(Hello World)

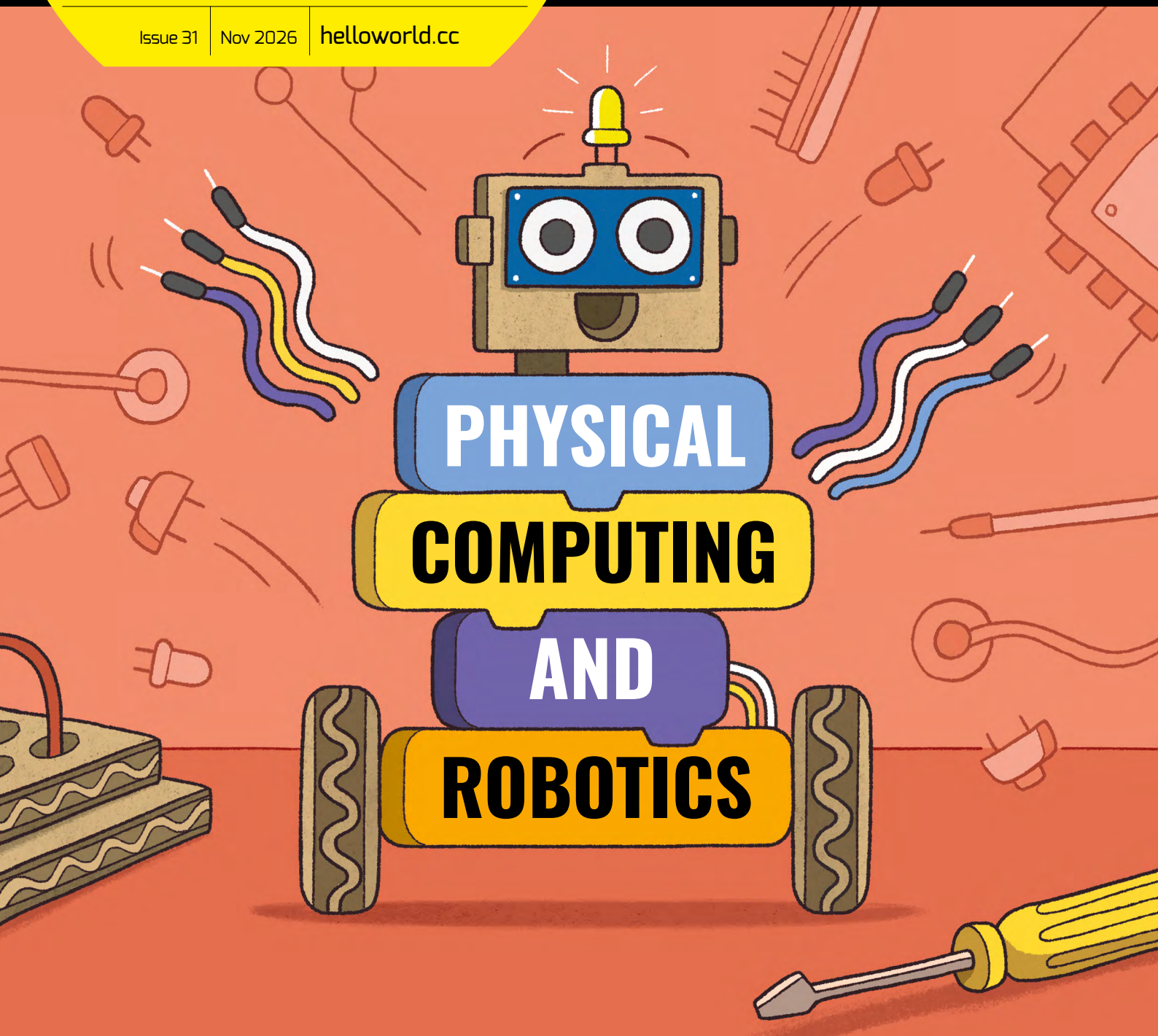
THE MAGAZINE FOR EDUCATORS TEACHING COMPUTING AND AI

WHY PHYSICAL COMPUTING MATTERS
Making abstract concepts tangible in the age of AI

PHYSICAL COMPUTING IN PRACTICE
Practical advice for overcoming barriers

GROUNDED COGNITION
How learning takes place in physical computing

Issue 31 | Nov 2026 | helloworld.cc



PLUS **PHYSICAL COMPUTING TIPS** · CYBERDECKS · **AI PRESSURE COOKER** · URBAN ROBOTICS · **MICRO:BIT** CODE CLUB · **DISABILITY IN THE MAKERSPACE** · PICRAT FRAMEWORK · **BEBRAS** · DESIGN THINKING **EVOLVING COMPUTING** · EPICS STUDY · **PLAYFUL COMPUTING CONFERENCE** · ASTRO PI · **CODE EDITOR**

Raspberry Pi Foundation



FREE

Computing and AI courses for professional development

Build skills and confidence in Python, Scratch, AI and machine learning, cybersecurity, computing pedagogy, and more — whether you're just starting out or growing your expertise.

Created by educators, for educators, and grounded in research.

Learn, connect, and celebrate progress on the Training Hub

Earn your Certificate

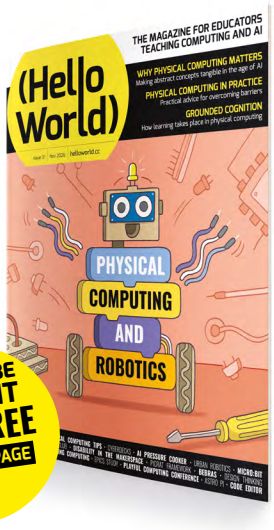
Visit: rpf.io/trainonline-hw

HELLO, WORLD!

In the age of AI, does physical computing have a place in computer science? As we scramble to stay informed and get to grips with AI's place in education, physical computing may feel like it's not worth the effort — but the educators in this issue all attest to its joy and educational value.

Educators are bringing physical computing and robotics to their communities in all kinds of settings. Eugene Pope tells us about his journey from starting a robotics programme at his underfunded urban school, to supporting his students to a first place win in a robotics competition (pages 24–25). Amanda Bastoni shares five ways to bring physical computing integrated with AI to rural learners (pages 38–41). And as the BBC micro:bit celebrates its tenth birthday, Chris Lovell advocates for micro:bits in the secondary classroom as a way to really make computing concepts stick.

Many teachers have shared that building their own confidence was key to bringing physical computing to their classrooms. James Robinson writes about how to start small and how to overcome common barriers to starting physical computing (pages 20–23). And if you want to know whether learning really happens in physical computing, have a read of Katharine Childs' insights article (pages 14–15).



SUBSCRIBE IN PRINT FOR FREE TURN TO PAGE 83

This issue brings together a wealth of shared knowledge, and your feedback helps shape the magazine and podcast. So tell us more about what you'd like to read or see in Hello World by completing our survey. It will be landing in your inbox soon.



Meg Wang
Editor

FEATURED THIS ISSUE



MONICA MCGILL

Monica advocates for resisting the AI pressure cooker and slowing down to consider what your students really need. Read more on pages 50–51.



FEATHER LAROCHE

On pages 54–55, Feather shares a computer science curriculum based in Native history to inspire Native youth to see themselves in computer science.



ROZA SHAIMURAT

Roza's students asked, 'What happens if a child becomes separated from their family without mobile service?' Find out what her students made on pages 70–71.

