Smart Mouse Trap

Kritapas Chutiwanichyakul 62011145
Thawat Jarupenpat 62011278
Tinchupeam Weerayutvilai 62011281

Bachelor of Engineering in Software Engineering
Department of Computer Engineering
School of Engineering
King Mongkut's Institute of Technology Ladkrabang
Academic Year 2021

Team Software Project Report – Academic Year 2021

Bachelor of Engineering in Software Engineering

Department of Computer Engineering, School of Engineering

King Mongkut's Institute of Technology Ladkrabang

Title: Smart Mouse Trap

Authors

1. Kritapas Chutiwaniychyakul Student ID: 62011145
2. Thawat Jarupenpat Student ID: 62011278
3. Tinchupeam Weerayutvilai Student ID: 62011281

Approved for submission

……

(Asst.Prof.Dr. Pipat Sookvatana)
Advisor

Date ……../……../……..

Acknowledgement

We would like to express our deepest gratitude to Asst.Prof.Dr. Pipat Sookvatana who has provided us with his insight, guidance and meaningful advice in this project. His dedication and commitment to our team is tremendous which allows us to grow into better individuals characters and software engineers.

We also would like to express our deepest appreciation to Asst.Prof.Dr. Ukrit Watchareeruetai for giving us insight and wisdom on this project. Without his teachings, it would be very difficult and challenging to complete the project.

Abstract

Smart Mouse Trap is a team software project that aims to help users to catch or trap a mouse in a most humane way as possible without having to kill a mouse on a spot

The intention for this project is to help users catch a mouse without having to run around and catch it. This project is also suitable for a people who is animal-lovers which wouldn't harm an animal in anyway, so to catches the mouse, you simply have to placed the trap box and wait for the mouse to enter it, then the camera will detect by using Image Processing that the mouse has entered, then it will send the data to the door and close it.

The Project will mainly be developed in Python, Java and Arduino IDE. The data will be sent by using MQTT Protocol.

Table of Contents

List of Figures

Figure 22 - Received Base64 Image

Figure 23 - Screen Capture of Mobile Application

Chapter 1
Introduction

1.1 Motivation

Mouse traps are products that people normally use in almost every household. They are products that help people catch mice without being tired. However, some old-fashioned traditional mouse traps are unable to catch mice if they haven't eaten the bait, and some of them require you to stick it with glue, which will end up harming animals. This problem can be solved in many different ways. In this paper, we used some MQTT techniques to transfer data from a camera and a mouse trap to allow each part to communicate with each other.

The goal is to reform the ordinary mouse trap to become a Smart Mouse Trap. As a result, when it detects some mice, it can lock them immediately and report the status to the mobile application.

1.2 Objectives

The Primary goal of this project is to develop a software that links with a hardware device that traps the mouse inside the cage by using a MQTT protocol to receive the data from Image Processing when detecting a mouse. The main objectives are:

- To develop an accurate Image Classification Model to let the neuron predict output correctly
- To develop an application that can monitor the status of the camera through MQTT
- To develop a mouse trap that can respond to data received in both manual and automated way

1.3 Team Responsibilities

- Kritapas Chutiwanichyakul : Mobile Application, ESP32Cam (Base64 Image)
- Thawat Jarupenpat : Image Processing (Animal Detection)
- Tinchupeam Weerayutvilai : Hardware Programming (ESP32, ESP32Cam, MQTT

Chapter 2

Project Schedule

2.1 Team Members' Roles

In this project each member got assigned to do different tasks of the project. We split the work into 3 main parts.

The first part is the application development which develops the user interface, MQTT connection (with text, and base64 image payload).

The second part is the image processing part which will be used to detect the mouse object by using the model that is the set of weight and bias that has been built by Python to predict the accurate output by using method Image Classification.

The last one is the hardware programming and MQTT connection part, this part will connect the application part and image processing part so it can work together.

Member	Roles
Kritapas Chutiwaniychyakul	● Develop Application ● MQTT connection ● Base64 image encoding
Thawat Jarupenpat	● Build Image Classification Model ● Prediction of Mouse
Tinchupeam Weerayutvilai	● Hardware Programming (ESP32, ESP32Cam, MQTT) ● Trap Design

Chapter 3

Background Knowledge

3.1 MQTT

MQTT is one of the standard messaging protocols for the Internet of Things (IoT). It is designed as an extremely lightweight publish/subscribe messaging transport that is ideal for connecting remote devices with minimal bandwidth. MQTT is used in a wide variety of industries, such as automotives, transportation, and smart homes, etc.

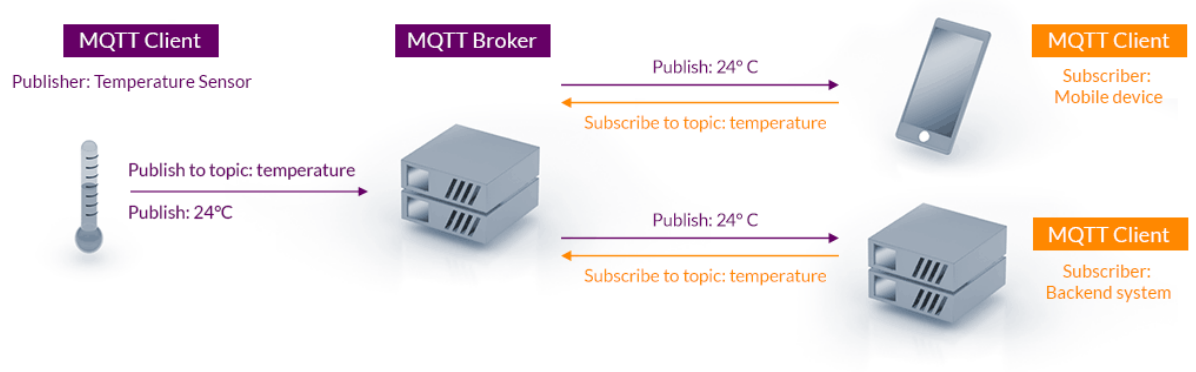

Figure 1 - MQTT Architecture

MQTT has been designed with a publish/subscribe architecture that allows clients to send and receive messages via an MQTT broker (act as a server).

