Vietnam National University Ho Chi Minh City

Ho Chi Minh City University of Technology

Department of Electronics

□···☼···□

GRADUATION INTERNSHIP REPORT

DUY ANH SYSTEM MANAGEMENT SOLUTION CO., LTD

ESP32 CAM PRODUCT RECOGNITION AND COUNTING SYSTEM USING SQL DATABASE AND DEVEXPRESS UI

Student

Nguyễn Tiến Phát - 2051165

Instructor

Associate Pf. TRƯƠNG QUANG VINH

Ho Chi Minh city, August 2024

THANKS

To complete this internship report, I would like to express my sincere thanks to Associate Pf. Truong Quang Vinh and Mr. Minh Quoc, who have guided and instructed me a lot during the process of making the report. My deepest thanks.

I would like to sincerely thank Duy Anh System Management Solutions Company for creating the best conditions for me to practice and learn practical knowledge during my internship at the company. At the same time, I would like to express my sincere thanks to the school for creating opportunities for me to practice at the company and helping me gain more practical experience to apply the theoretical knowledge taught by the teachers.

Because my own knowledge is still limited, the internship report may still have errors. I look forward to receiving comments from the teachers.

TABLE OF CONTENTS

TABLE OF FIGURES

1.INTRODUCTION

1.1 INTRODUCE ABOUT THE COMPANY

Duy Anh System Management Solutions Co.Ltd provides solutions to effectively manage businesses and manufacturing plants with the ERP (Enterprise Resource Planning) model. Starting in 2017, we built an ERP management system based on the IoT (Internet of Things) platform to help small and medium-sized businesses manage their operations effectively.

1.2 INTERNSHIP DURATION AND SCHEDULE

- Internship duratino: 2 months (from 17/06/2024 to 30/08/2024)
- Schedule:
    - Week 1:Devexpress UI
    - Week 2:Devexpress UI
    - Week 3:Research about SQL Server
    - Week 4:Python Programming for product recognition and counting system
    - Week 5: Python Programming for product recognition and counting system
    - Week 6:Simulation
    - Week 7:Simulation and debug
    - Week 8:Report and finish project

2.INTERNSHIP CONTENT

This report outlines the development and implementation of an automated attendance tracking system using the ESP32-CAM module, an SQL database, and DevExpress XAF XPO with .NET 6.0+. The goal of the project is to identify, classify products and count the number of products, thereby helping people easily manage product data.

The ESP32-CAM, a low-cost microcontroller with integrated camera capabilities, serves as the primary device for detect and capturing images of product as they enter a designated area. Utilizing YOLO algorithms, the system identifies and verifies each type of product, logging their amount and type in real-time. The captured data is then transmitted to an SQL database, ensuring secure and organized storage.

To visualize and manage the Product recognition and counting system, the project leverages DevExpress XAF XPO, a robust application framework for .NET. This framework enables the creation of a feature-rich, cross-platform application that facilitates the display, querying, and reporting of product data. By integrating these technologies, the system offers a comprehensive solution that enhances the accuracy, reliability, and accessibility of product recognition and counting system tracking.

Throughout this report, we detail the system architecture, including hardware and software components, implementation procedures, and the challenges encountered. Additionally, we discuss the potential benefits and future enhancements of the system, highlighting its scalability and adaptability to various environments. This project exemplifies the integration of IoT devices with modern software frameworks to create innovative and practical solutions for everyday problems.

2.1 OVERVIEW OF DEVEXPRESS XAF AND XPO

*DevExpress XAF (eXpressApp Framework) is a versatile and comprehensive application framework designed for .NET developers. It enables the rapid development of business applications by providing a robust foundation that simplifies common development tasks. XAF streamlines the creation of cross-platform applications, allowing developers to focus on business logic rather than repetitive boilerplate code.

- Modular architecture: XAF employs a mudular architecture that promotes reusability and separation of concerns. Developers can create modules that encapsulate specific functionality, which can then be reused across different projects.
- Cross-Platform Support: XAF supports the development of applications for Windows (WinForms and WPF) and Web (ASP.NET WebForms, MVC, and Blazor). This ensures that applications can reach a wide audience across different platforms.
- Itergrated Security System: XAF includes a comprehensive security system that handles authentication and authorization. It provides built-in support for role-based access control, ensuring that applications are secure and user permissions are managed effectively.
- Data Visualization and Reporting: XAF integrates seamlessly with DevExpress reporting and dashboard tools, enabling the creation of rich, interactive reports and dashboards that provide valuable insights into business data.
- Rich UI: XAF leverages DevExpress UI components to deliver a rich user experience. These components are highly customizable, ensuring that the application's interface is both functional and visually appealing.
- Model-Driven Development: XAF promotes a model-driven development approach, allowing developers to define application structure and behavior

through a metadata model. This approach reduces the amount of code needed and enhances maintainability.

*DevExpress XPO (eXpress Persistent Objects) is an ORM (Object-Relational Mapping) tool provided by DevExpress, a company known for its UI components and frameworks for .NET developers. XPO simplifies database access by mapping database tables to .NET classes, allowing developers to interact with databases using high-level, object-oriented code rather than writing raw SQL queries. Some key features are:

- Database-agnostic: XPO supports multiple database engines, including SQL Server, MySQL, Oracle, PostgreSQL, SQLite, and more. This flexibility allows developers to switch databases with minimal code changes.
- LINQ Support: XPO supports Language Integrated Query (LINQ), enabling developers to write database queries using familiar .NET syntax. This integration simplifies querying and enhances code readability.
- Automatic Schema Synchronization: XPO can automatically synchronize the database schema with the data model, reducing the need for manual database management tasks and ensuring consistency between the model and the database.
- Transactions and Concurrency: XPO provides robust support for transactions and concurrency, ensuring data integrity and consistency in multi-user environments.
- Data Binding: XPO integrates seamlessly with data-bound controls in .NET, making it easy to bind data to UI components for display and interaction.
- Flexible Mapping: XPO supports various mapping scenarios, including one-to-one, one-to-many, and many-to-many relationships. This flexibility allows developers to model complex data structures effectively.

