Control de un Almacén Automatizado por Medio de Python y con una Interfaz Gráfica

Adrian Gomez Alvarez[1], Dr. Luis Carlos Méndez González[2], Dr. Luis Alberto Rodríguez Picón[3], Dr. Iván Juan Carlos Pérez Olguín[4]

Resumen— Desde tiempos antiguos el ser humano se ha desempeñado en la tarea de la búsqueda de la automatización de los sistemas de almacenaje y aun que en gran cantidad y de manera visible se han presentado cambios y avances notables, nunca es suficiente para el hombre, por lo que gracias a esto existen una variedad amplia de diferentes sistemas y métodos de almacenaje hoy en día desde un almacenaje manual hasta uno automatizado encontramos en distintas empresas de la industria. Este proyecto busca facilitar y estandarizar el método de almacenaje en las empresas para una mayor efectividad y reducción de tiempos en cuanto al proceso de almacenaje. Para lograr este propósito se realizo una interfaz gráfica mediante el programa visual estudio code con la utilización del lenguaje Python y su debida programación para poder establecer una comunicación Modbus TCP/IP Server con el programa Factory I/O y de esta forma automatizar un sistema de almacenaje mostrando una interfaz gráfica al usuario para que este maneje de manera más rápida y segura el almacenamiento de cajas sobre pallets en un rack.

Palabras clave— automatización, sistemas de almacenaje, estandarizar, interfaz gráfica.

Introducción

Los almacenes son uno de los sectores con mayor éxito en desarrollo durante los últimos 10 años en todo el mundo (Hristov, Slavov, Damyanov, & Mladenov, 2022), los sistemas de almacenaje automatizado son la tecnología que nos permite la automatización del proceso de almacenamiento y recuperación de productos dentro de un almacén o centro de distribución. Dichos sistemas utilizan tecnología avanzada, como robots y sistemas de software de gestión de almacenes, de esta forma se pueden realizar tareas de almacenamiento y recuperación de productos de manera rápida y eficiente. Los sistemas de almacenaje automatizado pueden incluir diferentes tipos de tecnologías, como estanterías automatizadas, transportadores automatizados, vehículos guiados automatizados etc. Estos sistemas tienen varios beneficios, como una mayor eficiencia en el manejo de productos, una reducción de errores en la gestión de inventario y una mayor seguridad en el almacén. También pueden ayudar a aumentar la capacidad de almacenamiento y a reducir los costos operativos. Este proyecto se encuentra enfocado en la utilización de un lenguaje de programación Python con interfaz gráfica y una comunicación Modbus TCP/IP Server para el control de un almacén en el software de Factory I/O. Este método tiene como objetivo demostrar el control y correcto desempeño de un almacén por medio de un lenguaje de programación haciendo uso de interfaz gráfica, para lograrlo se establece la comunicación Modbus TCP/IP Server en el programa Factory I/O donde se almacenarán en un rack cajas sobre pallets en una escena de simulación. El desempeño se evaluará de acuerdo con el correcto funcionamiento final de esta simulación.

Descripción del Método

Los sistemas de almacenaje aun presentan algunos problemas notables e importantes. Las personas que día a día trabajan en estos procesos pueden observar diversos errores que influyen por el uso de la maquinaria o los sistemas mecanizados, así como otros factores como situaciones personales o variables del entorno en donde se desempeñan las actividades. El estado de ánimo, la fatiga, las condiciones de luz, piso, área o la comodidad en el trabajo son ejemplos de los factores que pueden afectar el proceso. Debido a estos factores el proceso se puede volver lento, poco productivo, inseguro y/o impreciso por lo que podría ser necesario aplicar más métodos de solución como el aumento de la mano de obra, aumento de espacio en área para almacenaje o equipo extra de almacén para facilitar trabajo, pero esto no nos asegura la efectividad lo que ocasionaría mayores gastos en nuestros procesos.

Los sistemas de almacenaje por automatización pueden solventar los problemas que involucran la mano de obra extra, aumentos de espacios en pisos sobre almacenaje etc. Estos sistemas se utilizan para simplificar procesos, usos de maquinaria y de movilidad de cierta complejidad, en los que otros sistemas son más tediosos, y generan mayor gasto en vez de aportación y de difícil solución. La aplicación de este tipo de tecnología en los procesos de almacenaje no solo puede mejorar la seguridad, forma, tiempo, calidad y la eficiencia del proceso, sino también controlar el flujo de materiales haciendo que su visualización y control sea más sencillo. Este enfoque permite mejorar e incrementar la calidad en las operaciones del proceso obteniendo un mejor desempeño en el tiempo y proporcionando una calidad superior.