3.2 Base64 Encoding

Base64 is a binary-to-text encoding that represents binary data in an ASCII string format by translating it into a radix-64 representation.

Source	w							o							o							
Bits	0	1	1	1	0	1	1	1	0	1	1	0	1	1	1	1	0	1	1	0	1	1	1	1
Base 64	d				2				9				v											

Figure 2 - Base64 Encoding

3.3 Image Classification

Image classification is the process of categorizing and labeling groups of pixels or vectors within an image based on specific rules. It used the concept of Machine learning by turning data into numbers.

3.4 Machine Learning

Machine Learning is a programming computer to optimize a performance using example data or past experience. The model that is obtained from machine learning is defined up to some parameters, and learning is the execution of a computer program to optimize the parameters of the model using the training data or past experience. The model may be predictive to make predictions in the future, or descriptive to gain knowledge from data.

3.4.1 Convolutional Neural Networks

Convolutional Neural Networks is the network that is mostly used in Image Classification. It is a type of artificial neural network used in image recognition and processing that is specifically designed to process pixel data. Neural Network is a system that is considered to be like a human brain which is used to think and produce an output.

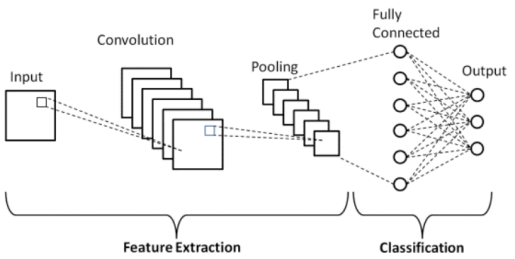

Figure 3 – Convolutional Neural Network

3.5 Tensorflow

Tensorflow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training models or sets of weights and bias and inference of deep neural networks. It is written in Python, C++ and CUDA. Platform that it used to develop tensorflow is mostly Linux and Windows and it's developed by Google Brain Team.

3.6 Docker

Docker is an open platform to generate OS and linux virtualization to deliver software in packages or containers. It enables developers to package applications into containers, and combine application source code with the operation system libraries and dependencies required to run that code in any environment. It is lightweight and doesn’t consume too much resources on your computer.

Chapter 4

Requirement Analysis / System Architecture / Design

4.1 Functional Requirement

1. The mouse trap's door will be controlled by a button.
2. The trap model should allow the user to uncover the cage in case the mouse has to be removed.
3. The system should allow the user to control the trap door.
4. The system should allow the user to monitor the camera via mobile application.
5. The system should tell the door status of the trap via mobile application.

4.2 Non-Functional Requirement

1. The system will communicate with parts via the MQTT protocol.
2. The sensor should be able to detect the mouse as soon as the mouse entered the trap
3. The system should be able to analyze if there is a mouse in the cage.
4. The mobile application user interface should provide a user-friendly experience.
5. The hardware controller board will be mainly developed using Arduino IDE

4.3 Use-Case Diagram

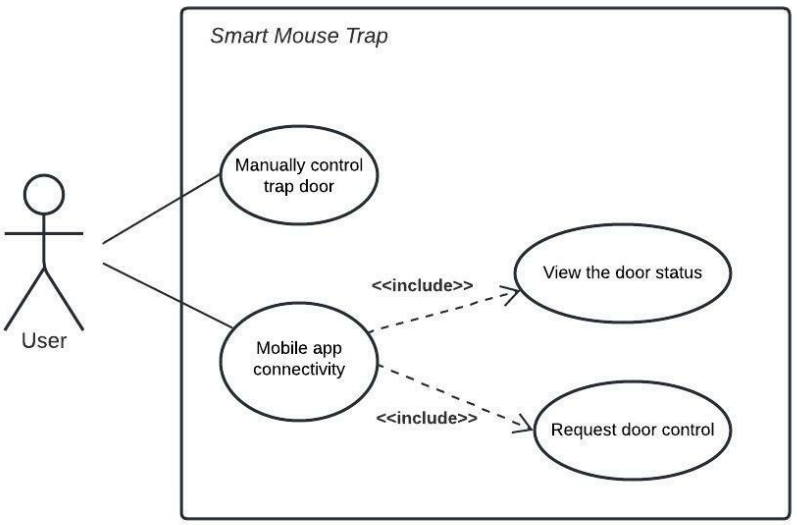

Figure 4 - Use-Case Diagram

The user should be able to manually control the trap door with the button. However, for ease of use, the system should be able to establish a mobile app connectivity to monitor the status and request door control.

