Software Requirements Specification

YOLO

Group - 97

Mert Kaan YILMAZ - 2381093

Contents

List of Figures

List of Tables

Revision History

Date	Reason For Changes	Version
10.04.2022	Initial Draft	0.1
15.04.2022	Final Draft	1.0

Table 1: Revision History of Software Requirements Specification Document

1 Introduction

This document is the System Requirements Specification for YOLO project created by a small group of people.

1.1 Purpose of the System

Your Own Living Object (YOLO) is a minimalistic, standalone, portable robot with open-source software that aims to increase children's creativity during playtime. The purpose of this project is to build a small robotic toy named YOLO, which uses light, touching, and movement as inputs and has a reasonable weight and height to be grabbed and easily moved around by the children, just like the other traditional toys. YOLO can provide new ideas for their stories according to the child's movement generated while playing. This way, a social robot that increases children's creativity is designed for and with children.

1.2 Scope

- The system shall make its user feel that the robot is alive when there is no interaction with the robot by doing puppeteer kind of behaviour.
- The system shall move with respect to child's play style.
- The system shall react to child's story with different behaviours by creating different movement patterns.
- The robot shall react with it's LED lights according to if it is touched or its different social behaviour.
- The robot should perform behaviours with its creativity technique.
- Users shall be able to develop new functionalities and add new behaviour models.

1.3 System Overview

This section of the document provides detailed information about the system.

1.3.1 System Perspective

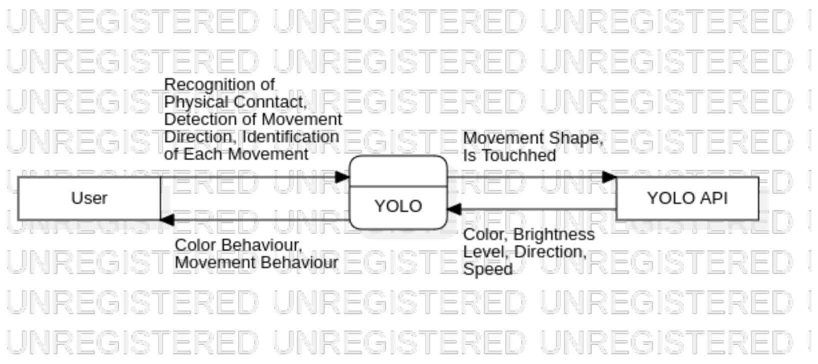

Recognition of Physical Conntact, Detection of Movement Direction, Identification of Each Movement
Movement Shape, Is Touchhed
User
YOLO
YOLO API
Color Behaviour, Movement Behaviour
Color, Brightness Level, Direction, Speed

Figure 1: System Context Diagram for YOLO

The YOLO software is not a part of a more extensive system, but it interacts with the physical components of the robot to get input and show behavioral outputs. Inside the robot's body, there is a Raspberry Pi that runs python scripts. In this way, the robot constantly stays connected with the real world with its sensors. There are three different sensors on the YOLO robot, such as touch, shape, and optical sensors, and they provide information to the YOLO system. In the system, these pieces of information are directed into relevant parts of the robot, which are called actuators, and according to the input, the robot shows some behaviors.

1.3.1.1 System Interfaces: Users who want to use YOLO robot, should download YOLO software from Github and install it on Raspberry Pi. After installation part, they are ready to get involved with the system interfaces. System interface can be separated by 2 subtitles such as hardware interfaces and software interfaces.

Hardware Interface: Hardware interface consist of different hardware structures such as sensors and actuators. These parts are which the user gets involved with. The input coming from touch and optical sensors are interpreted in Raspberry Pi with YOLO API, and after that necessary actions are taken in both software and hardware side.

Software Interface: YOLO run scripts on Rasberry Pi OS to generate outputs for different outputs. Software works with YOLO API with python scripts and according to input it generates required hardware outputs like behavioural actions.

1.3.1.2 User Interfaces: User interfaces are where users generally create and get inputs and outputs. These can be wheels, touch sensor, LEDS. There is no othher visual or digital interface between YOLO and user.

