

**Práctica Profesional I, 620449**

**Estudiante Practicante:**

Luis Pereira

**Profesor supervisor:**

Luis Vera

**Fecha:**

11-01-202

**Índice**

**Introducción**

La presente documentación brinda un análisis profundo de la destacada experiencia vivida por un estudiante de Ingeniería Civil en Informática de la Universidad del Bío-Bío durante su práctica profesional en el área de Informática Industrial. Esta práctica se llevó a cabo con el objetivo principal de contribuir al desarrollo y perfeccionamiento de guías didácticas y experiencias de laboratorio enfocadas en la automatización, mediante la aplicación de Python.

En línea con los principios fundamentales de la práctica, el estudiante se inmersión en la tarea de no solo desarrollar y actualizar guías didácticas, sino también elevar la calidad de las experiencias de laboratorio vinculadas a la automatización con Python. Estos objetivos meticulosamente delineados se alinean estratégicamente con la visión educativa de proporcionar a los estudiantes herramientas prácticas y contemporáneas que fortalezcan su comprensión y habilidades en el ámbito de la Informática Industrial.

Cabe destacar que esta práctica se llevó a cabo en el Centro de Investigación en Manufactura Integrada por Computadora (CIM) del laboratorio de la Universidad del Bío-Bío. Este entorno, equipado con tecnología de vanguardia, proporcionó el escenario propicio para el desarrollo de proyectos avanzados, tales como la programación para el control de PLC, la exploración de la comunicación serial y socket, la incursión en la visión artificial, la interacción con robots industriales, y la prestación de soporte directo a estudiantes.

Dentro de este contexto más amplio, la práctica no solo fue un espacio para la adquisición y aplicación de conocimientos específicos, sino también una oportunidad para que el estudiante se convirtiera en un facilitador activo del aprendizaje colectivo, compartiendo sus experiencias y conocimientos con sus pares. Este enfoque colaborativo y de apoyo directo demuestra la solidaridad y el espíritu comunitario del estudiante, contribuyendo al fortalecimiento del entorno académico en la Universidad del Bío-Bío.

A lo largo de las próximas secciones de este informe, se detallarán las actividades desarrolladas, los resultados alcanzados y la profundización en las áreas temáticas avanzadas, ofreciendo una visión completa y enriquecedora de la contribución de esta práctica al desarrollo académico y profesional del estudiante, así como al avance continuo de los estándares educativos de la universidad.

**Cuerpo del informe**

1. **Descripción del centro de practica**

* **Nombre o razón social:** LABORATORIO DE SISTEMAS INTEGRADOS DE MANUFACTURA – CIMUBB
* **Representante legal:** Luis Vera Quiroga
* **Dirección postal**: Collao 1399, 4051145 Concepción, Bío Bío
* **Sitio Web:** www.ubiobio.cl
* **Teléfono:** 56 (41) 311 1308
* **Identificación y Descripción del área en la cual hizo su práctica:** Informática Industrial



**DESCRIPCIÓN DE LA PRÁCTICA**

* **Resumen General de mi Práctica Profesional I**

Durante mi práctica profesional I en el área de Informática Industrial, realizada en el Centro de Investigación en Manufactura Integrada por Computadora (CIM) del laboratorio de la Universidad del Bío-Bío, me propuse contribuir al desarrollo y mejora de guías didácticas y experiencias de laboratorio relacionadas con la automatización mediante el uso de Python.

* **Objetivos de mi Práctica Profesional I:**

1. **Desarrollo de Guías Didácticas**: Me centré en actualizar y mejorar las guías didácticas existentes, asegurando su relevancia y utilidad para los estudiantes de Informática Industrial.
2. **Experiencias de Laboratorio:** Trabajé en la creación de nuevas experiencias prácticas de laboratorio, específicamente diseñadas para enriquecer la formación de mis compañeros de carrera.
3. **Nivelación en Python y Arduino:** Dedique esfuerzos a fortalecer mis habilidades en los lenguajes de programación Python y Arduino, consolidando así una base sólida para la implementación de soluciones prácticas.
4. **Desarrollo de Aplicaciones para el Laboratorio:** Participé activamente en la creación de nuevas aplicaciones prácticas alineadas con los contenidos curriculares, con el objetivo de mejorar la experiencia de aprendizaje en el laboratorio.
5. **Integración en una Aplicación para CIM:** Trabajé en la integración de diversas experiencias en una aplicación central destinada a coordinar eficientemente el sistema de Manufactura Integrada por Computadora (CIM).
6. **Documentación Tutorial:** Generé documentación detallada en formato tutorial, facilitando así la comprensión y aplicación de las experiencias prácticas de laboratorio por parte de mis compañeros.
7. **Soporte Remoto:** Me comprometí a trabajar en el proyecto, incluso desde mi hogar, para garantizar flexibilidad en los horarios y adaptarme a posibles limitaciones de presencialidad, asegurando la continuidad del proyecto.