Using XPO also give us some benefits:

- Simplified Data Access: XPO abstracts the complexities of database interactions, allowing developers to focus on business logic.
- Enhanced Productivity: By reducing boilerplate code and automating common tasks, XPO enhances developer productivity.
- Improved Maintainability: The use of a consistent data access layer improves the maintainability of the codebase.
- Seamless Integration: XPO integrates seamlessly with XAF, providing a cohesive solution for building data-driven applications.

2.2 ESP32 CAM

The ESP32-CAM is a very small camera module with the ESP32-S chip that costs approximately $10. Besides the OV2640 camera, and several GPIOs to connect peripherals, it also features a microSD card slot that can be useful to store images taken with the camera or to store files to serve to clients. The ESP32-CAM does not come with a USB connector, so it needs a FTDI programmer to upload code through the U0R and U0T pins (serial pins).

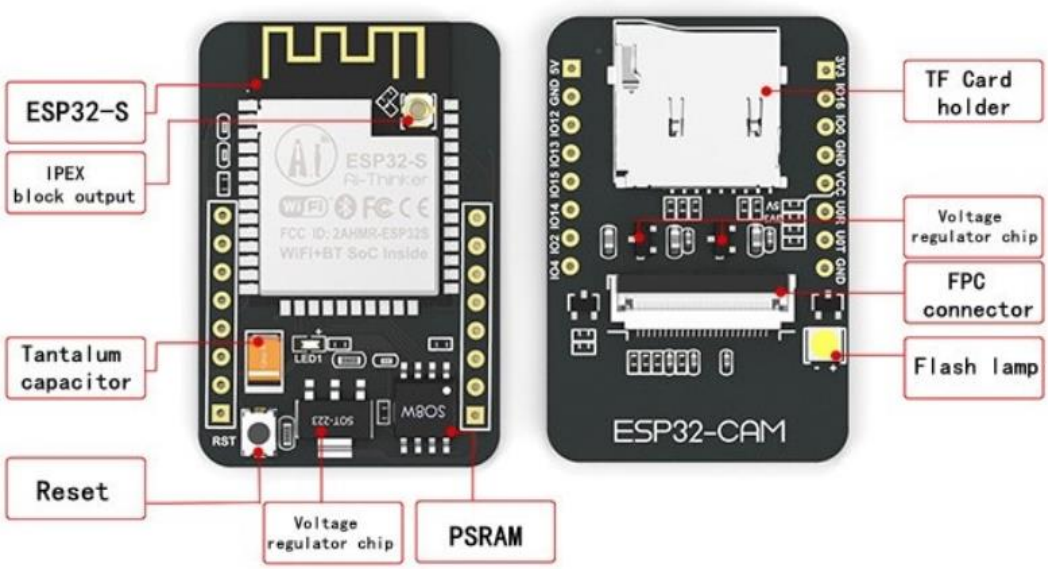


Figure 1. ESP32-CAMERA

The following figure shows the ESP32-CAM pinout (AI-Thinker module).

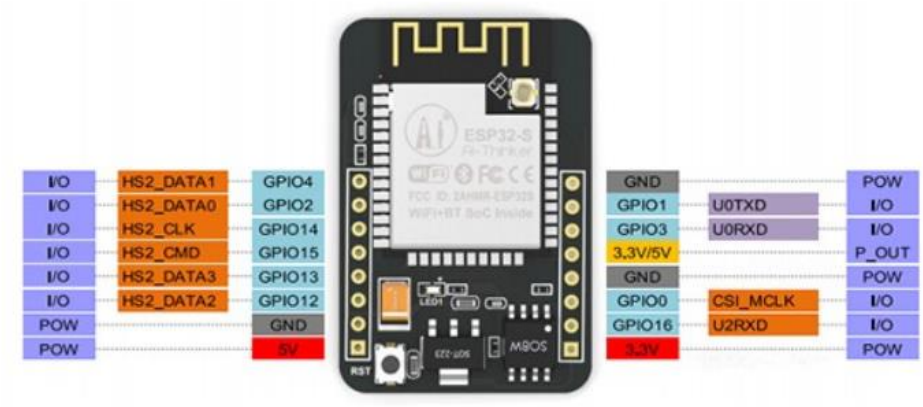


Figure 2 ESP32 pins layout

The ESP32-CAM provides an inexpensive way to build more advanced home automation projects that feature video, taking photos, and face recognition.

