University of Minho
School of Engineering

Integrated Master in Industrial Electronics and Computers Engineering

ARGOS

Threat Detection Watchdog Device

Authors:
Hugo Carvalho A85156
José Mendes A85951

Professor:
Dr. Adriano Tavares

October 19, 2020

Contents

List of Figures

1 Problem Statement

Armed robberies and assaults are a menace to the lives and assets of the victims and require swift action to be resolved.

In those circumstances, the identification and reporting of danger are left, most of the time, at the hands of the endangered. This causes a delay between the threat recognition moment and the moment when authorities take action. This means that any technology that can eliminate that delay might help to save people and property.

In response to that matter, this product aims to help identify and report threats resorting to a weapon recognition system powered by computer vision. Upon detecting a dangerous situation, the device will send useful information like surveillance footage frames, the number of weapons identified, and their position in the environment to a human operator.

2 Problem Statement Analysis

From the Problem Statement Analysis, we can extrapolate an overview of main components, features they provide and relationships between them. The schematic in figure 2.1 describes a system comprised by:

- A **Camera Module**, which will feed a stream of periodic frames from the environment into the system to be analyzed;
- A **Computational Unit** to which will be assigned the task of analyzing the aforementioned frames in search of events worth signaling and communicating those events to the Remote Client, which will be explained in further detail in the next item. It will also be responsible for storing noteworthy video frames and relevant data in a remote storage that can be accessed by itself and the Remote Client.
- A **Remote Client Back-end**, which will communicate directly with the Computational Unit to receive event alerts and a remote storage service to retrieve information about signaled events and image frames sent there by the Computational Unit.
- A **Remote Client Interface**, a Graphical User Interface that will bridge the gap between the user and the Remote Client, simplifying and streamlining the access to information about important events and video frames, thus allowing for a faster response to be delivered by human operators.

- A **Cloud Storage Service** for hosting the storage of the aforementioned video frames to be placed by the Computational Unit and retrieved by the Remote Client.
- A **Database** for the Computational Unit to store information about signaled events and the Remote Client to retrieve them.

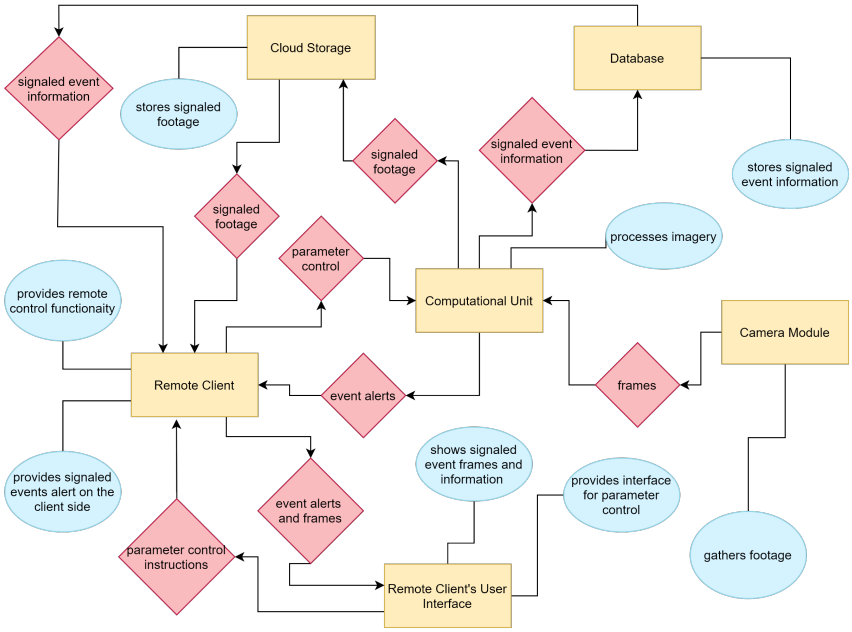

Figure 2.1: Problem Statement Diagram

3 System Overview

The diagram in figure 3.1 represents an initial big picture of the project to facilitate objective identification. Concerning the diagram, one can identify four main blocks:

- The **Raspberry Pi Board** block
- The **Camera Module** block
- The **Operational Infrastructure** block
- The **Reinforcement Learning Model** block

The Raspberry Pi block, powered by computer vision provided from the Camera Module will use a Reinforcement Learning (RL) Model to detect weapons in frames. This information will be stored in the cloud and in a database whilst notifying the remote client of possible threats.