4.4 Project Design

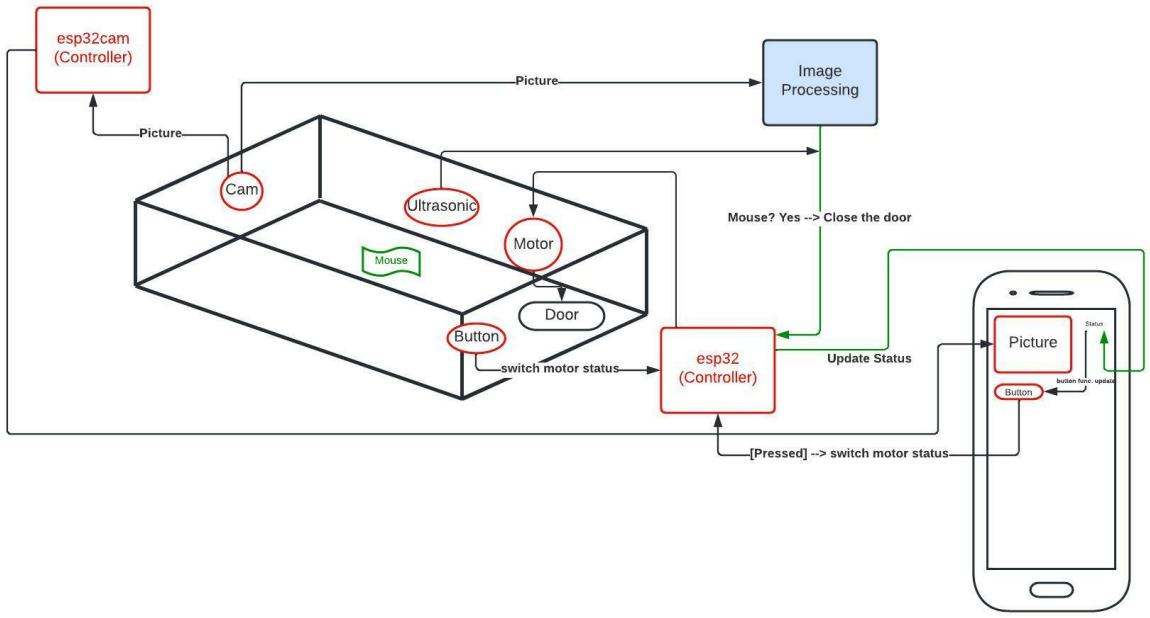

Figure 5 - Project design and the connectivity to other parts

The figure above shows the connectivity of hardware parts and the data sent from the controller to the mobile application (excluding the MQTT broker which will be mentioned later in this paper).

The trap door will be closed in case that both camera and ultrasonic confirm that there is a mouse inside.

4.4.1 Block Diagram

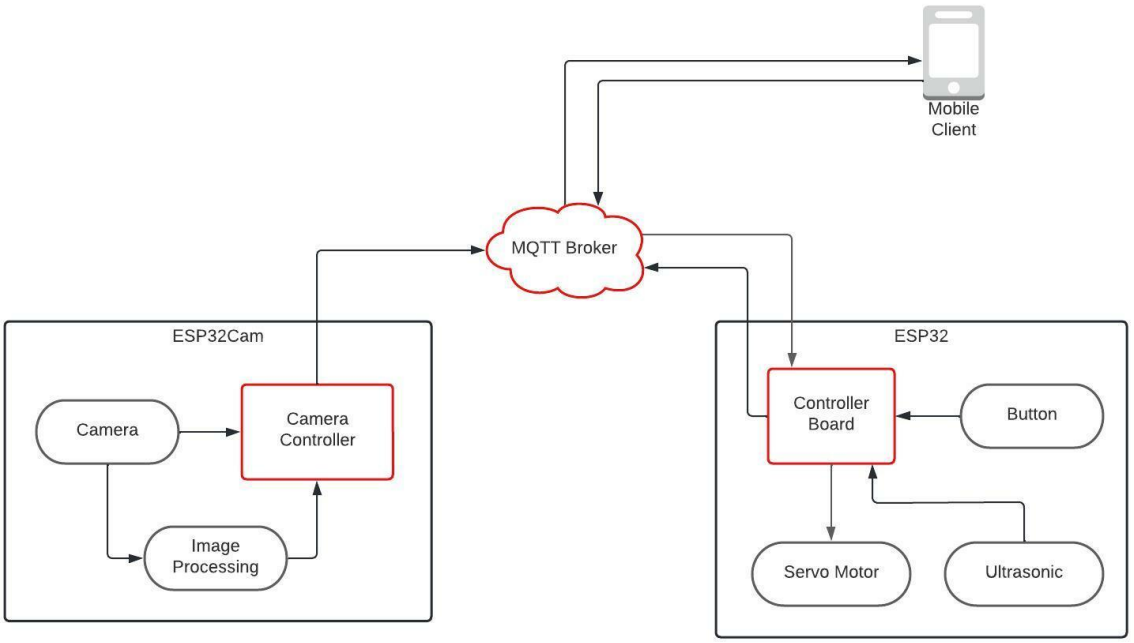

Figure 6 - System Block Diagram

The Block Diagram Figure shows you how the project connects to each part. We decided to use the MQTT protocol for sending and receiving data over the network, as mentioned in Chapter 4.5.1. This diagram shows that both ESP32Cam and ESP32 are connected to the broker, as well as the mobile client. Therefore, the ESP32 Controller board can be either controlled by a button manually or remotely over the internet via the mobile application.

