es')
            engine.say(cifra3) #ojo
```

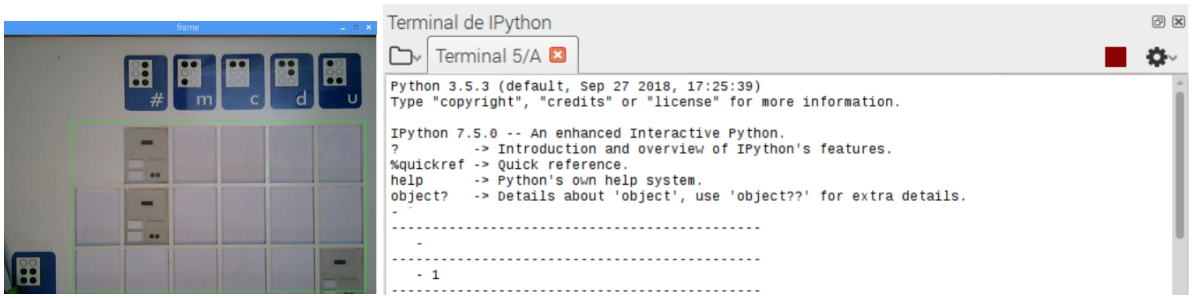
```
            engine.runAndWait()
            operacion= (cifra1)*(cifra2)
    #SUMA-------------------------------------------------------------------------
-----------------------------------
        if (g2.find('+') != -1):
            engine.say('el operador seleccionado es la suma')
            print('el operador seleccionado es la suma')
            engine.runAndWait()
            f=g.replace("#","")
            f2=g2.replace("+#","")
            f3=g3.replace("#","")
            #print(f,f2)
            cifra1= int(f)
            cifra2= int(f2)
            cifra3= int(f3)
            engine.say('el primer número ingresado es')
            engine.say(cifra1)
            engine.runAndWait()
            engine.say('el segundo número ingresado es')
            engine.say(cifra2)
            engine.runAndWait()
            engine.say('el resultado ingresado es')
            engine.say(cifra3) #ojo
            engine.runAndWait()
            operacion= cifra1+cifra2
            #print(g3)
     #-------------------------------------------------------------------------
-----------------------------------
     #RESTA-------------------------------------------------------------------------
-----------------------------------
        if (g2.find('-') != -1):
            engine.say('el operador seleccionado es la resta')
            engine.runAndWait()
            f=g.replace("#","")
            f2=g2.replace("-#","")
            f3=g3.replace("#","")
            #print(f,f2)
            cifra1= int(f)
            cifra2= int(f2)
            cifra3= int(f3)
            engine.say('el primer número es')
            engine.say(cifra1)
            engine.runAndWait()
            engine.say('el segundo número es')
            engine.say(cifra2)
            engine.runAndWait()
            engine.say('el tercer número es')
            engine.say(cifra3)
            engine.runAndWait()
            operacion= cifra1-cifra2
            #print(operacion)
            #f3=g3.replace("#","")   ######******##### numeral en la columna3
    #-------------------------------------------------------------------------
-----------------------------------
    #COMPARACIÓN-------------------------------------------------------------------
-----------------------------------
        if (operacion == cifra3):
            engine.say('Resultado correcto')
            engine.say('¡Felicitaciones!')
```

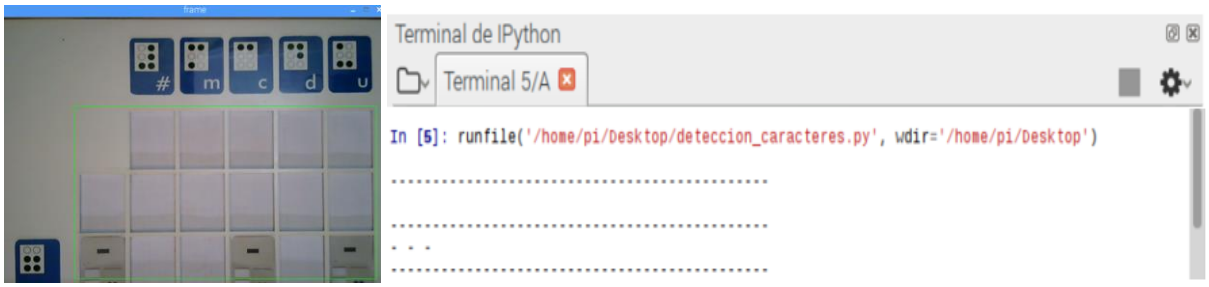
```
            print('Resultado correcto')
            print('¡Felicitaciones!')
            engine.runAndWait()
        else:
            print('Resultado erroneo')
            print('la solución es')
            print(operacion)
            print('Por favor, Intente otra vez.')
            engine.say('Resultado erroneo')
            engine.say('la solucion es')
            engine.say(g3)
            engine.say('Por favor, Intente otra vez.')
            engine.runAndWait()

    return



while(True):
    ret, frame = cap.read()
    flp = cv2.flip(frame,-1)
    frame1 = cv2.resize(flp,(700,500))
    frame_clone = frame1.copy()

    #ROI 1-------------------------------------------------------------------------
-----------------------------------
    cv2.rectangle(frame_clone, (x1, y1), (x2, y2), (0, 255, 0), 1)
    cv2.imshow('frame',frame_clone)
    roi = frame_clone[y1:y2, x1:x2]
    #-------------------------------------------------------------------------------
-----------------------------------
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

    #c=cv2.waitKey(1)
    #-------------------------------------------------------------------------------
-----------------------------------
    #Captura de imagen mediante tecla "v"-----------------------------------------
-----------------------------------
    if cv2.waitKey(1) == ord('v'):
        cv2.imwrite("capture.jpg", roi)
        captura = cv2.imread("capture.jpg")
        proc(captura)
        break
    else:
        None