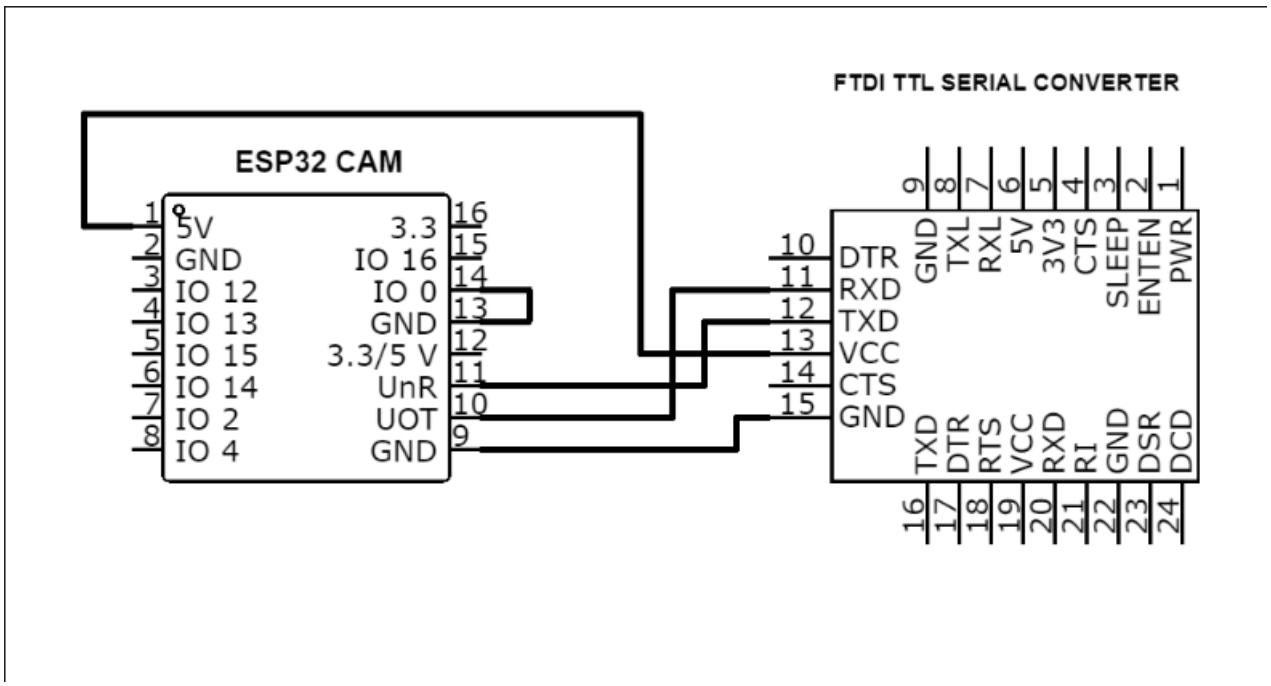


Figure 3 ESP32-Camera and FTDI serial converter

The ESP32-CAM module is based on ESP32 which is a low-power and small-size camera module. This module comes with an OV2640 camera & provides an onboard TF card slot. The 4MB PSRAM of this board is used for image buffering from the camera into video streaming and permits you to utilize higher quality

within your images without any crash of the ESP32. This board has an onboard LED to flash & numerous GPIOs for connecting peripherals.

ESP32 Cam includes two power pins; 5V & 3V3, so this module is powered through the 3.3V/5V pins. The VCC pin of this module usually outputs 3.3Volts from the on-board voltage regulator but it can be configured to 5V output with the Zero-ohm link close to the VCC pin. GND is the ground pin of the module.

- GPIO Pins: This module includes 32 GPIO pins in totality but many of them are internally used for the camera & the PSRAM. So this module has only 10 GPIO pins which can be assigned for different peripheral duties like; ADC, SPI, UART, and Touch.
- UART Pins: The ESP32-S chip includes two UART interfaces; UART0 & UART2. But simply the RX pin of UART2 like GPIO 16 is broken out to make UART0 usable UART only on the ESP32-CAM. These pins are used mainly for flashing & connecting to different UART devices like fingerprint sensors, GPS, distance sensors, etc.
- MicroSD Card Pins: These pins are used to interface the microSD card. These pins can also be used as regular inputs & outputs if you do not use a microSD card.
- ADC Pins: There are two ADC2 pins only on the ESP32-CAM which are broken out but these pins are internally used by the WiFi driver and they cannot be used whenever Wi-Fi is enabled.
- Touch Pins: These modules include seven capacitive touch-sensing GPIOs. Once a capacitive load like a human finger is close to the GPIO, then this module notices the change within capacitance.
- SPI Pins: The ESP32-CAM includes one SPI (VSPI) only in slave & master modes.
- PWM Pins: The ESP32-CAM includes ten channels of PWM pins which are controlled through a PWM controller. So the output of PWM can be

used to drive LEDs & digital motors.

2.3 SQL SERVER

SQL Server is a relational database management system (RDBMS) developed by Microsoft. It is designed to handle and store large amounts of data, allowing users to manage and retrieve that data efficiently. Here are some key features and components of SQL Server:

- Relational Database Management System (RDBMS): SQL Server uses a relational model to store data in tables with rows and columns. It allows for complex queries and transactions on the data.
- Transact-SQL (T-SQL): SQL Server uses T-SQL, an extension of SQL (Structured Query Language), to interact with the database. T-SQL includes procedural programming features such as variables, loops, and error handling.
- Security: SQL Server provides robust security features, including authentication, authorization, and encryption, to protect data.
- Scalability and Performance: SQL Server is designed to handle large volumes of data and high levels of concurrent access. It includes features like indexing, partitioning, and in-memory processing to optimize performance.
- Integration Services (SSIS): A tool for data integration and workflow applications. SSIS can extract, transform, and load (ETL) data from various sources.
- Reporting Services (SSRS): A tool for creating, deploying, and managing reports. SSRS allows users to create detailed reports from SQL Server data.
- Analysis Services (SSAS): A tool for online analytical processing (OLAP) and data mining. SSAS can be used to analyze data and create complex analytical models.
- Management Tools: SQL Server includes tools like SQL Server

Management Studio (SSMS) for database administration and development, and SQL Server Profiler for performance monitoring.

- High Availability and Disaster Recovery: SQL Server offers features like Always On Availability Groups, failover clustering, and backup/restore options to ensure data availability and recoverability.

SQL Server is used by organizations of all sizes for various applications, including web applications, enterprise resource planning (ERP) systems, customer relationship management (CRM) systems, and data warehousing.

2.4 ESP32 CAM PRODUCT RECOGNITION AND COUNTING SYSTEM

1. System Overview

The system consists of two main components:

1. **ESP32-CAM Module**: This microcontroller handles the image capture and serves the images over a network.
2. **PC with Python Script**: The Python script performs object detection on the captured images and manages the data storage in an SQL server.