(HW)

Hello World is the official magazine of the Raspberry Pi Foundation

Raspberry Pi Foundation

EDITORIAL

Editor
Meg Wang

Contributing Editor
Dominick Sanders

Subeditors
Louise Richmond, Gemma Coleman

Subscriptions
Dan Ladbrook

Social Media
Sean Sayers, Teresa Girao

DESIGN

criticalmedia.co.uk

Designer
Dougal Matthews

Photography
The Raspberry Pi Foundation, Adobe Stock

Cover
© PAPERFACE

CONTRIBUTORS

James Abela, Alyaa Abdulhussein, Greg Annandale, Amanda Bastoni, Mark Calleja, Katharine Childs, Sarah Ciras, Michael Conterio, Andrew Csizmadia, Sethi De Clercq, Tonya Coats, Nicki Cooper, Jessie Durk, Helen Gardner, Tracy Gardner, Kala Grice-Dobbins, Ben Hall, Graham Hastings, Rebecca Haynes, Matt Hewlett, Joanne Hodge, Trakeisha Howard, Laura James, Geetanjali Khatri, Emma King, Fiona Kuijpers, Marisa Laks, Feather LaRoche, Chris Lovell, Saúl Maldonado, Mamta Manaktala, Monica McGill, Aditya Mittal, Catherine Moore, Andy Perryer, Carrie Anne Philbin, Eugene Pope, James Robinson, Olivia Rodrigues, Becca Rose, Rajib Sarkar, Marc Scott, Sue Sentance, Roza Shaimurat, Gareth Shaw, Omar Shepherd

Contributing Partner

bcs Academy of Computing

CAS

This magazine is printed on paper sourced from sustainable forests and the printer operates an environmental management system which has been assessed as conforming to ISO 14001.

Hello World is published by the Raspberry Pi Foundation, 37 Hills Road, Cambridge, CB2 1NT. The publisher, editor, and contributors accept no responsibility in respect of any omissions or errors relating to skills, products, or services referred to in the magazine.
Except where otherwise noted, content in this magazine is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 Unported (CC BY-NC-SA 4.0).

recycle

(HW) CONTENTS



NEWS, FEATURES, AND OPINION

6 NEWS: CURRICULUM
A skills-based curriculum for computer science

8 NEWS
Offline support for our free browser-based Code Editor

10 NEWS
Experience AI across Latin America

11 NEWS: THE WORD LIST
Concepts you'll read about in this issue

12 INCLUSION COLUMN
What primary-age pupils know about and do with AI

14 GROUNDED COGNITION
How learning takes place in physical computing

16 EPICS PROJECT
How physical computing shapes digital confidence in UK schools

18 WHY PHYSICAL COMPUTING MATTERS
Making abstract concepts tangible

20 PHYSICAL COMPUTING IN PRACTICE
Practical tips for overcoming common barriers

24 ROBOTICS FOR URBAN STEM
How physical computing engages students in underserved communities

26 MAKE CODING TANGIBLE
Unplugged resources to support a new coding and robotics curriculum in South Africa

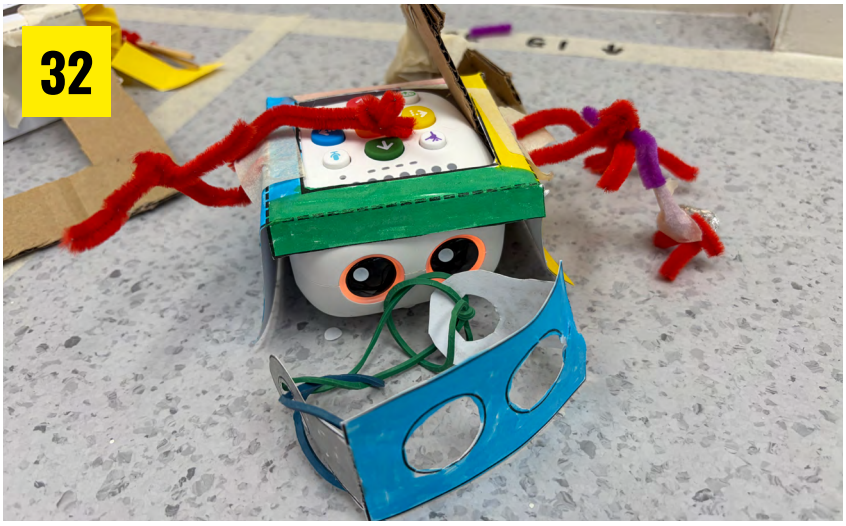
28 SHIFTING TO HANDS-ON PHYSICAL COMPUTING
Empowering students to become more independent

30 DISABILITY IN THE MAKERSPACE
Creating inclusive, accessible maker environments for all learners

32 TELLING TALES WITH ROBOTS
Combining storytelling, robotics, and cross-cultural collaboration in Southeast Asia

36 PHYSICAL COMPUTING AND MACHINE LEARNING
Integrating AI concepts into physical computing projects

38 BEYOND THE COMPUTING LAB
Five ways to bring physical computing and AI to rural learners







42 CHIBITRONICS
How a comic book led an educator to physical computing

44 MICRO:BIT FOR THE GCSE CLASSROOM
How physical computing can make learning stick for upper-secondary students

46 THINKING GLOBALLY, ACTING LOCALLY
A scalable, interdisciplinary framework for environmental data science

48 BREAKING DOWN BARRIERS
Professional development for physical computing confidence

50 THE EDUCATOR AI PRESSURE COOKER
Surviving the AI adoption mania

52 BEFORE THE FIRST LINE OF CODE
Why design thinking should come right at the start

54 NATIVE AMERICAN HISTORY AS CS
Connecting history, culture, and community with tech skills for the next seven generations

56 CREATIVITY AFTER HOURS
How Boys & Girls Clubs in Florida are creating innovators and tinkerers

LEARNING

RESOURCES & LESSON PLANS

58 TEACHING LOOPS ON THE PICRAT GRID
Using the PICRAT framework on one loops lesson

60 ASTRO PI MISSION SPACE LAB
A worked code example for teams of students to use with the Mission Space Lab Creator Guide

64 ALL HANDS ON DECK
How and why to join in the cyberdeck trend

68 COMPUTING EDUCATION THROUGH PLAY
Insights from educators at the Digital Schoolhouse Playful Computing Conference

70 NO SIGNAL? NO PROBLEM
How students designed an off-grid safety system

72 COOLEST PROJECTS
Young makers finding their community

CONVERSATION

63 BEBRAS
A fun computational thinking challenge

74 CAREER CONVERSATION
A young innovator making neuroscience accessible

76 INSIDER'S GUIDE
Physical computing survival tips

78 EVOLVING COMPUTING
The future of studying computing

80 CODE CLUB
Turning learners into the mentors of the future in Iraq

82 EMBRACING AI IN EDUCATION
A teacher's perspective



© CarlosBarquero/stock.adobe.com

The Certificate in Applied Computing gives young people the opportunity to develop, demonstrate, and apply their digital skills

CERTIFICATE IN APPLIED COMPUTING

Building students' essential digital skills

Ben Hall

The Raspberry Pi Foundation and Greater Manchester Combined Authority (GMCA) have developed the Certificate in Applied Computing, with stage three having launched in November 2026. This is a new initiative aimed at improving outcomes for young people in Greater Manchester by giving them the opportunity to improve their digital skills. It is delivered as part of the Greater Manchester Baccalaureate (MBacc), an educational framework introduced by GMCA to provide a prestigious technical and vocational route into further education or employment.