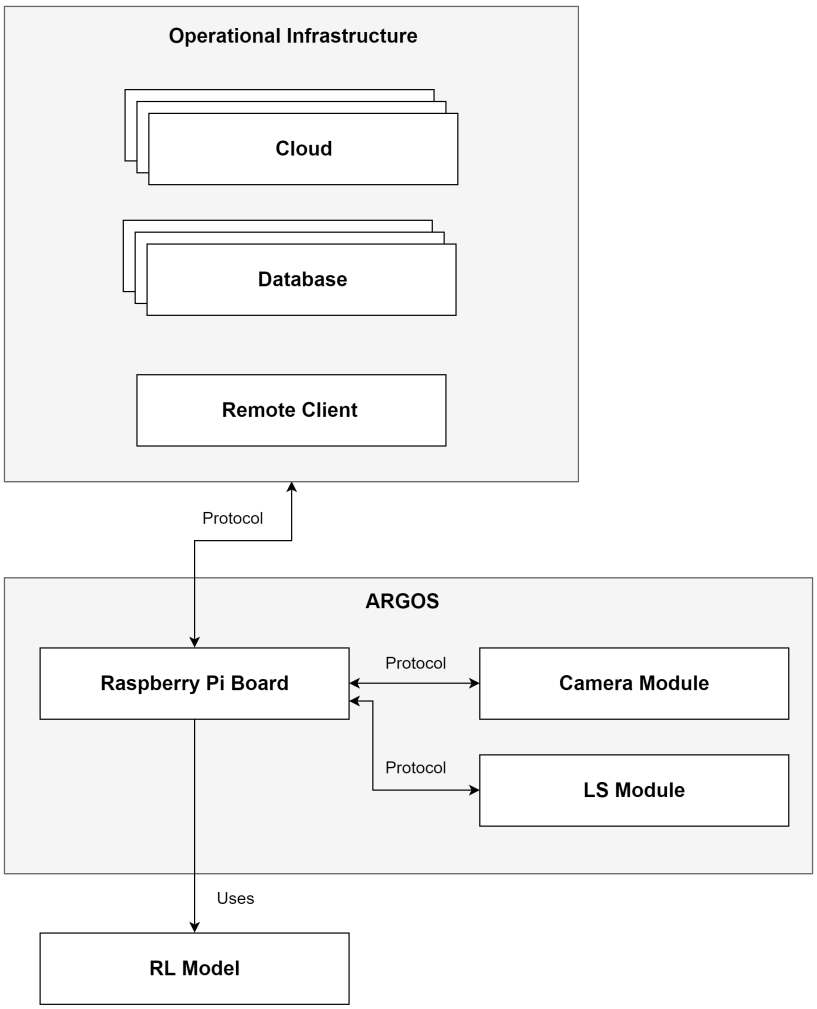

Figure 3.1: System Overview Diagram

4 Requirements and Constraints

The development requirements are divided into functional and non-functional if they are main functionalities or secondary ones, respectively. Additionally, the constraints of the project are classified as technical or non-technical.

4.1 Requirements

Functional Requirements

- Take video frames;
- Detect weapons in the frames;
- Store the information in a database;
- Notify external client;
- Allow remote camera control;

Non-Functional Requirements

- Have low power consumption;
- Have low latency in communication with the remote client;
- Have high throughput in communication with the cloud and database;
- Be upgradable;
- Be discrete and sturdy;

4.2 Constraints

Technical Constraints

- Use Raspberry Pi Model 4B as development board;
- Use Buildroot for cross-platform development and system configuration;
- Use POSIX Threads for Real-Time buildout;
- Use Embedded Linux;
- Implement (at least) one device driver;
- Use Object-Oriented Programming Languages;