cap.release()
cv2.destroyAllWindows()
```

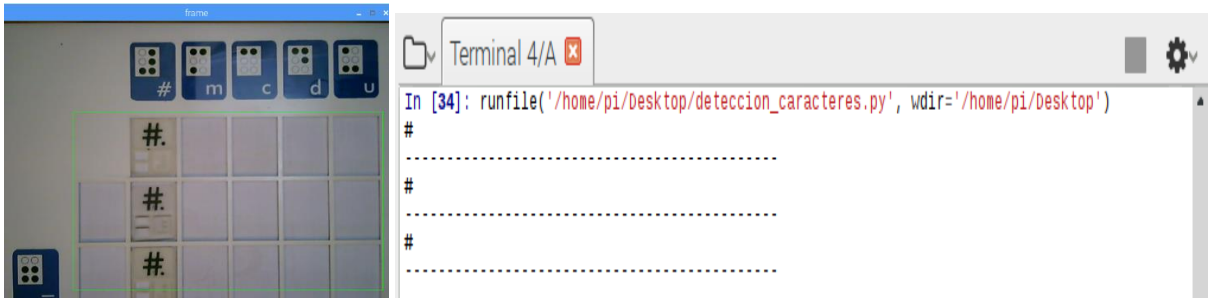
Anexo 2

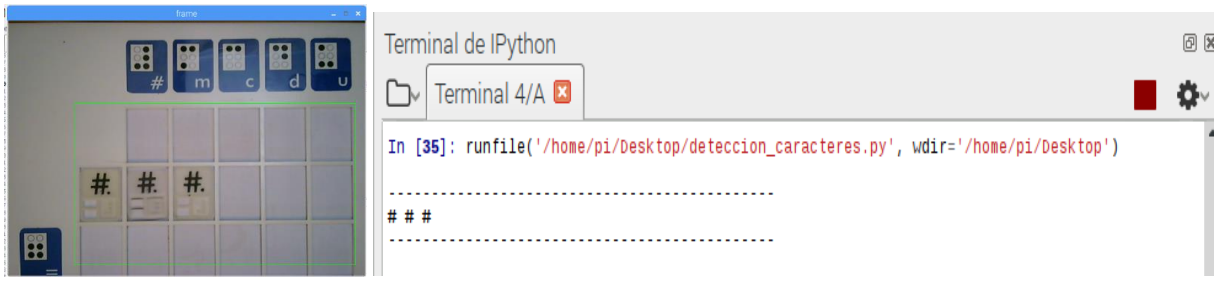
Capturas de imagen de detección de caracteres 103 luxes.







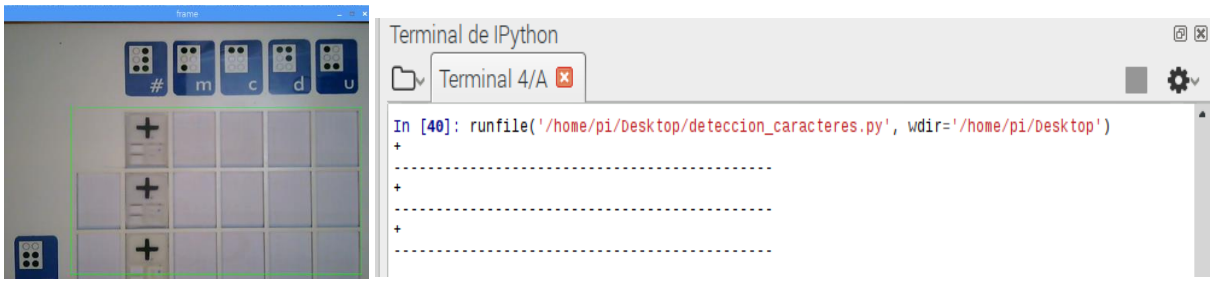



frame
Terminal de IPython
Terminal 4/A
In [35]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
# # #


frame
Terminal de IPython
Terminal 4/A
In [36]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
#
#
#


frame
Terminal de IPython
Terminal 4/A
In [38]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
#
#
#


frame
Terminal de IPython
Terminal 4/A
In [39]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
+++
In [40]:


frame
Terminal de IPython
Terminal 4/A
In [40]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
+
+
+


frame
Terminal de IPython
Terminal 4/A
In [41]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
+
+ +


frame
m c d u
+ + +
Terminal de IPython
Terminal 4/A
In [42]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
+
+ +

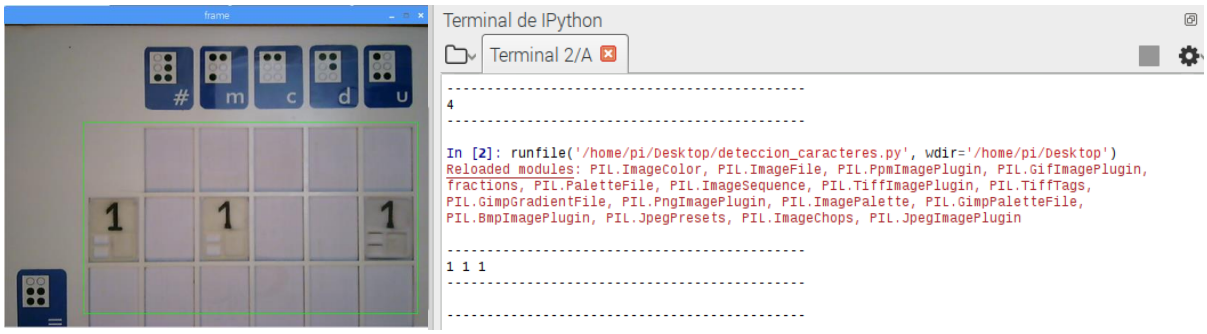

frame
m c d u
0 0 0
Terminal de IPython
Terminal 1/A
In [1]:
In [1]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
0
0
0


frame
m c d u
0 0 0
Terminal de IPython
Terminal 1/A
In [2]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
Reloaded modules: PIL.PngImagePlugin, fractions, PIL.GifImagePlugin, PIL.BmpImagePlugin, PIL.JpegPresets, PIL.PpmImagePlugin, PIL.ImageFile, PIL.ImageChops, PIL.GimpPaletteFile, PIL.TiffTags, PIL.PaletteFile, PIL.JpegImagePlugin, PIL.ImageColor, PIL.ImagePalette, PIL.TiffImagePlugin, PIL.ImageSequence, PIL.GimpGradientFile
0
0
0


frame
m c d u
0 0 0
Terminal de IPython
Terminal 1/A
In [3]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
0 0 0

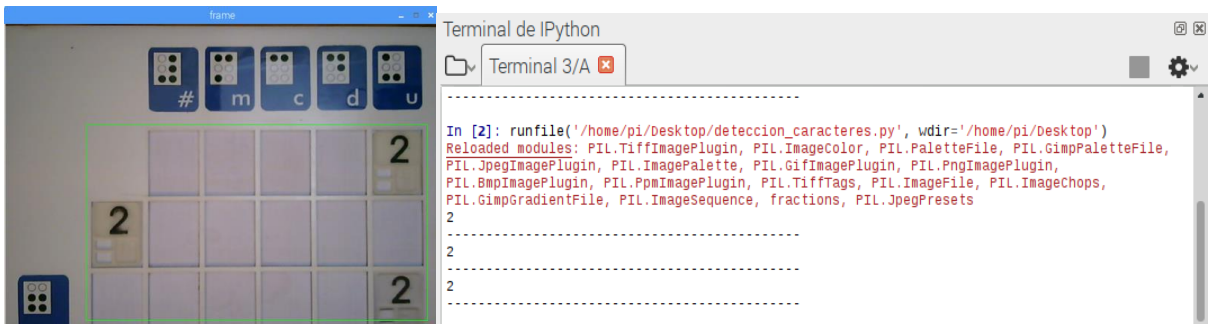

frame
m c d u
0 0 0
Terminal de IPython
Terminal 1/A
In [4]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
0
0
0


frame
m c d u
1
1
1
Terminal de IPython
Terminal 2/A
In [1]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
1
4


frame
m c d u
1 1 1
Terminal de IPython
Terminal 2/A
4
In [2]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
Reloaded modules: PIL.ImageColor, PIL.ImageFile, PIL.PpmImagePlugin, PIL.GifImagePlugin, fractions, PIL.PaletteFile, PIL.ImageSequence, PIL.TiffImagePlugin, PIL.TiffTags, PIL.GimpGradientFile, PIL.PngImagePlugin, PIL.ImagePalette, PIL.GimpPaletteFile, PIL.BmpImagePlugin, PIL.JpegPresets, PIL.ImageChops, PIL.JpegImagePlugin
1 1 1


frame
m c d u
1
1
1
Terminal de IPython
Terminal 2/A
In [4]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
1
1
1

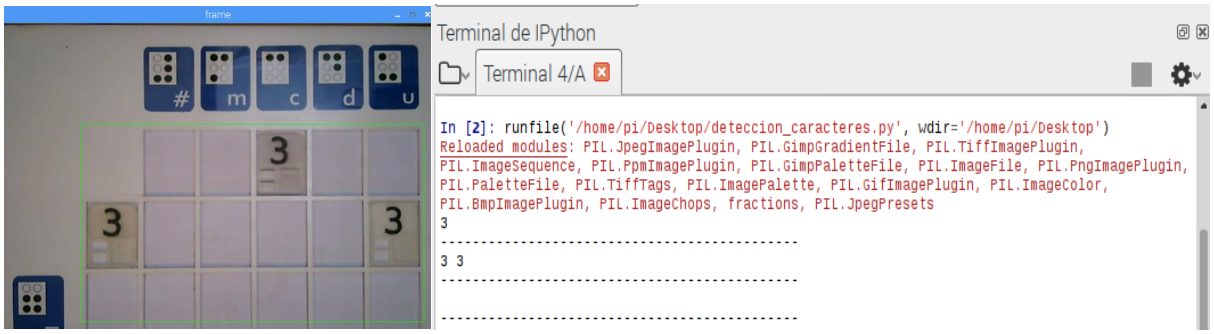

frame
m c d u
1
1
1
Terminal de IPython
Terminal 3/A
In [1]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
1
1
1


frame
m c d u
2
2
Terminal de IPython
Terminal 3/A
In [2]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
Reloaded modules: PIL.TiffImagePlugin, PIL.ImageColor, PIL.PaletteFile, PIL.GimpPaletteFile, PIL.JpegImagePlugin, PIL.ImagePalette, PIL.GifImagePlugin, PIL.PngImagePlugin, PIL.BmpImagePlugin, PIL.PpmImagePlugin, PIL.TiffTags, PIL.ImageFile, PIL.ImageChops, PIL.GimpGradientFile, PIL.ImageSequence, fractions, PIL.JpegPresets
2
2
2


frame
m c d u
2
2 2
Terminal de IPython
Terminal 3/A
In [3]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
2
2 2


frame
m c d u
2 2 2
Terminal de IPython
Terminal 3/A
In [4]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
2 2 2


frame
m c d u
2
2
2
Terminal de IPython
Terminal 4/A
In [1]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
2
2
2


frame
m c d u
3
3 3
Terminal de IPython
Terminal 4/A
In [2]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