The certificate is a multi-stage, student-focused programme that prepares young people aged 14 to 19 for further education, training, or employment. It is designed for learners who do not have a traditional computing background such as GCSE Computer Science. It focuses on accessibility and real-world relevance and has a staged, project-based approach. Through self-paced modules, learners build both foundational and advanced digital skills while creating a portfolio they can showcase to future employers.

The digital skills gap

The certificate emerged from a shared understanding by the Raspberry Pi Foundation and GMCA that students were leaving education lacking core digital skills. This picture is mirrored at a national level. The House of Commons Science and Technology Committee's 2016 'Digital Skills Crisis' report estimated that a lack of digital skills was costing the UK economy around £63 billion a year, and that 7 million working adults in the UK are without basic digital skills (**helloworld.cc/digital-skills-crisis**). The AQA examining board, in its 'Digitally native or digitally naïve?' report, found that only 46 percent of students are confident navigating essential workplace applications. In the same report, AQA found that the problem is further compounded by teachers reporting a lack of curriculum time and resources to address it (**helloworld.cc/digitally-native**).

In the planning phase, the Raspberry Pi Foundation spoke to educators, employers, and students. These discussions showed that there was a shared understanding that any programme should be free and accessible and give young people the opportunity to develop, demonstrate, and apply their digital skills.

Root causes and leadership

The lack of digital skills in young people has been caused by a number of factors. Rik Cross, director of professional development at the Raspberry Pi Foundation, explains: "Many of the issues date back to the curriculum reform in England introduced in 2013. That reform marked a shift from digital literacy towards a more computer science based curriculum focusing on skills such as programming." The intention of the change was to boost the number of students studying computer science while incorporating digital skills by integrating them across the wider curriculum. Because of a lack of resources and suitably trained educators, digital skills were deprioritised by many schools.

Andy Burnham, now UK prime minister, introduced the certificate when he was the Mayor of Greater Manchester: "Digital skills are a key part of the Greater Manchester Baccalaureate or MBacc. The MBacc is intended to help our economy get the skills it needs for the future and that's why I'm truly delighted to introduce the Raspberry Pi Foundation's Certificate in Applied Computing, a key part of the MBacc. This isn't just another certificate; it's been co-designed with young people right here in Greater Manchester and shaped by some of our biggest local employers. They've helped make sure the content is useful, up to date, and linked to real jobs."

Alignment with national education pathways

The move towards a more skills-based curriculum reflects a wider national trend. T levels are two-year technical qualifications in England, equivalent in size to three A levels, which combine classroom learning with an industry placement of at least 45 days. V levels (vocational qualifications), being introduced from September 2027, will each be the size of one A level and will let students explore a sector before deciding where to specialise.

The certificate complements both of these by developing digital skills that learners can apply across education and employment. It can provide a foundation for the more specialised technical learning of T levels, while its broad, applied approach aligns with the vocational focus of V levels. Its practical projects, employer input, and portfolio help learners on either pathway demonstrate problem-solving, independence, and the application of knowledge.

The certificate is free for all schools and is available now. Since launching in March 2025, over 2400 young people have started working towards the certificate. Educators and youth organisations who want to offer it can find out more at **helloworld.cc/rpf-applied-cert**. As digital skills become essential to almost every job, the certificate gives young people a way to build those skills, prove them, and carry them forwards into whatever comes next. (HW)

> "THE CERTIFICATE IN APPLIED COMPUTING IS DESIGNED FOR LEARNERS WHO DO NOT HAVE A TRADITIONAL COMPUTING BACKGROUND



The move towards a more skills-based curriculum reflects a wider national trend

© Syda Productions/stock.adobe.com



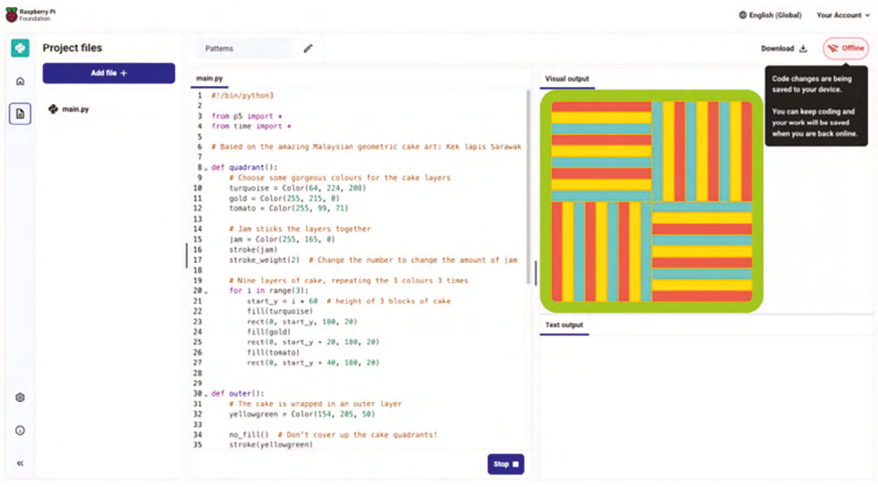
Our commitment is to design for diverse needs, and this means not assuming constant internet connectivity

© WavebreakMediaMicro/stock.adobe.com

KEEP CODING WHEN YOUR INTERNET DROPS

Offline support in the Code Editor

Greg Annandale



Better offline support makes the Code Editor more useful for learning and teaching

Our free, browser-based Code Editor now keeps working even when your internet connection doesn't. If you lose your connection in the middle of a lesson or project, you can carry on writing and running your code without disruption, even if the page is reloaded.

Why we built this

An unstable internet connection is a minor annoyance when you're reading a web page, but it's a much bigger problem when you're 20 minutes into creating a program.

Until now, losing your connection while using the Code Editor could mean losing your work: if you refreshed the page at the

wrong moment, you might have seen a blank screen or a browser error instead of your program.

We wanted to understand better what learners and educators around the world need from the Code Editor, so we carried out research in countries including India, Kenya, and South Africa. One topic we explored was the availability and reliability of internet connections in metropolitan and rural areas. While it came as no surprise that internet access is often slow, unreliable, or expensive in rural areas, we learnt that issues related to electricity supply, such as load-shedding and brownouts, are also prevalent in metropolitan areas. All this shapes what is realistically possible in a programming activity using an online editor.

And connectivity issues can occur in any school computing lab where 30 learners are all accessing the same Wi-Fi at once, or for a Code Club running off a mobile hotspot in a community centre, or for someone finishing their homework on patchy mobile data.

> AN UNSTABLE INTERNET CONNECTION IS AN ANNOYANCE WHEN READING, BUT A MUCH BIGGER PROBLEM WHEN PROGRAMMING

That's why it became a priority for us to improve the Code Editor's offline support by making it resilient to disconnection.

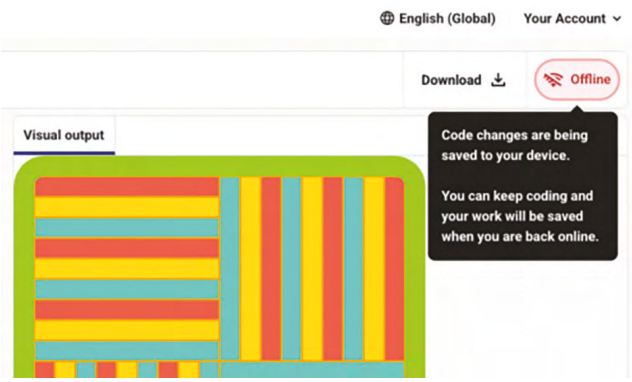
As we describe in our recently shared draft principles for safe and responsible education technology (**helloworld.cc/design-principles**), one of our commitments is to design for diverse needs, abilities, and contexts — including not assuming constant internet connectivity. This Code Editor update is a small, practical step towards fulfilling that commitment.

