Project Planning - Mouse Trap
Embedded Systems 2023/24

Authors:
Cristina Pêra (up201907321);
Diogo Ferreira (up201805258);
Telmo Ribeiro (up201805124);
Rogério Rocha (up201805123).

Project Description:

The project aims to develop a smart mouse trap system capable of effectively capturing mice while providing real-time notifications and control options to the user via a smartphone application.

We'll illustrate the system's overall functionality using an actor diagram. Next, we'll outline the connections between system components. Finally, we'll present a series of specifications, including architectural and software modeling diagrams.

Requirement analysis:

The mouse trap consists of a box equipped with sensors to detect when a mouse enters. Upon detection, the system sends an alert to the user's smartphone along with a picture of the captured mouse.

Functional requirements:

1. Detect when something is inside the box within 2 seconds.
2. Send a notification to the trap owner's smartphone within 3 seconds of detection.
3. Allow the user to remotely close or open the box via the smartphone application.
4. Provide real-time status updates of the trap (closed/opened) on the smartphone.
5. Enable the user to request a picture of the contents inside the box through the smartphone app.

Non-functional requirements:

1. The warning sent to the owner's smartphone should have a latency of less than 3 seconds.
2. The action to close or open the box via the smartphone app should take effect within 3 seconds.
3. The system should support multiple mouse traps controlled by the same user/smartphone.
4. The trap should aim for practicality through reduced weight/size.
5. The Android application should be intuitive to use.

Actor Model:

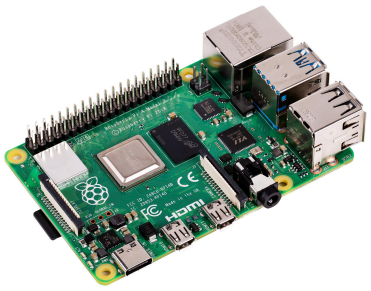
Fig. 1: Actor model

The system is described by the actor model represented in *Figure 1*.
There is a central actor called **CommCenter** (Communication Center) that acts as a broker between the **Mouse Trap** and **User** agents, being responsible for data processing and networking.

The **CommCente**r must be capable of receiving **events** from the **Mouse Traps** and **requests** from the **User**, propagating events, handling requests, and potential synchronizations between the two classes. It must also be able to generate requests regarding the **Mouse Traps** when certain conditions are met.
At no point in its functions should the **CommCenter** compromise the isolation of the **Mouse Traps'** environment.

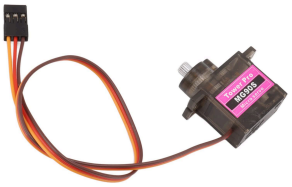
The **Mouse Traps** are systems that must function independently of each other, capable of detecting events and fulfilling requests.

The **User** is the system that provides feedback to the user and accepts controls from them, should also express events and generate requests.

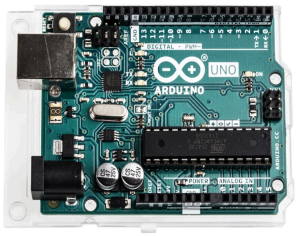


a) Raspberry-Pi 4

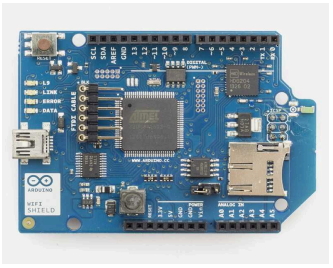
b) Webcam



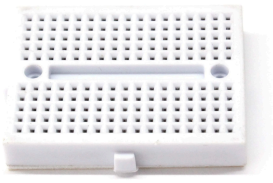
c) Micro Servo Motor



c) Arduino Uno Rev3



d) Arduino Wi-Fi Shield R3



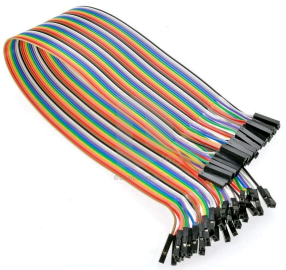
e) Mini BreadBoard

f) IR Sensors

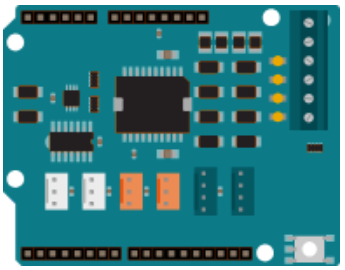
g) Universal Power Supply

h) LCDs

i) Smartphone Android



j) Conectores



k) Motor Shield Rev3

Fig. 2: Hardware Components required (Total estimated cost: 170 €)

Specification:

The specification establishes a connection between the **actor model**, outlined in the section with the same name, and the **hardware components**.