Reloaded modules: PIL.JpegImagePlugin, PIL.GimpGradientFile, PIL.TiffImagePlugin, PIL.ImageSequence, PIL.PpmImagePlugin, PIL.GimpPaletteFile, PIL.ImageFile, PIL.PngImagePlugin, PIL.PaletteFile, PIL.TiffTags, PIL.ImagePalette, PIL.GifImagePlugin, PIL.ImageColor, PIL.BmpImagePlugin, PIL.ImageChops, fractions, PIL.JpegPresets
3
3 3

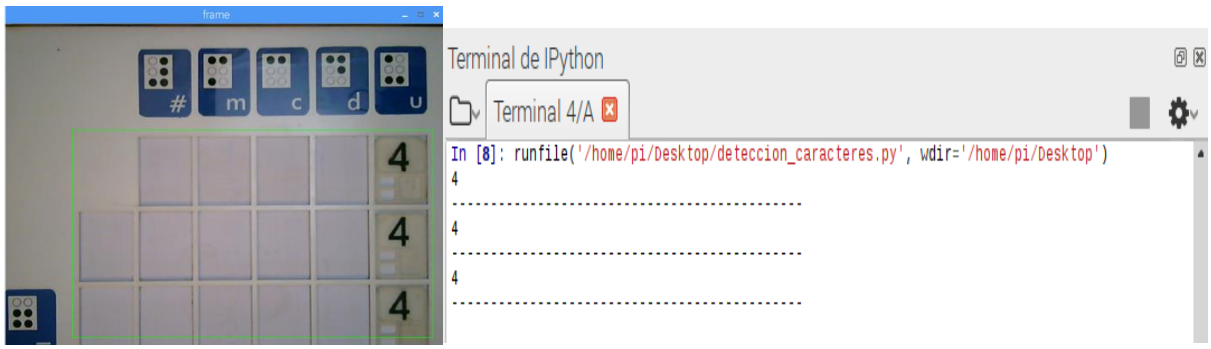

frame
m c d u
3
3
3
Terminal de IPython
Terminal 4/A
In [3]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
3
3
3


frame
m c d u
3 3
3
Terminal de IPython
Terminal 4/A
In [4]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
3 3
3


frame
m c d u
3
3 3
Terminal de IPython
Terminal 4/A
In [5]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
3
3 3

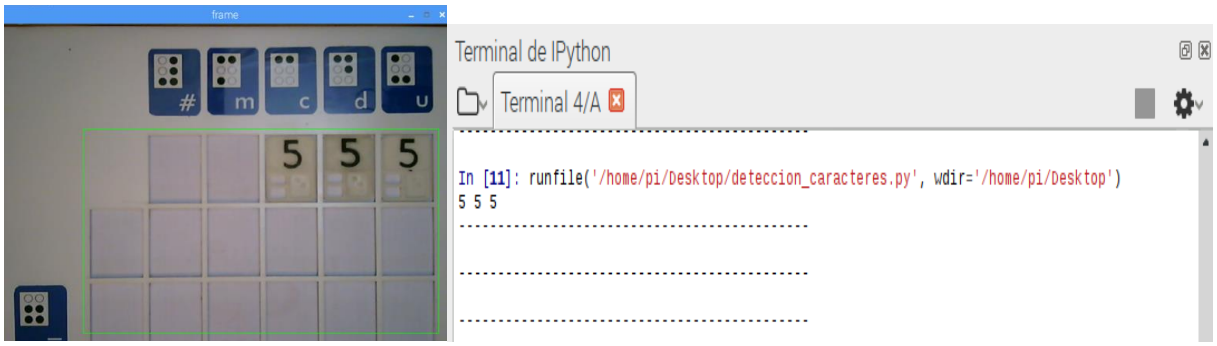

frame
m c d u
4
4
4
Terminal de IPython
Terminal 4/A
In [6]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
4
4
4

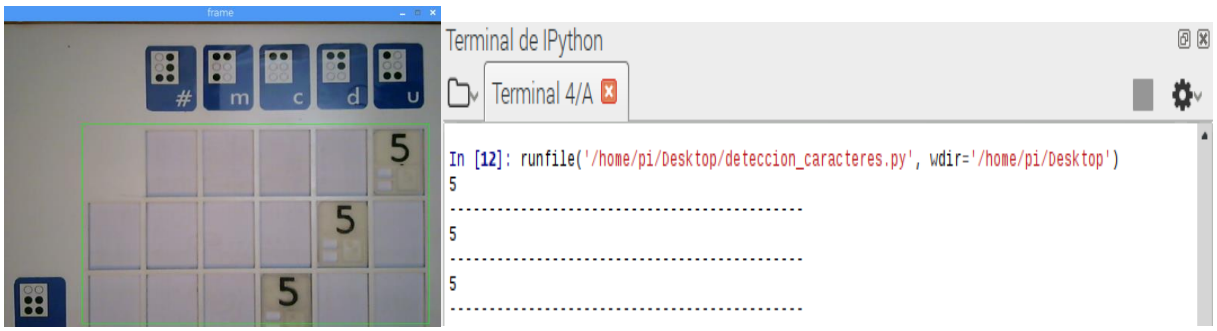

frame
m c d u
4 4
4
Terminal de IPython
Terminal 4/A
In [7]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
4 4
4


frame
m c d u
4
4
4
Terminal de IPython
Terminal 4/A
In [8]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
4
4
4


frame
m c d u
4 4 4
Terminal de IPython
Terminal 4/A
In [9]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
4 4 4


frame
m c d u
5
5
5
Terminal de IPython
Terminal 4/A
In [10]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
5
5
5

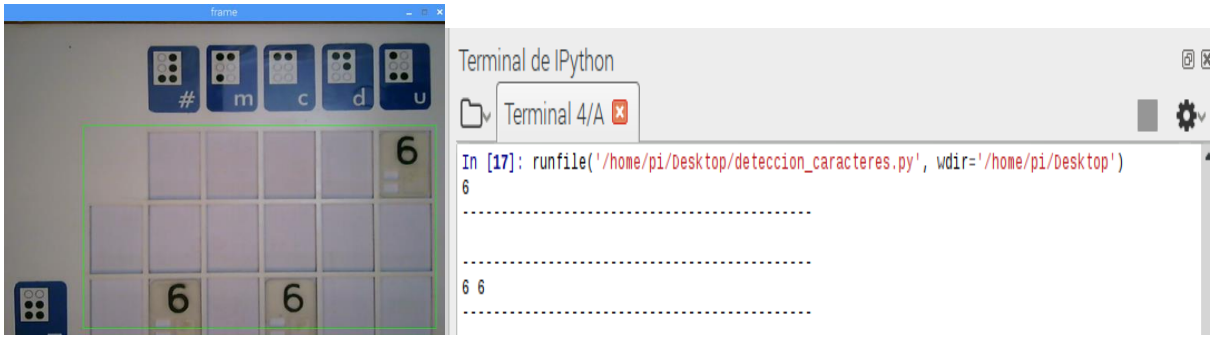

frame
m c d u
5 5 5
Terminal de IPython
Terminal 4/A
In [11]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
5 5 5


frame
m c d u
5
5
5
Terminal de IPython
Terminal 4/A
In [12]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
5
5
5


frame
m c d u
5
5 5
Terminal de IPython
Terminal 4/A
In [13]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
5
5 5


frame
m c d u
6 6 6
Terminal de IPython
Terminal 4/A
In [14]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
6 6
6


frame
m c d u
6 6 6
Terminal de IPython
Terminal 4/A
In [16]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
56 6
6


frame
m c d u
6 6 6
Terminal de IPython
Terminal 4/A
In [17]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
6
6 6


frame
m c d u
6 6 6
Terminal de IPython
Terminal 4/A
In [18]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
6 6 6


frame
m c d u
7 7 7
Terminal de IPython
Terminal 4/A
In [19]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
7
7 7


frame
m c d u
7 7 7
Terminal de IPython
Terminal 4/A
In [20]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
7 7 7


frame
m c d u
7
7 7
Terminal de IPython
Terminal 4/A
In [21]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
7
7 7


frame
m c d u
7
7
7
Terminal de IPython
Terminal 4/A
In [22]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
7
7
7


frame
m c d u
8
8 8
Terminal de IPython
Terminal 4/A
In [23]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
8
8 8


frame
m c d u
8 8
8
Terminal de IPython
Terminal 4/A
In [24]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
8 8
8


frame
m c d u
8 8 8
Terminal de IPython
Terminal 4/A
In [25]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
8 8 8

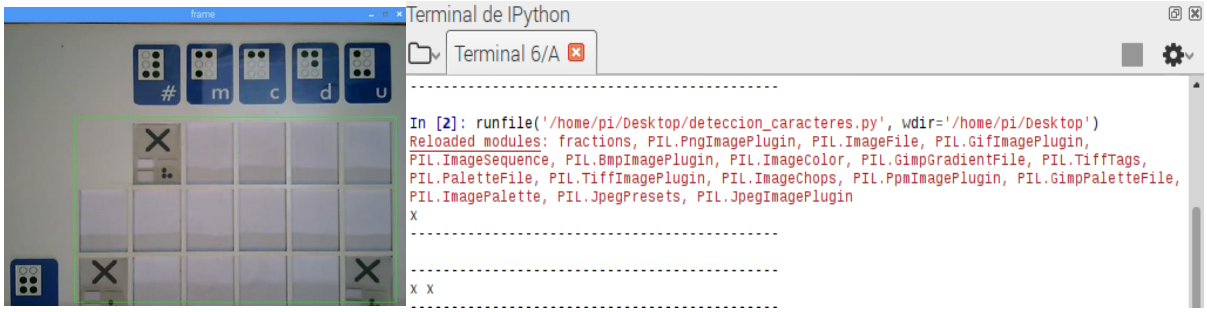

frame
m c d u
8
8 8
Terminal de IPython
Terminal 4/A
In [28]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
8
8 8


frame
m c d u
9
9 9
Terminal de IPython
Terminal 4/A
In [30]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
9
9 9


frame
m c d u
9
9
9
Terminal de IPython
Terminal 4/A
In [31]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
9
9
9


frame
m c d u
9 9 9
Terminal de IPython
Terminal 4/A
In [32]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
