Facultad de Ingeniería

Escuela de Ingeniería Civil Mecatrónica

Automatización de trampa delta para *Cydia Pomonella* o polilla de la manzana

Memoria para optar al título de
Ingeniera Civil Mecatrónica

Profesor Guía:
Matthew Bardeen

Paula Isabel Soto Pavez
Curicó - Chile
2021

CONSTANCIA

La Dirección del Sistema de Bibliotecas a través de su encargado Biblioteca Campus Curicó certifica que el autor del siguiente trabajo de titulación ha firmado su autorización para la reproducción en forma total o parcial e ilimitada del mismo.

Curicó, 2022

Automatización de trampa delta para *Cydia Pomonella* o polilla de la manzana

Paula Isabel Soto Pavez

Abril, 2021
Paula Isabel Soto Pavez

Resumen

La polilla de la manzana es una de las mayores plagas que afectan a pomáceas en Chile, encontrándose distribuida a lo largo del país. Esta plaga ataca cuando aún es una larva, provocando pérdidas que pueden llegar al 10 % en la producción de manzanas. Actualmente, el control de la plaga se realiza de forma manual mediante trampas de feromona sexual sintética que atraen a los machos hacia una superficie pegajosa antes de que se produzca la ovipostura. Luego, se realiza un monitoreo de la cantidad de polillas atrapadas, las cuales son contadas por una persona en campo.

Para lograr automatizar este proceso, se propone un sistema de captura de imágenes hecho a partir de un módulo ESP32-CAM y una cámara OV2640 ubicados dentro de la trampa, el cual registrará imágenes y junto a sensores de temperatura, obtener datos los cuales se enviarán a un centro de monitoreo para su posterior análisis. Las imágenes obtenidas serán procesadas para que se determine de forma automatizada la cantidad de polillas que hay en la trampa.

Dedicado a mis padres y abuelo...

Agradecimientos

A lo largo de mis años en la Universidad, he tenido que enfrentarme a diversos desafíos de los cuales feliz y orgullosamente he salido victoriosa. Hoy, he terminado el último de esos desafíos, lo que no hubiera sido posible sin el apoyo de importantes personas que me acompañaron en este camino. Es por esto que quiero agradecer a los profesores que en todos estos años me han entregado sus enseñanzas, en especial al profesor Matt Bardeen, quién con sus conocimientos y apoyo me guió a través de todas las etapas de este proyecto. También quiero agradecer al profesor Luis Concha por brindarme todo su apoyo y por confiar en mí para la realización de este proyecto.

Por otro lado, quiero agradecer a las personas más importantes en mi vida. A mi madre Sandra Pavez y a mi padre Rodrigo Soto por apoyarme y acompañarme en cada momento y decisión que he tomado y sin quienes no hubiera podido lograr todo lo que he logrado. También quiero agradecer a mi tata Juan Pavez quién ha estado siempre de mi lado, ha sido mi cómplice en mis travesuras y me ha brindado todo su amor y apoyo incondicional, y a mi nana Mireya Retamal quien, aunque ya no se encuentre entre nosotros, me ha acompañado, cuidado y ha estado siempre en mi pensamiento y corazón.

Finalmente, agradezco a mis amigos, a aquellos que han estado conmigo desde siempre y a aquellos que conocí en la Universidad, por ayudarme siempre que lo necesité y por todas las risas y los buenos momentos que compartimos en estos años.

Índice

Índice de figuras

Índice de tablas

Resumen de capítulos

1. **Introducción**

   En este capítulo se introduce al proyecto al presentar temas del estado del arte y trabajos previos realizados en el ámbito de este proyecto. Se plantea una hipótesis y los objetivos, tanto general como específicos. Se definen los alcances y las limitaciones del proyecto y la metodología a seguir para cumplir los objetivos propuestos.

2. **Desarrollo**

   En este capitulo se muestran los procedimientos a realizar para llevar a cabo el proyecto, se realiza la definición y descripción de los materiales a utilizar y se realizan las cotizaciones correspondientes.

3. **Resultados experimentales**

   En este capítulo se muestran las simulaciones realizadas, los algoritmos y sus correspondientes diagramas de flujo. Se presentan los resultados obtenidos a partir de las simulaciones y de las pruebas realizadas tanto en condiciones ideales como en condiciones reales en campo.

4. **Conclusión**

   Se presentan las conclusiones del proyecto, junto con análisis de los resultados y posibles trabajos futuros.

1. Introducción

1.1. Introducción general

La polilla de la manzana *(Cydia Pomonella)* es una de las mayores plagas que afectan a pomáceas y nogales en Chile y se encuentra distribuida a lo largo del país en todas las zonas de producción de estos frutos, siendo la manzana la principal afectada. La forma en que esta polilla ataca a la fruta es en la etapa de desarrollo, cuando aún es una larva, introduciéndose al fruto provocando pérdidas de producción y calidad que pueden llegar a ser entre 5 % y 10 % de la producción total de manzanas [1].

El manejo actual de esta plaga se realiza casi exclusivamente con la utilización de pesticidas aplicándose según calendario, por lo que los cultivos dependen del uso permanente y reiterado de éstos. Debido a estos antecedentes, se busca una manera más eficiente de controlar esta plaga, como son el monitoreo, el uso de pesticidas más seguros para el ambiente y para la salud de las personas y la necesidad de un menor número de aplicaciones de los plaguicidas para evitar impactos negativos en la producción [1].

El método más utilizado para el monitoreo de la plaga es mediante trampas delta de feromona sexual sintética, como la mostrada en la Figura 1. La trampa delta consiste en una estructura con forma de prisma triangular construida generalmente en polipropileno o cartón plastificado. En su base cuenta con una superficie pegajosa hacia la que el macho es atraído mediante feromonas sexuales. El objetivo es atraer a los machos adultos hacia ella y que se queden atrapados antes de que se realice la ovipostura. Estas trampas deben estar separadas como máximo unos 100 metros entre ellas y deben ser instaladas en la parte superior del árbol, además de realizar un cambio de la feromona cada dos meses, ya que la cantidad de ésta va disminuyendo con el tiempo [2].

Fig. 1: Trampa delta utilizada en la actualidad.
Fuente: Entomopraxis.

El monitoreo se basa en el conteo y registro de polillas atrapadas. La importancia de realizar un monitoreo de la polilla radica en que, en base a las capturas de machos, es posible conocer las fluctuaciones poblacionales de los adultos. Gracias a esto, es posible realizar una acertada estimación de cuándo ocurrirá la ovipostura y la posterior eclosión de huevos, siendo éste el mejor momento para controlar la plaga, con tal de realizar las aplicaciones de insecticidas de forma más eficiente [1].

El principal problema con esta metodología de monitoreo es que se debe hacer de forma manual, ya que es una persona en campo la encargada de realizar el conteo de las polillas y el registro de datos. Para mejorar el monitoreo, se propone un sistema de captura de imágenes que se encontrará ubicado dentro de la trampa y que registrará fotografías y las enviará hacia un centro de monitoreo para su posterior análisis. La automatización de la trampa se realizará utilizando un sensor de temperatura y una cámara que capture las fotografías para realizar la contabilización de las polillas de forma remota. La transmisión de estos datos se realizará vía Wi-Fi.

1.2. Estado del arte

1.2.1. Descripción y biología de la polilla

El adulto de *Cydia Pomonella* es una mariposa pequeña, de color grisáceo y posee rayas de color cobrizo en sus alas, su envergadura es de unos 17 mm, como se ve en la Figura 2. Esta especie se caracteriza por volar durante el atardecer, que es cuando se presenta su mayor actividad, mientras durante el día se encuentra inactiva y protegida en los árboles [3].

Fig. 2: *Cydia Pomonella*, comúnmente conocida como polilla de la manzana.
Fuente: Wikipedia.

La hembra pone los huevos en los frutos y las hojas. La larva recién eclosionada penetra el fruto, desarrollándose dentro de éste y dejándolo considerablemente dañado. Después de aproximadamente cinco semanas, la larva ha completado su desarrollo, alcanzando un tamaño de 1.5 a 1.8 mm, presentando un color negro en la cabeza y un cuerpo dorado [3].

1.2.2. Procesamiento digital de imágenes

El procesamiento digital de imágenes es el área de la ingeniería que se encarga de la extracción de mediciones, datos o información contenida en una imagen, incluyendo las

técnicas cuyo objetivo es facilitar la búsqueda e interpretación de la información contenida en ellas [4]. El procesamiento y análisis de imágenes se ha desarrollado en respuesta a tres grandes problemas:

- Digitalización y codificación de imágenes que facilite la transmisión, representación y almacenamiento de las mismas.
- Restauración de una imagen para interpretar de mejor manera su contenido.
- Descripción y segmentación de imágenes para aplicaciones de visión robótica o visión artificial.

Dentro del procesamiento de imágenes se incluyen todos los algoritmos que ayuden a resaltar, agudizar, contrastar ciertos aspectos de la imagen, así como también los que ayudan a eliminar efectos no deseados en la imagen.

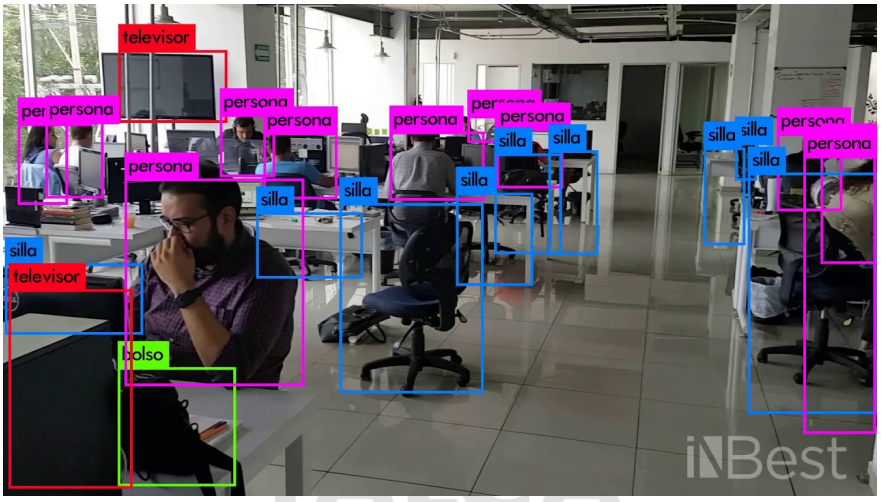


Fig. 3: Detección y reconocimiento de objetos en una imagen.
Fuente: iNBest.cloud.

La detección de objetos es una tecnología relacionada con la visión artificial y el procesamiento de imágenes cuyo objetivo es detectar objetos de una cierta clase como humanos, edificios, vehículos, etc., en vídeos o imágenes digitales. Los casos más complejos buscan la detección de caras y reconocimiento facial. La Figura 3 muestra un ejemplo de detección y reconocimiento de objetos.

1.2.3. Técnicas de monitoreo tradicionales

El monitoreo consiste en revisar periódicamente un cultivo para medir la densidad y estimar la distribución de plagas. Esta herramienta permite observar su evolución y así mismo dar el seguimiento oportuno para evitar repercusiones en la producción del cultivo [5]. Si bien el control químico resulta una poderosa herramienta para mantener en raya a la plaga, el uso inadecuado de plaguicidas ha introducido cantidades excesivas de sustancias químicas al ecosistema, alterándolo y generando la aparición de individuos resistentes. Por esto, el monitoreo permite una correcta gestión de los tratamientos, aplicando control químico adecuado y minimizando los riesgos derivados del uso indiscriminado de pesticidas [6].

La importancia de realizar el monitoreo radica en dar al productor tiempo para resolverlos antes de que el cultivo sufra graves daños, permitiendo el uso oportuno y eficiente de los insumos para el manejo de la plaga. La forma tradicional de realizar el monitoreo consiste en recorrer con regularidad la plantación y revisar detenidamente plantas al azar en busca de enfermedades o heridas en las hojas y los frutos [7].

Como se presentan circunstancias en las que el monitoreo en campo no se puede realizar existe el uso de trampas de monitoreo que atraen a los insectos mediante un atrayente o cebo, entregando información acerca de los períodos de mayor y menor abundancia y permitiendo conocer el comportamiento y dinámica de la plaga en una zona particular, logrando una mejor determinación de cuando se debe controlar la plaga [8].

1.2.4. Técnicas de monitoreo avanzadas

En la actualidad, existen métodos de realizar los monitoreos de forma más avanzada. El manejo integrado de plagas o MIP se define como el uso de una serie de medidas de control, tanto culturales, biológicas, químicas, mecánicas y físicas, que pretenden reducir las poblaciones de plagas que afecten a un cultivo para evitar que causen daños económicos y se permita su comercialización, sin causar efectos negativos a los habitantes, la fauna o el ecosistema [9].

El MIP se basa en el uso inteligente de varios métodos de control, con la intención de disminuir el uso de insecticidas. Algunos de estos métodos de control son:

- Control natural: permite y fomenta los enemigos naturales de la plaga.
- Control biológico: se relaciona con la liberación de parasitoides y depredadores para el control de una determinada plaga.
- Control químico: uso de insecticidas químicos para reprimir las plagas solo en casos necesarios, cuando la plaga haya causado daño irreversible al cultivo.
- Control cultural: se basa en realizar una buena preparación del terreno, rotación de cultivos, periodos cortos de siembra y uso de variedades resistentes a la plaga.
- Control físico: métodos mecánicos para remover la plaga. En cultivos de café se ha validado el uso de aspiradoras mecánicas para recoger los frutos infestados.

La agricultura de precisión es otra de las técnicas de monitoreo utilizadas actualmente. Consiste en la aplicación de tecnologías y principios para gestionar campos agrícolas en base a la observación, la medida y la actuación frente a la variabilidad de los cultivos. Requiere de tecnologías formadas tanto por el Sistema Global de Navegación por Satélite, sensores e imagen satelital y aerotransportada. La agricultura de precisión provee la habilidad de utilizar recursos más eficientemente, incluyendo semillas, fertilizantes, pesticidas y riegos [10]. Además, la agricultura de precisión resulta ser beneficioso económicamente, ya que al conocer exactamente dónde los recursos son requeridos, se utilizan de forma más eficiente [11].