2. System Components

A. ESP32-CAM

- **Camera Module**: Captures images at different resolutions.
- **Wi-Fi Module**: Connects the ESP32-CAM to a Wi-Fi network.
- **Web Server**: Serves the captured images to the Python script on request.
- **Microcontroller**: Executes the main loop to handle client requests, change camera resolution, capture images, and serve them.

B. PC (Python Environment)

- **Image Processing (OpenCV)**: Processes the images received from the ESP32-CAM to detect objects.
- **Object Detection Model (Deep Learning)**: Utilizes a pre-trained model like MobileNet SSD for object detection.
- **SQL Server Connection (PyODBC)**: Handles database interactions to store the detected object data.

- **Graphical User Interface (OpenCV Window)**: Displays the live feed from the ESP32-CAM and processed images.
- **File System**: Optionally stores captured images.

C. SQL Server

- **Database**: Stores information about detected objects, including the timestamp, object names, and their counts.

3. Workflow

A. ESP32-CAM Setup and Operation

1. **Initialization**:
    - The ESP32-CAM initializes the camera module and configures Wi-Fi to connect to a local network.
    - The device sets up a web server that listens for incoming requests to serve images.
2. **Handling Requests**:
    - The web server handles requests from the Python script. Depending on the request, the camera captures an image at a specified resolution (low, medium, or high).
    - The captured image is served back to the Python script via the network.

B. Python Script Operation

1. **Initialization**:

- The Python script initializes by setting up the SQL server connection, loading class names, and configuring the object detection model.

2. **Image Retrieval**:
    - The script periodically requests images from the ESP32-CAM via HTTP.
    - The image is retrieved and decoded for further processing.
3. **Object Detection**:
    - The image is processed using the MobileNet SSD model to detect objects.
    - Bounding boxes and labels are drawn on the image to visualize detected objects.
4. **Data Management**:
    - If an image is captured for analysis (triggered by pressing a specific key), the Python script logs the detected objects' data into the SQL server, including the timestamp, object names, and counts.
    - Optionally, the image may be saved to the local file system.
5. **User Interaction**:
    - The user can interact with the script via key presses:
        - **H Key**: Captures and processes an image, then saves it and logs the data.
        - **ESC Key**: Terminates the program.
6. **Shutdown**:

- The script closes the OpenCV windows and the database connection before exiting.

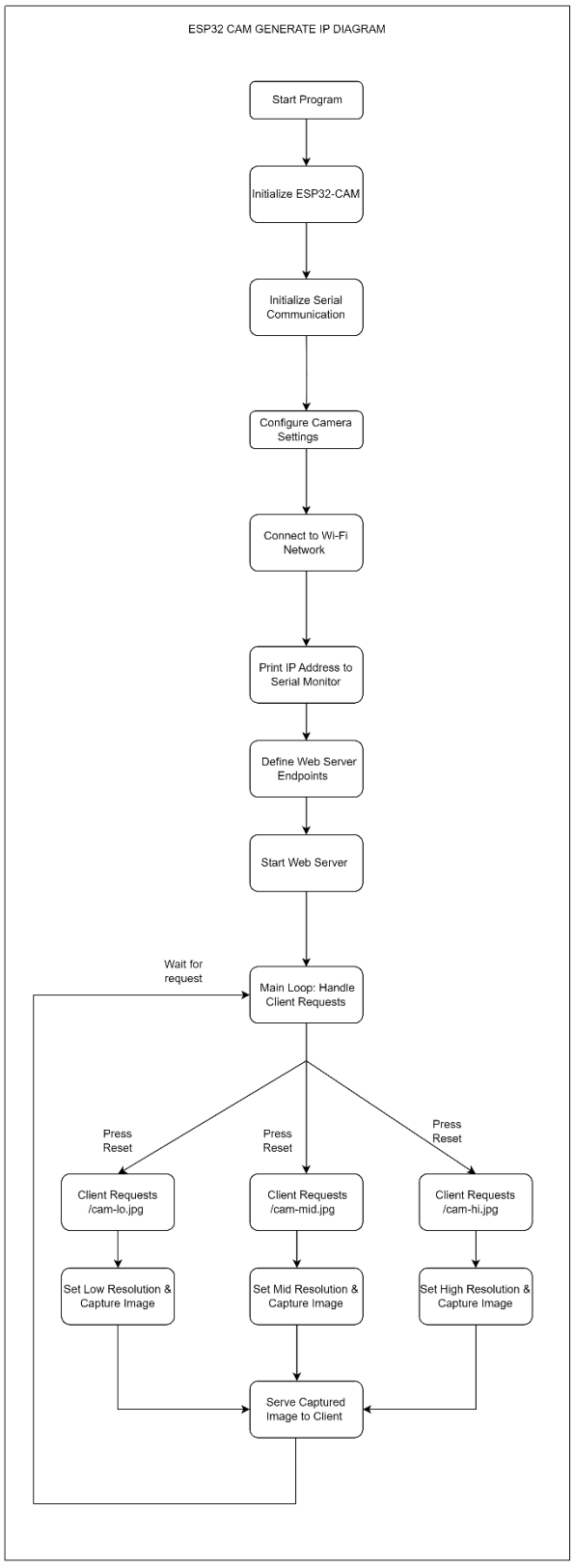
4. System Interactions

1. **ESP32-CAM to Python Script**:
   - Image data is served over HTTP to the Python script running on the PC.
2. **Python Script to SQL Server**:
   - Detected object data is written to an SQL database.
3. **User to Python Script**:
   - The user interacts via keyboard inputs to control image capture and program termination.

5. System Diagram

The diagram represents the flow of data and control signals across the system.