1.3.1.3 Communications Interfaces

1.3.2 System Functions

Function	**Summary**
Hello Behaviour	When the YOLO is on and detects child's first physical contact, robot performs hello behaviour.
Puppeteer Behaviour	When the YOLO is on and detects child's physical contact but it knows that's not the first time touching, robot refrains from performing any movement.
Make Attention Call	When child do not get involved with the robot for some time, the robot will perform attention call behaviour.
Reactive Behaviour	If children perform angular movements patterns with the robot, it will react to that movement previously performed by the children.
Proactive Behaviour	The robot can make an autonomous behavior to create new ideas during play time, so it can start moving around randomly to attract children for playing with them.
Blinking Behaviour	While the robot is working autonomously, it can show some blinking behaviours to express it's feelings.
Terminate Robot	If user wait enough without any interaction with robot, it shuts itself down.

Table 2: System Functions

1.3.3 Stakeholder Characteristics

The general target of the YOLO project is children who will use play with this robot to create their own creative scenes/stories. Basically, children use the robot as a character for their stories. Since design purpose of the robot is based on stimulating creativity, the most suitable user type is children. On the other hand, developers who have python programming knowledge background are also different group of users. Since the YOLO software project is open-source, they can contribute the project and design new functions to the overall system.

1.3.4 Limitations

- **Regulatory Requirements and Policies:** The YOLO project is completely ann open-source project which uses Creative Commons Public Licenses. Therefore, all the software and hardware codes are accessible for everyone.
- **Hardware Limitations:** Since the functionalities of the robot is limited, hardware options are also limited.
- **Interfaces to other applications:** There is no need for an interface, since the user will interact physically with the robot.
- **Parallel Operation:** The YOLO software is simple enough to work as non-paralel, so there is no such limitation.
- **Audit Functions:** The YOLO project has not any audit functions.
- **Control Functions:** Since the YOLO project is controlled by a single user, there is no control functions.
- **Higher-order Language Requirements:** There is no limitation on used programming language.
- **Signal Handshake Protocols:** There is no such limitation.
- **Quality Requirements:** Since YOLO is not part of a bigger system, there will be no limitations on stability, maintainability, and so on. Therefore, there is no quality requirement limitations

- **Criticality of the application:** Since YOLO cannot be counted as a critical system, there won't be critical effects if it fails.
- **Safety and security considerations:** Since users will be in contact with robot physically, and there is no possibility to record anything about user, there is no safety and security problem/limitation.
- **Physical/Mental considerations:** There is no physical/mental limitations.

1.4 Definitions

- API, Application Programming Interface

2 References

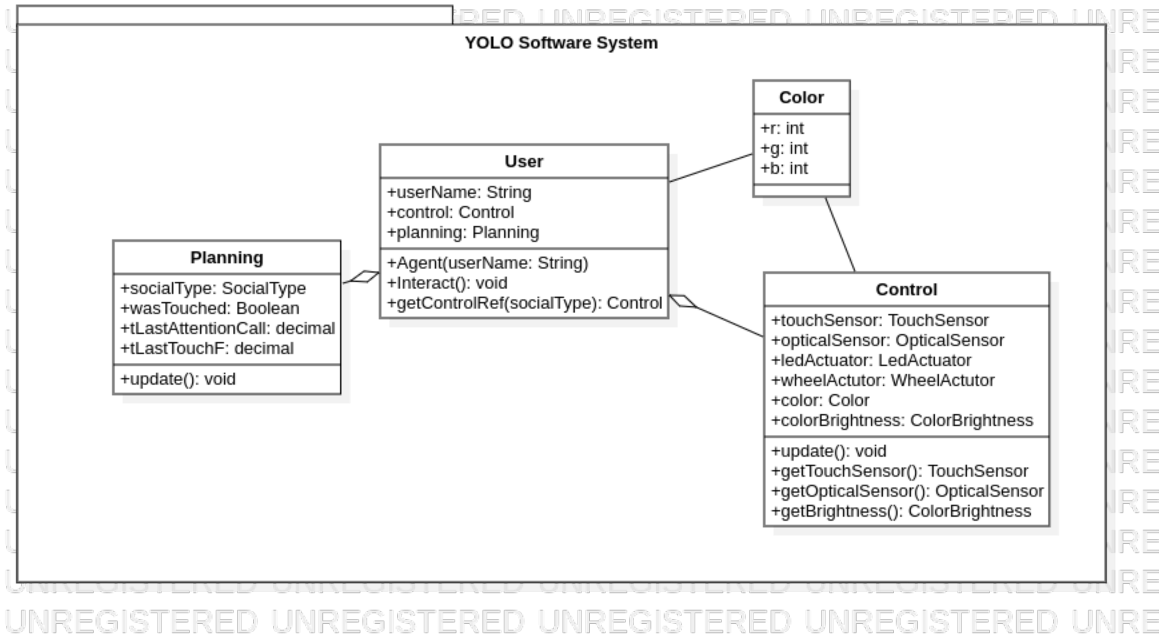
[1] 29148-2018 - ISO/IEC/IEEE International Standard - Systems and software engineering– Life cycle processes –Requirements engineering.

