T.C.
BAHÇEŞEHİR UNIVERSITY

FACULTY OF ENGINEERING AND NATURAL SCIENCES

CAPSTONE PROJECT PROPOSAL

OBJECT RECOGNITION FOR SECURITY AUTOMATION

Albayrak Beyza – Electrical and Electronics Engineering

Anıl Ege – Electrical and Electronics Engineering

Ozkurt Ekin - Computer Engineering

Tutar Elif – Electrical and Electronics Engineering

Ultav Berk - Computer Engineering

Advisors:

Assoc. Prof Erkut Arıcan
Assoc. Prof Suzan Üreten

ISTANBUL, December 2023

STUDENT DECLARATION

By submitting this report, as partial fulfillment of the requirements of the Capstone course, the students promise on penalty of failure of the course that.

- they have given credit to and declared (by citation), any work that is not their own (e.g. parts of the report that is copied/pasted from the Internet, design or construction performed by another person, etc.).
- they have not received unpermitted aid for the project design, construction, report or presentation.
- they have not falsely assigned credit for work to another student in the group, and not take credit for work done by another student in the group.

ABSTRACT

OBJECT RECOGNITION FOR SECURITY AUTOMATION

The project aims to detect products captured by a camera and transfer images of the identified products to a computer environment with the establishment of a conveyor structure-based security automation system. The choice is made to utilize Python for our project's programming language. The images transferred to the computer environment will be processed using the image processing capabilities of the OpenCV library, a Python library. Our main goal is identifying any potentially hazardous objects with using a camera that is integrated into the system to detect materials placed on the moving conveyor belt. With enclosed sides and a top made of iron and wood materials, the conveyor system is positioned a little above the ground. For object detection, this structure offers a well-lit, organized setting. Together with integrated electrical components, the system has an RFID structure that provides users with a wireless control mechanism for improved security management.

The choice of Python is motivated by several factors. Firstly, the image processing component, which constitutes a significant portion of the project, can be more easily integrated into Python compared to Java. Secondly, Python enables faster prototyping and development, facilitating a more agile development process. Additionally, Python offers more versatile libraries for creating deep learning models. Lastly, Python is known for its strength in data analysis and processing, especially when dealing with large datasets.

In summary, after the image captured by the camera is transferred to the computer environment, image processing will be performed using pre-trained models in Python. The processed images will be classified based on whether the recognized objects pose security concerns.

Key Words: Python, OpenCV, Arduino, Convolutional Neural Networks, image processing, camera, Data Storage and Analysis, Deep Learning, Recurrent Neural Networks, Transfer Learning, Classification, Deep Features, Feature Learning, RGB-D Object Recognition, Conveyor belt, RFID(Radio Frequency Identification)

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

IEEE - The Institute of Electrical and Electronics Engineers

ONT - Object Recognition and Tracking

STA - Security and Threat Awareness

PEM - Power Efficiency Management

IMM - Integration and Modularization Methods

UIAN - User Interface and Notifications

CEK - Conditions and Environmental Constraints

AUS - Application Areas and Use Scenarios

ORS: Object Recognition System

IoT: Internet of Things

CNN: Convolutional Neural Network

PIR: Passive Infrared

API: Application Programming Interface

TLS: Transport Layer Security

OTA: Over-the-Air

GUI: Graphical User Interface

WLAN: Wireless Local Area Network

USB: Universal Serial Bus

Edge Computing: Edge Processing or Edge AI

RFID: Radio Frequency Identification

1. OVERVIEW

1.1. Identification of The Need

Modern security systems are compelled to be smarter and more effective in the face of evolving technology and increasing security needs. Traditional security systems often exhibit a passive and reactive structure, leading to delays in responding to immediate events. In this context, a security system with the capability for object detection and recognition in enclosed spaces or specific areas can offer a more proactive approach and the potential for pre-emptive intervention.

1.2. Definition of The Problem

The identified issue targeted by the project is the detection of products deemed as dangerous before entering secure areas. To address this concern, a camera system will be employed to detect these potentially hazardous products. The camera will capture images of the detected items and transfer them to a computer environment. Subsequently, the computer will process these images to determine whether the identified products pose a threat. In cases where a product is classified as dangerous, the system will issue an alert, prompting the implementation of a solution. Moreover, the security officer will be able to access a user interface using an RFID card to view the object that the system is currently alerting them about, helping to maintain constant control over security.

The principal aim of this project is to devise a comprehensive approach for the prompt and precise identification of hazardous materials by means of a camera system. The integration of image processing techniques on a computer platform will enable the system to make informed decisions regarding the safety status of the identified items. Consequently, the system will issue warnings for products deemed dangerous, contributing to the resolution of the identified security concern.

1.2.1. Functional Requirements

The Python based camera object recognition project necessitates seamless integration with real-time data from the camera, efficient image processing using OpenCV, and the implementation of pre-trained models for accurate object detection. The system must feature a user-friendly interface, configurable settings, and a robust alert mechanism for potential threats. Additionally, scalability, compatibility, security measures, comprehensive documentation, and rigorous testing procedures are pivotal for the project's success.

In essence, the project aims to develop a streamlined and adaptable solution for real-time object recognition, combining advanced image processing techniques with user-friendly features and robust security measures to enhance security automation.

1.2.2. Performance Requirements

Objects will be detected using a webcam which is connected to a computer via USB cable, but the image processing task will not be performed on the Arduino. Instead, object recognition will be carried out on the computer using code and image processing algorithms. The data from the webcam will be processed by the computer, and when an object is detected, the results will be sent back to the Arduino to control outputs such as LEDs, buzzer and LCD display. This way, while saving on Arduino's hardware resources, more complex image processing tasks can be successfully performed using the computer's more powerful processing capabilities.

1.2.3. Constraints

Our project is being developed through collaboration among a team of five members, consisting of three Electrical and Electronics Engineering students and two Computer Engineering students. We encounter specific constraints while pursuing our objective. These limitations significantly impact the project's success by directing the creation and application of our system. In this section we'll go into more detail about the main obstacles that have been found to our project's growth and successful conclusion.

Camera And Image Processing Performance:

The cameras and the image processing system's performance are crucial. The camera needs to identify items on the conveyor system quickly and correctly. This calls for the acquisition of high-resolution images and the analysis of those images in real time, necessitating reliable computer hardware and effective image processing techniques. Maintaining low latency and excellent object identification accuracy is critical to the dependability of the system.

RFID Integration and User Interface:

The RFID card reader integration is an additional essential element. The system and the RFID reader must work together seamlessly to ensure dependable operation and minimal latency between the activation of the user interface and the reading of the card. The RFID tags must be appropriately recognized by the system. Security staff must be considered in the design of the user interface, which should provide a simple and intuitive experience. When an RFID card is read, it should instantly display the name and timestamp of the most recent object that was flagged. This interface needs to be responsive, simple to use, and offer clear, succinct information to enable prompt decision-making.

Real-Time Processing and Hardware Limitations:

The system must be able to process and respond in real time. To ensure security and efficacy, every step of the process from image capture to analysis and alarm generation must be completed as quickly as possible. Fast data transfer rates and great processing power should be supported by the camera system and computer systems that are chosen. Hardware performance must be maximized while staying within the project budget and taking cost-effectiveness into account.

Team Collaboration and Reliability:

The project's success depends on good task distribution and teamwork. The task distribution in a team of five should take use of the specializations of each team member: three are electrical and electronics engineers and two are computer engineers. To guarantee good teamwork, communication, and overall project efficiency, roles and duties must be clearly defined.

Economical Constraints:

Economic considerations are a major determinant of the project's feasibility and extent, given the school's 3000 TL budget constraint. This budgetary restriction requires careful management of all costs associated with electronic components, including cameras, belts, Arduinos, and conveyor system parts.

1.3. Conceptual Solutions

The project aims to develop a security system that integrates object recognition using a camera with the capability to assess the safety or hazard status of identified objects. The system leverages Python for image processing and user interface development, combined with a modular conveyor belt setup for efficient object handling.

Using the OpenCV library, the system captures and processes images to detect and identify objects in real-time. The conveyor belt system, enclosed within a structure made of iron and wood, ensures a controlled environment for accurate detection, minimizing external interferences and providing optimal lighting. Analyzed images are then evaluated against predefined security criteria to determine if they pose any risk.

A user-friendly interface, created with Python's Tkinter library, displays the results, and provides clear visual feedback. If a hazardous object is detected, an alert system with a buzzer, LED, and LCD display is activated to immediately inform the user. Additionally, the system features RFID technology for enhanced security and wireless control over the conveyor belt, further improving the

management and safety of the system.

This approach ensures a comprehensive, efficient, and user-friendly security solution, combining advanced image processing, real-time hazard detection, responsive user alerts, and robust conveyor belt integration.

1.3.1. Literature Review

The Arduino Uno R3: This is a development board that runs on a microcontroller. An ATmega328P microcontroller is present. It offers integration with several modules and can connect to sensors via analog and digital input/output connectors.

Dc Motor: DC motor converts electrical energy into mechanical motion, with a fixed stator and a rotating rotor. For our project, we chose 12V-6500rpm?

Arduino Power Supply: This is the power supply needed for the Arduino board to function.
It can be supplied with either an external adaptor or a battery. Normally, it runs on 5V.

Jumper wires and breadboard: These connecting components are what prototypes are made of. Circuit components can be temporarily placed, and connections made using a breadboard. These connections are made via jumper wires.

LED: This type of light source is powered by electricity. It can be utilized for visual feedback and is controlled by microcontrollers.

Buzzer: This component offers auditory feedback. Audio signals or alerts can be sent for specific occasions or circumstances using it.

LCD Display: This component shows the name of the detected objects when camera detects objects which are separated as ‘Dangerous Object’.

RFID-RC522: This component is powered by a second Arduino which is in the COM12 port. It controls the User Interface code according to the card reader.

Strip LEDs: It is used to make a bright area inside the conveyor belt system for Webcam performance.

Accumulator: This component is used for giving power to Strip LEDs.