[1] Adrian Gomez Alvarez alumno de la carrera de Ingeniería Mecatrónica de la Universidad Autónoma de Ciudad Juárez al169701@alumnos.uacj.mx

[2] El Dr. Luis Carlos Méndez González es Profesor Investigador del departamento de Ingeniería Industrial y Manufactura en la Universidad Autónoma de Ciudad Juárez luis.mendez@uacj.mx

[3] El Dr. Luis Alberto Rodríguez Picón es Profesor Investigador del departamento de Ingeniería Industrial y Manufactura en la Universidad Autónoma de Ciudad Juárez luis.picon@uacj.mx

[4] El Dr. Iván Juan Carlos Pérez Olguín es Profesor Investigador del departamento de Ingeniería Industrial y Manufactura en la Universidad Autónoma de Ciudad Juárez ivan.perez@uacj.mx

EL método utilizado en este proyecto consta de los siguientes factores importantes. Comunicación Modbus: es un protocolo de comunicación que se utiliza ampliamente en la industria para conectar dispositivos de control y monitoreo. Modbus TCP es una variante del protocolo Modbus que utiliza Ethernet como medio de comunicación. En este artículo, exploramos la comunicación Modbus TCP por medio de Python y su aplicación en una simulación de un almacén automatizado en Factory I/O.

Python y Modbus TCP—Python es un lenguaje de programación popular utilizado en la industria para la automatización de procesos. La biblioteca pyModbusTCP.client proporciona una implementación de Modbus TCP en Python. Esta biblioteca es fácil de usar y se puede utilizar para leer y escribir registros Modbus en dispositivos remotos, para este caso es de vital importancia para generar la comunicación y establecer acciones que se ejecuten en tiempo y forma de manera correcta para completar las tareas que son asignadas en el código de programación.

```python
import sys,time,serial,threading
from PyQt6 import QtWidgets,uic, QtGui, QtCore
from PyQt6.QtWidgets import*
from PyQt6.QtGui import QImage
from time import sleep
from pyModbusTCP.client import *
c = ModbusClient(host='127.0.0.1', port=503)
path = 'C:\\Users\\adria\\OneDrive\\Documentos\\Factory IO\\My Scenes'
con = 0
pos = []

def recoleccioni():
    c.write_single_coil(2, 1)
    sleep(1)
    c.write_single_coil(4, 1)
    sleep(1)
    c.write_single_coil(2, 0)
    sleep(1)

def recolecciond():
    c.write_single_coil(3, 1)
    sleep(2)
    c.write_single_coil(4, 1)
    sleep(1)
    c.write_single_coil(3, 0)
    sleep(1)
    c.write_single_coil(4, 0)
    sleep(1)

def depositod():
    c.write_single_coil(4, 1)
    sleep(1)
    c.write_single_coil(3, 1)
    sleep(2)
    c.write_single_coil(4, 0)
    sleep(1)
    c.write_single_coil(3, 0)
    sleep(1)

def alm(i, j, con):
    pos.append(con)
    sleep(2)
    while c.read_discrete_inputs(0) == [True]:
        c.write_single_coil(7, 1)
        sleep(0.1)
    c.write_single_coil(7, 0)
    while c.read_discrete_inputs(1) == [True]:
        c.write_single_coil(0, 1)
        c.write_single_coil(1, 1)
    print('La caja número {} será ubicada en la fila {} celda {}'.format(str(con), str(i+1), str(j)))
    c.write_single_coil(1, 0)
    c.write_single_coil(0, 0)
    recoleccioni()
    c.write_single_register(0, con)
    sleep(1)
    while c.read_discrete_inputs(7) == [True] or c.read_discrete_inputs(8) == [True]:
        sleep(0.1)
    depositod()
    c.write_single_register(0,57)
    sleep(1)
    while c.read_discrete_inputs(3) == [False]:
        sleep(0.1)
```

Imagen 1: Librerías y funciones definidas utilizadas en el proyecto de un almacén automatizado.