* **Breve Descripción de mis Actividades Realizadas:**

1. A lo largo de la práctica, ejecuté una variedad de actividades, desde la nivelación en Python y Arduino hasta el desarrollo de soluciones avanzadas. Actualicé y probé guías de laboratorio existentes, creé nuevas aplicaciones para enriquecer las experiencias prácticas y trabajé en la integración de conocimientos en una aplicación central para el sistema CIM.
2. Además, asumí el compromiso de proporcionar soporte remoto, demostrando así mi flexibilidad y responsabilidad en la gestión del proyecto. Mi práctica no solo consistió en la aplicación de conocimientos teóricos, sino que también incluyó la colaboración activa con otros estudiantes, compartiendo experiencias y conocimientos para fortalecer nuestro aprendizaje colectivo. En resumen, mi práctica fue una inmersión completa en el campo de la Informática Industrial, combinando teoría y práctica de manera efectiva para alcanzar los objetivos propuestos.

* **Actividad (es) realizada(s). Para cada una de las actividades realizadas debe indicar: tareas específicas, áreas involucradas del Centro de Práctica, métodos y técnicas, así como herramientas**

**y plataformas hardware y/o software usadas. Finalmente debe indicar resultados obtenidos.**

**Actividades realizadas**

1. **Capacitación en Python para procesamiento de imagen con webcam:**

No estuve involucrado en ninguna área del Centro de Practica por el momento, solo recibí una serie de tutoriales para estudiar desde mi casa de forma remota, los tutoriales que revisé eran de procesamiento de imagen con webcam, donde se utilizó mucho las librerías tkinter para las interfaces gráficas y OpenCV (Open Computer Visión Library) en Python, los tutoriales revisados tenían como fin, capacitarme en Python para luego actualizar todos los programas e incluir mejoras en ellos, corregir problemas y dejar un código más limpio, esta actividad duro hasta casi finales de septiembre, luego de finalizar esta capacitación, domine la librería tkinter para la creación de interfaces gráficas, aprendí mucho acerca de la librería OpenCV para visión artificial, aprendí diversos métodos para realizar seguimiento en tiempo real, umbralización, aprendí que la imagen que captura la webcam viene en escala BGR (Blue Green Red) y que para procesarla debemos pasarla a escala de RGB (Red Green Red), también que podemos realizar umbralización la imagen con cierto umbral que nosotros establezcamos, aprendí a realizar análisis de manchas, dado que todo lo que iba haciendo se juntaba en una aplicación grande al finalizar el tutorial, tenia la oportunidad de retroalimentar con todo lo anteriormente aprendido.

Tabla

Descripción generada automáticamente

Figura 1

­­

1. **Actualización y revisión de primeros programas del tutorial**

En mi rol actual, me asignaron la tarea de revisar y actualizar tutoriales, específicamente aquellos relacionados con la instalación de Python y Visual Studio Code. Al abordar el tutorial de instalación, noté que las versiones en las imágenes estaban desactualizadas debido al tiempo transcurrido desde su creación. Por lo tanto, procedí a actualizar las imágenes y verificar la coherencia de los pasos en el tutorial. Además, exploré y probé el programa de "Iniciar cámara", intentando implementar un ciclo while basándome en prácticas de YouTube. Sin embargo, rápidamente me di cuenta de que esta técnica no funcionaba de manera óptima y descubrí la importancia de la librería PIL, las variables globales, y la utilidad de llamar a funciones con el método after para ejecutarlas recursivamente con un intervalo de tiempo en milisegundos.

Posteriormente, me centré en mejorar el programa de "Tomar foto usando webcam". El desafío aquí fue dotar al programa de funcionalidades como seleccionar una carpeta, asignar un nombre a las imágenes, guardarlas y establecer un temporizador para capturar fotos automáticamente. En este proceso, me topé con la necesidad de incorporar la librería file para manejar la apertura de carpetas desde el backend del programa. Surgió un problema adicional cuando las imágenes se sobrescribían debido a nombres idénticos, lo cual abordé implementando un contador para evitar repeticiones.

