

# SMART OBJECTS SAFARI

## Design of Smart Objects

thematic: sustainable mobility, transport and regulation



## Introduction

This protocol aims at **co-designing smart objects**, that are **intelligent objects** in the context of **smart cities**, capable of **perceiving the external environment through inputs, processing observed data and performing actions through actuators**.

Smart objects can also **interact with each other or with humans**. In the design it is important to make participants reflect on the **structure of the smart object and on the interaction with the external environment**.

The resources provided in this document include guidelines for performing the protocol with students of different levels and grades, detailing the proposed activities with phases, duration and practical examples to enable interested moderators or educators to replicate the proposed workshops.

This document also includes **evaluation tools for verifying the level of involvement and learning achieved** as a result of the activities conducted and the impact on awareness, on nature of smart objects and their role in smart cities.

The proposed structure was validated by researchers from a number of Italian universities with the support of the non-profit association Perlatecnica ETS and with a particular contribution from the Department of Informatics of the University of Salerno and the Free University of Bolzano, obtaining good results in terms of involvement and learning, results that will be detailed at the end of the manual.

### Interdisciplinarity



**Technology and engineering**  
**Physics & Chemistry**  
**Art & Design**

### Sustainable Development Goals

9  
INDUSTRY, INNOVATION  
AND INFRASTRUCTURE



15  
LIFE  
ON LAND



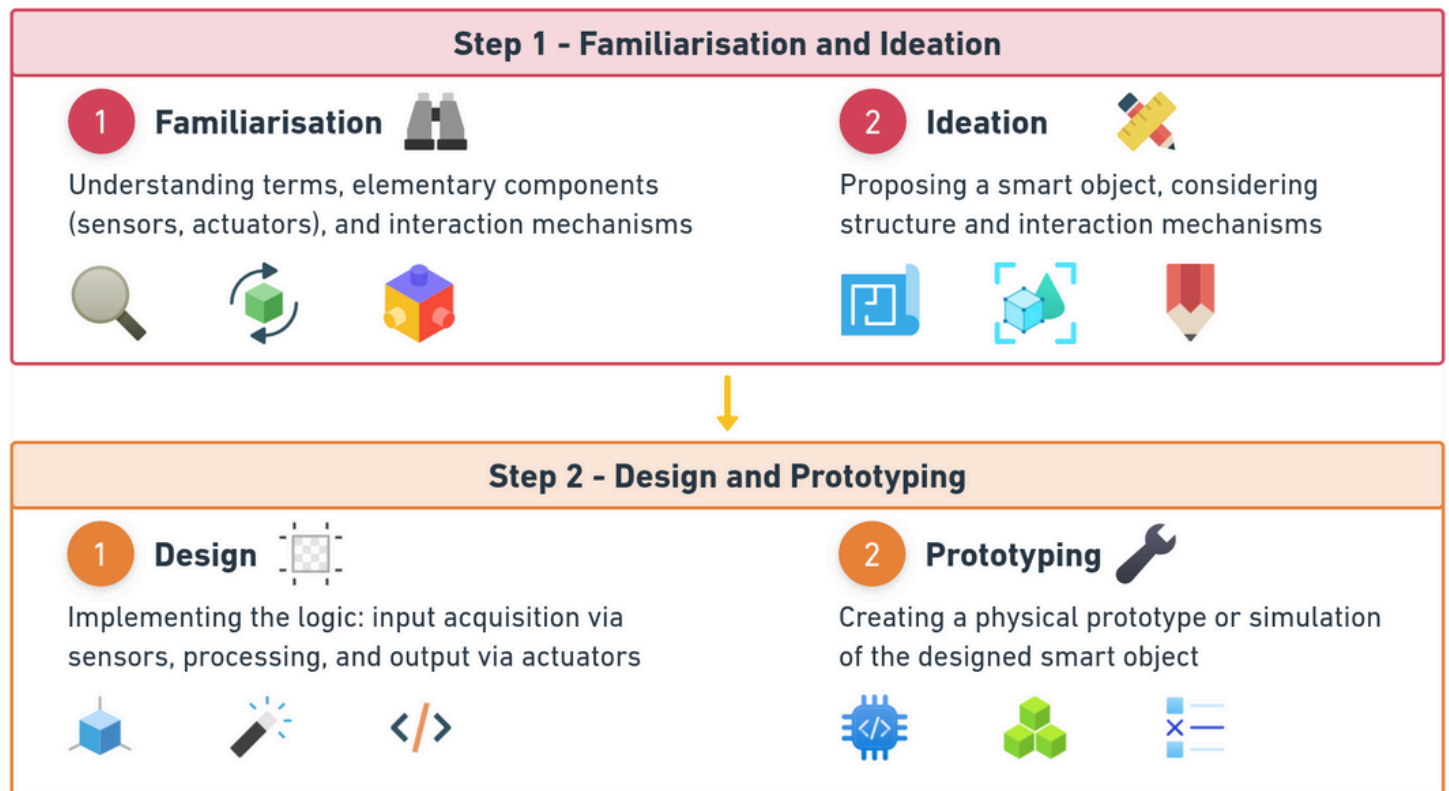


# Overview

## Protocol Structure

The Smart Object Safari Protocol is divided into **two complementary phases** building upon each other to create a comprehensive understanding of smart objects collaborative design.

Each phase integrates several sub-objectives, presented hereunder:



### Step 1 - Familiarisation and Ideation

During this step, participants are asked to familiarize themselves with the basic concepts behind this protocol and, subsequently, to propose an idea of the embryonic state of the object they want to design. The sub-objectives are the following:

1. The **familiarisation** with the terms, the elementary components of a smart object, such as sensors and actuators, and the interaction mechanisms of these elements analyzed in isolation
2. The **ideation** in which participants are asked to propose a smart object.



Depending on the age of the participants, it may be necessary to bind a scope and/or mission that contextualizes the objects to be designed.

For example, students can be invited to hypothesize how to make a house or park smart, or alternatively, a school or gymnasium. They can be invited to design objects that provide safety, monitor the health of the environment or make a shared space interactive and fun.

While the former are proposals for application areas, the latter are proposals for missions. It is important to think about the structure of the object and the mechanism of interaction right from the conception phase.