En la Imagen 1 se observar cómo es que se importan de la librería de “Modbus” todos los paquetes que servirán para poner en marcha el almacén automatizado y también crear la conexión correcta, además de las líneas de código que mueven el robot del almacén para hacer el acomodo de las cajas y sumando a eso la interfaz gráfica para que el sistema este completo y sea funcional en la industria.

```python
def reco(i, j, con):

    print('La caja número {} ubicada en la fila {} celda {} será removida'.format(str(con), str(i+1), str(j)))
    c.write_single_register(0, con)
    sleep(1)
    while c.read_discrete_inputs(7) == [True] or c.read_discrete_inputs(8) == [True]:
        sleep(0.1)
    recolecciond()
    c.write_single_register(0,57)
    sleep(1)
    while c.read_discrete_inputs(3) == [False] or c.read_discrete_inputs(7) == [True] or c.read_discrete_inputs(8) == [True]:
        sleep(0.1)
    depositod()
    while c.read_discrete_inputs(7) == [True] or c.read_discrete_inputs(8) == [True]:
        sleep(0.1)
    c.write_single_coil(5, 1)
    c.write_single_coil(6, 1)
    while c.read_discrete_inputs(6) == [True]:
        sleep(0.1)
    sleep(2)
    c.write_single_coil(5, 0)
    c.write_single_coil(6, 0)
```

Imagen 2: While utilizados para hacer la entrega y recolección de cajas en el almacén.

En la Imagen 2 se muestran líneas de código donde se definen dos funciones, "alm" y "reco", que realizan una serie de tareas en el sistema automatizado. Ambas funciones reciben tres parámetros: "i", "j" y "con", que se utilizan para ubicar una caja en una fila y celda específicas. En la función "alm", se agrega un valor al final de una lista llamada "pos" y luego se espera por dos segundos. Después, se entra en un bucle "while" que se ejecutará siempre y cuando un sensor específico del sistema esté activo. Dentro del bucle, se escribe un valor en una bobina de control, se espera un breve tiempo y luego se escribe otro valor en una

bobina diferente. Después de salir del bucle, se imprime un mensaje que indica la ubicación de la caja. En la función "reco", se imprime un mensaje que indica que se va a retirar una caja de una ubicación específica. Luego, se realiza una serie de tareas similares a las de la función "alm", donde se espera a que se cumplan ciertas condiciones y realizar acciones adicionales. Finalmente, se activan dos bobinas de control durante un breve tiempo antes de desactivarlas de nuevo.

```
class control(QMainWindow):

    def __init__(self,parent=None):
        super(control,self). __init__(parent)
        uic.loadUi(path + '\\Interfaz.ui',self)
        self.manual.clicked.connect(self.cambiarman)
        self.automatico.clicked.connect(self.cambiaraut)
        c.open()

    def cambiaraut(self):
        self.hide()
        segunda_ventana = automatico(self)
        segunda_ventana.show()

    def cambiarman(self):
        self.hide()
        segunda_ventana = manual(self)
        segunda_ventana.show()


class automatico(QMainWindow):
    def __init__(self,parent=None):
        super(automatico,self). __init__(parent)
        uic.loadUi(path + '\\automatico.ui',self)
        self.regresar.clicked.connect(self.primer_pantalla)
        self.almacenar.clicked.connect(self.com)
        self.recolectar.clicked.connect(self.rec)
```

```
def com(self):
    try:
        total = int(self.total.toPlainText()) + 1
        con = 1
        if 1<=total-1<=56:
            for i in range (0, 6, 1):
                for j in range (1, 10, 1):
                    if con == total:
                        break
                    op = envio(i,j,con)
                    if op == 1:
                        QMessageBox.critical(self, 'Error', 'Fallo en la conexión con el módulo. Vuelva a intentarlo.', QMessageBox.StandardButton.Ok)
                        break
                    elif op == 2:
                        total += 1
                        con += 1
                    else:
                        con += 1
                if op == 1:
                    break
        else:
            QMessageBox.critical(self, 'Error', 'El Valor debe de encontrarse dentro del rango 1 a 56.', QMessageBox.StandardButton.Ok)

    except:
        QMessageBox.critical(self, 'Fallo en la operación', 'Verifique que los datos ingresados sean correctos.', QMessageBox.StandardButton.Ok)
```

Imagen 3: Clases definidas para la interfaz gráfica.