Como ejemplos de esta técnica se encuentran el desarrollo de un sistema de monitoreo de plagas remoto, el cual se ha diseñado con información que se va obteniendo en tiempo real construido en base a sensores inalámbricos. Los usuarios pueden obtener información histórica y acceder a ella desde un sitio web. El sistema ha sido probado en campo desde el

año 2008, y ha demostrado ser capaz de monitorear parámetros ambientales y de población de la plaga en tiempo real [12].

Aplicaciones de monitoreo en tiempo real aplicando la agricultura de precisión se han desarrollado para la inspección de los tratamientos que se entregan a un campo, demostrando que pueden identificarse numerosos tipos de enfermedades asociadas a las plantas y plagas usando sensores para mantener un constante monitoreo en los cultivos [13].

La utilización de tecnologías de agricultura de precisión, empleando una red inalámbrica de sensores para monitorear el clima y trampas para insectos en plantaciones de duraznos. Aquí, se diseñaron e implementaron tres módulos: un sistema de adquisición de datos climáticos y de población de la plaga, un sistema de adquisición de datos de enfermedades en los árboles y un sistema de procesamiento y visualización de la información [14].

1.3. Trabajos previos

Diversos trabajos se han realizado incorporando las nuevas tecnologías en el control y monitoreo de las plagas y cultivos. La mayoría de estos trabajos se enfocan en la mejora de la calidad de los cultivos a partir de estudios realizados al fruto, ya sea para determinar si han sido atacados por plagas o para comprobar el buen uso de los planes de siembra o riego.

Investigadores de la Universidad Pedagógica y Tecnológica de Colombia, crearon un sistema que emplea una red inalámbrica de sensores para monitorear el clima y trampas de insectos, un sistema de adquisición de datos y un sistema de procesamiento y visualización. El sistema adquiere variables climáticas de velocidad y dirección del viento, temperatura, humedad relativa y pluviosidad, además de trampas de insectos que permiten el conteo de éstos. Esta información es transmitida mediante un sistema de transmisión basado en ZigBee. Otro sistema permite adquirir la información proveniente del conteo de hojas y frutos, los cuales se transfieren a un centro de seguimiento donde se almacenan y presentan al usuario la distribución espacial de la enfermedad. Finalmente, se crea una aplicación en software consistente en un programa que reúne la información de las distintas fuentes y las presenta de forma gráfica para una mejor visualización y entendimiento para el usuario [14].

Por otro lado, investigadores de la Universidad de California en Davis están utilizando sistemas de cámaras para detectar el estrés en los cultivos cuando están infestados de plagas. Además, se incluye el uso de sistemas de drones para monitorear los campos y el desarrollo de un sistema de rieles que puede ser utilizado dentro de los invernaderos. El principio fundamental detrás del uso de estas tecnologías se basa en el hecho de que los cultivos saludables reflejan la luz de forma distinta a los que están estresados [15].

Un sistema de detección y clasificación de defectos en las frutas mediante procesamiento digital de imágenes es creado para la inspección de los frutos y la identificación de cicatrices, rozaduras, manchas en la piel, surcos o estrías, las cuales determinan la calidad de la fruta separándolas en clases. Para ello, se entrena al sistema con distintas clases de la fruta para que sea capaz de clasificarlas, se toma una fotografía de la fruta y mediante algoritmos de segmentación se procesa la imagen y se obtienen sus características, se reconocen los patrones y se comparan para identificar en qué clase corresponde [16].

Como conclusión de esta sección se determina que la implementación de una mejora en las formas actuales de monitoreo mediante la utilización de sensores de temperatura y humedad y una cámara en conjunto con un algoritmo de procesamiento de imágenes es viable para determinar la cantidad de polillas atrapadas en una trampa y por ende, determinar el mejor

momento para atacar la plaga.

1.4. Hipótesis

Según los estudios realizados y basándose en algunos trabajos previos, se establece que es factible la implementación de un sistema de sensores, captura y posterior procesamiento de imágenes para determinar la cantidad de polillas de la manzana en una trampa delta y mejorar el control de la plaga.

1.5. Objetivos

1.5.1. Objetivo general

Automatizar el conteo de la polilla de la manzana en trampas delta para la predicción de la plaga, con respecto a técnicas de monitoreo tradicionales.

1.5.2. Objetivos específicos

- Desarrollar un sistema de monitoreo de polillas utilizando cámaras y sensores.
- Determinar la cantidad de polillas dentro de una trampa delta mediante el desarrollo de algoritmos que permitan su detección y conteo.
- Diseño y construcción de una estructura para una trampa basándose en el diseño actual de las trampas delta que pueda contener el sistema de cámara y sensores.
- Facilitar el acceso a la información obtenida a partir de los sensores mediante la creación de una interfaz de usuario.

1.6. Alcances y limitaciones

1.6.1. Alcances

- Creación y fabricación de la estructura de una trampa, la cual será impresa en 3D.
- Se implementará solo una trampa, a modo de prototipo.
- Debido a que la trampa se encontrará a la intemperie, se considera la utilización de filamentos ASA para la impresión.

1.6.2. Limitaciones

- Debido a que los módulos ESP32 funcionan vía Wi-Fi, debe existir una red Wi-Fi en campo para lograr la conexión.
- El rango de alcance promedio de una red Wi-Fi es de 35 metros aproximadamente por lo que, para cubrir una plantación real, se debe recurrir a amplificadores/repetidores de señal de largo alcance.
- Es posible que exista un pequeño error en el conteo de las polillas debido a las diferencias de luz presentes al momento de capturar las fotografías.

1.7. Metodología

Para poder cumplir con los objetivos especificados anteriormente, se deberán realizar las siguientes actividades:

1.7.1. Estudio sobre la polilla y metodologías de monitoreo

Debido a que se estará trabajando con polillas, es necesario conocer el comportamiento, morfología y reproducción de éstas para obtener resultados más eficientes al realizar el proyecto. Además, es necesario conocer los distintos métodos de monitoreo y control de la plaga que se utilizan actualmente para determinar un correcto mejoramiento de éstos.

1.7.2. Estudio teórico sobre algoritmos de procesamiento de imágenes

Se revisa bibliografía sobre los distintos métodos y algoritmos con los cuales se puede realizar el procesamiento de imágenes. Específicamente, se estudian algoritmos de conteo de objetos para realizar el conteo de las polillas. Algunos de estos algoritmos revisados son los algoritmos de detección de borde como Transformada de Hough, Método de Canny, detector de bordes Sobel, entre otros; y SimpleBlobDetector, el cual genera una lista de objetos encontrados a partir de una imagen binaria, detectando los píxeles negros y marcándolos como un conjunto, obteniendo un objeto en la imagen.

1.7.3. Estudio y acercamiento al lenguaje de programación *Python*

Para este proyecto, se determina la utilización del lenguaje de programación *Python* para la ejecución de los códigos y algoritmos a utilizar, debido a que la mayoría de la documentación revisada se encuentra escrita en este lenguaje; y la librería *OpenCV*, la cual está especialmente creada para aplicaciones de visión artificial. Debido a que dentro de la formación académica de la carrera de Ingeniería Civil Mecatrónica no se enseña *Python* como método de programación, será necesario el estudio de este lenguaje y la realización de pruebas y ejemplos antes de adentrarse a realizar el proyecto.

1.7.4. Diseño de una estructura en 3D utilizando software *Inventor*

Para la estructura principal de la trampa, se creará un diseño utilizando el software *Inventor* basado en las trampas delta utilizadas actualmente y en la necesidad de incorporar la cámara y sensor dentro de la misma. Además, se diseñará pensando en que se ubicará en exteriores, por lo que debe soportar diversas condiciones climáticas, y pensando también en cómo se anexará la trampa al árbol para que cumpla su función.

1.7.5. Identificación de los componentes a utilizar

De acuerdo a lo revisado en la sección Estado del Arte y a las necesidades que pretende satisfacer este proyecto, se identifican los componentes que cumplan con los requerimientos establecidos para lograr los objetivos. Debido a que es crucial conocer la temperatura ya que ésta afecta a la reproducción de la polilla, es necesaria la utilización de un sensor de temperatura. Otros sensores que midan otras variables climatológicas podrían emplearse de igual

manera, ya que mientras más características se tengan del ambiente, mejor se podrá monitorear a la plaga. En este proyecto, se contempla la utilización de un sensor de temperatura y humedad DHT11.

Otra de las características principales que presenta este proyecto es la de monitorear la cantidad de polillas que se encuentren en la trampa. Para ello, se necesita de una cámara ubicada dentro de la trampa que capture fotografías y se comunique de manera inalámbrica con un PC que realice el conteo. De acuerdo a estos antecedentes, se identifica el módulo ESP32-CAM como adecuado para realizar este proyecto, ya que corresponde a un módulo de pequeña dimensión que funciona con WiFi, por lo que la comunicación con el PC puede ser realizada mediante WiFi, mientras los dos dispositivos se encuentren conectados a la misma red.

Para la estructura de la trampa, se realiza el diseño para posteriormente ser impresa en 3D. Se quiere realizar de esta forma ya que se busca que la estructura sea rígida y duradera. Debido a que se ubicará a la intemperie, debe estar impresa en un material que presente resistencia frente a distintas condiciones climáticas y buenas características mecánicas. Es por esto que se establece que el mejor filamento de impresión para la estructura es el filamento ASA, debido a su robustez mecánica y su alta resistencia térmica, a rayos UVA, al agua y a sustancias químicas.

2. Desarrollo

En cuanto al desarrollo del proyecto, se presentan las siguientes etapas que deberán ser realizas para lograr en conjunto el producto final:

2.1. Circuitos electrónicos

En primer lugar, se realizarán todas las conexiones requeridas en el sistema, estas son las del módulo ESP32-CAM con el sensor de acuerdo a los distintos datasheets de los componentes. El diagrama de conexionado se muestra en la Figura 4. También se deberá tener en cuenta que, debido a que la trampa estará ubicada en los árboles, requerirá una fuente de alimentación externa. Para esto, se plantea utilizar un panel solar junto con un cargador de baterías LiPo diseñado para Arduino. El tamaño y ubicación de los componentes dentro de la estructura debe estar definido antes de realizar el modelo 3D, ya que estos deben calzar dentro de la estructura sin que interfieran con la efectividad de la trampa. Después de realizar las conexiones y comprobar que todo está en orden, se realiza el ensamblado del sistema electrónico en la estructura.

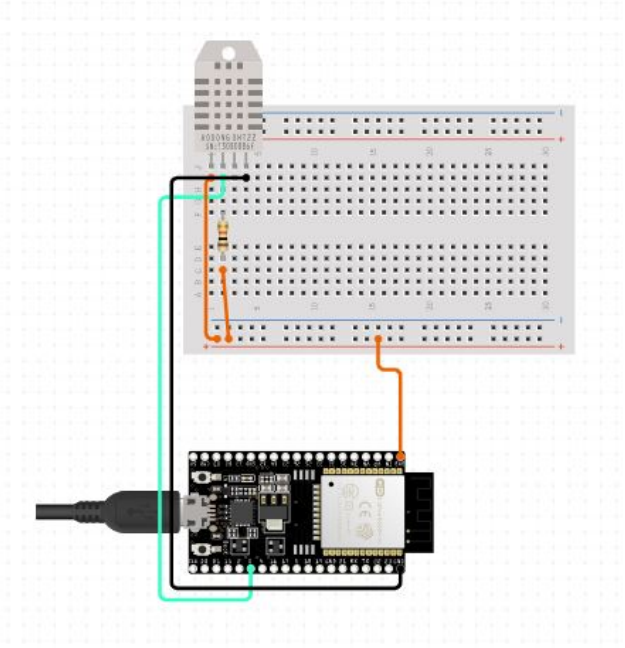


Fig. 4: Circuito de conexionado del módulo ESP32-CAM con el sensor DHT11.
Fuente: Elaboración propia.

2.2. Diseño 3D

Se deberá realizar un diseño de la estructura en algún software 3D. Para este paso, se utilizará el software *Inventor*. En este punto, se deberán tener en consideración las dimensiones de la estructura, la forma en la que posteriormente será ensamblada y cómo se organizará el sistema electrónico dentro de ésta. Una vez realizado el diseño, se procederá a realizar la

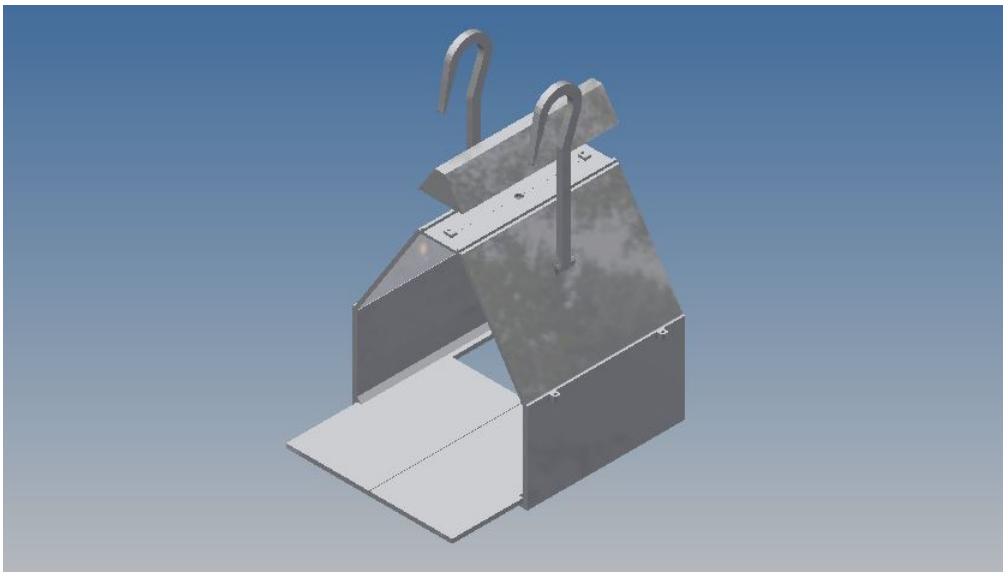


Fig. 5: Estructura de la trampa realizada en *Inventor*.
Fuente: Elaboración propia.

impresión de este, luego se ensamblarán las partes de acuerdo al diseño. En la Figura 5 se aprecia el diseño realizado de la trampa. El plano se presenta en el Anexo 1.