## **Step 2 - Design and Prototyping**

The second step is entirely dedicated to implementation, using a workshop approach combining short presentations of the constructs through a frontal approach and allowing students to immediately put the introduced construct into practice. They can build the project as the constructs are presented, using an incremental approach. When they feel it is not necessary to introduce the introduced construct in their project, they are invited to propose a simple example to demonstrate that the construct is clear and to justify why it is not necessary to exploit it in the designed object. The complexity of the examples proposed during the introductory part must be incremental, starting from the simplest version (such as no loops) to the most complex constructs. In the absence of time, constructs should be introduced giving priority to those that enable the implementation of intelligent objects.

The sub-objectives are the following:

- The **design** of the object logic by implementing the acquisition of 'inputs' via sensors, their processing and the determination of the reaction in terms of externalized output via actuators,
- The **prototyping** of the object, by means of physical prototype or simulation.

---

## **Getting started**

**Duration:** The total duration of the activities is at least **2 hours per day for 3 days**, as verified during activities moderated by researchers and educators from the Perlatecnica Association and the Universities of Salerno and the Free University of Bolzano.

**Level of difficulty:**



**Material needed:** The tools used can be *phygital* (digital + physical), favoring the physical component, especially in the familiarisation and prototyping phases.

**Some tips for organisation:**

- The **familiarisation** phase can be implemented as an individual or collaborative activity, using a traditional lecture approach or favoring interaction. For example, through a game-based approach or language-based exploration focusing on the etymology of words to master the vocabulary. A physical phase where the structure of the individual components is shown should be included.
- The **design** phase can be entirely physical, conducted as an individual or collaborative activity. It favors representing the object in terms of making it intelligent, engaging sensors and actuators, simulating behavior through IF... THEN... scenarios, and contextualizing the interaction through storytelling techniques. The design phase can also be purely digital, adopting textual or block programming languages depending on the age and skills of the students involved. While secondary school students can approach C or C++ as programming languages, primary school students or inexperienced ones should use visual languages like Make Code, which minimizes syntactic complexity and allows participants to focus on the logical implementation of the intelligent object.
- The **prototyping** phase should be as physical as possible to allow participants to 'get their hands dirty', gain first-hand experience building (simple) circuits, and verify the extent to which the realized object corresponds to the hypothesis. A collaborative approach is suggested to enable reflection and brainstorming.