What this means for learners and teachers

The Code Editor was built with resilience in mind from the start, for example running code in your browser rather than on a server, and saving your project locally on your device as you go.

Our recent additions ensure that once you have opened the Code Editor while online, your browser holds on to even more of the parts you need to keep coding without a connection. Now, if you go offline midway through using the Editor:

- You'll see a clear message in the Editor flagging that you're offline
- A full page refresh won't result in a connection error or blank screen
- If you are logged in, everything you do in the Editor while you are offline is saved to your account automatically once you are back online
- You can still download your code using the download button, if you know you are going to be offline for a while and need to save your program file on your computer or a storage device



A message shows in the Editor when the internet connection is down

How it works, in brief

This update is built on a service worker, a script that your internet browser runs in the background (**helloworld.cc/service-worker**). When you first load the Code Editor, the service worker stores the files the Editor needs in your browser's cache. If the network connection then breaks, the service worker serves those stored files instead of trying and failing to fetch them. When you come back online, it automatically refreshes the cache so you're not left running old files.

What is not possible at the moment

The Code Editor isn't a full offline app that you install. It's a browser-based app that we have now upgraded to cope better with losing its internet connection, which means:

- You need to have loaded the Code Editor at least once while online, in the browser and on the device you are going to use
- When you go offline, you can't open projects you have not already loaded
- If you clear your browser data while offline, any changes you have made to projects will disappear
- You can't log into or out of your account while offline, and saving to your account pauses until you're back online
- Any feature that needs the network, such as sharing projects with students in Code Classroom (**helloworld.cc/code-classroom**), won't be available until you reconnect

We will keep working to extend what parts of the Editor work while you are offline.

Try it and tell us how you get on

The Code Editor (**helloworld.cc/code-editor**) works in your browser with no set-up and will always be free for educators and learners. We hope this update makes it more useful for people in lots of different settings.

This update is one of those features some people will never notice, which is rather the point: it should feel like nothing went wrong. If you try out the updated Code Editor with an unstable connection, we would love you to let us know how well it works for you (**helloworld.cc/code-editor-feedback**).

We would like to thank Cisco for the generous funding that made this work possible. (HW)



EXPERIENCE AI

Our focus now turns to bringing Experience AI into classrooms across the region

© Daniel/stock.adobe.com

Reaching classrooms across Latin America

Saúl Maldonado

Earlier this year, we announced that we would be expanding Experience AI (experience-ai.org), our AI literacy programme, across Latin America to bring AI education to nine more countries through support from Google.org. Working with local partner organisations, we offer Experience AI training and resources that teachers can use to enable young people to gain a foundational understanding of AI technologies, their social and ethical implications, and the role that AI can play in their lives.

Over the past few months, Experience AI launch events have already taken place in Brazil, Chile, Argentina, and Mexico. At every event, we met educators, policymakers, universities, non-profit organisations, and technology leaders and explored how AI literacy can become part of classrooms across their country. Experience AI brought us all together around a shared goal: giving teachers the confidence and resources they need to help young people understand and engage with AI critically and responsibly.

Growing the Experience AI partner network in Latin America

Across Latin America, the local organisations we are partnering with to implement Experience AI bring deep expertise in teacher professional development and education. Partners include Fundación Sadosky in Argentina, Fundación Kodea in Chile, Sincroniza Educação in Brazil, and Educación para Compartir in Mexico.

Each partner is implementing the programme in a way that reflects its local education system and context. We work with them individually and also facilitate their collaboration around shared challenges.

A delivery model designed for lasting impact across Argentina

In Argentina, Experience AI was officially launched in June during a Google event in Buenos Aires.

Our Argentinian partner Fundación Sadosky is taking a distinctive approach to Experience AI teacher professional development by working through a network of public universities across the country, rather than relying on a central training team. Through this approach, Experience AI training will reach educators in diverse regions across the country as Fundación Sadosky builds on local expertise and existing educational networks.

Growing local capacity and developing regional trainers means Experience AI will make a lasting impact for teachers and learners, enabling high-quality AI education long after the programme's first rollout.

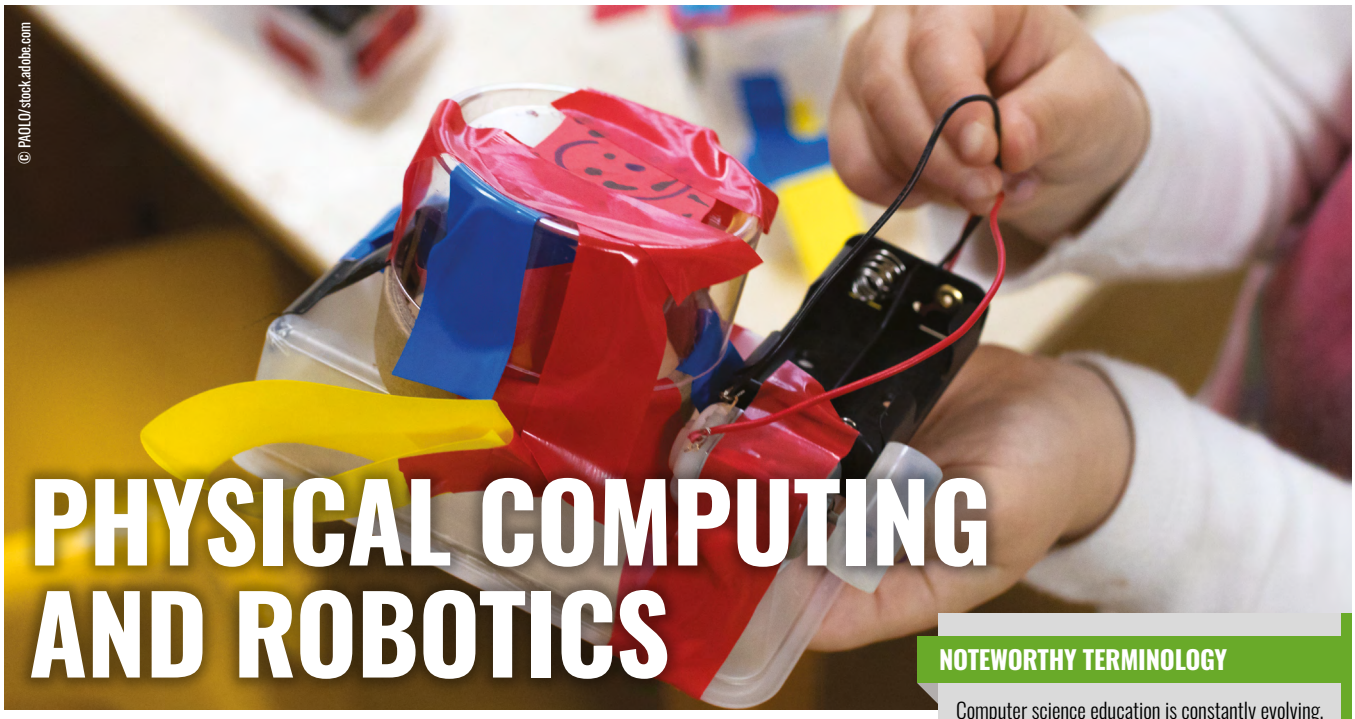
The first cohort of participating universities is now working with teachers across several Argentinian provinces before the programme expands nationwide. Experience AI aims to support around 1700 secondary school teachers and reach an estimated 153,000 students.