Plaintext


ESP32 CAM GENERATE IP DIAGRAM
Start Program
Initialize ESP32-CAM
Initialize Serial Communication
Configure Camera Settings
Connect to Wi-Fi Network
Print IP Address to Serial Monitor
Define Web Server Endpoints
Start Web Server
Wait for request
Main Loop: Handle Client Requests
Press Reset
Press Reset
Press Reset
Client Requests /cam-lo.jpg
Client Requests /cam-mid.jpg
Client Requests /cam-hi.jpg
Set Low Resolution & Capture Image
Set Mid Resolution & Capture Image
Set High Resolution & Capture Image
Serve Captured Image to Client

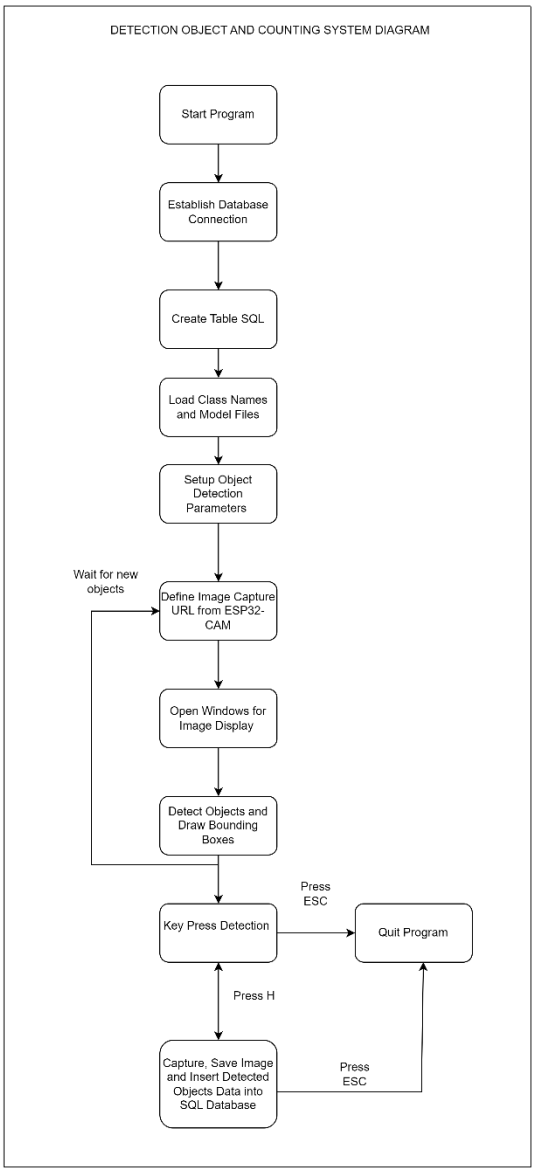

DETECTION OBJECT AND COUNTING SYSTEM DIAGRAM
Start Program
Establish Database Connection
Create Table SQL
Load Class Names and Model Files
Setup Object Detection Parameters
Wait for new objects
Define Image Capture URL from ESP32-CAM
Open Windows for Image Display
Detect Objects and Draw Bounding Boxes
Press ESC
Key Press Detection
Quit Program
Press H
Capture, Save Image and Insert Detected Objects Data into SQL Database
Press ESC

Figure 4

3.CONCLUSION AND RESULT

*After 2 months of internship at DUY ANH SYSTEM MANAGEMENT SOLUTION CO., LTD, I have improved myself and my work performance, learned many new technologies, tools, and new social skills. And especially, it has trained me to be more dedicated to the products I make, more responsible to society, and this is a truly useful asset for me in my future career as a Hardware Engineer. Regarding the project, the project has been completed and met all the requirements of the topic, although there are still some items that have not been achieved as expected, but Mr. Vinh and Mr. Quoc have always supported me enthusiastically. Thanks to that support, I was able to complete the assigned projects on time and achieve good quality.

*Below is some figures when I simulate my project:

