Universidade do Minho
Escola de Engenharia

Mestrado Integrado em Engenharia Eletrónica Industrial e Computadores

---

Embedded Systems

Collision Avoidance and Transit Signals Detection Device

Professor: Adriano Tavares

Authors: João Faria A85632
Tiago Costa A86122

October 22, 2020

Contents

List of Figures

1 Problem Statement

With the ever-evolving environment in this day and age, some efforts were being made regarding the self-driving capability. Moreover, with the necessity of mass promoting autonomous driving, in this course, the system developed will facilitate that reality by being able to detect transit signals and by avoiding obstacle collision in real-time. That sensed information is to be sent to the driver by the infotainment system, a display attached to the Raspberry.

2 Problem Statement Analysis

After considering the problem the next step is to identify the features of the problem by detailing what entities are involved and which relations they share. The centre of data processing (in ESRG course) is Raspberry Pi that collects information about the camera, infrared sensors and transmits it to speakers, dc motor, and display. All of that is represented in the diagram below.

Figure 1: Problem Statement Analysis

3 System

3.1 System Overview

To read and control the forementioned features, the input/output list is below described:

Inputs:

- IV sensor, to sensor obstacules near the vehicle;
- Camera, to detect and recognize like the target and transit signals;
- Display, as the user input interface for adjusting the controlling parameters such as temperature and light (brihghtness of the display itself);

Outputs:

- Speakers, to transmit audio alerts back to the user;
- DC Motor of the robot vehicle;
- Display, works as an output too, displaying cabin information;
- Heating subsystem, to rise the temperature according to the user reference value;
- STM-32, to send control commands to the respetive actuators;

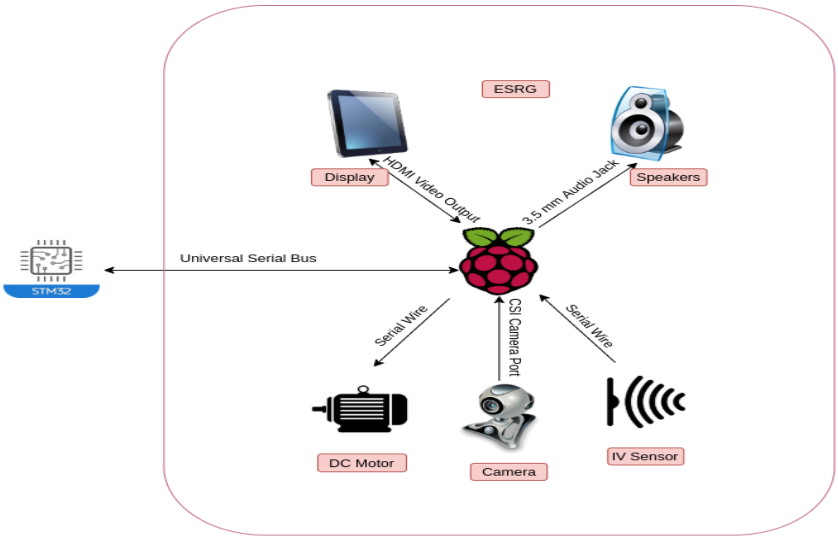

Figure 2: System Overview

3.2 Requirements and Constraints

Here we have 2 different types of requirements, functional requirements and non functional requirements. The functional requirements must specifies what the system is capable of doing and the non functional requirements specifies how the system will perform a certain action.

3.2.1 Project Requirements

Functional Requirements

- Avoid obstacule collision;
- Detect and recognize transit signals via camera vision;
- Detect and reach intended target;
- Reproduce audio alerts such as: ”Stop” or ”Continue”;

Non Functional Requirements

- Low cost;
- Responsive;
- Reliable;
- Interoperability;
- Power Efficient;
- Adaptability;

3.2.2 Project Constraints

The constraints define the product limitations. They can be technical and non-technical. Technical constraints are associate with the technical part of the project whilst non-technical ones are project management related.

Technical Constraints

- Use RaspberryPi V4 board;
- Use pthreads;
- Use object-oriented programming languages;
- Embedded Linux by buildroot;
- Cross compiling toolchain;
- Device Drivers;

Non-Technical Constraints

- Group composed by 2 members;
- Deadline at end of semester;
- Limited budget;
- Soft Real Time Embedded System;
- Time to prototype;
- Soft Real-Time System;

3.3 System Architecture

The system hardware and the software architecture will be planned in more detail below to accomplish all the product features described in the aforementioned topics. This planning is part of a methodology that recommends breaking the global problem into minor problems to help to understand how things work independently. Finished the preparation, it is easier to build the project components in a structured fashion. So, the next figures present the established solution to fit the referenced architectures.