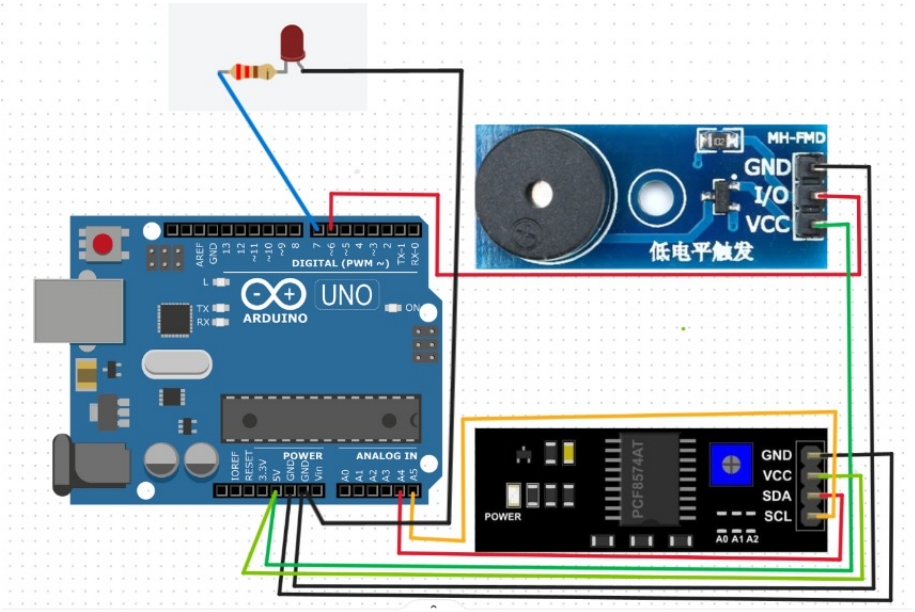

Figure 1. Buzzer-Led-LCD Display connection with Arduino Uno R3

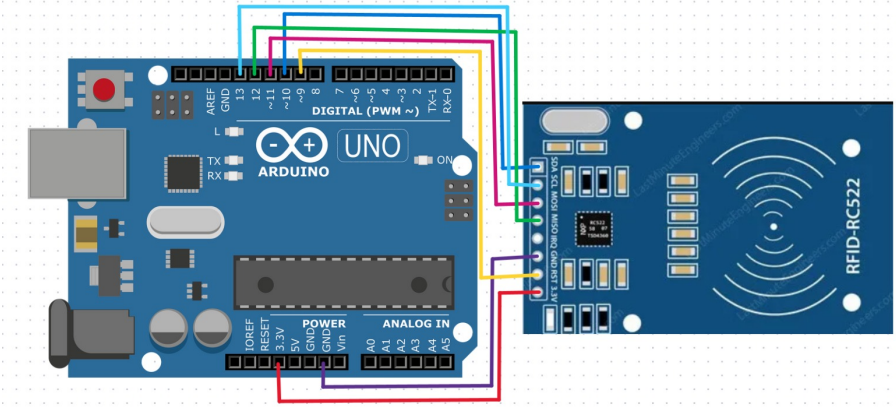

Figure 2. RFID connection with Arduino Uno R3

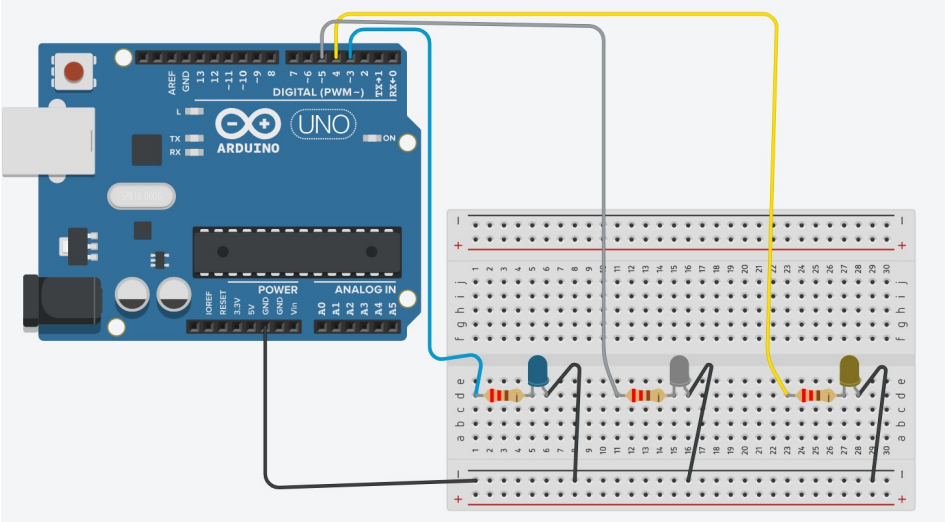

Figure 3. RFID's Led connection with Arduino Uno R3

1.3.2. Concepts

We developed three innovative concepts for our project, focusing on maximizing both functionality and practicality. Each concept is designed to effectively prevent harmful objects from passing through a designated reference point, ensuring accurate object recognition and the seamless execution of security measures. Let's delve into a detailed analysis of these concepts:

1. Conveyor System Utilizing a Belt in a Closed System: This system incorporates a continuously moving belt engineered by students to allow objects to proceed smoothly. The belt is enclosed within a closed system, providing a controlled environment that enhances detection accuracy and minimizes external interferences. A key feature of this system is the integration of a webcam, which simplifies project integration through its straightforward setup. This feature significantly enhances system functionality by providing high-quality image recognition and accuracy. The system's indoor operation benefits from LED lighting, ensuring optimal visibility and performance. The use of the Arduino Uno allows for efficient control and coordination of the system. The motor used in this design can handle voltages from 12 to 24 volts, enabling the belt to carry even heavy objects without disruption. Despite initial challenges with the belt's thickness and weight, adjustments were made to ensure smooth operation. This concept offers a robust, reliable solution with enhanced detection capabilities, making it highly practical and efficient for various applications.

2. Conveyor System Utilizing a Belt with Perforated Iron Wire: Unlike the first concept, this design also uses a continuously moving belt but is surrounded by perforated iron wire. The structure of the iron wire affects the overall weight of the system but does not impact the motor's performance. However, using suitable iron wire can increase the overall cost due to the need for specific measurements and materials. The open structure can lead to potential vulnerabilities, such as external light sources affecting the webcam's performance and possible security concerns. The integration of a webcam continues to be a feature here, aiding in the development and adaptability of the system for various applications. While this concept is functional and cost-effective, the open nature of the iron wire may not provide the same level of security and stability as the closed system in the first concept. These limitations make it less reliable compared to the first concept.

3. Conveyor System with RFID and Rail: This concept combines elements of the first design with additional features to enhance functionality. It includes a conveyor system with a belt and an RFID card reader to control and verify objects in open spaces. The system is engineered to ensure seamless operation, and the RFID technology adds an extra layer of security and accuracy. The inclusion of a webcam simplifies integration and improves overall system efficiency, ensuring better image quality

and detection accuracy. Additionally, we aim to integrate this system wirelessly to enhance control over the security barriers and take our conveyor belt system to the next level. Given the importance of security system integration, this concept provides a comprehensive solution that enhances both security and object recognition.

Conclusion:

After carefully examining the concepts based on cost, complexity, performance, and features, we have determined that the third concept, the Conveyor System with RFID and Rail, stands out as the most suitable option. Its key advantages include better system integrity, cost-effectiveness, and superior performance. The wireless integration and enhanced control over security barriers further elevate this system's capabilities. These factors make it an ideal choice for further development and potential implementation in practical settings, providing a robust solution for our security and object recognition needs. The importance of integrating into a security system cannot be overstated, and this concept ensures a seamless and effective solution for our project requirements.

	Conveyor system Utilizing a Belt in a Closed System	Conveyor system Utilizing a Belt with Perforated Iron Wire	Conveyor system Utilizing a Belt in a Closed System with RFID
Cost	medium	high	medium
Complexity	low	medium	medium
Performance	high	low	high
Features	medium	low	high

Table 1. Comparison of the three conceptual solutions

1.4. Physical Architecture

In the physical architecture section, we aimed to share a diagram depicting how the project would appear from an external perspective. The Arduino system is powered by being connected to the computer via USB. A webcam connected to our computer will capture images of objects within an enclosed space. Depending on whether the observed object is deemed dangerous or safe, if the object is evaluated as dangerous, the buzzer and LED connected to the Arduino will operate, providing a warning for security control. All detected objects are simultaneously recorded in our SQLite 3 database with their type, name, and the time they were read.

Additionally, I would like to mention a change we made this term. We connected another Arduino to the computer via USB, and when we scan our card with RFID, we can directly connect to the user interface and instantly view the hazardous objects.

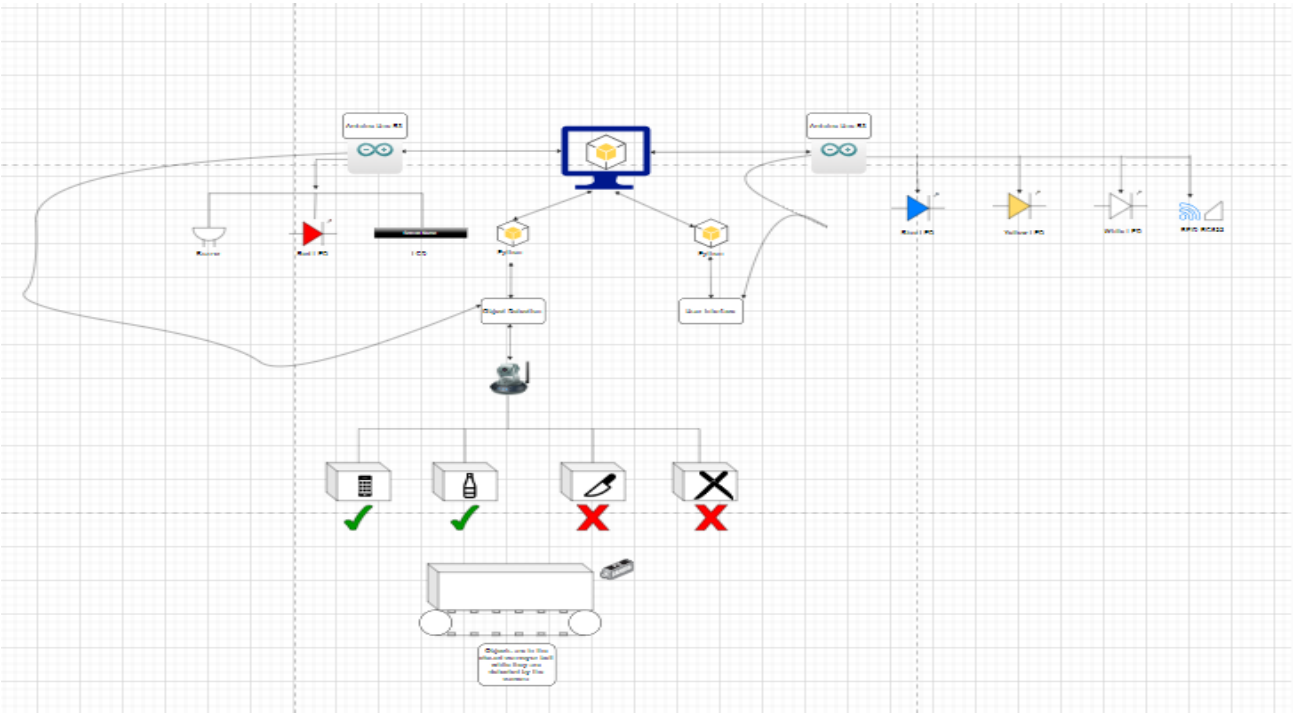

Figure 4. Physical architecture diagram for the system

As the project progresses, when an object is added while our device is active, the camera detects the object. The detected object is transmitted to our Python code software. The check in Python code determines whether the detected object is in the 'Safe' or 'Dangerous' category. If the object is safe, there will be no reaction in the Arduino codes. If the detected object is dangerous, the LED, LCD screen and buzzer connected to the Arduino in the COM9 port will work. All detected objects are written to the database created with sqlite3. The user who logs in to the User Interface can view the objects registered in the database.

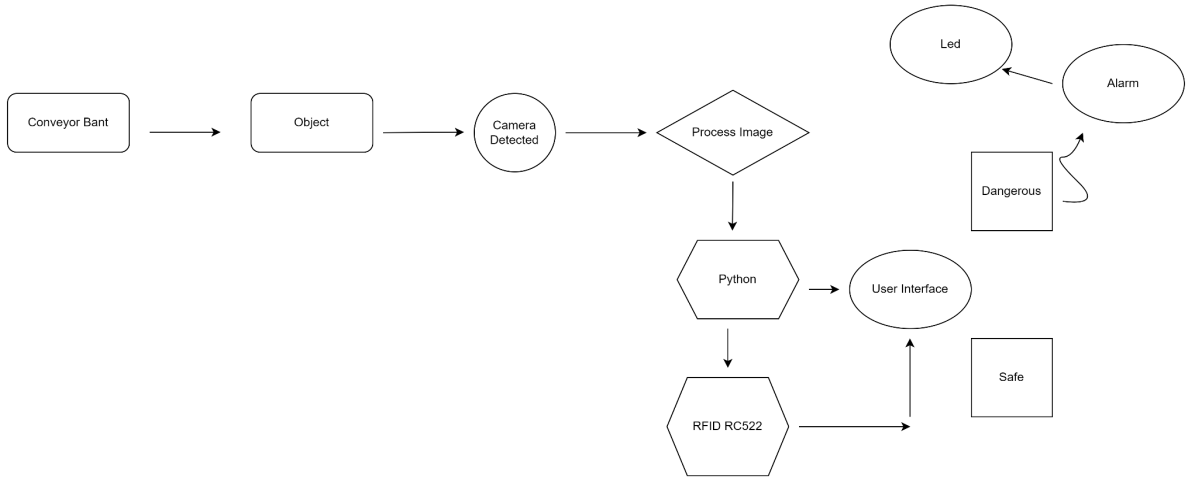

Figure 5. Process chart for the system

A security system based on camera detection, image processing using Python and user interfaces will be developed in this project. The system uses Python to process images captured by the camera and analyze them, then displays their results through a user interface. Furthermore, processed images will be evaluated for possible risk and an alert system consisting of a buzzer, LED and LCD display shall be used to inform the user when dangerous objects are detected.

Step 1: Camera Detection and Image Processing Using the OpenCV library in Python Camera detection and image processing are at the heart of this project. The images taken by a camera will be processed to find objects, and the image of those identified objects shall be obtained.

Step 2: Image Processing and Analysis Using a Python script.

The images of objects obtained in the first step will be analyzed. The properties of objects will be determined by algorithms for image processing, which then generate data over the following stages.

Step 3: User Interface Development

A user interface will be created using the Python Tkinter Graphical User Interface Library for integration of results from processed objects. The status and analysis results for objects will be visually displayed by this interface to the user.

Step 4: Detection of Hazardous Objects and Alert System A predefined security criteria will be used to evaluate the analyzed objects. A hardware warning system consisting of a buzzer and LED will be activated when an object is considered dangerous to alert the user.