Non-Technical Constraints

- Project deadline at the end of the semester;
- Pair workflow;
- Limited budget;

5 System Architecture

5.1 Hardware Architecture

The block diagram in figure 3.1 shows a schematic of the planned hardware architecture, divided into three main blocks:

- The **Raspberry Pi** block, comprised by the Raspberry Pi board and components that will allow it to easily communicate various operational states to the user;
- The **Camera Module** block, with a camera module and a light sensor that will communicate with the Raspberry Pi through serial interfaces to provide video frames and ambient light information in order to adjust camera parameters;
- The **Operational Infrastructure** block, which is made up of all the external components needed to remotely control and manage information extracted from the environment.

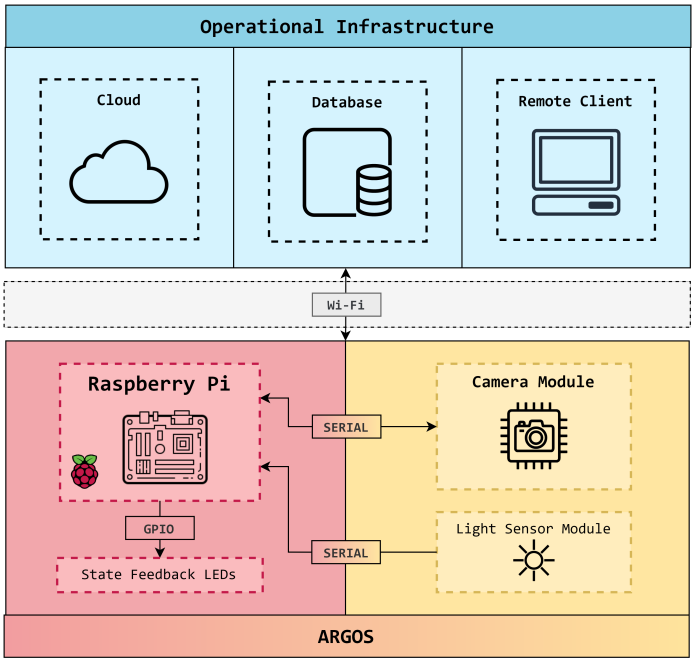

Figure 5.1: Hardware Architecture Diagram

5.2 Software Architecture

The software architecture is divided into three layers, those being:

- The **Operating System** layer, which is comprised by the Operating System drivers and Board Support Packages;
- The **Middleware** layer, which includes software for abstracting the lower level packages and streamlining the development of complex algorithms;
- The **Application** layer, where the core functionality of the program is built, with resource to the API's in the lower level layers.

