



Palestine Polytechnic University  
College of Information Technology and Computer Systems Engineering  
Department of Computer Systems Engineering

## Smart Aqua Guardian

Aziz Amro  
Haya Alami  
Qusai Hourob

Supervised by:  
Dr. Amal Dweik  
Dr. Ayman Wazwaz

*To fulfill the requirements for a bachelor's degree in the field of Computer Systems Engineering*

2023-2024

# Acknowledgment

In the name of the Almighty, the most compassionate, the most merciful, who bestowed upon us strength and knowledge, enabling the successful completion of this project. Embarking on this endeavor provided us with valuable experience. And for this, we extend our gratitude to all those who contributed. Our heartfelt appreciation goes to our graduation project supervisors, Dr. Amal Al-Dweik and Dr. Ayman Wazwaz, for their unwavering guidance, encouragement, and support throughout the semester. We also extend our thanks to the dedicated educators in the College of Information Technology and Computer Engineering, who selflessly shared their wisdom, shaping us into diligent engineers. A special acknowledgment is due to our families, whose generous encouragement and steadfast support have been constant throughout our lives. To our friends, we express sincere gratitude for your unwavering support during this significant chapter of our journey. In conclusion, we extend our thanks to everyone who played a role in supporting and encouraging us, ultimately contributing to the successful completion of our graduation project.

## Abstract

Smart Rescue System for Private Swimming Pools is an AI-based solution designed to make swimming pools safer for children. Swimming pools are fun and enjoyable places. But, they could also be very dangerous if no adults control were around. Children's safety is one of the most demanding requirements when building such pools. Our system represents a significant step forward in improving the quality of pools' safety for children and the piece of mind for their parents, making sure the pools stay as what they're supposed to be, a fun enjoyable place for everyone. Therefore, this system aims to leverage the power of artificial intelligence to provide a rescue mechanism that alert parents when a child approaches a critical distance from the pool. And if the child is drowning, the system will raise this child upward the pool.

The system consists of cameras, sensors and an AI system that scans, captures and analyzes the actions of individuals around and inside the pool. The system then classifies whether the individual is a child or an adult. If the system detects a child approaching, parents will be alerted immediately. Further, if the child is drowning, the system will raise up a liftable plate installed at the ground level of the pool rescuing the child from drowning or at least a permanent brain damage.

The system is tested with different cases and it is successfully detects the danger times , suspected cases and reacts accordingly in a timely fashion within 10 seconds.

يقدم نظامنا حلا مبتكرا يعتمد على تكنولوجيا الذكاء الاصطناعي لجعل حمامات السباحة بيئة أكثر أمانا للأطفال، برغم أن حمامات السباحة تعتبر مكانا ممتعا، إلا أنها يمكن أن تكون خطيرة جدا بغياب الرقابة الكافية للبالغين. وتعتبر أهمية سلامة الأطفال متطلبا أساسيا عند بناء مثل هذه المرافق، وهذا النظام يسعى لاستخدام فعالية الذكاء الاصطناعي لتوفير آلية إنقاذ تنبه الوالدين عند اقتراب الأطفال من مسافة خطرة قرب حمام السباحة، وفي حالة الاشتباه بالغرق، يتدخل النظام برفع الطفل لسطح الماء، مما يقلل من مخاطر الإصابة.

يمثل نظامنا خطوة كبيرة إلى الأمام في تحسين جودة سلامة حمامات السباحة للأطفال وراحة أولياء أمورهم، أملين بأن تمثل حمامات السباحة كما يفترض أن تكون، مكانا ممتعا ومسليا للجميع.

يتكون النظام من كاميرات وأجهزة استشعارات بالإضافة إلى نظام ذكاء اصطناعي يسهم في رصد وتحليل سلوكيات الأفراد حول وداخل حمامات السباحة. يقوم النظام بتصنيف الأفراد بناء على فئة العمر، حيث يقوم بتنبيه الوالدين فور اقتراب طفل صغير. وفي حالة الغرق، يتخذ النظام إجراء فوريا يرفع مستوى الصوت وتحريك لوحة قابلة للرفع لضمان إنقاذ الطفل من الخطر المحتمل أو التداول مع أثار دائمة على الصحة العقلية ثم اختيار النظام وفي حالات مختلفة ونجح في الكشف عن أوقات الخطر والحالات المشتبه بها ويتفاعل وفقا لذلك في الوقت المناسب خلال 10 ثوان.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Preface . . . . .	1
1.2	Problem statement . . . . .	1
1.3	Aims and objectives . . . . .	2
1.4	Requirements . . . . .	2
1.4.1	Functional . . . . .	2
1.4.2	Non-functional . . . . .	3
1.5	System Description . . . . .	3
1.6	Limitations and constraints . . . . .	4
1.7	Schedule . . . . .	4
1.8	Report outline . . . . .	4
<b>2</b>	<b>Background</b>	<b>5</b>
2.1	Preface . . . . .	5
2.2	Theoretical background . . . . .	5
2.2.1	Computer vision . . . . .	5
2.2.2	Internet of Things . . . . .	5
2.2.3	Edge computing . . . . .	5
2.2.4	IoT protocols . . . . .	6
2.2.5	CNNs (Convolutional Neural Networks) . . . . .	6
2.2.6	Air compressors and pneumatic valves . . . . .	7
2.3	Literature review . . . . .	7
2.3.1	Automated rescue system for children in private home swimming pools . . . . .	7
2.3.2	The Design of a Smart Rescue System for Private Swimming Pools . . . . .	7
2.4	Summary . . . . .	8
<b>3</b>	<b>System Design</b>	<b>9</b>
3.1	Preface . . . . .	9
3.2	Project setup and prototype design . . . . .	9
3.2.1	Pool model . . . . .	9
3.2.2	Pool ground . . . . .	10
3.2.3	Cylinders . . . . .	11
3.2.4	Pneumatic valve . . . . .	11
3.2.4.1	5/2 way single solenoid . . . . .	11
3.2.4.2	5/2 way double solenoid . . . . .	12

3.3	System components and design alternatives .....	12
3.3.1	Hardware components .....	12
3.3.1.1	Controllers .....	12
3.3.1.2	Age identifier .....	13
3.3.1.2.1	Hardware options .....	13
3.3.1.2.2	Software options .....	14
3.3.1.3	Proximity detector .....	14
3.3.1.4	Camera .....	15
3.3.1.5	Alert .....	16
3.3.1.6	Air compressor .....	16
3.3.1.7	Power Supply .....	16
3.3.1.8	Relay .....	17
3.3.1.9	Push button .....	17
3.3.1.10	Piezoelectric Buzzer .....	18
3.3.2	Software components .....	18
3.3.2.1	Drowning detector .....	18
3.3.2.1.1	Custom trained CNN(Convectional Neural Network) .....	18
3.3.2.1.2	YOLO .....	18
3.3.2.2	Communication protocols .....	19
3.3.2.3	Brokers .....	20
3.4	Conceptual system design .....	20
3.5	Algorithms and methodologies .....	22
3.5.1	Proximity detection pseudocode .....	22
3.5.2	Drowning detection pseudocode .....	22
3.5.3	Algorithms .....	22
3.5.3.1	Age Detection using OpenCV in Python .....	22
3.5.3.2	Drowning detection model .....	24
3.5.4	Sequence diagrams .....	25
3.6	Schematic diagram .....	27
3.7	Summary .....	30
<b>4</b>	<b>Implementation</b> .....	<b>31</b>
4.1	Preface .....	31
4.2	Hardware implementation .....	31
4.2.1	Prototype setup .....	31
4.2.2	Alert and proximity detector .....	31
4.2.3	Rescue System .....	32
4.2.3.1	Controlling and interfacing circuits .....	32
4.2.3.2	Cylinders setup .....	32
4.2.3.3	Pneumatic supply .....	33
4.3	Software implementation .....	33
4.3.1	Server setup .....	33
4.3.1.1	Mosquitto broker installation and configuration .....	33
4.3.1.2	Security implementation .....	34

4.3.1.2.1	Authentication . . . . .	34
4.3.1.2.2	Authorization . . . . .	36
4.3.2	AI models integration . . . . .	36
4.3.2.1	Age identification model . . . . .	37
4.3.2.2	Drowning detection model . . . . .	37
4.3.3	Drowning model dataset . . . . .	37
4.3.4	Clients setup . . . . .	38
4.3.4.1	ESP32s . . . . .	38
4.3.4.1.1	Rescue ESP32s . . . . .	40
4.3.4.1.2	Alert and proximity detector ESP32 . . . . .	41
4.3.4.2	Manager client . . . . .	41
4.3.5	App build and setup . . . . .	43
4.3.5.1	Streams . . . . .	43
4.3.5.2	Buttons . . . . .	44
4.4	challenges . . . . .	44
4.5	Summary . . . . .	45
<b>5</b>	<b>Testing and Results</b>	<b>46</b>
5.1	Preface . . . . .	46
5.2	Hardware testing . . . . .	46
5.2.1	Unit testing . . . . .	46
5.2.1.1	Esp32 . . . . .	46
5.2.1.2	Raspberry Pi 4 . . . . .	46
5.2.1.3	Cameras . . . . .	46
5.2.1.4	Pneumatic valves . . . . .	46
5.2.1.5	Pneumatic cylinder . . . . .	47
5.2.1.6	Ultrasonic sensor . . . . .	47
5.2.1.7	Alert . . . . .	47
5.2.1.8	Electrical switch . . . . .	47
5.2.1.9	Metal net and weight support . . . . .	47
5.2.2	Software components . . . . .	47
5.2.2.1	Raspberry Pi OS installation . . . . .	48
5.2.2.2	Age detection . . . . .	48
5.2.2.3	Drowning detection . . . . .	49
5.2.2.4	Broker . . . . .	50
5.2.2.5	Response time . . . . .	50
5.3	Summary . . . . .	50
<b>6</b>	<b>Conclusion and future work</b>	<b>51</b>
6.1	Preface . . . . .	51
6.2	Conclusion . . . . .	51
6.3	Future work . . . . .	51

# List of Figures

1.1 System description .....	3
2.1 Edge Computing Architecture. ....	6
2.2 Convolutional Neural Network Architecture. ....	6
3.1 Prototype design .....	9
3.2 Perforated aluminium plat.....	11
3.3 5/2 way single solenoid.....	12
3.4 5/2 way double solenoid .....	12
3.5 Kinect sensor V2.....	14
3.6 TCRT5000L sensor.....	14
3.7 Motion detector options .....	15
3.8 Air compressor.....	16
3.9 Power supply.....	17
3.10 Relay device.....	17
3.11 Push button switch .....	17
3.12 Buzzer piezosounder .....	18
3.13 Yolo Algorithm .....	19
3.14 System block diagram.....	21
3.15 System conceptual diagram .....	21
3.16 Proximity detection algorithm .....	22
3.17 Drowning detection algorithm .....	22
3.18 Age identification model algorithm .....	24
3.19 Drowning detection model algorithm .....	25
3.20 Sequence diagram of proximity detection system.....	26
3.21 Sequence diagram of rescue system.....	27
3.22 Schematic diagram of outside-pool components .....	28
3.23 Schematic diagram of inside-pool components .....	29
4.1 Prototype components.....	31
4.2 Alert and proximity detector connections .....	32
4.3 Rescue system connections.....	32
4.4 Cylinders setup .....	33
4.5 Pneumatic supply .....	33
4.6 Mosquito installation .....	34
4.7 Mosquito configurations file.....	34

4.8	Disable anonymous connections to the broker.....	35
4.9	Set the path to users/passwords file in Mosquitto .....	35
4.10	Creating users in Mosquitto .....	35
4.11	Setting the path to the ACL file .....	36
4.12	Mosquitto ACL file.....	36
4.13	Age identification model Output.....	37
4.14	Output sample of the drowning detection model .....	37
4.15	drowning model dataset .....	38
4.16	Libraries installation .....	38
4.17	Define connection credentials .....	39
4.18	Setup function.....	39
4.19	Ping and reconnect functions .....	40
4.20	Rescue ESP32 setup.....	40
4.21	Callback function.....	40
4.22	Alert and proximity detector ESP32 setup.....	41
4.23	Alert and proximity detector Callback function.....	41
4.24	Manager client setup and connection .....	42
4.25	Callback function for MANAGER client.....	42
4.26	App connection setup.....	43
4.27	System App.....	44
5.1	Age Model test.....	48
5.2	Drowning Model test .....	49



# List of Tables

1.1	Project schedule in the first and the second semester.....	4
2.1	Comparison with other projects .....	8
3.1	Comparison of Aluminum, Iron, and Copper for Underwater Applications .....	10
3.2	Differences between cylinders .....	11
3.3	Differences between microcontrollers.....	13
3.4	Differences between motion detectors .....	15
3.5	Differences between cameras.....	15
3.6	Differences between IoT protocols .....	19
3.7	Differences between brokers.....	20

# Chapter 1

## Introduction

### 1.1 Preface

The safe and intelligent development and design of a private pool refers to the creation of a swimming pool environment that prioritizes safety and utilizes advanced technology and design principles to mitigate risks and enhance the overall pool experience. The importance of implementing a system that focuses on child safety, drowning prevention, and real-time control in private swimming pools cannot be overstated. It directly addresses many critical issues of enormous importance. Most important is the protection of children, as drowning remains a leading cause of accidental deaths, especially among young people [1]. It efficiently identifies risks using advanced technology like computer vision and deep learning, enhancing safety measures. Real-time alerts and rapid response capabilities are pivotal in emergencies, potentially saving lives and instilling trust. This system provides peace of mind to parents and pool owners, allowing enjoyment without constant worry. In short, this system is a crucial step towards enhancing safety and preventing catastrophic accidents, providing protection and peace of mind and demonstrating a commitment to safety and well-being.

### 1.2 Problem statement

When designing a new house, families strive to create the most entertaining and enjoyable living space possible. Swimming pools have become a popular addition, primarily for children to have fun while parents supervise and ensure their safety.

Currently, pools are designed by installing gates around the pool area and having a lifeguard on duty to assist in case of emergencies. Additionally, the water level is carefully monitored to ensure that it is appropriate for the age and swimming ability of the swimmers.

Regrettably, despite the proximity of these swimming pools to parents, there are numerous devastating accidents involving children drowning because they were left to swim alone, away from adult supervision. This turns what should be a fun experience into tragic moments where children lose their lives. Even in cases where parents do notice their children in a dangerous situation, they often cannot call for help in time. In response to this pressing challenge, our project aims to eliminate this fear. We propose the development of a 2-levels automated rescue system. In which the first level will alert parents in case of children proximity, and the second level will handle the drowning situations by a liftable ground that lifts children upward the pool in these cases.

### 1.3 Aims and objectives

In this project, we propose a system that aims to provide the following features:

1. The system aims to develop a smart private swimming pool system. In order to achieve this aim, the following objectives should be accomplished:
  - (a) Make a survey to determine the safe conditions and different swimming scenarios to detect the danger cases.
  - (b) Search for data sets for using an AI mechanisms to be used in implementing the smart system.
  - (c) Train an AI model to detect drowning situations in video streams.
2. We aim to build an alert system that parents are accessible to all the time. In order to achieve this aim, the following objectives should be accomplished:
  - (a) Install buzzers and lights inside the house to have parents alerted in case of any danger.
  - (b) Build a mobile application to make sure parents are alerted even if they're outside the house, and allow parents to control and monitor the pool area.
3. The system aims to build a two level rescue system. In order to achieve this aim, the following objectives should be accomplished:
  - (a) Design a first level of rescue outside the pool to detect when the child is in a critical distance from the pool, and alert parents upon.
  - (b) Design a second level of rescue inside the pool to ensure that when the child is drowning and still no act of rescue was taken, a liftable ground will be raised up carrying the drowning kid to safety.