Conclusion: As an example of safety systems, this project consists of camera identification, image processing and user interface design. Python is used to implement every step of the project using several popular libraries for accessibility and ease of use.

2. WORK PLAN

As a project group we are divided as a departmental of engineering.
For computer part, work separations are decided and given like that:

Ekin Özkurt: The project manager for the computer engineering part, he will also manage the project's resources and report on the progress during the project. In our project, Ekin created the user interface and participated in the object detection part. All Arduino codes, connections and communications done by himself.

Berk Ultav: Assisted in object detection work by saving identified objects to the database. He worked on enabling the feature for the user of the interface to view the objects saved in the database through the User Interface.

For the electrical part, work separations are decided and given like that:

Elif Tutar: Responsible for the design of the conveyor belt system and overseeing the entire system's construction and implementation. She ensures that the system components are integrated seamlessly and function correctly. She supported the measurement and implementation of the conveyor system's dimensions to ensure proper functionality. Additionally, she was involved in the reporting processes, the control of RFID connections, and assisted with research and procurement of project components.

Ege Anıl: He is the connection manager, responsible for setting up and maintaining the connections necessary for image detection and processing. He conducted tests and worked on the connections between the Arduino Uno R3 and the camera. He supported the measurement and implementation of the conveyor system's dimensions to ensure proper functionality. Additionally, he was responsible for selecting and configuring the appropriate motor for the project, and he assisted with research and purchasing of components. Ege also contributed to the reporting processes.

Beyza Albayrak: Responsible for integrating and maintaining the LED and buzzer connections for the security alert system. Beyza facilitated communication between teams and instructors, conducted research and procurement of project components, handled motor and Arduino connections, integrated, and tested the RFID system, managed switch connections and testing, and conducted tests and worked on the connections between the Arduino Uno R3 and the camera. She also contributed to the reporting processes.

2.1. Work Breakdown Structure (WBS)

The term "Work Breakdown Structure" describes the overall technique of breaking down the project's requirements into digestible, hierarchical parts to facilitate the observation and monitoring of the many operations. Individuals with project expertise and team members must have a deeper understanding of the goals.

The Work Breakdown Structure (WBS) is an essential tool for project management, planning, and tracking in detail. It draws a line between the divided portions of a project by presenting its main goals and constituents in a hierarchical manner. By dividing the job into smaller, more manageable pieces, this structure hopes to increase work comprehension and optimize resource usage.

WBS is important in complicated projects like "Object Recognition System for Security Automation." In this project, a conveyor system with an integrated camera is used to identify hazardous items using Arduino, cameras, and sensors inside a closed system. This mechanism is specifically built to stop potentially harmful items like knives from passing through. The WBS acts as a roadmap for organizing and monitoring every stage of the project in addition to outlining the duties and specifications of each component. More specialized activities are carried out by main components including electrical, computer, communication, control, and integration, which help the project move forward piece by piece.

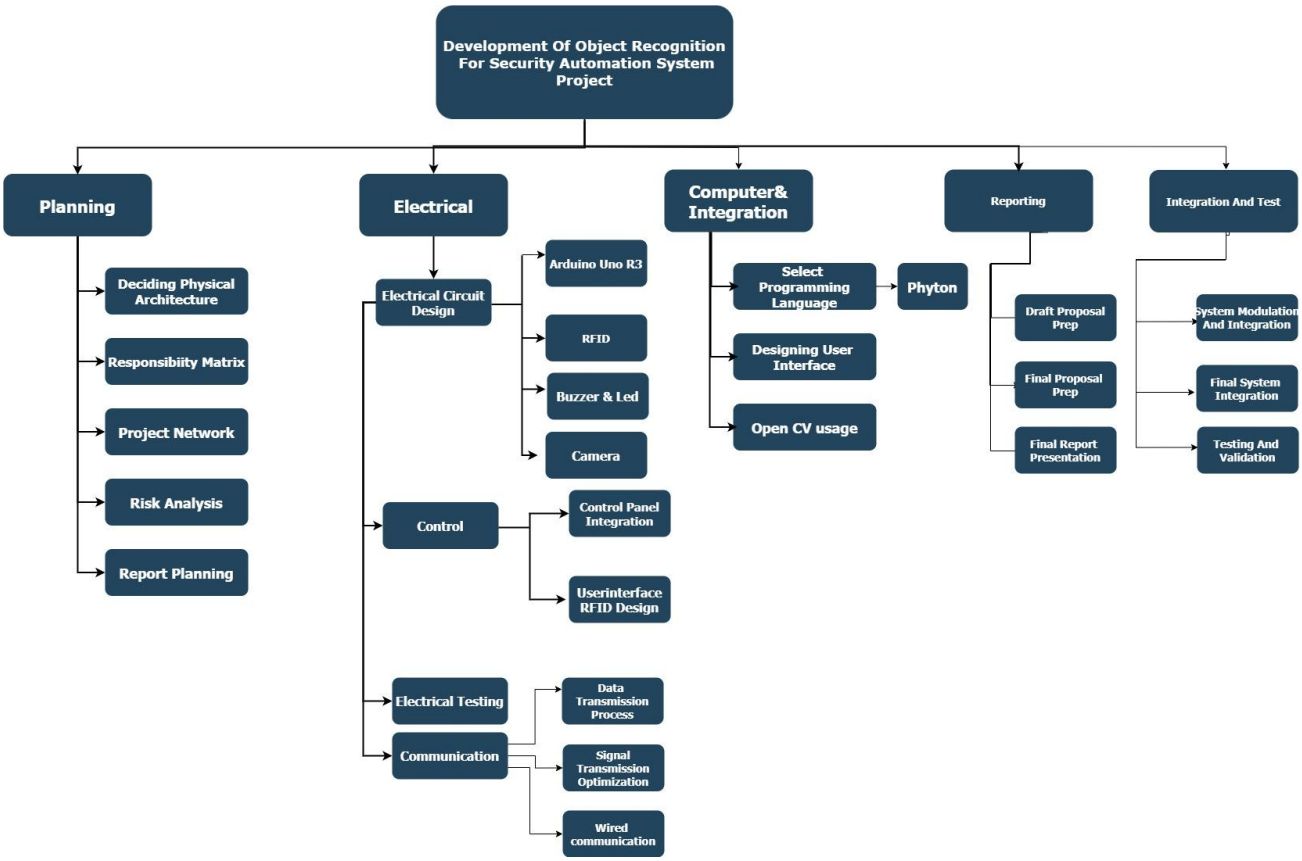

Figure 6. Work breakdown structure for the project.

2.2. Responsibility Matrix (RM)

In our project, we divided our departmental responsibilities into department groups. In this project, where the computer and electrical departments are connected, everyone is responsible for the e will contribute to the reporting part.integrated control systems. Everyone will contribute to the reporting part.

TASK	BERK	EKİN	ELİF	BEYZA	EGE
ELECTRİCAL			S	R	R
COMPUTER	S	R			
COMMUNİCATİON	R	R	R	S	R
CONTROL	R	R	R	R	S
PLANNİNG	S	R	R	R	S
REPORT	R	R	R	R	R
INTEGRATİON	R	R	S	R	S

R:Responsible S:Support

Table 2. Responsibility Matrix for the team

2.3. Project Network (PN)

- Hardware (electrical) and software (computer and integration) are the two parallel approaches that correspond to the project's main components.

- 1.1.1 & 1.2.1 Order parts: Parts are ordered for the computer and electrical components.

- 1.1.1.1 Testing Parts & 1.2.1.1 Testing & Integration Parts: Electrical components that are received are tested, and computer components are tested and integrated.

- 1.1.2.1 Testing & Integration components come after 1.1.2 order parts for the electrical portion, suggesting further testing and integration of purchased parts.

- 1.3 Language selection & 1.3 Assemble & Integration parts: The computer's components are

built and integrated, and Python is selected as the programming language.

- Electro-mechanical part integration and optimization: This stage includes integrating and refining electrical and mechanical elements that are linked to computer and electrical pathways. It highlights the need for smooth computer software and hardware interaction.

- Computer and control component integration and optimization: This stage ensures appropriate hardware and software connectivity by integrating and optimizing computer control components.

- Object Recognition System Integration & Optimization: This section integrates and optimizes the system's primary feature, object recognition. It probably entails configuring OpenCV and Python pre-trained models to use camera efficiently, Arduino, and computer operate together.

- Testing and Validation: Extensive testing is done to confirm that object recognition is accurate and dependable and that all components function as intended.

- Data Collection & Documentation: Test results are gathered and recorded, which is essential for comprehending system functionality and planning future enhancements.

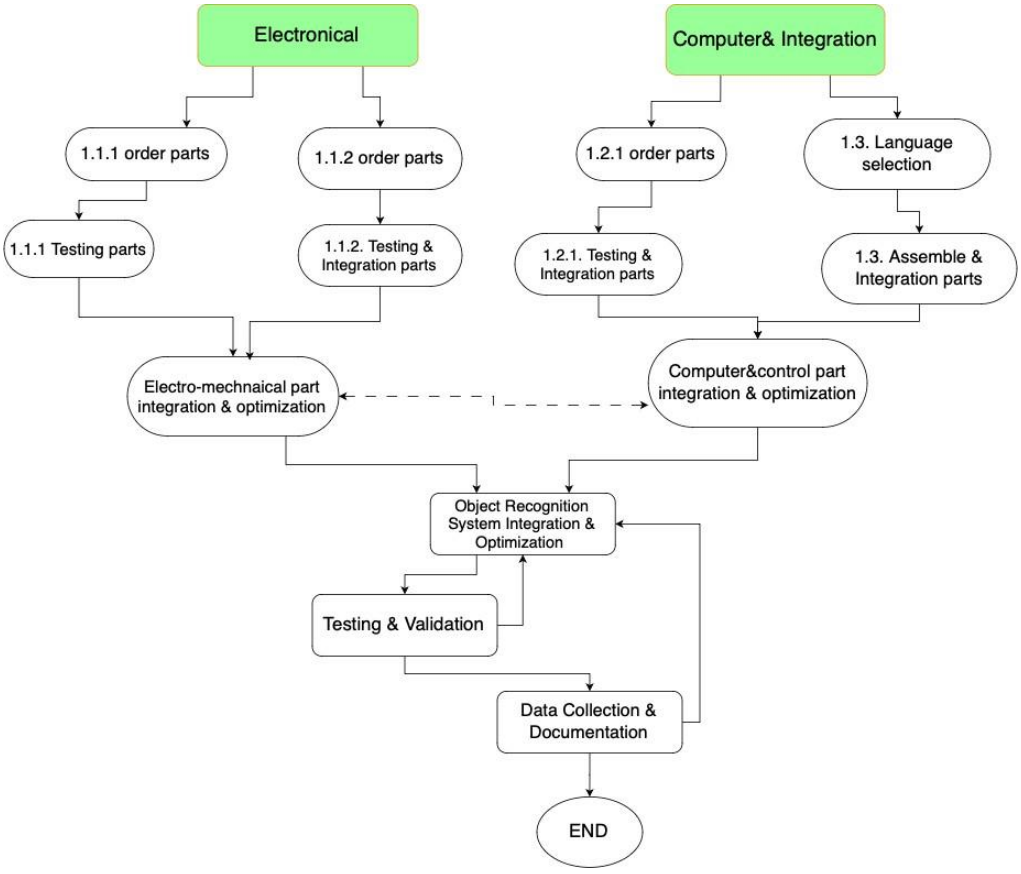

Figure 7. The project network.

2.4. Gantt chart

As indicated on the Gantt map table, tasks for which work was scheduled have been defined during the transition to the next phase of project development in the Second Semester. Tasks have been divided between the different departments. Among all tasks, the most important task had been marked by a different color. Each of the project members has been labeled in a box on the Right side of the Gantt chart, each marking his own unique color. Moreover, the departments have been separated and given individual colors to each department. In order to cover the entire project team, an overall color has been assigned.

The tasks on the left were filled according to the plan where tasks were assigned before moving to the project, after creating the Gantt chart in the form of a table.In the electromechanical part, Elif, Ege, and Beyza worked together on the design and development of the conveyor system. Elif designed the conveyor system, while Ege and Elif took on tasks related to the motor and connections. Beyza was responsible for RFID connections and communication with the computer team. As shown in the Gantt chart, the electromechanical part was completed after research and connections. This part was then integrated with the progress made in the coding phase. Electrical and electronic engineering students filled the Electromechanics part. Two groups of departments working together were planned to fill in the communication and control part. Computer engineering students were expected to add to what electrical and electronic engineering students had done. The assignment of tasks to students in computer engineering has filled the Arduino coding and Python part. The Arduino part was predominantly handled by Ekin and also the part of creating the user interface will be done by Ekin. The processing of images from the camera is done by Berk and Ekin. The task of saving this data to the database and later retrieving the saved data to the user interface is Berk's responsibility.All coding stages have been completed in such a way that the entire group will be evaluated. When the coding phase reaches a certain point in the last several weeks, it was planned to start integrating part of this subsystem with all project team members involved. The entire team has completed the documents and reports task with a view to starting in weeks which have passed since some of the data was gathered on the project, completing the Gantt chart.