Thus, the **Raspberry Pi** will play the role of the **CommCenter Agent**, the **broker** between the **Mouse Trap Agent** and the **User Agent**, and the part responsible for the multimedia control of the **Mouse Trap Agent**. This choice is explained by the need to reduce the required components.

In addition to the **broker** function, the **CommCenter** will trigger some requests when certain conditions are met.

The **Mouse Trap Agent - Multimedia Controller** must be able to act on the camera and propagate the results to the **Communication Center**.

The two tasks of the **Raspberry Pi** must be executed in parallel and isolated since it is performing two different agents.

The **Arduino** will be responsible for playing the role of the **Mouse Trap Agent**, specifically the part named **Mouse Trap Controller**. The sum of this with components such as sensors, actuators, and shields constitutes the **Mouse Trap System**.

The **Mouse Trap Agent - Mouse Trap Controller** must fulfill requests, detect and propagate events, as well as delegate the responsibility of capturing multimedia to the **Mouse Trap Agent - Multimedia Controller**.

The **Mobile Phone** is responsible for constituting the **User Agent**.

The **User Agent** generates requests and expresses events through a GUI.

Event Map:
***Se** - change of sensor states;*
***Pe** - camera information collected;*
***Oe** - motor state = open;*
***Ce** - motor state = close;*

Request Map:
***Pr** - request for camera information collection;*
***Or** - request to change the motor state to open;*
***Cr** - request to change the motor state to close;*

Additional:
***Pc** - camera information.*

Blocks Diagram overview:

The block diagrams depicted in *Figures 3, 4 & 5* are utilized to illustrate the system in terms of the logical units comprising it. If we regard these logical units as hardware components, then the diagram depicts the hardware architecture. Alternatively, if the logical units correspond to software modules, then we say the diagram describes the software architecture.

Hardware Architecture:

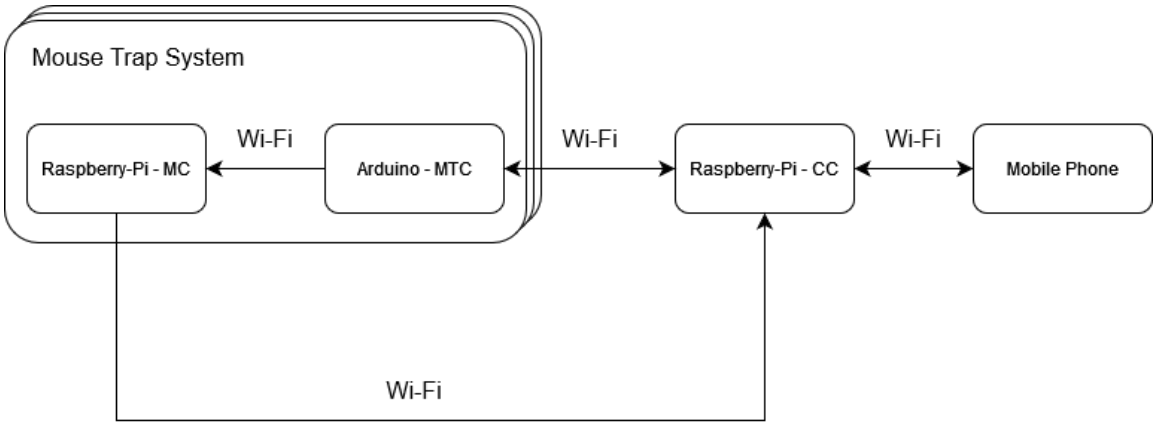


Fig. 3: Hardware architecture diagram (theoretical/ideal)

In the hardware architecture diagram (Fig. 3), we can identify four components:

1. **Arduino - MouseTrapController** (Arduino Uno + Mini Breadboard + Micro Servo Motor + IR Sensor + LCD + Wi-Fi Shield)
2. **Raspberry-Pi - MultimediaController** (Raspberry-Pi 4 + Webcam)
3. **Raspberry-Pi - CommunicationCenter** (Raspberry-Pi 4)
4. **Mobile Phone** (Android)

Going forward we will often use variants of the models and components which may be partially or fully abbreviated, nonetheless, the effort was made to keep the meaning clear throughout this report.

It is noteworthy that the diagram outlines the initial case we projected. To minimize the required materials, the implementation will involve only one **Mouse Trap System** and will utilize only one **Raspberry Pi 4** so that the functionalities of the **Raspberry Pi - CC** and **Raspberry Pi - MC** will be served in a single component.