9 9 9

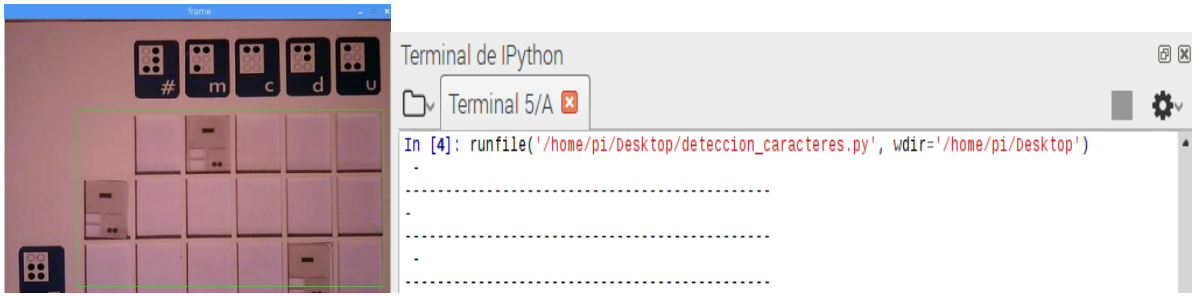

frame
#
m
c
d
u
Terminal de IPython
Terminal 6/A
In [1]:
In [1]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
X
X
X


frame
#
m
c
d
u
Terminal de IPython
Terminal 6/A
In [2]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
Reloaded modules: fractions, PIL.PngImagePlugin, PIL.ImageFile, PIL.GifImagePlugin, PIL.ImageSequence, PIL.BmpImagePlugin, PIL.ImageColor, PIL.GimpGradientFile, PIL.TiffTags, PIL.PaletteFile, PIL.TiffImagePlugin, PIL.ImageChops, PIL.PpmImagePlugin, PIL.GimpPaletteFile, PIL.ImagePalette, PIL.JpegPresets, PIL.JpegImagePlugin
X
X X


frame
#
m
c
d
u
Terminal de IPython
Terminal 7/A
In [2]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
Reloaded modules: PIL.JpegImagePlugin, PIL.PngImagePlugin, PIL.PaletteFile, PIL.ImageColor, PIL.ImageChops, PIL.PpmImagePlugin, PIL.ImageFile, PIL.ImagePalette, PIL.JpegPresets, PIL.GimpPaletteFile, PIL.TiffImagePlugin, PIL.GimpGradientFile, PIL.ImageSequence, fractions, PIL.GifImagePlugin, PIL.TiffTags, PIL.BmpImagePlugin
X
X
X


frame
#
m
c
d
u
Terminal de IPython
Terminal 7/A
In [4]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
X
X

Capturas de imagen de detección de caracteres 16 luxes.




Terminal de IPython
Terminal 4/A
In [23]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
#-

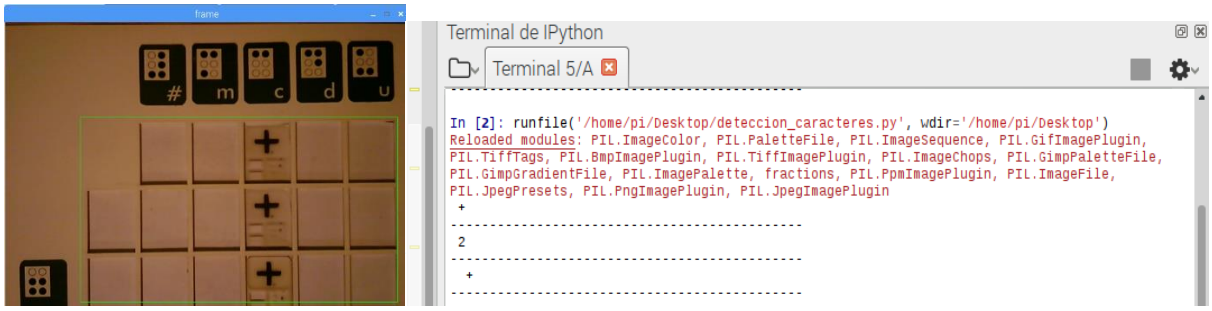

Terminal de IPython
Terminal 4/A
In [24]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
#
# #


Terminal de IPython
Terminal 4/A
In [25]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
#
#
#

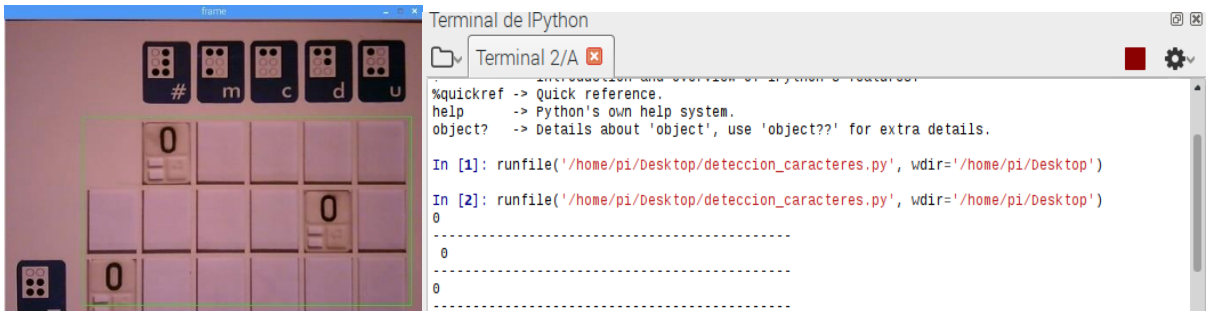

Terminal de IPython
Terminal 4/A
In [26]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
# #
#


Terminal de IPython
Terminal 4/A
In [27]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
+
+
+


frame
Terminal de IPython
Terminal 4/A
In [28]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
+ + +


frame
Terminal de IPython
Terminal 5/A
In [2]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
Reloaded modules: PIL.ImageColor, PIL.PaletteFile, PIL.ImageSequence, PIL.GifImagePlugin, PIL.TiffTags, PIL.BmpImagePlugin, PIL.TiffImagePlugin, PIL.ImageChops, PIL.GimpPaletteFile, PIL.GimpGradientFile, PIL.ImagePalette, fractions, PIL.PpmImagePlugin, PIL.ImageFile, PIL.JpegPresets, PIL.PngImagePlugin, PIL.JpegImagePlugin
+
2
+


frame
Terminal de IPython
Terminal 5/A
In [3]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
+
+
+
In [4]:


frame
Terminal de IPython
Terminal 2/A
%quickref -> Quick reference.
help -> Python's own help system.
object? -> Details about 'object', use 'object??' for extra details.
In [1]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
In [2]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
0
0
0


frame
Terminal de IPython
Terminal 2/A
In [4]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
0
0
0


frame
m c d u
0
0
0
Terminal de IPython
Terminal 2/A
In [5]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
0

0

0


frame
m c d u
0
0
0
Terminal de IPython
Terminal 2/A
In [8]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')

000


frame
m c d u
1
1
1
Terminal de IPython
Terminal 2/A
In [9]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
1

41

1


frame
m c d u
1
1
1
Terminal de IPython
Terminal 2/A
In [12]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
1

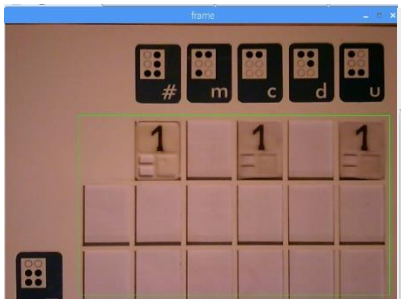
1

1

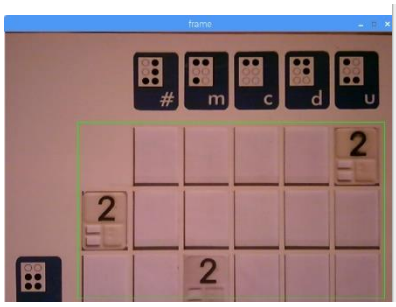

frame
m c d u
1
1
1
Terminal de IPython
Terminal 2/A
In [14]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
1

1

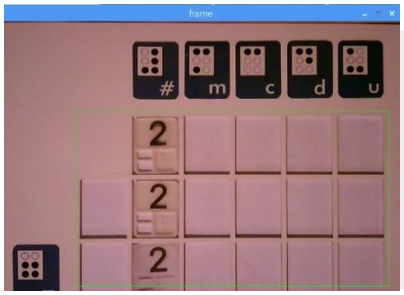
1



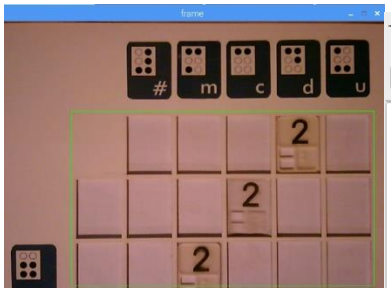