[2] Patrícia Alves-Oliveira, Samuel Gomes, Ankita Chandak, Patrícia Arriaga,Guy Hoffman, Ana Paiva. Software architecture for YOLO, a creativity-stimulating robot. (2020)

[3] YOLO software open-source code https://github.com/patricialvesoliveira/YOLO-Software

3 Specific Requirements

3.1 External Interfaces



YOLO Software System
Color
+r: int
+g: int
+b: int
User
+userName: String
+control: Control
+planning: Planning
+Agent(userName: String)
+Interact(): void
+getControlRef(socialType): Control
Planning
+socialType: SocialType
+wasTouched: Boolean
+tLastAttentionCall: decimal
+tLastTouchF: decimal
+update(): void
Control
+touchSensor: TouchSensor
+opticalSensor: OpticalSensor
+ledActuator: LedActuator
+wheelActutor: WheelActutor
+color: Color
+colorBrightness: ColorBrightness
+update(): void
+getTouchSensor(): TouchSensor
+getOpticalSensor(): OpticalSensor
+getBrightness(): ColorBrightness
UNREGISTERED UNREGISTERED UNREGISTERED UNREGISTERED UNRE

Figure 2: External Interfaces of YOLO System

3.2 Functions

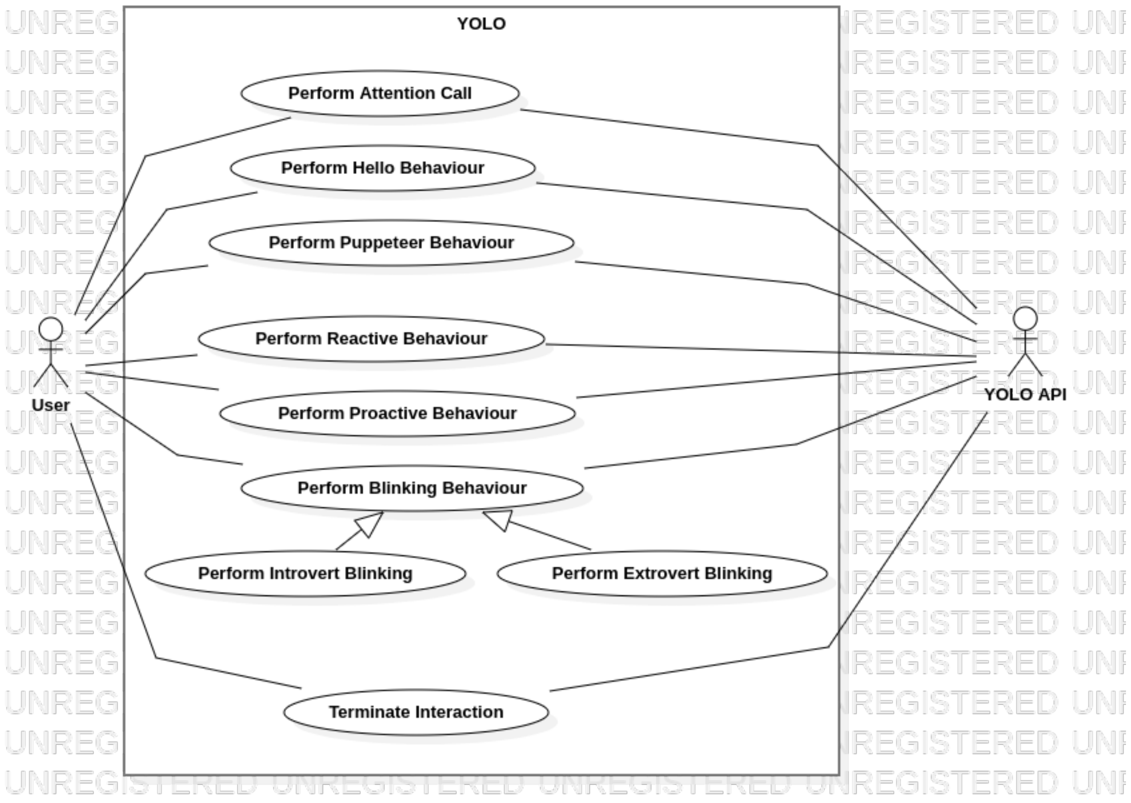

YOLO
Perform Attention Call
Perform Hello Behaviour
Perform Puppeteer Behaviour
Perform Reactive Behaviour
Perform Proactive Behaviour
Perform Blinking Behaviour
Perform Introvert Blinking
Perform Extrovert Blinking
Terminate Interaction
User
YOLO API

Figure 3: Use Case Diagram for YOLO

Use-Case Name	Terminate Interaction
Actors	User, Actuators
Description	This is some kind of a way to turn off the robot. If user waits long enough without interacting, robot shuts itself. (no longer accept interactions)
Data	-
Preconditions	-
Stimulus	Not touching the robot.
Basic Flow	Step 1 – User waits long enough without touching Step 2 – The robot stops waiting for new interaction and terminates.
Alternative Flow#1	-
Alternative Flow#2	-
Exception Flow	Step 1 – If user touches the robot before reaching maximum interaction time, it does not terminate."
Post Conditions	The robot stops waiting for interaction and terminates.

Table 3: Terminate Interaction

Use-Case Name	Perform Hello Behaviour
Actors	User, Actuators
Description	When the YOLO is on and detects child's physical contact and it know that's exactly the first time of touching, robot performs hello behaviour.
Data	Getting touched input from touch sensor.
Preconditions	Maximum interaction time should not be achieved.
Stimulus	Getting touched
Basic Flow	Step 1 – User turn on the robot Step 2 – For the first time, touching to activate touch sensor. Step 3 – User can see the hello behaviour.
Alternative Flow#1	-
Alternative Flow#2	-
Exception Flow	Step 2 – If user does not touch to robot for the first time, it does some other behaviour instead of hello behaviour."
Post Conditions	Robot goes back to idle behaviour.

Table 4: Perform Hello Behaviour

Use-Case Name	Make Attention Call
Actors	User, Actuators
Description	While the robot is in interaction time and user decides not to touch on the robot, after some time (exceeding attention call threshold), the robot will perform attention call behaviour.
Data	BLA
Preconditions	Time since last touch should exceed attention call threshold.
Stimulus	Not touching in interaction time.
Basic Flow	Step 1 – Robot shoul be in interaction time Step 2 – User should not touch robot. Step 3 – User should wait long enough to exceed attention call threshold.
Alternative Flow#1	-
Alternative Flow#2	-
Exception Flow	Step 2 – If user touches robot, it will perform different behaviour.
Post Conditions	The robot performs attention call behaviour.