### 1.4 Requirements

To specify the system requirements, the following functional and non-functional requirements can be considered:

#### 1.4.1 Functional

1. According to CDC, drowning is the first and second cause of death for children between 1-4 and 5-12 years old, respectively[1]. So, the system should easily differentiate between children under 12 years old and adults.
2. The system should be able to detect and monitor children in swimming pools.
3. The system should be able to send immediate alerts and real-time notifications when needed.
4. The system should be activated to rescue a child in case of a drowning situation.
5. The system should be able to use AI to detect drowning incidents.
6. The system should be integrated with a mobile app to enable parents to remotely monitor the swimming pool area, ensuring their peace of mind.
7. The mobile application should allow parents to override the action of the rescue system, ensuring the presence of human action in case of emergencies.

### 1.4.2 Non-functional

1. Security: The system must ensure strong security and privacy measures to protect user data and system access by using authentication methods.
2. Reliability: The system must have reliable data storage and backup systems to prevent data loss and reduce false alarms. by thorough testing and quality assurance protocols to minimize the risk of false alarms and enhance overall system reliability. Additionally, in the event of a component failure, the net plate will automatically ascend, maintaining a protective response even under component faults.
3. Response time: The system must have real-time response, which is crucial to generate alerts and activate the rescue system. According to hackensack meridian health, it should be less than 20 seconds [2].
4. Availability: The system must be highly available to ensure continuous monitoring and protection.

## 1.5 System Description

Our system's primary objective is to make comprehensive safety mechanism designed to prevent child drowning incidents and providing swift assistance during pool emergencies through a three-stage process:

**Stage 1:** When motion is detected near the pool, a sensor activates to scan the pool area for any human presence. The collected data is then analyzed to identify and differentiate individuals based on their age.

**Stage 2:** If a child (individuals under 12 years old) is detected, the system activates an alert system. This triggers both a home-based buzzer alert for parents and alerts through a mobile app. Additionally, an external camera scans the pool area using AI technology to identify signs of drowning, sending an immediate alert to parents if detected.

**Stage 3:** In the event that the child remains in distress for a specified duration (Explained in chapter 3), the system activates an emergency rescue system that deploys a liftable net for assistance, positioned at ground level in the pool area.

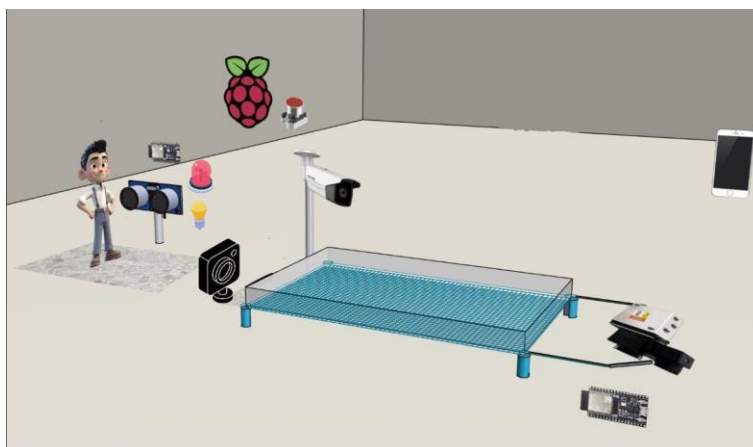


Figure 1.1: System description

## 1.6 Limitations and constraints

- **Cost constraint:** Our system model will simulate the pool environment. The proposed solution is fit to real pools. However, it will require more professional, yet expensive components. So, due to the high cost of these components, we will design the prototype based on what's efficient to fit the designed model.
- **The continuous internet connection:** Internet connectivity is necessary to establish the accessibility of parents via the mobile application. However, the system operates correctly without the need of internet.
- **Total weight of objects inside the pool:** The system is subjected to the maximum weight capacity that the metal net can support. However, as we use real videos and no need to have a child in the pool, this constraint can be neglected.

## 1.7 Schedule

The tasks of the system implementation and operation are distributed along the first and the second semester summarized in Table 1.1.

Table 1.1: Project schedule in the first and the second semester

Week	The first semester			The second semester			
	1 - 4	5 - 10	11 - 15	1 - 5	6 - 9	10 - 14	15
Selection of project Idea							
Collecting the Data							
System Design							
System Implementation							
System testing							
system operation							
Documentation							

## 1.8 Report outline

This report is organized as follows: Chapter 2 provides a brief introduction to the key technologies and algorithms that will be employed in the project, along with a literature review that compares our project to similar ones. Chapter 3 outlines the project's design, encompassing both hardware and software aspects. It discusses design choices, the conceptual background of the software, and presents a schematic diagram. Chapter 4 explains the system implementation and implementation challenges. Chapter 5 explains the test process of the system components. Finally, chapter 6 concludes the results of the work and recommendations of improvements.

# Chapter 2

## Background

### 2.1 Preface

This chapter introduces the theoretical foundation essential to our project. Following this, we'll dive into a literature review, comparing our project with what's been done before. This comparison helps us highlight the unique aspects and innovations our project brings to the table. In essence, this chapter provides the background needed to understand our project's roots and its place among previous efforts in the field.

### 2.2 Theoretical background

In this section, we'll explain the core components of our project. Computer vision, which involves processing inputs such as photos and videos. These inputs are fed into an algorithm that checks for drowning.

#### 2.2.1 Computer vision

Computer vision is branch of AI that visualizes the real world. using digital images from cameras, videos, and deep learning models [3]. Computer vision is widely used in applications where objects or situations classification is needed.

#### 2.2.2 Internet of Things

Internet of things refers to the physical items equipped with sensors, processing power, software, and other technologies that link to other systems and devices over the Internet or other communications networks and exchange data with them [4].

#### 2.2.3 Edge computing

Edge computing is a distributed computing framework that brings enterprise applications closer to data sources such as IoT devices or local edge servers [5]. It reduces latency, increases efficiency, and enhances privacy and security by processing data locally instead of relying solely on centralized cloud servers, as shown in Figure 2.1.

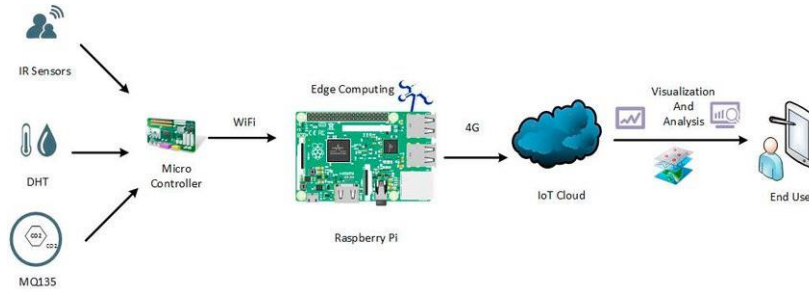


Figure 2.1: Edge Computing Architecture.

### 2.2.4 IoT protocols

IoT protocols are sets of rules and standards that govern the way IoT devices communicate with each other and with other systems over the internet [6]. There are many IoT protocols and standards available, and different projects and use cases might require different kinds of devices and protocols. Some of the most important IoT protocols and standards include MQTT, CoAP, HTTPs.

### 2.2.5 CNNs (Convolutional Neural Networks)

CNN's are specialized neural networks designed for computer vision tasks, utilizing convolutional and pooling layers to extract hierarchical features. Their architecture enables effective pattern recognition, making them widely used in image classification, object detection, and other visual tasks [7]. Transfer learning, where pre-trained models are adapted to new tasks, enhances their performance in diverse applications, as shown in Figure 2.2.

# Convolutional Neural Network

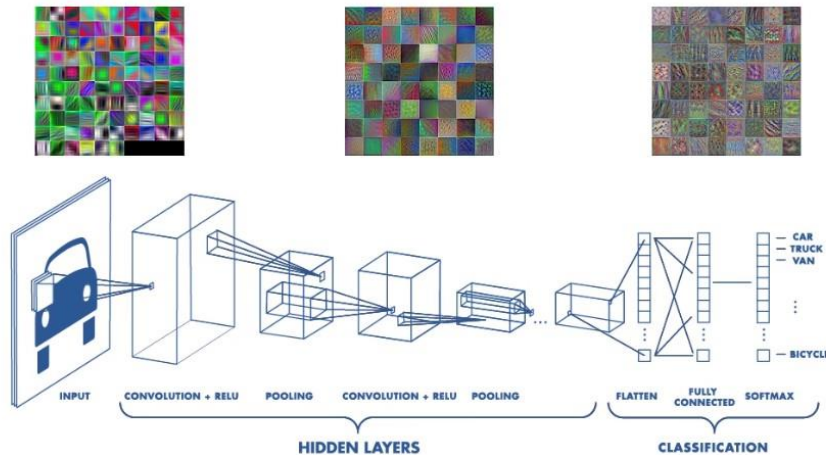


Figure 2.2: Convolutional Neural Network Architecture.

### 2.2.6 Air compressors and pneumatic valves

Air compressors generate compressed air, and pneumatic valves regulate the flow and direction of this air within systems. These valves control the release, passage, and direction of compressed air, playing a crucial role in various pneumatic applications, such as machinery automation and industrial processes [8].

## 2.3 Literature review

The safety of swimming pools has been the one of the most important concern for designers. Many projects proposed different design suggestions to satisfy this requirement, following are discussions of such works.

### 2.3.1 Automated rescue system for children in private home swimming pools

The project discussed in [9], revolves around a safety plate at the pool's bottom that remains inactive when no children are around. When a child approaches, optical sensors trigger the plate to lift via pneumatic cylinders, preventing drowning. An alarm alerts those nearby, and if the family responds, an adult retrieves the child and resets the system, returning it to a standby mode. Adults can either deactivate the safety mechanism to swim, and reactivate it afterward, or keep the system active while swimming. Additionally, users can manually raise the plate for travel or leaving by using a standby switch. This project satisfied all the requirements but it failed to differentiate between human and other objects. It's different from our system regarding the activation trigger. Our system activates the lift ground to lift when a child is actually drowning, alerting the parents when it approaches. Unlike this system that triggers the rescue mechanism when a child is approaching. Studies have shown that children between 1-4 years old, can swim (after suitable training) without adult attendance[10]. So, we believe that the suitable trigger is when a child is drowning. We also consider the case where parents are away from the house or the pool area. A mobile access is needed in these situations. Other differences are related to the hardware and software implementation of the components (Explained in details in Chapter 3).

### 2.3.2 The Design of a Smart Rescue System for Private Swimming Pools

The system mentioned in [11], begins with a Kinect sensor scanning the pool area for human presence. Data collected is sent to a Raspberry Pi, the system's core. The Raspberry analyzes the data, determining human presence, their approximate age based on height, and location. It controls both the alarm unit and the rescue process.

The alarm system has two stages: the first alerts parents when a child is near the pool without adults present, while the second triggers when the child reaches a critical distance from the pool, simultaneously activating the rescue system.

The rescue mechanism comprises four pneumatic cylinders placed at the pool's corners, linked with a net covering the pool's bottom. When engaged, these cylinders raise the net to the water's surface, preventing drowning incidents.

Same as the previous project, the difference between this system and ours is the activation trigger, the mobile access and some components. Table 2.1 lists the more differences between our project and the previous mentioned projects.



Table 2.1: Comparison with other projects

	Automated rescue system for children in private home swimming pools	The Design of a Smart Rescue System for Private Swimming Pools	Our project
Alarm system	Lights and buzzers	Lights and buzzers	1. Lights and buzzers 2. Mobile notifications
Proximity detection	2 columns of 3 optical sensors and transmitters each, placed at each side of the pool	A motion sensor placed around the pool activates the Kinect sensor, which is used to determine the age of the person and their distance to the pool	A motion sensor placed around the pool activates the camera, which is used to determine the age of the person using age determination model .
Activation of rescue system	If at most 2 sensors were triggered, they assume it would be a child, triggering the plate to rise up	If the child is critically close to the pool, the distance is measured by the algorithm	AI model detects drowning. If so, it will be activated.
How to set/reset the system manually	Standby switch	Emergency buttons around the pool	1. Remote Access via mobile 2. Emergency buttons around the pool.

## 2.4 Summary

Training AI models and using computer vision algorithms (i.e. CNNs) will fit to be a good solution in such systems. Previous projects didn't have this implementation (Further explanations will be found in Chapter 3).

# Chapter 3

## System Design

### 3.1 Preface

This chapter provides an overview of the essential hardware and software components intended for our project. It explores various alternatives for each component, presents a conceptual description of the system, and introduces a general block diagram. Additionally, the chapter delves into system algorithms and methodologies through the use of flowcharts. Schematic diagrams depict the interactions and interfaces between components.

### 3.2 Project setup and prototype design

Since the project's real working environment is private swimming pools, we decided to simulate this by building a small prototype to simply deliver the idea and prove the effectiveness of such project.

#### 3.2.1 Pool model

A small pool model will consist of four legs stand holds a (60cm x 50cm x 60cm) plastic box open from the top. This box will not be filled of water in the prototype implementation. Figure 3.1 shows the Prototype design.



Figure 3.1: Prototype design

### 3.2.2 Pool ground

In a fluid, objects experience perpendicular forces due to weak molecular bonds and pool wall interactions. The pressure ( $P$ ) at a submerged object's level is defined as the ratio of force ( $F$ ) to cross-sectional area ( $A$ ). Archimedes' Principle governs the behavior of submerged objects, stating that the buoyant force ( $FB$ ) equals the weight of displaced fluid, acting vertically upward through the object's center of gravity [12].

$$P = \frac{F}{A} \quad (3.1)$$

The buoyant force ( $FB$ ) can be expressed as the difference between upward ( $F_{up}$ ) and downward ( $F_{down}$ ) forces:

$$FB = F_{up} - F_{down} \quad (3.2)$$

Utilizing the pressure definition ( $P = F/A$ ), the buoyant force equation becomes:

$$FB = P_{up} \cdot A - P_{down} \cdot A \quad (3.3)$$

Substituting hydrostatic gauge pressure ( $P_{gauge} = \rho gh$ ) yields:

$$FB = \rho g(h_{up} - h_{down}) \cdot A \quad (3.4)$$

Simplifying, the buoyant force formula becomes:

$$FB = \rho g V_{obj} \quad (3.5)$$

where  $h_{obj}$  is the object's height difference ( $h_{up} - h_{down}$ ) and  $V_{obj}$  is the object's volume. For total submersion in fluid ( $\rho_w$  density of water,  $\rho$  density of fluid,  $\rho_{obj}$  density of the object), the buoyant force is:

$$FB = (\rho_w - \rho_{obj}) \cdot g V_{obj} \quad (3.6)$$

We have considered various types of materials for designing the metal plate that will be placed on the pool floor. Table 3.1 compares between different options of the material that can be appropriately used[13].