Como se observa en la Figura 5, la estructura consiste de tres partes principales: el cuerpo, la tapa y la base de la estructura. El cuerpo presenta una forma "pentagonal" y unos ganchos en su parte superior para poder colgarse la trampa de una rama del árbol o de un poste cercano. El circuito de la cámara y sensor se ubicará en la parte superior de la trampa, la tapa se encargará de mantenerlo resguardado de aves, insectos y diversas condiciones climáticas. Esta tapa puede ser completamente removible para acceder al circuito. La base de la trampa también es removible para poder colocar y quitar la superficie pegajosa en la que se atraparán las polillas.

El diseño realizado está basado tanto en el diseño actual de las trampas delta como en el ángulo de captura de la cámara. De acuerdo a esto, la trampa tiene las siguientes dimensiones: 200 mm de ancho, 180 mm de largo y 246 mm de alto, sin contar los ganchos. Además, esta estructura la componen nueve piezas, diseñadas así por dos motivos: en primer lugar, para que las piezas quepan dentro del área de impresión de la impresora 3D y también para que pueda ser desmontada si ya no se necesita y pueda ser guardada fácilmente.

2.3. Programación de algoritmos

Para que la cámara pueda tomar las fotografías, debe ser programada primero. Una de las ventajas de utilizar el módulo ESP32 es que se puede programar en la IDE de *Arduino*. Luego, se debe desarrollar el código para que la cámara capture las fotografías y las guarde en el PC para su posterior procesamiento. Debido a que el sensor también está conectado al módulo ESP32, la programación de éste se realiza en *Arduino* también. Luego, utilizando el software Visual Studio Code para programar con *Python*, se desarrollan códigos que permiten tomar las variables obtenidas del sensor y ordenarlas dentro de una base de datos accesible desde la aplicación de interfaz de usuario. En cuanto al procesado de las imágenes, se utiliza la

librería *OpenCV* para convertir las imágenes y modificarlas para extraer sus características.

2.3.1. Visualización de imágenes en navegador web

La Figura 6 presenta un diagrama de flujo que explica el algoritmo de visualización de imágenes en un navegador web y su posterior captura y guardado en el PC. Estos códigos se presentan en los Anexos 2 y 3.

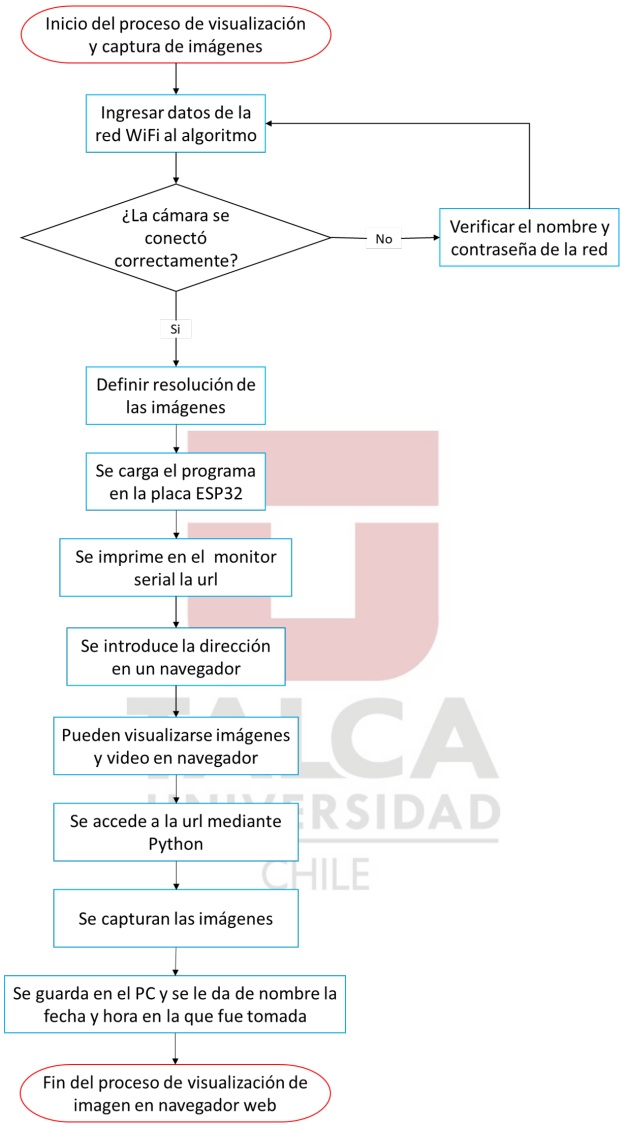


Fig. 6: Diagrama de flujo algoritmo visualización de imágenes en navegador web.
Fuente: Elaboración propia.

El primer paso para iniciar la captura de imágenes consiste en conectar la cámara a una red WiFi, introduciendo en el código realizado en Arduino, el nombre y la contraseña de la red. Luego se define la resolución de las imágenes y se carga el programa a la placa. En el monitor serial, se imprime la url que debe introducirse en el navegador para visualizar las imágenes. Esta dirección corresponde a la dirección IP de la cámara más la resolución definida, tal como se observa en la Figura 7.

```
CAMERA OK
http://192.168.1.43/cam-hi.jpg
```

Fig. 7: Captura del monitor serie que muestra la url de la cámara.
Fuente: Elaboración propia.

Luego de comprobar que la cámara se conecta a la red y que funciona correctamente, la url obtenida anteriormente se ingresa en un código Python que permite la captura de imágenes cada cierto tiempo definido. Este código accede a la url y realiza una captura de lo que está viendo la cámara en ese instante, para posteriormente guardar esta captura dentro del computador.

2.3.2. Obtención de datos del sensor

La Figura 8 presenta un diagrama de flujo que explica el proceso de obtención de datos desde el sensor a la placa. Estos códigos se presentan en los Anexos 4 y 5.

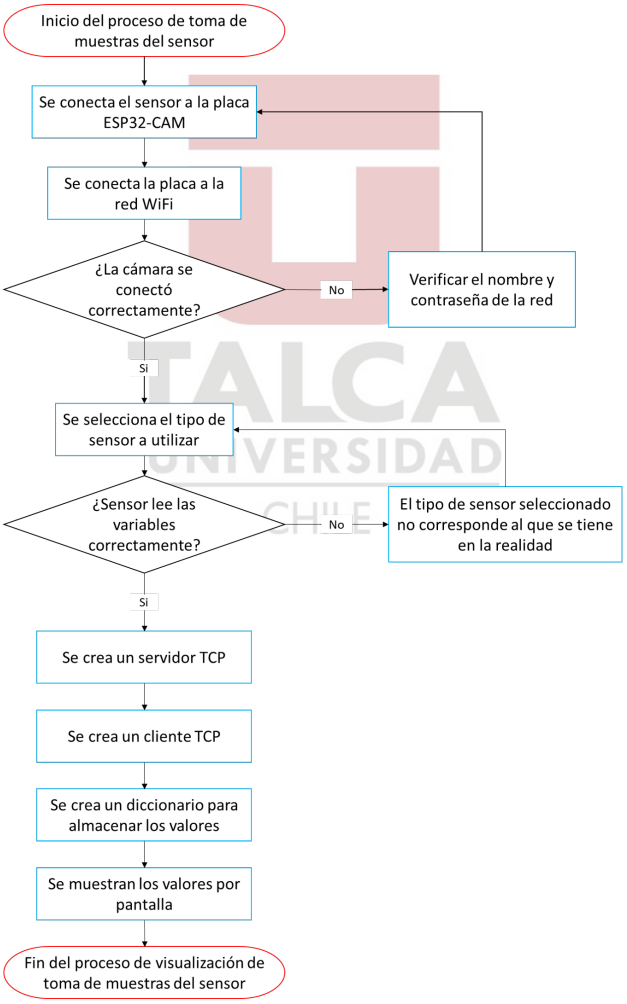


Fig. 8: Diagrama de flujo para la obtención de datos del sensor.
Fuente: Elaboración propia.

En primer lugar, se realiza un código en Arduino para programar la placa junto con el sensor. Para esto, se debe conectar la placa a la red WiFi del mismo modo que en el código anterior, introduciendo el nombre y contraseña de la red. Luego, se determina el tipo de sensor que se está utilizando, ya que existen dos tipos de sensor DHT: el DHT11 y el DHT22, para este caso se utiliza el DHT11. El sensor lee las variables de temperatura y humedad gracias a la librería "DHT", creada específicamente para trabajar con los sensores DHT11 y DHT22.

A continuación, se crea un servidor TCP, el cual almacena los datos obtenidos para que un cliente pueda pedirlos. Esto se hace desde el código Python.

Para mostrar los valores por pantalla, se creo un diccionario que almacena los valores recibidos y los actualiza cada cierto tiempo determinado. Este tiempo se define igual al tiempo de captura de las imágenes para así tener la temperatura y humedad que existe en el momento de la captura.

2.3.3. Conteo de objetos

La Figura 9 presenta un diagrama de flujo que muestra el proceso de conteo de los objetos presentes en una imagen. Este código puede encontrarse en el Anexo 6.

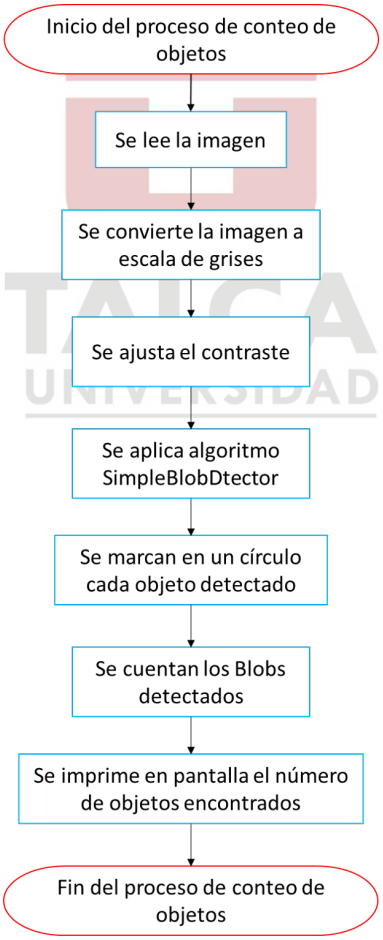


Fig. 9: Diagrama de flujo para conteo de objetos en una imagen.
Fuente: Elaboración propia.

Para realizar el conteo, en primer lugar se debe leer la imagen para realizar su procesado. Para esto se utiliza la librería OpenCV de Python, la cual está especialmente realizada para trabajar con procesamiento de imágenes.

Para comenzar el proceso, primero se convierte la imagen a escala de grises para determinar mejor sus características. Luego, se ajusta el contraste para que los objetos más oscuros resalten del fondo y en este punto se aplica el algoritmo llamado SimpleBlobDetector, incluido en la librería OpenCV. Este algoritmo detecta grupos de píxeles negros en la imagen marcándolos como un conjunto, obteniendo así un objeto. Finalmente, se realiza una cuenta de los Blobs detectados para saber el número de objetos que se encontraron en la imagen.

2.4. Materiales

Para la realización del proyecto se necesitarán los siguientes componentes, de los cuales se presentan algunas de sus características. Además, se muestra una tabla en la que se detallan las cotizaciones y proveedores de los componentes.

2.4.1. Características y especificaciones

1. Módulo ESP32-CAM (Figura 10): Es un pequeño módulo Wi-Fi que soporta cámara, además de varios pines para conectar otros periféricos, contiene también una ranura para insertar una tarjeta microSD en la que se pueden almacenar archivos. Integra una pequeña cámara OV2640, el cual es capaz de operar hasta 15 cuadros por segundo, permitiéndole al usuario un control total sobre la resolución y el formato de la imagen. Cabe mencionar que este módulo no posee un puerto USB, por lo que es necesario la utilización de un módulo FTDI para programarla.

Fig. 10: Módulo ESP32-CAM.

Especificaciones:

- Voltaje de Alimentación ESP 32 CAM: 5V
- Módulo Wi-Fi BT 802.11b/g/n
- Tipo de cámara: OV2640
- Formato: UXGA de 2MP
- CPU 32 bits de doble núcleo de baja potencia
- Frecuencia principal de hasta 240 MHz
- Velocidad de reloj de hasta 160 MHz

2. Módulo FTDI (Figura 11): Es un adaptador para poder conectar el módulo ESP32-CAM a un puerto serial. Los pines de salida de esta tarjeta fueron pensados para trabajar con Arduino y con tarjetas basadas en Arduino de 3.3V o 5V, pero también puede ser utilizada para cualquier aplicación que necesite comunicación serial.

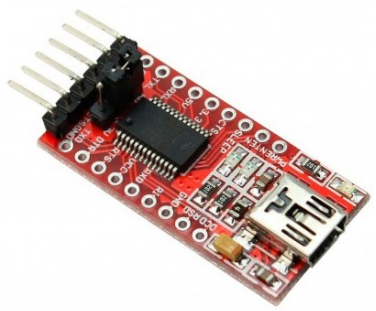


Fig. 11: Módulo FTDI.

Especificaciones:

- Chip: FT232RL
- Entrega Alimentación en 5V o 3.3V, seleccionable mediante jumper
- Posee reflejados los pines VCC, 5V, 3.3V, GND, TX, RX, RTS, CTS y DTR
- Interface: Mini USB
- Dimensiones del módulo: 33 x 17 mm
- Protección de sobrecorriente con fusible automático de 500mA

3. Sensor DHT11 (Figura 12): Este sensor se caracteriza por tener la señal digital calibrada por lo que asegura una alta calidad y fiabilidad a lo largo del tiempo, ya que contiene un microcontrolador de 8 bits integrado. Está constituido por dos sensores resistivos (temperatura NTC y humedad). Además, presenta una rápida respuesta en sus mediciones. Cabe mencionar que se necesita de una resistencia de 10[kΩ] para conectar el sensor al módulo ESP32-CAM.

