İZMİR KATİP ÇELEBİ UNİVERSITY

FACULTY OF ENGINEERING AND ARCHITECTURE

DEPARTMENT OF MECHATRONICS ENGINEERING

Smart Tool Cabinet

MEE401 – GRADUATION PROJECT I

FINAL REPORT

Mehmet Berke PARLAT 160412014

Hasan ÜNLÜ 180412035

Metehan SARIOĞLAN 190412063

IZMIR-2024

Abstract

The Smart Tool Cabinet project presents an innovative solution utilizing a C# Windows Forms application integrated with image processing techniques to revolutionize tool management and status control within industrial and workshop settings. This multifaceted system aims to enhance efficiency, security, and accessibility in handling diverse tool inventories. The core of this project involves a user-centric C# Windows Forms application that acts as the central interface for administrators and technicians. The application streamlines tool management through a comprehensive database that catalogs tool specifications, maintenance history, and current locations within the cabinet. Leveraging intuitive design principles, the interface prioritizes ease of navigation and user interaction. Moreover, the implementation incorporates cutting-edge image processing capabilities to monitor tool status within the cabinet. Utilizing cameras or optical sensors, the system captures real-time images of tool compartments, employing image recognition algorithms to detect tool presence, absence, or misplacement. Apart from this, the camera used on the smart tool cabinet checks the condition of worn and deformed machining tools thanks to image processing. It warns that processing tools that are too deformed cannot be used and orders new processing tools. This innovative approach enables automated tracking and alerts for discrepancies in the tool inventory, ensuring swift and accurate status control.

Table of content

Table of figures

Abbreviations

CCD	: Charge Coupled Device
CMOS	: Complimentary Metal Oxide Semiconductor
FPS	: Frame Per Second
TCM	: Tool Condition Monitoring

1. INTRODUCTION

The Smart Tool Cabinet project is a sophisticated system designed for industries and workshops to effectively manage and organize their tools. Developed as a C# Windows Forms application, it focuses on improving the use of tools, reducing the chances of misplacing them, and enhancing the workflow for technicians and engineers. The software provides an intuitive interface for administrators to monitor tool inventory, classify different tools, and keep track of how they're used in real-time. It employs RFID or barcode scanning technologies, which makes identifying and assigning tools to their specific storage spots both fast and precise.

Image processing is the use of computational algorithms to perform operations on images to extract useful information or improve the visual quality of the image. It involves processing digital images using mathematical operations such as convolution and filtering. Image processing techniques are often used in fields such as computer vision, medical imaging and military surveillance. Some common applications of image processing include image enhancement, image restoration, image segmentation, and object recognition. In our project, we will use image processing for quality control of the materials in the smart tool cabinet.

Tool wear is an unpreventable and irreversible process that can lead to nonconforming geometry, poor surface finish, and even catastrophic failures like tool breakage. This results in costly downtime and damaged components. Thus, developing a tool condition monitoring (TCM) system is quite useful for proactive condition-based maintenance scheduling. There has been extensive research on TCM in tool breakage based on indirect monitoring of online process data such as cutting force, acoustic emission, vibration, and hydraulic pressure. However, few attempts have been made to directly measure the tool wear. For many manufacturers, however, the wear condition of tool wear is examined based merely on expert opinion by classifying the wear conditions into multiple levels or stages. For example, based on the technician's experience, the wear condition can be classified as low, medium, and high levels. However, tool wear is a continuously accumulative process which is responsible for cutting performance degradation. The qualitative nature of current industry methods results in huge information loss and heavily relies on operators' process knowledge and experience, which could lead to high variability between tool inspectors.

The objective of this paper is to develop a reliable and automatic wear characterization system that can be used to characterize the condition of tool wear without requiring an expert's knowledge. The developed system is designed to integrate image acquisition and image processing methods. A measurement system analysis was performed to quantify the repeatability and reproducibility of the system based on the tool wear. This analysis considered different levels of tool wear.

Developing a reliable tool wear quantification system can help to reduce quality cost due to poor tool condition, perform cost effective condition-based maintenance

scheduling, and avoid downtime due to catastrophic accidents during the machining operations. As the quality of the cutting tool is directly related to the quality of the product, the level of tool wear should be kept under control during machining operations The recognition of the general conditions of a cutting tool has a major role in the optimization of machining processes, since the accurate prediction of the exact moment for tool change results in many cases in an effective economy: a longer cutting tool life can be achieved, tolerances can be under control and rejection of pieces by deterioration of the tool conditions can be prevented.

In order to control the evolution of tool wear, the interface chosen between the actual working procedure and the computer was the digital image of the cutting tool detected by a camera. In this paper, cutting tool images of standard size and pixel density determined during machining tests were produced by processing image files through matrices. These standard images of tool wear were used for intelligent processing of image data aimed at automatic and real-time tool wear evaluation.