Table 3.1: Comparison of Aluminum, Iron, and Copper for Underwater Applications

Property	Aluminum	Iron	Copper
<b>Corrosion Resistance</b>	Excellent due to oxide layer	Requires protective coatings	Good, forms a protective patina
<b>Density</b>	Lightweight	Denser than aluminum	Higher density compared to aluminum
<b>Strength</b>	Good strength-to-weight ratio	Strong but denser than aluminum	Less strong than iron, good overall strength

From equation 3.6, the more dense the object is, the more force is needed to lift it up. So, in conclusion, aluminum will be the best choice for the plate since it's strong, doesn't rust and less dense than other metals. This plate will be perforated to prevent water being displaced each time the plate's lifted up. Figure 3.2 shows the perforation pattern.

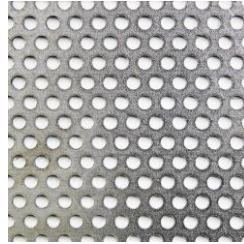


Figure 3.2: Perforated aluminium plat

### 3.2.3 Cylinders

Table 3.2 compares different methods for lifting a net to the surface. Some use electric motors, others use hydraulic systems. However, we will use a pneumatic system in our system. Pneumatic motors are safer than electric motors, especially in water near people. If there's a leak in the pneumatic system, it won't cause harm. In hydraulic systems, leaks could contaminate the water with oil. Pneumatics are also simpler, cheaper, and do the lifting job well.

Table 3.2: Differences between cylinders

Cylinders			
Feature	Hydraulic Cylinders [14]	Pneumatic Cylinders [15]	Electric Cylinders [16]
Working Principle	Use hydraulic fluid for motion	Use compressed air for motion	Use electric power for motion
Complexity	Medium complexity	Simplest complexity	Medium/High complexity
Control	Smoother and more controllable motion	Quick response times	Precise control and positioning capabilities
Speed	Generally lower speed due to fluid viscosity	Generally higher speed due to air compressibility	Variable speed capabilities
Cost	Low cost	High cost	High cost

### 3.2.4 Pneumatic valve

#### 3.2.4.1 5/2 way single solenoid

A 5/2-way single solenoid valve, as shown in Figure 3.3, is a pneumatic control valve with five ports and two positions, commonly used for controlling the direction of airflow in a system[17]. The direction of the airflow is controlled by an electrical signal. If that signal is HIGH, the direction will be on one side. But, if the signal is LOW, the direction of the flow will be on the other side.



Figure 3.3: 5/2 way single solenoid

### 3.2.4.2 5/2 way double solenoid

A 5/2-way double solenoid valve, shown in Figure 3.4, is another type of pneumatic control valve. But, it has two control signals, each directs the airflow to a single direction.



Figure 3.4: 5/2 way double solenoid

We will use 5/2 way single solenoid because the interfacing will require less connections. (One connection is enough to control the valve).

## 3.3 System components and design alternatives




This part describes the hardware and software components and their design alternatives.

### 3.3.1 Hardware components

#### 3.3.1.1 Controllers

As previously noted, a microcontroller is required to oversee and regulate the system. The microcontroller options to be utilized are detailed in Table 3.3.

Table 3.3: Differences between microcontrollers

Microcontrollers			
Feature	ESP32[18]	Raspberry Pi 4b[19]	NVIDIA Jetson Nano [20]
Processor	Dual-core Tensilica LX6, 240 MHz	Quad-core ARM Cortex-A72, 1.5 GHz	Quad-core ARM Cortex-A57, 1.43 GHz
RAM	520KB SRAM	2GB, 4GB, or 8GB LPDDR4 RAM	4GB LPDDR4 RAM
Storage	Up to 16 MB Flash	MicroSD card slot	16 GB eMMC, MicroSD card slot
GPU	None	Broadcom VideoCore VI	128-core NVIDIA Maxwell GPU
Wireless Connectivity	Integrated Wi-Fi, Bluetooth	Wi-Fi, Bluetooth	Gigabit Ethernet
Camera Interface	Yes, multiple options	CSI-2 interface	MIPI CSI-2, D-PHY 1.1
Number of Pins	18 x 12 bits ADC input channels	40 pins	40 pins
I/O Ports	GPIO, I2C, I2S, SPI, UART, PWM	GPIO, I2C, I2S, SPI, UART, PWM	GPIO, I2C, I2S, SPI, UART, PWM
Power Supply	5V via USB or LiPo battery	5V/3A via USB-C	5V/4A via power adapter
Images			

Using the ESP32 and Raspberry Pi 4B together for the project brings a good mix of features. The ESP32 is known for being budget-friendly and energy-efficient. It's good at connecting with sensors, managing small tasks, and making things communicate well. This helps in sending data smoothly between different devices. On the other hand, the Raspberry Pi 4B is powerful and can handle complex tasks, like processing artificial intelligence. It's a great choice for doing things like detecting drownings using advanced computer vision. By combining the ESP32 and Raspberry Pi 4B, we create a system that benefits from both cost-effectiveness and processing power, making a smart private swimming pool system that works efficiently.

In other words, ESP32 will be connected to the motion sensor, cylinders' valves and the alert components (i.e lights and buzzers). Sensors and actuators require small processing power that ESP32 will be a good fit for. Raspberry PI will be the core of the system that controls the ESP32s, receives heavy streams from the cameras, process them and apply the needed AI algorithms.

### 3.3.1.2 Age identifier

The age identifier differentiates between children and adults in determining whether to activate or not the alert system. We have many hardware and software options to evaluate for this part of the system.

#### 3.3.1.2.1 Hardware options

**Kinect sensor V2** : The Kinect sensor, shown in Figure 3.5, determines a person's height by utilizing infrared technology to measure the distance between the sensor and key points on the individual's body, such as the head and feet. This depth perception allows for an accurate calculation of the person's height. Studies have shown that children under 14 years old's height is

under 159 cm[21]. After processing the readings of the Kinect and determining the height of the person approaching. Any height under that number will be considered a child.



Figure 3.5: Kinect sensor V2

**TCRT5000L IR reflective sensor** : This sensor, shown in Figure 3.6, sends IR beams and detect if a beam reflects. This can be calibrated to detect the height of the person, with further calculations, its age. A bunch of this sensor can be aligned vertically and when a person approaches, we predict the height of it by counting the number of sensors that detected a reflected beam.



Figure 3.6: TCRT5000L sensor

#### 3.3.1.2.2 Software options

We found many software algorithms that can be used as an age identifier. Table 3.4 differentiates between these algorithms.

**Age Detection using OpenCV in Python** : A pre-trained model that detects faces and identify the age of each person detected[22]. This model classifies the age of the detected person into the following classes: 0-2, 4-6, 8-12, 15-20, 25-32, 38-43, 48-53, 60-100

**Age identifier based on height** : Using mediapipe project in python, we can generate a skeleton image of the detected person. And with some calculations of the x,y,z coordinates of eyes and toes, we can identify the height. As mentioned before, from the height, we can predict the age.

We've decided to use deep learning for age identification due to its accuracy, as the hardware option sometimes provides inaccurate results. Accurate height calculations require depth cameras. Due to its high cost, we decided to eliminate this option.

#### 3.3.1.3 Proximity detector

Age identifying camera send image stream to the controller. So, to reduce the traffic and the processing time, we can activate the camera only when needed. Meaning that, in case of a person approaching the camera will be activated to send the stream to the controller to be processed.

Otherwise the camera will remain idle. Proximity detector can be implemented by using a sensor to detect motion, and placing that sensor strategically around the pool. Table 3.5 shows the differences between the sensors we've considered to implement this part of the system.

Table 3.4: Differences between motion detectors

Motion detector		
Type	Analog/Digital	Delay
Passive Infrared(PIR) sensor	digital	0.3s - 200s
Ultrasonic sensor HC-SR04	Analog	few milliseconds

Ultrasonic sensor is conventionally a distance sensor. But, it can be calibrated as a motion sensor by constantly reading the distance and if the distance drops under a certain level, a moving object is assumed to be the cause of that change. Because of The high delay of the PIR sensor makes it a bad choice for our application. The two motion detectors are shown in Figure 3.7.

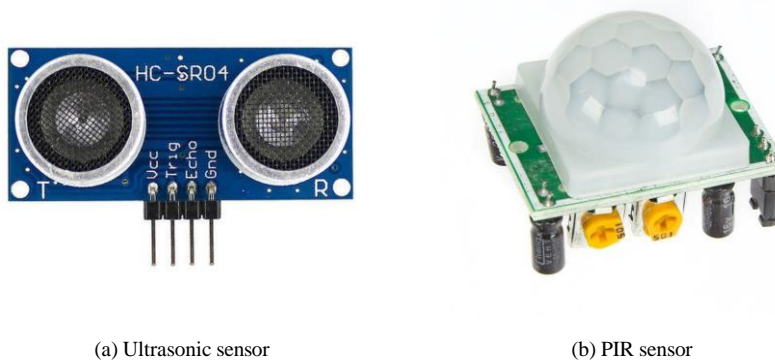


Figure 3.7: Motion detector options

### 3.3.1.4 Camera

The camera is essential for real-time monitoring, drowning detection in the smart private swimming pool system, ensuring timely responses to potential risks and emergencies. The camera options to be utilized are detailed in Table 3.6.

Table 3.5: Differences between cameras

Cameras		
Feature	Mini Action A7 [23]	ESP32 Camera [24]
Image Sensor	CMOS	OV2640
Video Resolution	720p@30fps, 1080p@15	1080p@30fps, 720p@60fps, 480@90fps
Compatibility	Compatible with Raspberry Pi models	Compatible with ESP32 boards
Price	\$10	\$10-\$20



Selecting the Mini Action A7 for our project offers a budget-friendly and seamlessly integrated solution, empowering the system with video capabilities for drowning alerts, and remote monitoring in the smart private swimming pool setup.

### 3.3.1.5 Alert

For an alert system designed to notify parents when their child is near the pool or has fallen into the pool, a combination of various alert mechanisms would be ideal to ensure that parents receive timely and effective notifications. Here's a suggested approach:

- **Mobile Notifications:** It utilize mobile notifications to send real-time alerts to the parents' smartphones. These notifications can include a brief message describing the situation, such as "Child near pool" or "Emergency: Child in pool."
- **Sound Alerts:** The alert sound should be distinct and attention-grabbing to ensure it is not easily overlooked.
- **LED Notifications:** Uses LED notifications, they are visual indicators, providing an additional layer of alerting, especially in situations where sound may not be heard or could be disruptive.

### 3.3.1.6 Air compressor

An air compressor, shown in Figure 3.8, is a device that transforms electrical power, utilizing an electric motor, into potential energy stored in pressurized air (i.e., compressed air) [8]. It will supply pneumatic power to our system, enabling it to lift the metal plates.



Figure 3.8: Air compressor

### 3.3.1.7 Power Supply

A power supply, shown in Figure 3.9, is an electrical apparatus that delivers electric power to electronic devices, such as laptops, servers, or other electronic equipment. Its primary role is to transform electric current from a source into the appropriate voltage, current, and frequency required to operate the device. This conversion may involve changing from AC to DC or DC to DC. Pneumatic valves require 12V DC to operate, so a power supply will satisfy this need [25].



Figure 3.9: Power supply

### 3.3.1.8 Relay

A relay, shown in Figure 3.10, used to connect or disconnect a circuit by using an electrical signal to control an electromagnet, which in turn connects or disconnects another circuit [26]. Raspberry Pi 4's output voltage is 5V (less than the required voltage to control a pneumatic valve). Because we used a power supply, the relay allows us to control the connection between the power supply and the control port of the valves by the output of the Raspberry Pi.



Figure 3.10: Relay device

### 3.3.1.9 Push button

A push button switch, shown in Figure 3.11 is a mechanical device used to control an electrical circuit in which the operator manually presses a button to actuate an internal switching mechanism. They come in a variety of shapes, sizes, and configurations, depending on the design requirements[27].



Figure 3.11: Push button switch

### 3.3.1.10 Piezoelectric Buzzer

A piezoelectric buzzer, shown in Figure 3.12, is an audio signaling device like a beeper. The main function of this is to convert the signal from audio to sound. It is powered through DC voltage[28]. We will use this device as a sound notification in our alert system.



Figure 3.12: Buzzer piezosounder

## 3.3.2 Software components

### 3.3.2.1 Drowning detector

Drowning detector is the trigger that activates the rescue system in case of a drowning situation. This detector has to be efficient and reliable in the decision making. So, we have listed different options to use as a detector.

#### 3.3.2.1.1 Custom trained CNN(Convexional Neural Network)

CNNs, are automatically learning hierarchical representations of features from data, significantly reducing the need for manual feature extraction, especially in the context of images and videos [7]. Primarily used in classification tasks, a CNN is trained on datasets containing images or videos belonging to specific classes. After training, the model can accurately classify new inputs of the same type (image or video) and provide a percentage indicating the accuracy of its predictions. This capability makes CNNs widely employed in computer vision applications, where they demonstrate proficiency in tasks such as image classification, object detection, and video analysis.

#### 3.3.2.1.2 YOLO

A Python project on GitHub implemented a drowning detection algorithm using YOLO [29]. YOLO is one of the most popular object detection algorithms, as shown in Figure 3.13, It aims to predict a class of an object and the bounding box that defines the object location on the input image[30]. This project's developer, after doing some researches, they discovered that in case of drowning, the person tends to stand in a vertical position trying to resist falling down. Therefore, their algorithm detects the person once they enter the pool, storing their position in the frame, in order to detect when that person is falling by doing calculations on the change of position with respect to the initial stored one.

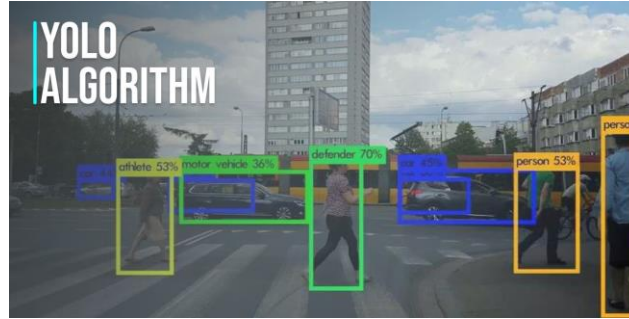


Figure 3.13: Yolo Algorithm

Our choice of the algorithm to be used to implement the drowning detection in our system has to be reliable, efficient and easy to implement and apply. Using YOLO doesn't satisfy the reliability condition. The assumed conditions of the drowning situation are not always found when the person drowning is a child. Children might not understand that they are actually drowning and try to fight that as the algorithm proposed. So, our choice will be a custom trained CNN that's fed videos of people drowning, taken from different angles, and be trained on. We chose the custom trained CNNs. The name of the model is: Drowning Detection And Prevention In Swimming Pools[31]

### 3.3.2.2 Communication protocols

Protocols are essential in our project for facilitating communication and data exchange between various components, ensuring seamless integration and functionality. The chosen protocols, including MQTT, CoAP and HTTPS, are listed in Table 3.7 to illustrate their respective features and help in making informed decisions based on project requirements and constraints.