```
Terminal de IPython
Terminal 3/A
In [1]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
1 1 1
----------------------------------------

----------------------------------------

----------------------------------------
```



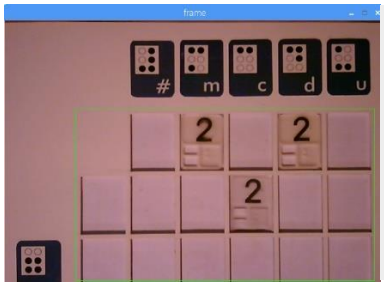

```
Terminal de IPython
Terminal 3/A
----------------------------------------

In [2]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
Reloaded modules: PIL.GifImagePlugin, fractions, PIL.GimpPaletteFile, PIL.JpegPresets,
PIL.ImageSequence, PIL.PaletteFile, PIL.TiffImagePlugin, PIL.GimpGradientFile,
PIL.PngImagePlugin, PIL.JpegImagePlugin, PIL.PpmImagePlugin, PIL.ImageFile, PIL.TiffTags,
PIL.ImagePalette, PIL.ImageChops, PIL.ImageColor, PIL.BmpImagePlugin
2
----------------------------------------

----------------------------------------
 2
----------------------------------------
```



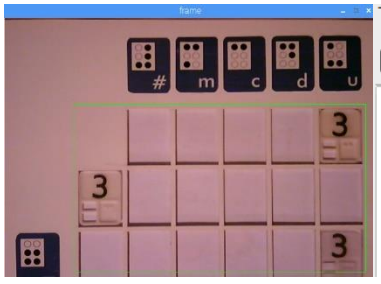

```
----------------------------------------

In [7]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
2
----------------------------------------
2
----------------------------------------
 2
----------------------------------------
```



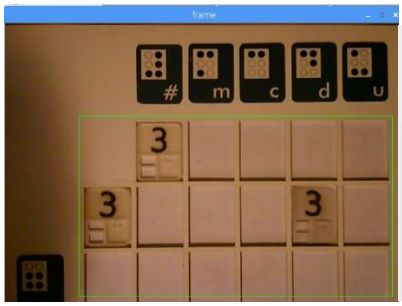

```
Terminal de IPython
Terminal 3/A
In [8]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
2
----------------------------------------
 2
----------------------------------------
 2
----------------------------------------
```




```
Terminal de IPython
Terminal 3/A
In [9]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
 2 2
----------------------------------------

----------------------------------------

----------------------------------------
```



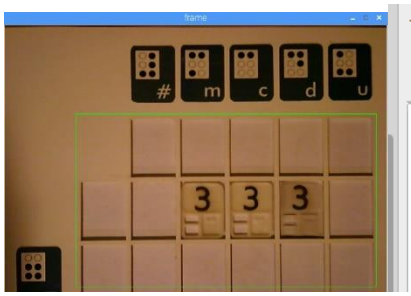

```
Terminal de IPython
Terminal 3/A
In [11]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
 3
----------------------------------------
3
----------------------------------------
  3
----------------------------------------
```



Terminal de IPython

Terminal 3/A

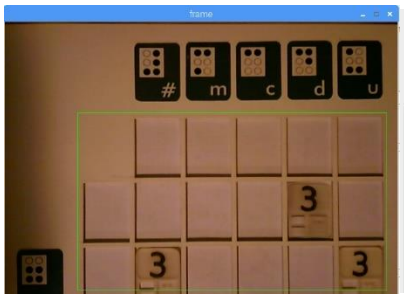
```
In [12]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
3
--------------------------------------------
3 3
--------------------------------------------