2. LITERATURE REVIEW

An extensive study and analysis were carried out, centering on topics like tool management systems, the design of user interfaces, and the basics of C# programming. This thorough examination yielded valuable knowledge about the most effective methods used in the industry. This knowledge greatly influenced the design and operational features of the application.

There are various papers that have described the monitoring of tool wear and the life of turning tools. Rehorn et al offered a review of sensors and signal processing methodologies used for tool condition monitoring in turning, drilling and milling. Cakir and Isik used the tool life and condition monitoring system to study variations in the cutting forces when turning AISI 1050 steel with coated and uncoated tungsten carbide ISO P25 tools.

Kim et al. has been developed a magnetic jig for fixing the camera and lighting system to accomplish the objective of on machine tool measurement of flank wear for a 4-fluted end mill. They compared the signal to noise ratio of measurements using microscope and CCD camera incorporating with a novel jig. They inserted the fibre optic guided lighting system into the lens for further improvement. However, this work is more prone to the measuring system instead of image processing technique.

Kerr et al. utilized four different texture analysis techniques namely histogram based processing, grey level co-occurrence technique, frequency domain based technique and fractal method to analyze the texture of the worn region of turning and milling insert. They obtained the best result by frequency domain or Fourier spectrum analysis techniques because this technique is position and illumination invariant. However, they have captured the tool tip portion of turning insert instead of the flank face portion which is not a standard practice. Jackson et al. has been proposed a novel technique for accurate edge detection algorithm utilizing neural network technique for tool wear detection. They have utilized the scanning electron microscopic images of the flank wear images of a 4-fluted high speed steel milling cutter. However, they did not do any online monitoring using CCD or CMOS camera.

Atli et al. developed a new measure namely DEFROL (deviation from linearity) to classify between sharp and dull drilling tool from their images. However, the emphasis has been done on the change of point angle and linearity deviation of the cutting edges due to the wearing effect, but no study has been taken care regarding the flank wear in this technique. So this technique was not suitable to measure the flank wear which was used to define the tool life in standard practice (ISO 3685). Makki et al. did a real time capturing of drill bit image at the time of 100 rpm rotation. Then they processed those captured images by edge detection and accurate segmentation technique to find out the tool wear (only the deviation of the lip portion) and tool run-out in the image plane. However, the measurement of flank wear and tool run out perpendicular to the image plane cannot be possible by their technique.

3. METHODOLOGY

Technical Learning:

Time and effort were invested in learning the basics of C# programming, a crucial skill needed to build the application. This educational phase included getting to grips with the programming language's syntax, its various data structures, and essential principles, setting a solid foundation for the project's coding stage.

Interface Design:

An elaborate flowchart was designed to map out the different stages of the application's user interface. It detailed how users would interact with the application, the routes they would take to navigate through it, and the primary features it would offer. Moreover, an initial design for the welcome screen was created, focusing on engaging users right from the start and guiding them on how to proceed.

How Does Image Processing Work?

The basic steps in image processing include the following:

1. Image acquisition: Imaging involves capturing an image using a camera or other device.
2. Pre-processing: Pre-processing; It involves preparing the image for processing, which may include steps such as noise reduction, contrast enhancement, and color correction.
3. Feature extraction: Feature extraction involves identifying and extracting important features in the image, such as edges, corners, and textures.
4. Analysis: Analysis involves using algorithms to analyze the image and extract meaningful information from it.
5. Visualization: Visualization involves presenting the results of image processing in a way that is easy for humans to understand, such as visual images or graphs.
6. Output: Output involves storing or transmitting the results of image processing for further use or analysis.

Image Processing Features

There are many features that can be used in image processing. Some of these include:

1. Color conversion: Color conversion involves converting images from one color space to another, for example, RGB to grayscale or RGB to CMYK.
2. Filtering: Filtering involves applying a filter to an image to soften, sharpen, or highlight certain features.
3. Edge detection: Edge detection involves identifying edges in an image and marking them with lines or curves.
4. Morphological processes: Morphological operations involve applying transformations such as expansion, erosion, opening and closing to shapes in an image.

5. Partitioning: Segmentation involves dividing an image into separate regions or sections, each corresponding to a different object or background.
6. Feature extraction: Feature extraction involves identifying and extracting important features of an image, such as lines, corners, or textures.
7. Image restoration: Image restoration involves removing noise or blurriness in an image or filling in missing pixels.
8. Object recognition: Object recognition involves identifying and labeling objects in an image.
9. Image recording: Image registration involves aligning or recording multiple images of the same scene taken from different perspectives or at different times.
10. Image synthesis: Image synthesis involves creating new images from scratch or from existing images.