Table 3.6: Differences between IoT protocols

Feature	MQTT [32][33]	HTTPS [32][33]	CoAP [34]
Type	Publish-Subscribe	Request-Response	Request-Response
Use Case	IoT, Messaging	Secure Web Communication	IoT
Connection Type	Lightweight Persistent	Persistent	Lightweight
Reliability	Quality of Service (QoS) Reliable	Reliable	Confirmable/Non-confirmable
Overhead	Low	High	Low
Message Ordering	May be unordered	Ordered	Unordered
Security	Yes, with features like authentication and encryption	Yes, with SSL/TLS	Yes, with DTLS
Latency	Low	Higher due to handshake and encryption	Low
Use in Web Browsers	No	Yes	Yes

For our project, MQTT is recommended between nodes due to its lightweight, asynchronous communication model, low overhead, reliability, security features, and scalability, making it well-suited for real-time alerts and multi-node communication.

### 3.3.2.3 Brokers

In our project, a broker is essential to facilitate communication between the various nodes, ensuring seamless data exchange and coordination. The broker acts as a mediator, managing the flow of information between devices, sensors, and controllers in the smart private swimming pool system. In Table 3.8, we will compare three types of brokers—AWS IoT Core, HiveMQ, and Mosquitto—based on key features and suitability for our project requirements.

Table 3.7: Differences between brokers

Feature	AWS IoT Core [35]	Mosquitto [36]	HiveMQ [36]
Cloud Provider	Amazon Web Services	N/A (Local)	N/A (Local)
Ease of Use	User-friendly dashboard	User-friendly GUI	User-friendly GUI
Security Features	Advanced Security Features	Basic Security Features	Advanced Security Features
Authentication	IAM, X.509 Certificates	Username/Password	Various Authentication Methods
Device Management	Yes	Limited	Yes
Pricing Model	Pay-as-you-go	Open Source	Open Source

For our project, We chose Mosquitto broker for its combination of cost-effectiveness and advanced security features, offering various authentication methods to enhance the overall security of our project.

## 3.4 Conceptual system design

Our system will be designed to save children from drowning in private pools using sensors and automated rescue measures. The system's core processor is a Raspberry Pi 4 microcontroller connected to software-based age identification model, an alert unit featuring a buzzer and LEDs, and an AI model for drowning detection in camera streams above the pool with alerts sent to parents through a mobile application as shown in the general block diagram of the system in Figure 3.14. The ultrasonic sensor will constantly be reading the distance to the nearest object around it. When a person approaches, the distance read by the sensor will drop to under 50 cm. this change will trigger the camera connected to the Raspberry Pi to identify the person's age. If the identified age is under 12 years old, the Raspberry Pi will publish a message to the ESP32 connected to the lights and buzzers, activating them. At the same time, the mobile application receives the alert message and notify parents via app notifications. Meanwhile, the raspberry will be receiving stream from the camera, it will be applying the computer vision algorithm that detects drowning. If a drowning situation is detected, the system will set a 10-seconds timer which is 50% of the maximum time that the child must be rescued within[2]. At the same time, raspberry will send to the mobile API to send an alert message to parents' mobiles. After the timer goes off, if the person is still drowning, the system will activate the rescue mechanism in which the raspberry will send to the ESP32s that are connected to the valves to activate the cylinders to rise upward. Manual set/reset can be obtained by the buttons connected to the raspberry, as shown in the system conceptual diagram in Figure 3.15.

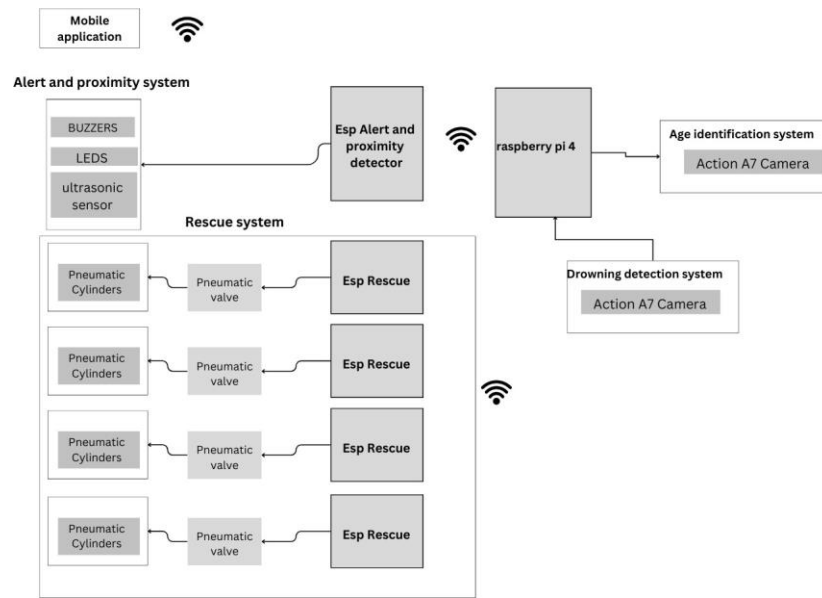


Figure 3.14: System block diagram

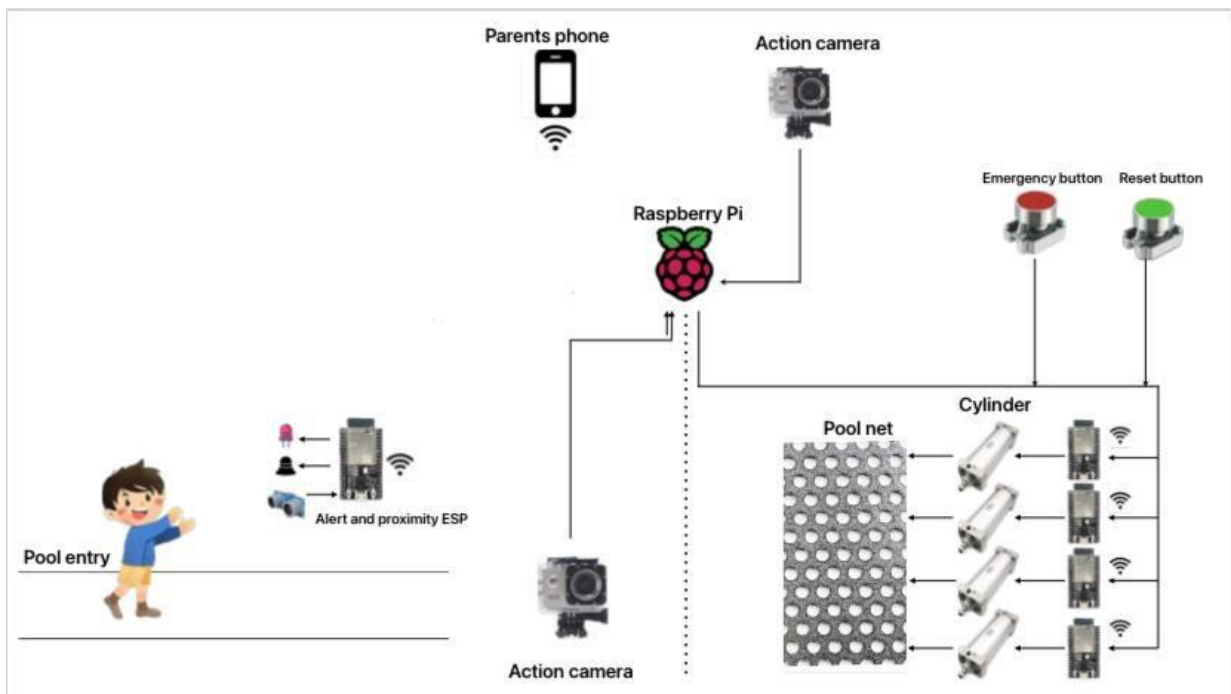


Figure 3.15: System conceptual diagram

## 3.5 Algorithms and methodologies

### 3.5.1 Proximity detection pseudocode

This pseudocode use ultrasonic sensors to measure distance as shown in Figure 3.16 . If the distance is below a certain threshold and the individual is a child, trigger the alert system.

---

```

while true do
  distance = measureDistance()
  if distance ≥ 50cm then
    | continue
  else
    | age = getAge()
    | if age ≤ 12 then
    | | sendAlert()
    else
    | | continue
    end
  end
end
end

```

---

Figure 3.16: Proximity detection algorithm

### 3.5.2 Drowning detection pseudocode

This pseudocode captures frames from the camera as shown in Figure 3.17 , send it to raspberry pi to make the required process and makes a decision based on the process

---

```

while true do
  start=time.now
  detections=[]
  while time.now - start ≤ 10 do
    | videoStream = captureStream()
    | detections.append(drowningOrSwimming(videoStream))
  end
  if drowningCountPercentage(detections) > 0.5 then
    | alert()
    | rescue()
  end
end
end

```

---

Figure 3.17: Drowning detection algorithm

## 3.5.3 Algorithms

### 3.5.3.1 Age Detection using OpenCV in Python

This section describes the steps and processes involved in implementing an age identification model , they are:

1. Setup and Imports
  - Import necessary libraries: OpenCV, NumPy, and others required for image processing and model handling.
2. Load Pre-trained Models

- Load the pre-trained age prediction model "ageNet" and face detection model (faceNet) using OpenCV's `dnnReadNetFromCaffe()` method.
- Specify the model architecture and weights files for both the age prediction and face detection models.

### 3. Image Preprocessing

- Read the input image using `cv2.imread()`.
- Prepare the image for face detection by resizing and converting it to a blob using `cv2.dnn.blobFromImage()`.

### 4. Face Detection

- Pass the blob through the face detection model to obtain the detections.
- Loop through the detected faces, extract the face ROI (Region of Interest), and prepare it for age prediction.

### 5. Age Prediction

- Convert the extracted face ROI to a blob and pass it through the age prediction model.
- Obtain the age predictions, which are usually in the form of age ranges (e.g., 0-2, 4-6, 8-12, 15-20, 25-32, 38-43, 48-53, 60-100).

### 6. Display Results

- Draw bounding boxes around detected faces and annotate them with the predicted age ranges on the original image.
- Display the final image with annotations using `cv2.imshow()`.

### 7. Cleanup

- Release resources and close any open windows using `cv2.destroyAllWindows()`.

### 8. Pseudo Code As shown in Figure 3.18 the pseudo code for age identification model using python and OpenCv.



```
# Load Libraries
import necessary libraries
# Download and Load Models
download/locate pre-trained models, load using OpenCV DNN
# Initialize Video Capture
set up video capture (webcam or video file)
# Process Video Frames
while capturing frames: read frame, convert to blob, detect faces
for each face:
extract, convert to blob,
predict age
# Display Results
annotate frame with age and bounding boxes, display frame
# Exit Conditions
check for exit (e.g., key press)
# Release Resources
release capture, close windows
```

Figure 3.18: Age identification model algorithm

### 3.5.3.2 Drowning detection model

This section describes the steps and processes involved in implementing a drowning detection model, they are :

#### 1. Initialization

- Import necessary libraries including OpenCV, Paho MQTT, and functions for model inference.
- Set up exception handling to catch and print any exceptions that occur
- Initialize MQTT client for communication.

#### 2. Model Configuration

- Specify the name and version of the Roboflow model to be used.
- Set up the camera source, typically a webcam or a video stream.
- Retrieve the Roboflow model using the provided API key.

#### 3. Detection Loop

- Continuously capture frames from the camera.
- Run inference on each frame using the Roboflow model.
- Process the inference results:
  - If the model predicts a drowning event, draw a bounding box around the detected object and label it accordingly.

- Update the output list with the current prediction.
- Display the annotated frame in a window.
- If the user presses 'q', exit the loop.

#### 4. Result Analysis

- Count the occurrences of "drowning" and "no prediction" in the output list.
- Calculate the number of valid predictions (excluding "no prediction").
- Determine if the proportion of "drowning" predictions exceeds a certain threshold ,publish a message through MQTT indicating a possible drowning event.

#### 5. Cleanup

- Release the camera and close all OpenCV windows.

#### 6. Pseudo Code As shown in Figure 3.19 the pseudo code for drowning detection model.

```

# Load Libraries
import necessary libraries (cv2, time, sys, traceback,)
(paho.mqtt.client, get_roboflow_model)
# Set Up Exception Handling
define exception handling function, set sys.excepthook
# Download and Load Model
download/locate pre-trained drowning detection model using get_roboflow_model
# Initialize Video Capture
set up video capture (webcam or video file)
# Process Video Frames
while capturing frames: read frame, preprocess frame, detect drowning using model
# Annotate and Display Results
annotate frame with detection results, display frame
# Alert System
if drowning detected (e.g., count exceeds threshold):
    trigger alert (e.g., send MQTT message)
# Exit Conditions
check for exit (e.g., key press)
# Release Resources
release capture, close windows
# Publish Results

```

Figure 3.19: Drowning detection model algorithm

### 3.5.4 Sequence diagrams

Figure 3.20 shows the sequence diagram of the proximity detection system. Figure 3.21 shows the sequence diagram of the rescue system.