Turning vision into action

It's been very rewarding coming together with local partners, **Google.org**, universities, and education communities across Latin America around our shared vision of high-quality AI education for all young people.

After the launch events, the focus now turns to practical action: supporting teachers and bringing Experience AI into classrooms across the region.

As the programme grows, we also look forward to working with new partners across Latin America to bring AI literacy to even more educators and young people. By 2028, we and our Latin American partners aim to support 24,000 more educators and reach more than 1.25 million young people through Experience AI. (HW)


© PAOLO/stock.adobe.com

PHYSICAL COMPUTING AND ROBOTICS

Spotlight on terms that appear in this issue of Hello World

Accelerometer: an onboard motion sensor that measures acceleration forces along multiple axes, allowing physical computing devices to detect gestures like shaking, tilting, or jumping (pages 42–43).

Breadboard: a reusable, solderless construction board used for prototyping electronic circuits, allowing components like microcontrollers, LEDs, sensors, and jumper wires to be temporarily connected (pages 48–49).

Capacitive touch input: a touch-sensitive input mechanism that detects electrical charges from human touch (such as pressing conductive gold logos or foil pads) to trigger events without mechanical buttons (pages 44–45).

Cyberdeck: a custom-built, highly portable, often retro or futuristic DIY mobile computer set-up designed by enthusiasm-driven makers in physical computing communities (pages 64–67).

ESP32: a low-cost, high-performance microcontroller board featuring integrated Wi-Fi and Bluetooth capabilities. Named after its manufacturer, Espressif Systems, and its 32-bit processor architecture, the ESP32 is used in embedded hardware projects, IoT prototypes, and wireless communication systems (pages 70–71).

Haptic feedback: physical, tactile feedback (such as feeling a device shake) that provides concrete sensorimotor confirmation of program execution in physical computing (pages 14–15).

LoRa (long range): a low-power, long-range wireless radio communication protocol that allows devices to transmit telemetry and messages across distances off the grid without requiring cellular networks or Wi-Fi (pages 70–71).

Microcontroller: a compact, integrated circuit board (such as a BBC micro:bit, Arduino, Chibi Chip, or Raspberry Pi Pico) designed to govern specific operations in an embedded system by executing user code, reading sensor inputs, and controlling physical outputs (pages 48–49).

PICRAT framework: a matrix tool that allows teachers to reflect on how effectively they are using technology in their classroom (pages 58–59).

NOTEWORTHY TERMINOLOGY

Computer science education is constantly evolving, and so is the language we use to describe this vibrant discipline. Here are 13 terms you'll come across in this issue, contextualised through the lenses of our contributing writers. Do you agree with these definitions? How are these words appearing in your work? Get in touch!

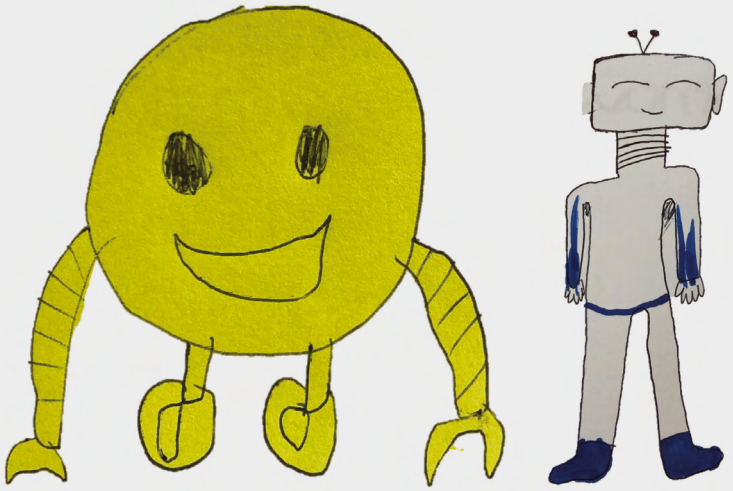
Rubber duck method: a debugging and problem-solving strategy used in makerspaces where students pretend they are speaking their code or design ideas out loud to a rubber duck. This can help students process and clarify their thoughts (pages 30–31).

Servo motor: a rotary actuator that provides precise angular control over physical components (such as opening gates or moving robotic joints) based on signals received from a microcontroller (pages 28–29).

Single-board computer (SBC): a complete, fully functional computer built on a single circuit board (such as Raspberry Pi) containing microprocessors, memory, and input/output interfaces (pages 64–67).

Sonar sensor: an ultrasonic sensor that emits sound waves to measure distance and detect physical obstacles or human presence in an environment (pages 28–29).

COLUMN



IF WE HAVE AI, WHY DO WE NEED TO GO TO SCHOOL?

Teaching primary learners about AI

"If we have AI, why do we need to go to school?" This question was asked by a Year 2 pupil (ages 6–7) as part of our year-long collaborative research project to investigate what children know about AI and how they are using it. It's a big question. Putting aside the wider conversation around the purpose of education, it demonstrates the pressing need for AI literacy education for our children.

Artificial intelligence is becoming embedded within the everyday lives of children, through search engines, smart speakers, social media platforms, and generative AI chatbots. Around the world, some nations are addressing this reality proactively: in the United Arab Emirates, AI is now a mandatory part of the curriculum for 4- to 18-year-olds; India has introduced AI as a subject from Class 6 (ages 10+); Czechia has included AI as part of the mandatory digital competence in the national curriculum since 2023. Others are moving more cautiously. In England, the recent Curriculum and Assessment Review recommended that all pupils should learn how to use artificial intelligence effectively from 2028. This feels like a long way in the future, so we decided to explore what children currently know and are doing with AI, and what an AI literacy curriculum might look like in practice at primary level.

The Teaching about AI in Primary Schools project was established by Learn Sheffield with eight primary schools during the 2025–26 academic year. Project leads taught purpose-designed lessons and contributed evidence through surveys, teacher interviews, pupil voice activities, and case studies.



Catherine Moore leads the Computing & Digital Innovation Centre in Sheffield (**helloworld.cc/learn-sheffield**), supporting schools with computing and the wider implementation of educational technology, including AI. She has over 20 years' experience of working with students with special educational needs and disabilities and their teachers, and is passionate about making computing accessible and inclusive for all learners.

AI literacy lessons

The materials comprised two lessons for each year group from Year 2 to Year 6 (ages 6–11), to be taught as part of the existing computing curriculum. The lessons cover the technological and human dimensions of AI literacy, as defined by the Council of Europe (**helloworld.cc/learners-ai-guide**) — that is, how AI actually works and its impact on humans, including risks and real-world uses. The practical dimension introduces difficulties with this age group, as the majority of generative AI tools are not designed for younger children and can present a safeguarding risk. We therefore minimised this element. The lessons include unplugged activities, discussion, and interactive activities using

age-appropriate online tools such as Google's 'Quick, draw!' and 'AI for Oceans' from Code.org. The materials include full lesson plans, slides, and teacher videos to support non-specialists in their delivery.

Findings from the research

As part of the project, I conducted pupil interviews in two very different schools, with 34 children from Year 2 to Year 6 (ages 6–11), both before and after the lessons were taught. Here's what emerged.