EQUIPMENT

In this project, we utilized Arduino microcontrollers and pressure sensors as key components. The Arduino, known for its versatility and ease of programming, serves as the central control unit. It interfaces with multiple pressure sensors, which are crucial for monitoring and recording pressure changes in our experiments. These sensors, selected for their precision and responsiveness, provide real-time data that is essential for accurate analysis. The integration of Arduino with these sensors demonstrates a seamless blend of reliable hardware and sophisticated programming, vital for the success of our project.

Industrial image processing systems Camera Control Unit (Industrial camera and Lens), Lighting Unit; It consists of image processing software, computer and components. Before commissioning industrial image processing applications, hardware components must be verified to ensure that the system can perform 100% visual control.

Selection of equipment to be used for the camera control system:

- Should you use an Area Scan camera or a Line Scan camera?
- Should you use a Global Shutter camera or a Rolling Shutter camera?
- Frame Rate value of the camera is min. value?
- Sensor type and layers of the camera?
- What should be the Interface (Interface and Connection) of the camera?
- What will the Lighting System be like?
- What are the characteristics of the lens to be used (type, angle of view, depth, focal length, field of view, deformation)?

Some of these items need to be answered to ensure 100% visual control of the system, while others need to be answered to reduce costs. As an example, let's say the part is manually fed into a fixture and rests on the fixture for 2 seconds. In this case, Rolling Shutter and low FPS camera will be able to perform this control. Using a Global Shutter high FPS camera for this system would increase the cost slightly.

Rolling Shutter and a low FPS camera will be sufficient for us for both cost and image processing through photography.

CAMERA

Logitech C270 HD 720P camera can be used for image processing due to its low cost and its focusing feature. Providing images up to 30 FPS, as well as its focusing feature and more affordable cost compared to similar cameras, make the camera one of the first choices.

Light Selection in Image Processing

In image processing applications, if the amount of light falling on the object is not sufficient, it will be difficult to achieve the desired results. Therefore, in order to get a quality image, objects must be adequately and appropriately illuminated.

How should lighting be chosen?

- Which feature of the object we want to be visible
- What material the object is made of (reflects or absorbs the light falling on it)
- Geometric structure of the object
- Color of the object

These are the factors that affect the choice of lighting.

For example, if the diameter of the bolt is to be checked, illumination from below with backlight for outer diameter measurement brings us closer to the correct result, but prevents us from measuring the inner diameter. If you want to measure the inner diameter, a different lighting should be preferred. (Mentioned in the 0° illumination heading.)

Since we will be measuring the outer diameter rather than the inner diameter in our project, we will prefer back light.

Back light:

If we want the main image of the object to become clear, we can highlight the edges and corners of the object by illuminating it from underneath. For example, the use of backlight is the most ideal lighting method in applications such as shape control of the object, transmission of position information flowing through the distribution line to the robot, presence/absence of holes on the parts or quantity information. Since this illumination will highlight only the external shape of the object, it will prevent us from seeing details such as patterns, roughness, and color on the object.

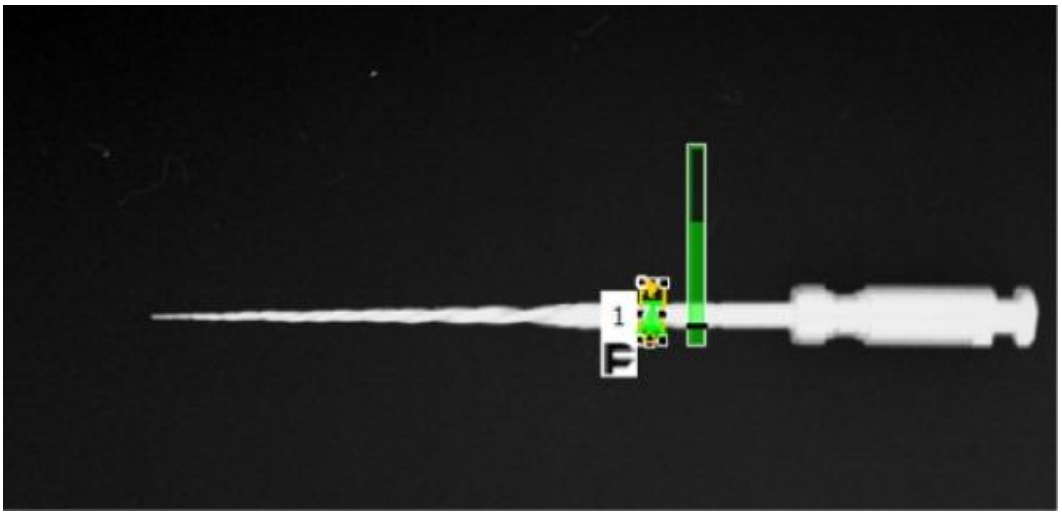

Figure 1: Camera view of the product illuminated with back light for needle tip diameter measurement.