Otra tarea que se me encomendó fue actualizar el programa "Tomar foto usando webcam y transformarla a escala de grises". Esta tarea resultó ser más sencilla, ya que pude aprovechar gran parte del código existente. La principal modificación se centró en cambiar la función de captura de imagen para transformarla de escala BGR a escala de grises mediante el método COLOR\_BGR2GRAY de la librería cv2.

En este proceso de mejora, adquirí conocimientos valiosos, como la lectura y escritura de archivos, la implementación de temporizadores, la gestión de errores mediante try y except, así como la práctica de bloquear botones hasta que se cumplan ciertas condiciones para evitar problemas potenciales. Además, aprendí a evitar la repetición de nombres de archivos dentro de una carpeta mediante el uso de un contador y un condicional. Como parte de este proceso, también desarrollé un tutorial completamente nuevo, detallando cada función utilizada para proporcionar una comprensión más profunda de los cambios realizados.

A continuación, mostrare imágenes del programa actualizado, dado que el programa “Tomar foto usando webcam en escala de grises” es idéntico a “Tomar foto usando webcam”, pondré la imagen de un programa con tal de evitar agregar demasiadas imágenes y alcanzar a poner la mayor cantidad de actividades que realice durante mi práctica profesional.

Texto

Descripción generada automáticamente con confianza media

Programa sin actualizar

Texto

Descripción generada automáticamente con confianza media

Programa actualizado

1. **Actualización de programas para umbralización y análisis de manchas**

Una vez finalizada la actividad anterior me puse en contacto con el supervisor de forma remota, para saber que sigue, se me asigno esta vez actualizar el programa de “Umbralizacion RGB”, en forma remota, este programa estaba mal debido a que realizaba una umbralización en escala de grises, la idea era actualizar este programa para que la imagen capturada en BGR que luego se transforma a RGB, dividirla en 3, y ahora como cada una esta en blanco y negro debido a que no están fusionadas, puedo realizar la umbralizacion cada una por separado, agregue esta nueva función creada por mi:

Texto

Descripción generada automáticamente

Figura 3.1 “Función para realizar umbralización”

La función en cuestión está diseñada para llevar a cabo la umbralización de una imagen, y su comportamiento depende del parámetro que recibe. Este parámetro determina el método de umbralización que se aplicará utilizando la función threshold de la librería cv2 (OpenCV). Es importante destacar que los valores de umbral utilizados no son fijos, sino que son proporcionados por el usuario a través de la interfaz del programa.

Para realizar la umbralización, la función sigue los siguientes pasos:

1. **División de la imagen en canales RGB:**

Utiliza la función split de cv2 para separar la imagen en sus tres canales de color: rojo (R), verde (G) y azul (B).

1. **Umbralización de cada canal:**

Aplica el método de umbralización a cada uno de los canales por separado, utilizando los umbrales proporcionados por el usuario a través de la interfaz del programa. En este punto, se están creando tres versiones umbralizadas de la imagen, una para cada canal.

1. **Unión de los canales umbralizados:**

Utiliza la función merge de cv2 para combinar nuevamente los tres canales umbralizados en una sola imagen.

1. **Mostrar la imagen umbralizada:**

Dependiendo del parámetro que se haya recibido, la función mostrará la imagen umbralizada en un cuadro específico de la interfaz gráfica. Esto se relaciona con la figura 2.3 en la interfaz del programa.

Es importante notar que el diseño de la función se ha optimizado para evitar la creación de funciones adicionales. Además, se ha aprovechado la capacidad de asignar funciones lambda como comandos al presionar un botón en la interfaz. Esto permite enviar un parámetro específico dependiendo del botón presionado, lo que facilita la adaptabilidad de la función a diferentes contextos dentro del programa. Este enfoque se visualiza en el código mediante la asignación de una lambda al comando del botón, lo que proporciona flexibilidad en la ejecución de la umbralización según las necesidades del usuario o la interfaz gráfica.

