Design challenges for the TPBot

In this challenge, students step into the role of real-world inventors using the design thinking process as their guide. They'll tackle a 'big picture' problem that can be solved with a TPBot, by imagining, planning, and testing their own unique solutions. By starting with user stories - short snapshots of what a person needs, students learn to design with a purpose. Building not just technical skills, but the empathy and curiosity needed to truly understand the people they are designing for.

Curriculum links

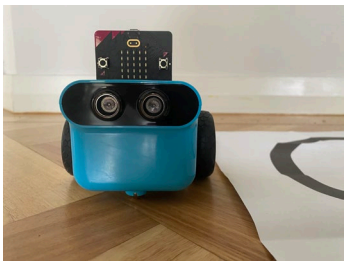
We encourage teachers to adjust the content and focus to suit the needs of their students. The activity can be adapted for students between Year 3 and Year 6. More details are provided in the supporting documents.

Digital Technologies

Required resources

You will need the following materials (most supplied in our Lending Library kit):

- BBC micro:bit V2
- TPBot smart car (with batteries)
- USB cable for download to USB
- Laptop or tablet with Microsoft MakeCode editor
- Access to physical materials (boxes, coloured paper, construction materials) to simulate the environment (optional)



The activity has been designed to take place over multiple classroom sessions to allow sufficient time to go through the design thinking process, and research and develop the required skills to create their project.

Pre-requisite knowledge

Prior to this session, students should:

- understand basic programming structures: sequences, simple loops, and conditionals (if statements).
- have basic experience using MakeCode or block-based editors.
- understand how to download code to a micro:bit.
- be familiar with TPBot - we suggest that they have completed the experiences in the CSER Introduction to TPBot activities

Design thinking challenge

Following a design thinking process, students will use their TPBot to solve a real-world problem. They will determine the scenario (the problem to be solved). It can be framed as four stages: understand, ideate, model and reflect.

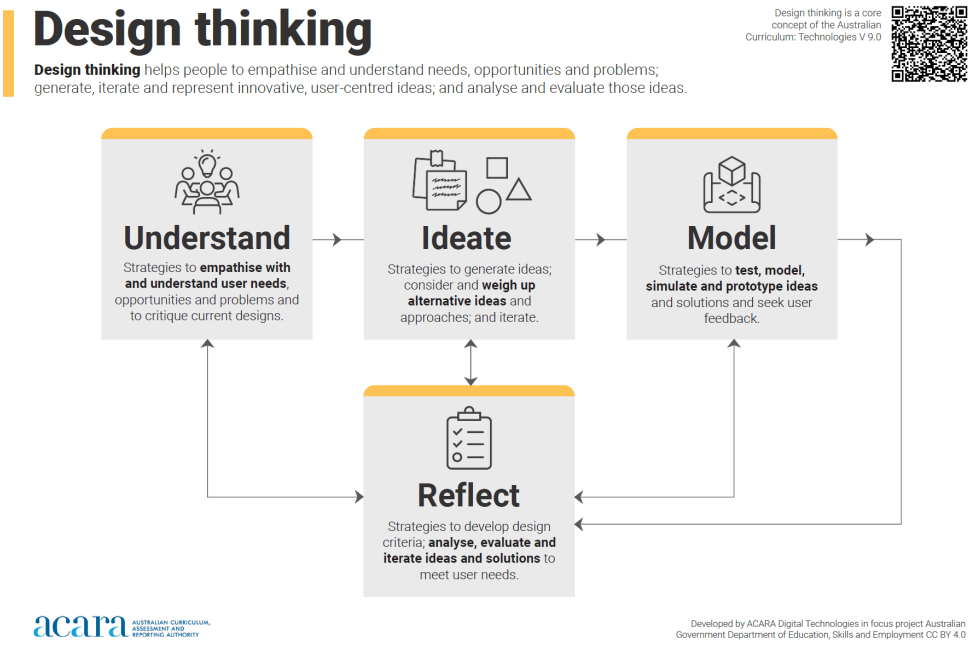


Image: ACARA

According to ACARA, user stories are "Short, simple descriptions of key software features that end users want from a digital solution. They often describe why the user wants particular features, and are part of the iterative design process." Students can be introduced to the following framework to help them create user stories.