4. EXPERIMENTAL RESULTS AND ANALYSIS

SOFTWARE

In the software development aspect of our project, we chose C# and Windows Forms Application as our foundational technologies. C#, with its strong object-oriented features, provided us with a powerful and versatile programming environment. It enabled us to develop complex functionalities while maintaining code clarity and ease of maintenance. The use of Windows Forms, a graphical user interface toolkit, allowed us to build a visually appealing and user-centric interface. This framework made it possible to create interactive elements and controls that enhance user interaction, ensuring a more engaging experience. The synergy between C# and Windows Forms was instrumental in facilitating a streamlined development process. We leveraged C#'s integration capabilities with various libraries and APIs to extend the functionality of our application. This approach not only enhanced the application's performance but also allowed for the creation of a more dynamic and responsive interface, catering to the specific needs of our project. Overall, the combination of C# and Windows Forms Application played a pivotal role in achieving a high-quality, robust, and user-friendly software solution.

In enhancing our software solution, the interface was meticulously designed to integrate an RFID reader, user profiles, visual representations of tools, and real-time data from Arduino and pressure sensors. The RFID technology enables efficient tracking and management of tools, seamlessly linking them to the user profiles within the system. By incorporating images of tools, users can easily identify and select the needed equipment. Furthermore, the integration of Arduino and pressure sensors plays a crucial role in verifying the correct selection of tools, as it provides immediate feedback on tool handling and usage. This multifaceted interface design ensures a comprehensive, user-friendly, and technologically advanced experience for effective tool management.

OpenCV is an open source image processing library developed by Intel in 1999. OpenCV, which has C, C++, Python and Java interfaces, can run on different platforms such as Windows, Linux, Mac OS, iOS and Android. Thanks to the BSD (Berkeley Software Distribution) license, it can be used free of charge in all kinds of projects. OpenCV is designed to increase computational efficiency and for real-time applications.

OpenCV library consists of 4 main components named CV, MLL, HighGUI and CXCore.

- CV component: Includes image processing and high-level image processing algorithms.
- MLL component: Contains machine learning commands including many statistical classifiers and clustering tools.
- HighGUI component: Contains I/O operations and functions for storing and loading video and images.

- CXCore component: Contains sections such as basic data structures and content, drawing images.

In order to perform the necessary operations in our study, we chose the OpenCVSharp library, one of OpenCV's wrappers developed for C#.

Opencvsharp

It is another library written for the .Net framework. It aims to develop image processing software for .Net languages. Unlike EmguCV, there is no extra license cost since it is open source. It can be used for projects developed with .Net framework 2.0 and above, and with mono support, applications can also be developed for platforms such as Linux and MacOS. It is developed by Shimat and is a library that remains up to date.