Table 5: Make Attention Call

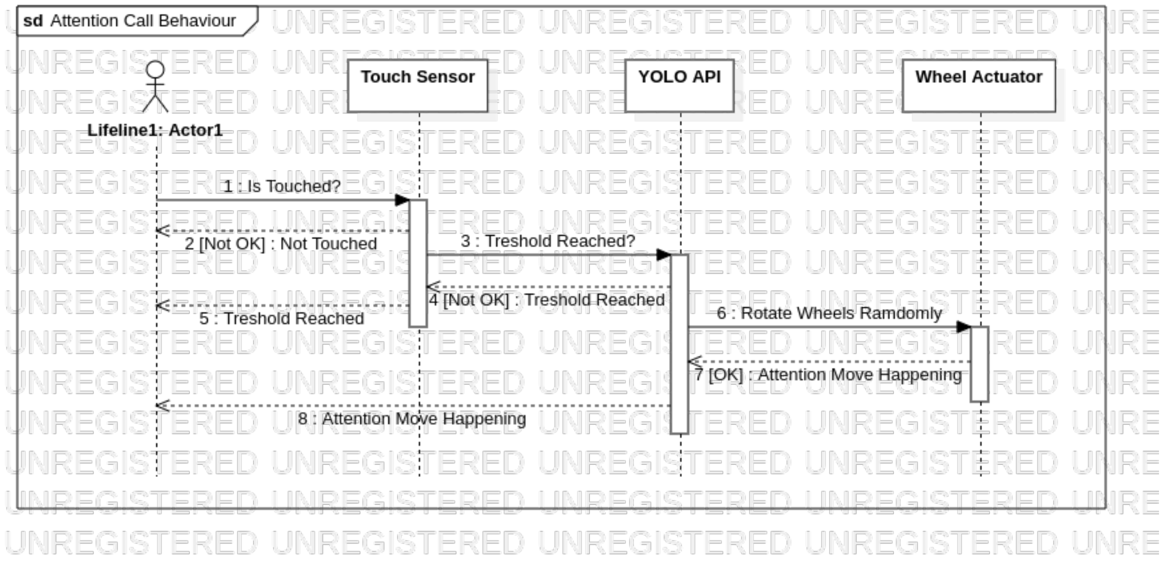

sd Attention Call Behaviour
Lifeline1: Actor1
Touch Sensor
YOLO API
Wheel Actuator
1 : Is Touched?
2 [Not OK] : Not Touched
3 : Treshold Reached?
4 [Not OK] : Treshold Reached
5 : Treshold Reached
6 : Rotate Wheels Ramdomly
7 [OK] : Attention Move Happening
8 : Attention Move Happening

Figure 4: Sequence Diagram Attention Call Behaviour

Use-Case Name	Perform Puppeteer Behaviour
Actors	User, Actuators
Description	When the YOLO is on and detects child's physical contact but it knows that's not the first time touching, robot refrains from performing any movement.
Data	Getting touched input from touch sensor.
Preconditions	Maximum interaction time should not be achieved.
Stimulus	Getting touched
Basic Flow	Step 1 – User turn on the robot Step 2 – At any time of playing, touching to activate touch sensor. Step 3 – User can see the puppeteer behaviour.
Alternative Flow#1	-
Alternative Flow#2	-
Exception Flow	Step 2 – If user touches to robot for the first time, it does some other behaviour instead of puppeteer behaviour."
Post Conditions	Robot goes back to idle behaviour.