3.3.1 Hardware Architecture

In this section we take a view of Hardware Architecture and how the inputs and outputs are grouped and connected to the main processor that in this case is the board STM32F767ZI. Figure below illustrates that:

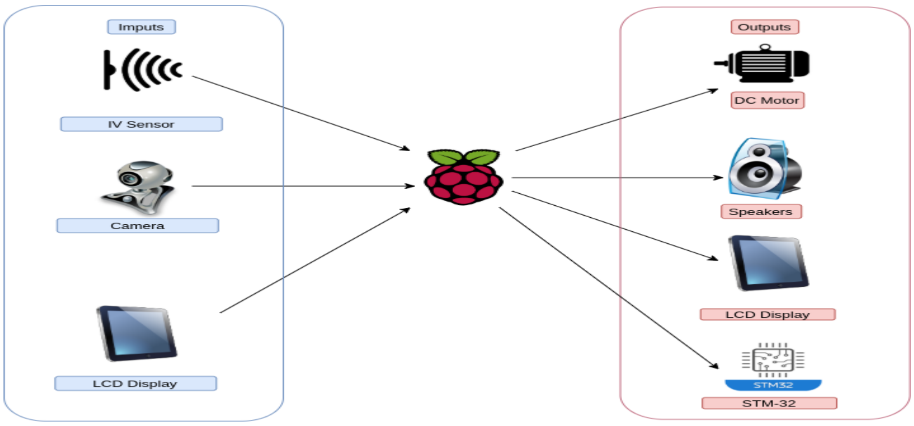

Figure 3: Hardware Architecture

3.3.2 Software Architecture

In the software architecture, specifically on the lower layer are represented the device drivers required for camera, USB, DC motor as well as for Touchscreen and Audio Driver. The middle layer components are data acquisition/processing and graphics library. Finally, the application layer is composed of the reinforcement learning, GUI and sound output part.

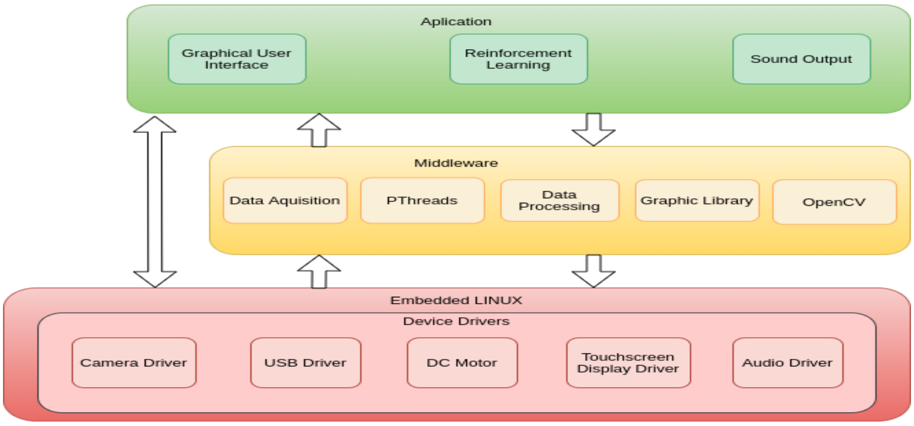

Figure 4: Software Architecture

4 Gantt Diagram

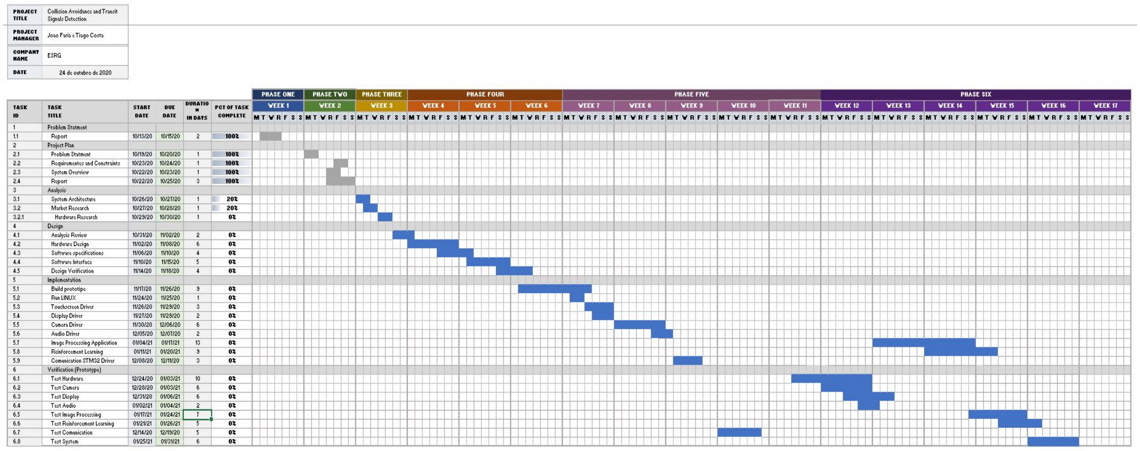

Figure 5: Gantt Diagram