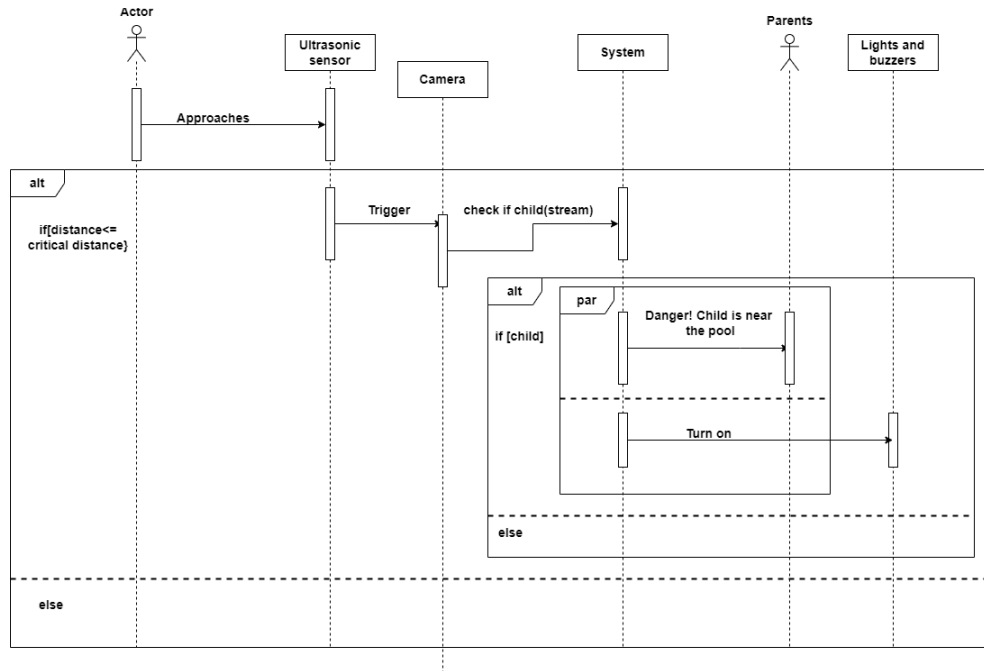


Figure 3.20: Sequence diagram of proximity detection system

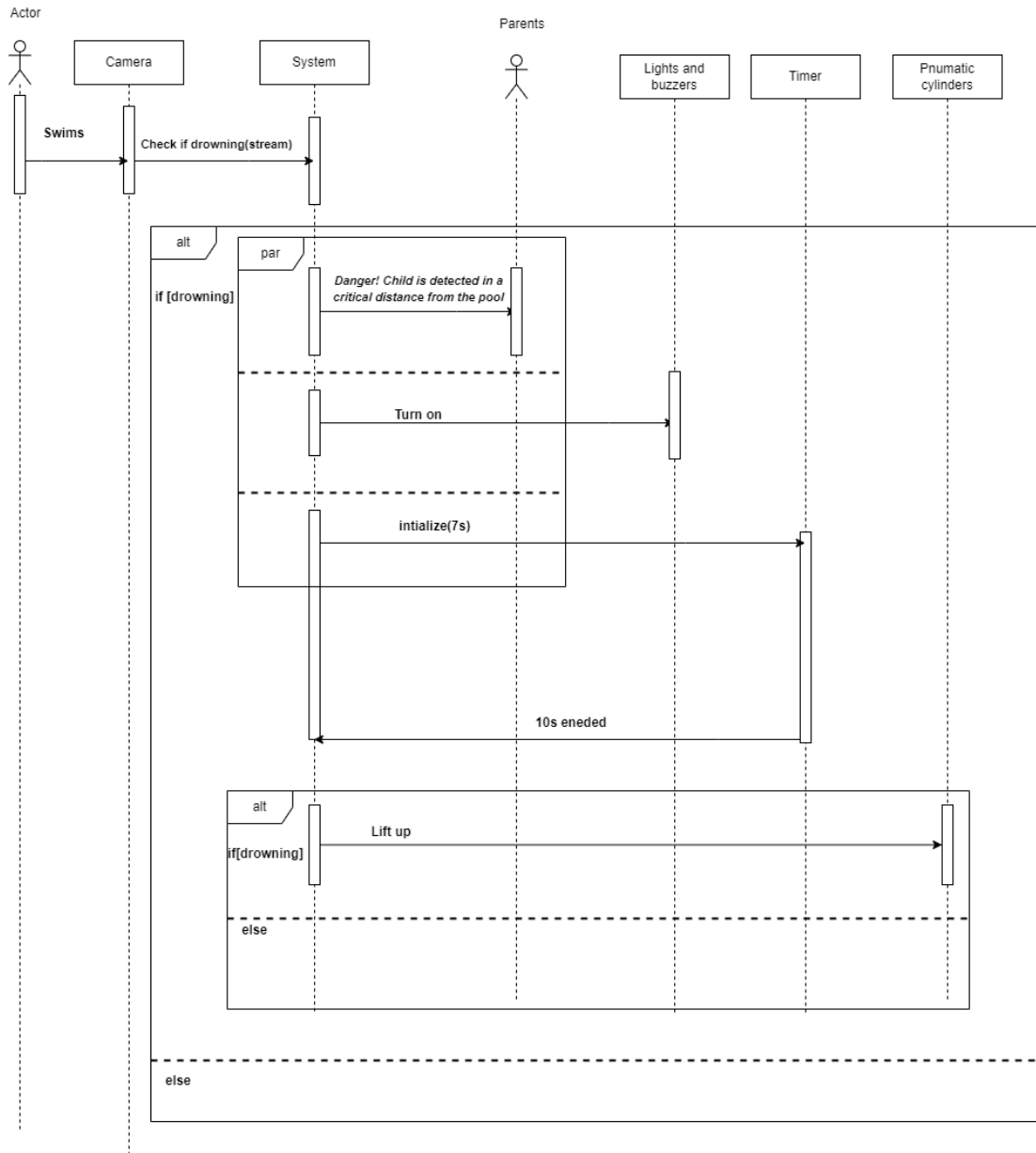
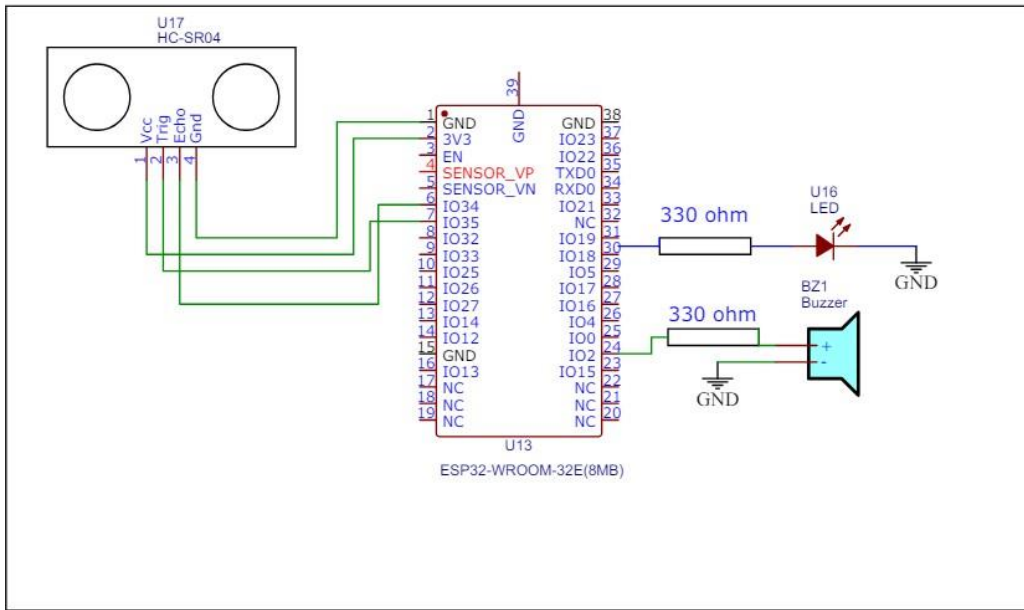


Figure 3.21: Sequence diagram of rescue system

### 3.6 Schematic diagram

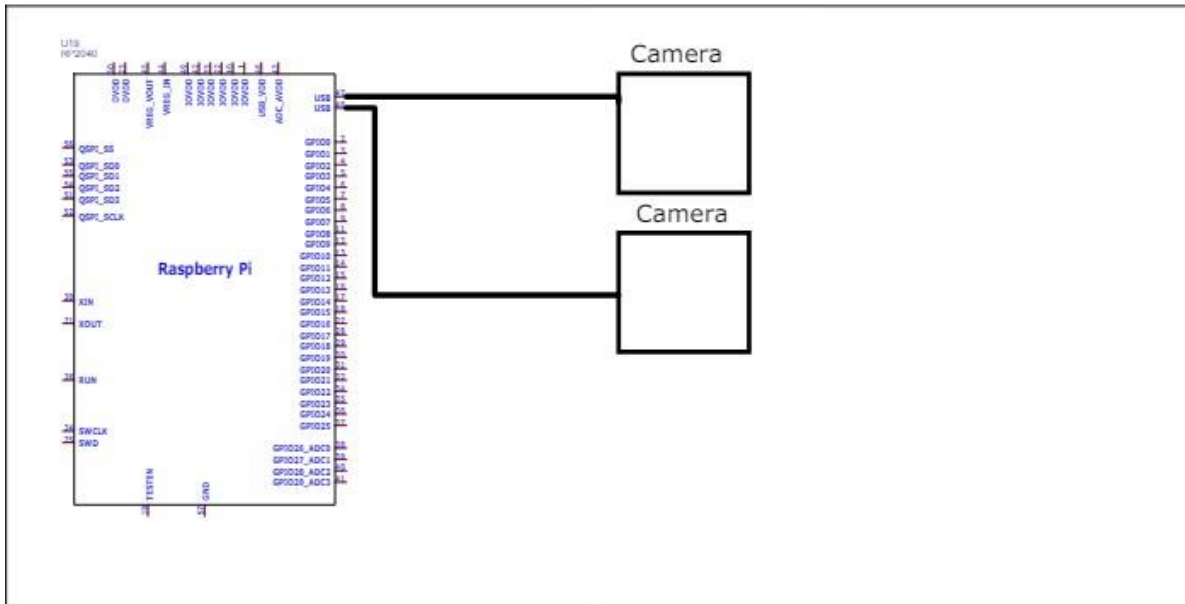
In Figure 3.22 shows the connection between the main component outside the pool, We have connected the alert and proximity system using an ESP microcontroller to create a proximity detector and alert mechanism as shown in Figure 3.22-a We have connected the age identification camera and drowning detection camera to the Raspberry Pi 4. The cameras are interfaced with the Raspberry Pi 4 as shown in Figure 3.22-b. In Figure 3.24 shows the connection between the main components inside the pool.

Proximity and Alert components



(a) Proximity and alert esp

Raspberry PI connections



(b) Raspberry pi4 connection

Figure 3.22: Schematic diagram of outside-pool components

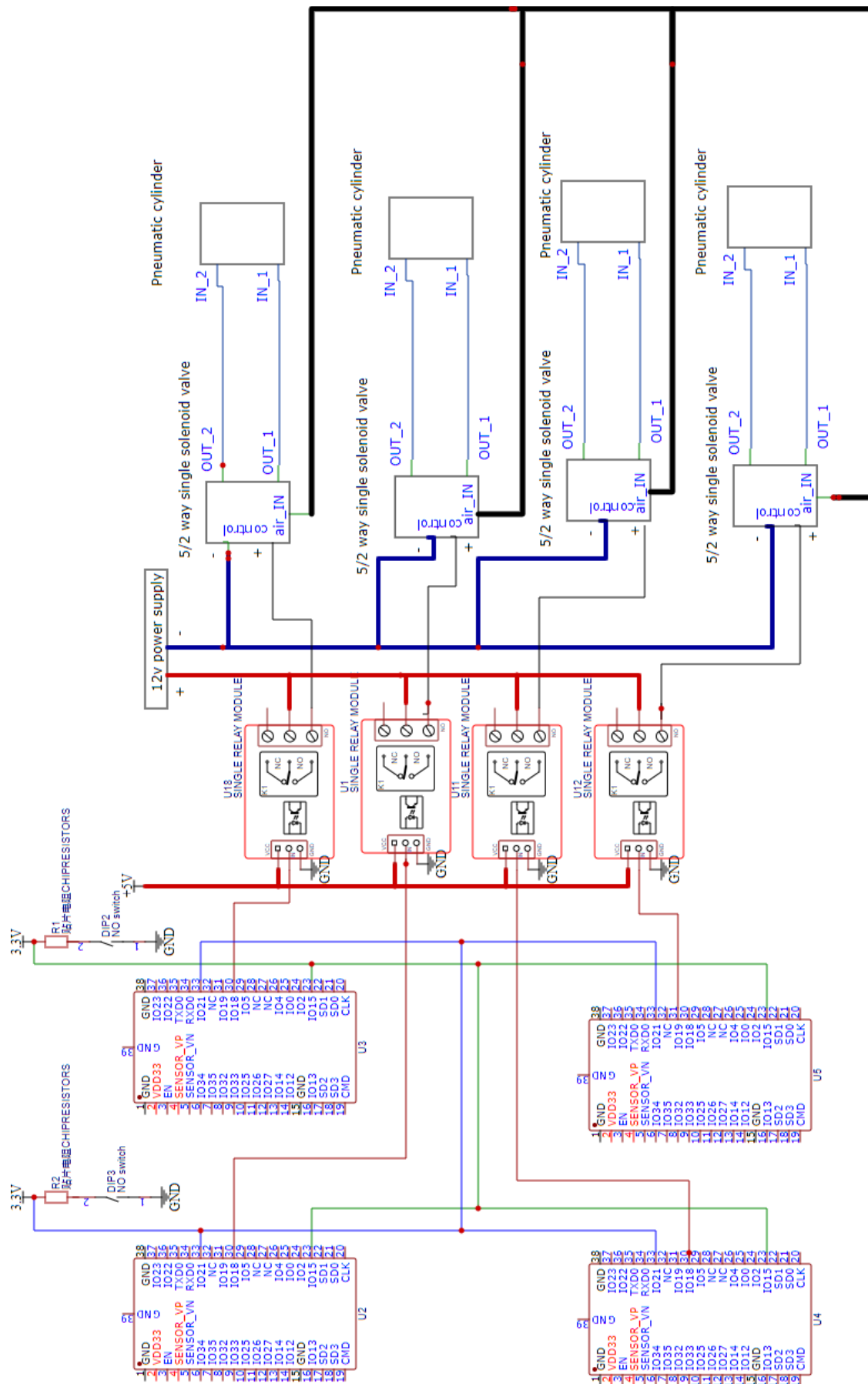


Figure 3.23: Schematic diagram of inside-pool components

### **3.7 Summary**

In this chapter, we have discussed the system hardware and software components with their alternatives. The conceptual description of the system and the general flow of the system with all necessary diagrams are presented, too.

# Chapter 4

## Implementation

### 4.1 Preface

This chapter provides an overview of the software and hardware implementation, issues and challenges related to the implementation.

### 4.2 Hardware implementation

In the previous chapter, we showed how the components interconnect with each other. This section will present how we assembled these components and how we connected them with the ESP32s and Raspberry Pi 4.

#### 4.2.1 Prototype setup

Our prototype features a metal table designed to support the cylinders, their controlling circuits, and all pool components, as depicted in Figure 4.1. The pool prototype itself is a rectangular prism box made of plastic. Additionally, a rectangular perforated aluminum plate is used to prevent water from being raised with the rescued person by the rising net, the bottom side of the box was perforated to allow the cylinders to rise through.

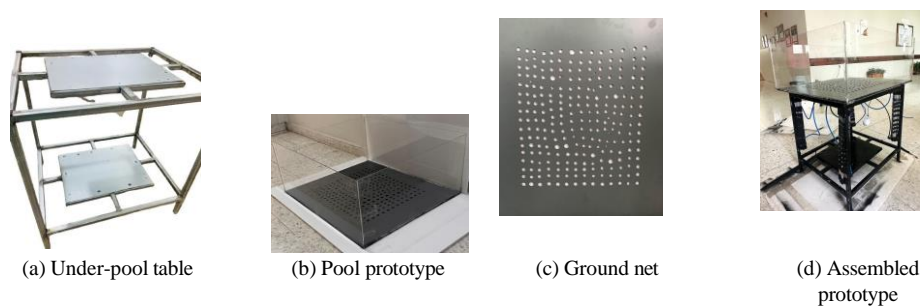


Figure 4.1: Prototype components

#### 4.2.2 Alert and proximity detector

We connected the ultrasonic buzzer and LED to the same ESP device, which serves as the alert and proximity detector unit placed at the pool entry. This setup is detailed in the schematic diagram in the design chapter, as shown in Figure 4.2.



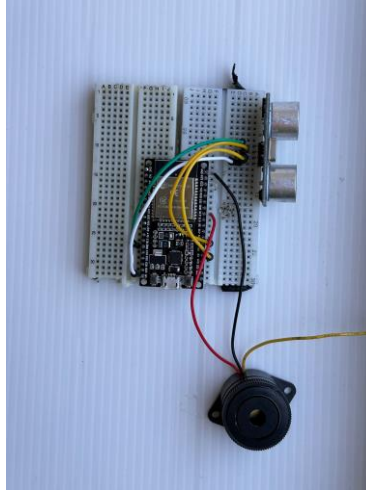


Figure 4.2: Alert and proximity detector connections

### 4.2.3 Rescue System

Our rescue system consists of 3 main units: Controlling and interfacing circuits, cylinders and pneumatic supply for the system.

#### 4.2.3.1 Controlling and interfacing circuits

We used four 5/2-way single solenoid valves as the interfacing components, each connected to its own ESP32 module and relay to ensure high system availability. Each ESP32 module controls one valve, and they are placed on the underside of the table As shown in Figure 4.3.