# Glossary

Keywords & Concepts	Definitions
<b>Actuator</b>	A component of a smart object that performs a physical action in response to data or signals (e.g., motor, light, speaker).
<b>Conditions</b>	A set of conditions combined using logical operators (AND, OR) to define more advanced behaviors based on multiple criteria.
<b>Cycles / Loops</b>	A programming concept used to repeat an action or behavior multiple times (e.g., <i>onstart</i> , <i>forever</i> , or <i>repeat</i> in MakeCode).
<b>MakeCode</b>	A visual programming platform used to develop the behaviors of smart objects, primarily through blocks that represent coding functions.
<b>Self-Driving Vehicles</b>	Vehicles capable of moving without human intervention, using sensors, algorithms, and integrated intelligent systems.
<b>Sensor</b>	A device integrated into a smart object, capable of measuring environmental data (e.g., temperature, light, pressure) to inform the object's behavior.
<b>Smart Objects</b>	Physical objects equipped with sensors and actuators, enabling them to collect, interpret, and respond to environmental data.
<b>Storytelling</b>	A technique used to describe and contextualize the ideas of smart objects through short narratives, making it easier to understand their interactions with the environment.
<b>Workshop Approach</b>	A pedagogical method that combines theoretical presentations with immediate hands-on practice, promoting learning through the realization of concrete projects.



# Protocol

## Step 1 - Familiarisation and Ideation



**Background and description of the problem to be solved in this step:** During this first step, students will familiarise themselves with the basic concepts that enable discovering the Smart Object Safari protocol and, subsequently, propose an initial idea for the object they want to design.

**Learning Objectives:** Acquire the necessary terminology to proceed with the ideation phase. Terms to master: smart object, sensor, actuator, examples of sensors, examples of actuator. Stimulate imagination by inventing new smart objects based on the available sensors and actuators.

## Conceptualisation

Before starting the activities, it is necessary to collect the details on the **students' perception and level of awareness of smart cities and smart objects, or their expectations**. This pre-assessment will be needed to later estimate the impact of the proposed activities and perform the evaluation of the protocol.

Use engaging activities to transform traditional pre-assessment into participatory experiences. For instance:

- Consider implementing an **emoji meter assessment** where students select expressions representing their comfort level with specific topics you will be covering. For instance, ask them to show emojis reflecting their feelings about block programming, smart city concepts, or their familiarity with smart objects. This visual representation quickly reveals student attitudes toward each subject area.
- Alternatively, try a **knowledge positioning** exercise by designating different areas of your classroom as knowledge zones ranging from beginner to expert. Having students physically move to these zones creates an immediate visual map of your class's collective understanding of each topic you plan to teach.
- An **interactive expectations wall** can also be used by having students placing sticky notes on a chart indicating both their current proficiency and where they hope to be by the end of the unit. This becomes a living document that can be revisited throughout your teaching to track progress and celebrate growth.

These interactive assessment strategies provide valuable diagnostic information in an engaging manner, helping you identify knowledge gaps while establishing a positive learning environment from the start.

## Students Investigation

### Familiarisation - Creating Foundational Knowledge

During this first phase, the students need to acquire the necessary terminology to proceed with the ideation. Teachers can introduce the structure of the proposed activities in terms of duration, modalities, tools used, expectations and proceed with the introduction of smart objects by providing answers to questions such as:

#### 1. Introductory Questions



**Have you ever wondered what a smart object is? What makes an object intelligent?** Its intelligence lies in the ability to react autonomously to what it perceives around itself—from the environment, from interaction with people, and from other objects.

**Can all objects become intelligent?** Potentially yes, if equipped with a mechanism to autonomously react to external stimuli.



## 2. Examples of Smart Objects

**Is a washing machine intelligent?** A washing machine can be intelligent if it can adapt the duration of the wash cycle to the load or the level of dirt on the clothes.

**Can blinds become intelligent?** Yes, if they adjust their behavior based on light intensity.



## 3. What components make up a smart object?

**The object we want to make intelligent + One or more sensors to perceive external stimuli + One or more actuators to react.** Sensors can be compared to human senses, while actuators can be likened to limbs or voice that humans use to perform actions.