1. Before the lessons, learners had little understanding of what AI is and how it works.
When asked what AI is, 58 percent of pupils talked about or drew robots, and could not explain how it worked. After the lessons, this number dropped to 35 percent, and a number of pupils demonstrated more accurate mental models, talking about data centres, humans training AI, and pattern analysis.

2. Children's attitudes towards AI were mixed, and they could name a number of risks.
Before the lessons, pupils talked about the technology as being helpful, but they also had a surprisingly good awareness of the risks. Several pupils talked about the environmental impact, while others discussed how it might make you lazy or could be used for 'bad things'. There were also a number of mentions that AI might take over the world. For example, one Year 5 pupil said before the lessons, "I'm starting to get worried because it's changing lives in a bad way so it can corrupt young people to rely on AI for everything." After the lessons, the existential threat was less common in the responses, and children were more likely to talk about the danger of AI getting things wrong or being utilised to make fake content.

3. Use of AI chatbots starts early and is growing.
Over two-thirds of pupils had used an AI chatbot, predominantly ChatGPT, at the time of the second interviews in June 2026. This included half of the Year 2 pupils (ages 6–7). Many of these learners are using the technology without adult supervision, which is concerning due to the safeguarding risks. After the lessons, a number of pupils modified their behaviour, choosing not to use the AI summary in Google and asking parents before using a chatbot.

4. Teacher confidence and attitudes are key to effective AI literacy education.
Although around three-quarters of teachers are using AI to support their role, there is still a gap in their technical knowledge of how the technology works. For example, half of the teachers surveyed in this project had never heard of the term 'large language model'. After delivery of the lessons, 87 percent of the teachers felt more confident talking to pupils about AI. The attitude of teachers towards the technology was also a factor in their readiness to teach AI literacy. Many teachers involved in the project were concerned about the effect of AI on their pupils and unsure about the appropriateness of teaching it to young children. Targeted professional development helped them to understand the importance of teaching AI literacy to pupils, and they were reassured that the lesson materials were age-appropriate and addressed any concerns.

One AI project lead said, "The lessons prompted rich professional conversations, helping staff clarify misconceptions, share strategies, and build a more cohesive understanding of how to teach AI safely and effectively."

Recommendations

So, what do we recommend as a result of this research?

- Primary-age pupils are already using the technology without a full understanding of how it works or the risks involved, so we need to be teaching them AI literacy from an early age.
- Schools and settings should have an AI policy in place so teachers and pupils are clear what they can and cannot do with AI.
- Teachers require sufficient training and comprehensive resources in order to be able to deliver AI literacy effectively.
- AI shouldn't just be taught in computing — consider the impact of AI in other subject areas and teach about safe and effective use where you would cover online safety.
- Consider the question of equity — do all young people have the same access to high-quality AI literacy education? Is bias addressed as part of the lessons?
- Parental engagement is also key to keeping children safe around AI, so consider how to involve them in the conversation.

Finally, 96 percent of pupils enjoyed the lessons and wanted to learn more. Returning to that Year 2 pupil's question, "If we have AI, why do we need to go to school?", the answer lies in what makes education human. AI can provide quick answers, but school is where children learn to question, collaborate, and belong. By combining AI literacy with the trusted relationships of the primary classroom, we prepare our pupils not just for a digital future, but for a human one.

The final report can be downloaded at **helloworld.cc/ai-primary-schools**, and there are some free AI literacy materials, including an AI glossary and learning outcomes, on our page here: **helloworld.cc/sheffield-ai**. (HW)

#INSIGHTS

HOW LEARNING TAKES PLACE IN PHYSICAL COMPUTING

STORY BY Katharine Childs

In physical computing, students learn computing by making digital artefacts that interact with the world around them. Advocates argue that it's a fun, engaging approach for children. But how does learning actually take place in these activities? I used cognitive science principles to review prior physical computing research studies, because how we think affects how we learn, which affects how we teach.

Beyond the brain as a processor

For decades, cognitive science depicted the mind as an abstract processor that takes in information, manipulates symbols, and stores knowledge in memory. This view suggests that learning is amodal and independent of the body, the senses, and the environment. But a newer idea, grounded cognition, challenges this picture. Through a framework called the Situated Action Cycle (**Figure 1**), this idea suggests that our cognitive processes are grounded in our sensorimotor systems (such as touch, sight, and movement), our emotional states, and the physical and social environment around us. When students understand what a loop is, that understanding is not stored as a pure abstraction. The concept is interwoven with the sights, sounds, movements, and feelings they experienced. For teaching, this suggests that students will build a more robust understanding if concepts are encountered through embodied, sensory, and emotionally meaningful experiences.

Physical computing seems almost purpose-built for learning using a grounded cognition approach. But how does learning take place, and are there ways in which the learning experience could be improved? To explore this question, I reviewed 37 research studies about physical computing activities that used microcontroller devices such as the micro:bit, Crumble, Arduino, or Raspberry Pi Pico. What I found was encouraging and thought-provoking and has direct implications for classroom practice.

Is the activity relevant to the student?

One of the most consistent findings was that physical computing activities tend to be anchored in real-world contexts. Students were not just writing abstract code: they were building cookie theft alarms, collision-detecting cars, health monitors for boxers, e-textile artworks, and reaction games. According to grounded cognition, learning using real scenarios matters cognitively. When a task is connected to a student's goals, values, and sense of identity, it is appraised as self-relevant, and this directly shapes how effectively concepts are encoded and retained. To increase the self-relevance of a task, one study proposed a 'trading zone', in which teachers ensure