Texto

Descripción generada automáticamente

Figura 3.2 “botón con acción de ejecutar función con cierto parámetro”

Imagen que contiene Texto

Descripción generada automáticamente

Figura 3.3: “Programa umbralización RGB sin actualizar”

Imagen que contiene PowerPoint

Descripción generada automáticamente

Figura 3.4: “Programa umbralización RGB actualizado”

También se me asignó la tarea de actualizar el programa "Análisis de manchas" de forma online, lo cual implica realizar la umbralización en una imagen y contar tanto las manchas negras como las blancas. La nueva funcionalidad que se buscaba incorporar incluía dos cajas de texto y dos botones adicionales para analizar las manchas blancas y negras respectivamente.

En este proceso, me enfrenté a un desafío, ya que no existía un método directo para contar contornos blancos en la imagen. Sin embargo, durante mi investigación online, descubrí una función que invertía la imagen. Aprovechando esta táctica, decidí invertir la imagen umbralizada en escala de grises según un umbral seleccionado por el usuario. Dado que la imagen estaba invertida, al analizar las manchas negras en realidad estábamos contando las manchas blancas en la imagen original.

Para calcular el porcentaje de manchas negras, utilicé la función countNonZero de cv2 para obtener la cantidad de píxeles negros y luego dividí esta cantidad por el total de píxeles en la imagen. Repetí el mismo procedimiento para calcular el porcentaje de manchas blancas, pero en este caso, la imagen invertida se utilizó como parámetro.

Además, desarrollé un programa adicional que permite cargar una imagen y devuelve el porcentaje de manchas blancas y negras, manteniendo todo el proceso de manera online. Este enfoque proporciona una manera eficaz de analizar y comparar las manchas blancas y negras en una imagen umbralizada, ampliando así la funcionalidad del programa original, todo ello realizado de forma remota.

Imagen que contiene Texto

Descripción generada automáticamente

Figura 3.5 “Programa Análisis de manchas sin actualizar y con problemas”

Texto

Descripción generada automáticamente

Figura 3.6 “Programa Análisis de manchas Actualizado”

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Figura 3.7 “Programa Analizar imagen creado”

Finalmente para terminar la tercera actividad se me asigno actualizar el programa final que era la unión de gran parte de las funcionalidades incluidas durante esta actividad pero ahora permite obtener las coordenadas del cuadro, recortar una imagen y realizar todas las operaciones vistas durante la actividad 3, había que arreglar algunos problemas con el programa final sin actualizar, debido a que no realizaba una umbralización RGB correctamente, no recortaba bien la imagen, no incluía el análisis de manchas blancas y negras, entonces la idea es que el programa que yo cree ahora, contenga esas funcionalidades y haga bien la umbralización, ahora solo debe fusionar los canales en vez de mostrarlo uno por uno como en la figura 3.4, durante el desarrollo de este programa no hubo mayores dificultades ya que las funciones ya estaban creadas y solo fue integrarlas al programa final, el resultado fue:

Interfaz de usuario gráfica, Aplicación, Sitio web

Descripción generada automáticamente

Figura 3.8 “Programa final sin actualizar”

Interfaz de usuario gráfica, Aplicación, Sitio web

Descripción generada automáticamente

Figura 3.9 “Programa final actualizado”

**4) Creación de programas para detección, análisis de rectángulos y seguimiento**

En la cuarta actividad que me fue asignada, trabajé de forma remota para desarrollar programas en Python que pudieran detectar rectángulos, analizar estos rectángulos, y realizar un seguimiento de puntos en una cuadrícula. Mi labor comenzó con la revisión de nuevos tutoriales proporcionados por practicantes anteriores que habían trabajado en proyectos similares.

El objetivo principal era actualizar un programa de detección de rectángulos, en el cual se utilizó el método findContours de la librería cv2 para contar los contornos de una imagen capturada desde una cámara web, previamente convertida en una imagen binaria para facilitar la detección de contornos. Se estableció un umbral mínimo de área para reconocer un rectángulo, basándose en los contornos y su área.

Pantalla de computadora con letras

Descripción generada automáticamente con confianza media

Figura 4.1: ”Parte de programa detección de rectángulos”

La estrategia implementada implicaba identificar posibles vértices de los rectángulos utilizando el método approxPolyDP de cv2. Cuando se encontraban 4 posibles vértices, se procedía a dibujar líneas mediante el método drawContours de cv2. Adicionalmente, se incorporó un contador para llevar un registro de la cantidad de rectángulos detectados en la imagen.