## 4. Sensors We Can Use

Thermometer to measure temperature

Accelerometer to detect and measure the acceleration of a moving object

Humidity sensor

Compass to manage orientation

Infrared sensors to determine distance from an external agent

Contact sensors for external agent interaction

Light sensors

**And more...**

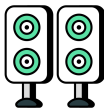
## 5. Actuators We Can Use

Speakers to convey a message

Displays to show a message

Motors

**And more...**



It is advisable to combine definitions with concrete examples of smart objects to make the familiarization phase more concrete (and digestible).

## Ideation Process

This phase aims to stimulate imagination by inventing new smart objects based on the available sensors and actuators.

At this stage, it is crucial to use a workshop approach, allowing students to work individually or in groups answering the mission given in the conceptualization stage: **"Designing smart objects that help or hinder self-driving vehicles"**.

Based on the familiarisation activity performed during the conceptualisation phase, students are familiar with the terminology and basic elements. Teachers can proceed with an activity to have the students propose **intelligent objects in terms of object representation by detailing their structure and components**, describing the object's behavior in terms of **WHEN/IF... THEN... conditions** and detailing the interaction with other objects, vehicles, and humans through **(short) narratives through storytelling**.

Students should be given the equipment or directions needed to complete the task. This phase can be realized as an **unplugged activity by narrating the structure and interaction with traditional tools**, such as pen and paper, possibly supported by card-based approaches. Storytelling can also be realized through a **digital approach using available storytelling and archiving tools**, such as the Storylet project (<http://www.isislab.it:19984/il-progetto-storylet>).

In order to facilitate students, especially younger and less experienced ones, determine the context and/or mission to which the proposed objects should respond. In the context of the SteamCity project initiatives, the suggested mission is **"Designing smart objects that help or hinder self-driving vehicles"**.

## Example of an Unplugged approach of the activity

In this activity, students will design smart objects for urban environments, specifically focusing on objects that can assist or hinder self-driving vehicles. The activity uses an unplugged approach with ideation boards and card decks, allowing students to engage with technology concepts without requiring computers.

Before the lesson, prepare an **ideation board** for each student and three decks of cards: **objects that could be made "smart," various sensors, and different actuators**. You might include traffic lights, benches, trash cans, and street signs as objects; proximity sensors, temperature sensors, and light sensors in the sensor deck; and LEDs, speakers, and motors in the actuator deck.




Begin by confirming that students understand the basic concepts of smart objects, sensors, and actuators from previous lessons. Then introduce the activity with a contextual story:

“




Yesterday, I received a call from our town's mayor who asked if we could help design smart objects to assist or hinder self-driving vehicles in our city. The mayor has granted us complete freedom to create objects for any purpose—ranging from entertainment to traffic management, or from information kiosks to emergency response systems.

”

### Overview of the Iteration Board

<b>OBJECT</b> 	Description of object behavior as a <b>story reporting who/what triggers the behavior and the object's reaction</b>
<b>SENSOR</b>  <b>ACTUATOR</b> 	Description of behavior in terms of <b>WHEN &lt;CONDITION&gt; THEN &lt;REACTION&gt;</b>

### Example of the Iteration Board

<b>TRAFFIC LIGHT</b> 	The traffic light interacts with self-driving vehicles. When a vehicle approaches, it gives way to vehicles, stopping pedestrians.
<b>DISTANCE</b>  <b>LED</b> 	WHEN vehicles approaching THEN show yellow light for pedestrians for 10 seconds show green light for vehicles for 60 seconds show red light for vehicles show green light for pedestrians

As students work on their own designs, circulate around the room providing guidance. Encourage them to think about both aspects of behavior description. The structured WHEN-THEN format helps ensure their design is **logically consistent**, while the narrative description helps them **consider real-world interactions**.

Some students might find it challenging to envision how their object interacts with the environment. Ask prompting questions like: **"Who or what would use this object?"** or **"What problem does this solve for people or vehicles?"** Others might struggle to translate narrative into structured rules. Help them by asking, **"What exact condition triggers this behavior?"** and **"What happens step by step afterward?"**.