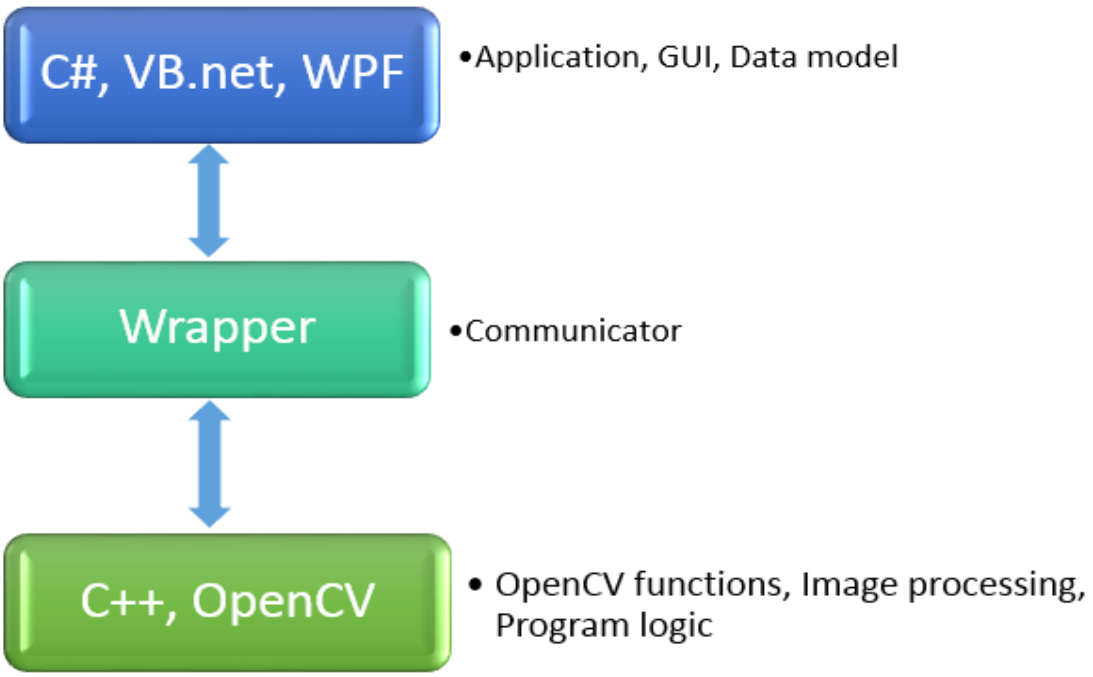

Figure 2: Working logic of wrappers

Setting up a C# Project with OpenCV

Here's how to create a new C# project:

1. **Open Visual Studio**: Launch Visual Studio and select "Create a new project" from the welcome screen.
2. **Choose Project Type**: In the "Create a new project" dialog, search for "C# Console App (.NET Core)" or "C# Console App (.NET Framework)" based on your preference. Select it and click "Next."
3. **Project Details**: Provide a name for your project and choose a location to save it. Click "Create" to create the project.

Your new C# project should now be open in Visual Studio.

Adding OpenCV Dependencies

Now that your project is set up, the next step is to add OpenCV dependencies to it. These dependencies are necessary to call OpenCV functions within your C# code.

1. **Download OpenCVSharp**: OpenCVSharp is a wrapper library for OpenCV that allows you to use OpenCV via C#. To download it, go to the NuGet Package Manager (Right-click on your project in the Solution Explorer -> Manage NuGet Packages).
2. **Search and Install**: In the NuGet Package Manager, search for "OpenCVSharp" and install the latest stable version. This package should include most of the OpenCV libraries you'll need.
3. **Add DLL Reference**: This is an optional step if you installed OpenCV manually and want to point to the specific DLL files.
    - Right-click on your project in the Solution Explorer and click "Add" -> "Reference."
    - Navigate to the "Browse" tab and locate the OpenCV "bin" folder (where the DLLs are stored).
    - Select the necessary DLL files and click "Add."

```
public void SetColorFilter(ColorFilterTypes colorFilterType)
{
    Bitmap temp = (Bitmap)_currentBitmap;
    Bitmap bmap = (Bitmap)temp.Clone();
    Color c;
    for (int i = 0; i < bmap.Width; i++)
    {
    for (int j = 0; j < bmap.Height; j++)
    {
                c = bmap.GetPixel(i, j);
                 int nPixelR = 0;
                 int nPixelG = 0;
                 int nPixelB = 0;
                 if (colorFilterType == ColorFilterTypes.Red)
                 {
                         nPixelR = c.R;
                         nPixelG = c.G - 255;
                             nPixelB = c.B - 255;
                 }
                 else if (colorFilterType == ColorFilterTypes.Green)
                 {
                         nPixelR = c.R - 255;
                    nPixelG = c.G;
                           nPixelB = c.B - 255;
                 }
                 else if (colorFilterType == ColorFilterTypes.Blue)
                 {
                           nPixelR = c.R - 255;
                           nPixelG = c.G - 255;
                           nPixelB = c.B;
                 }
                 nPixelR = Math.Max(nPixelR, 0);
                 nPixelR = Math.Min(255, nPixelR);

                 nPixelG = Math.Max(nPixelG, 0);
                 nPixelG = Math.Min(255, nPixelG);

                 nPixelB = Math.Max(nPixelB, 0);
                 nPixelB = Math.Min(255, nPixelB);

                 bmap.SetPixel(i, j, Color.FromArgb((byte)nPixelR,
                   (byte)nPixelG, (byte)nPixelB));
            }
        }
        _currentBitmap = (Bitmap)bmap.Clone();
}
```

Figure 3: Color filter example with C#

görüntü alma.py > goruntualma.py > ...

```python
import cv2
import numpy as np

kamera=cv2.VideoCapture(0)

while True:
    ret,goruntu=kamera.read()

    cv2.imshow("webcam",goruntu)

    if cv2.waitKey(30) & 0xFF ==('r'):
        break

kamera.release()

cv2.destroyAllWindows()

```

Figure 4 Taking images from camera

resim.py > ...

```python
import cv2
import numpy as np

resim= cv2.imread("3mm-matkap2.jpg")

cv2.imshow("smart tool cabinet",resim)

cv2.waitKey(0)
cv2.destroyAllWindows()