Forma

Descripción generada automáticamente

Figura 4.2: “Resultado programa detección de rectángulos”

Ahora terminado este programa debía crear un programa que nos permita iniciar cámara, detecte rectángulos, los muestre en un cuadro el rectángulo recortado, permita capturarlo en escala de grises y que se guarden en un tercer cuadro para que podamos umbralizarlo de acuerdo a un umbral establecido por el usuario, no se utilizó ningún método distinto a los utilizados anteriormente, solo fue integrar funciones de otros programas y armar la interfaz gráfica.

Interfaz de usuario gráfica

Descripción generada automáticamente

Figura 4.3: “Creación de programa procesamiento de rectángulos”

En la continuación de este proyecto, la tarea consiste en crear un programa de seguimiento de un punto que permita monitorear continuamente un área específica. La cámara estaría encendida las 24 horas del día, apuntando hacia un punto donde se espera que haya variaciones mínimas, relacionadas con el proceso de secado y la humedad de las maderas.

Para implementar este programa, se han utilizado diversas técnicas:

* **Transformada de Hough para Detectar Líneas:**

La transformada de Hough se emplea para identificar líneas en la imagen. Este paso es útil para analizar las variaciones y características en el área monitoreada.



* **Seguimiento Óptico de Lucas-Kanade:**

Se utiliza el método de seguimiento óptico de Lucas-Kanade para rastrear un punto de interés a lo largo del tiempo en secuencias de imágenes. Esto permite seguir los cambios en el punto seleccionado, crucial para monitorear la variación esperada en el área de interés.

Texto

Descripción generada automáticamente

Figura 4.5: “Seguimiento óptico de Lucas-Kanade:”

* **Umbral de Canny para Identificar Transiciones de Intensidad:**

Se aplica un umbral de Canny para identificar transiciones de intensidad significativas en la imagen. Esto contribuye a resaltar los cambios importantes en la escena, facilitando la detección de puntos de interés.

Texto

Descripción generada automáticamenteFigura 4.6: “Umbral Canny”

* **Captura y Almacenamiento de Coordenadas:**

El programa captura las coordenadas del punto de interés a intervalos regulares, probablemente cada segundo, y almacena estas coordenadas en un archivo. Este archivo servirá como un registro temporal de la posición del punto.

Pantalla de computadora con números

Descripción generada automáticamente con confianza media

Figura 4.5: “Función para encontrar los puntos”

Texto

Descripción generada automáticamente

Figura 4.6: “Rastreo, seguimiento y dibujo de punto”

Durante el desarrollo de esta actividad, enfrenté desafíos significativos, especialmente en la implementación de programas para la detección de puntos y el procesamiento de rectángulos. La necesidad de aprender nuevas técnicas y métodos en línea fue esencial para superar estos obstáculos.

En el caso del programa de procesamiento de rectángulos, una de las dificultades fue lograr que el programa recortara automáticamente el rectángulo capturado desde la cámara. La solución implicó establecer condiciones específicas y utilizar métodos para contar contornos, lo cual resultó fundamental para el éxito del programa.

En cuanto al programa de seguimiento de puntos, enfrenté retos similares. La tarea de investigar y comprender la transformada de Hough para la detección de líneas, utilizarla para encontrar intersecciones y, finalmente, aplicar un umbral Canny con el método Canny, además de aprender a utilizar el método Lucas-Kanade para rastrear un punto de interés en el tiempo, se convirtió en un desafío significativo. A pesar de las dificultades, la combinación de estas técnicas permitió alcanzar el objetivo del programa.

**5) Creación, presentación y soporte en CIM de programas**

Durante esta actividad desarrolle 3 programas en Python, uno llamado “Recurrencias”, el cual creaba una matriz 5x5 con valores aleatorios entre el 0 y 255 para luego guardar la matriz con los valores separados por espacio en un archivo de texto, esto con el fin de simular las intensidades de los diferentes colores que conforman una imagen, el programa hasta ahora no incluía ningún método que no se haya usado con anterioridad.

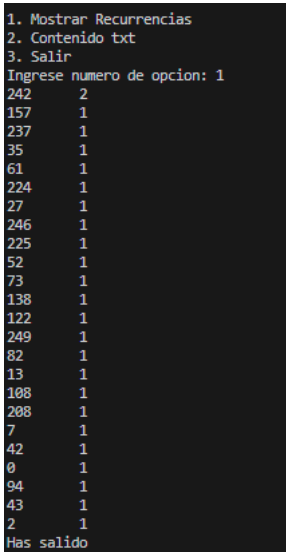


Figura 5.1: “Resultado programa en consula recurrencias”

El segundo programa asignado implica la creación de una interfaz gráfica que realiza una tarea similar al programa anterior. En este caso, se llena una matriz cuadrada de tamaño N con valores aleatorios entre 0 y 255. El propósito sigue siendo simular el análisis de intensidades de colores en las imágenes.