Table 6: Perform Puppeteer Behaviour

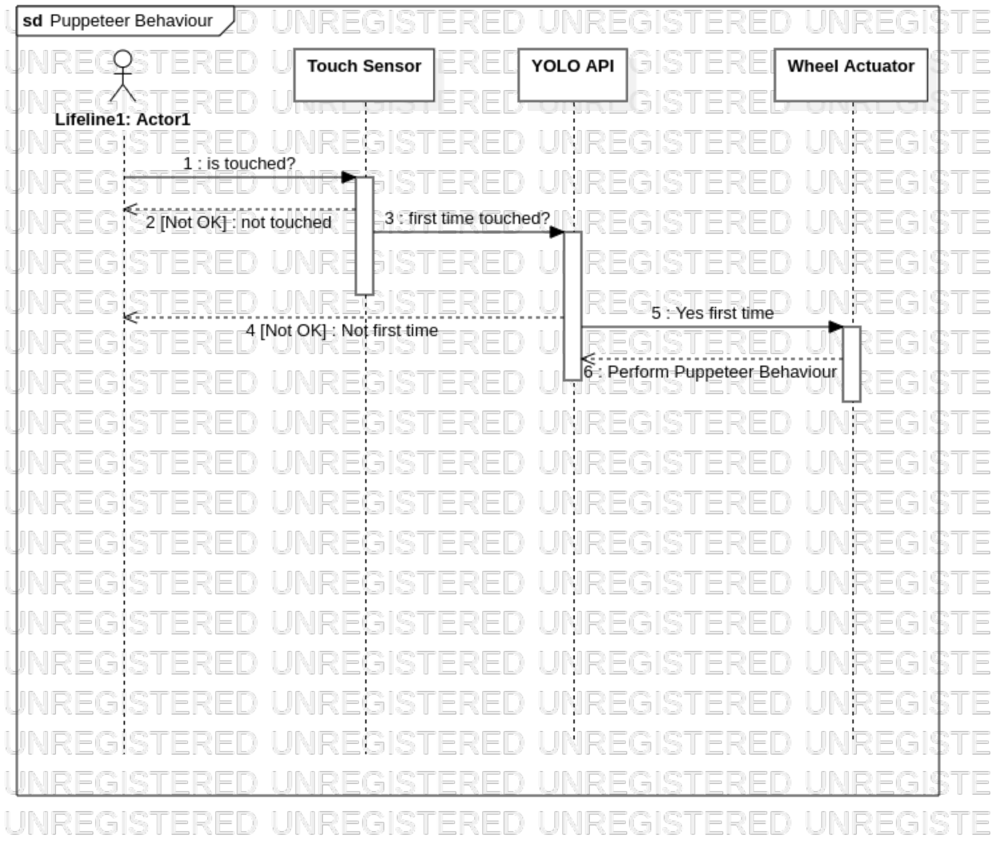


Figure 5: Sequence Diagram for Puppeteer Behaviour

Use-Case Name	Perform Reactive Behaviour
Actors	User, Actuators
Description	If children perform angular movements patterns with the robot (pretending, e.g., that the robot is avoiding obstacles, similarly to what children do when they play with car toys), the robot detects these and can react to them either by imitating them or by doing a different movement. In this case, the robot is reacting to a movement previously performed by children.
Data	Speed and direction from movement sensor.
Preconditions	Maximum interaction time should not be achieved and the robot needs to be moved.
Stimulus	Getting touched
Basic Flow	Step 1 – User turn on the robot Step 2 – At any time of playing, touching to activate touch sensor. Step 3 – User moves the robot with angular shape of movement. Step 4 – The robot either imitate the move or doing another move.
Alternative Flow#1	-
Alternative Flow#2	-
Exception Flow	Step 2 – If user choose not to touch the robot, it does some other behaviour.”
Post Conditions	Robot moves with different patterns or make move similar to one user moved the robot.