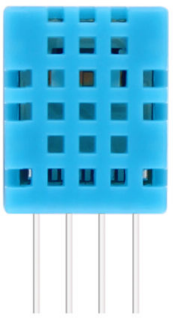


Fig. 12: Sensor de temperatura y humedad DHT11.

Especificaciones:

- Rango de temperatura: 0 a 50°C
- Precisión: 0.2°C
- Voltaje de trabajo: 5 [V]

4. Filamentos ASA para impresión 3D (Figura 13): El ASA es un material terpolímero amorfo termoplástico similar al ABS. Es denominado como plástico de ingeniería por mantener su aspecto y resistencia al impacto incluso después de estar expuesto a largos intervalos de tiempo al aire libre, lluvia, frío y al agua salada del mar.

Fig. 13: Filamentos ASA.

Especificaciones:

- Resistente a los rayos ultravioletas
- Resistente al agua
- Tiene una gran resistencia a químicos y a todo tipo de tratamientos químicos
- Mejor resistencia térmica que el ABS
- Idóneo para realizar prototipos duraderos para su uso en exteriores a la intemperie

5. Pernos, tuercas y golillas (Figura 14): Se requieren de pernos y tuercas para la unión de las piezas y armado de la trampa.

Fig. 14: Pernos, tuercas y golillas.

Especificaciones:

- Medidas: 3 x 20 mm
- Unidades por paquete: 20 unidades

6. Piso pegajoso para trampa delta (Figura 15): Lámina adhesiva de color amarillo para trampeo masivo, detección y monitoreo de plagas. Se coloca en la base de la trampa delta para que los machos de la polilla se atrapen.

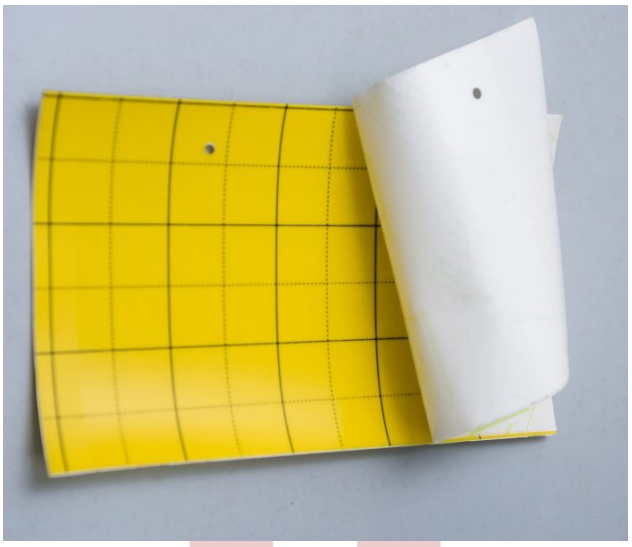


Fig. 15: Piso pegajoso.

Especificaciones:

- Medidas: 15 x 25 cm

7. Feromona sexual para *Cydia Pomonella* (Figura 16): Las feromonas son modificadores del comportamiento que se encuentran en el medio ambiente de forma natural, permiten interceptar la comunicación entre individuos para establecer rápidamente la ubicación de las plagas.

Fig. 16: Feromona sexual para *Cydia Pomonella*.

8. Solar Charger Shield V2.2 (Figura 17): Cargador solar que se monta directamente sobre una placa Arduino y permite alimentar la placa con energía solar con una batería de respaldo. La shield carga la batería de forma eficiente y alimenta la placa Arduino con una tensión estabilizada de 5V. Es ideal para proyectos en exteriores, redes de sensores, etc., donde la alimentación es un problema.

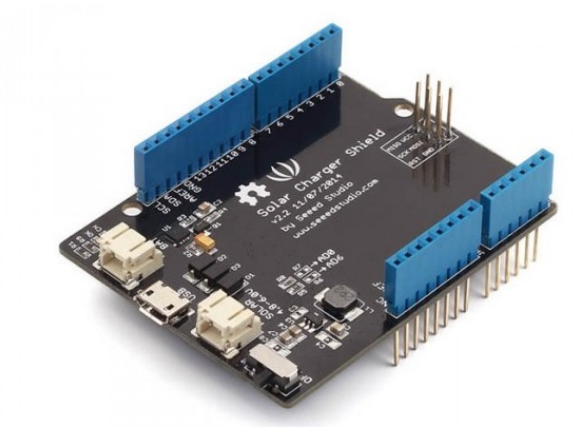


Fig. 17: Solar Charger Shield V2.2 para Arduino.

Especificaciones:

- Salida: 3W
- Corriente de carga hasta 900mA
- Indicador LED de estado e batería (Rojo: cargando, verde: cargada)
- Conector de carga micro USB
- Entrada de batería LiPo: 3.0 a 4.5 V
- Entrada USB: 4.75 a 5.25 V
- Entrada por célula solar: 4.8 a 6 V
- Dimensiones: 58 x 63 mm

9. Panel solar (Figura 18): Panel solar especial para realizar prácticas electrónicas o alimentar circuitos de bajo consumo.

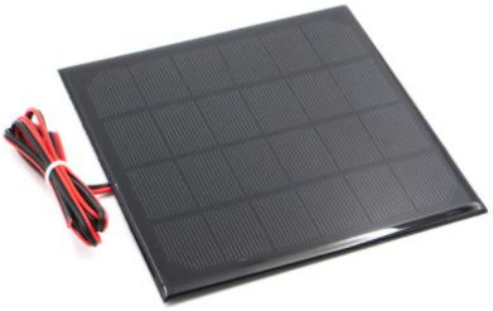


Fig. 18: Panel solar de 6V.

Especificaciones:

- Potencia máxima: 2.5W
- Máximo voltaje de salida: 5V
- Máxima corriente de salida: 0.5A
- Voltaje de circuito abierto: 6V
- Corriente de corto circuito: 0.56A
- Dimensiones: 145 x 145 mm

10. Batería LiPo DTP603450 (Figura 19): Batería delgada y liviana basada en la química se iones de litio. Viene con conector JST-PH estándar de dos pines. La batería incluye protección incorporada contra sobre voltaje, sobre corriente y voltaje mínimo

Fig. 19: Batería LiPo DTP603450.

Especificaciones:

- Voltaje: 3.7V
- Corriente: 1000mA
- Conector: JST-PH (2mm de separación entre pines)
- Dimensiones: 51 x 33.5 x 6.2 mm

2.4.2. Cotizaciones

Tabla 1: Listado de componentes y precios.
Fuente: Elaboración propia.

Nº	ITEM	PROVEEDOR	VALOR UN ($)	ENVÍO ($)	TOTAL ($)
1	ESP32-CAM	Makers Chile	9990	4500	14490
2	Módulo FTDI	Makers Chile	2990	4500	7490
3	Sensor DHT11	Makers Chile	1690	4500	6190
4	Pack pernos	Easy	1790	6390	8180
5	Kit trampa delta*	ControlBest	7440	5500	12940
6	Shield de carga solar	Aliexpress	4969	2172	7141
7	Panel solar	Aliexpress	5499	1318	6817
8	Batería LiPo	MicroRobotics	6476	4528	11004

En la Tabla 1 se ordenan todos los componentes a utilizar con sus respectivos valores.

*El kit incluye una trapa delta de cartón plastificado (la cual no se utilizará en este proyecto), un piso pegajoso y una feromona sexual.

El valor total de todos los componentes a utilizar es de $74252, sin contar el valor de la impresión ya que éstas se realizan en dependencias de la Universidad. En el Anexo 7 se presentan las cotizaciones más detalladas, junto con los links de compra.

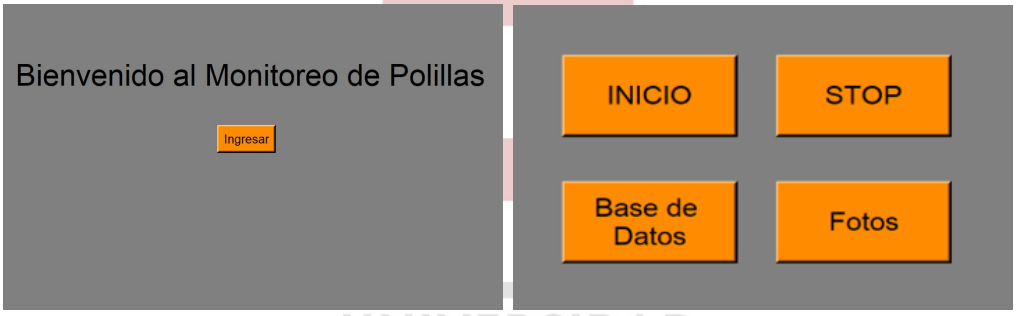
3. Resultados experimentales

3.1. Resultados

3.1.1. Interfaz gráfica de usuario

A modo de resultado, se presenta la aplicación creada para que el usuario tenga un fácil acceso tanto a los datos como a la inicialización y finalización de la captura de fotografías. Al abrir la aplicación se presenta una página de bienvenida al usuario y un botón para ingresar. Luego de ingresar, se muestran cuatro botones: INICIO, STOP, Base de Datos y Fotos. El botón INICIO da inicio a la captura de fotografías en la trampa. Estas fotografías serán capturadas cada cierto tiempo que defina el usuario, en este caso, cada diez segundos a modo de prueba. El botón STOP detiene la captura de fotografías. Estos botones están para que el usuario pueda fácilmente dar inicio y término a la captura. El botón de Base de Datos mostrará todos los datos obtenidos desde el inicio del programa, es decir, los registros de fecha, hora, temperatura, humedad y cantidad de polillas obtenidos cada vez que se tome una fotografía. Finalmente, el botón Fotos abrirá la carpeta en donde se almacenan las fotografías capturadas por la trampa.

En la Figura 20 se muestra la aplicación de usuario. En la Figura 20a se muestra la página de bienvenida y en la Figura 20b se muestran los botones que ejecutan los programas.



(a) Página de bienvenida para ingresar a la aplicación.

(b) Botones que permiten al usuario ejecutar las distintas acciones.

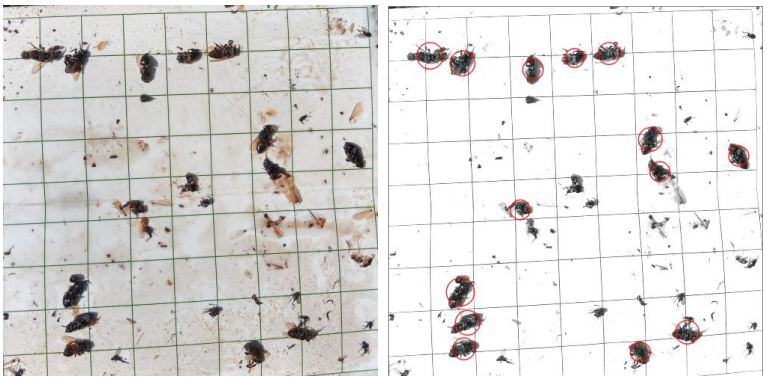
Fig. 20: Interfaz de usuario.
Fuente: Elaboración propia.

El código creado para la realización de esta interfaz se muestra en el Anexo 8.

3.1.2. Pruebas preliminares

A continuación, se muestra el funcionamiento del algoritmo de captura de imágenes y detección de objetos en una trampa delta de polillas usada anteriormente, que me fue facilitada para poder realizar pruebas. En la Figura 21 se muestran la foto original (Figura 21a) y la foto con los objetos detectados (Figura 21b). Se observa que existe una diferencia en la cantidad de polillas detectadas y la cantidad de polillas que realmente existe en la trampa, ya que la cantidad real de éstas es de 15, mientras que las detectadas fueron 14. Se realiza una serie de pruebas para determinar el costo de utilizar éste método para la detección de las polillas. Para esto, se toman diversas fotografías a distintas horas del día para ver si afecta la distinta iluminación en la detección de las polillas. Además, se realizan pruebas del sensor

para determinar la temperatura y humedad existente cada vez que se toman las fotografías. Los resultados de estas pruebas se muestran en la Tabla 2.



(a) Fotografía de una trampa delta usada.

(b) Objetos detectados dentro de la imagen.

Fig. 21: Polillas encontradas en una trampa delta antes y después de la detección de objetos. Fuente: Elaboración propia.

Tabla 2: Pruebas del método de detección de polillas. Fuente: Elaboración propia.

N◦	Fecha	Hora	Temp (°C)	Humedad (%)	Reales	Detectadas	Error
1	2020-10-06	20:06	17.3	52	15	14	1
2	2020-10-06	20:15	17.2	51.5	14	11	3
3	2020-10-06	20:15	17.2	51.5	14	11	3
4	2020-10-06	20:16	17.2	51.2	10	7	3
5	2020-10-07	10:51	15.6	58	13	12	1
6	2020-10-07	10:51	15.6	58	13	15	2
7	2020-10-07	10:52	15.7	58	14	14	0
8	2020-10-07	10:52	15.6	58.1	14	15	1
9	2020-10-07	15:24	20.4	47	13	13	0
10	2020-10-07	15:24	20.4	47	13	13	0
11	2020-10-07	15:26	20.5	46.9	13	14	1
12	2020-10-07	15:26	20.6	47	15	14	1
Promedio					13.42	12.75	1.3

Como se observa en la tabla, en nueve de los doce casos realizados existe error entre la cantidad de polillas real y las detectadas, presentando una exactitud promedio del 95 %. Se aprecia que la hora a la que se presentas mayores errores en el conteo es a las 20:15, hora en la que el sol se está poniendo, mientras que tanto en la mañana como en la tarde, los conteos difieren en una o dos polillas máximo, existiendo casos en los que el conteo fue exacto. Por lo anterior, se determina que el cambio en la luz provocado por la puesta del sol afecta en el conteo de las polillas haciendo que la cantidad detectada sea inferior a la cantidad real de polillas existentes.