En la Imagen 3 se muestran las líneas de código para la clase de control la cual es la ventana principal que se carga cuando se inicia la aplicación. Esta clase tiene dos botones, manual y automático, y al hacer clic en ellos, se cambia a la segunda ventana, que es la clase manual o la clase automático. La clase automático es la ventana que se abre al hacer clic en el botón automático de la clase control. Esta ventana tiene tres botones, regresar, almacenar y recolectar. El botón almacenar llama al método “com”, que toma el valor ingresado en el campo total y recorre un bucle que envía comandos al módulo de almacenamiento y recuperación de objetos para almacenar los objetos en su ubicación correspondiente. El botón recolectar llama al método “rec”, que realiza una tarea similar para recuperar objetos.

Por otro lado, otro de los softwares utilizados es Factory I/O el cual es un software de simulación de automatización industrial que se utiliza para diseñar y probar sistemas de control en un entorno virtual. En el almacén automatizado utilizado como ejemplo, se realizó una comunicación entre el programa Visual Studio Code con lenguaje en Python que utiliza Modbus TCP para controlar el sistema. El almacén consta de conveyor’s, un rack para materiales de trabajo y un robot que se utiliza para transferir cajas de una ubicación actual del conveyor a una ubicación fija dentro del rack de almacenaje lo cual ayuda a facilitar la movilidad del material dentro del área de trabajo.

A continuación, en la Imagen 4 se muestra el entorno de simulación utilizado para realizar las pruebas de control del almacén.

Imagen 4: Simulación utilizada en Factory I/O

Se utiliza el protocolo Modbus TCP para controlar la posición del conveyor, sensores, actuadores, robot y monitorear la ubicación del material, de la siguiente forma es como se obtienen de manera fácil las ubicaciones y descripciones de los actuadores y sensores(Imagen 5), como mención adicional es importante reconocer que los actuadores dentro del software de Factory I/O para

este caso son llamados en Python como Coil's y los sensores son llamados como Inputs, además se puede observar en la esquina superior izquierda (Imagen 5) se señala que la conexión de Modbus TCP con Python es correcta con la palomita en verde, es como se verifica que los dos softwares están funcionando de manera correcta.

Imagen 5: Driver de Factory I/O

Como primer paso, se configuro Factory I/O para utilizar Modbus TCP. Se abre Factory I/O y selecciona "Opciones" en el menú principal. Se selecciono "Protocolos de comunicación" y luego "Agregar". Seleccione "Modbus TCP" y complete la información necesaria, incluida la dirección IP que para este caso la dirección es "127.0.0.1" y el número de puerto del dispositivo que se utilizó para la comunicación que de igual forma para este caso en particular es el puerto "503". Para controlar el robot utilizando Modbus TCP, se creó una instancia del objeto ModbusClient en el código Python y se conectó al servidor de Factory I/O.

Para utilizar pyModbusTCP, primero se debe instalar la biblioteca en el sistema. Esto se puede llevar a cabo utilizando pip install "pyModbusTCP", pip es un administrador de paquetes para Python. Se ejecuto este comando en la línea de comandos para instalar pyModbusTCP y de esta forma se pueden utilizar todas las librerías de pyModbusTCP para el desarrollo de la programación además de conocimientos avanzados que se necesitan para realizar las líneas de código.

A continuación se ve como se utilizó el software de Designer para elaborar una interfaz gráfica con diseño personalizado para dar forma y estructura más profesional y de manera fácil por así llamarlo funcional para el usuario y de esta forma cuente con un acceso fácil para ser utilizado, empezamos con la estructura de la pantalla principal la cual menciona como se observar dos modos de funcionamiento los cuales son "Modo manual" que permite acomodar en el almacén automático las cajas en el rack a como el usuario lo desee claro que como en esta simulación se cuenta con limitantes como lo es el espacio limitado de 56 espacios el usuario contara con 56 opciones disponibles para elegir a su disposición la que más le convenga y la que sea de mayor y fácil acceso, si, el usuario elige la fila 4. columna 3, entonces la caja ira y se ubicara en esta dirección de forma que si el usuario quiere volver a posicionar una caja en esta misma ubicaciones no podrá ya que está ocupada y la interfaz gráfica le arrojara una ventana emergente la cual le advertirá que la ubicación actual se encuentra ocupada por el momento, de esta forma el usuario podrá seleccionar otra ubicación para la caja y así tener una destino correcto, así de fácil como el usuario introduce cajas al rack de esta misma forma el usuario puede solicitar que se retiren las cajas del mismo y con las mismas instrucciones antes mencionadas las cuales es introducir la ubicación de la caja especificando columna y fila y de esta forma el robot buscara la dirección e ira a retirar las caja del rack para disponer el material a producción.