For students who finish quickly, suggest creating **more complex objects** that respond to multiple conditions. For example, a smart street lamp that dims when no one is nearby to save energy, but brightens when it detects movement, and flashes in an emergency. If the ideation board seems too constraining for some students, offer **blank paper as an alternative**. The important thing is that they consider both the components (object, sensors, actuators) and the behavior (narrative and structured).

While this example focuses on smart objects for self-driving vehicles, the same approach works for other contexts. You might challenge students to design objects for a smart park, smart school, or smart home. Simply adjust the available object cards to match the new context. The dual description approach remains valuable across contexts. **The narrative description helps students connect their design to human needs and experiences, while the structured description prepares them for the programming phase that will come later.**



**Hints for Teachers:** When guiding students through this activity, keep several important considerations in mind:



- **Focus on autonomous objects.** The smart objects should be automatic systems that perceive external conditions and respond accordingly. Many students, especially those with gaming experience, might be tempted to design objects that primarily respond to direct human interaction rather than environmental conditions. Gently redirect them toward truly autonomous systems.
- **Help students balance complexity in their designs.** Discourage overly simple objects that use only a single sensor and actuator with a trivial response. When you encounter such basic designs, encourage students to add complexity by considering additional conditions or more sophisticated responses. Conversely, some students might design extremely complex systems with numerous components and intricate behaviors. In these cases, help them prioritize the most important behaviors to focus on.
- **Ensure coherence between components and behavior.** One of your key roles is verifying that students' object representations (sensors and actuators) align logically with their described behaviors. If a behavior requires detecting distance but no proximity sensor is included, point out this inconsistency.
- **Consider incorporating peer review by randomly assigning ideas to students and having each present someone else's design.** This approach encourages careful listening during presentations and helps students see their own ideas through others' perspectives.

## Conclusion & Further Reflexion

Allow approximately **15 minutes at the end of the ideation phase for reflection and sharing**. Collect the proposed ideas and encourage students to briefly present their designs to the class. This sharing session serves multiple purposes. It promotes **reflection through collaborative brainstorming**, where students can see how others approached the same challenge. The discussion naturally leads to evaluation of possible improvements or modifications to the designs. Students and teacher together can assess the feasibility of implementing each design in later phases.

This sharing phase also gives you an opportunity to **standardize the complexity of projects based on your students' skills, the tools you have available, and practical constraints like time limitations or sensor availability**. If some projects seem too ambitious given your classroom circumstances, this is the time to help refine them.

This activity bridges conceptual understanding and practical implementation. By thinking both in stories (how their object fits into the world) and in structured rules (how it will be programmed), students develop both creative and computational thinking skills.



## Step 2 - Design and Prototyping



**Background and description of the problem to be solved in this step:** The second step is entirely dedicated to implementation, combining short presentations of the constructs and putting the introduced construct into practice. Students can build the project as the constructs are presented, using an incremental approach. When they feel it is not necessary to introduce the introduced construct in their project, they are invited to propose a simple example to demonstrate that the construct is clear and to justify why it is not necessary to exploit it in the designed object. The complexity of the examples proposed during the introductory part must be incremental, starting from the simplest version (such as no loops) to the most complex constructs. In the absence of time, constructs should be introduced giving priority to those that enable the implementation of intelligent objects.

**Learning Objectives:** Acquire or consolidate/refine knowledge of programming constructs such as loops, conditions and comparisons. Apply the presented constructs to the implementation of the proposed object.

### Conceptualisation

This implementation phase follows the ideation session where students designed smart objects to help or hinder autonomous vehicles. During these implementation sessions, students will bring their designs to life using block-based programming. The teaching approach combines brief introductions to programming concepts with immediate hands-on application, allowing students to build their projects incrementally as they learn each new concept.

The activities are structured to accommodate students with varying levels of programming experience. Those who are more experienced can extend their implementations with more complex behaviors, while beginners can focus on basic functionality. Throughout the process, students are encouraged to refine their original ideas based on what they learn about implementation possibilities.

#### Opening Roundtable Discussion

Before discovering the activity, conduct a brief roundtable discussion about block programming. This gives you insight into students' prior knowledge and attitudes while creating a collaborative atmosphere. Ask questions such as **“Has anyone used block programming before? Which platforms have you tried?”**, **“What do you find easy or challenging about programming?”** or **“What kinds of projects would you like to create with programming?”**.