Para disminuir el error que existe en el conteo de las polillas en las horas de menos luz natural, se prueba la utilización de un flash presente en la cámara para que la luz en

(a) Fotografía tomada sin flash. (b) Fotografía tomada con flash.

Fig. 22: Comparativa de fotografías tomadas en la noche en total oscuridad sin y con la utilización de flash. Fuente: Elaboración propia.

la trampa sea uniforme al momento de sacar la fotografía. En la Figura 22, se muestra la comparativa entre una fotografía capturada a las 23:22 horas sin y con el flash. En la Figura 22a se ve que la fotografía es completamente oscura y no es posible identificar nada en ella, sin embargo, al activar el flash, la fotografía capturada es la que se ve en la Figura 22b y se observa claramente la mejora en la fotografía y es posible distinguir en ella todas las polillas.

Al utilizar el flash durante el día, también se aprecia una mejora en el conteo de las polillas, debido a que el flash proporciona una luz uniforme en la trampa, eliminando sombras o cambios en la intensidad de la luz. En la Tabla 3 se muestran los resultados de la pruebas realizadas con el flash activado.

Tabla 3: Pruebas del método de detección de polillas con flash activado. Fuente: Elaboración propia.

N○	Fecha	Hora	Temp (°C)	Humedad (%)	Reales	Detectadas	Error
1	2020-10-09	17:24	22.2	50	15	17	2
2	2020-10-09	17:24	22.2	50	15	15	0
3	2020-10-09	17:24	22.2	50.1	16	16	0
4	2020-10-09	17:24	22.2	50	14	14	0
5	2020-10-09	17:25	22.3	50	15	15	0
6	2020-10-09	21:04	18	56	12	12	0
7	2020-10-09	21:04	18	56.1	12	12	0
8	2020-10-09	21:04	18	56.1	13	15	2
9	2020-10-09	21:05	17.9	56.1	14	14	0
Promedio					14	14.4	0.4

Como se observa en la tabla, de los nueve casos estudiados solo dos presentaron error, disminuyendo considerablemente el error promedio en comparación a los casos anteriores, siendo ahora la exactitud promedio del 97.2 %. Gracias a esto es posible realizar un conteo de forma más precisa. Cabe mencionar que todas estas pruebas fueron realizadas con la trampa usada.

Con los datos obtenidos de las pruebas, se crea una base de datos accesible desde la interfaz de usuario que muestra todos los datos registrados. La Figura 23 muestra esta base de datos con valores de prueba.

Fecha	Hora	Cantidad	Temperatura	Humedad
2020/12/09	17:57:29	10	31.7	21
2020/12/09	17:59:44	10	31.5	21
2020/12/09	18:00:21	10	31.7	21
2020/12/09	18:00:38	10	31.5	21
2020/12/09	18:01:05	10	31.7	21
2020/12/09	18:02:11	10	31.5	21
2020/12/09	18:04:06	10	31.7	21
2020/12/09	18:04:39	10	31.7	34
2020/12/09	18:05:11	10	31.7	23
2020/12/09	18:30:50	10	31.5	21
2020/12/09	18:32:32	10	31.7	21
2020/12/09	18:32:40	10	31.5	21
2020/12/09	18:32:47	10	31.7	21
2020/12/09	18:32:54	10	31.6	25
2020/12/09	18:39:45	10	31.5	21
2020/12/09	18:40:27	10	31.5	21
2020/12/09	18:43:12	10	31.4	21
2020/12/09	18:46:33	10	31.5	21

Fig. 23: Base de datos creada con los valores de las pruebas realizadas.
Fuente: Elaboración propia.

Además, en este instante se realizan pruebas de la duración de la batería. Para esto, se deja cargando la batería con el panel solar hasta que alcanza su capacidad máxima y se pone a trabajar la cámara conectada a la batería para que registre fotografías y datos cada treinta minutos. Con esto, se determina que la autonomía del sistema con el programa corriendo sin detenerse es de 6.5 horas.

3.1.3. Construcción de la trampa

Una vez listas las piezas que fueron impresas en la Universidad, se arma la estructura de acuerdo a los planos.

Fig. 24: Trampa impresa armada.
Fuente: Elaboración propia.

La Figura 24 muestra la estructura ya armada. Cabe mencionar que se tuvieron algunas complicaciones al momento de armar la trampa, ya que el diseño realizado tenía algunas partes muy pequeñas o de difícil ensamblaje, por esto, se tuvo que recurrir a utilizar más pernos de los que se tenían presupuestados en un comienzo, además de una pequeña modificación que se realizó en los costados de la trampa puesto que algunas partes que eran muy pequeñas se rompieron. A pesar de esto, la trampa pudo ser ensamblada de igual manera y es funcional, sin embargo, queda como trabajo futuro modificar el diseño para que la estructura sea de fácil armado y totalmente desmontable.

3.1.4. Pruebas en campo

Para probar la trampa en condiciones reales de operación, es llevada al Fundo el Sauce, ubicado en la Ruta K-555 camino a San Clemente, de la ciudad de Talca, comuna Talca, en donde existe una plantación de manzanos. Para mayor referencia, en la Figura 25 se adjunta el mapa de la plantación.

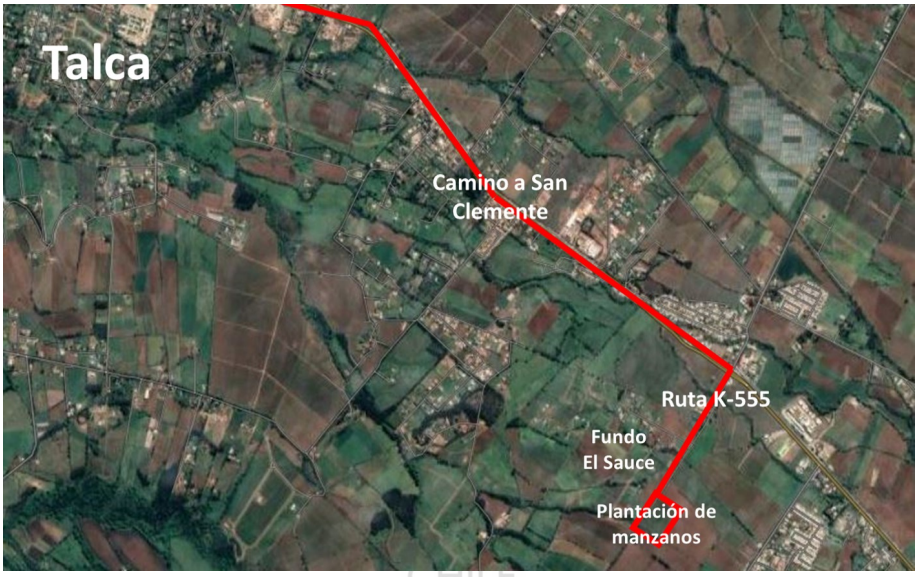


Fig. 25: Mapa de la plantación de manzanos donde la trampa es probada.
Fuente: Google Maps.

La plantación de manzanos tiene una extensión de 5.86 hectáreas. Para esta extensión, es necesaria la utilización de varias trampas, ya que deben ubicarse entre cuatro a seis trampas por hectárea. Para motivos de este informe y la realización de las pruebas, se ubica la trampa aproximadamente al centro de la plantación. Cabe mencionar que esta plantación no cuenta con ningún tipo de monitoreo para las polillas y se utiliza solo la aplicación de pesticidas sin ningún tipo de control.

Dentro de la plantación, la trampa se ubica en un árbol de manzano, en la parte superior de la copa del árbol, pues este es el lugar de mayor efectividad de las trampas delta, como se observa en la Figura 26. La trampa es instalada junto con el panel solar para que cargue la batería que alimenta la cámara, con el piso pegajoso y feromona y todos los componentes electrónicos en su interior.

La trampa es probada en esta plantación durante dos días y una noche, donde se observó que la mayor actividad de las polillas se concentró casi en su totalidad en la noche.

Fig. 26: Trampa instalada en el manzano.
Fuente: Elaboración propia.

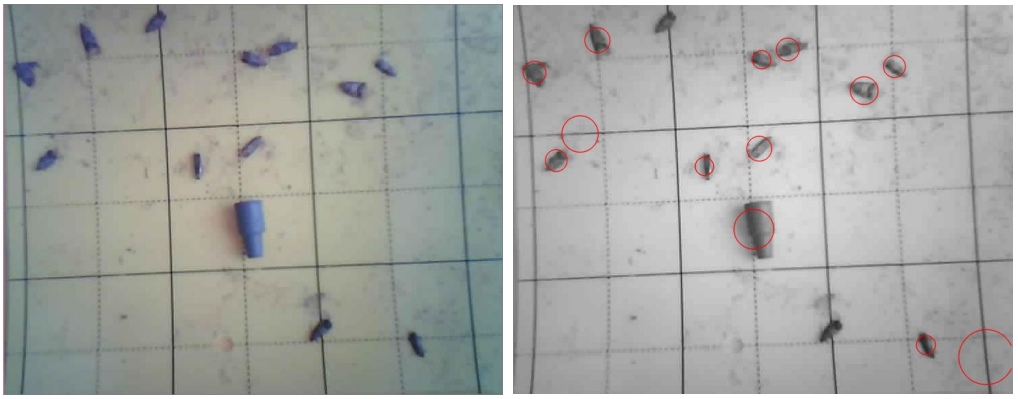
Tabla 4: Pruebas del método de detección de polillas en plantación de manzanos.
Fuente: Elaboración propia.

No	Fecha	Hora	Temp (°C)	Humedad (%)	Reales	Detectadas	Error
1	2020-12-29	11:46	25	21	0	0	0
2	2020-12-29	11:48	25	20.9	0	0	0
3	2020-12-29	20:37	27	18.5	2	2	0
4	2020-12-29	20:42	27	18.5	2	2	0
5	2020-12-30	08:23	15.6	30	12	13	1
6	2020-12-30	08:24	15.6	30	12	13	1
Promedio					7	7.5	0.5

Como el campo en donde fue probada la trampa no cuenta con una red Wi-Fi a la que conectarse, se tuvo que utilizar los datos móviles de un teléfono celular a modo de zona Wi-Fi, por esta misma razón, no se pudo dejar la trampa funcionando durante todas las horas que estuvo instalada en la plantación. Sin embargo, la trampa se revisó periódicamente durante este tiempo y se hacía funcionar la aplicación cada vez que se revisaba, aproximadamente cada nueve horas. Los resultados obtenidos se presentan en la Tabla 4.

Lamentablemente, por motivos de tiempo y recursos no se pudo realizar las pruebas durante un tiempo más prolongado, sin embargo, las pruebas realizadas sirven para determinar la efectividad de la trampa y los algoritmos. A partir de los resultados obtenidos se concluye que al igual que en las pruebas preliminares, el algoritmo funciona con un pequeño error en el conteo, presentando una exactitud promedio de 93.9 % en condiciones reales. Sin embargo, como se mencionó, este error no es determinante al momento de aplicar el pesticida. Con estas pruebas se comprueba además que el algoritmo funciona con la cámara conectada exclusivamente al Wi-Fi y alimentada mediante el panel solar.

La Figura 27 muestra la comparativa de una de las fotografías capturadas en campo de las polillas de la manzana. La Figura 27a muestra la fotografía normal, mientras que la Figura 27b muestra los objetos detectados. Cabe mencionar que el algoritmo muestra la feromona como un objeto detectado, por lo que al momento de mostrar la cantidad de polillas, el programa resta 1 del total de objetos detectados.



(a) Fotografía de la trampa en campo. (b) Objetos detectados en fotografía en campo.

Fig. 27: Muestra de fotografía tomada en campo junto con la detección de los objetos presentes en ella.
Fuente: Elaboración propia.

Si bien el número de objetos contabilizados es mayor solo en una unidad a las polillas que hay en la trampa, en la Figura 27 se observa que hay dos polillas que no fueron detectadas por el algoritmo, mientras que existen otras dos detecciones en lugares donde no hay polillas. Esto puede mejorarse mediante la calibración del sistema, ya que éste se encuentra calibrado de acuerdo a las pruebas preliminares realizadas con la trampa anterior.

La Figura 28 muestra la base de datos creada a partir de las pruebas en campo.

Fecha	Hora	Cantidad	Temperatura	Humedad
2020/12/29	11:46:33	0	25	21
2020/12/29	11:48:06	0	25	20.9
2020/12/29	20:37:45	2	27	18.5
2020/12/29	20:42:54	2	27	18.5
2020/12/30	08:23:11	13	15.6	30
2020/12/30	08:24:40	13	15.6	30

Fig. 28: Base de datos creada a partir de las pruebas en campo.
Fuente: Elaboración propia.

Como se vio en la sección 3.1.2, la autonomía del sistema es de 6.5 horas. De la forma en que está diseñado el programa, este corre cada cierta cantidad de tiempo (en el caso de las pruebas preliminares, treinta minutos) sin detenerse, pero al llevarlo a la realidad, este tiempo puede variar ya que como se observa en los resultados de la Tabla 4, la mayor cantidad de polillas aparece durante la noche, siendo innecesaria la toma tan seguida de fotografías durante el día. Al modificar el algoritmo de captura de fotografías y datos para que se adecúe a los horarios de las polillas, la duración de la batería puede variar.

4. Conclusión

4.1. Sumario

Recordando los objetivos del proyecto, en esta sección se presentarán las evidencias de que éstos has sido cumplidos. Los objetivos específicos son:

- *Desarrollar un sistema de monitoreo de polillas utilizando cámaras y sensores.* - En la sección 2.1, se presentó el circuito electrónico que corresponde al sistema desarrollado y los componentes que lo constituyen; en las secciones 2.3.1 y 2.3.2, se presentaron los algoritmos que permiten conectar la cámara y obtener las fotografías y los datos del sensor hacia un computador; finalmente, en la sección 3.1.2, se presentaron las pruebas preliminares y en la sección 3.1.4, las pruebas en campo realizadas para probar los algoritmos y los resultados obtenidos en cada una de las pruebas, determinando que el objetivo planteado se cumple.