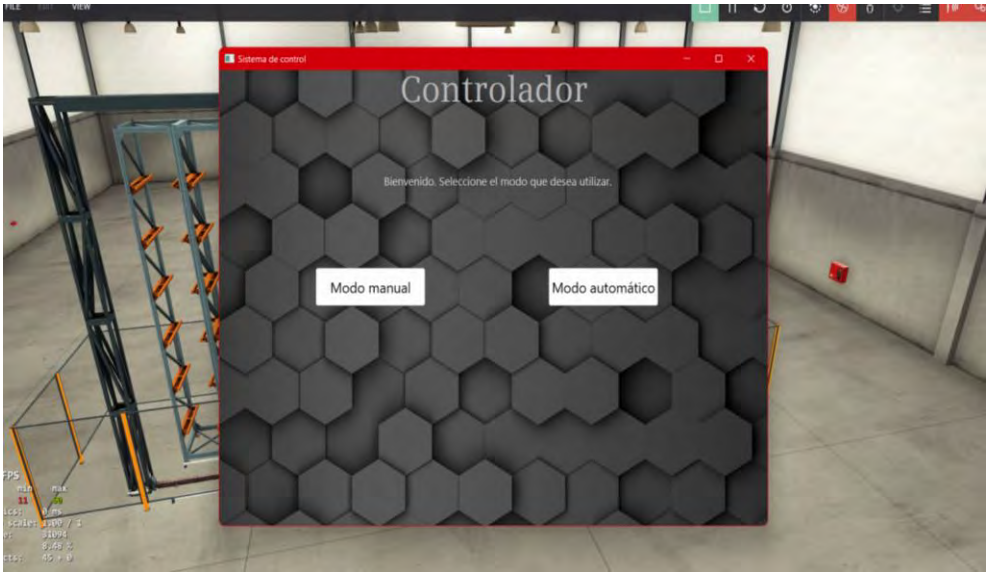

Imagen 6: Interfaz gráfica.

Otro de los métodos agregados en esta interfaz gráfica con programación es el "Modo automático" la cual es una forma más sencilla de acomodar las cajas en el rack, En la Imagen 7 como se observa a continuación es más fácil para el usuario introducir la cantidad total de cajas a almacenar y al momento de mandar esta instrucción al programa de simulación, este en automático empezara a enviar las cajas de manera ordenada al rack, si uno de los espacios está ocupado, este sistema se saltara dicho espacio y seguirá con el ordenamiento de las cajas de manera automática sin interrupciones y sin problemas.

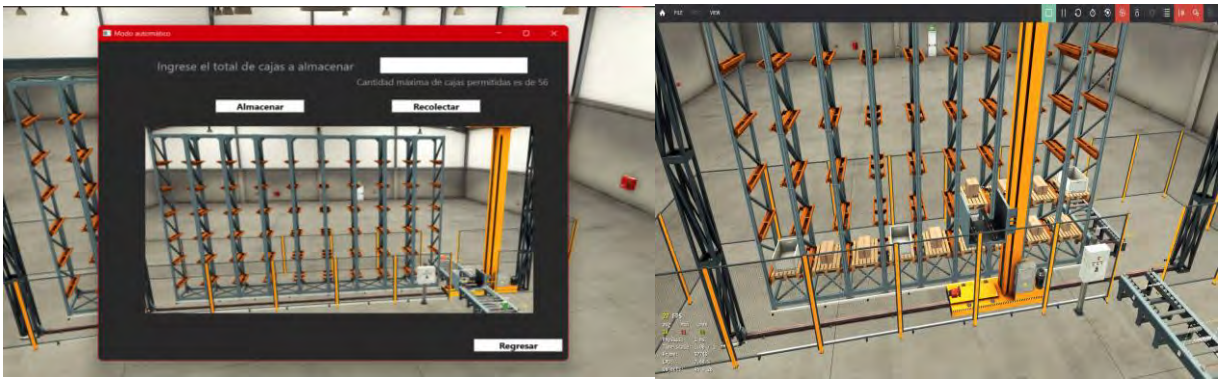

Imagen 7: Uso de la interfaz grafica y almacen en funcion.