--------------------------------------------
```



Terminal de IPython

Terminal 3/A

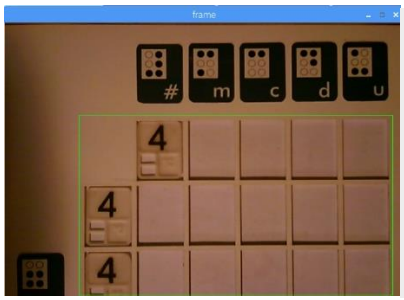
```
In [14]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')

--------------------------------------------
  3 3 3
--------------------------------------------

--------------------------------------------
```



Terminal de IPython

Terminal 3/A

```
In [16]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')

--------------------------------------------
3
--------------------------------------------
 3  3
--------------------------------------------
```



Terminal de IPython

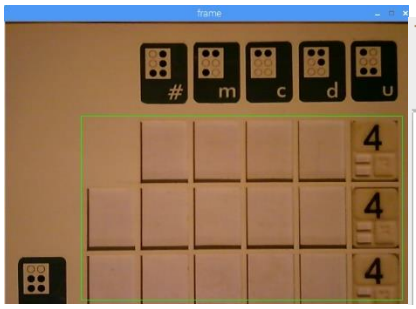
Terminal 3/A

```
In [18]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
4
--------------------------------------------
4
--------------------------------------------

--------------------------------------------
```

Terminal de IPython

Terminal 3/A

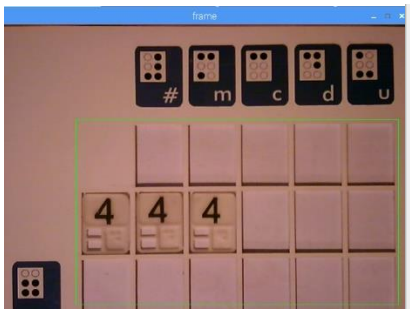
```
In [19]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
4
------------------------------------------
 4
------------------------------------------

------------------------------------------
```



Terminal de IPython

Terminal 3/A

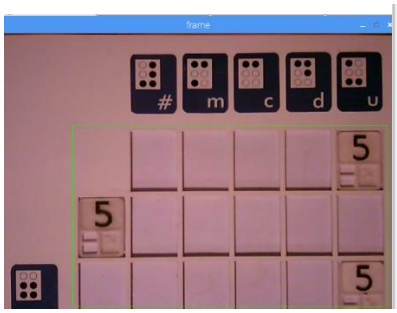
```
In [20]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
 4
------------------------------------------
  4
------------------------------------------
  4
------------------------------------------
```



Terminal de IPython

Terminal 3/A

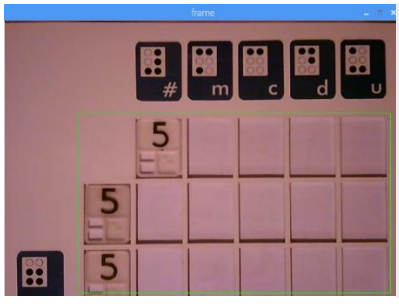
```
In [21]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')

------------------------------------------
44 4
------------------------------------------

------------------------------------------
```



Terminal de IPython

Terminal 3/A

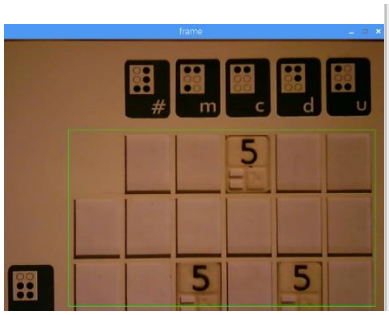
```
In [22]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
 5
------------------------------------------

------------------------------------------
  5
------------------------------------------
```



Terminal de IPython

Terminal 3/A

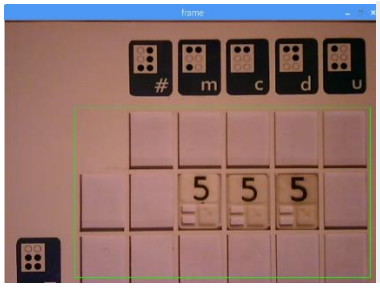
```
------------------------------------------
In [26]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
5
------------------------------------------

------------------------------------------

------------------------------------------
```



Terminal de IPython

Terminal 3/A

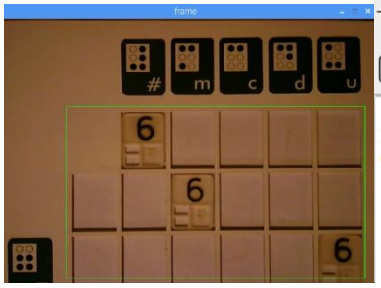
```
In [27]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
 5
----------------------------------------

----------------------------------------
  5
----------------------------------------
```



Terminal de IPython

Terminal 3/A

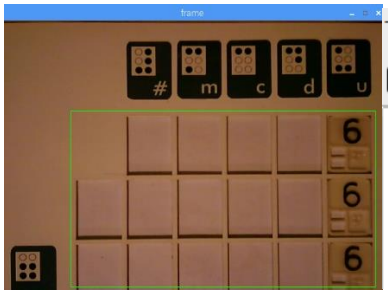
```
In [29]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')

----------------------------------------
 5 5 5
----------------------------------------