- *Determinar la cantidad de polillas dentro de una trampa delta mediante el desarrollo de algoritmos que permitan su detección y conteo.* - En la sección 2.3.3, se presentó el algoritmo de detección y conteo de objetos en una imagen que fue aplicado a las fotografías capturadas en el objetivo anterior; en las secciones 3.1.2 y 3.1.4, se realizaron las pruebas preliminares y en campo que permitieron contar las polillas presentes en cada una de las pruebas, cumpliendo así este objetivo.

- *Diseño y construcción de una estructura para una trampa basándose en el diseño actual de las trampas delta que pueda contener el sistema de cámara y sensores.* - En la sección 2.2, se presentó el diseño 3D de la estructura realizado en *Inventor*, explicando sus características y dimensiones; en la sección 3.1.3, se mostró la trampa impresa y ensamblada; finalmente, en la sección 3.1.4, se presentó la trampa ubicada en el campo cumpliendo la función de atrapar las polillas y resguardar el circuito en su interior. Con esto mencionado, se determina que el objetivo se cumple.

- *Facilitar el acceso a la información obtenida a partir de los sensores mediante la creación de una interfaz de usuario.* - En la sección 3.1.1, se presentó la interfaz gráfica de usuario creada para facilitar al usuario el acceso a la información. En esta sección se mostró también como se constituye esta aplicación, los botones que posee y la función que cumple cada uno de ellos. Con esto, el objetivo planteado es cumplido.

Por lo tanto, de acuerdo a lo estudiado y realizado en este proyecto, se concluye que el objetivo general:

Automatizar el conteo de la polilla de la mazana en trampas delta para la predicción de la plaga, con respecto a técnicas de monitoreo tradicionales,

se cumple, y esto ha sido demostrado gracias a las simulaciones realizadas y los resultados presentados.

4.2. Conclusiones

De acuerdo a los resultados obtenidos en la sección 3, se determina que la hipótesis planteada en la sección 1.4 es verdadera, ya que el sistema diseñado permitió un correcto

monitoreo de muestra de polillas en la trampa, automatizando el control de la plaga. Esto demostró un correcto funcionamiento en el diseño del algoritmo y en la conexión de las partes electrónicas. Sin embargo, como se evidenció en las secciones 3.1.2 y 3.1.4, en algunos de los casos existe una diferencia en el conteo de las polillas; a pesar de esto, se concluye que este error es despreciable ya que al tratarse de un error de una o dos polillas máximo, el costo de este error no es alto al momento de determinar la utilización de los plaguicida, ya que la precisión del sistema nunca baja del 90 %, siendo la precisión promedio total de todas las pruebas realizadas de 95.2 %.

En la sección 3.1.2 se determinó que la autonomía del sistema utilizando la batería es de 6.5 horas, sin embargo, este tiempo no es suficiente para que el sistema alcance a estar toda la noche funcionando, por lo que se debe modificar el algoritmo para que trabaje de forma diferente o utilizar una batería con mayor capacidad.

Además, al realizar la impresión en 3D de las piezas que componen la estructura de la trampa, se determinó en la sección 3.1.3, que algunas de las piezas presentaron problemas al momento del ensamblaje debido a que el diseño no fue el más apropiado. Para mejorar esta situación, se requiere de algunos cambios en el diseño de la estructura; a pesar de estos problemas, la trampa tal como se presenta en este informe cumplió correctamente con el propósito de capturar las polillas y resguardar el sistema en su interior sin exponerlo al exterior.

4.3. Trabajos futuros

Los trabajos futuros que se pueden realizar para complementar este proyecto son:

- En cuanto a la interfaz de usuario, si bien ésta es capaz de iniciar y detener la captura de fotografías y datos del sensor, de mostrar las fotografías capturadas en la carpeta y generar una base de datos en donde se ordenan estos datos, aún se le pueden realizar algunos cambios para mejorar aún más su funcionamiento o simplemente realizar cambios que mejoren su estética.
- Calibración del algoritmo de detección de objetos para las polillas capturadas en campo.
- Modificación del diseño de la estructura de la trampa para un mejor y más fácil ensamblaje de las piezas.
- Modificación del algoritmo y posterior realización de pruebas para determinar algún cambio en la duración de la batería. De no detectarse cambios, se debe considerar aumentar la capacidad de la batería.
- Crear una red de trampas para poder monitorear una plantación en toda su extensión, tal como se haría normalmente con las trampas delta. Cabe mencionar que para esto será necesario una red Wi-Fi de largo alcance o repetidores para ampliar la señal, dependiendo de las dimensiones de la plantación.

Referencias

[1] C. Quiroz, F, Luengo, C. Salas, P. Abarca, P. Bermudez, G, Lobos, P. Larraín, F. Rodríguez, J. Riquelme and S. Santelices, "Manejo integrado de plagas del nogal en la provincia de Choapa", *Boletín INIA N° 324*, pp. 47-56, 2016. Instituto de Invesigaciones Agropecuarias, Centro Regional Intihuasi, Centro Experimental Choapa, La Serena, Chile.

[2] Agro Feromonas, "Trampas de feromonas y otros métodos de uso, ¿Cuál es su principio de acción?". Accedido en 4-Octubre-2019 a https://agroferomonas.com/informacion-tecnica-y-cientifica/

[3] J. Torrens and O. Tortosa, "Redescripción de Mastrus ridibundus (Hymenoptera: Ichneumonidae), parasitoide introducido en la Argentina para el control de Cydia pomonella (Lepidoptera: Tortricidae)", *Revista de la Sociedad Entomológica Argentina*, vol. 67, no. 3-4, pp. 109-112, 2008. Sociedad Entomológica Argentina, Buenos Aires, Argentina.

[4] N. Aguirre, "Implementación de un sistema de detección de señales de tráfico mediante visión artificial basado en FPGA", 2013. Universidad de Sevilla, Sevilla, España.

[5] Intagri, "El monitoreo: herramienta indispensable en los programas de MIP y MIE de hortalizas", 2016. Instituto para la Innovación Tecnológica en la Agricultura, Celaya, México.

[6] R. Cid, "Aplicación eficiente de fitosanitarios", capítulo 3, 2014. Instituto Nacional de Tecnología Agropecuaria, Ganadería y Pesca, Buenos Aires, Argentina.

[7] C. Liptak and Dr. T. Motis, "Monitoreo de cultivos para la detección temprana de plagas de insectos", *ECHO Notas de Desarrollo*, no, 136, 2017. Florida, USA.

[8] Red Agrícola, "Monitoreo con trampas mejoró el manejo de la polilla de la manzana", 2017. Accedido en 29-Julio-2020 a https://www.redagricola.com/cl/monitoreo-trampas-mejoro-manejo-la-polilla-la-manzana/

[9] A. Bustillo, "Los insectos y su manejo en la caficultura colombiana", pp. 94-109, 2008. Cenicafé, Federación Nacional de Cafeteros de Colombia, Chinchiná, Colombia.

[10] M. Kumar, P. Singh and S. Kumar, "Precision agriculture in potato cultivation", *Potato science and technology for sub-topics*, 2020. Government of India, Department of Agricultural Research, Education and Indian Council of Agricultural Research, Ministry of Agriculture and Farmers Welfare, New Delhi, India.

[11] P. Reddy, "Precision agriculture", *Agro-ecological approaches to Pest management for Sustainable Agriculture*, 2017.

[12] J.-A. Jiang, C.-L. Chuang, C.-L. Tseng and E.-C- Yang, "Design, implementation and testing of an M2M-based remote agroecological monitoring system for oriental fruit fly, *Bactrocera Dorsalis* (Hendel)", *Lecture notes in Electrical Engineering*, no. 146, pp. 35-58, 2012.

[13] K. Ennouri, M. A. Triki and A. Kallel, "Applications of remote sensing in pest monitoring and crop management", *Bioeconomy for Sustainable Development*, 2020.

[14] A. Jiménez, D. Ravelo and J. Gómez, "Sistema de adquisición, almacenamiento y análisis de información fenológica para el manejo de plagas y enfermedades de un duraznero mediante tecnologías de agricultura de presición", vol. 14, no. 27, pp. 41-51, 2010. Universidad Distrital Francisco José de Caldas, Bogotá, Colombia.

[15] Agriculturers, "El uso de sensores para optimizar el manejo integrado de plagas", 2017. Accedido en 21-Abril-2020 a https://agriculturers.com/el-uso-de-los-sensores-para-optimizar-el-manejo-integrado-de-plagas/

[16] E. Leonairo and J. León-Téllez, "Detección y clasificación de defectos en frutas mediante el procesamiento digital de imágenes", *Revista Colombiana de Física*, vol. 35, no. 1, 2003. Grupo de Óptica Láser, Universidad del Cauca, Popayán, Colombia.

Anexos

1. **Planos de la estructura**

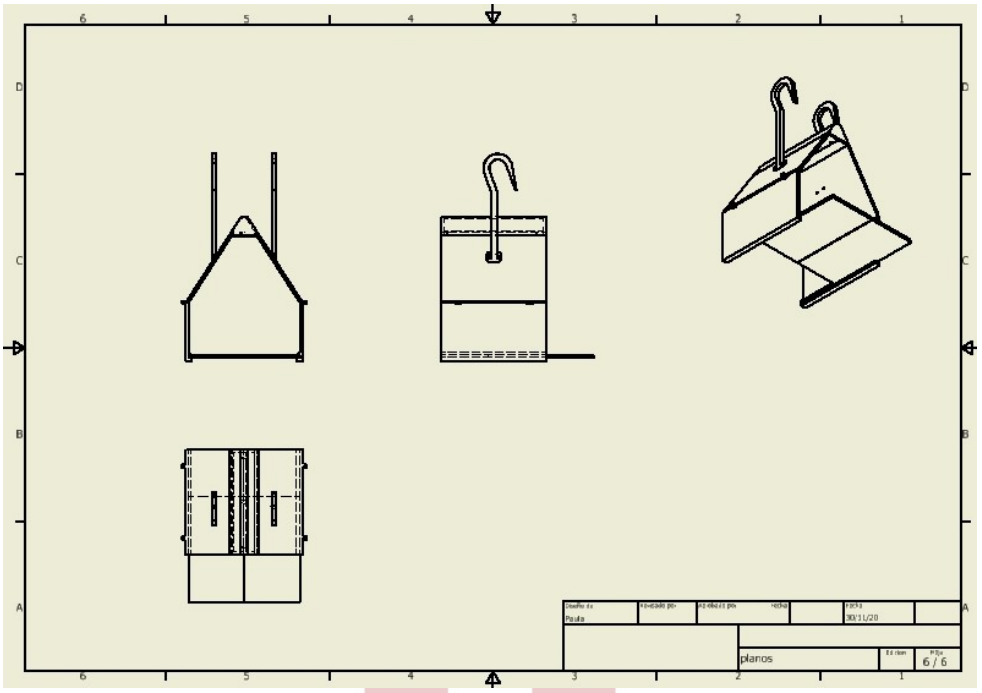


Fig. 29: Plano de la estructura.
Fuente: Elaboración propia.

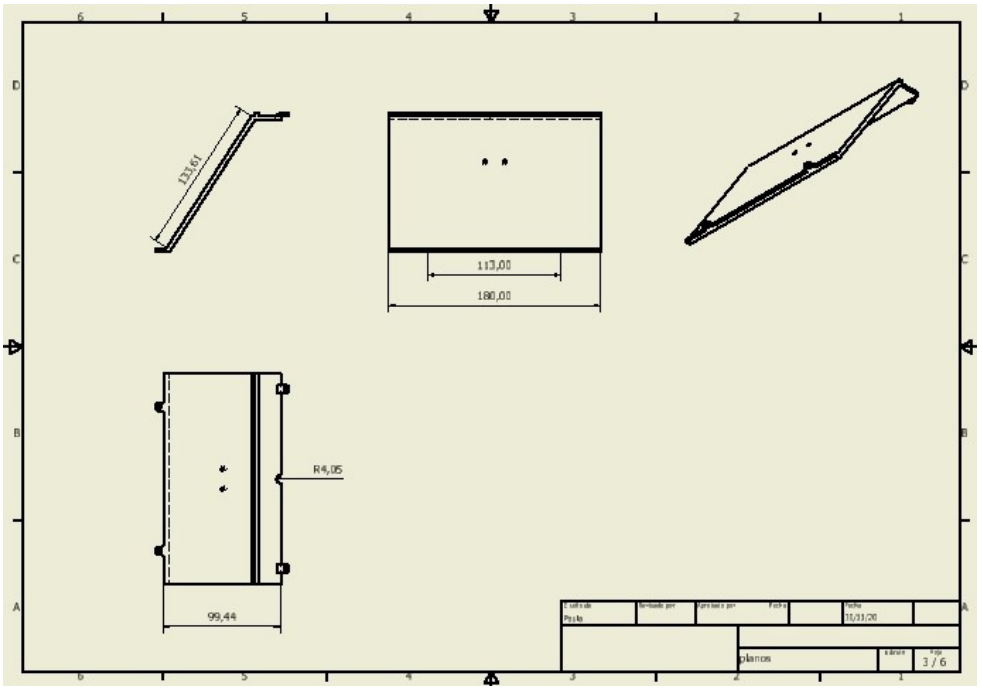


Fig. 30: Plano de la parte superior de la estructura.
Fuente: Elaboración propia.