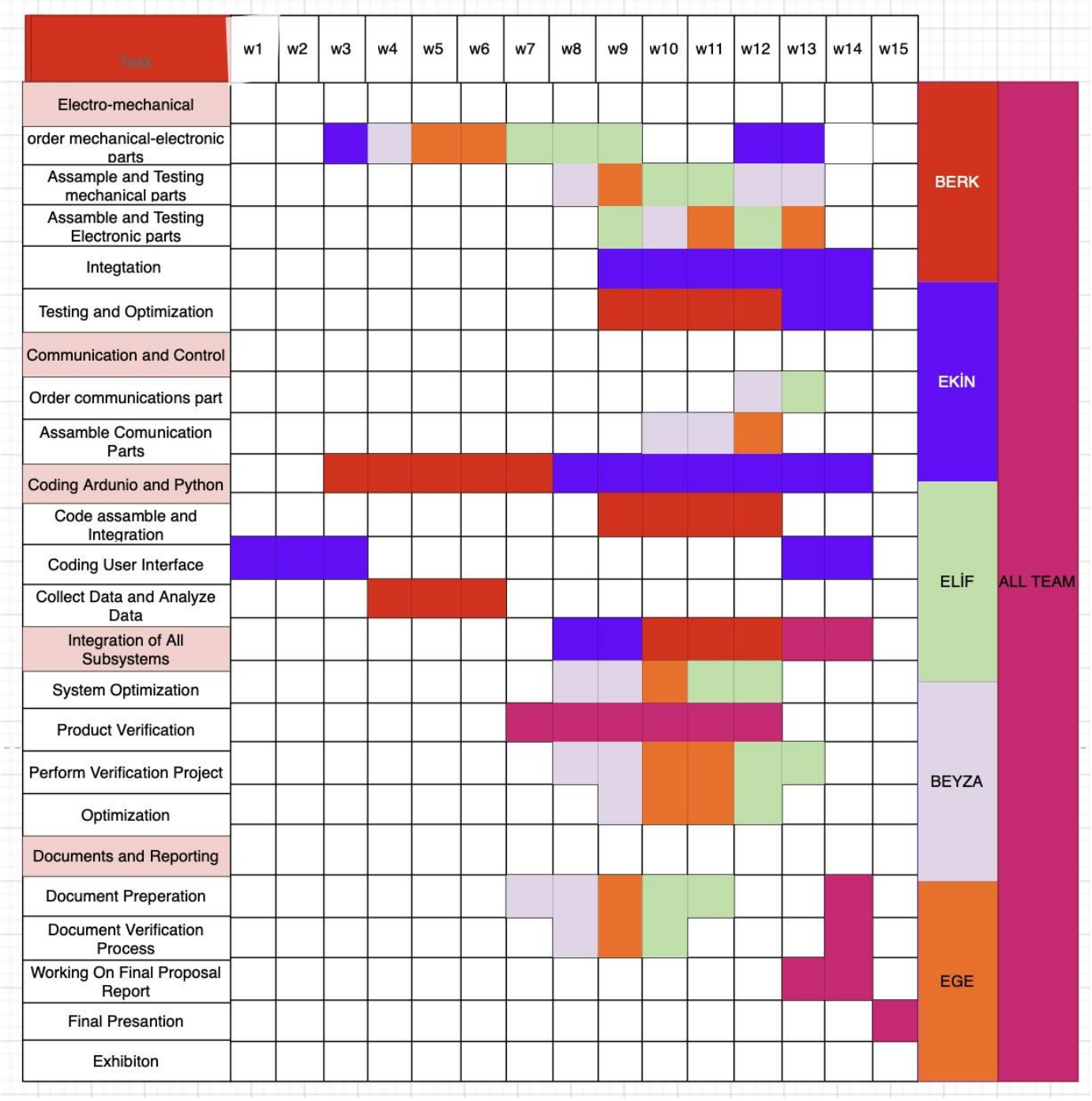

Figure 8. Gantt Chart

2.5. Costs

We decided to purchase a USB-connected webcam instead of the OV7670 camera to use in our project, due to the electrical department's inability to solve the data transfer and communication problem. We increased the use of Arduino to two. We bought a 12V/2.6A battery as a power source to power the strip LEDs we installed to illuminate the interior of the conveyor system. We bought a power supply for Arduinos, but since they communicate with the computer via serial port, they will also draw power from the computer.

List of the products	***Costs***
Arduino Uno R3	**976,83 TL**
Arduino Uno R3 Replica	**235 TL**
RFID-RC522	**52 TL**
MRV Power Supply 12V	**400 TL**
Stript Leds	**130 TL**
Piranha USB WebCam	**450 TL**
Varta 9V Power Supply for Arduino x 2	**100 TL**
Conveyor Belt System	**4500 TL**
Jumper wires and breadboard	**199,30 TL**
Led	**13,61 TL**
Buzzer	**22,46 TL**
Toy gun	**68,13 TL**
Knife	**43 TL**
ESP32-CAM	**373,19 TL**
Total cost	**7563,52 TL**
Budget supported by the university	**3000 TL**

Table 3. Cost of Tables

2.6. Risk Assessment

Technological Risks and Security Concerns

1. **Data Security and Privacy**:

- The system processes and stores sensitive data, which must be protected against unauthorized access. There is a risk of data breaches, hacking, or loss of data, which could compromise the security of the system and user privacy. Implementing robust encryption methods, secure data storage solutions, and regular security audits are essential to mitigate these risks.

2. **System Reliability and Stability**:

- The integration of various components such as cameras, RFID, sensors, and the Arduino platform introduces potential points of failure. If any component fails, the entire system's functionality can be compromised. Regular maintenance, thorough testing, and having backup systems in place can help ensure system reliability and stability.

3. **Software Vulnerabilities**:

- The system relies heavily on software for image processing, object detection, and user interface operations. Bugs, vulnerabilities, or software malfunctions can pose significant risks. Using secure coding practices, conducting regular software updates, and thorough testing can mitigate these risks.

Integration Challenges and Operational Risks

Encountered Risks:

1. **Camera Reading Issues with Moving Objects**:

- When objects pass on the conveyor belt, the camera struggled to read them due to the belt's speed. To resolve this, we implemented a switch to stop the belt temporarily, allowing the camera to capture a clear image of the object. If the object is identified as hazardous, the switch can reverse the belt to remove the object from the system.

2. **RFID Integration**:

- We initially faced issues with integrating the RFID card into the system. To overcome this, we utilized a different connection method with the UserInterface, ensuring seamless operation. This solution addressed the technological risk of failing to recognize the RFID card.

Potential Risks:

1. **LED Failure**:

●In the event of an LED failure, the system may not be adequately illuminated, causing the camera to struggle with object detection in low-light conditions. Regular maintenance checks and keeping spare LEDs on hand can mitigate this risk.

2. **Camera Malfunction in Low Light**:

●The camera may fail to detect objects correctly in a dark environment if the LED fails. Installing backup lighting solutions and ensuring the camera's low-light performance can help manage this risk.

3. **RFID Card Misplacement**:

●If the RFID card is lost, the user will be unable to access the system to review or stop the operations. Implementing a secure card management protocol, such as having backup cards and a logging system for card usage, can mitigate this risk.

4. **Switch Malfunction**:

●If the switch used to stop and reverse the conveyor belt fails, the system could malfunction, leading to potential security breaches. Regular testing and maintenance of the switch mechanism are necessary to prevent this risk.

5. **Component Compatibility Issues**:

●Ensuring that all components (cameras, RFID, sensors, Arduino, motors) work seamlessly together is a significant challenge. Compatibility issues can arise, leading to operational inefficiencies or failures. Conducting extensive compatibility testing and having a well-documented integration plan can address these challenges.

By addressing these encountered and potential risks through careful planning, regular testing, and robust security measures, the project can achieve its goal of creating an effective object recognition and security automation system.

RISK LEVEL		Severity of the event on the project success		
		Minor	Moderate	Major
Probability of the event occuring	Unlikely	VERY LOW	LOW	MEDIUM
	Possible	LOW	MEDIUM	HIGH
	Likely	MEDIUM	HIGH	VERY HIGH

VERY LOW	This event is very low risk and so does not require any plan for mitigation. In the unlikely event that it does occur there will be only a minor effect on the project.
LOW	This event is low-risk; a preliminary study on a plan of action to recover from the event can be performed and noted.
MEDIUM	This event presents a signficant risk; a plan of action to recover from it should be made and resources sourced in advance.
HIGH	This event presents a very signficant risk. Consider changing the product design/project plan to reduce the risk; else a plan of action for recovery should be made and resources sourced in advance.
VERY HIGH	**This is an unacceptable risk. The product design/project plan must be changed to reduce the risk to an acceptable level.**

Table 4. Risk Matrix

Failure Event	Probability	Severity	Risk Level	Plan of Action
Camera reading issues with moving objects	Likely	Major	HIGH	Implement a switch to stop the belt temporarily, allowing the camera to capture a clear image.
RFID card integration	Likely	Moderate	HIGH	Use a different connection method with the UserInterface to ensure seamless operation.
LED failure	Possible	Moderate	MEDIUM	Perform regular maintenance checks and keep spare LEDs on hand.
Camera malfunction in low light	Possible	Major	HIGH	Install backup lighting solutions and ensure the camera's low-light performance.
RFID card misplacement	Possible	Moderate	MEDIUM	Implement a secure card management protocol, such as backup cards and a logging system.
Switch malfunction	Unlikely	Major	MEDIUM	Regularly test and maintain the switch mechanism to prevent failures.
Component compatibility issues	Possible	Moderate	MEDIUM	Conduct extensive compatibility testing and have a well-documented integration plan.

Table 5. Risk Assessment

3. SUB-SYSTEMS

3.1. Electrical Arduino Integration Part

3.1.1. Requirements

Voltage and Current needs:

Webcam (Piranha USB): Voltage: 5V (USB powered) - Current: 150-200mA during data transmission and picture taking - Peak Current: 250mA.

Arduino Uno: Voltage: 5V (powered via USB) - Current: 50mA (idle) - Peak Current: 200mA (with all components active)

LCD Screen: Voltage:5V - Current:20mA - Peak Current:30mA

Buzzer: Voltage:5V - Current:10mA - Peak Current:20mA

LED: Voltage:2V (with current limiting resistor) - Current:15mA - Peak Current:20mA

RFID-RC522: Voltage:3.3V - Current:30mA - Peak Current:50mA

Strip LED (inside conveyor system): Voltage:12V - Current:500mA - Peak Current:700mA.

Motors (4 for conveyor belt): Voltage:12V - Current:100mA each - Peak Current:150mA each.

Power Source Stability: Type; AC-DC Adapter with a battery backup (for strip LEDs)

Longevity: Continuous operation for at least 6 hours

Capacity: AC-DC Adapter:5V,2A for Arduino and peripherals, Battery for LED Strip:12V,2A

Power Regulation Strategy: Use voltage regulators to ensure steady 5V supply, Capacitors placed near critical components to smooth out voltage fluctuations, use buck converters to efficiently step down from 12V to 5V for the LEDs and other 5V components.

Fluctuation Tolerances: Webcam: ±5% (4.75V to 5.25V)

Arduino Uno: ±5% (4.75V to 5.25V)

LCD Screen: ±10% (4.5V to 5.5V)

Buzzer and LEDs: ±10% (4.5V to 5.5V)

RFID-RC522: ±5% for 3.3V and ±10% for 5V

Strip LED: ±5% (11.4V to 12.6V)

Motors: ±10% (10.8V to 13.2V)

Power-on Surge Features:

Surge Current Measurement:

Initial surge current can peak up to 1A due to the inrush current of motors and LEDs.

Protection Measures:

Implement soft-start circuits to gradually ramp up power.

Use inrush current limiters (NTC thermistors) to manage surge current.

Add fuses rated slightly above the peak current needs to prevent overcurrent damage.

By meticulously addressing voltage and current needs, ensuring power source stability, accounting for fluctuation tolerances, and incorporating power-on surge protection features, we can ensure reliable and efficient operation of the "Object Recognition for Security Automation" system.