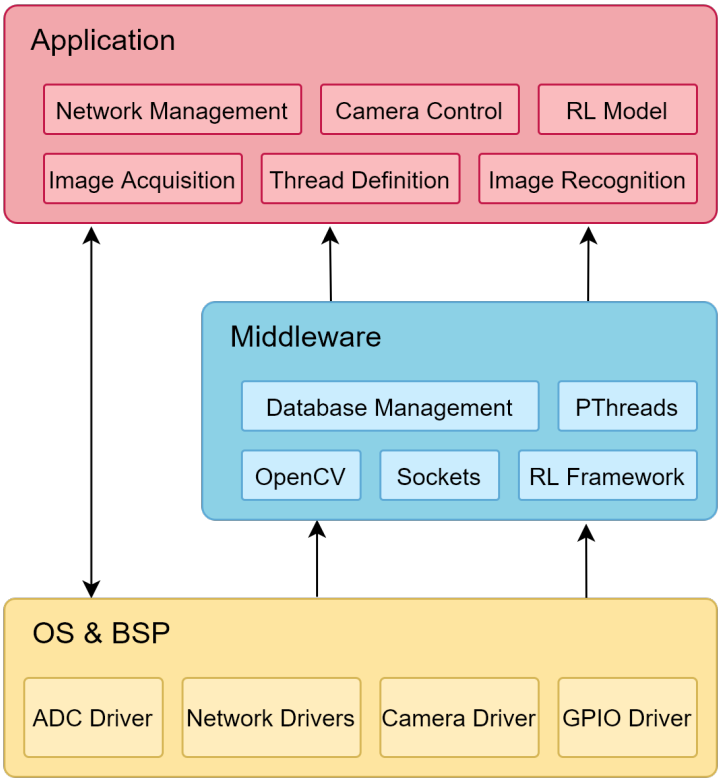

Figure 5.2: Software Architecture Diagram

6 Gantt Diagram

	Task Name	Duration	Start	ETA
1	**M0: Planning**			
1.1	Project Choice	2 days	15.10.20	16.10.20
1.2	Problem Statement	3 days	17.10.20	19.10.20
1.3	Project Plan	7 days	20.10.20	26.10.20
2	**M1: Analysis**			
2.1	System Overview	2 days	20.10.20	21.10.20
2.2	Requirements and constraints	1 day	22.10.20	22.10.20
2.3	System Architecture	2 days	23.10.20	24.10.20
2.4	System Analysis	2 days	25.10.20	26.10.20
2.5	Market Research	2 days	27.10.20	28.04.12
2.6	Estimated Budget	2 days	29.10.20	30.10.20
2.7	Diagrams	10 days	31.10.20	9.11.20
3	**M2: Design**			
3.1	Analysis Review	3 days	10.11.20	12.11.20
3.2	Hardware Specification	5 days	13.11.20	17.11.20
3.3	Component Shipping	14 days	18.11.20	7.12.20
3.4	Software Specification	10 days	18.11.20	27.11.20
3.5	ML Model Design and Training	8 days	28.11.20	5.12.20
4	**M3: Implementation**			
4.1	Hardware Testing	7 days	8.12.20	11.12.20
4.2	Software Implementation	32 days	1.12.20	31.12.20
4.3	Hardware Implementation	32 days	1.12.20	31.12.20
4.4	ML Model Conversion	5 days	1.12.20	5.12.20
5	**M4: Testing & Verification**			
5.1	Software Unit Testing	4 days	1.1.21	4.1.21
5.2	Software Integrated Testing	5 days	4.1.21	8.1.21
5.3	Hardwareware Unit Testing	4 days	1.1.21	4.1.21
5.4	Hardware Integrated Testing	5 days	4.1.21	8.1.21
5.5	Verification	3 days	9.1.21	11.1.21

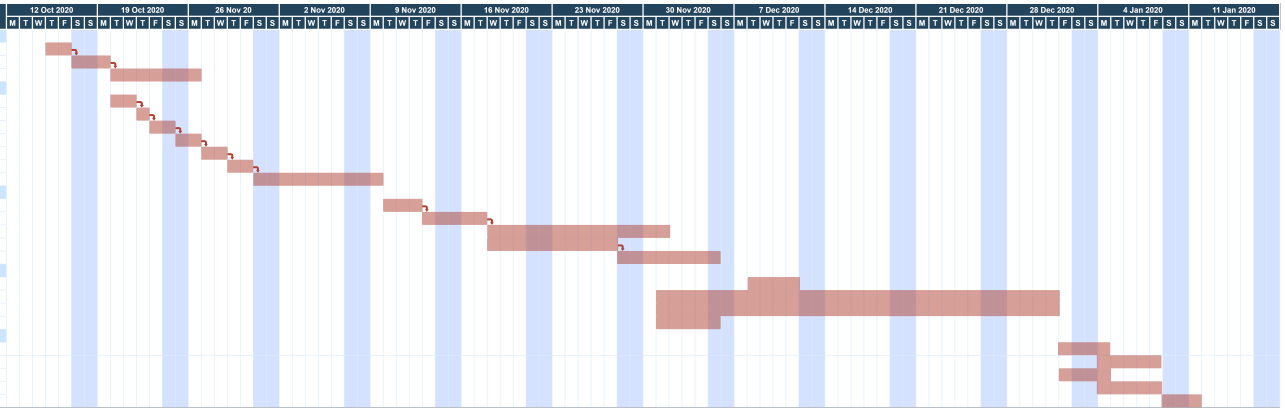

Figure 6.1: Gantt Diagram