Table 7: Perform Reactive Behaviour

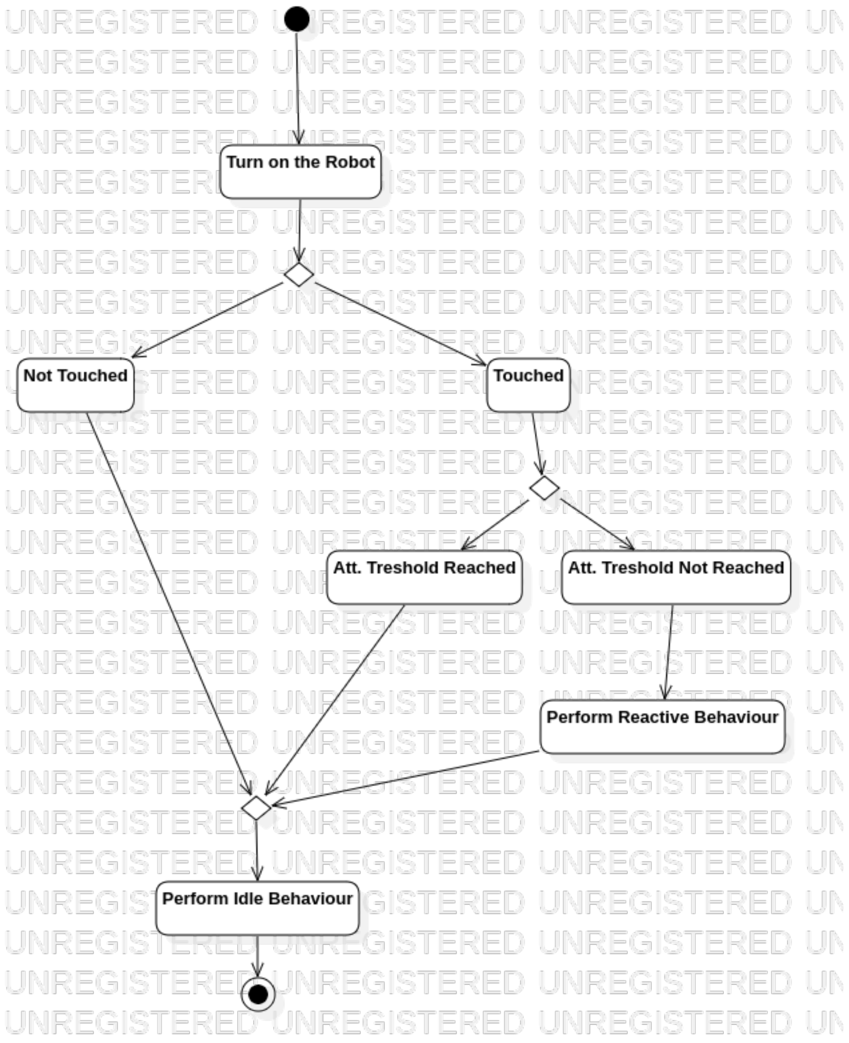


Figure 6: Activity Diagram for Reactive Behaviour

Use-Case Name	Perform Proactive Behaviour
Actors	User, Actuators
Description	The robot can make an autonomous behavior to create new ideas during play time, which means that the robot, without being previously touched/stimulated by children, can start moving around randomly to attract them for playing..
Data	-
Preconditions	The robot should not be touched for some time. (Attention call threshold should not be reached)
Stimulus	Not being touched.
Basic Flow	Step 1 – User turn on the robot Step 2 – User choose not to touch the robot. Step 3 – The robot initiates new random behaviours autonomously.
Alternative Flow#1	-
Alternative Flow#2	-
Exception Flow	Step 2 – If user choose to touch the robot in interaction time, it does another behavior instead of proactive behaviour.”
Post Conditions	Robot goes back to idle behaviour.

Table 8: Perform Proactive Behaviour

Use-Case Name	Perform Blinking Behaviour
Actors	User, Actuators
Description	While the robot is working autonomously, it can show some blinking behaviours to express it's feelings. For instance, if robot show introvert personality, blinking happens smoothly with less light, but for extrovert personality, it blink frequently with more light.
Data	-
Preconditions	The robot needs to work autonomously without touching.
Stimulus	Not touching in interaction time.
Basic Flow	Step 1 – Robot should be in interaction time Step 2 – User should not touch robot. Step 3 – User should wait to see emotions of the robot.
Alternative Flow#1	-
Alternative Flow#2	-
Exception Flow	Step 2 – If user touches robot, it will perform different behaviour.
Post Conditions	The robot goes back to idle behaviour.

Table 9: Perform Blinking Behaviour

3.3 Usability Requirements

- A user shall be able download and install the required source code easily.
- A user shall be able to use the robot after the installing process.
- A user shall create new functionalities for the project without breaking the current functionalities.

3.4 Performance Requirements

- Touch sensor response time should be short to perform different behaviours against different timing conditions.
- The number of simultaneous users to be maximum one, since it's not a web application and it's a toy to play.