----------------------------------------
```



Terminal de IPython

Terminal 3/A

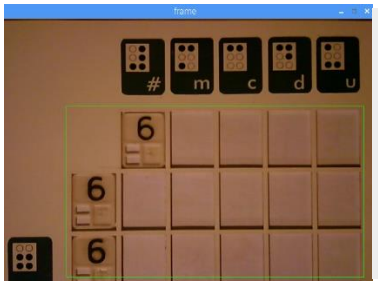
```
In [30]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
6
----------------------------------------
 6
----------------------------------------
  6
----------------------------------------
```



Terminal de IPython

Terminal 3/A

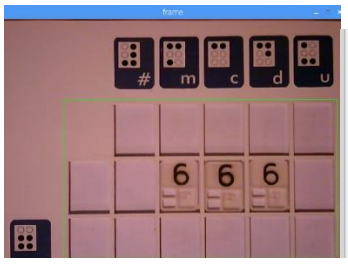
```
In [31]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
 6
----------------------------------------
  6
----------------------------------------
  6
----------------------------------------
```



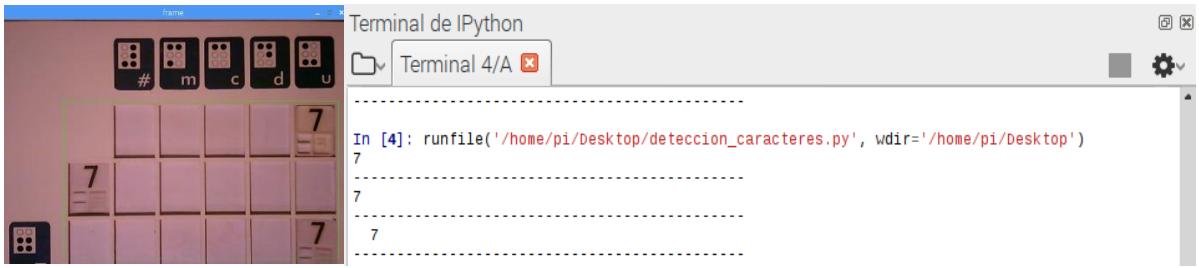
Terminal de IPython

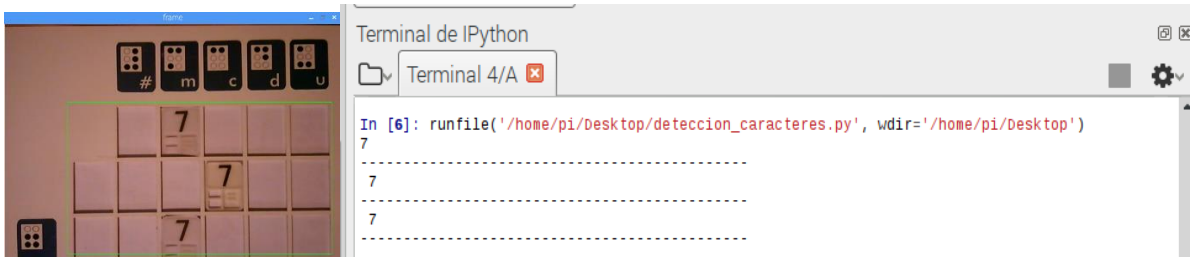
Terminal 3/A

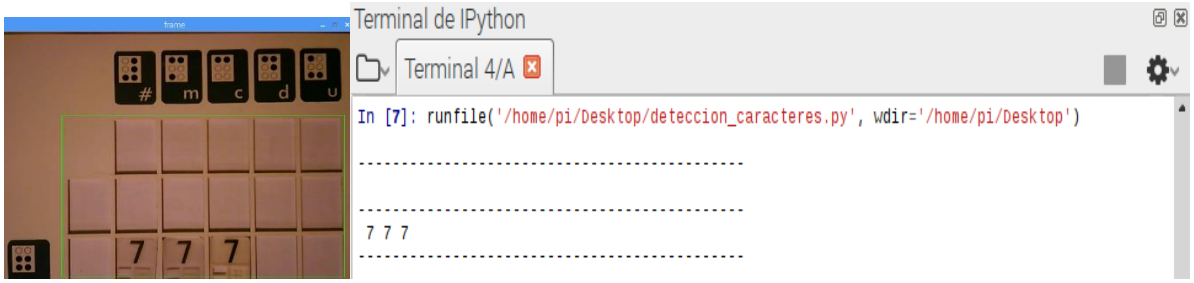
```
In [33]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
6
----------------------------------------
6
----------------------------------------
6
----------------------------------------
```

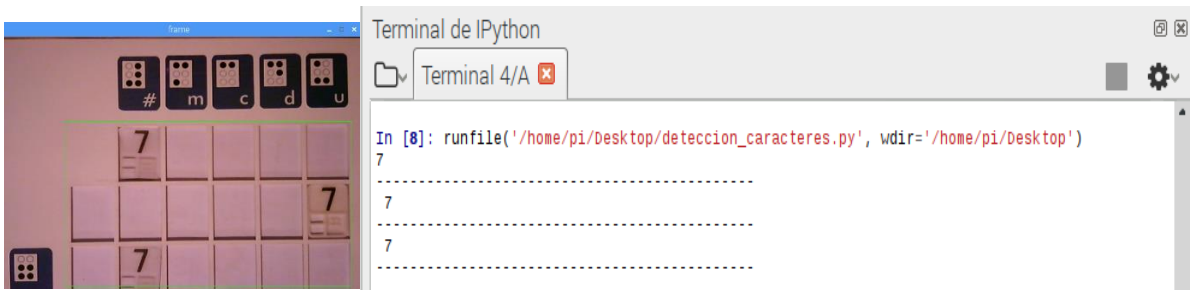


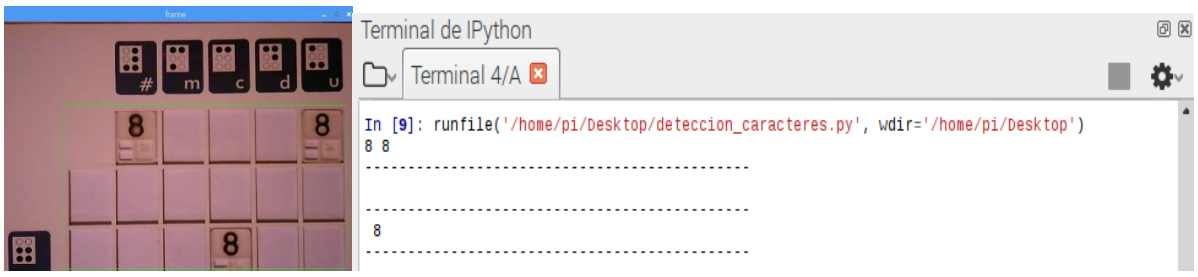

```
In [1]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')

----------------------------------------
 6 6 6
----------------------------------------