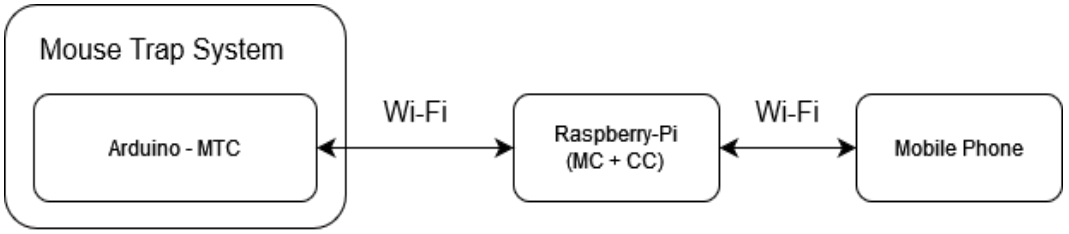


Fig. 4: Hardware architecture diagram (practical)

The communication between the **Arduino-MTC** and the **Raspberry Pi** is bidirectional and is done through a Wi-Fi channel.

The communication between the **Raspberry Pi** and the Mobile Phone occurs similarly to the description above.

Software Architecture:

The software architecture is composed of blocks that make up the various software modules running on **multiple devices**[1]. However, it is important to keep the software architecture independent from the hardware components, so we can later relate software modules and hardware components in a deployment diagram. The terminology used in the software architecture is the same used in the actors model.

As evidenced by *Figure 5*, the firmware is composed of control logic, message management, and communication channels, in our case **TCP/IP**. The control logic implements the functionalities present in the mouse trap system (**MTS**) when the main signals are sent. The message management is responsible for the coding/decoding of messages and for the message exchange logic. Transport-wise, the communications are managed by synchronous sockets in the **Wi-Fi TCP/IP** connections. The message management is also backed by **multithreading** for ease of management and connection maintenance.

Similarly, the User application is composed of communication channels, control logic, and message management. Additionally, it implements the user interface.

[1] Due to hardware constraints we will only simulate multiple systems, using a single hardware system.

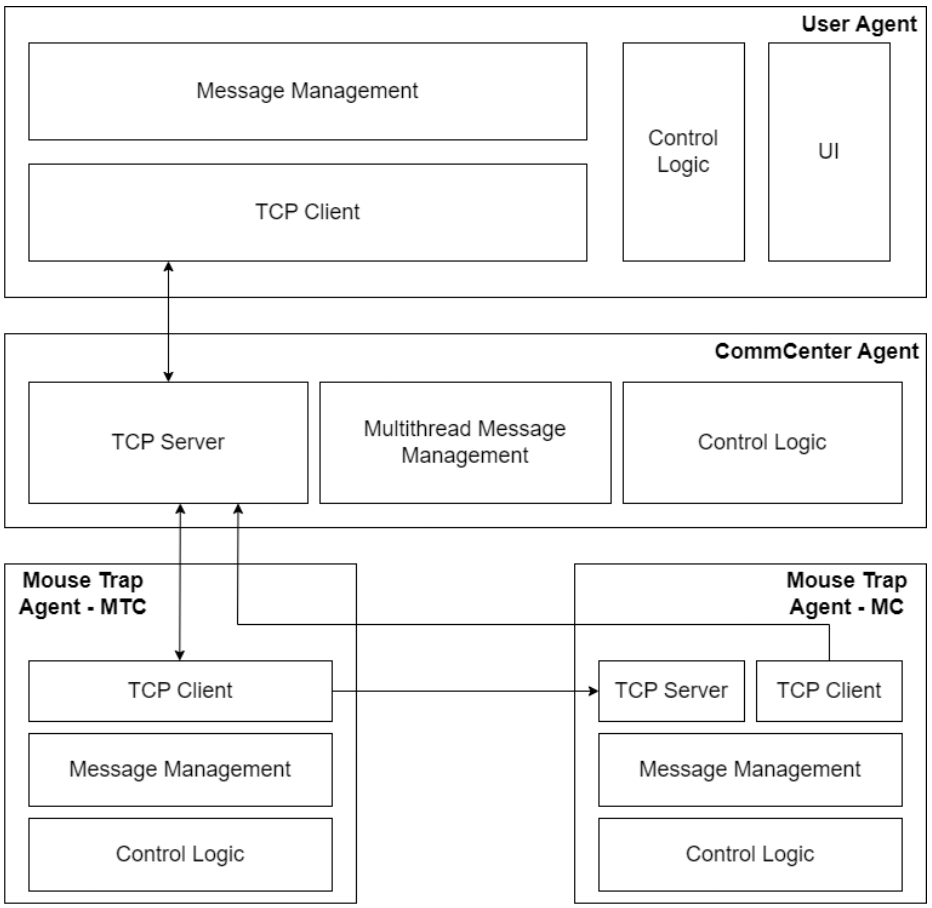


Fig. 5: Software architecture diagram

Deployment diagram:

The deployment diagram represents the configuration and architecture of the system and the connection between the software and hardware components. This diagram is shown in *Figure 6*.