- The user shall not have to wait for doing other actions to complete.

3.5 Logical Database Requirements

- ...

3.6 Design Constraints

- Users are able to get involved with YOLO robot while the robot is performing some behaviour.
- System aims to find a cheap way to develop and distribute the software. Therefore, all the hardware design and software development methods, source codes are open source and licensed under Creative Commons Attribution 4.0 International.

3.7 Software System Attributes

3.7.1 Reliability

- All the YOLO system (its hardware and software) code is open source.

3.7.2 Availability

- Software is always available when the robot has charge.

3.7.3 Security

- None of the sensors store information from the user such as photograph and sound. Therefore, there is no data going outside of the system.
- Robots learning data is stored locally while playing.

3.7.4 Maintainability

- If there is an update or problem on software, it must be able to get updated by using the terminal with the open-source code.
- Additional features must not harm the integrity of whole existing system functionality.

3.7.5 Portability

- YOLO software works on python, so overall code can be easily imported.
- YOLO software resides in a Raspberry Pi. If user wants to reach that code and change or port some part of it, user can reach it via wifi connection to Raspberry Pi via wifi router.

3.8 Supporting Information

YOLO softwre is an open-source project for children. A user can use the robot as an application programming interface that gives any user the opportunity to design personalized behaviors for YOLO. Additionally, users with programming background cann modify the design of the system and contribute to the project or his/her device as an open-source developer.

4 Suggestions to improve the existing system

In my opinion, the robot's functionality, both hardware and software wise, is very limited. The idea behind this project is to create a social robot, so it needs to react some external behaviours and also needs to create its own movements/stories. There might be other functionalities such as making noises. This way, in story telling arc, children can think as if the robot tries to speak with them and this kind of behaviour make the story look way more real.