Processing Delays in Real Time: Maximum Processing Delay: The system should process images within 500 milliseconds from the moment an image is captured to when it is ready for object identification. This standard ensures that the system meets industry demands for speed in object identification.

Data Transfer Rates: Necessary Data Transfer Rates

From Camera to Computer:

USB Webcam (Piranha): Minimum 480 Mbps (USB 2.0 standard) to handle real-time video streaming and image capture.

From Arduino to Computer:

Serial Communication: Minimum 115200 baud rate to ensure prompt transmission of control signals and sensor data.

Inter-Component Communication:

Arduino to Computer: Maximum 100 milliseconds delay for serial data transmission to maintain real-time processing requirements.

Wi-Fi Connection Establishment: ESP32 modules should establish Wi-Fi connections within 2 seconds to ensure minimal disruption in operations.

Reliability and Error Rates:

Acceptable Failure Frequencies: Mean Time Between Failures (MTBF)

Webcam: 10,000 hours - Arduino Uno: 50,000 hours - RFID Reader: 30,000 hours

Maximum Downtime per Hour:

Less than 1 minute per operational hour to ensure high system availability and reliability.

Temperature Variations: Thermal Management Approach

Heat Sinks and Forced Air Cooling:

Install heat sinks on Arduino and other heat-sensitive components.

Use small fans for forced air cooling in enclosed environments.

Operating Temperature Range:

Maintain component temperatures between 0°C to 70°C.

Impact of Temperature Fluctuations:

Ensure that all components are rated for industrial temperature ranges to maintain performance and reliability.

Monitor temperature using sensors and implement automatic shutdown in case of overheating.

Electromagnetic Interference (EMI):

Possible EMI Sources:

Motors in the conveyor belt system

External wireless devices

Shielding and Filtering:

Use metal enclosures for sensitive electronics to shield from EMI.

Implement ferrite beads and EMI filters on power lines and data cables.

Isolation Approaches:

Use optocouplers for isolating signal paths where necessary.

Vibration and Shock Resistance: Vibration and Shock Specifications

G-force Levels:

Components must withstand vibrations up to 5 g and shocks up to 10 g.

Testing for Solder Joint Integrity:

Conduct vibration tests using a vibration table to simulate operational environments.

Ensure that all components are securely mounted with appropriate adhesives and mechanical fasteners.

Component Mounting:

Use vibration-damping mounts and gaskets to protect against mechanical stress.

By setting these standards and implementing these measures, the "Object Recognition for Security Automation" system can maintain high performance, reliability, and resilience in various operational environment.

3.1.2. Technologies and Methods

Firmware Compatibility

Device Firmware:

Arduino: Ensure the latest stable version of the Arduino firmware is used, which supports communication protocols necessary for object recognition tasks.

Camera Module: Replace the OV7670 with a Piranha USB webcam, ensuring compatibility with Python-based image processing software and the Arduino ecosystem.

Firmware Update Mechanisms:

Arduino Bootloader Updates: Configure the Arduino bootloader to enable easy updates. Use tools like AVRDUDE for updating the firmware via USB without requiring physical access to the hardware.

Driver Availability

Operating System Support:

Drivers for Communication: Verify that the chosen operating system (Windows, Linux, macOS) has up-to-date and reliable drivers for USB communication with the Arduino. Ensure the drivers support high-performance tasks and provide low-level hardware access.

Camera Module Drivers:

Piranha USB Webcam: Ensure drivers for the Piranha webcam are available and compatible with the operating system. The webcam should provide necessary frame rates and image resolutions for real-time processing. Utilize standard UVC drivers available on most operating systems for easy integration.

Real time Data Streaming

Protocol Selection:

Video Data Streaming: Use Real-Time Streaming Protocol (RTSP) or Real-Time Transport Protocol (RTP) for streaming video data from the webcam to the computer. These protocols are designed for low-latency, real-time streaming applications.

Data Communication: Use Serial communication (UART) for reliable data transfer between Arduino and the computer.

Bandwidth and Throughput:

Video Data Requirements: Calculate the bandwidth required for video data based on the resolution and frame rate of the Piranha webcam. Ensure that the USB interface and network infrastructure can handle this throughput without issues. For example, a 720p webcam at 30fps requires approximately 20 Mbps.

Latency Considerations:

Minimal Latency Protocols: Choose protocols that provide minimal latency to meet the real-time requirements of the object recognition system. RTSP and RTP are preferred for video streaming due to their low latency characteristics.

Optimization Techniques: Implement buffering strategies and optimize the communication code to reduce latency further. Use efficient data processing algorithms in Python to ensure quick turnaround times for object recognition.

By carefully selecting and integrating the appropriate technologies, firmware versions, and communication protocols, the "Object Recognition for Security Automation" project can achieve reliable, real-time performance, ensuring seamless operation and timely threat detection.

Methodology for Data Transfer Synchronization

Communication Handshake Protocols:

Initial Connection Protocols: Implement a robust handshake procedure to establish a connection between the Arduino and the computer before any data transfer begins. This can involve a series of signals or messages exchanged to confirm both sides are ready for communication.

Example: Arduino sends an "INIT" signal, the computer responds with an "ACK" signal to confirm readiness

Data Packet Acknowledgment Systems: Each data packet sent from the Arduino to the computer should be acknowledged by the receiving end. This ensures that the packet has been received successfully.

Example: After sending a data packet, Arduino waits for an "ACK" (Acknowledgment) signal from the computer before sending the next packet.

Error Recovery Procedures: Implement procedures to detect and handle errors during data transmission. This includes sending a "NAK" (Negative Acknowledgment) signal when a packet is not received correctly, prompting the sender to retransmit the packet.

Example: If the Arduino does not receive an acknowledgment within a specified timeout period, it resends the packet.

Data Integrity Checks:

Checksums or Cyclic Redundancy Check (CRC): Use checksums or CRC methods to ensure the integrity of data packets during transmission. Each packet includes a checksum or CRC value that the receiver can verify against the received data.

Example: Arduino calculates a CRC value for each data packet before sending, and the computer verifies this CRC upon receipt.

Automated Retry Procedures: Establish automated retry mechanisms to handle cases of data corruption. If a data integrity check fails, the receiver requests a retransmission of the corrupted packet.

Example: If the CRC verification fails, the computer sends a "RETRY" signal to the Arduino, which then retransmits the corrupted packet.

Technology Selection and Methodology

The phase involves ensuring that the chosen hardware and software components are not only suitable individually but also compatible and capable of working together to create a reliable, real-time object identification system. The methodology includes:

Verifying Compatibility: Confirming that hardware components like the Piranha USB webcam, Arduino, and computer drivers are compatible and support necessary protocols for object recognition tasks.

Establishing Communication: Setting up robust communication protocols, including handshake procedures, data packet acknowledgments, and error recovery mechanisms, to ensure reliable data transfer.

Ensuring Data Integrity: Implementing checksums or CRC for data integrity and automated retry procedures to handle data corruption.

This approach ensures a well-coordinated and responsive system, maintaining data integrity and real-time performance throughout its operation.

3.1.3. Conceptualization

The conveyor system's carefully planned design prioritizes safety and maximizes the detecting area. It has an elevated construction with enclosed sides and a top cover. The central component of our system is a camera that is positioned so that it can see above the conveyor belt and keep a watch on the items that pass by it. The camera continuously takes pictures of the objects in motion while the conveyor belt is in use. These photos are then easily transferred to a computer system, where OpenCV, a powerful and adaptable Python image processing toolkit, is used to carefully analyze the photographs. Our system's visual processing enables it to quickly recognize things that may pose a threat to safety, allowing for timely intervention.

When our system finds a potentially dangerous object, it starts responding right away. To alert others to the existence of a potentially dangerous object, a buzzer is set off. Concurrently, an LCD panel prominently displays a warning message, giving a clear and visible signal of the discovered hazard.

The conveyor system's incorporation of an RFID reader makes a substantial contribution to the project's overall usefulness and integrity. The main tool for recognizing hazardous compounds is an image processing system; however, by adding RFID technology, the system's functionality is improved by offering more levels of data tracking and user engagement.

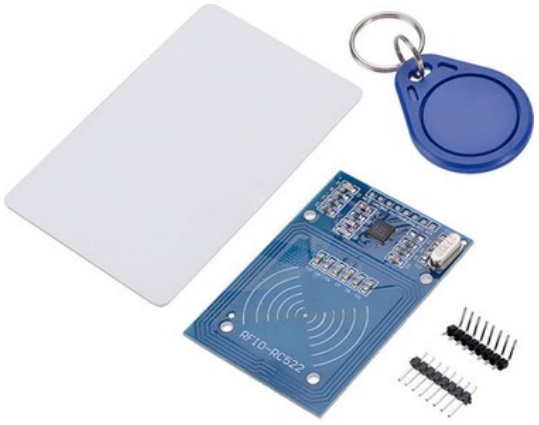

Figure 9. RFID Cards

The RFID reader is connected to a different Arduino circuit, which allows us to seamlessly integrate the digital interface with tangible things (RFID cards). Can examine figure 8. The technology launches the computer's user interface upon scanning an RFID card, providing instant access to vital data regarding identified hazardous items. This integration improves the system's capacity to efficiently record and analyze data while also streamlining the user experience.

3.1.4. Physical Architecture

Our security automation system's physical architecture is carefully built to ensure effective monitoring and detection of potentially dangerous items on a conveyor belt. The system consists of a conveyor belt that is raised off the ground a little bit, enclosed by strong wooden panels on both the top and sides to create a controlled environment. The iron framework offers stability and support, and the wooden panels offer protection from outside interference like dust and changing light.

Above the conveyor belt is a fixed-position camera. In order to provide constant, shadow-free illumination and improve image quality for precise processing, LED lights are placed strategically.

Using algorithms to identify potentially dangerous objects, OpenCV software is run on a computer system that is connected to the camera and processes acquired images in real-time. When a potentially dangerous item is detected, an LCD screen and an audible buzzer provide quick alerts. The LCD shows warning words, and the buzzer sounds an alarm. This architecture can be seen figure 10.

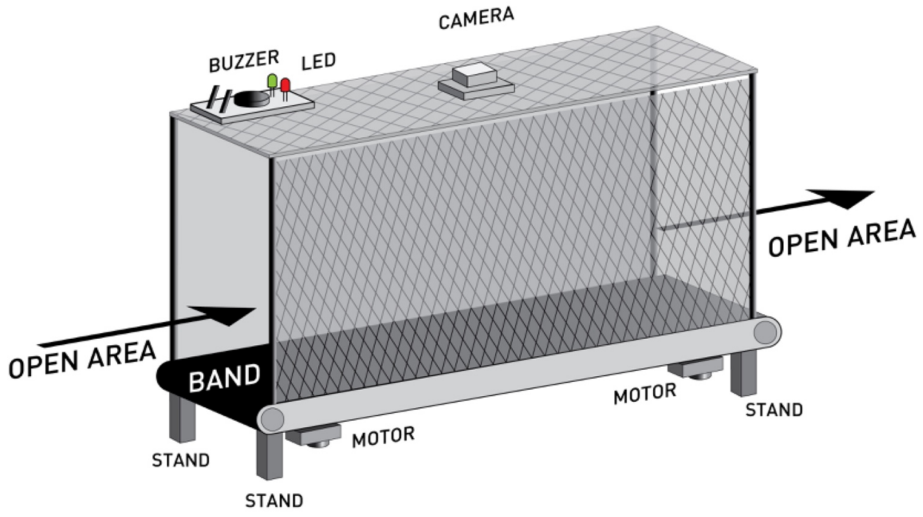

Figure 10. Conveyor System

An RFID card reader integrated with a different Arduino circuit is also part of the system. The Arduino initiates a computer user interface that presents comprehensive details about the dangerous object, such as its name and the moment it was discovered, when a security guard scans an RFID card. A dependable power source powers the entire system, guaranteeing that the conveyor belt, camera, processing unit, alert systems, and RFID reader all run continuously. This physical architecture's combination of iron and wood allows all the parts to work together seamlessly, ensuring reliable identification of potentially dangerous objects on the conveyor belt and quick action when necessary.

3.1.5. Materialization

To successfully implement our Object Recognition System for Security Automation, we will procure and assemble various hardware and software components. The hardware components include a conveyor belt system made of iron and wood, with a rubber or fabric belt, driven by a 12V-24V DC motor with speed control. Electronic switches will control the start, stop, and reverse functions of the belt, while proximity sensors will detect the presence of objects.

The camera system will consist of a high-resolution webcam integrated with Arduino for real-time image capture. LED lights will ensure proper illumination of objects for accurate detection, and adjustable camera mounts will provide stability. The RFID system will include an RFID reader for reading access control cards and different RFID cards for opening and closing the User Interface.