4.4.2 System Design for Networking

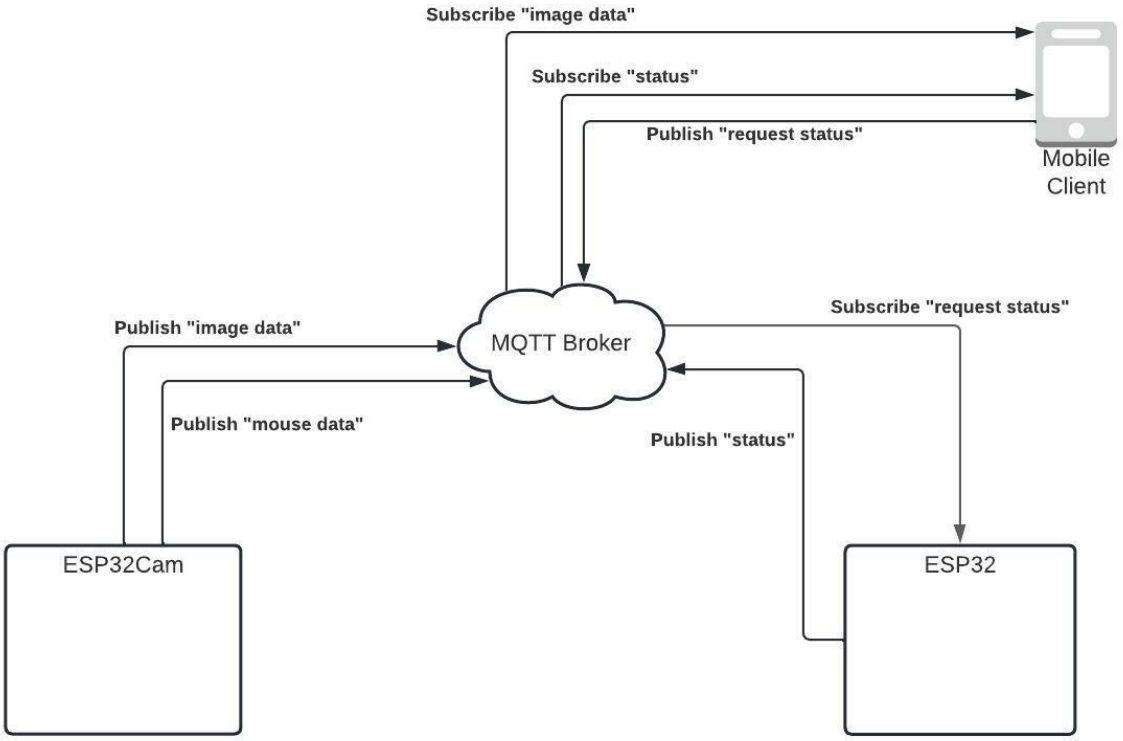

Figure 7 - System Design for Networking

This diagram shows how clients communicate with each other over the MQTT protocol using the "publish" and "subscribe" models. The ESP32Cam will send the image and mouse data for the android application client to be able to monitor, as well as send a request to open or close the trap door.

4.4.3 Schematic Design (ESP32)

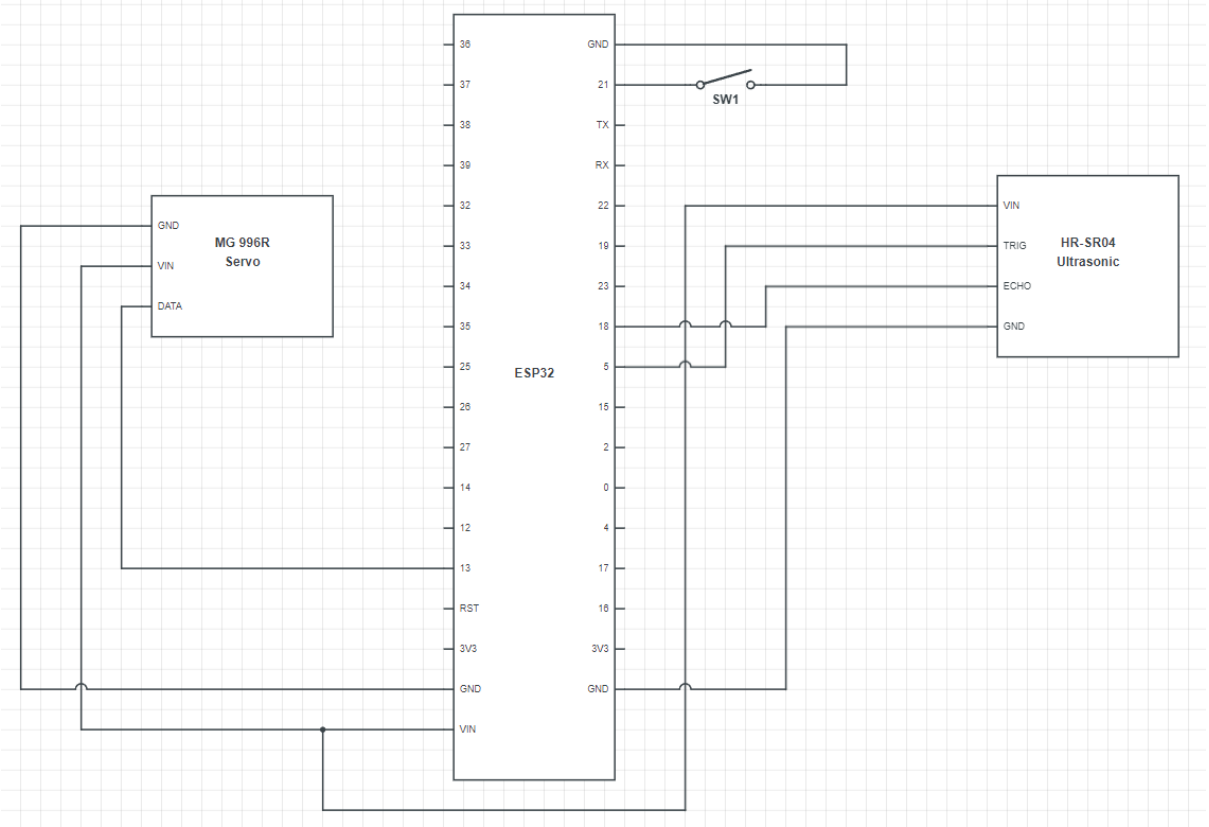

Figure 8 - Schematic Design

The figure will tell you which GPIO are used to connect to the servo MG996R, button switch, and ultrasonic sensor HC-SR04.

4.5 System Architecture

4.5.1 MQTT Architecture

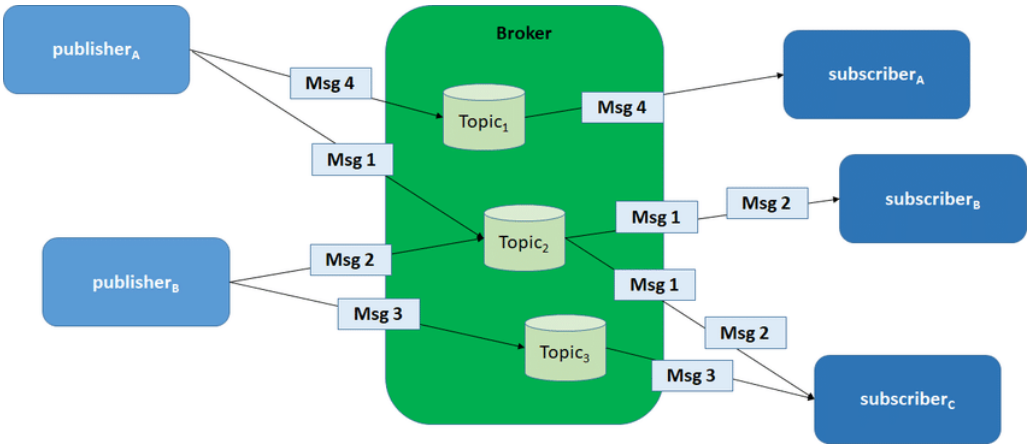

Figure 9 - MQTT Topics Architecture

With MQTT Protocol, it allows us to send and receive messages by connecting every part to the broker. The message payloads will be sent within the topic. To receive the published message, the client must subscribe to the same topic. The broker will send the message it receives only to those subscribers on the same topic as a result.