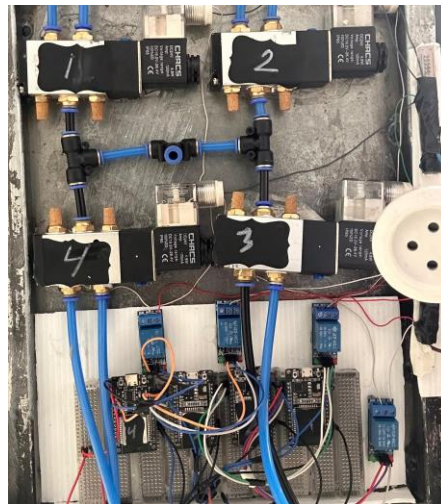


Figure 4.3: Rescue system connections

#### 4.2.3.2 Cylinders setup

We placed each cylinder on a table leg to ensure proper airflow, as shown in Figure 4.4.



Figure 4.4: Cylinders setup

#### 4.2.3.3 Pneumatic supply

We connected the pneumatic supply to a pneumatic throttle to control the net raising speed, as shown in Figure 4.5.



Figure 4.5: Pneumatic supply

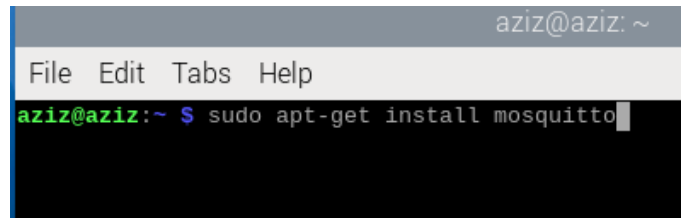
### 4.3 Software implementation

#### 4.3.1 Server setup

Our system is implemented to intercommunicate through MQTT protocol. Raspberry pi will host the Mosquitto broker. For this to happen, some configurations must be made.

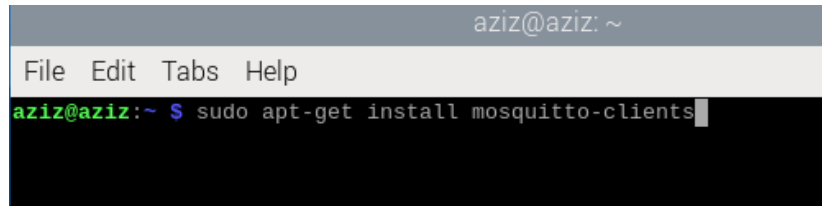
##### 4.3.1.1 Mosquitto broker installation and configuration

As mentioned in Chapter 3, we chose Mosquitto broker to establish the MQTT communications. Figure 4.6 shows the installation process on the Raspberry Pi.



```
aziz@aziz: ~  
File Edit Tabs Help  
aziz@aziz:~$ sudo apt-get install mosquitto
```

(a) Mosquitto

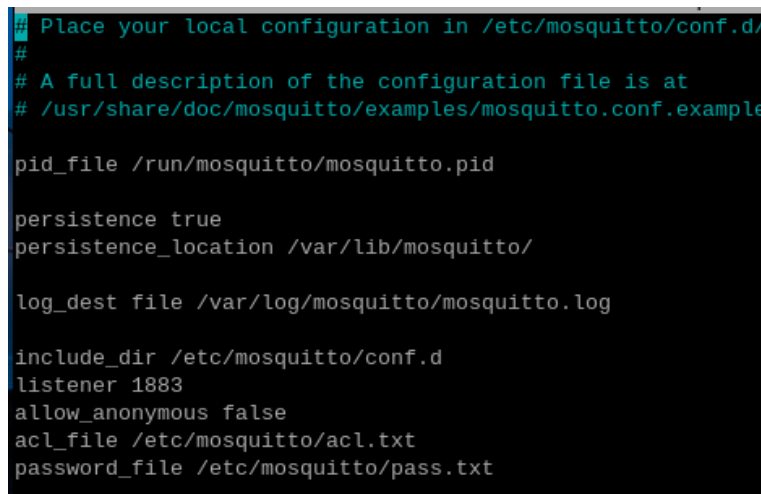


```
aziz@aziz: ~  
File Edit Tabs Help  
aziz@aziz:~$ sudo apt-get install mosquitto-clients
```

(b) Mosquitto clients

Figure 4.6: Mosquitto installation

Figure 4.7 shows the configuration file of the broker.



```
## Place your local configuration in /etc/mosquitto/conf.d  
#  
# A full description of the configuration file is at  
# /usr/share/doc/mosquitto/examples/mosquitto.conf.example  
  
pid_file /run/mosquitto/mosquitto.pid  
  
persistence true  
persistence_location /var/lib/mosquitto/  
  
log_dest file /var/log/mosquitto/mosquitto.log  
  
include_dir /etc/mosquitto/conf.d  
listener 1883  
allow_anonymous false  
acl_file /etc/mosquitto/acl.txt  
password_file /etc/mosquitto/pass.txt
```

Figure 4.7: Mosquitto configurations file

For example: This broker is configured to run on TCP port 1883. Other parameters will be explained in the next section.

### 4.3.1.2 Security implementation

Mosquitto allows system admins to secure the access to the broker and topics via two processes.

#### 4.3.1.2.1 Authentication

Mosquitto can be configured to prevent anonymous clients to connect to the broker. As shown in Figure 4.9

```
## Place your local configuration in /etc/mosquitto/conf.d/
#
# A full description of the configuration file is at
# /usr/share/doc/mosquitto/examples/mosquitto.conf.example

pid_file /run/mosquitto/mosquitto.pid

persistence true
persistence_location /var/lib/mosquitto/

log_dest file /var/log/mosquitto/mosquitto.log

include_dir /etc/mosquitto/conf.d
listener 1883
allow_anonymous false
acl_file /etc/mosquitto/acl.txt
password_file /etc/mosquitto/pass.txt
```

Figure 4.8: Disable anonymous connections to the broker

So, in order to connect to the broker, the client has to determine its name and password. Each publish/subscribe message the client sends, Mosquitto checks if the user is registered and it validates the password. To add users in Mosquitto, the path of the username/password file has to be set in the configuration file, as shown in Figure 4.9.

```
## Place your local configuration in /etc/mosquitto/conf.d/
#
# A full description of the configuration file is at
# /usr/share/doc/mosquitto/examples/mosquitto.conf.example

pid_file /run/mosquitto/mosquitto.pid

persistence true
persistence_location /var/lib/mosquitto/

log_dest file /var/log/mosquitto/mosquitto.log

include_dir /etc/mosquitto/conf.d
listener 1883
allow_anonymous false
acl_file /etc/mosquitto/acl.txt
password_file /etc/mosquitto/pass.txt
```

Figure 4.9: Set the path to users/passwords file in Mosquitto

Figure 4.10 shows the process of adding users with passwords in Mosquitto.

```
aziz@aziz:~$ sudo mosquitto_passwd -b '/etc/mosquitto/pass.txt' CYLINDER1 1111
aziz@aziz:~$ sudo mosquitto_passwd -b '/etc/mosquitto/pass.txt' CYLINDER2 2222
aziz@aziz:~$ sudo mosquitto_passwd -b '/etc/mosquitto/pass.txt' CYLINDER3 3333
aziz@aziz:~$ sudo mosquitto_passwd -b '/etc/mosquitto/pass.txt' CYLINDER4 4444
aziz@aziz:~$ sudo mosquitto_passwd -b '/etc/mosquitto/pass.txt' ULTRA-ALERT UA123
aziz@aziz:~$ sudo mosquitto_passwd -b '/etc/mosquitto/pass.txt' MANAGER strong
aziz@aziz:~$
```

Figure 4.10: Creating users in Mosquitto

#### 4.3.1.2.2 Authorization

To give each user limited access to the topics, We have to set the path of an ACL (Access Control List) file in the configurations of the Mosquitto, as shown in Figure 4.11

```
# Place your local configuration in /etc/mosquitto/conf.d
#
# A full description of the configuration file is at
# /usr/share/doc/mosquitto/examples/mosquitto.conf.example

pid_file /run/mosquitto/mosquitto.pid

persistence true
persistence_location /var/lib/mosquitto/

log_dest file /var/log/mosquitto/mosquitto.log

include_dir /etc/mosquitto/conf.d
listener 1883
allow_anonymous false
acl_file /etc/mosquitto/acl.txt
password_file /etc/mosquitto/pass.txt
```

Figure 4.11: Setting the path to the ACL file

In this file, each user is given certain access to whatever topics they need to publish/subscribe to. Figure 4.12 shows the ACL file of our broker.

```
GNU nano 7.2
user CYLINDER1
topic read test/cylinder

user CYLINDER2
topic read test/cylinder

user CYLINDER3
topic read test/cylinder

user CYLINDER4
topic read test/cylinder

user ULTRA-ALERT
topic read test/alert
topic write test/ultra

user MANAGER
topic readwrite #

user app-user1
topic read test/stream/cam1
topic read test/stream/cam2
topic write test/cylinder/control
```

Figure 4.12: Mosquitto ACL file

### 4.3.2 AI models integration

Our system consists of two main AI models, each model has to be calibrated to suit our system requirements.

#### 4.3.2.1 Age identification model

This model uses OpenCV and pre-trained deep learning models to detect faces and estimate their ages in real-time from a video feed. The script loads the face and age detection models, processes frames from a specified camera, and applies the models to detect faces and predict their ages. Figure 4.13 shows the output image of the model.

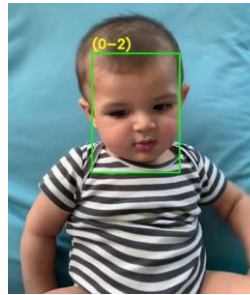


Figure 4.13: Age identification model Output

#### 4.3.2.2 Drowning detection model

This model is a pre-trained model with a dataset to classify objects across 3 different classes (such as "swimming", "drowning", "person"). However, these classes are suitable for our specific project requirements. Figure 4.14 shows a sample output of this model.

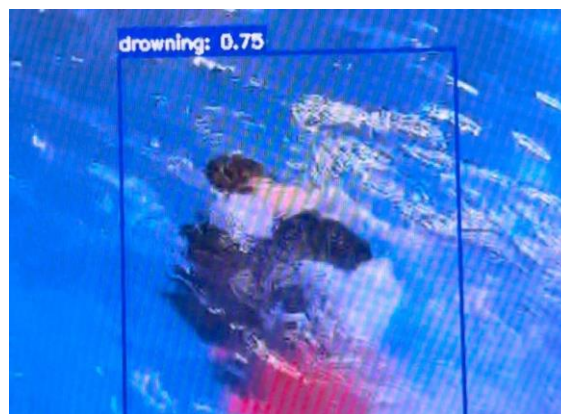


Figure 4.14: Output sample of the drowning detection model

#### 4.3.3 Drowning model dataset

Our model was trained on a dataset specifically curated for detecting drowning people. As shown in Figure 4.15, The dataset is composed of a total of 8925 images, categorized into three distinct sets: 7358 images for training, 1046 images for validation, and 521 images for testing. Each image was selected to represent various conditions to ensure the robustness of the model. The training set enabled the model to learn and identify relevant features, the validation set was used to fine-tune the model, and the test set provided an independent evaluation of the model's performance. This structured approach ensures a comprehensive assessment of the model's accuracy and effectiveness in real-world scenarios.

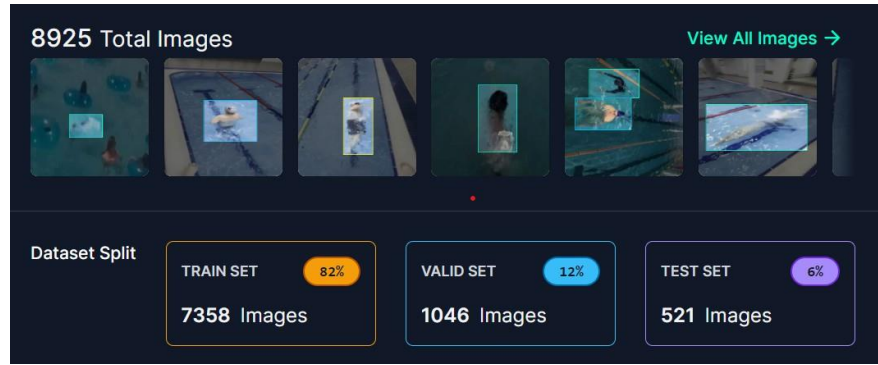


Figure 4.15: downing model dataset

#### 4.3.4 Clients setup

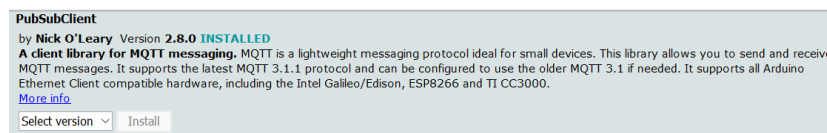
Once we configured Mosquitto and the Raspberry pi, we had to setup the clients of the broker, such as: All the ESP32s of our system, the manager client which integrates the subsystems together and manages the communications between them. Each client is explained in section 4.3.4.1 and 4.3.4.2.

##### 4.3.4.1 ESP32s

Several libraries have to be installed on the Arduino IDE for the ESP32s to function such as MQTT client.

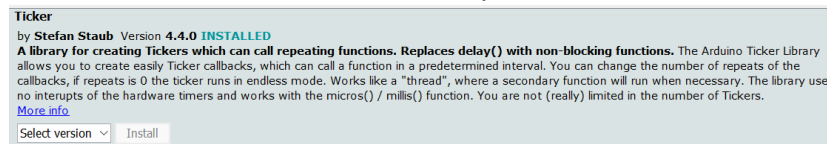
- WiFi: we need this library to access the built-in wifi module in the ESP32s.
- PubSubClient: This library is designed to work with ESP32 devices. With it you can send data to and receive data from MQTT brokers(in our case: Mosquitto).
- Ticker: This library is used to set up a scheduled task to every 5s. This task sends ping signal to the broker to ensure the client is connected.

Figure 4.16 shows the process of installing the libraries on Arduino IDE.



(a)

PubSub library



(b)

Ticker library

Figure 4.16: Libraries installation

Figure 4.17, shows how we defined the credentials of Wifi and MQTT connections in the ESPs. And the callback function that receives the messages from the broker.



```

char* ssid = "SSID";
char* password = "WIFI-PASSWORD";
char* mqttServer = "IP OF BROKER";
int mqttPort = 1883;

#define CHIP_NAME "CLIENT USERNAME"
#define USERNAME "CLIENT USERNAME"
#define PASSWORD "CLIENT PASSWORD"

WiFiClient espClient;
PubSubClient client(espClient);
Ticker timer; |

```

Figure 4.17: Define connection credentials

Figure 4.18 shows the setup function in the ESP32s.

```

void setup() {
    Serial.begin(115200);
    WiFi.begin(ssid, password);

    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.println("Connecting to WiFi");
    }
    Serial.println("Connected to the WiFi network");

    client.setServer(mqttServer, mqttPort);
    client.setCallback(callback);

    while (!client.connected()) {
        Serial.println("Connecting to MQTT...");

        if (client.connect(CHIP_NAME, USERNAME, PASSWORD)) {
            Serial.println("connected");
        } else {
            Serial.print("failed with state ");
            Serial.print(client.state());
            delay(2000);
        }
    }
    timer.attach(5, pingMqtt);|
}

```

Figure 4.18: Setup function

In this function, we connect the ESP32 to the WiFi. The ESP32 will not start to operate unless it establishes the connection to the broker. Once that happens, it pings it every 5s to ensure its availability. Figure 4.19 shows the ping and reconnect function.



```

void pingMqtt() {
    client.publish("ping", CHIP_NAME);
}
|
void reconnect() {
    if (!client.connected()) {
        client.connect(CHIP_NAME, USERNAME, PASSWORD);
    }
}

```

Figure 4.19: Ping and reconnect functions

In our system, we have two functional types of ESP32s: Activating the rescue system and the Proximity detector with the Alert system activation.