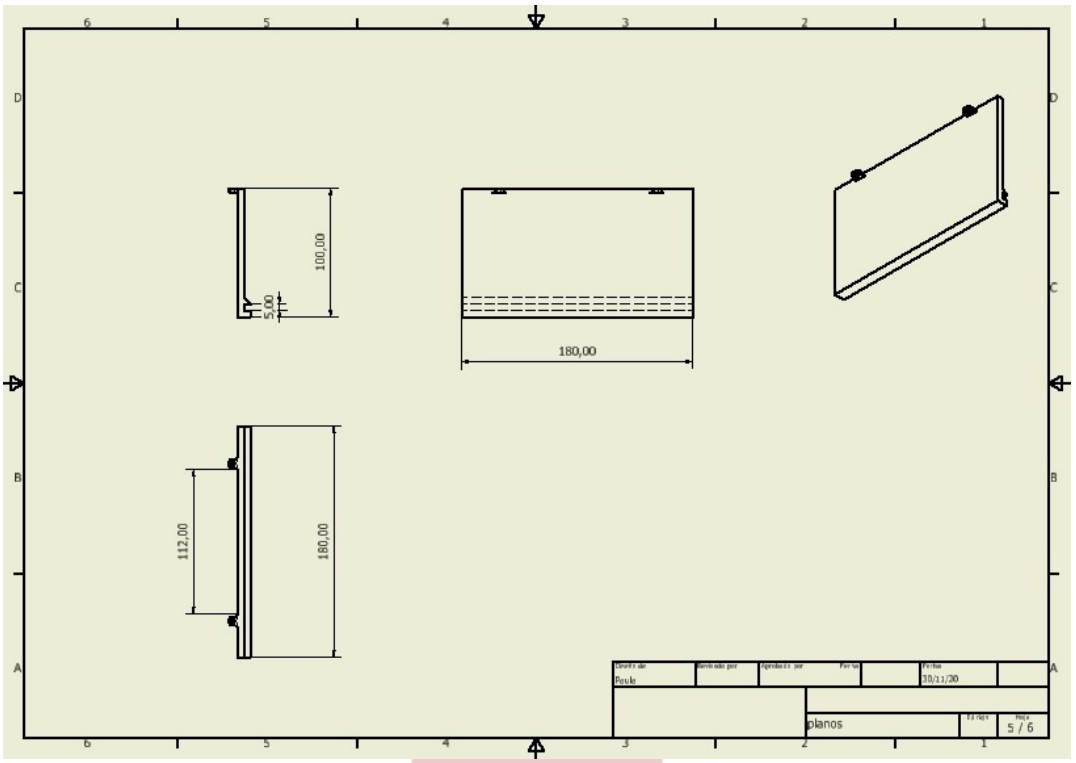


Fig. 31: Plano del lateral de la estructura.
Fuente: Elaboración propia.

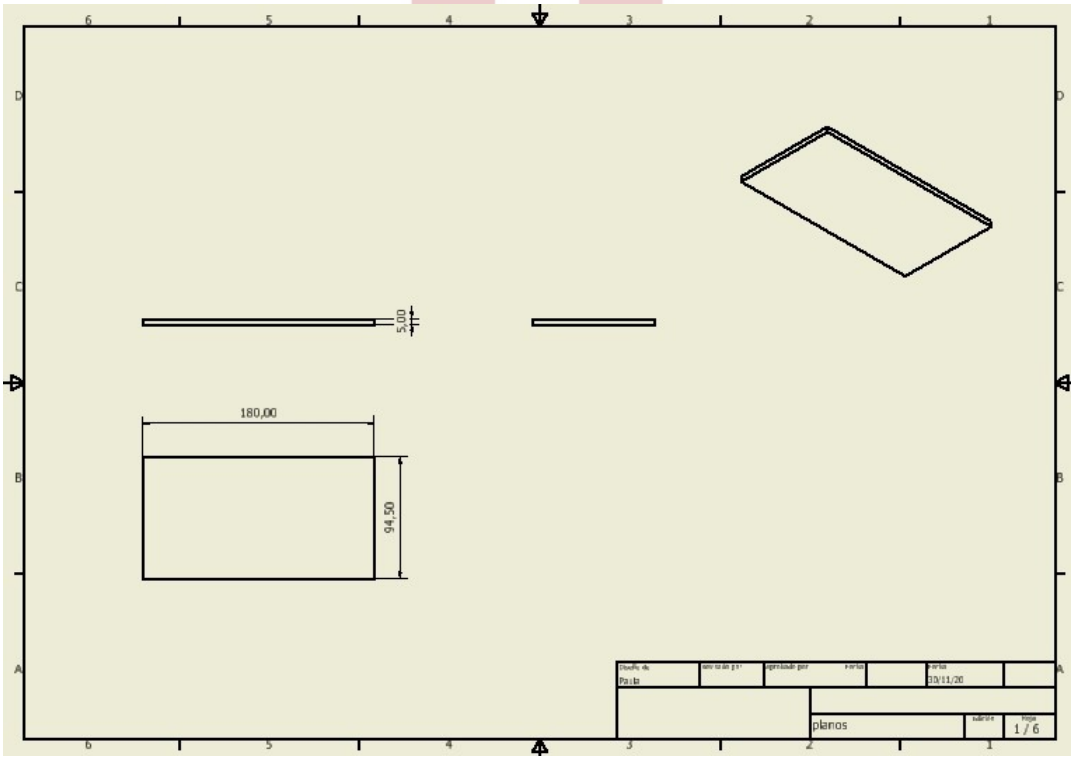


Fig. 32: Plano de la base de la estructura.
Fuente: Elaboración propia.

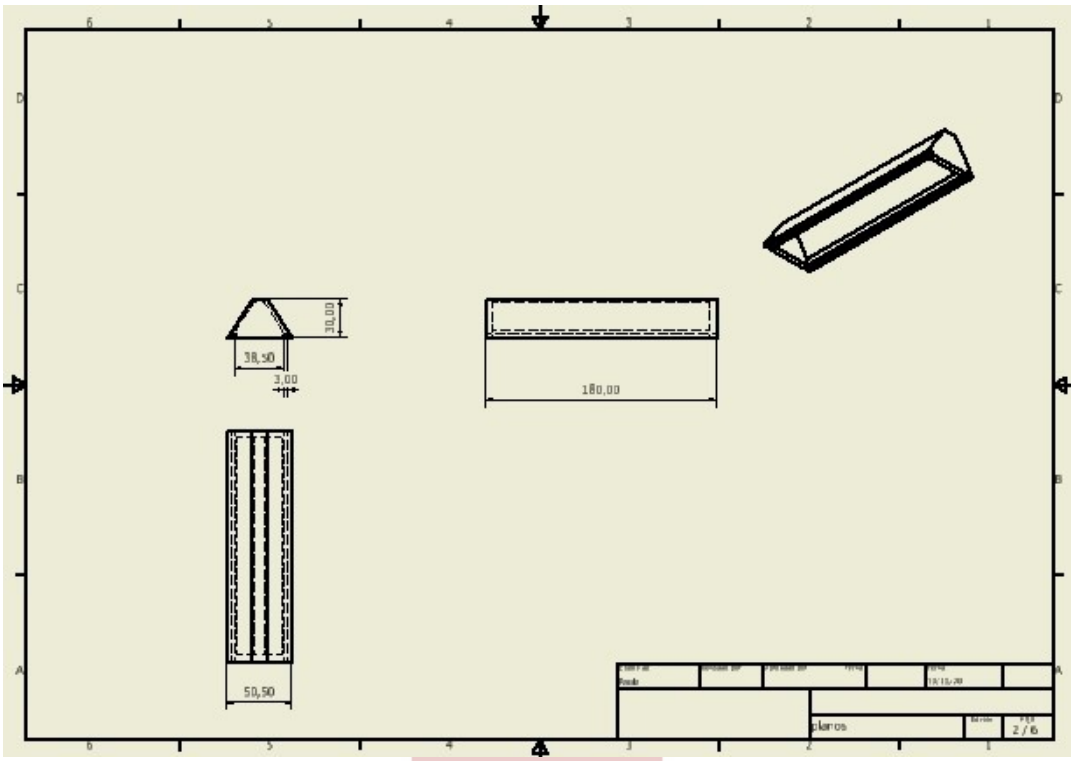


Fig. 33: Plano de la tapa de la estructura.
Fuente: Elaboración propia.

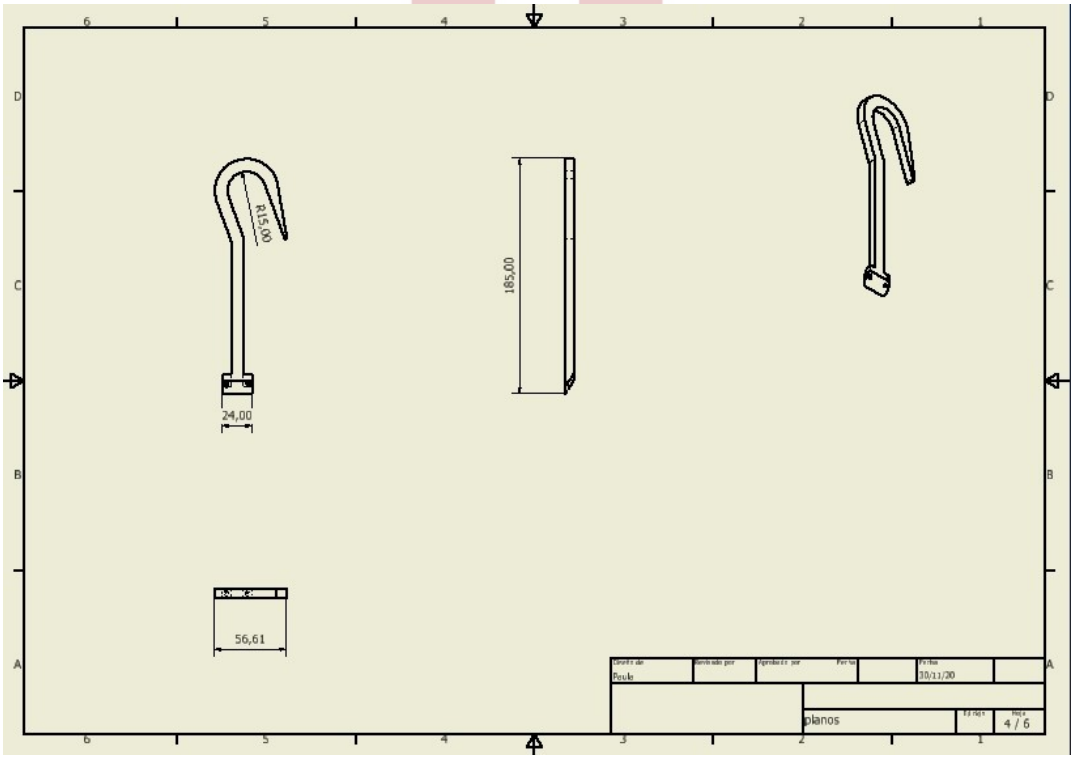


Fig. 34: Plano de los ganchos de la estructura.
Fuente: Elaboración propia.

2. Algoritmo de visualización de imagen desde la cámara

```
#include <WebServer.h>
#include <WiFi.h>
#include <esp32cam.h>

const char* WIFI_SSID = "PMaule2.4";
const char* WIFI_PASS = "rscontador141414";
WebServer server(80);
static auto loRes = esp32cam::Resolution::find(320, 240);
static auto hiRes = esp32cam::Resolution::find(800, 600);
static auto res1 = esp32cam::Resolution::find(1600, 1200);
void
handleBmp()
{
  if (!esp32cam::Camera.changeResolution(loRes)) {
    Serial.println("SET-LO-RES FAIL");
  }
  auto frame = esp32cam::capture();
  if (frame == nullptr) {
    Serial.println("CAPTURE FAIL");
    server.send(503, "", "");
    return;
  }
  Serial.printf("CAPTURE OK %dx%d %db\n", frame->getWidth(), frame->
    getHeight(),
                static_cast<int>(frame->size()));
  if (!frame->toBmp()) {
    Serial.println("CONVERT FAIL");
    server.send(503, "", "");
    return;
  }
  Serial.printf("CONVERT OK %dx%d %db\n", frame->getWidth(), frame->
    getHeight(),
                static_cast<int>(frame->size()));
  server.setContentLength(frame->size());
  server.send(200, "image/bmp");
  WiFiClient client = server.client();
  frame->writeTo(client);
}
void
serveJpg()
{
  auto frame = esp32cam::capture();
  if (frame == nullptr) {
    Serial.println("CAPTURE FAIL");
    server.send(503, "", "");
    return;
  }
  Serial.printf("CAPTURE OK %dx%d %db\n", frame->getWidth(), frame->
    getHeight(),
                static_cast<int>(frame->size()));
  server.setContentLength(frame->size());
  server.send(200, "image/jpeg");
  WiFiClient client = server.client();
  frame->writeTo(client);
```

```
52 }
53 void
54 handleJpgLo()
55 {
56   if (!esp32cam::Camera.changeResolution(loRes)) {
57     Serial.println("SET-LO-RES FAIL");
58   }
59   serveJpg();
60 }
61 void
62 handleJpgHi()
63 {
64   if (!esp32cam::Camera.changeResolution(hiRes)) {
65     Serial.println("SET-HI-RES FAIL");
66   }
67   serveJpg();
68 }
69 void
70 handleJpgRes1()
71 {
72   if (!esp32cam::Camera.changeResolution(res1)) {
73     Serial.println("SET-HI-RES FAIL");
74   }
75   serveJpg();
76 }
77 void
78 handleJpg()
79 {
80   server.sendHeader("Location", "/cam-hi.jpg");
81   server.send(302, "", "");
82 }
83 void
84 handleMjpeg()
85 {
86   if (!esp32cam::Camera.changeResolution(loRes)) {
87     Serial.println("SET-HI-RES FAIL");
88   }
89   Serial.println("STREAM BEGIN");
90   WiFiClient client = server.client();
91   auto startTime = millis();
92   int res = esp32cam::Camera.streamMjpeg(client);
93   if (res <= 0) {
94     Serial.printf("STREAM ERROR %d\n", res);
95     return;
96   }
97   auto duration = millis() - startTime;
98   Serial.printf("STREAM END %dfrm %0.2ffps\n", res, 1000.0 * res /
       duration);
99 }
100
101 // flash lamp IO
102 #define LAMP 4
103
104 void setup()
105 {
```

```
106 pinMode(LAMP, OUTPUT);
107
108   Serial.begin(115200);
109   Serial.println();
110   {
111     using namespace esp32cam;
112     Config cfg;
113     cfg.setPins(pins::AiThinker);
114     cfg.setResolution(hiRes);
115     cfg.setBufferCount(2);
116     cfg.setJpeg(80);
117     bool ok = Camera.begin(cfg);
118     Serial.println(ok ? "CAMERA OK" : "CAMERA FAIL");
119   }
120   WiFi.persistent(false);
121   WiFi.mode(WIFI_STA);
122   WiFi.begin(WIFI_SSID, WIFI_PASS);
123   while (WiFi.status() != WL_CONNECTED) {
124     delay(500);
125   }
126   Serial.print("http://");
127   Serial.print(WiFi.localIP());
128   //Serial.println("  /cam.bmp");
129   //Serial.println("  /cam-lo.jpg");
130   Serial.println("/cam-hi.jpg");
131   //Serial.println("  /cam-res1.jpg");
132   //Serial.println("  /cam.mjpeg");
133   //server.on("/cam.bmp", handleBmp);
134   //server.on("/cam-lo.jpg", handleJpgLo);
135   server.on("/cam-hi.jpg", handleJpgHi);
136   //server.on("/cam-res1.jpg", handleJpgRes1);
137   //server.on("/cam.jpg", handleJpg);
138   //server.on("/cam.mjpeg", handleMjpeg);
139   server.begin();
140 }
141 void
142 loop()
143 {
144    digitalWrite(LAMP, HIGH);
145   server.handleClient();
146 }
```