4.6 Mouse Detection Flowchart

Image Processing for Animal Detection flowchart:

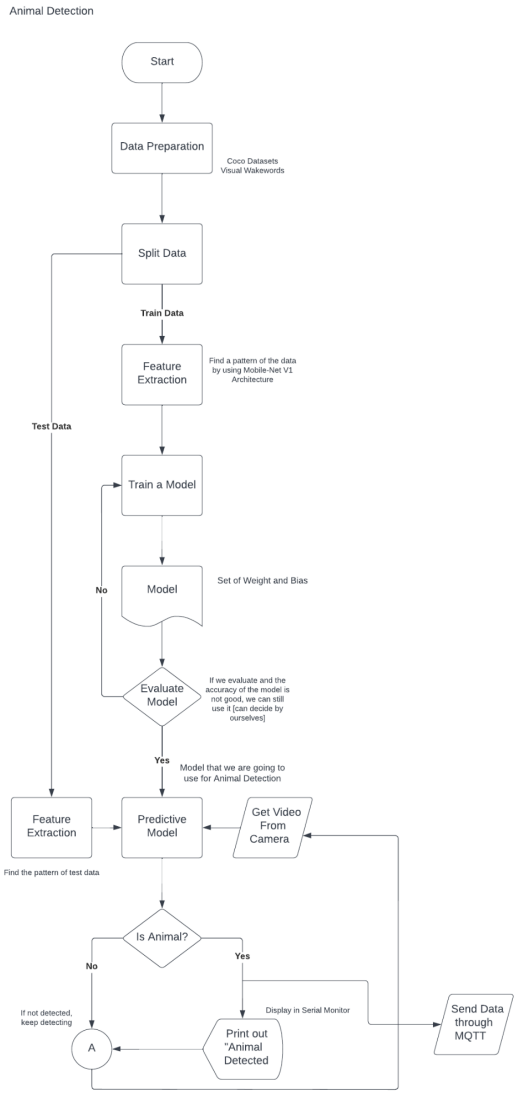

Figure 10 – Flowchart of Building Model and Animal Prediction & Detection

This Flowchart shows the process of how to get the final prediction model that is going to be used in IoT Devices and detect an animal. As mentioned in Chapter 5.2.3 . We use Visualwakeword from Coco dataset for labeling and preparing data for training. After the training process explained in Chapter 5.3.1 is finished, you will be able to get the output model called Predictive Model, then we will use that in IoT Device and let the machine predict whether this is an animal or not, then send the data through MQTT as mentioned in Chapter 4.4.2.

Chapter 5

Software Development

For this chapter, we will explain the concepts, tools, and techniques that are used for developing the project.

5.1 Development Tools

5.1.1 Arduino IDE

We selected Arduino IDE as the IDE to program the ESP32 and ESP32Cam board. Arduino IDE contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino hardware to upload programs and communicate with them. We can easily change the board module and we can easily upload the library that we need to use while developing the project on this IDE.

5.1.2 Android Studio

We selected Android Studio as the IDE to develop our application. Android Studio is a powerful code editor with smart editing and code refactoring for the Android Application.

5.2 Software Development Technologies

5.2.1 Programming Languages

- Java
- Python 3.9
- C and C++ (Arduino IDE)

5.2.2 Libraries and Frameworks

Image Processing

- Docker – OS & Software Library Visualization
- Tensorflow 2.0.8 – Machine and Deep Learning Framework

Arduino Library

- PubSubClient
- Servo
- base64
- Wifi
- dummy
- esp_camera

Android

- Paho MQTT Android

5.2.3. Data

5.2.3.1 Coco Datasets

COCO is a large-scale object detection, segmentation, and Classification dataset.
Coco has several features:

- Object segmentation Recognition in context of a large pixel resolution segmentation.
- 330,000 images
- Total of object in Images, 1.5 million objects
- 80 object categories
- Maximum of 5 different object per image

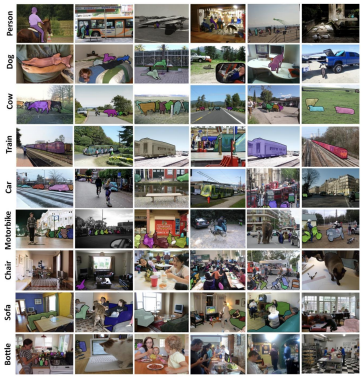

Figure 11 – Coco Datasets

5.2.3.2 Visualwakeword Datasets

Visualwakeword Datasets use Coco Datasets to reduce the label classes from 80 object classes to only 2 object classes which are Person and Non-Person to reduces the consummation of resources and perfect to use for IoT Devices which have small

memory and processing power.

5.2.4 Port Forwarding

For the data networking part we host our own MQTT broker and we need to use the port forward method to allow everyone in the team to be able to access the broker from the network. We are using the internet service provided by AIS and we need to use THDDNS to set the MQTT port. First we need to set the Domain name and then select port which one to use for the MQTT port. Then we need to access the router setting to set the IPv4 Port Mapping to set a virtual server on a network and allow the server to be accessed from the internet. First we choose the internal host which is one of our devices that is in the network like in Figure X. IPv4 Port Mapping then we need to set which protocol we want to use, set the internal port number which is 1883 as a default port for MQTT and external port number is 5050 according to the Figure X we need to use port 5050 because that is the only port that will allow any device to be able to connect the MQTT broker.