```

Figure 5: Open image

```python
import aspose.words as aw

doc = aw.Document()
builder = aw.DocumentBuilder(doc)

shape = builder.insert_image("Input.jpg")
shape.image_data.save("Output.png")
```

Figure 6: Make the image png

```python
import cv2

resim=cv2.imread("3mm-matkap2.jpg")

print(resim)


cv2.waitKey(0)
cv2.destroyAllWindows()

```

Figure 7: Splitting the image into matrices

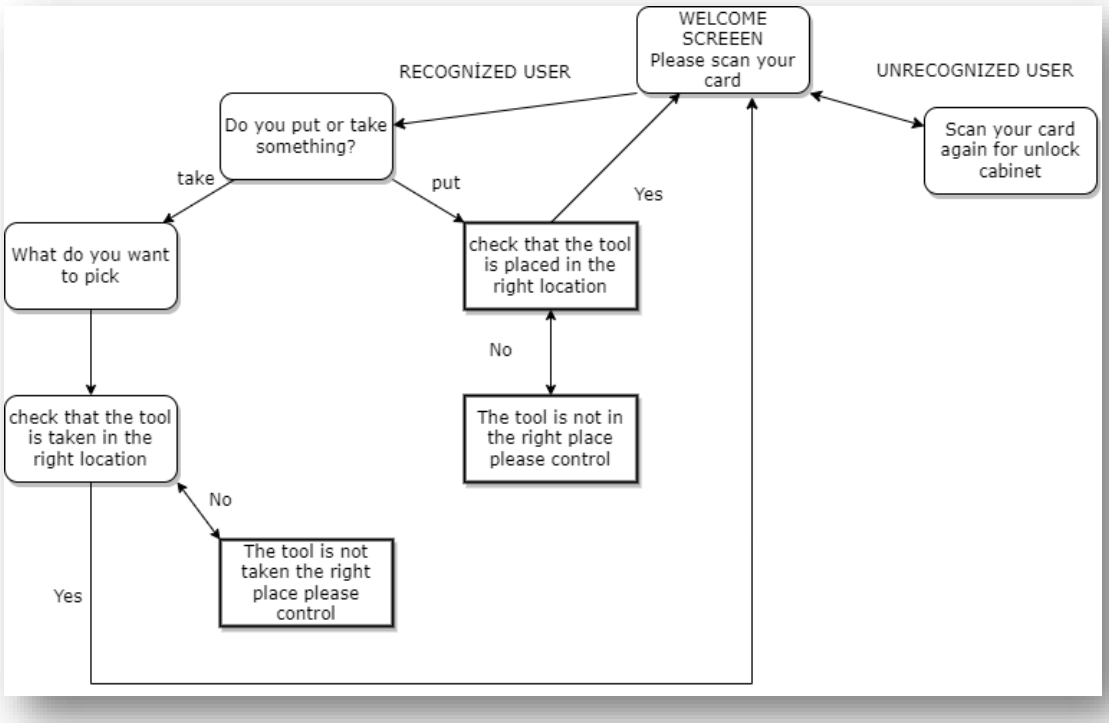

Figure 8: Flow chart for stages of the interface

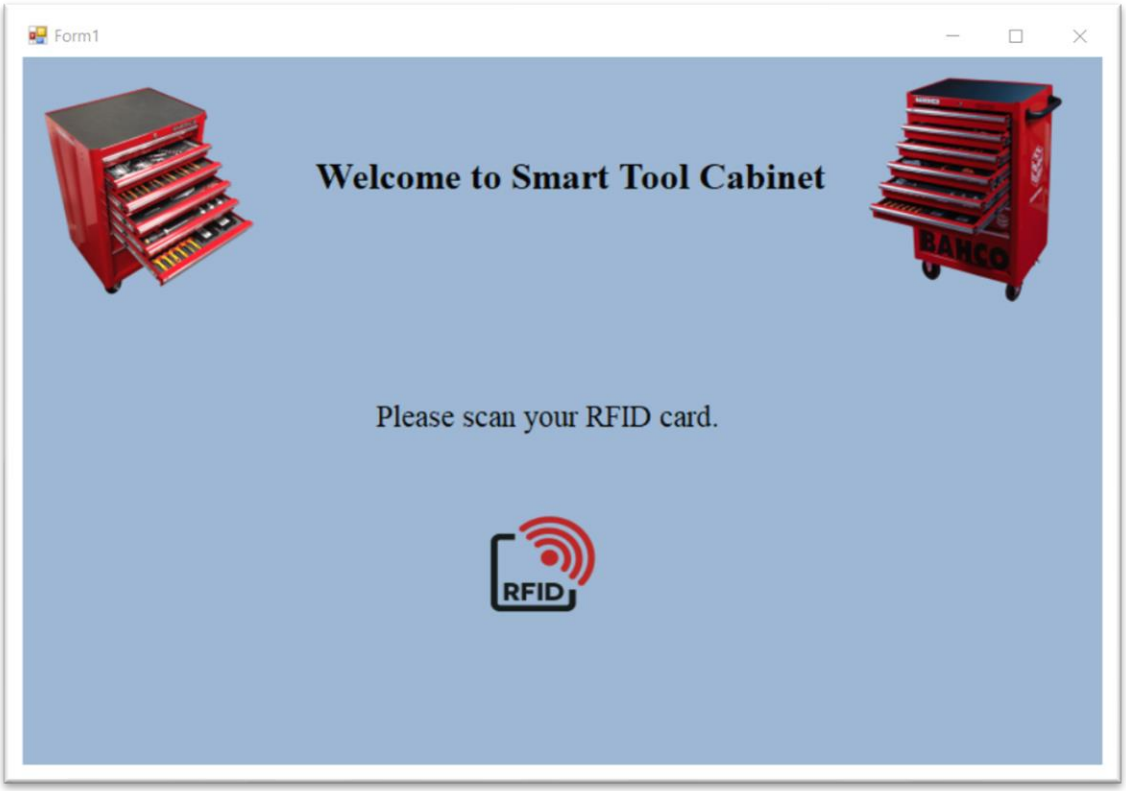

Figure 9: Welcome screen of interface

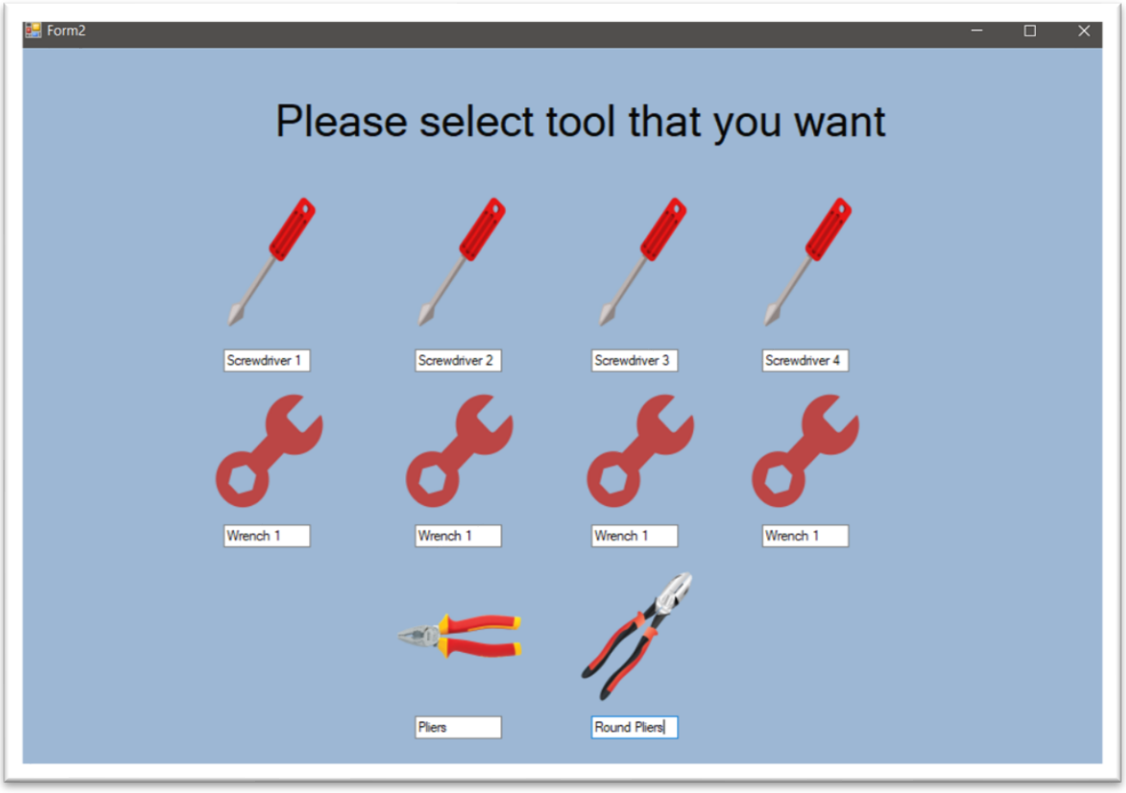

Figure 10: Tool select screen of interface

5. DISCUSSION

The integration of RFID and Arduino technologies with the C# application has proven to be highly effective. The RFID system's accuracy in tool tracking and the real-time feedback from pressure sensors ensure that tools are not only accounted for but also maintained properly. The user interface, crafted with Windows Forms, offers intuitive navigation and ease of use, significantly enhancing the user experience. The combination of these technologies has surpassed our initial expectations in creating an efficient tool management system.