3. Algoritmo de captura y guardado de las fotografías

```
from PIL import Image
import requests
from io import BytesIO
import os, sys
import datetime

datetime_object = datetime.datetime.now()
print(datetime_object)
d1 = str(datetime_object)
output = d1.replace(":","")
output = output.replace(" ","_")
output = output[0:17]+".jpg"

url = "http://192.168.12.226/cam-lo.jpg"

response = requests.get(url)
img = Image.open(BytesIO(response.content))

try:
    img.save(output)
except IOError:
    print("cannot convert", infile)

datetime_object = datetime.datetime.now()
print(datetime_object)
```

4. Algoritmo toma de datos del sensor

```
#include <esp_camera.h>
#include <Thread.h>
#include <ThreadController.h>
#include <DHT.h>
#include <WiFi.h>

#define CAMERA_MODEL_WROVER_KIT

Thread hiloDHT = Thread();

const char* ssid = "paula";
const char* password = "07210207";

WiFiServer servidorTCP(8266);
WiFiClient clienteTCP;

#define DHTPIN 14
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
ThreadController controladorHilos = ThreadController();

void leeDHT() {

  float hum = dht.readHumidity();
  float temp = dht.readTemperature();

  String datos = "Temperatura: ";
  datos+= temp;
  datos+= "\nHumedad: ";
  datos+= hum;
  datos+= "\n";
  Serial.print(datos);
  clienteTCP.print(datos);
}

void setup() {
    WiFi.begin(ssid, password);
  Serial.begin(115200);
  delay(100);

  dht.begin();

  Serial.print("\r\nConectando a: ");
  Serial.println(ssid);

  while(WiFi.status() != WL_CONNECTED){
    delay(100);
  }

  Serial.print("Conectado, IP: ");
  Serial.println(WiFi.localIP());

  servidorTCP.begin();

```

```
55    hiloDHT.onRun(leeDHT);
56    hiloDHT.setInterval(1000);
57
58
59  }
60
61  void loop() {
62
63    if (!clienteTCP.connected()){
64      clienteTCP = servidorTCP.available();
65      if (clienteTCP.connected())
66      Serial.println("Cliente conectado");
67      controladorHilos.add(&hiloDHT);
68    }
69
70   controladorHilos.run();
71  }
```

5. Algoritmo muestreo de datos del sensor

```
from tkinter import *
import socket
import threading

global hilo
global hiloRecepcion

padX = 10
padY = 20

ESP_IP = "192.168.1.43"
ESP_PORT = 8266

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

root = Tk()
root.title("Controlador")

tempVar = StringVar()
humVar = StringVar()

datos = {"Temperatura:": tempVar, "Humedad:": humVar}

frame = Frame(root)

lbl_titulo = Label(frame, text="Controlador")
lbl_titulo.grid(row=0, column=0, pady=padY, padx=padX)

lbl_temp = Label(frame, textvariable=tempVar)
lbl_hum = Label(frame, textvariable=humVar)

lbl_temp.grid(row=2, column=0, padx=padX, pady=padY)
lbl_hum.grid(row=2, column=1, padx=padX, pady=padY)
frame.pack()

s.connect((ESP_IP, ESP_PORT))

def recibeDatos():
    while True:
        cadena = s.recv(1024)
        lineas = cadena.splitlines()
        for linea in lineas:
            dato = linea.split()
            datos[dato[0].decode()].set(dato[0].decode()+" "+dato[1].
    decode())

hilo = threading.Event()
hiloRecepcion = threading.Thread(target = recibeDatos)
hiloRecepcion.setDaemon(True)
hiloRecepcion.start()

root.mainloop()
```

6. Algoritmo de detección y conteo de objetos en la imagen

```
import cv2
import numpy as np

#--------------------Leer la imagen--------------------------------

path = r'C:\Users\Paula\Desktop\polillas2\8.jpg'
im = cv2.imread(path)
cv2.imshow("Original image",im)

gris = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
cv2.imshow("Original image",gris)
#bw_img = cv2.threshold(gris,127,255,cv2.THRESH_BINARY)
#cv2.imshow("Original image",bw_img)

#-------------------Cambia el contraste de la imagen-------------

contrast_img = cv2.addWeighted(gris, 1.5, np.zeros(gris.shape,gris.dtype)
    ,0,0)
cv2.imshow("contraste",contrast_img)


#-------------------Detecci n de cuerpos-------------------------

# Setup SimpleBlobDetector parameters.
params = cv2.SimpleBlobDetector_Params()

# Change thresholds
params.minThreshold = 10 #10
params.maxThreshold = 200 #200

# Filter by Area.
params.filterByArea = True # True
params.minArea = 500 #1500

# Filter by Circularity
params.filterByCircularity = True #True
params.minCircularity = 0.1 #0.1

# Filter by Convexity
params.filterByConvexity = True #True
params.minConvexity = 0.0 #0.87

# Filter by Inertia
params.filterByInertia = True #True
params.minInertiaRatio = 0.0 #0.01

# Create a detector with the parameters
ver = (cv2.__version__).split('.')
if int(ver[0]) < 3:
    detector = cv2.SimpleBlobDetector(params)
else:
    detector = cv2.SimpleBlobDetector_create(params)

# Detect blobs.
```

```
54 keypoints = detector.detect(contrast_img)
55
56 # Draw detected blobs as red circles.
57 # cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS ensures
58 # the size of the circle corresponds to the size of blob
59 cuenta = 0
60 for i in keypoints:
61     cuenta = cuenta + 1
62
63 im_with_keypoints = cv2.drawKeypoints(contrast_img, keypoints, np.array
       ([]), (0, 0, 255), cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
64
65 # Show blobs
66 cv2.imshow("Keypoints", im_with_keypoints)
67 cv2.waitKey(0)
68
69 print(cuenta)
```

7. Cotizaciones

COTIZACIÓN PROYECTO						
No	ITEM	PROVEEDOR	CANTIDAD	VALOR UNITARIO ($)	VALOR ENVÍO ($)	VALOR TOTAL ($)
1	ESP32 CAM	Makers Chile	1	9990	4500	14490
2	MÓDULO FTDI	Makers Chile	1	2990	4500	7490
3	SENSOR TEMPERATURA Y HUMEDAD DHT11	Makers Chile	1	1690	4500	6190
4	PACK PERNOS, TUERCAS Y GOLILLAS	Easy	1	1790	6390	8180
5	KIT TRAMPA DELTA (INCLUYE TRAMPA DELTA, PISO PEGAJOSO Y FEROMONA)	ControlBest	1	7440	5500	12940
6	SOLAR CHARGER SHIELD V2.2 PARA ARDUINO	Aliexpress	1	4969	2172	7141
7	PANEL SOLAR	Aliexpress	1	5499	1318	6817
8	BATERÍA LIPO	MicroRobotics	1	6476	4528	11004
					TOTAL ($)	74252

Fig. 35: Cotizaciones materiales.
Fuente: Elaboración propia.

Links de compra para los materiales:

1. ESP32-CAM

https://makerschile.cl/producto/esp32-cam-camara-wifi-live-arduino-nodemcu-lolin-lua-iot/

2. Módulo FTDI

https://makerschile.cl/producto/ft232rl-3-3v-5-5v-ftdi-usb-a-ttl-modulo-adaptador-serial/

3. Sensor temperatura y humedad DHT11

https://makerschile.cl/producto/sensor-temperatura-y-humedad-relativa-dht11-dht-arduino-pic-2/

4. Pack pernos, tuercas y golillas

https://www.easy.cl/tienda/producto/perno-cocina-3-x-20-mm-doble-ranura-x20-unidades-importper-1164104p

5. Kit trampa delta

https://www.instagram.com/p/CHvFvXijF7b/?utm_source=ig_web_copy_link

6. Solar Charger Shield V2.2

https://n9.cl/1e4yy

7. Panel Solar

https://n9.cl/2gkc8

8. Batería LiPo DTP603450

https://www.robotics.org.za/DTP603450

8. Creación de la interfaz de usuario

```
from tkinter import *
from tkinter import messagebox
import schedule, time, threading, sys, datetime, os, requests, cv2
from PIL import Image
from io import BytesIO
import urllib.request as urllib
import numpy as np

raiz = Tk()
raiz.state("zoomed")


#--------------------Crear la raiz
    ------------------------------------------------

raiz.title("Monitoreo de Polillas") #t tulo de la ventana
raiz.iconbitmap("C:\\Users\Paula\Desktop\Polillas\polilla.ico") #icono de
    la ventana
raiz.geometry("1000x600") #dimensiones por defecto de la ventana
raiz.config(bg="grey") #cambia el color de fondo de la ventana


#-------------------------Salir de la aplicaci n
    ------------------------------

def salir():
    valor = messagebox.askquestion("Salir", "  Est   seguro que desea
    salir de la aplicaci n?")

    if valor == "yes":
        raiz.destroy()


#--------------------Crear el frame
    ------------------------------------------------

miFrame = Frame(raiz)
#miImagen = PhotoImage(file="C:\\Users\Paula\Desktop\Polillas\campo_fondo
    .gif") #poner una imagen
#otroLabel = Label(miFrame, image=miImagen)
#otroLabel.place(x=10, y=10) #ubicacion de la imagen
miFrame.pack(fill="both", expand="True") #el frame se expande al tama o
    de la raiz
miFrame.config(bg="grey", width="1000", height="600", cursor="arrow") #
    color de fondo del frame, tama o del frame, cursor


#-------------------------Botones
    ------------------------------------------------

def botones():
    ingresar = Label(miFrame, text="pagina1", font=("Arial", 10), bg="
    grey", width=200, height=100)
    ingresar.place(x=0, y=0)
```

```
45
46      Inicio = Button(miFrame, text="INICIO", font=("Arial", 30), width=10,
        height=2, bg="darkorange", bd=5, activebackground="saddlebrown",
    command=start)
47      Inicio.place(x=410, y=250)
48      Inicio.config(cursor="hand2")
49
50      Stop = Button(miFrame, text="STOP", font=("Arial", 30), width=10,
    height=2, bg="darkorange", bd=5, activebackground="saddlebrown",
    command=stop)
51      Stop.place(x=710, y=250)
52      Stop.config(cursor="hand2")
53
54      BaseDatos = Button(miFrame, text="""Base de
55 Datos""", font=("Arial", 30), width=10, height=2, bg="darkorange", bd=5,
    activebackground="saddlebrown", command=contar_obj) #""" triple
    comilla para saltos de linea en el texto
56      BaseDatos.place(x=410, y=450)
57      BaseDatos.config(cursor="hand2")
58
59      Fotos = Button(miFrame, text="Fotos", font=("Arial", 30), width=10,
    height=2, bg="darkorange", bd=5, activebackground="saddlebrown",
    command=abrir_carpeta)
60      Fotos.place(x=710, y=450)
61      Fotos.config(cursor="hand2")
62
63
64 #-------------------------Bienvenida
    ------------------------------------------
65
66 Bienvenida = Label(miFrame, text="Bienvenido al Monitoreo de Polillas",
    font=("Arial", 50), bg="grey")
67 Bienvenida.place(x=160, y=150)
68
69 ingresar = Button(miFrame, text="Ingresar", command=botones, font=("Arial
    ", 20), bg="darkorange", bd=5, activebackground="saddlebrown", cursor=
    "hand2")
70 ingresar.place(x=610, y=300)
71
72 #-----------------------Funci n de inicio
    ------------------------------------
73
74 def start():
75      global termino
76      global t
77      messagebox.showinfo("INICIO", "La captura de fotograf as ha
    comenzado")
78      termino = threading.Event()
79      t = threading.Thread(target=ciclo_de_tiempo, args=(termino, "task"))
80      t.start()
81
82
83 #-------------------------Funci n de stop
    ------------------------------------
84
```

```
85 def stop():
86     valor = messagebox.askquestion("STOP", "  Est   seguro que desea
       detener la captura de fotograf as?")
87
88     if valor == "yes":
89         global termino
90         global t
91         print("La captura ha finalizado.")
92         termino.set()
93         t.join()
94         messagebox.showinfo("STOP", "La captura de fotograf as ha
       finalizado")
95
96
97 #------------------------------Base de datos
       ------------------------------------
98
99 def nueva_pag():
100    label = Label(miFrame, text="P gina de Base de Datos", font=("Arial"
       , 20), bg="grey", width=200, height=200)
101    label.place(x=20, y=20)
102
103    Volver = Button(miFrame, text="Volver", font=("Arial", 20), bg="
       darkorange", bd=3, activebackground="saddlebrown", command=botones)
104    Volver.place(x=0, y=0)
105    Volver.config(cursor="hand2")
106
107
108 #-----------------Para abrir la carpeta de las fotos
       ------------------------
109
110 def abrir_carpeta():
111    path = "C:/Users/Paula/Documents/UTAL"
112    path = os.path.realpath(path)
113    os.startfile(path)
114
115
116
117 raiz.protocol("WM_DELETE_WINDOW", salir)
118 raiz.mainloop()
```