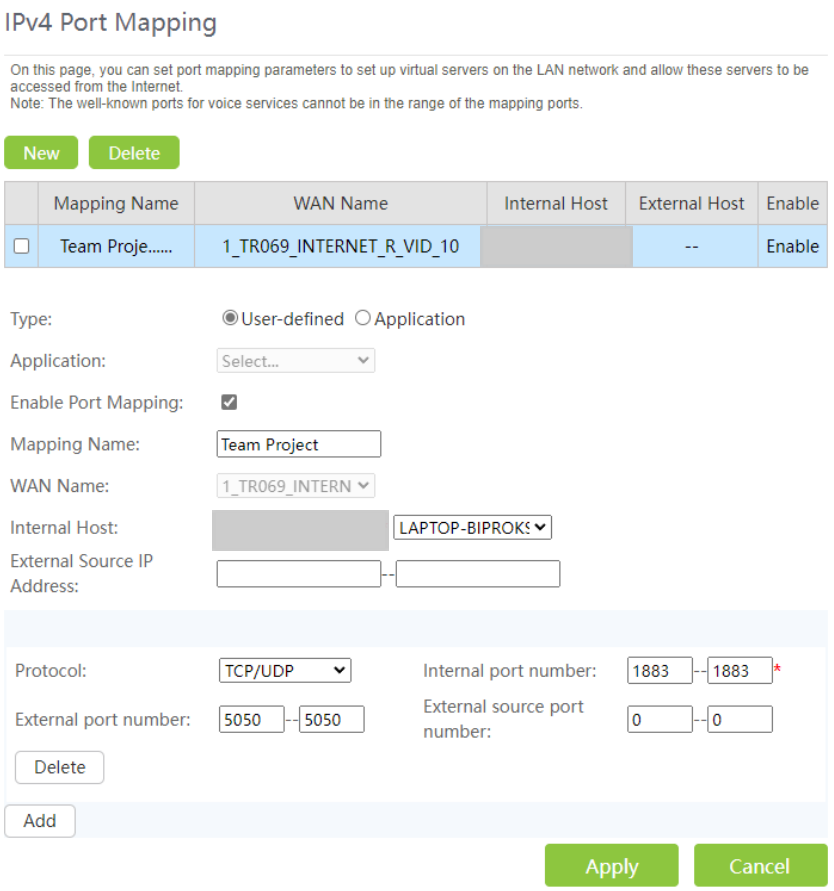

Figure 12 - IPv4 Port Mapping

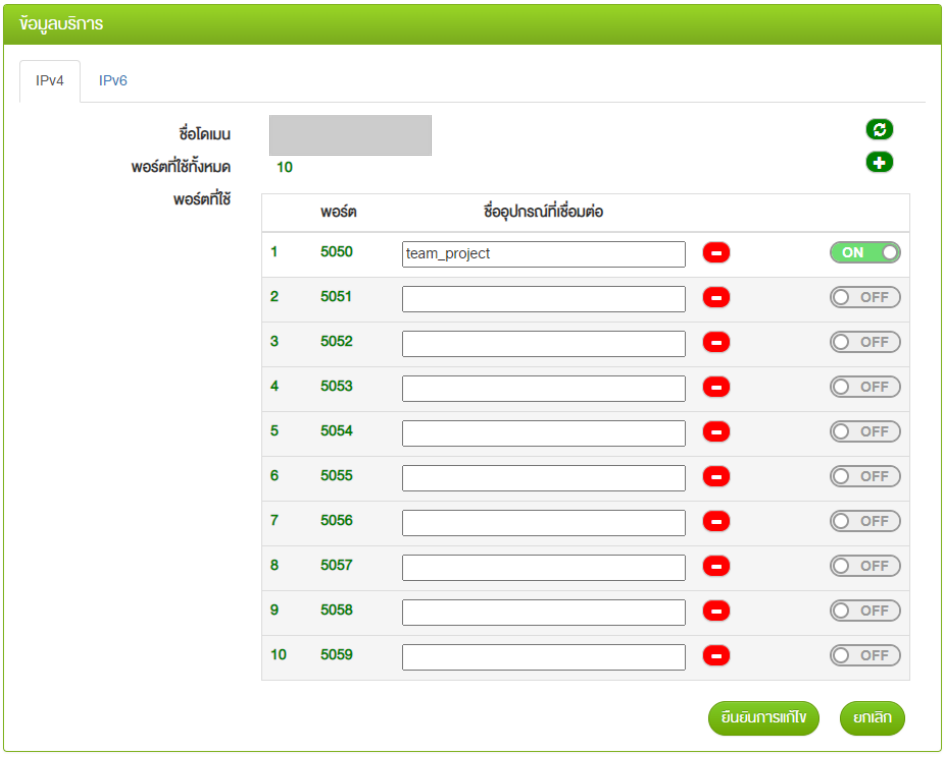

Figure 13 - THDDNS Menu

5.3 Technologies Concept / Algorithm

5.3.1 Training Model

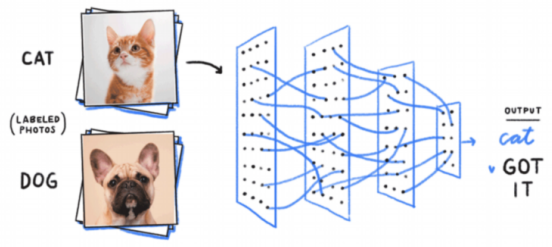

Figure 14 – Simple Neuron Network For Animal Classification

Component of Neuron Network:
1. Input Layer → Receive an Input {Image, Video}
2. Hidden Layer → Processing Prediction according to weights + bias
3. Output Layer → Prediction Output

Tensorflow uses optimizer methods to find appropriate weights which provide the output Y as close as the target output Y as much as possible. In our work, we use gradient descent methods to optimize weights.