Our project's success resonates with the findings in contemporary literature, particularly in the realms of automated inventory management and user-interface design. The novel application of image processing for tool condition assessment, as demonstrated in our project, provides a unique contribution to the field. This approach, scarcely covered in existing studies, presents a new perspective on leveraging visual data for enhanced tool management.

The implications of our project extend beyond the immediate industrial environment. Its success suggests potential applications in various sectors requiring precise tool management, from automotive to aerospace industries. The project's integration of various technologies paves the way for more automated, interconnected, and intelligent tool management systems, potentially revolutionizing how industries approach inventory and resource management.

The insights gained from this project contribute significantly to the evolving field of industrial tool management, highlighting the importance of technological integration in advancing operational efficiency and accuracy.

CONCLUSIONS AND RECOMMENDADIONS

The "Smart Tool Cabinet" project, encompassing RFID, Arduino, pressure sensor technologies, and sophisticated image processing within a C# Windows Forms application, represents a groundbreaking achievement in industrial tool management. This project transcends traditional tool tracking and maintenance methods, offering a comprehensive solution that significantly enhances operational efficiency. The integration of image processing technology is particularly noteworthy, as it provides an advanced level of precision in monitoring tool conditions and usage. This feature not only improves the accuracy of the tool management system but also adds a layer of intelligence to the inventory control process.

The project's successful implementation of these combined technologies has set a new benchmark in the field, illustrating the immense potential of technological synergy. The use of RFID and pressure sensors ensures meticulous tracking and maintenance of tools, while the addition of image processing allows for real-time condition assessment, contributing to a more proactive maintenance approach. Moreover, the C# Windows Forms application offers a user-friendly interface, making it accessible and easy to use for industrial personnel. This aspect is crucial in ensuring the widespread adoption and effectiveness of the system.

Furthermore, the project's implications extend beyond the immediate sphere of tool management. It serves as a model for similar applications in various industrial contexts, demonstrating how technological integration can revolutionize traditional practices. The project thus not only addresses current challenges in tool management but also opens up new possibilities for future innovations.

In conclusion, the "Smart Tool Cabinet" project is a testament to the transformative power of combining cutting-edge technologies in industrial applications. It marks a significant step forward in the field of tool management and sets the stage for future advancements that could further enhance efficiency, accuracy, and user experience in industrial operations. This project serves as a beacon, guiding the way towards a more technologically integrated and efficient future in industrial management.

FUTURE WORKS

1. Advanced User Interface Development: Further refinement and customization of the user interface to adapt to varying user preferences and requirements.

2. Integration with IoT Systems: Consider integrating the system with broader Internet of Things (IoT) infrastructure for enhanced connectivity and remote monitoring capabilities.

3.Machine Learning Implementation: Employ machine learning algorithms to predict tool maintenance needs and optimize inventory management.

4.Cross-Platform Deployment: Explore the development of a cross-platform application to increase accessibility across different operating systems and devices.

5.Sustainability Considerations: Investigate eco-friendly options in the manufacturing and operation of the tool management system.

6.User Training Programs: Develop comprehensive training modules for users to maximize the system's efficiency and effectiveness.

7. Long-Term Field Testing: Conduct extended field tests to evaluate the system's durability and performance over time in various industrial settings.

These recommendations aim to not only enhance the current capabilities of the "Smart Tool Cabinet" project but also to ensure its adaptability and sustainability in the ever-evolving industrial landscape.

REFERENCES

Loizou, J., Tian, W., Robertson, J., & Camelio, J. (2015). Automated wear characterization for broaching tools based on machine vision systems. *Journal of Manufacturing Systems*, *37*, 558-563.

D'Addona, D. M., & Teti, R. (2013). Image data processing via neural networks for tool wear prediction. *Procedia Cirp*, *12*, 252-257.

Mikołajczyk, T., Nowicki, K., Bustillo, A., & Pimenov, D. Y. (2018). Predicting tool life in turning operations using neural networks and image processing. *Mechanical systems and signal processing*, *104*, 503-513.

Dutta, S., Pal, S. K., Mukhopadhyay, S., & Sen, R. (2013). Application of digital image processing in tool condition monitoring: a review. CIRP J Manuf Sci Technol 6 (3): 212–232.

Bergs, T., Holst, C., Gupta, P., & Augspurger, T. (2020). Digital image processing with deep learning for automated cutting tool wear detection. *Procedia Manufacturing*, *48*, 947-958.