Es importante mencionar que esta programación fue diseñada para cambiar de modos de uso de manera actual, lo que significa que, si el usuario desea cambiar de Modo automático a Modo manual en el momento, este puede hacerlo sin complicaciones y de esta forma ahorrar tiempo, esfuerzo y genera mayor productividad y beneficios para la empresa. Lo cual hoy en día es una de las mayores demandas dentro de las industrias, ya que estas buscan una productividad eficiente y segura que no genere algún contra tiempo en el día a día para lograr los objetivos y el almacén es una de las causas raíz de las cuales surgen problemas de tiempo y orden para que estas metas se cumplan.

Resultados

Como resultados obtenidos en el proyecto, se puede observar que la comunicación vía Modbus TCP en Python con el software de Factory I/O haciendo uso de Interfaz gráfica funciona de manera correcta por lo que el almacén automatizado funciona para la tarea que fue diseñada. La aplicación tiene dos modos de operación: manual y automático. En el modo automático, el usuario puede indicar la cantidad de cajas que se deben almacenar. La aplicación asigna automáticamente la ubicación de las cajas en la matriz de almacenamiento y luego las recolecta en el mismo orden. En el modo manual, el usuario puede seleccionar la ubicación de la caja que desea almacenar o recuperar. Con estos comandos asignados al almacén automático, se obtienen mayores resultados de productividad y ahorros de tiempos en el surtido de materiales para una empresa.

Conclusiones

Se puede concluir por los resultados que fueron obtenidos que el almacén automatizado funciona de manera correcta y demuestra que es una forma más fácil y segura para realizar este tipo de procesos dentro de una industria que ocupe este tipo de métodos de almacenaje lo cual para ser ciertos hoy en día la mayoría de las empresas cuenta o necesita contar con este tipo de almacenes para facilitar los movimientos y aumentar la seguridad de sus empleados y de sus productos, claro está que aun que este tipo de procesos pueden llegar a ser muy costosos para la empresa no es imposible en un futuro el adquirir estos sistemas ya que es un precio que lo vale y en el futuro una práctica que tendrá mucha competencia, por lo que esto facilitara aún mas el poder adquirir un sistema de almacenaje automatizado.

Referencias

K. Mehta, R. Joshi, H. M. Jadav, S. V. Kulkarni, B. H. Soni and A. Mali, "Notice of Removal: Integration of MODBUS/TCP master monitoring and control system using python for high power RF system," 2015 International Conference on Electrical, Electronics, Signals, Communication and Optimization (EESCO), Visakhapatnam, India, 2015.

S. Tamboli, M. Rawale, R. Thoraiet and S. Agashe, "Implementation of Modbus RTU and Modbus TCP communication using Siemens S7-1200 PLC for batch process," 2015 International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM), Avadi, India, 2015.

J. Owusu, A. Afful, V. Y. Kai Loung, P. Mills-Robertson and I. Abdul-Rashid, "Time Synchronization of Medium Voltage Substation IEDs Using Modbus and Python," 2021 IEEE 11th IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE), Penang, Malaysia, 2021.

T. M. Münsberg, L. Hvam, S. A. Lundsteen, M. Støjfer-Hønberg, M. Csik and L. Tsintzou, "Four Initiatives to Standardize Warehouses to Increase Digitalization and Automation," 2022 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), Kuala Lumpur, Malaysia, 2022.

V. D. Hristov, D. V. Slavov, I. S. Damyanov and G. D. Mladenov, "Machine Learning for Automation of Warehouse Activities," 2022 8th International Conference on Energy Efficiency and Agricultural Engineering (EE&AE), Ruse, Bulgaria, 2022.

V. D. Hristov, D. N. Saliev and D. V. Slavov, "Artificial Intelligence Systems for Warehouses Stocks Control," 2022 8th International Conference on Energy Efficiency and Agricultural Engineering (EE&AE), Ruse, Bulgaria, 2022.