-Gradient Descent is an algorithm used for minimizing loss function by generating the best possibilities of weights

-Loss Function is the percentage of numbers that have been predicted wrong by neurons compared to the real one.

5.3.2 Deployment

After that we get the final model which we obtain from the training process. We have to convert a model into a smaller model to fit the IoT device that has small memory and processor power, and then deploy the model into the IoT device.

5.3.3 MobileNet-V1

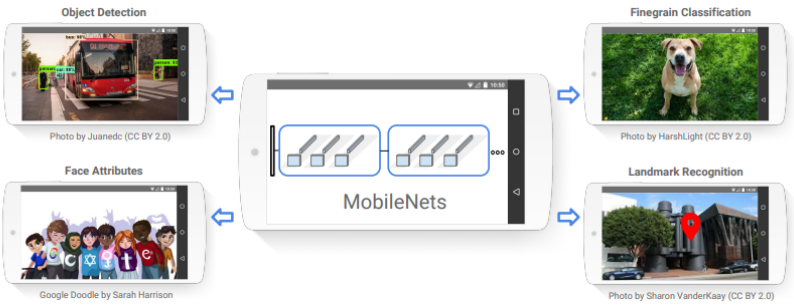

Figure 15 – MobileNet V1

MobileNet is a lightweight model for mobile and embedded applications. In this work we use mobilenet-v1 for the classification process. It is very popular because of the speed and accuracy, and it reduces the storage, therefore mobilenet-v1 is suitable for deploying models in an ESP32 board.

Chapter 6

Result

6.1 Hardware Model and Output Screenshots

6.1.1 Mouse Detection

The main functionality of the trap is to detect the mouse with an ultrasonic sensor and camera.

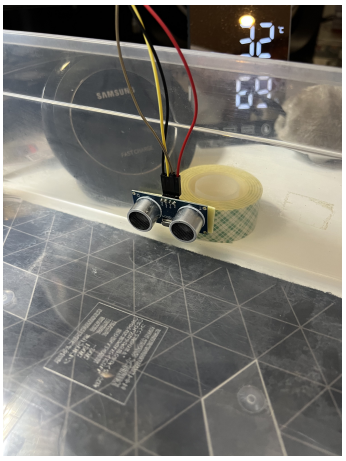

Figure 16 - Ultrasonic Sensor

Figure 17 - Overview of the trap

Animal Detection Output Serial

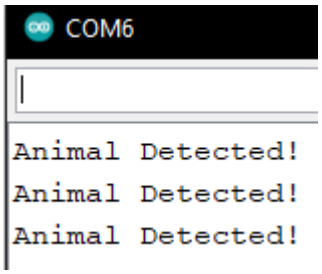

Figure 18 - Printed Out Animal Detected on Serial

If the Camera detects an animal inside the cage, it will print out “Animal Detected” on the serial monitor.

Image Processing result from MQTT Broker:

```
C:\mosquitto>mosquitto_sub -h projecttech.thddns.net -t "esp32cam/image_processing" -p 5050
1
1
1
1
```

Figure 19 – Mouse Data Detection to MQTT

Onces the camera detects that there is an animal inside the cage, the mouse detected data will be sent through MQTT to the MQTT broker. The results show 1 indicates that there is a mouse inside the cage.

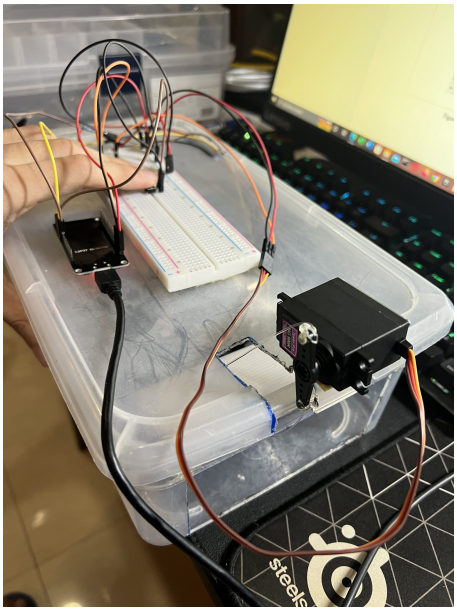

Figure 20 - Trap door opened

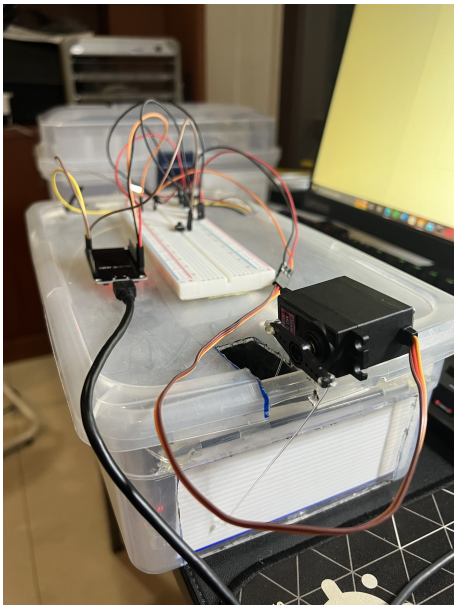

Figure 21 - Trap door closed

The controller board will compare the data received with an ultrasonic sensor. The cage will lock itself if the mouse has been confirmed.