For microcontrollers, we will use the Arduino Uno R3 as the primary microcontroller for processing inputs and controlling outputs. The alert system will comprise LEDs for visual indicators, buzzers for audible alarms, and an LCD display for real-time status updates and alerts. Miscellaneous items like cables, connectors, power supplies, and protective enclosures for sensitive electronics will also be required.

The software components will include Python for image processing using the OpenCV library and user interface development with Tkinter, as well as the Arduino IDE for coding and uploading to the microcontrollers. We will use specific libraries and tools like OpenCV for real-time image processing, Tkinter for the GUI, and an RFID library for integrating RFID functionality with the Arduino.

Integration and testing will involve careful assembly of the conveyor belt, webcam, RFID, and alert systems. Proper wiring and secure mounting of the webcam and sensors on the conveyor belt system will be essential. Software development will focus on developing and testing the code for image processing, object detection, and system control, along with creating a user-friendly interface for monitoring and controlling the system.

Functionality tests will be conducted on each component individually and then as an integrated system, followed by calibration of sensors, webcam, and motors to ensure optimal performance. Safety checks will verify that the alert system works correctly, and that the conveyor belt stops and reverses as needed. Comprehensive documentation, including user manuals and maintenance guides, will be provided to ensure proper operation and regular maintenance of the system.

By following this materialization plan, we will ensure that all components are acquired, assembled, and tested efficiently, leading to the successful implementation of our Object Recognition System for Security Automation.

3.1.6. Evaluation

Upon completing the assembly and integration of the system, we conducted a thorough evaluation to ensure it met the project's objectives and performance criteria. This evaluation involved performance testing to assess the system's ability to accurately detect and identify objects in real-time and verify the responsiveness and accuracy of the alert system, including the LEDs, buzzers, and LCD display. Reliability testing was conducted by running the system continuously over extended periods to ensure stability and reliability, monitoring for any potential failures or issues in the hardware or software components. Usability testing ensured the user interface was intuitive and easy to use for security personnel. Security testing evaluated the RFID system's effectiveness in controlling access and preventing unauthorized use and checked the system's resilience to potential security threats and data breaches.

3.2. Python Coding Language

As computer engineering students, our role in the project involves implementing the subsystem using Python. This covers the developments and improvements we will make to the project using Python. In order for the project to work successfully, Arduino components such as LED, buzzer, LCD, RFID-RC522 and the USB camera used should be integrated into the code, such as the User Interface, object recognition system, etc. that we will write. It is very important that it integrates seamlessly with Python codes on different subjects.

3.2.1. Requirements

The Python subsystem for the "Object Recognition for Security Automation" project is designed to fulfill the following requirements:

Real-time Image Processing: Efficiently processing images captured at the Piranha Webcam camera in real-time.

Object Classification: Use OpenCV to implement an efficient object detection and classification as desired.

Storage of Data: Data which is collected from camera view is written under sqlite3 based database from Python code.

Communication Interface: To ensure a perfect transfer of data, set up a secure communication interface with cables from serial ports between Python and Arduino.

User Interface Integration: Develop a simple user interface to display the list of specified objects together with their security classification, detection time and label. Also, users can view the previous registered users from the interface.

Alert Triggering Mechanism: Implement a mechanism on Arduino COM9 to trigger alerts on (buzzer, LED and LCD) when potentially dangerous objects are detected, and on the other Arduino which is located on COM12 trigger 3 different LEDs according to RFID card read and open-close User Interface code upon the card.

3.2.2. Technologies and Methods

The OpenCV library for image processing and object detection is exploited by the Python subsystem. Serial communication is the way to communicate between Arduinos and Python. To simplify and ease integration, the user interface is developed with a Python library Tkinter.

3.2.3. Conceptualization

The design of modular and scalable systems is part of the conceptualization process. The conceptualization of the Python code involves defining functions for picture processing, object classification, communication handling and user interface updates. The conceptual document also notes a seamless integration between the Python subsystem and the Arduino components.

3.2.4. Physical Architecture

The Python subsystem's physical structure is based on a modular approach. Modules for image processing, classification of objects, communication and user interface are included. These modules interact with each other to provide a complete security system that is enhanced by object recognition.

3.2.5. Materialization

It involves translating conceptual designs into practical code at the materialization stage. The processing of images, object classification and user interface interaction requires the creation of Python scripts. The code will be arranged in such a way as to provide for modularity, readability, and ease of maintenance.

3.2.6. Evaluation

An evaluation phase is focused on verifying the Python subsystem's performance and functionality. Real time processing speed, accuracy of object detection and classification and user interface responsiveness are metrics. Continuous test and refinement ensure that the Python subsystem is seamlessly integrated into the Arduino components, thereby contributing effectively to a complete success of this security automation project.

4. INTEGRATION AND EVALUATION

4.1. Integration

1.Reminder of Sub-Systems for Integration: Reminder of Sub-Systems for Integration: As we move forward with our project, it is important to stress the integration of key subsystems. The main components shall be as follows:

Arduino Uno R3 and a webcam connected to the computer via USB: Responsible for image capture and initial object detection.

Python-based object detection algorithm: Processes images from the webcam connected to the computer via USB, classifies objects, interacts with the SQLite3 database we created, and interfaces with the user interface in real time.

User Interface (UI): As the main point of interaction for the user, displays a list of identified objects and their safety classification. We can view the identified hazardous items through the UI and take action accordingly.

2.Project Network PN and Gantt charts: We have set up the Project Network PN and the Gantt chart in order to effectively manage the project. These instruments, which facilitate a structured approach to the integration process, provide an overview of tasks and their interconnections.

3.Organization and schedule of planned integration: Referring to the Project Network (PN) and Gantt chart, the integration phase will be systematically organized. There will be several stages in the integration process:

Stage 1: Webcam-Python Image Processing Integration: Using our object recognition algorithm in Python, we identify objects from the images captured by the webcam.

Stage 2: Here, we establish a connection with Arduino. Images from the camera are processed using our object recognition algorithm in Python. If the identified object is prohibited, the buzzer and LED on the Arduino are activated. Simultaneously, the objects detected by our Python code are recorded in our SQLite3 database as either safe or dangerous objects. The data in our SQLite database can be viewed in the "View Products" section of our user interface, showing what the object is, when it was detected, and its category.

Stage 3: In Stage 2, when we detect a dangerous object, we scan our RFID card connected to the Arduino to immediately learn what the object is and take action. Scanning another card will close the user interface.

4.2. Evaluation

The assessment of both operational and performance aspects is crucial to ensure that the "Object Recognition for Automation Security" project can be successfully implemented. Future improvements will be based on this evaluation. In addition, the product's functional and performance requirements are set out below:

Functional Requirements: Real time object identification and classification. The communication between Arduino and Python components is correct. Activation of security alerts (buzzer and LED) upon detection of potentially dangerous objects. A user-friendly interface displaying a list of recognized objects, as well as their safety classification.

Performance Requirements: Object detection and alert activation have a low latency. Object classification accuracy is high. A reliable communication between Arduino and Python. An intuitive and responsive user interface.

Plans for experimentation and data analysis:

Object detection accuracy experiment: Perform experiments using a variety of test scenarios and objects. Data on object detection accuracy are collected. To identify and correct any errors, analyze the collected data. If necessary, add improvements to the algorithm for detecting objects.

Communication Latency and Reliability Testing: See the response times between Arduino and Python components. In order to verify the reliability, simulate different network conditions. Data collection on communication delays and interruptions. Perform data analysis to optimize communication protocols for better performance.

Alert Activation Time Experiment: Evaluate the time it takes for the security alerts (buzzer and LED) to activate. Use different conditions to gather data on the time of alert activation. Analyze the data to guarantee an accurate and reliable notification. In order to satisfy performance requirements, adapt the system as necessary.

User Interface Responsiveness Study: Review the responsiveness and user friendliness of your interface. Use feedback from users on design and functionality of interfaces. To improve the user experience, analyze feedback and adjust. In order to improve the user experience, consider implementing additional features.

Long-Term Reliability Testing: To identify potential problems, the system needs to be continuously run for an extended period. Data are collected on system stability and performance over time. In order to address any issues related to the system's reliability, analyze its longer-term data. Use the findings to implement updates and improvement.

5. SUMMARY AND CONCLUSION

In summary, the "Object Recognition for Security Automation" project represents a significant advancement in security automation by utilizing advanced technologies for enhanced threat detection and response mechanisms. This project's primary objective is to use a Piranha brand USB webcam connected to a computer for object recognition in security applications, replacing the previously considered OV7670 camera.

During the first phase, various design alternatives were explored, including conveyor systems and X-ray mechanisms. However, the prohibitive costs associated with these approaches led us to a more streamlined solution. Ultimately, we implemented a conveyor belt system driven by four motors, with the webcam mounted at the open end to detect objects. This setup allows for efficient and cost-effective object detection using camera technology instead of X-ray mechanisms.

The system includes two Arduino Uno boards, each with specific roles. One Arduino is equipped with an LCD screen, a buzzer, and a red LED. When a dangerous object is detected, these components are activated to provide immediate security signals. The second Arduino is integrated with an RFID-RC522 reader and three LEDs. When an RFID card is scanned, it triggers the user interface to either display or hide based on the card ID, and the color of the illuminated LED changes accordingly.

To ensure proper lighting for the webcam, a strip of LEDs is installed inside the conveyor belt system, powered by a battery. The Arduinos receive power from the computer, and communication with the Python-based object recognition code is carried out via serial ports. The project has designated RFID cards as the WiFi module to manage system access and control.

In conclusion, the "Object Recognition for Security Automation" project has successfully leveraged the capabilities of the Piranha USB webcam, Arduino boards, and Python to create an efficient and cost-effective security solution. In security-sensitive environments, the use of object recognition technology enhances proactive threat mitigation, making it a valuable asset. This project highlights the importance of integrating novel technologies to address today's security challenges, paving the way for future developments in safety automation.

REFERENCES

[1] Bogdan Knysh and Yaroslav Kulyk , Improving a model of object recognition in images based on a convolutional neural network , 2021-06-30 https://journals.uran.ua/eejet/article/view/233786

[2] Tessie , OV7670 Camera Module: Datasheet, Specifications and Comparison, 2021-11-05 https://www.utmel.com/components/ov7670-camera-module-datasheet-specifications-and- comparison?id=797

[3] Object Recognition in Security: Everything You Need to Know, 2023-09-07 https://www.algotive.ai/blog/object-recognition-in-security

[4] Akshay Gupta, Computer Vision Using OpenCV – With Practical Examples, 2021-05-25 https://www.analyticsvidhya.com/blog/2021/05/computer-vision-using-opencv-with-practical- examples/

[5] Elseiver, Computer Science Review Volume 40, A systematic literature review on prototyping with Arduino: Applications, challenges, advantages, and limitations, 2021-05 https://www.sciencedirect.com/science/article/abs/pii/S1574013721000046?via%3Dihub

[6] Mudassar Tamboli, ESP32+OV7670 — WebSocket Video Camera, 2018-06-09 https://medium.com/@mudassar.tamboli/esp32-ov7670-websocket-video-camera-26c35aedcc64

[7] Indrek, OV7670 Camera module to PC with Arduino, 2021-11-03 www.youtube.com/watch?v=R94WZH8XAvM&ab_channel=Indrek

[8] Robojax, How to setup and use ESP32 Cam with Micro USB WiFi Camera, 2022-07-13 https://www.youtube.com/watch?v=RCtVxZnjPmY

[9] Electronic Clinic, ESP32 Cam: ESP32 Camera Programming using Arduino, AI Thinker, Issues Fixed, esp32 IoT project, 2020-07-12 https://www.youtube.com/watch?v=DdybJZ58mlI

[10] Zain Mumtaz, An Automation System for Controlling Streetlights and Monitoring Objects Using Arduino,2018-09 https://www.mdpi.com/1424-8220/18/10/3178

[11] Ninad Mehendale, Interfacing Camera Module OV7670 with Arduino https://deliverypdf.ssrn.com/delivery.php?ID=376121095093094107003092031098007123096024026051006017127100015071122093006127019028042012019003037044061067066006088119 1

22
0650130800220300861160730231071050101160720650650111021260230181200950820001000
09 20031161260250290710920830300820931081220961141 14&EXT=pdf&INDEX=TRUE

[12] Çağatay Odabaşı Youtube Channel https://www.youtube.com/watch?v=Fe1b2sbfUIo

[13] Vipul Kumar, How to Detect Objects in Real-Time Using OpenCV and Python

https://towardsdatascience.com/how-to-detect-objects-in-real-time-using-opencv-and-python-c1ba0c2c69c0