Figure 5

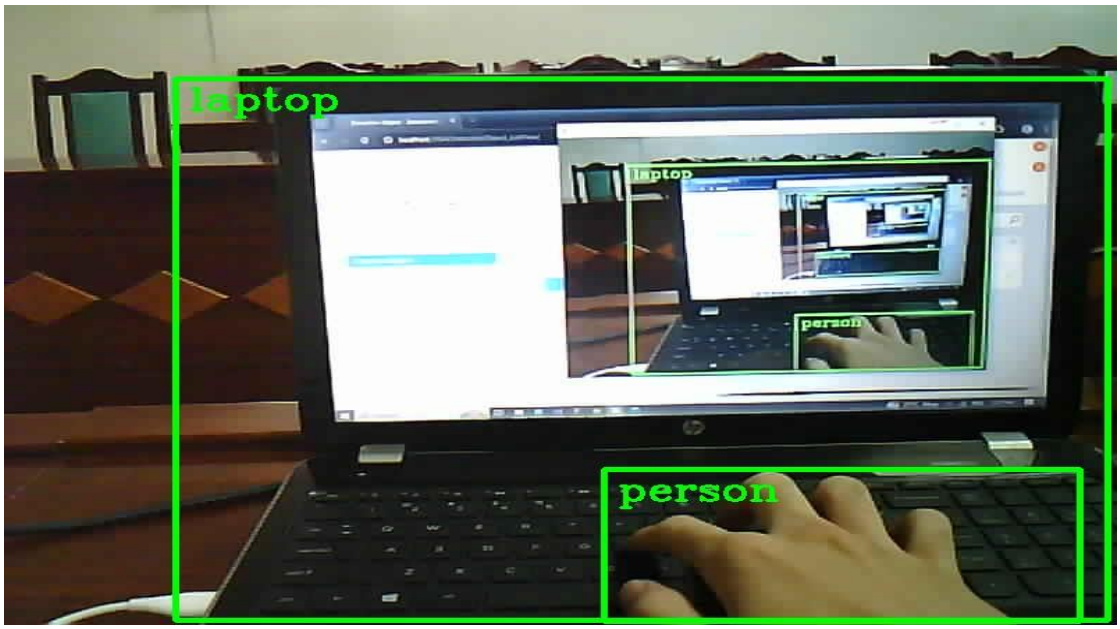


Figure 6

Figure 7

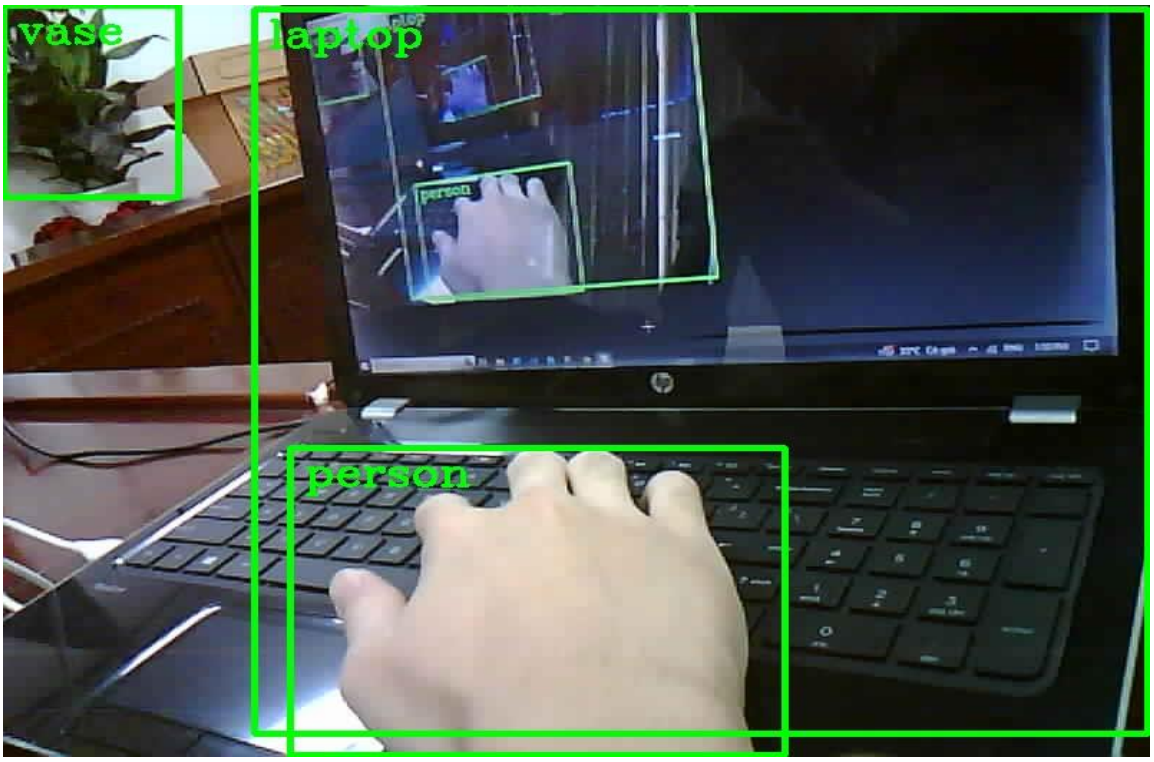


Figure 8

Figure 9

Figure 10

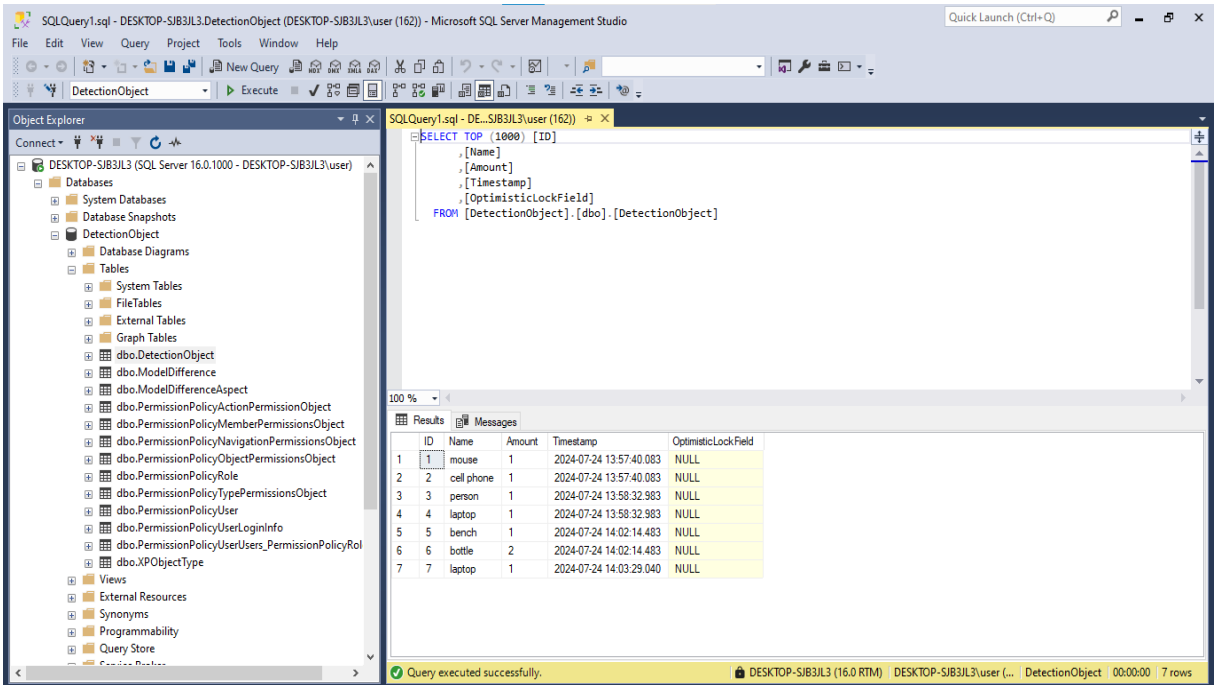


Figure 11

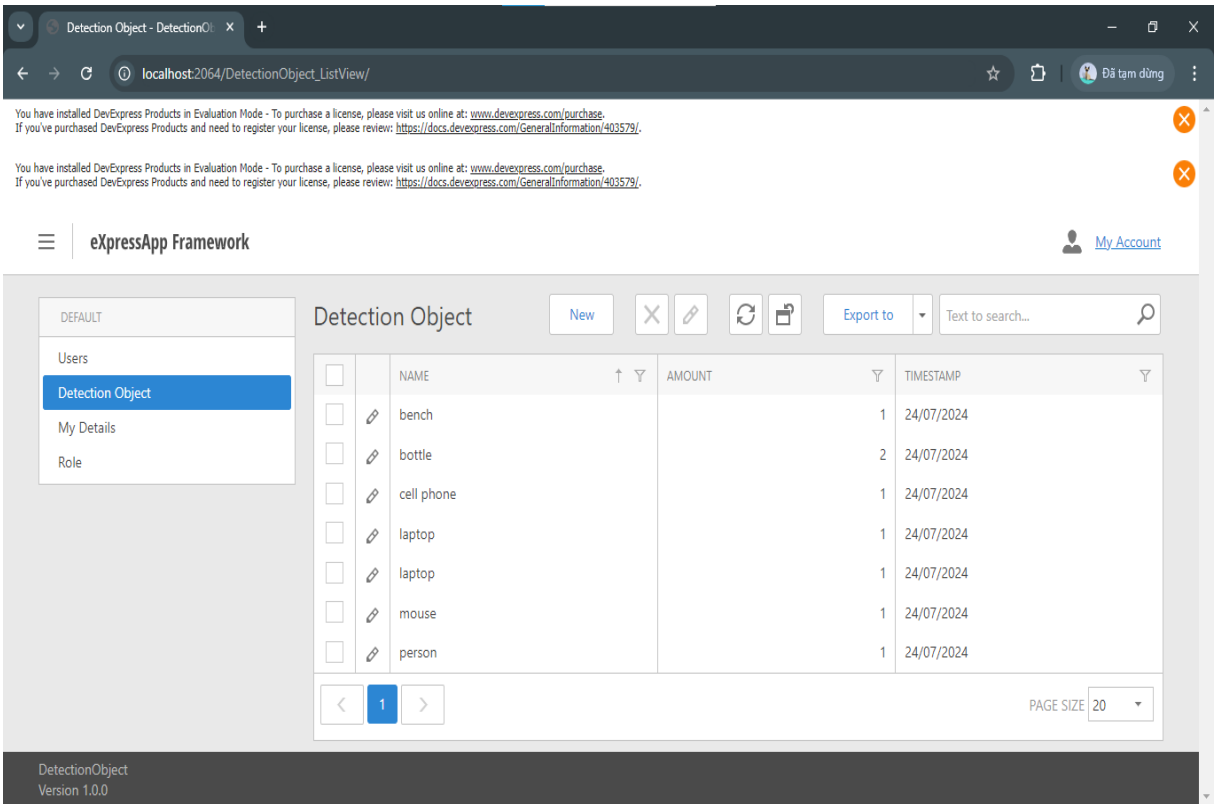


Figure 12

Figure 13

4.REFERENCE

1)https://how2electronics.com/esp32-cam-based-object-detection-identification-with-opencv/

2) https://forum.arduino.cc/t/recognition-of-objects/1208767

3) https://lastminuteengineers.com/getting-started-with-esp32-cam/

4) https://docs.devexpress.com/eXpressAppFramework/112670/expressapp-framework

5.APPENDIX

***Arduino Code for generate IP address for ESP32 CAM**

```
#include <WebServer.h>
#include <WiFi.h>
#include <esp32cam.h>
const char* WIFI_SSID = "Laptop1";
const char* WIFI_PASS = "123456789";
WebServer server(80);
static auto loRes = esp32cam::Resolution::find(320, 240);
static auto midRes = esp32cam::Resolution::find(350, 530);
static auto hiRes = esp32cam::Resolution::find(800, 600);
void serveJpg()
{
  auto frame = esp32cam::capture();
  if (frame == nullptr) {
    Serial.println("CAPTURE FAIL");
    server.send(503, "", "");
    return;
  }
  Serial.printf("CAPTURE OK %dx%d %db\n", frame->getWidth(), frame-
>getHeight(),
                static_cast<int>(frame->size()));
```

```
  server.setContentLength(frame->size());