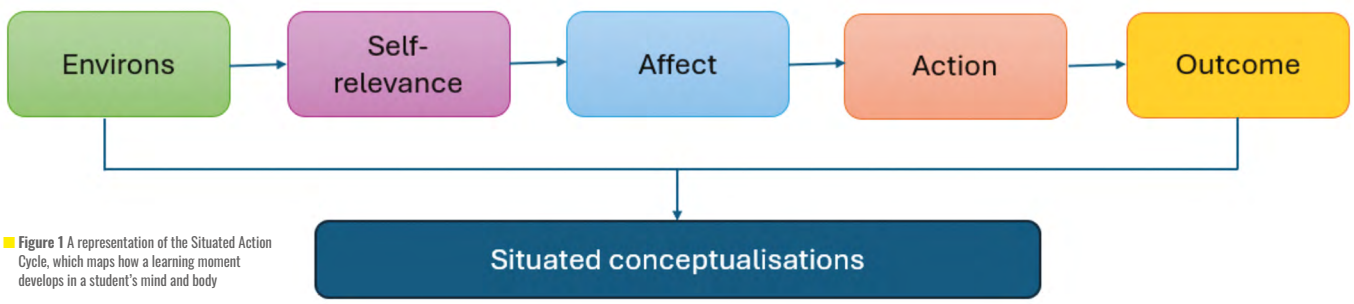
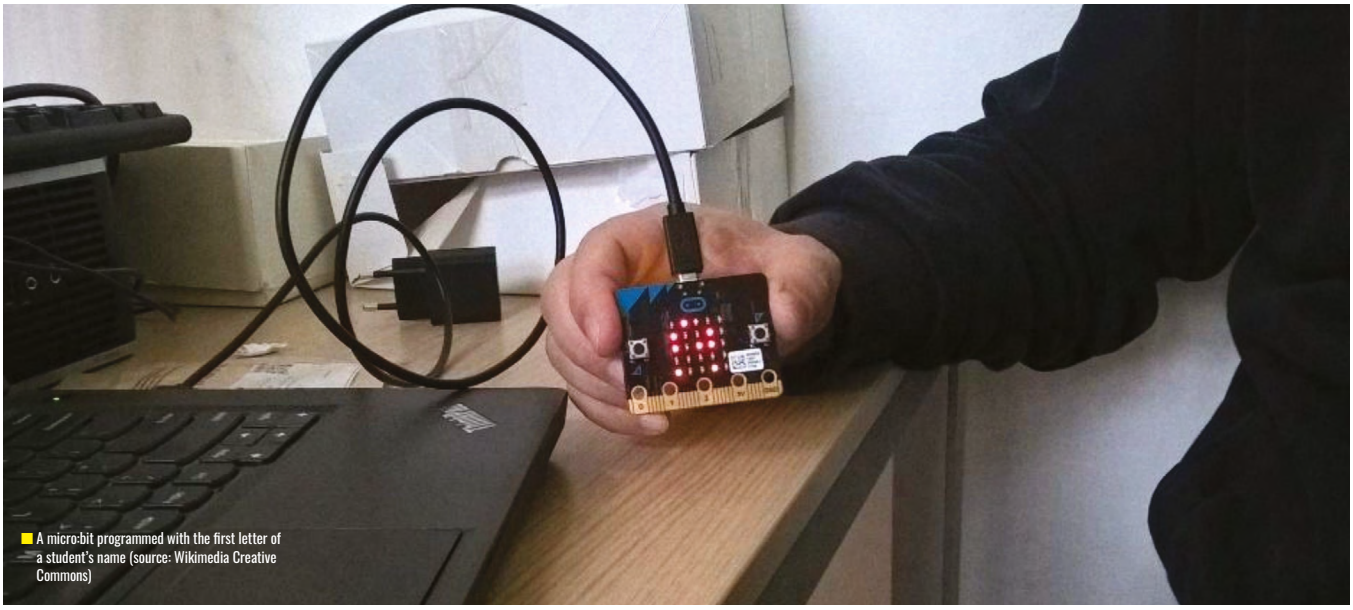


Figure 1 A representation of the Situated Action Cycle, which maps how a learning moment develops in a student's mind and body



A micro:bit programmed with the first letter of a student's name (source: Wikimedia Creative Commons)

that core programming concepts are taught, while students have opportunities to choose contexts that matters to them.

Key takeaway: The more opportunities students are given for choice, such as which project theme to pick or which real-world problem to solve, the better they understand the concepts they are taught.

Emotions drive student motivation

Ask computing teachers why they teach physical computing and 'It's fun' is likely to appear somewhere in the answer. The research confirms this: across the studies I reviewed, many reported positive student emotions, such as excitement, enjoyment, and fun, particularly when students first encountered the devices or reflected on what they had made. However, students also experienced negative emotions. Some felt fear, stress, or overwhelm, particularly before starting an activity or when encountering unfamiliar concepts.

Grounded cognition argues that emotions directly drive motivation and action. Students who feel excited are motivated to take on the cognitive challenge of learning a new programming concept. Students who can recognise frustration rising and respond constructively (for example, by simplifying their design, asking for help, or reading instructions) are better placed to persist through difficulty.

Key takeaway: Create space for students to manage frustration by building in reflection checkpoints and modelling how to respond when something goes wrong.

> "LEARNING THROUGH EMBODIED, SENSORY, EMOTIONALLY MEANINGFUL EXPERIENCES BUILDS A MORE ROBUST UNDERSTANDING

Using touch and sight to anchor concepts

Physical computing is a hands-on activity. But it is worth being precise about what kinds of physical interaction best support learning. My analysis found that students' motor actions in physical computing activities often served a confirmatory purpose: pressing a button, shaking the device, or watching an LED light up to check that the code was working. This haptic and visual feedback provides concrete evidence of whether a program works as intended and grounds the abstract concept in a perceptual experience for the student. However, there were fewer examples of students using physical interaction in an exploratory way at the beginning of an activity. Grounded cognition suggests that concepts are encoded more robustly when they are anchored in prior sensorimotor experiences.

Key takeaway: Support students to use sensorimotor information at the start of an activity to explore physical objects, notice their properties, and connect them to the programming constructs they will be using.

The hardest leap: from physical to digital

Several studies described the persistent difficulty students face in translating a physical phenomenon into a digital implementation. Even when activities were carefully contextualised in real-world scenarios, students struggled to move from their understanding of a concrete object to the code that should represent it. Supporting students in making a deliberate move from the physical to the digital world is important in teaching.

Key takeaway: Sequence activities so that students encounter a concept in its most concrete form first by touching, sketching, or physically demonstrating what they want to happen before moving towards writing code.

Physical computing has always been a rich context for learning. Understanding why it works and where it can work better can help us make it work for every student in the room. (HW)

#INSIGHTS

SMALL DEVICE, EPIC(S) IDEAS

What teachers and pupils told us about physical computing in the UK

STORY BY Carrie Anne Philbin, Jessie Durk, and Sue Sentance

The Exploring Physical Computing in Schools (EPICS) project investigates how teachers and children engage with physical computing in the classroom (helloworld.cc/EPICS). Over the course of this five-year study, we are working closely with a small group of pupils, teachers, and parents across the UK, following their experiences over time to understand how physical computing shapes digital confidence and creativity.

Alongside this longitudinal research, we are also gathering evidence from a much larger group through surveys of pupils aged 7–10 and their teachers. The first survey, carried out in 2024, provides a valuable snapshot of how physical computing is experienced in primary classrooms and complements the in-depth case studies at the heart of the project. Further surveys will be conducted in 2026 and 2027 and at the end of the study, allowing us to track changes and compare them with the experiences of our core participants.

We are now well into the third year of the project. What have we learnt so far? Here are some of the key findings from our first survey.

Pupils are engaged with the micro:bit

The pupils who took part in the survey showed a good understanding of the micro:bit's most visible and interactive features. They recognised that it can display images, respond to button presses, and play sounds. Fewer pupils, however, mentioned what is happening behind the scenes, such as how the device uses sensors, power, or communication. This suggests that children are often focused on what they can see the micro:bit doing, rather than thinking about it as a programmable computer. Pupils were not explicitly asked to program the micro:bit before responding to the survey, but this serves as a reminder that children often take technology at face value.

Pupils also described the micro:bit as easy and enjoyable to use, highlighting its value in engaging learners in physical computing. While the responses were generally short and did not explain pupils' thinking in any detail, they show clearly that pupils had positive first experiences with the device. The findings provide strong evidence that pupils find the micro:bit accessible and motivating, although they do not tell us much about how confident pupils feel in their wider computing abilities.