Puput Dani Prasetyo Adi, Yuyu Wahyu, Performance evaluation of ESP32 Camera Face Recognition for various projects, 2022-02-18 https://pubs.ascee.org/index.php/iota/article/view/512/146

[14] Aditya Reddy Lellapati- Bharath Kumar Pidugu , Development of Advanced Alerting System using Arduino interfaced with ESP32 CAM , 2022-10-08 https://ieeexplore.ieee.org/abstract/document/9988762

APPENDIX,

OBJECT DETECTION: The code that detects objects, communicates with Arduinos, enables them to perform their tasks, and saves the detected objects to the database.

```python
import cv2
from datetime import datetime
from detecto.core import Model
import sqlite3
import serial
import time
import subprocess

class Detector(object):
    def __init__(self):
        self._model = Model()

    def predict(self, img):
        return self._model.predict(img)

class LaptopCamera(object):
    def __init__(self, device_id, width=320, height=320):
        self._device_id = device_id
        self._device = cv2.VideoCapture(device_id)
        self._device.set(cv2.CAP_PROP_FRAME_WIDTH, width)
        self._device.set(cv2.CAP_PROP_FRAME_HEIGHT, height)

    def __del__(self):
        self._device.release()

    def get_frame(self):
        ret, frame = self._device.read()
        if ret:
            return frame
        else:
            print("No image received!")
            return None

class DetectorNode(object):
    def __init__(self, node_name, camera, detector, threshold, db_file, arduino_serial_port, rfid_serial_port):
        self._camera = camera
        self._detector = detector
        self._threshold = threshold
        self._db_file = db_file
        self._ser = serial.Serial(arduino_serial_port, 9600)
        self._rfid_ser = serial.Serial(rfid_serial_port, 9600)
        print("Serial connection established with Arduinos")


        self._conn = sqlite3.connect(db_file)
        self._cursor = self._conn.cursor()


        self._cursor.execute('''CREATE TABLE IF NOT EXISTS detected_objects
                             (id INTEGER PRIMARY KEY AUTOINCREMENT,
                              product TEXT NOT NULL,
                              category TEXT NOT NULL,
                              detection_time TEXT NOT NULL)''')
        self._conn.commit()

        self.user_interface_process = None

    def run(self):
        while True:
            self.check_rfid_commands()
            frame = self._camera.get_frame()
            if frame is None:
                continue
            predictions = self._detector.predict(frame)
            detected_objects = self.draw_bbox(frame, predictions)
            self.save_to_database(detected_objects)
            self.send_command_to_arduino(detected_objects)
            cv2.imshow("window", frame)
            if cv2.waitKey(10) & 0xFF == ord('q'):
                break

    def draw_bbox(self, img, p):
        detected_objects = []

        for label, bbox, probs in zip(*p):
            if probs < self._threshold:
                continue
            pt1 = (int(bbox[0]), int(bbox[1]))
            pt2 = (int(bbox[2]), int(bbox[3]))
            cv2.rectangle(img, pt1, pt2, (255, 0, 0), 3)
            cv2.putText(img, label, pt1, cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)
            if label != 'person':
                detected_objects.append({'Product': label, 'Detection Time': datetime.now().strftime("%Y-%m-%d %H:%M:%S")})

        return detected_objects
```

```
    def save_to_database(self, detected_objects):
        try:
            for obj in detected_objects:
                product = obj['Product']
                detection_time = obj['Detection Time']
                if product in ['knife', 'scissors']:
                    category = 'Dangerous Object'
                else:
                    category = 'Safe Object'


                self._cursor.execute("INSERT INTO detected_objects (product, category, detection_time) VALUES (?, ?, ?)",
                                     (product, category, detection_time))
                self._conn.commit()
        except Exception as e:
            print("Hata oluştu:", e)

    def send_command_to_arduino(self, detected_objects):
        try:
            for obj in detected_objects:
                product = obj['Product']
                if product in ['knife', 'scissors']:
                    self._ser.write(f'{product}\n'.encode())
                    time.sleep(2)
                    self._ser.write(b'0\n')
        except Exception as e:
            print("Hata oluştu:", e)

    def check_rfid_commands(self):
        try:
            if self._rfid_ser.in_waiting > 0:
                command = self._rfid_ser.readline().decode().strip()
                if command == "start_ui":
                    self.start_user_interface()
                elif command == "stop_ui":
                    self.stop_user_interface()
        except Exception as e:
            print("Hata oluştu:", e)

    def start_user_interface(self):
        if self.user_interface_process is None:
            self.user_interface_process = subprocess.Popen(["python", "C://Users//ekino//Desktop//Capstone//sunum//UserInterfaceBG.py"])
            print("UserInterface is started!")

    def stop_user_interface(self):
        if self.user_interface_process is not None:
            self.user_interface_process.terminate()
            self.user_interface_process = None
            print("UserInterface is closed!")

def main():
    camera = LaptopCamera(0)
    detector = Detector()
    db_file = 'detected_objects_sqlite3.db'
    node = DetectorNode("detector_node", camera, detector, 0.85, db_file, 'COM9', 'COM12')
    node.run()
    cv2.destroyAllWindows()

if __name__ == "__main__":
    main()
```

USER INTERFACE : The code in which the user responsible for security logs in and after logging in can see the list of objects classified according to the time, name and type of detection.

```python
import tkinter as tk
from tkinter import *
from tkinter import messagebox, filedialog, scrolledtext, font
import json
import re
import sqlite3
from PIL import Image, ImageTk

is_fullscreen = False
fullscreen_button = None
products_window = None
users_window = None
products_window_open = False
users_window_open = False
registered_users = {}
app = None
login_window = None
entry_username = None
entry_password = None
register_window = None



def close_login_window():
    global login_window
    login_window.withdraw()

def reopen_login_window():
    global login_window
    login_window.deiconify()

#kayıt olma
def register():
    def close_register_window():

        register_window.destroy()
        reopen_login_window()

    close_login_window()
    register_window = tk.Toplevel()
    register_window.title("Register")
    register_window.protocol("WM_DELETE_WINDOW", reopen_login_window)
    register_window.configure(bg="black")  # Arka plan rengini beyaz yap
    register_window.geometry("750x650")
    register_window.resizable(False, False)
    register_window.update_idletasks()
    x_coordinate = (register_window.winfo_screenwidth() - register_window.winfo_reqwidth()) / 2
    y_coordinate = (register_window.winfo_screenmmheight() - register_window.winfo_reqheight()) / 2
    register_window.geometry("+%d+%d" % (x_coordinate, y_coordinate))

    #Arka plan
    try:
        image = Image.open("BackPhoto//regw.jpg")
        image = image.resize((750, 650), Image.BICUBIC)
        background_image = ImageTk.PhotoImage(image)
        background_label = tk.Label(register_window, image=background_image)
        background_label.image = background_image
        background_label.place(x=0, y=0, relwidth=1, relheight=1)
    except Exception as e:
        print("Image upload error:", e)

    registration_frame = Frame(register_window, bg="black")
    registration_frame.place(relx=0.5, rely=0.5, anchor=CENTER)

    label_firstname = Label(registration_frame, text="                                                NAME:", bg="black", fg="white", font=("Helvetica",12,"bold"))
    label_firstname.grid(row=0, column=0, padx=5, pady=5)
    entry_firstname = Entry(registration_frame, font=("Helvetica", 12))
    entry_firstname.grid(row=0, column=1, padx=5, pady=5)

    label_lastname = Label(registration_frame, text="                                SURNAME:", bg="black", fg="white", font=("Helvetica", 12, "bold"))
    label_lastname.grid(row=1, column=0, padx=5, pady=5)
    entry_lastname = Entry(registration_frame, font=("Helvetica", 12))
    entry_lastname.grid(row=1, column=1, padx=5, pady=5)

    label_username = Label(registration_frame, text="                            USERNAME:", bg="black", fg="white", font=("Helvetica", 12, "bold"))
    label_username.grid(row=2, column=0, padx=5, pady=5)
    entry_username = Entry(registration_frame, font=("Helvetica", 12))
    entry_username.grid(row=2, column=1, padx=5, pady=5)

    label_password = Label(registration_frame, text="                            PASSWORD:", bg="black", fg="white", font=("Helvetica", 12, "bold"))
    label_password.grid(row=3, column=0, padx=5, pady=5)
    entry_password = Entry(registration_frame, show="*", font=("Helvetica", 12))
    entry_password.grid(row=3, column=1, padx=5, pady=5)

    label_password_confirm = Label(registration_frame, bg="black", fg="white", font=("Helvetica", 12, "bold"))
    label_password_confirm.grid(row=4, column=0, padx=5, pady=5)
    entry_repassword = Entry(registration_frame, show="*", font=("Helvetica", 12))
    entry_repassword.grid(row=4, column=1, padx=5, pady=5)

    label_confirm = Label(registration_frame, text ="(Re-enter the password you entered above!)", bg ="black", fg="white", font=("Helvetica", 10))
    label_confirm.grid(row=5, column=1, padx=5, pady=5)

    def toggle_password_visibility():
        current_show_state = entry_password["show"]
        if current_show_state == "*":
            entry_password.config(show="")
            show_password_button.config(image=hide_img, command=toggle_password_visibility)
            entry_repassword.config(show="")
        else:
            entry_password.config(show="*")
            show_password_button.config(image=show_img, command=toggle_password_visibility)
            entry_repassword.config(show="*")

    show_img = ImageTk.PhotoImage(Image.open("BackPhoto//opene.jpg").resize((30,30)))
    hide_img = ImageTk.PhotoImage(Image.open("BackPhoto//closede.png").resize((30,30)))
    show_password_button = Button(registration_frame, image=show_img, bg="white", bd=0, command=toggle_password_visibility)
    show_password_button.grid(row=3, column=2, padx=5, pady=5)


    label_birthdate = Label(registration_frame, text="BIRTH DATE (GG/AA/YYYY):", bg="black",  fg="white", font=("Helvetica", 12,"bold"))
    label_birthdate.grid(row=6, column=0, padx=5, pady=5)
    entry_birthdate = Entry(registration_frame, font=("Helvetica",12))
    entry_birthdate.grid(row=6, column=1, padx=5, pady=5)

    def add_slash(event):
        entry = event.widget
        value = entry.get()
        if len(value) == 2 or len(value) == 5:
            entry.insert(END, "/")

    entry_birthdate.bind("<KeyRelease>", add_slash)