#### 4.3.4.1.1 Rescue ESP32s

Figure 4.20 shows the setup of the rescue ESP32s.

<pre>#define CYLINDER 2 </pre>	<pre>pinMode(cylinder, OUTPUT); client.subscribe("test/cylinder");</pre>
(a) Cylinder pin	(b) Cylinder mode and subscription

Figure 4.20: Rescue ESP32 setup

In this code, we defined the output pin of the relay that's connected to the cylinder's valve. And subscribe to the cylinders messages topic in Mosquitto (test/cylinder) to receive all messages. Figure 4.21 shows the callback function.

```

void callback(char* topic, byte* payload, unsigned int length) {

    Serial.print("Message arrived in topic: ");
    Serial.println(topic);

    String message="";
    Serial.print("Message:");
    for (int i = 0; i < length; i++) {

        message+=(char)payload[i];
    }
    Serial.println("Message: "+message);
    Serial.println("-----");
    if(message=="true")
        digitalWrite(CYLINDER, 1);
    if(message=="false")
        digitalWrite(CYLINDER, 0);
}

```

Figure 4.21: Callback function

In this function, if the message from the broker was true on topic test/cylinder, the cylinder will be commanded to rise. If false, it will reset to default.

#### 4.3.4.1.2 Alert and proximity detector ESP32

Figure 4.22 shows the setup of the alert and proximity detector ESP32. In this code, we defined the ultrasonic and buzzer pins. And subscribed to the alert messages topic in Mosquitto (test/alert) to receive all messages in that topic.

<pre>const int trigPin = 2; const int echoPin = 4; const int buzzer = 19;</pre>	<pre>client.subscribe("test/alert"); pinMode(trigPin, OUTPUT); pinMode(echoPin, INPUT); pinMode(buzzer, OUTPUT);</pre>
(a) Ultrasonic and Buzzer pin	(b) Ultrasonic and buzzer modes and subscriptions

Figure 4.22: Alert and proximity detector ESP32 setup

Figure 4.23 shows the callback function of the (test/alert) topic

```
void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Message arrived in topic: ");
  Serial.println(topic);
  String message="";
  Serial.print("Message:");
  for (int i = 0; i < length; i++) {
    message+=(char)payload[i];
  }
  if(message=="true"){
    digitalWrite(buzzer,1);
    delay(500);
    digitalWrite(buzzer,0);
  }
  if(message=="false")
    digitalWrite(buzzer,0);
}
```

Figure 4.23: Alert and proximity detector Callback function

In this function, if the message from the broker was true on topic test/alert, the buzzer will be commanded to ring.

#### 4.3.4.2 Manager client

Since any access to Mosquitto topics is done by a client, and after limiting the access of each client for it to be restricted to read or write from and to whatever topic it should to, there must be a special client that has full access to topics. This client's role is to manage what and when everything is published. As shown in Figure 4.12, this client (MANAGER) was given the full access. We've implemented this client using python.

Figure 4.24 shows the client connection code to Mosquitto.

```

import paho.mqtt.client as paho

client= paho.Client(paho.CallbackAPIVersion.VERSION2, "MANAGER")
client.username_pw_set("MANAGER","strong")
broker="BROKER-IP"
client.on_message=on_message
print("connecting to broker ",broker)
client.connect(broker)#connect
client.subscribe("test/ultra")#subscribe
client.subscribe("ping")
client.subscribe("test/cylinder/control")
client.loop_forever()

```

Figure 4.24: Manager client setup and connection

In this code, MANAGER client connects to Mosquitto with its credentials, and subscribes to the topics that are published to by other clients in order to receive the data and manages the behavior of the system upon them.

Figure 4.25 shows the callback function to receive the data from all subscribed topics.

```

#define callback
def on_message(client, userdata, message):
    if(message.topic=="ping"):
        print("pinged" ,message.payload)
        update(con,message.payload.decode("utf-8"),"ONLINE")
    if(message.topic=="test/ultra"):
        if message.payload.decode("utf-8")== "true":
            age=ad.get_Age(camera=0) # Stream source
            if(age<=14):
                client.publish("test/alert","true")
    if(message.topic=="test/cylinder/control"):
        if message.payload.decode("utf-8")== "true":
            client.publish("test/cylinder","true")
        if message.payload.decode("utf-8")== "false":
            client.publish("test/cylinder","false")

```

Figure 4.25: Callback function for MANAGER client

In this code, messages on topics

- ping: contain the client name. Manager, once receives a message, it updates the status of the client to ONLINE with the update time.
- test/ultra: if true, it means that the proximity detector ESP32 detected motion around the pool. Then, it activates the age identification model from the camera stream. If the age of the person detected is less than or equals 14, we activate the alert system.

- test/cylinder/control: are only received from the app user, if true, it means that the user wants to rise up the cylinders. Otherwise, resets them.

### 4.3.5 App build and setup

This app is easy-to-build IoT applications dashboard. It contains several built-in panels that can be configured to publish/subscribe to MQTT brokers. Since our Mosquitto broker only accepts known clients, this app is configured to connect to Mosquitto with app-user1 as shown in Figure 4.12. Figure 4.26 shows the connection configuration in this app.

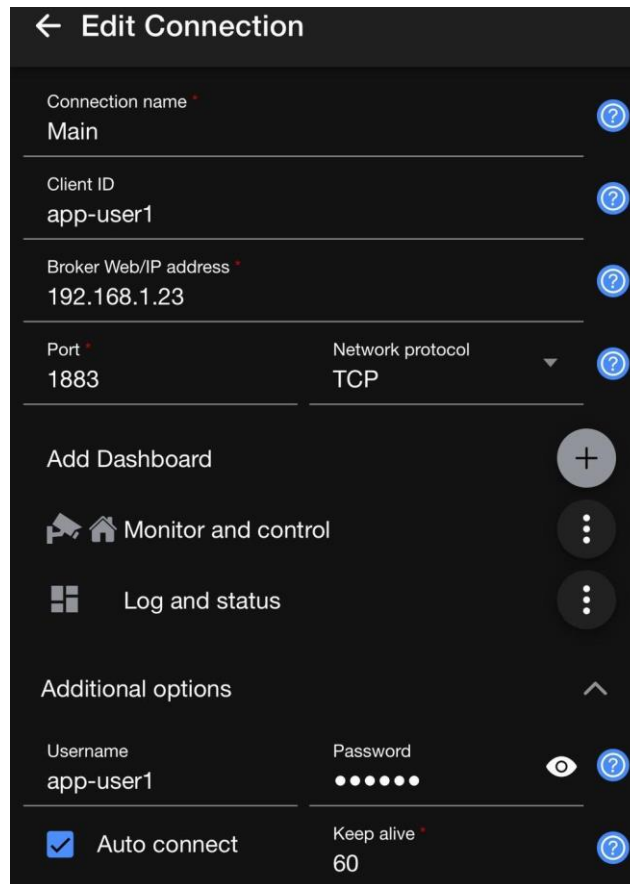


Figure 4.26: App connection setup

#### 4.3.5.1 Streams

IoT MQTT panel involves two cameras monitoring a private pool: CAM1 at the entrance and CAM2 above the pool. Both cameras capture live video, encode each frame in base64, and publish these frames to the MQTT topic test/stream. The MQTT client application subscribes to this topic, receiving the base64-encoded frames. The client then decodes these frames back into images and displays the live streams in real-time. This setup allows continuous monitoring of the pool entrance and the pool itself, enabling prompt detection of safety issues or unauthorized access.

### 4.3.5.2 Buttons

Two buttons, "Activate Rescue" and "Reset Rescue," are included in the UI to control the ground net. The "Activate Rescue" button publishes true to test/cylinder/control to raise the net in emergencies, while the "Reset Rescue" button publishes false to reset the net. This setup ensures efficient, real-time monitoring and control of the pool safety mechanisms. Figure 4.27 shows the app home page.

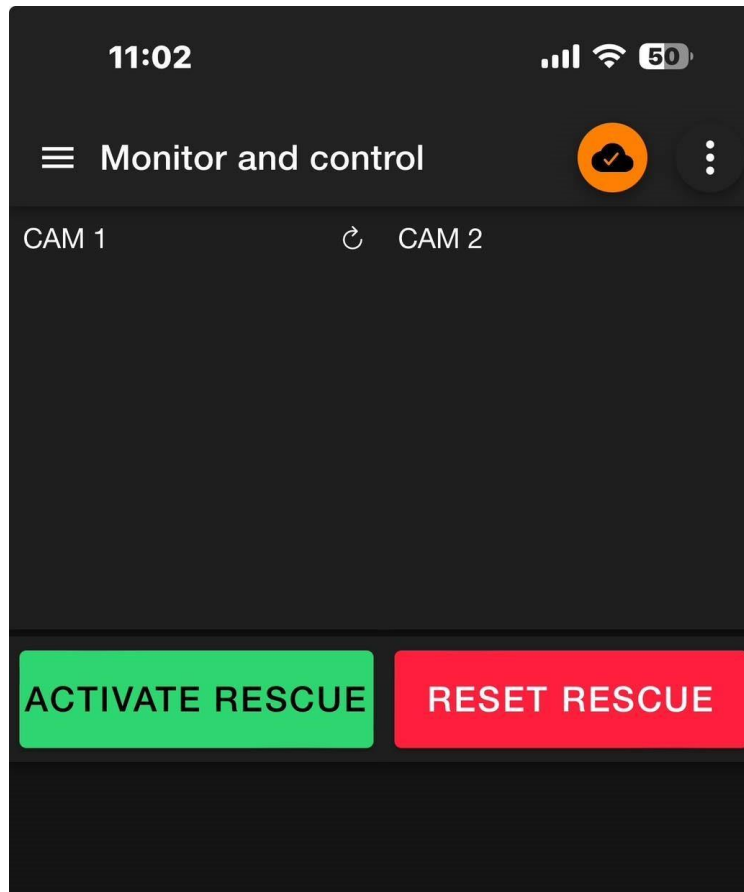


Figure 4.27: System App

## 4.4 challenges

- In our project, we face the challenge of not having enough time to build the application from scratch. To address this, we found that using IoT MQTT Panel is suitable for our purpose and helps us control our project effectively.
- We initially decided to use a Kinect sensor as a proximity detector, but then we discovered that our Kinect v2 is not compatible with the Raspberry Pi. So, we went back to the age identification using AI.
- Our phone's 3G connection cannot handle 8 simultaneous connections and causes delays between the servers and ESP32s. To solve this, we used a 3G router to effectively maximize the number of connections.

- We faced interference between the ESP32s and valves, which caused errors in the system. To solve this, we placed a plastic layer between the ESP32s and the metal table.
- We initially tried using a single button for both emergency and system reset, but this caused delays in execution during emergencies. To address this, we decided to use two separate buttons: one for resetting the system and one for emergency situations.
- To maximize resource usage, we merged the alert and proximity detector functions onto the same ESP32.
- We can't put water in the prototype pool for testing because it leaks into the ESP32s and valves. To solve this, we tested the system using real videos taken in actual pools.

## 4.5 Summary

In this chapter we reviewed the hardware and software implementation each component was fully explained and then we discussed the issues and challenges we faced during the implementation phase.

# Chapter 5

## Testing and Results

### 5.1 Preface

The testing of all system components and the results will be covered in this chapter.

### 5.2 Hardware testing

This section discusses the testing process of each of our hardware components.

#### 5.2.1 Unit testing

##### 5.2.1.1 Esp32

We connected ESP32 controller with power banks. Red LED lights up showing it's on. While the ESP32 is connecting to the broker, the built-in LED blinks, and it turns off when the connection is successful.

##### 5.2.1.2 Raspberry Pi 4

We performed thorough testing on the Raspberry Pi 4 to confirm its proper functionality. This testing involved verifying the correct operation of all hardware components, including the microSD card, power supply, and all USB ports, as well as ensuring network connectivity. The Raspberry Pi 4 passed all tests successfully, and we are confident in its ability to effectively support our project's requirements.

##### 5.2.1.3 Cameras

We connected both cameras to the Raspberry Pi USB ports. Both worked as expected and captured the streams.

##### 5.2.1.4 Pneumatic valves

Valves testing involved supplying it with compressed air from the compressor. The valve passed the air from the normally open channel. After supplying 24V DC, a red LED turned on and the air ways switched to the other channel.

### **5.2.1.5 Pneumatic cylinder**

Each cylinder was connected to a valve, each valve output is connected to an air port of the compressor. We tested the cylinder by switching its valve's state. There was no air leak and we got the expected results from the cylinder as it was rising and resetting without delays.

### **5.2.1.6 Ultrasonic sensor**

We performed thorough testing on the ultrasonic sensor to confirm its proper functionality within our system. This testing involved verifying the accuracy and reliability of distance measurements in various environmental conditions. We checked the sensor's response time and consistency by measuring known distances and comparing the readings to the expected values. Additionally, we tested the sensor's integration with our control system to ensure it correctly triggers the appropriate responses. The ultrasonic sensor passed all tests successfully, and we are confident in its ability to provide precise and dependable distance measurements for our application.

### **5.2.1.7 Alert**

We performed thorough testing on the alert system, which includes a buzzer for sound alerts and an LED for visual alerts, to confirm their proper functionality within our setup. This testing involved verifying the correct operation of both the buzzer and LED by triggering them through various control signals. We assessed the sound intensity and clarity of the buzzer, ensuring it is audible from the required distance. Similarly, we checked the LED's brightness and visibility under different lighting conditions. Additionally, we tested the integration with our control system to ensure timely and accurate activation of alerts. The alert system passed all tests successfully, and we are confident in its ability to effectively signal alerts as required by our application.

### **5.2.1.8 Electrical switch**

We performed thorough testing on the push-up electrical switch to confirm its proper functionality within our system. This testing involved verifying the integrity of all connections and ensuring that the switch operates smoothly and reliably. We conducted multiple activation and deactivation cycles to check for consistent performance and responsiveness. Additionally, we tested the switch's integration with our control system to ensure it correctly triggers the intended actions without any delays or faults. The push-up electrical switch passed all tests successfully, and we are confident in its ability to reliably control the system's functions as required.

### **5.2.1.9 Metal net and weight support**

A weight of 30kg was distributed on the ground, without water inside. The net was successfully raised with no problems. 30kg is almost the average weight of a 10 years old child[37]

## **5.2.2 Software components**

This section discusses the testing process of each of our software testing.



### 5.2.2.1 Raspberry Pi OS installation

We installed and tested various versions of “Raspbian OS” until we found the one that fits our requirements. Raspberry Pi OS (64-bit) We used it in the project since it provided us what we need.

### 5.2.2.2 Age detection

We tested our age detection model with 1063 pictures. We made sure to use lots of different pictures with different ages, lighting, and poses to see how well the model works in real-life situations. Figure 5.1 shows some examples of how our model works with pictures of kids.