----------------------------------------
```


frame
m c d u
7 7 7
Terminal de IPython
Terminal 4/A
In [4]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
7
7
7

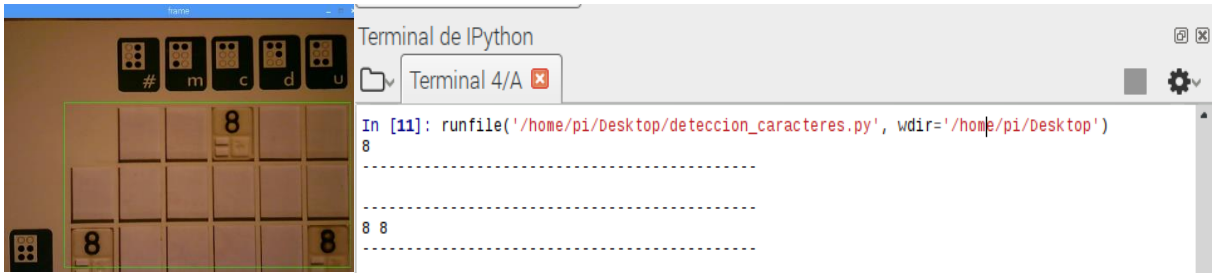

frame
m c d u
7 7 7
Terminal de IPython
Terminal 4/A
In [6]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
7
7
7

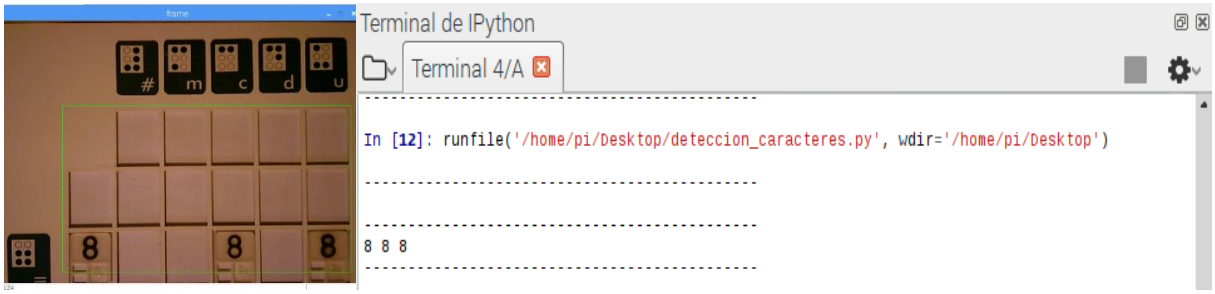

frame
m c d u
7 7 7
Terminal de IPython
Terminal 4/A
In [7]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
7 7 7

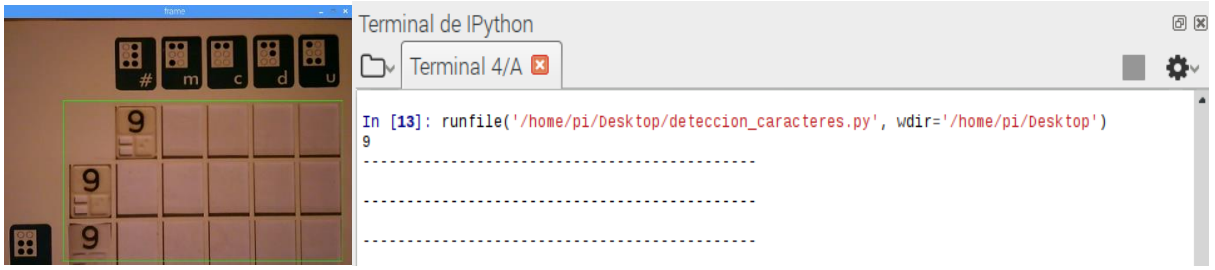

frame
m c d u
7 7
Terminal de IPython
Terminal 4/A
In [8]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
7
7
7

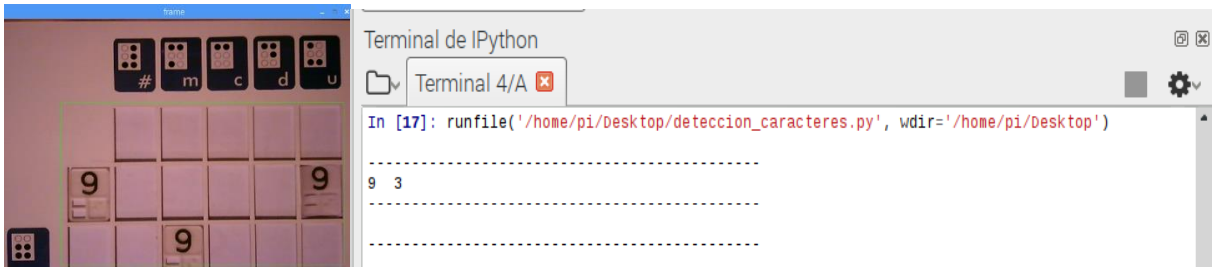

frame
m c d u
8 8 8
Terminal de IPython
Terminal 4/A
In [9]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
8 8
8

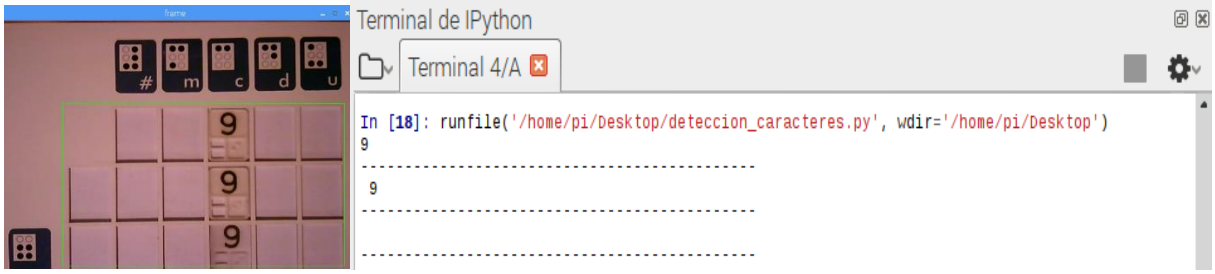

frame
m c d u
8 8 8
Terminal de IPython
Terminal 4/A
In [10]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
8
8
8

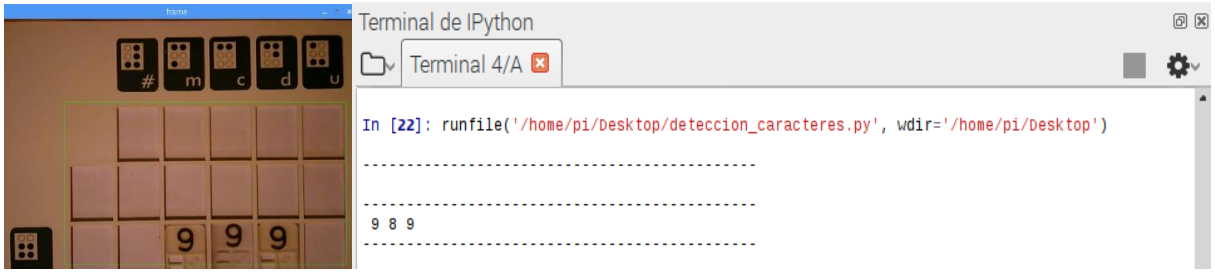

frame
Terminal de IPython
Terminal 4/A
In [11]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
8
8 8

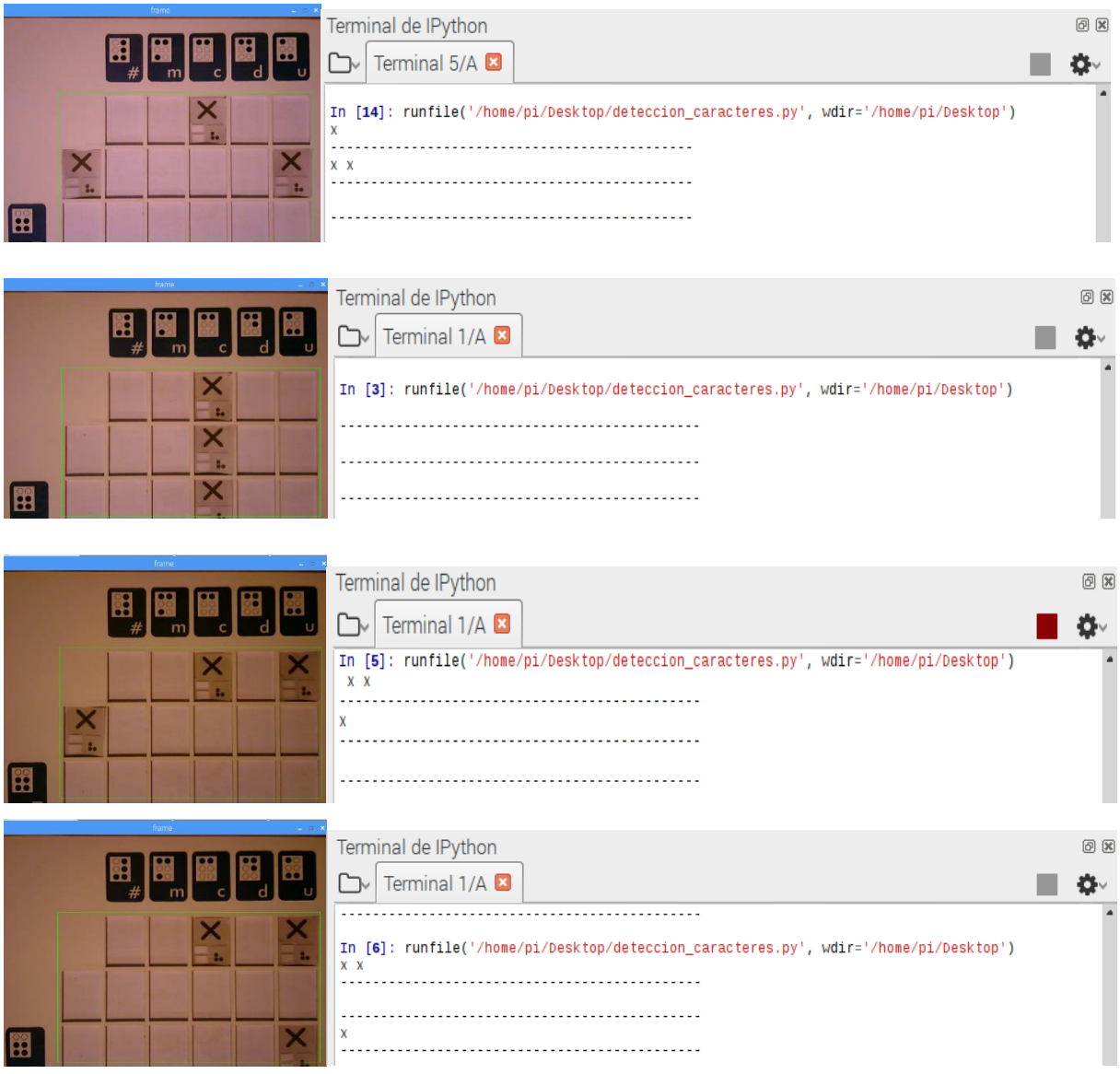

frame
Terminal de IPython
Terminal 4/A
In [12]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
8 8 8


frame
Terminal de IPython
Terminal 4/A
In [13]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
9


frame
Terminal de IPython
Terminal 4/A
In [17]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
9 3


frame
Terminal de IPython
Terminal 4/A
In [18]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
9
9


frame
Terminal de IPython
Terminal 4/A
In [22]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
9 8 9


frame
m c d u
Terminal de IPython
Terminal 5/A
In [14]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
x
x x
Terminal de IPython
Terminal 1/A
In [3]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
Terminal de IPython
Terminal 1/A
In [5]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
x x
x
Terminal de IPython
Terminal 1/A
In [6]: runfile('/home/pi/Desktop/deteccion_caracteres.py', wdir='/home/pi/Desktop')
x x
x