This discussion serves multiple purposes, revealing the range of experience levels in your class and helping students connect with peers who have similar interests or complementary skills. It also allows more experienced students to share their enthusiasm with beginners. Additionally, it gives you insight into which concepts might need more attention, while establishing programming as a collaborative rather than solitary activity. Take note of students' responses to tailor your guidance throughout the implementation sessions.

#### Setting the Context

Begin by connecting the implementation work with the previous ideation session:



Now we will move to the programming phase to implement your smart object ideas. During our implementation sessions, we will introduce programming constructs that will allow you to implement your smart object's behavior. Remember that we are implementing the logic that determines how your object reacts to external stimuli, keeping in mind the structure of your object in terms of sensors and actuators. We will first look at practical examples together, and then you'll apply what you've learned to implement your own object.



Remind students that the implementation phase allows them to program the behavior and logic of their object, assuming the presence of sensors and actuators that would be configured during a prototyping phase. Younger or less experienced students might forget the overall vision during programming, so it's crucial to regularly refer them back to their original idea to check consistency and evaluate possible modifications to their proposed object.

## Students Investigation

### 1. Introduction to loops

For the first 10 minutes, the teacher can present the **concept of loops through slides and examples in MakeCode**, explaining what a loop is and how to use it. Using an incremental complexity approach, begin with the **absence of a loop (modeled in MakeCode by the onstart block)** and proceed with the **simplest loop, the forever loop** according to MakeCode terminology, which is an essential element in designing a smart object.

An example is provided at the following link [https://makecode.microbit.org/\\_C2yeP6E4xJrR](https://makecode.microbit.org/_C2yeP6E4xJrR) where the teacher can start from a simple code (such as an animation of a beating heart) in the onstart block showing that it is executed only once. The teacher can move the code from the onstart block to the forever loop encouraging students to notice the differences. In the spectrum from no-loop to forever, the teacher can then introduce intermediate examples, such as the repeat loop by showing that the loop runs for a specific number of iterations. The teacher may invite students to guess how to repeat the loop 5 times, then 10.

#### Students' Programme



Once the teacher perceives that the students have understood the mechanism of loops and how to implement them in MakeCode, they can stop sharing the screen and ask students to **implement the initial behavior of their object and a basic behavior to be repeated cyclically according to the idea they developed in the previous session**. Encourage them to distinguish between behavior that should happen only at startup (initialization) and what should happen cyclically.



It should be made clear that students **may modify their idea during the course**, but must explicitly **note down their thoughts on the modification and keep track of the reason for the modification** (recognized malfunction, discovery of new implementable functionality, etc.).

**Set up a file or collection point for the projects.** Have all students share their programs there (it can be a Google Shared File where they paste the sharing link to their MakeCode Editor for instance). At each stage, you can use this record to **assess the programmes provided by the students** to verify that the proposed solution contains the required blocks and ask each student to briefly introduce their project at its current state. **Encourage students to identify any logical and syntactical errors and guide them in making corrections.**

### 2. Introduction of simple conditions and comparisons

Along the lines of the introduction of loops, the teacher can introduce simple conditions and comparisons through examples of increasing complexity by sharing the screen and presenting slides and examples in MakeCode. Starting from the example available at [https://makecode.microbit.org/\\_Led0ghTFY8xy](https://makecode.microbit.org/_Led0ghTFY8xy), the teacher can:

- Show the example of interaction with button A
- Modify the example so that it exhibits the same behavior if button B is pressed
- Show a different behavior if the same condition occurs, e.g. play a sound instead of printing a string on the screen
- Add the else to have an input-dependent behavior. For example, IF button A is pressed, you say HELLO, otherwise you say ZZZ

Starting from the example available at [https://makecode.microbit.org/\\_ewq3W0EwmHjJ](https://makecode.microbit.org/_ewq3W0EwmHjJ), the teacher may:

- Show the example of a heart when the temperature is below 20
- Modify the conditions showing the different behavior according to the comparison by using > or = instead of <
- Change the input, e.g. by replacing the temperature sensor with the light sensor
- Change the behavior by replacing the output, e.g. replacing the screen with a text or changing the heart to a sun
- Add the else, e.g. showing a sun if the temperature is above 20, otherwise a snowflake

### Students' Programme



Once the guided presentation of the examples has been completed and any doubts of the students have been clarified, the teacher stops sharing the screen and challenges the students to **refine the behavior of their object by adding at least one condition and comparison.**



Set up a new section in the file for collecting the projects. Have all students **share their program by saving a new version of their project.** Open them to verify that the proposed solution contains **conditions** and ask each student to **briefly introduce their project at its current state.** Encourage students to identify any logical and syntactical errors and guide them in making corrections or suggesting extensions of the behavior.