  server.send(200, "image/jpeg");
  WiFiClient client = server.client();
  frame->writeTo(client);
}
void handleJpgLo()
{
  if (!esp32cam::Camera.changeResolution(loRes)) {
    Serial.println("SET-LO-RES FAIL");
  }
  serveJpg();
}
void handleJpgHi()
{
  if (!esp32cam::Camera.changeResolution(hiRes)) {
    Serial.println("SET-HI-RES FAIL");
  }
  serveJpg();
}
void handleJpgMid()
{
  if (!esp32cam::Camera.changeResolution(midRes)) {
    Serial.println("SET-MID-RES FAIL");
```

```
  }
  serveJpg();
}
void  setup(){
  Serial.begin(115200);
  Serial.println();
  {
    using namespace esp32cam;
    Config cfg;
    cfg.setPins(pins::AiThinker);
    cfg.setResolution(hiRes);
    cfg.setBufferCount(2);
    cfg.setJpeg(80);

    bool ok = Camera.begin(cfg);
    Serial.println(ok ? "CAMERA OK" : "CAMERA FAIL");
  }
  WiFi.persistent(false);
  WiFi.mode(WIFI_STA);
  WiFi.begin(WIFI_SSID, WIFI_PASS);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
  }
```

```
  Serial.print("http://");
  Serial.println(WiFi.localIP());
  Serial.println("  /cam-lo.jpg");
  Serial.println("  /cam-hi.jpg");
  Serial.println("  /cam-mid.jpg");

  server.on("/cam-lo.jpg", handleJpgLo);
  server.on("/cam-hi.jpg", handleJpgHi);
  server.on("/cam-mid.jpg", handleJpgMid);
  server.begin();
}
void loop()
{
  server.handleClient();
}
```