ESP32Cam will also send the image to MQTT broker by convert it into base64 image format

```
C:\mosquitto>mosquitto_sub -h projecttech.thddns.net -t "esp32cam/image_data" -p 5050

/9j/4AAQSkZJRgABAQEAAAAAAAD/2wBDAAoHCAkIBgoJCAkLCw
oMDxkQDw4ODx8WFxIZJCAmJiQgIyIoLToxKCs2KyIjMkQzNjs9
QEFAJzBHTEY/Szo/QD7/2wBDAQsLCw8NDx0QEB0+KSMpPj4+Pj
4+Pj4+Pj4+Pj4+Pj4+Pj4+Pj4+Pj4+Pj4+Pj4+Pj4+Pj4+Pj4+
Pj4+Pj7/xAAfAAABBQEBAQEBAQAAAAAAAAAAAQIDBAUGBw
gJCgv/xAC1EAACAQMDAgQDBQUEBAAAAX0BAgMABBEFEiExQQYT
UWEHInEUMoGRoQgjQrHBFVLR8CQzYnKCCQoWFxgZGiUmJygpKj
Q1Njc4OTpDREVGR0hJSlNUVVZXWFlaY2RlZmdoaWpzdHV2d3h5
eoOEhYaHiImKkpOUlZaXmJmaoqOkpaanqKmqsrO0tba3uLm6ws
PExcbHyMnK0tPU1dbX2Nna4eLj5OXm5+jp6vHy8/T19vf4+fr/
```

Figure 22 - Received Base64 Image

We can choose to open and close the door by using a button, via application, or let the system detect the mouse and automatically close the door.

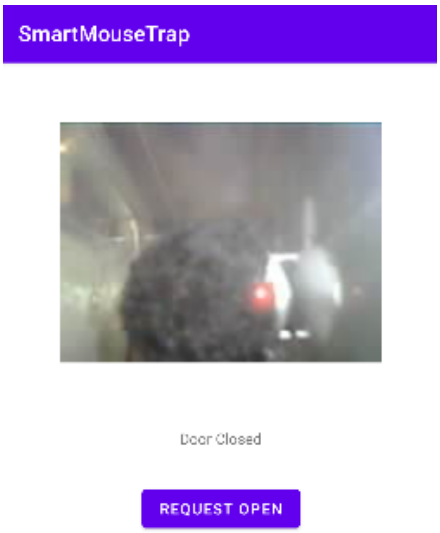

Figure 23 - Screen Capture of Mobile Application

Chapter 7

Conclusion and Future Work

In conclusion, we have developed an application which will get the data via MQTT broker and the ESP32 and ESP32Cam will be the one to publish the message to the MQTT broker. The application will display the picture that is captured from ESP32Cam and the user can control the trap door via the mobile application.

The system has met our requirements that we defined at the beginning. During the development we encountered many problems, but the hardest one is the hardware problem. We realized that the ESP32Cam dram is not enough to do image processing and encode the image to base64 file at the same time. We solve this problem by using two ESP32Cam, one for encoding the image and another one for image processing. In the future we will improve by having better hardware and trap design for example using 3D printers for the cage.

In this project we've learned multiple life lessons. First, we know that project planning is very important for a long and large project. We need to have a proper project schedule and discipline so that we can get the work done on time otherwise the workflow would be delayed and the development process will get slower.

Finally, we realized that the main problem that we encountered during the project was that the hardware processor power was not enough to run two processing parts at the same time, which means we have to seperate the processor board and run two different parts. One board is for Image Processing to run an animal detection and send the data through MQTT, and the other board is used for sending running hardwares components and sending string encoder 64 bit to applications.

Reference

Unknown, U. (n.d.). *The standard for IOT messaging*. MQTT. Retrieved May 25, 2022, from https://mqtt.org/

12, P. on O., 4, P. on O., 30, P. on S., 20, P. on S., & 14, P. on S. (2021, May 11). *What is MQTT and why you need it in your IOT architecture*. BehrTech. Retrieved May 25, 2022, from https://behrtech.com/blog/mqtt-in-the-iot-architecture/

Unknown, U. (n.d.). *Base64 - MDN web docs glossary: Definitions of web-related terms: MDN*. MDN Web Docs Glossary: Definitions of Web-related terms | MDN. Retrieved May 25, 2022, from https://developer.mozilla.org/en-US/docs/Glossary/Base64

Woolhadotcom. (n.d.). *Deno - Base64 Encoding & Decoding Examples*. Woolha. Retrieved May 25, 2022, from https://www.woolha.com/tutorials/deno-base-64-encoding-decoding-examples

Team, T. A. (n.d.). *Arduino Integrated Development Environment (IDE) V1: Arduino documentation*. Arduino Documentation | Arduino Documentation. Retrieved May 25, 2022, from https://docs.arduino.cc/software/ide-v1/tutorials/arduino-ide-v1-basics

Unknown, U. (n.d.). *Figure 2: Iot communication architecture and protocols. - researchgate.net*. Communication architecture. Retrieved May 25, 2022, from https://www.researchgate.net/figure/IoT-Communication-Architecture-and-Protocols_fig1_329183740

Unknown, U. (n.d.). *What is Docker?* IBM. Retrieved May 25, 2022, from https://www.ibm.com/cloud/learn/docker

Unknown, U. (n.d.). *Common objects in context*. COCO. Retrieved May 25, 2022, from https://cocodataset.org/#home

Unknown, U. (2022, May 1). *Tensorflow*. Wikipedia. Retrieved May 25, 2022, from https://en.wikipedia.org/wiki/TensorFlow

Chowhery, A., Warden, P., Shelens, J., Howard, A., & Rhodes, R. (2019, June 12). *ArXiv:1906.05721v1 [CS.CV] 12 jun 2019*. Computer Vision and Pattern Recognition. Retrieved May 25, 2022, from https://arxiv.org/pdf/1906.05721.pdf

Howard, A. G., Zho, M., Chen, B., Kalenichenko, D., Wang , W., Weyand, T., Andreetto, M., & Adam, H. (2017, April 17). Andrew G. Howard Menglong Zhu Bo Chen Dmitry Kalenichenko ... - arxiv. Efficient Convolutional Neural Networks for Mobile Vision Applications. Retrieved May 25, 2022, from https://arxiv.org/pdf/1704.04861v1.pdf