User story

As a < insert a user > I want < insert a goal > so that < insert a reason >

for whom? do what? why?

In the following table, we have detailed the design thinking stages for a suggested TPBot Emergency response robot challenge as an example of the thinking required at each stage.

We have provided an example user story that could be used for students in Year 3 & 4 however, it is expected that students in Years 5 & 6 develop their own user stories.

Example scenario and process

Emergency response robot	**Design challenge:** Your emergency response vehicle (TPBot) must carry supplies to a disaster zone, navigating paths, choosing between safe/unsafe routes, and stopping at marked rescue points. **User story:** As a helper in an emergency, I want the TPBot to travel safely and stop at rescue points with supplies, so that people can get help quickly and safely.
Understand	Before you build anything, you need to be a "detective." You must understand who you are helping and what the actual problem is.
	Explore the problem of creating an emergency response vehicle (TPBot) and who it will help, such as people who need fast assistance in a disaster scenario. Identify needs and constraints like responding quickly, travelling safely along lines, avoiding obstacles and clearly signalling that it is an emergency vehicle. Identify **design criteria** for evaluating the final solution, for example: • Does the TPBot navigate the planned route safely and accurately? • Does it stop at or avoid obstacles? • Is the emergency signal clear and easy to recognise? • Did the TPBot deliver supplies?
Ideate	This is the "brainstorming" phase. There are no bad ideas! The goal is to come up with as many creative solutions as possible.
	Brainstorm different ways an emergency response vehicle could move, sense and communicate to solve the problem. Sketch and discuss options such as siren sounds, flashing lights, speed changes and route choices, then decide which features best help the robot reach people safely and quickly. Use labelled diagrams to explain features.
Model	Now it's time to make your ideas "real." A prototype isn't a finished product; it's a quick model used to see if your idea works.
	Turn ideas into a working emergency-response prototype by creating an algorithm which will then be implemented using MakeCode. Include features of the TPBot and micro:bit such as sensors, lights, actions and sound. Create any props or items to help illustrate the scenario. Test the robot on the floor track, observe how it follows lines, avoids obstacles and signals emergencies, then refine the code if required.
Reflect	Look back at what you made. Does it actually solve the problem? This is where you get feedback and think about how to make it even better.
	Consider how the emergency response vehicle met the original needs and design criteria and what could be changed next time. Consider feedback from peers and the teacher, discuss what worked or did not (for example, if the siren was clear or the robot stopped in time), and link learning to how real emergency vehicles use similar technologies.

Use or adapt any of the ideas below to challenge the students to follow the design thinking process to use the TPBot and micro:bit to solve a problem. **You could also encourage students to create their own scenario.**

Possible design challenges	
TPBot tour guide	Program the TPBot to move around a mini city or museum layout, stopping at key locations where they trigger messages on the micro:bit LED (or spoken explanations by students) about landmarks, historical events, or artworks.
Sustainable city cleaner	Code the TPBot to act as a street sweeper that follows paths (taped lines) to collect or push paper rubbish into a recycling zone, integrating discussion about sustainability.
Dance or parade routine	Choreograph a TPBot dance to music, using timed movements, turns, and LED patterns to create a performance, optionally syncing multiple robots as a group project.
Sports or competition bot	Create a mini soccer game or challenge where TPBots must push a puck into a goal area or stay inside a circular ring using sensors to detect borders.