*** Python Code for ESP32 CAM Product recognition and counting system**

```
import cv2
import urllib.request
import numpy as np
import pyodbc
from datetime import datetime
from collections import Counter
```

```
# Database connection using Windows authentication
conn = pyodbc.connect('DRIVER={SQL Server};'
                      'SERVER=DESKTOP-SJB3JL3;'
                      'DATABASE=DetectionObject;'
                      'Trusted_Connection=yes;')
cursor = conn.cursor()

# Create table if it does not exist
cursor.execute('''
   IF NOT EXISTS (SELECT * FROM sysobjects WHERE name='DetectionObject' and xtype='U')
   CREATE TABLE DetectionObject (
       id INT IDENTITY(1,1) PRIMARY KEY,
       timestamp DATETIME,
       name NVARCHAR(50),
       amount INT
   )
''')
conn.commit()

# Object detection setup
url = 'http://192.168.137.120/cam-hi.jpg'
winName = 'ESP32 CAMERA'
```

```
captureWinName = 'Captured Image'
cv2.namedWindow(winName, cv2.WINDOW_AUTOSIZE)
cv2.namedWindow(captureWinName, cv2.WINDOW_AUTOSIZE)

classNames = []
classFile = 'coco.names'
with open(classFile, 'rt') as f:
    classNames = f.read().rstrip('\n').split('\n')

# Debug: Print the number of classes loaded
print(f"Number of classes loaded: {len(classNames)}")

configPath = 'ssd_mobilenet_v3_large_coco_2020_01_14.pbtxt'
weightsPath = 'frozen_inference_graph.pb'

net = cv2.dnn_DetectionModel(weightsPath, configPath)
net.setInputSize(320, 320)
net.setInputScale(1.0 / 127.5)
net.setInputMean((127.5, 127.5, 127.5))
net.setInputSwapRB(True)

def process_image(img, display_window=None):
    classIds, confs, bbox = net.detect(img, confThreshold=0.5)
```

```
    if len(classIds) != 0:
        detected_objects = []
        displayed_objects = set()

        for classId, confidence, box in zip(classIds.flatten(), confs.flatten(), bbox):
            # Debug: Print classId to ensure it's within range
            print(f"Detected classId: {classId}")

            if 1 <= classId <= len(classNames):
                name = classNames[classId - 1]
                detected_objects.append(name)

                # Display each object only once
                if name not in displayed_objects:
                    cv2.rectangle(img, box, color=(0, 255, 0), thickness=3)
                    cv2.putText(img, name, (box[0] + 10, box[1] + 30),
cv2.FONT_HERSHEY_COMPLEX, 1, (0, 255, 0), 2)
                    displayed_objects.add(name)
            else:
                print(f"Invalid classId: {classId}")

        # Count the number of each detected object in the current frame
        object_counts = Counter(detected_objects)
```

```
        # Write data to SQL server if display_window is the captured window
        if display_window == captureWinName:
            timestamp = datetime.now()
            for name, amount in object_counts.items():
                cursor.execute("INSERT INTO DetectionObject (timestamp, name, amount)
VALUES (?, ?, ?)",
                               timestamp, name, amount)
                conn.commit()

    if display_window:
        cv2.imshow(display_window, img)

captured_image = None  # Global variable to store the captured image

while True:
    imgResponse = urllib.request.urlopen(url)
    imgNp = np.array(bytearray(imgResponse.read()), dtype=np.uint8)
    img = cv2.imdecode(imgNp, -1)

    process_image(img, winName)

    key = cv2.waitKey(5)
```

```
    if key & 0xFF == 27:  # ESC key to break
        break
    elif key & 0xFF == ord('h'):  # 'H' key to capture image
        timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
        captured_image = img.copy()  # Store the captured image
        cv2.imwrite(f'captured_{timestamp}.jpg', captured_image)
        print(f"Image captured and saved as captured_{timestamp}.jpg")
        cv2.imshow(captureWinName, captured_image)
        process_image(captured_image, captureWinName)

cv2.destroyAllWindows()
conn.close()
```

*C# Code for Devexpress UI

```
using System;
using DevExpress.Xpo;
using DevExpress.Persistent.Base;
using DevExpress.Persistent.BaseImpl;
using DevExpress.Persistent.Validation;
namespace DetectionObject.Module.BusinessObjects
{
    [DefaultClassOptions]
    public class DetectionObject : XPBaseObject
    {
```

```
public DetectionObject(Session session) : base(session) { }
int id;
[Key(true)]
public int ID
{
    get => id;
    set => SetPropertyValue(nameof(ID), ref id, value);
}
private string name;
private DateTime timestamp;
private int amount;
public string Name
{
    get => name;
    set => SetPropertyValue(nameof(Name), ref name, value);
}
public int Amount
{
    get => amount;
    set => SetPropertyValue(nameof(Amount), ref amount, value);
}
public DateTime Timestamp
{
```

```
            get => timestamp;
            set => SetPropertyValue(nameof(Timestamp), ref timestamp, value);
        }
    }
}
```