La dificultad principal surgió al tratar de crear un gráfico que mostrara la recurrencia de todos los valores entre 0 y 255. Para abordar este problema, se utilizó la librería matplotlib de Python para la creación del gráfico. Durante este proceso, se encontró un obstáculo al intentar mostrar los valores en el eje de las abscisas. La solución se alcanzó mediante la creación de la función de la "figura 5.2".

Adicionalmente, se adquirió conocimiento sobre cómo ajustar el tamaño de las barras en el gráfico al manipular los parámetros del método bar de la librería matplotlib, modificar nombre de los ejes de ordenadas y abscisas, probar distintos tipos de gráficos, abrir y leer archivos, esto permitió personalizar la presentación visual del gráfico según las necesidades del programa.

Texto

Descripción generada automáticamente

Figura 5.2: “Función para mostrar histograma”

El resultado obtenido fue:

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

Figura 5.3: “Resultado programa histograma”

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente Gráfico, Histograma

Descripción generada automáticamente

Figura 5.4: “Recurrencias de cada valor” Figura 5.5: “Histograma”

Tuve que ir al laboratorio CIM dado que el computador del profesor Luis Vera no reconocía bien el programa por falta de algunas librerías, por lo que fui a dar soporte técnico y también le mostré a los estudiantes como utilizar el programa.

Los dos programas previos, el relacionado con recurrencias y el de histograma, tenían como objetivo proporcionar una comprensión más profunda del funcionamiento del back-end de un programa. Ahora, me enfoco en describir el nuevo programa que debía crear en Python. Este programa tiene la función de ecualizar una imagen cargada desde la PC. La ecualización busca que las intensidades de la imagen estén más uniformemente distribuidas, como se observa en la figura 5.7. Al comparar las intensidades de la imagen sin ecualizar (12000) con la imagen ecualizada (4000), se evidencia una mejora en la uniformidad, lo que facilita el trabajo con las imágenes.

Para lograr esto, realicé investigaciones en plataformas como YouTube y desarrollé una interfaz gráfica. Descubrí que para la ecualización, era necesario utilizar el método equalizeHist de la biblioteca OpenCV (cv2), como se muestra en la figura 5.6. Este método se emplea específicamente para mejorar la distribución de las intensidades de píxeles en una imagen, lo que conduce a una mejora en la calidad visual general de la imagen ecualizada. En resumen, este nuevo programa proporciona una herramienta eficaz para ecualizar imágenes, mejorando así la uniformidad de las intensidades y facilitando su posterior manipulación.

Texto

Descripción generada automáticamente

Figura 5.6: “Método para ecualizar imagen”

El resultado del programa fue exitoso y también fui al CIM a mostrar el programa debido a problemas con el computador del profesor.

Interfaz de usuario gráfica

Descripción generada automáticamente

Figura 5.7: “Resultado programa Histograma”

6) Creación de programa con template matching

Desarrollé un programa en el Centro de Innovación y Manufactura (CIM) para realizar pruebas constantes de detección de patrones. Utilicé una cámara web en el laboratorio y seguí un tutorial proporcionado por el profesor Luis Vera. El programa, creado en Python con el método de coincidencia de plantillas de OpenCV (cv2), permite capturar un objeto con un patrón desde la cámara, guardar plantillas en una carpeta y compararlas. Establecí un umbral de coincidencia en 0.7 para indicar la presencia de coincidencias entre la imagen capturada y las plantillas.

Además, desarrollé un programa adicional para generar templates. Modifiqué un programa de recorte de imágenes para incluir la funcionalidad de guardar las imágenes. También creé otro programa que, además de realizar la coincidencia de plantillas, recorta automáticamente el rectángulo de madera y realiza un análisis de manchas blancas y negras. Esto es crucial ya que ciertas piezas con un patrón específico pasarán por un control PLC, y el programa debe realizar el análisis para múltiples piezas.