### 3. Introduction to Complex Conditions

As with the introduction of the other constructs, the teacher shares the screen and begins to present the complex conditions through slides and examples in MakeCode. Starting from the example available at <https://makecode.microbit.org/LK0UUU2pUXoo>, the teacher:

- Shows the example where a heart is shown if it is cold (temperature is below 20) or the A button is pressed
- Modifies the example by showing the differences replacing OR with AND
- Modifies the example according to different inputs, e.g. by substituting the light for the button, e.g. if it is hot, and it is daytime, it shows the sun
- Adds the else, e.g. as an alternative to the sun showing the moon

### Students' Programme



Once the guided presentation of the examples is finished and any doubts of the students have been clarified, the teacher stops sharing the screen and challenges the students to **refine the behavior of their object by adding at least one complex condition.**



As for the previous step, have all students **share their programs.** Open them to verify that the proposed solution contains the complex condition. As it is the last step, ask students one by one to share and present their projects collectively, highlighting whether they hinder or help autonomous vehicles. Everyone is invited to propose improvements or identify errors. Ask the students to refine and finalize the proposal of ideas according to the advice received. This phase ends with the final collection of entries.

## Conclusion & Further Reflexion

This implementation phase has guided students through **applying programming concepts to bring their smart object designs to life.** By incrementally introducing and applying **loops, simple conditions, and complex logic,** students have built **functional programs that simulate their objects' behavior.**

Throughout this process, students were encouraged to think both about the **technical aspects** of programming and the **real-world behavior of their smart objects.** The repeated cycles of implementation followed by reflection and refinement helped students develop not just coding skills but also critical thinking about how their designs would function in practice.

The final sharing session allowed students to see **diverse approaches to similar challenges and learn from their peers' work**. This collaborative learning environment reinforces that there are multiple valid solutions to the same problem, while still emphasizing the importance of logical consistency and functionality.

For many students, this may have been their first experience implementing an original design in code. The combination of **creativity** from the ideation phase with the **logical thinking** required for implementation helps students understand how programming can be used to **solve real-world problems** and enhance our environment with smart technology.

As they continue their learning journey, students can build on these foundational programming concepts to **create increasingly sophisticated smart object designs and implementations**.



# Bibliography

- [1] Eftychia Roumelioti, Maria Angela Pellegrino, Rosella Gennari, and Mauro D'Angelo. 2022. What Children Learn in Smart-Thing Design at a Distance: An Exploratory Investigation. In Methodologies and Intelligent Systems for Technology Enhanced Learning, 11th International Conference. Springer International Publishing, Cham, 22-31. [https://doi.org/10.1007/978-3-030-86618-1\\_3](https://doi.org/10.1007/978-3-030-86618-1_3).
- [2] Eftychia Roumelioti, Maria Angela Pellegrino, Mehdi Rizvi, Mauro D'Angelo, and Rosella Gennari. 2022. Smart-thing design by children at a distance: How to engage them and make them learn. International Journal of Child-Computer Interaction 33 (2022), 100482.
- [3] Maria Angela Pellegrino, Eftychia Roumelioti, Mauro D'Angelo, and Rosella Gennari. 2021. Engaging Children in Remotely Ideating and Programming Smart Things. In CHIItaly 2021: 14th Biannual Conference of the Italian SIGCHI Chapter (CHIItaly '21). Association for Computing Machinery, New York, NY, USA, Article 20, 1-5. <https://doi.org/10.1145/3464385.3464728>.
- [4] Maria Angela Pellegrino and Mauro D'Angelo. 2021. Engaging Children in Smart Thing Ideation via Storytelling. I-CITIES. <https://icities2021.unisa.it>