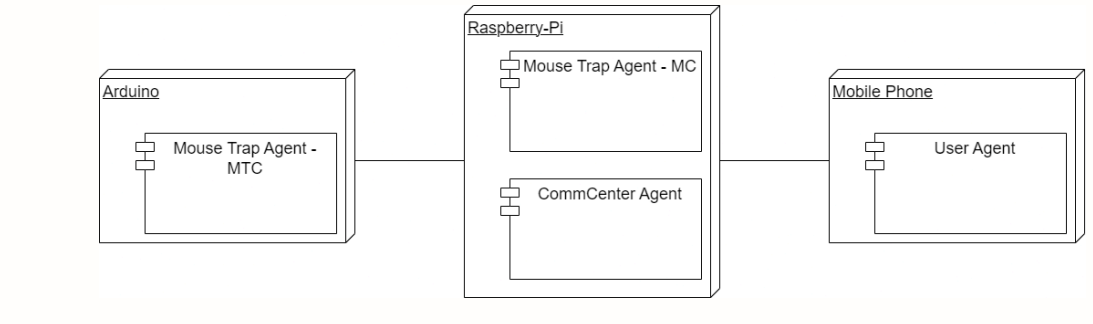


Fig. 6: Deployment diagram

Software Modeling:

In this section, we provide detailed explanations of the three flowcharts designed to depict the functions of the system's three main components: the **Mouse Trap Controller** (**MTC**), the **Multimedia Controller** (**MC**), and the **Communication Center** (**CommCenter**). In each iteration of the main "Loop" block within all three charts, a Wi-Fi connection test is conducted. If an error is identified, the communication channels are reset. Conversely, if no errors are detected, the received message is decoded to determine the appropriate course of action.

For the **Mouse Trap Controller**, the chart aims to discern the type of request received (**Or** - open door request / **Cr** - close door request). Following this determination, the corresponding action **(Write Digital**) is executed, and a notification is sent via Wi-Fi. Additionally, if an object is detected inside the trap (**Read Analog**), a response is once again transmitted via Wi-Fi.

The flowchart of the **Multimedia Controller** is more straightforward, focusing solely on identifying if a photo request was issued (**Pr**). If so, it proceeds to capture the photo and transmit it via Wi-Fi to the **CommCenter**.

Finally, the **CommCenter**'s flowchart illustrates its role as a broker. Upon receiving a request from the **User Agent**, it forwards the message to the **Mouse Trap Controller**. On the other hand, if it receives a notification regarding an event (**Se** - change of sensor states / **Pe** - camera information collected / **Oe** - motor state = open / **Ce** - motor state = close), from either the **Mouse Trap Controller** or the **Multimedia Controller** it relays the information gathered to the **User Agent**.