Interfaz de usuario gráfica

Descripción generada automáticamente

Figura 6.1: “Programa template matching”

Interfaz de usuario gráfica

Descripción generada automáticamente

Figura 6.2: “Programa de inspección automatizada”

**7) Creación de programas para comunicación serial y socket**

Se me asignó la tarea de actualizar un programa que se comunica a través de comunicación serial con Arduino. La mayor parte del desarrollo tuvo lugar en el Centro de Innovación y Manufactura (CIM). Inicialmente, el programa encendía luces rojas, azules y verdes en una protoboard. Mi objetivo era mejorar el programa para que funcionara con un Arduino.

Para llevar a cabo este proyecto, utilicé Arduino IDE, donde creé un código con sintaxis de C para establecer la comunicación con mi programa en Python, que actúa como el medio para enviar señales al Arduino. Con la actualización, ahora el Arduino tiene la capacidad de encender luces rojas, amarillas y azules, y puede manejar luces de cualquier valor gracias a su luz RGB con tres puertos.

En el entorno Arduino IDE, instalé la librería DHT para permitir que el programa envíe señales al Arduino y este devuelva, en bytes, los valores de temperatura y humedad en tiempo real. Estos valores se decodifican y se muestran en pantalla. Utilicé la librería serial de Python, junto con los métodos write y decode, y el carácter de retorno "\r" para cumplir con el objetivo.

A pesar de enfrentar varias dificultades, se logró probar el código con éxito por parte de los estudiantes del IPP. Se realizaron pruebas utilizando un Arduino Mega del laboratorio CIM, y el programa funcionó correctamente.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Figura 7.1: “Programa Comunicación Serial Arduino y Camara”

El motivo de porque se puede iniciar cámara es debido a que no todos los estudiantes IPP pueden probar el programa así que se dejara un computador con una cámara apuntando al Arduino y un programa para tener acceso al computador, entonces los estudiantes tienen acceso 24/7 al programa.

El programa enfrentó un problema al intentar enviar señales al robot Scorbot en el laboratorio CIM a través de la comunicación serial. Inicialmente, al crear un programa para el robot y enviar mensajes, no se recibía la respuesta esperada. La investigación reveló que el problema estaba relacionado con la capacidad del programa para leer datos de manera efectiva mientras se ejecuta simultáneamente.

La solución encontrada fue utilizar la librería threading, que permite crear hilos secundarios en el programa. En este caso, se abrió un hilo secundario utilizando la librería threading para leer datos continuamente desde el puerto serial. Este enfoque resolvió el problema al asegurar que el programa estuviera siempre listo para recibir datos del robot Scorbot, lo que facilitó la comunicación bidireccional efectiva.

Después de implementar esta solución, el programa pudo recibir mensajes específicos de fábrica enviados por el robot Scorbot. Además, se realizaron pruebas exitosas al enviar comandos desde el programa para controlar el movimiento del robot, retirar o devolver objetos, demostrando así la capacidad funcional del programa en la interacción con el robot Scorbot a través de la comunicación serial.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Figura 7.3: “Controlador Scorbot”

En el último programa que desarrollé en el CIM para el control de paletts y estaciones, me enfrenté a la tarea de integrar el control PLC con el programa del robot Scorbot. La tesis y documentación proporcionadas incluían una amplia variedad de comandos para detener palettes en estaciones específicas y liberar estaciones. Para lograr esta integración, fue necesario ajustar los parámetros de la conexión serial al robot.

En particular, se modificaron los parámetros de conexión para ajustarse a los requisitos específicos de la comunicación. Cambié el byte size a 7, la paridad a "even" y el número de bits de stop a 2. Estos ajustes se realizaron para garantizar una configuración precisa de la comunicación serial entre el programa y el robot Scorbot.

La adaptación de estos parámetros fue esencial para lograr una comunicación efectiva y confiable entre el programa de control y el robot. Con estas modificaciones, el programa fue capaz de interpretar y ejecutar los numerosos comandos del PLC, permitiendo así el control preciso de las paletas y estaciones en el entorno de trabajo.



Figura 7.4: “Parámetros conexión robot control PLC”

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Figura 7.5: “Control PLC”

1. Conclusiones de la práctica. Debe indicar los beneficios obtenidos por la empresa producto

de su cometido, así como los beneficios personales obtenidos por usted en cuanto a lo aprendido,

sean estos conocimientos, habilidades o aptitudes (considerando el perfil de la carrera).

1. Bibliografía. Debe ajustarse al formato APA, agregando la referencia en el cuerpo del informe.