```

```
    def save_registration_info():

        firstname = entry_firstname.get().capitalize().strip()
        lastname = entry_lastname.get().capitalize().strip()
        username = entry_username.get().strip()
        password = entry_password.get().strip()
        repassword = entry_repassword.get().strip()
        birthdate = entry_birthdate.get().strip()

        if password != repassword:
            messagebox.showerror("Error", "The passwords you enter must match in order for you to register!")
            return

        if not firstname or not lastname or not username or not password or not repassword or not birthdate:
            messagebox.showerror("Error", "Please fill in all information completely!")
            return

        pattern = r'^\d{2}/\d{2}/\d{4}$'
        if not re.match(pattern, birthdate):
            messagebox.showerror("Error", "The birth date format you entered is invalid. Please re-enter in DD/MM/YYYY format!")
            return

        day, month, year = map(int, birthdate.split("/"))
        if month < 1 or month > 12 or day < 1 or day > 31 or year < 0 or year > 9999:
            messagebox.showerror("Error", "The date of birth you entered contains invalid values!")
            return

        if not username or not password:
            messagebox.showerror("Warning!", "Please enter username and password!")
        elif username in registered_users:
            messagebox.showerror("Error" , "This user is already registered in the system!")
        else:
            registered_users[username] = password
            save_registered_users()
            messagebox.showinfo("Successful!", "Registration completed successfully")

        register_window.withdraw()
        reopen_login_window()

    button_save = Button(registration_frame, text="Save", command=save_registration_info, bg="gray", fg="black")
    button_save.grid(row=7, column=1, padx=5, pady=10)

    button_back = Button(registration_frame, text="Back", command=close_register_window, bg="gray", fg="black")
    button_back.grid(row=7, column=0, padx=5, pady=10)

#giriş yapma

def login():
    global entry_username, entry_password
    username = entry_username.get()
    password = entry_password.get()

    if username in registered_users and registered_users[username] == password:
        messagebox.showinfo("Your login request is successful" , f"Welcome, {username}")
        show_main_page()
        close_login_window()
    else:
        messagebox.showerror("Your login request is negative", "Invalid username or password!")

def show_main_page():

    global is_fullscreen, fullscreen_button,app

    app.withdraw()
    main_page =tk.Toplevel()
    main_page.title("Main Screen")
    main_page.geometry("640x412")
    main_page.resizable(False,False)

    try:
        image = Image.open("BackPhoto//yan.png")
        image = image.resize((640,412), Image.BICUBIC)
        background_image = ImageTk.PhotoImage(image)
        background_label = tk.Label(main_page, image = background_image)
        background_label.image = background_image
        background_label.pack(fill="both", expand=True)
    except Exception as e:
        print("Image upload error:", e)

    def go_back_to_login():
        main_page.destroy()
        login_window.deiconify()

    button_go_back = tk.Button(main_page, text="Back", command=go_back_to_login, bg="gray", fg="black")
    button_go_back.pack(side=tk.BOTTOM, pady=10)

    def show_products():
        global products_window_open, products_window
        if not products_window_open or not products_window.winfo_exists():
            products_window = tk.Toplevel()
            products_window.title("Products")
            products_window_open = True

            x = main_page.winfo_rootx() + main_page.winfo_width()
            y = main_page.winfo_rooty()

            products_window.geometry(f"500x400+{x}+{y}")
            products_window.resizable(False, False)

            try:
                image = Image.open("BackPhoto//scam.png")
                image = image.resize((500,400), Image.BICUBIC)
                background_image = ImageTk.PhotoImage(image)
                background_label = tk.Label(products_window, image=background_image)
                background_label.image = background_image
                background_label.place(x=0, y=0, relwidth=1, relheight=1)
            except Exception as e:
                print("Image upload error:", e)


            def load_saved_products_from_db():
                db_file = "detected_objects_sqlite3.db"
                try:
                    conn = sqlite3.connect(db_file)
                    cursor = conn.cursor()
                    cursor.execute("SELECT product, category, detection_time FROM detected_objects")
                    products = cursor.fetchall()
                    for product in products:
                        product_name = product[0]
                        category = product[1]
                        detection_time = product[2]
                        text_editor.insert(tk.END, f"Product:", "bold")
                        text_editor.insert(tk.END, f"{product_name}\n", "normal")
                        text_editor.insert(tk.END, f"Category:", "bold")
                        text_editor.insert(tk.END, f"{category}\n", "normal")
                        text_editor.insert(tk.END, f"Detection Time:", "bold")
                        text_editor.insert(tk.END, f"{detection_time}\n\n", "normal")
                    conn.close()
                except sqlite3.Error as e:
                    messagebox.showerror("Error", f"SQLite error: {e}")
```

```
            text_editor = scrolledtext.ScrolledText(products_window, wrap=tk.WORD, width = 43, height=50)
            text_editor.pack(pady=5)

            bold_font = font.Font(text_editor, text_editor.cget("font"))
            bold_font.configure(weight = "bold")
            text_editor.tag_configure("bold", font = bold_font)
            text_editor.tag_configure("normal", font=("Arial", 10))

            load_saved_products_from_db()

        else:
            products_window.lift()


    def show_registered_users():
        global users_window_open , users_window
        if not users_window_open:
            users_window = tk.Toplevel()
            users_window.title("Registered Users")
            users_window_open = True
            button_show_registered_users.update()
            x = 0
            y = 0
            users_window.geometry(f"433x500+{x}+{y}")
            users_window.resizable(False,False)
            background_frame = tk.Frame(users_window)
            background_frame.pack(fill="both", expand=True)

            try:
                image = Image.open("BackPhoto//person.png")
                image = image.resize((500,433), Image.BICUBIC)
                background_image = ImageTk.PhotoImage(image)
                background_label = tk.Label(background_frame, image=background_image)
                background_label.image = background_image
                background_label.place(x=0, y=0, relwidth=1, relheight=1)
            except Exception as e:
                print("Image upload error:", e)

            #Kayıtlı kullanıcıları listeleme
            label_user_list = tk.Label(users_window, text="Registered Users:")
            label_user_list.pack(pady=10)

            listbox_users = tk.Listbox(users_window)
            listbox_users.pack()

            for username in registered_users:
                listbox_users.insert(tk.END, username)
            #Kullanıcı silme özelliği
            def delete_user():
                selected_index = listbox_users.curselection()
                if selected_index:
                    selected_username = listbox_users.get(selected_index)
                    del registered_users[selected_username]
                    listbox_users.delete(selected_index)
                    save_registered_users()
                    messagebox.showinfo("Successful!" ,f"{selected_username} user has been successfully deleted from the system")
                else:
                    messagebox.showerror("Error", "Please select a user!")

            def toggle_delete_button_visibility():
                if entry_username.get() == "admin":
                    button_delete_user.pack(pady=5)
                else:
                    button_delete_user.pack_forget()

            button_delete_user = tk.Button(users_window, text="Delete selected user" , command=delete_user)

            def on_users_window_close():
                global users_window_open
                users_window_open = False
                users_window.destroy()
                button_delete_user.destroy()

            users_window.protocol("WM_DELETE_WINDOW", on_users_window_close)
            toggle_delete_button_visibility()
            users_window_open = True
        else:
            users_window.lift()

    button_show_products = tk.Button(main_page, text="View Products", command=show_products, bg="gray", fg="black")
    button_show_products.place(relx=0.75, rely=0.65, anchor=tk.CENTER)
    button_show_registered_users = tk.Button(main_page, text="View Registered Users", command=show_registered_users, bg="gray",fg="black")
    button_show_registered_users.place(relx=0.63, rely=0.967, anchor=tk.CENTER)
    button_show_registered_users.bind("<Button-1>", lambda event: toggle_delete_button_visibility())

    def toggle_delete_button_visibility():
        if entry_username.get()== "admin":
            button_delete_user.pack(pady=5)
        else:
            button_delete_user.pack_forget()


    #ürünler doğru bir şekilde işaretlenip ana sayfa kapandıktan sonra giriş ekranını kapatırken sorulan soru
    def on_closing_app():
        if messagebox.askokcancel("Exit" ,"Are you sure you want to exit the application?"):
                    app.destroy()

    app.protocol("WM_DELETE_WINDOW", on_closing_app)

#kayıt olmuş kullanıcıların verilerinin tutulması
def save_registered_users():
    with open("registered_users.json","w") as file:
        json.dump(registered_users, file)

def load_registered_users():
    try:
        with open("registered_users.json", "r") as file:
            return json.load(file)
    except FileNotFoundError:
        return {}


registered_users = load_registered_users()
```

```python
def create_login_window():
    global login_window, entry_username, entry_password, app

    login_window = tk.Tk()
    login_window.title("Login Screen")
    login_window.geometry("750x650")  # Çerçevenin boyutunu değiştirdik
    login_window.resizable(False, False)

    # Arka plan resmi Canvas'i oluştur
    background_canvas = tk.Canvas(login_window, bg="white", width=600, height=400)  # Canvas boyutunu da değiştirdik
    background_canvas.pack(fill="both", expand=True)

    # Arka plan resmini yerleştir
    image = Image.open("BackPhoto//seclogo.jpg")
    background_image = ImageTk.PhotoImage(image)
    background_canvas.create_image(0, 0, anchor="nw", image=background_image)

    # Ana çerçeve oluştur
    main_frame = tk.Frame(background_canvas, bg="black", width=600, height=400)  # Çerçevenin boyutunu da değiştirdik
    main_frame.place(relx=0.5, rely=0.8, anchor="center")  # Çerçevenin konumunu alt kısma doğru taşıdık

    # Yazıları ortala ve büyüt
    font_style = ("Helvetica", 14)  # Yazı tipi ve boyutu
    label_username = tk.Label(main_frame, text="USER NAME:", bg="black", fg="white", font=font_style)
    label_username.grid(row=0, column=0, padx=10, pady=5, sticky="w")
    entry_username = tk.Entry(main_frame, font=font_style)
    entry_username.grid(row=0, column=1, padx=10, pady=5)

    label_password = tk.Label(main_frame, text="PASSWORD:", bg="black", fg="white", font=font_style)
    label_password.grid(row=1, column=0, padx=10, pady=5, sticky="w")
    entry_password = tk.Entry(main_frame, show="*", font=font_style)
    entry_password.grid(row=1, column=1, padx=10, pady=5)

    # Kayıt ol butonu oluşturulması
    button_register = tk.Button(main_frame, text="REGISTER", command=register, bg="gray", fg="black", font=("Helvetica",11))
    button_register.grid(row=2, column=0, columnspan=2, pady=10)

    # Giriş yap butonu oluşturulması
    button_login = tk.Button(main_frame, text="LOGIN", command=login, bg="gray", fg="black", font=("Helvetica",11))
    button_login.grid(row=3, column=0, columnspan=2, pady=10)

    app = login_window
    login_window.mainloop()


create_login_window()
```

ARDUINO UNO COM9: When an object classified as dangerous is detected, the LED and buzzer work and the name of the object is printed on the LCD screen.

buzzer_led_lcd.ino

```ino
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

const int ledPin = 7;
const int buzzerPin = 6;
LiquidCrystal_I2C lcd(0x27, 16, 2);

String receivedString = "";

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(buzzerPin, OUTPUT);
  Serial.begin(9600);
  lcd.begin();
  lcd.backlight();
  lcd.clear();
}

void loop() {
  if (Serial.available()) {
    char receivedChar = Serial.read();
    if (receivedChar == '\n') {
      if (receivedString != "0") {
        digitalWrite(ledPin, HIGH);
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Tehlikeli Nesne:");
        lcd.setCursor(0, 1);
        lcd.print(receivedString);

        for (int i = 0; i < 3; i++) {
          digitalWrite(buzzerPin, HIGH);
          delay(1500);
          digitalWrite(buzzerPin, LOW);
          delay(500);
        }
        digitalWrite(ledPin, LOW);
        lcd.clear();
      }
      receivedString = "";
    } else {
      receivedString += receivedChar;
    }
  }
}

```

ARDUINO UNO COM12:The code that turns on and off the User Interface code and turns on the LEDs according to the RFID card scanned.

rfidtest.ino

```ino
#include <SPI.h>
#include <MFRC522.h>

#define SS_PIN 10
#define RST_PIN 9

// LED pin tanımlamaları
#define BLUE_LED_PIN 2
#define YELLOW_LED_PIN 3
#define WHITE_LED_PIN 4

MFRC522 mfrc522(SS_PIN, RST_PIN);

void setup() {
  Serial.begin(9600);
  SPI.begin();
  mfrc522.PCD_Init();
  Serial.println("RFID modülü hazır.");

  // LED pin modlarını ayarla
  pinMode(BLUE_LED_PIN, OUTPUT);
  pinMode(YELLOW_LED_PIN, OUTPUT);
  pinMode(WHITE_LED_PIN, OUTPUT);

  // Başlangıçta beyaz LED'i yak
  digitalWrite(BLUE_LED_PIN, LOW);
  digitalWrite(YELLOW_LED_PIN, LOW);
  digitalWrite(WHITE_LED_PIN, HIGH);
}

void loop() {
  // Seriden gelen veri kontrolü
  if (Serial.available() > 0) {
    String command = Serial.readStringUntil('\n');
    command.trim();
    if (command == "close") {
      // Tüm LED'leri kapat
      digitalWrite(BLUE_LED_PIN, LOW);
      digitalWrite(YELLOW_LED_PIN, LOW);
      digitalWrite(WHITE_LED_PIN, LOW);
      return;
    }
  }

  if (!mfrc522.PICC_IsNewCardPresent()) {
    return;
  }
  if (!mfrc522.PICC_ReadCardSerial()) {
    return;
  }

  String uid = "";
  for (byte i = 0; i < mfrc522.uid.size; i++) {
    uid += String(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
    uid += String(mfrc522.uid.uidByte[i], HEX);
  }
  uid.toUpperCase();
  uid.trim();
  Serial.println(uid);

  if (uid == "13 35 D6 83") {
    Serial.println("start_ui");
    digitalWrite(BLUE_LED_PIN, HIGH);
    digitalWrite(YELLOW_LED_PIN, LOW);
    digitalWrite(WHITE_LED_PIN, LOW);
  } else if (uid == "D2 06 EB 1B") {
    Serial.println("stop_ui");
    digitalWrite(BLUE_LED_PIN, LOW);
    digitalWrite(YELLOW_LED_PIN, HIGH);
    digitalWrite(WHITE_LED_PIN, LOW);
  } else {
    digitalWrite(BLUE_LED_PIN, LOW);
    digitalWrite(YELLOW_LED_PIN, LOW);
    digitalWrite(WHITE_LED_PIN, HIGH);
  }

  mfrc522.PICC_HaltA();
}
```