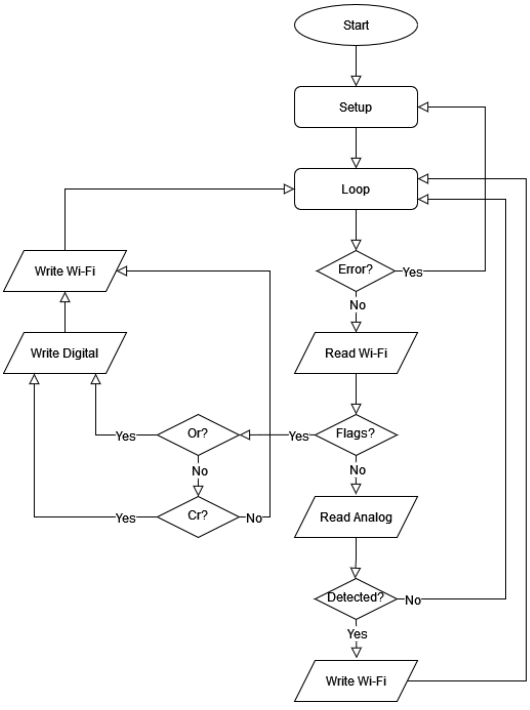


Fig. 7: Flowchart Mouse Trap Agent - Mouse Trap Controller

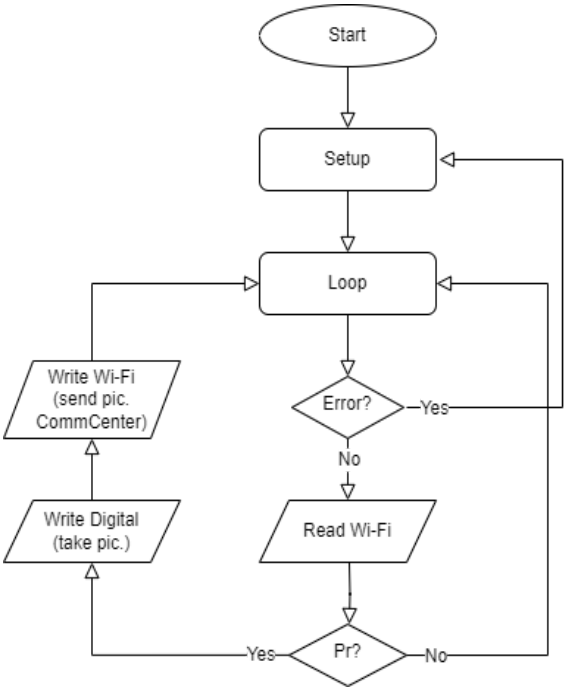


Fig. 8: Flowchart Mouse TrapAgent - MC

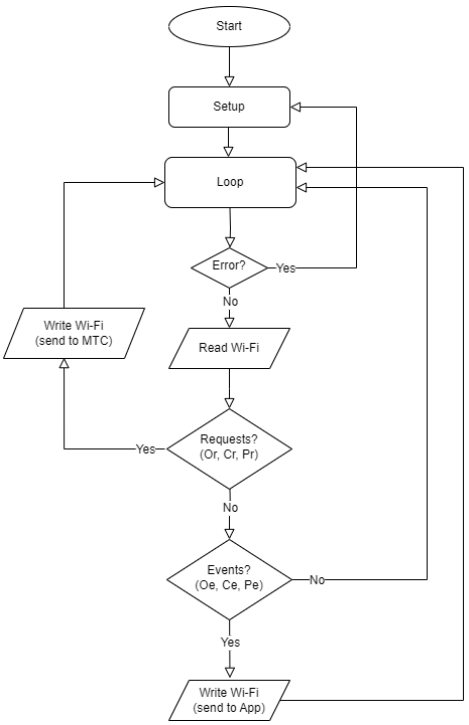


Fig. 9: Flowchart CommCenter

Gantt Chart overview:

The figure presented below, *Figure 10*, illustrates a view of the project's timeline and tasks using a Gantt chart. Through this visual representation, the Gantt chart delineates the schedule and tracks the progress of important project activities.

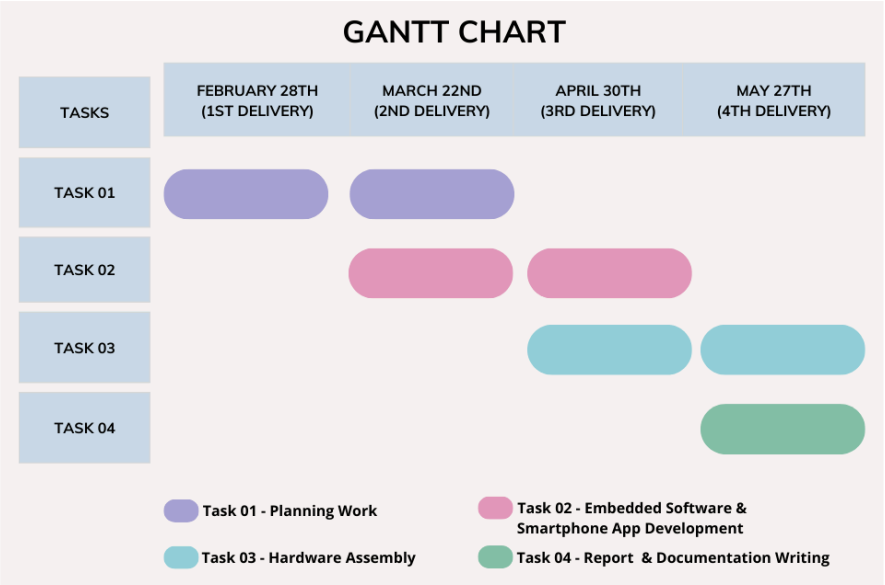


Fig. 10: Gantt Chart