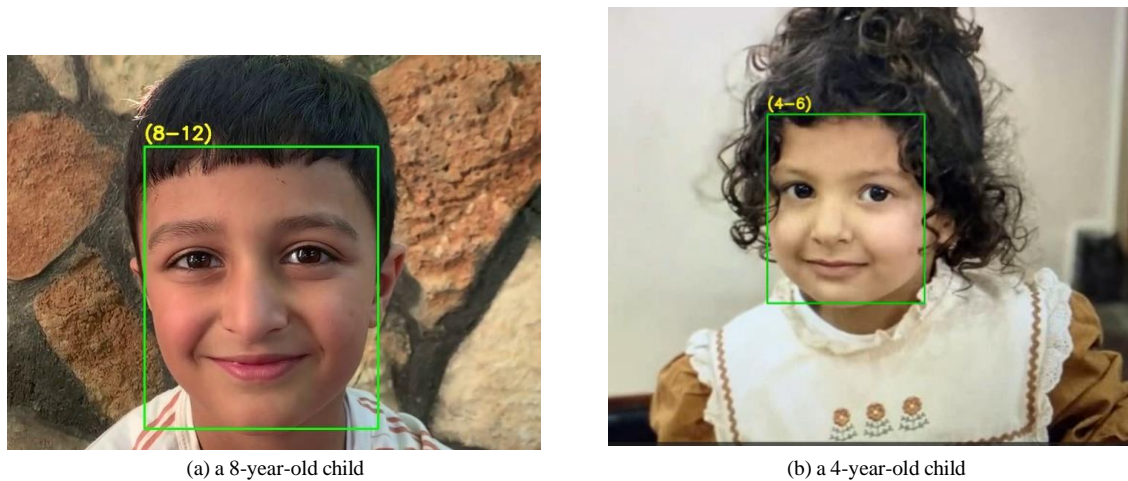


Figure 5.1: Age Model test

We evaluated our age detection model using a dataset that produced the following results: 563 true positives, 300 true negatives, 100 false positives, and 100 false negatives. These numbers were used to construct a confusion matrix, providing a detailed breakdown of the model’s performance:

The accuracy of the model can be calculated using the formula:

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total Number of Cases}}$$

Substituting the given values:

		actual class	
		P	N
predicted class	P	563	100
	N	100	300

Thus, the model’s accuracy is approximately 81.2%. This indicates that the model correctly identified age groups in the dataset with a high degree of reliability. The confusion matrix shows that the model successfully identified 563 instances as the correct age group and correctly recognized 300 instances as not belonging to a specific age group. However, it also made 100 false

positive and 100 false negative errors, suggesting areas for improvement in reducing misclassifications.

### 5.2.2.3 Drowning detection

We tested our model using 7 videos, consisting of a total of 882 frames. For each frame, the model predicted whether it belonged to the positive or negative class. The resulting predictions were used to construct the confusion matrix, which summarized the performance as follows: 390 true positives, 446 true negatives, 30 false positives, and 16 false negatives. From these values, we calculated the model's accuracy using the formula:

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total Number of Cases}}$$

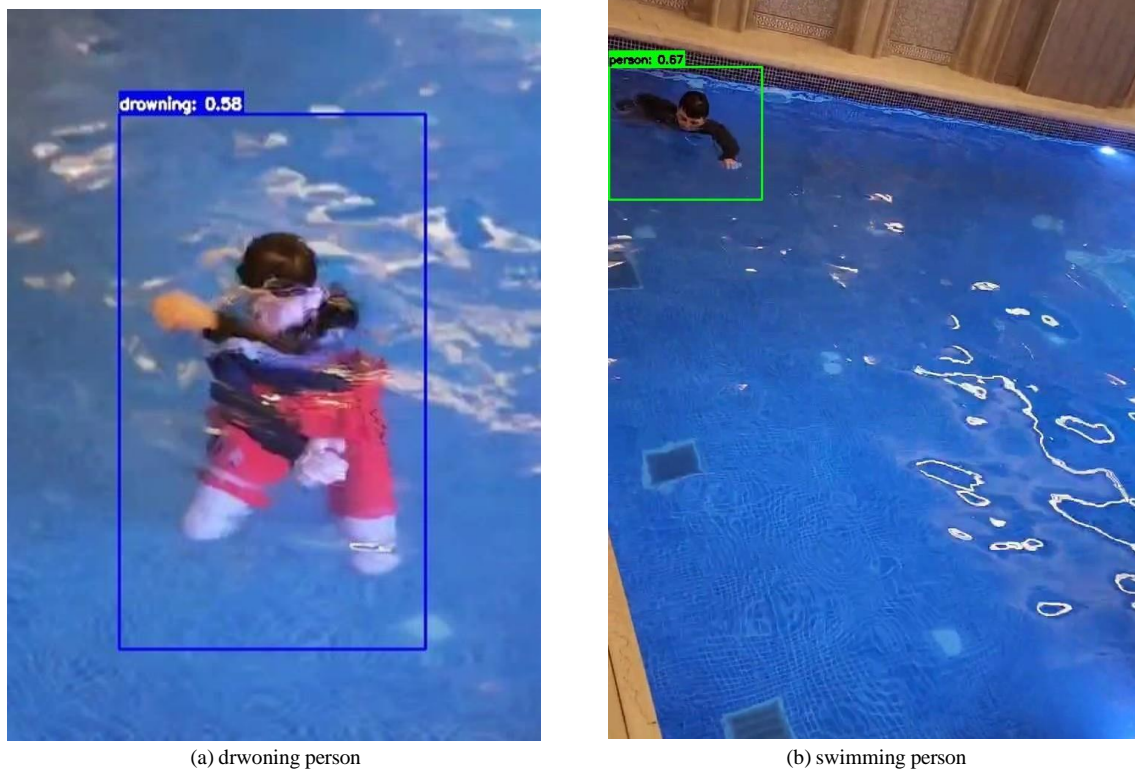


Figure 5.2: Drowning Model test

This yielded an accuracy of approximately 94%, indicating that the model correctly identified the class of the frames with high reliability.

		actual class	
		P	N
predicted class	P	390	30
	N	16	446

#### **5.2.2.4 Broker**

The communication protocol we used is MQTT, implemented via the Mosquitto broker. We selected Mosquitto because it offers a scalable and reliable solution for IoT devices. We installed the Mosquitto broker and tested it for reliability and performance. The testing demonstrated stable message transmission with low latency, averaging around 5-10 milliseconds per message. Additionally, we verified the broker's ability to handle multiple client connections simultaneously and maintain data integrity. The Mosquitto broker passed all tests successfully, and we are confident in its ability to provide robust and efficient communication for our system.

#### **5.2.2.5 Response time**

As mentioned before, our system analyses the streams each 10 seconds to decide whether there is a drowning situation(drowning frames are more than 50% of the total frames detected) or not. If yes, the net will rise. The total time taken from the moment of initiating the rescue process, to the full rise of the net is 500ms. Factors like: air compressor power, height of the pool, internet traffic and microcontroller performance are essential.

### **5.3 Summary**

In this chapter we discussed the hardware and software testing.

## Chapter 6

# Conclusion and future work

### 6.1 Preface

The chapter introduces a summary of the project, the future directions, and future work.

### 6.2 Conclusion

The Smart Rescue System for Private Swimming Pools project has successfully developed and demonstrated an innovative, AI-based solution to significantly enhance the safety of children in swimming pool environments. This system integrates advanced technologies, including computer vision, IoT, and AI, to provide real-time monitoring and emergency response capabilities. The project effectively addresses the critical issue of drowning by utilizing cameras and sensors to detect when a child approaches or enters the pool, and subsequently alerting parents. Additionally, the system features a liftable plate mechanism that activates to prevent drowning incidents, thereby providing a critical intervention that can save lives and reduce the risk of permanent injury. Through comprehensive testing and validation, the system has shown high reliability in detecting potential dangers and responding appropriately and swiftly. This project represents a significant advancement in pool safety technology, offering a practical and efficient solution that brings peace of mind to parents and enhances the overall safety of private swimming pools. The successful implementation and positive results of this project underscore its potential impact and pave the way for future enhancements and broader adoption.

### 6.3 Future work

- **Enhanced AI Models:** Improve the accuracy and speed of the AI model by incorporating advanced machine learning algorithms and larger training datasets. This will help in better distinguishing between different types of activities and ensuring quicker response times.
- **Scalability and Deployment:** Focus on the scalability of the system to support larger commercial pool environments, including public pools and hotel facilities. This will involve optimizing the system for higher numbers of simultaneous users and larger monitoring areas.
- **User Interface and Experience:** Enhance the user interface for better usability and interaction, providing parents with real-time data, alerts, and control options through a mobile application.

# References

- [1] Centers for Disease Control and Prevention, "Global drowning research & prevention," <https://www.cdc.gov/drowning/global/index.html> (accessed Jan. 13, 2024).
- [2] Hackensack Meridian Health, "8 truths about drowning and 'dry drowning' revealed," <https://www.hackensackmeridianhealth.org/en/healthu/2019/07/09/8-truths-about-drowning-and-dry-drowning-revealed> (accessed Jan. 13, 2024).
- [3] SAS, "Computer vision: What it is and why it matters," [https://www.sas.com/en\\_us/insights/analytics/computer-vision.html](https://www.sas.com/en_us/insights/analytics/computer-vision.html) (accessed Jan. 13, 2024).
- [4] Coursera, "What is the internet of things (IoT)? with examples," <https://www.coursera.org/articles/internet-of-things> (accessed Jan. 13, 2024).
- [5] IBM, "What is edge computing?" <https://www.ibm.com/topics/edge-computing> (accessed Jan. 13, 2024).
- [6] C. Gregersen, "A complete guide to IoT protocols standards in 2023," Nabto, <https://www.nabto.com/guide-iot-protocols-standards> (accessed Jan. 13, 2024).
- [7] IBM, "What are convolutional neural networks?" <https://www.ibm.com/topics/convolutional-neural-networks> (accessed Jan. 13, 2024).
- [8] A.C. Egypt, "What is an air compressor and what is it used for?" Air Compressor Technique Blog, <https://www.aircompressorblog.com/what-is-an-air-compressor> (accessed Jan. 13, 2024).
- [9] A. Amayra and K. Al Rjoub, "Automated rescue system for children in private home swimming pools," Thesis, 2012.
- [10] CBS News, "How young is too young for children to learn to swim?" <https://www.cbsnews.com/news/how-young-is-too-young-for-children-to-learn-to-swim> (accessed Jan. 13, 2024).
- [11] M. Saleem and Y. Abusaimh, "The design of a smart rescue system for private swimming pools," Thesis, 2018.
- [12] Khan Academy, "What is buoyant force? (Article) — Fluids," <https://www.khanacademy.org/science/physics/fluids/buoyant-force-and-archimedes-principle/a/buoyant-force-and-archimedes-principle-article> (accessed Jan. 13, 2024).
- [13] "Density of metals," Wermac, [https://www.wermac.org/materials/density\\_of\\_metals.html](https://www.wermac.org/materials/density_of_metals.html) (accessed Jan. 16, 2024).
- [14] Yates Industries, "Hydraulic cylinder," <https://www.yatesind.com/what-is-a-hydraulic-cylinder> (accessed Jan. 12, 2024).
- [15] IQS Directory, "Pneumatic cylinders," <https://www.iqsdirectory.com/articles/air-cylinder/pneumatic-cylinders.html> (accessed Jan. 12, 2024).

- [16] SMAC Corporation, "Electric cylinders," <https://www.smac-mca.com/products/electric-cylinders> (accessed Jan. 12, 2024).
- [17] Measure Monitor Control, "Five way (5/2 or 5/3) solenoid valves," <https://www.measuremonitorcontrol.com/valves/solenoid/five-way> (accessed Jan. 13, 2024).
- [18] R. Teja, "Getting started with ESP32: Introduction to ESP32," ElectronicsHub, <https://www.electronicshub.org/getting-started-with-esp32> (accessed Jan. 13, 2024).
- [19] Raspberry Pi, "Raspberry Pi 4 Model B specifications," <https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications> (accessed Jan. 13, 2024).
- [20] NVIDIA Developer, "Jetson Nano," <https://developer.nvidia.com/embedded/jetson-nano> (accessed Jan. 13, 2024).
- [21] Disabled World, "Height and weights of teens," <https://www.disabled-world.com/calculators-charts/height-weight-teens.php> (accessed Jan. 13, 2024).
- [22] A. F. Bassem Marji, "Age detection using OpenCV in Python," Python Code, <https://thepythoncode.com/article/predict-age-using-opencv> (accessed Jan. 13, 2024).
- [23] SouqGreen, "4K Ultra HD 1080p sport action camera WiFi 16MP waterproof DV video recorder," [https://zikzakstore.com/en/product\\_right\\_sidebar.php?cd=803&e=](https://zikzakstore.com/en/product_right_sidebar.php?cd=803&e=) (accessed Jan. 14, 2024).
- [24] L. Jackson and Name, "ESP32-CAM: The complete machine vision guide," Arducam, <https://www.arducam.com/esp32-machine-vision-learning-guide> (accessed Jan. 13, 2024).
- [25] Cincon, "What is power supply? Basic introduction of power supply," [https://www.cincon.com/newsdetail\\_en.php?id=7659](https://www.cincon.com/newsdetail_en.php?id=7659) (accessed Jan. 13, 2024).
- [26] L. Prasad, "What is relay? How it works? Types, applications, testing," ElectronicsHub, <https://www.electronicshub.org/what-is-relay-and-how-it-works> (accessed Jan. 13, 2024).
- [27] CUI Devices, "Push button switches 101," <https://www.cuidevices.com/blog/push-button-switches-101> (accessed Jan. 16, 2024).
- [28] Elprocus, "What is a buzzer: Working & its applications," <https://www.elprocus.com/buzzer-working-applications> (accessed Jan. 16, 2024).
- [29] Nico31415, "Drowning detector: Using YOLO object detection," GitHub, <https://github.com/nico31415/drowning-detector> (accessed Jan. 13, 2024).
- [30] E. Zvornicanin, "What is YOLO algorithm?" Baeldung on Computer Science, <https://www.baeldung.com/cs/yolo-algorithm> (accessed Jan. 13, 2024).
- [31] Y. Tetar, "Drowning detection and prevention in swimming pools dataset," accessed Apr. 18, 2023. <https://universe.roboflow.com/yashwanee-tetar-jm88u/drowning-detection-and-prevention-in-swimming-pools>.
- [32] R. Mitchell, "HTTP vs MQTT - Which communication protocol should you use in your IoT application?" Electromaker, <https://www.electromaker.io/blog/article/http-vs-mqtt> (accessed Jan. 13, 2024).

- [33] GeeksforGeeks, "Difference between MQTT and HTTP protocols," <https://www.geeksforgeeks.org/difference-between-mqtt-and-http-protocols> (accessed Jan. 13, 2024).
- [34] Radiocrafts, "CoAP - Constrained Application Protocol," <https://radiocrafts.com/technologies/coap-constrained-application-protocol> (accessed Jan. 13, 2024).
- [35] Catchpoint, "The guide to MQTT broker," <https://www.catchpoint.com/network-admin-guide/mqtt-broker> (accessed Jan. 13, 2024).
- [36] Saashub, "Mosquitto vs HiveMQ - Compare differences & reviews," <https://www.saashub.com/compare-mosquitto-vs-hivemq> (accessed Jan. 13, 2024).