Sample code ideas	Many sample codes for TPBot are available on Elecfreaks - Learn page. These include skills that could be useful in this challenge including: • Seeking light - set the TPBot heading to the light • Line following and obstacle avoidance - drives along with the black line and stops when it detects obstacles. • Light controls - control the colour of the LED lights. • Shy TPBot - adjusts driving speed according to the sound level of the surrounding environment • Running control - Learn to control the movement of TPBot

Why is this relevant? (Real world connections)

This lesson connects to the real world by showing students how robots and automated systems can be programmed to follow instructions, just like machines used in warehouses, hospitals, and farms. It helps them understand that digital systems, sensors, and coding are part of many everyday technologies, preparing them for future jobs and problem-solving in a digital world.

Assessment

Observation can be used to check students' ability to carry out tasks aligned to the Australian Curriculum. We have included some suggested questions for teachers to reflect on and to guide these observations.

Checking for understanding

- Were the students able to develop a clearly labelled and suitable solution to the design challenge?
- Could they explain how the solution met the design criteria and user story requirements?
- Did they compare different designs and justify their choices?
- Were they able to communicate their design process effectively?
- Did they create a clear, logical algorithm including branching and iteration?
- Did their MakeCode program reflect their algorithm?
- Could the students identify areas for improvement in their design solution, algorithm or code?

For more assessment resources we recommend the Assessment resources on the Digital Technologies Hub.

Teacher professional learning opportunities

We would like to thank the Australian Government Department of Education for funding our Lending Library and associated resource development.

Adelaide University

STEM PL csermoocs.adelaide.edu.au

We run a range of STEM programs for Australian teachers, including our online CSER MOOC courses, free professional learning events, and our National Lending Library.

Our free, self-paced online courses available from CSER and Maths in Schools in the following areas:

- Decoding Digital Technologies
- Digital Technologies + X
- Cyber Security and Awareness
- Teaching AI in the classroom
- Maths in Schools: Foundation - Year 2, Year 3 - 6 and Year 7 - 10

www.csermoocs.adelaide.edu.au

Design thinking: Project plan summary

(you might need a page for each stage, but a final summary could be placed here)

Stage	What it means	My Action	The Big Question
Understand	**The detective phase**	Describe the problem you are trying to solve and who you are trying to help?	Who is this for, and what do they really need?
User story: **Design Criteria:**			
Ideate	**The brainstorm phase**	Sketch or describe many ideas and select the best one.	How many different ways can we solve this?
Prototype	**The maker phase**	Build a quick model using cardboard, tape, or apps to show your idea.	How does my solution solve the problem?
Reflect	**The evaluate phase**	Get feedback (Plus, Minus, Interesting) and think of ways to improve.	How can I improve my solution?

Australian Curriculum

This activity is suitable for students between Year 3 and Year 4 **Digital Technologies:**

Students in Years 3 and 4 learn to:

- define problems with given design criteria and by co-creating user stories (AC9TDI4P01)
- generate, communicate and compare designs (AC9TDI4P03)
- discuss how existing and student solutions satisfy the design criteria and user stories (AC9TDI4P05)
- follow and describe algorithms involving sequencing, comparison operators (branching) and iteration (AC9TDI4P02)
- implement simple algorithms as visual programs involving control structures and input (AC9TDI4P04).

By the end of Year 4 students create simple digital solutions and use provided design criteria to check if solutions meet user needs. They follow and describe simple algorithms involving branching and iteration and implement them as visual programs.

Students in Years 5 and 6 learn to:

- define problems with given or co-developed design criteria and by creating user stories (AC9TDI6P01)
- generate, modify, communicate and evaluate designs (AC9TDI6P04)
- evaluate existing and student solutions against the design criteria and user stories and their broader community impact (AC9TDI6P06)
- design algorithms involving multiple alternatives (branching) and iteration (AC9TDI6P02)
- implement algorithms as visual programs involving control structures, variables and input (AC9TDI6P05).

By the end of Year 6 students develop and modify digital solutions and define problems and evaluate solutions using user stories and design criteria. They design algorithms involving complex branching and iteration and implement them as visual programs including variables.

This work is licenced under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. CC BY-NC-SA
Computer Science Education Research (CSER) Group, Adelaide University