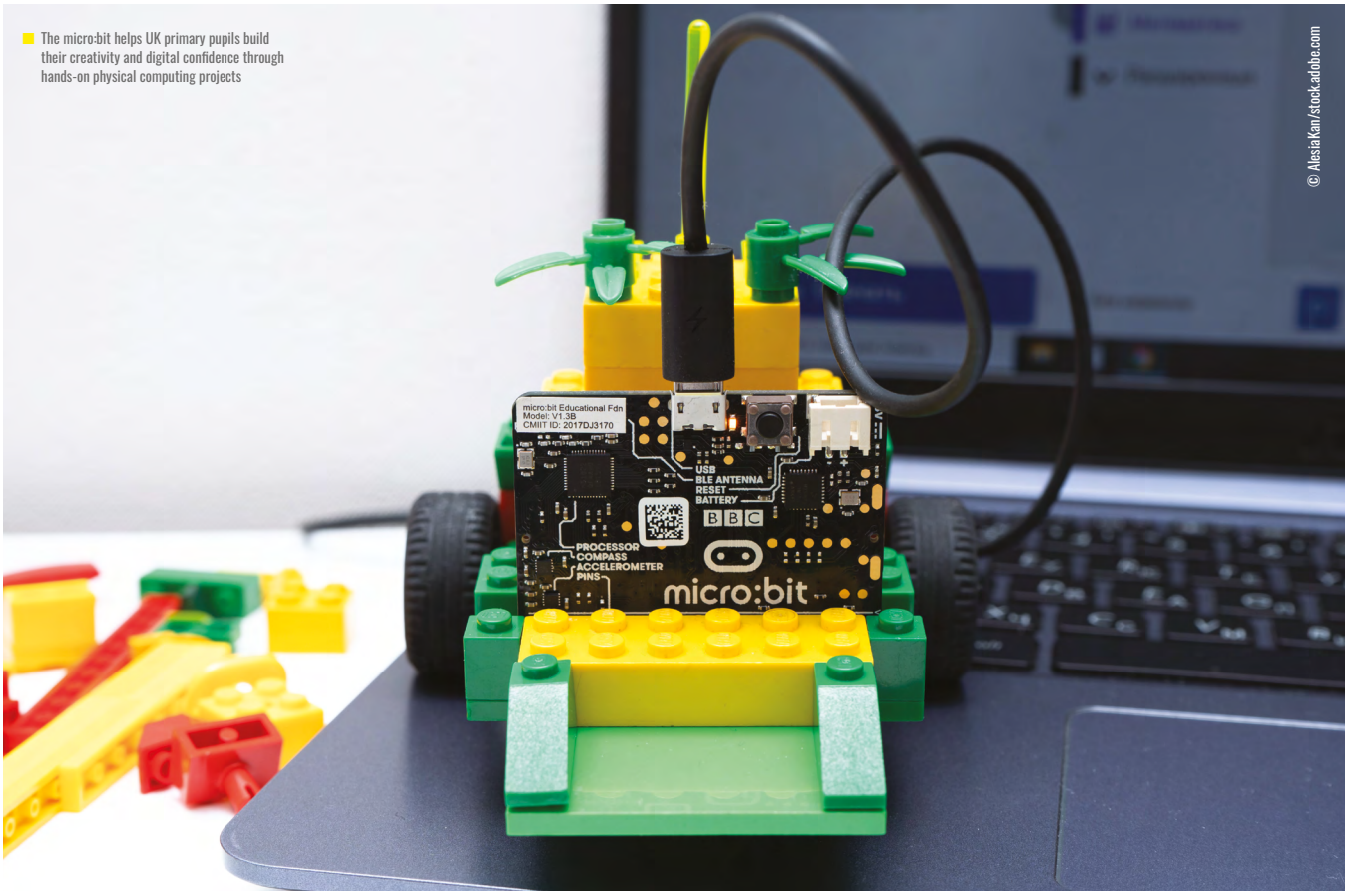
Pupils demonstrate creativity and agency

The findings suggest that physical computing helps spark pupils' creativity. Almost every pupil could think of at least one new way of using the micro:bit, showing that physical computing encourages imaginative thinking. However, many of their ideas were inspired by familiar classroom experiences, and they were often brief and lacking in detail. Pupils most commonly suggested projects linked to communication, art, music, entertainment, and health, showing that they were beginning to connect technology with everyday life. This also highlights the importance of giving pupils regular opportunities to extend and deepen their creative thinking.

Pupils also showed an early awareness of how technology can benefit other people. Many suggested ideas that could help families, professionals, or other groups, recognising that technology can be used to solve real-world problems. Although these explanations were often quite general, some pupils proposed more original ideas, including projects to support healthcare, well-being, and animal welfare. These responses suggest that, with the right learning experiences,

EPICS

The Exploring Physical Computing in Schools (EPICS) project is a UK-wide, five-year, longitudinal, mixed-methods project investigating the attitudes and beliefs of primary-school pupils and teachers about physical computing and physical computing pedagogy. We are grateful for the generous donations from the Micro:bit Educational Foundation, the BBC, and Nominet, which have enabled us to carry out this research.



The micro:bit helps UK primary pupils build their creativity and digital confidence through hands-on physical computing projects

© AlesiaKan/stock.adobe.com

> PUPILS DESCRIBED THE MICRO:BIT AS EASY AND ENJOYABLE TO USE, HIGHLIGHTING ITS VALUE IN ENGAGING LEARNERS

pupils can begin to develop a deeper understanding of how computing can have a positive impact on society.

Teachers see clear benefits for learning

Teachers were overwhelmingly positive about the educational value of physical computing, particularly the use of the micro:bit. They believed it helped pupils build confidence, develop creativity, deepen their understanding of computing, and become more engaged with technology. Most teachers also believed that pupils were growing in confidence as computing learners and becoming increasingly creative in their work. They were slightly more cautious about pupils' ability to make independent choices about projects or to connect their learning with real-life issues. Teachers described classrooms in which collaboration and inclusive participation were encouraged, although opportunities for pupils to continue physical computing at home, or involve parents, were less common.

Teachers' perceptions about gender

Most teachers felt that boys and girls were equally motivated to take part in physical computing and enjoyed the experience. Overall, they saw the micro:bit as an inclusive tool that engages all pupils, regardless of gender, but some subtle differences did emerge. A small number of teachers associated boys with getting started quickly, working independently, and being more willing to experiment, while girls were more often associated with planning, collaboration, and discussion. Although these views were not widely held, their consistency suggests that underlying gender expectations may still influence how some teachers interpret pupils' approaches to physical computing, even when they believe opportunities and engagement are the same for everyone.

UK teachers get involved

If you are an upper-primary teacher in the UK and have received your class set of micro:bits through the 'BBC micro:bit — the next gen' campaign (**helloworld.cc/microbit-next-gen**), we would love you to get involved in the EPICS project. There are two ways to take part:

- Both you and your pupils fill in questionnaires (we only need responses from Year 6/Primary 7 children)
- You fill in your teacher questionnaire (if you teach any upper-primary year group)

You can learn more by visiting **helloworld.cc/EPICS-survey**. (HW)

WHY PHYSICAL COMPUTING AND ROBOTICS MATTER

Making abstract concepts tangible while developing creativity, confidence, and digital agency

Physical computing has been part of computing education for a long time. From programmable toys in early years to robots and microcontrollers in primary and secondary schools, educators have long recognised the value of connecting code with the physical world. Yet, as conversations about computing increasingly focus on artificial intelligence and software, it is worth asking whether physical computing and robotics still deserve a central place in the curriculum.

The answer is an emphatic yes

If anything, they are more relevant than ever. Today's digital technologies are rarely confined to screens. Software interacts with the physical world through sensors, controllers, networks, and intelligent devices. From wearable health trackers to environmental monitoring systems and smart homes, computing exists all around us.

Helping young people understand this world means helping them understand more than programming. They need opportunities to explore how software interacts with hardware, how data is collected and interpreted, and how digital systems influence people's lives. Physical computing and robotics provide these opportunities. They remind learners that technology is not simply something to consume, but something they can understand, question, create, and improve.

Making computing tangible

Many of the concepts we teach in computing are necessarily abstract. Algorithms, variables, events, inputs, outputs, and control structures can be difficult for learners to grasp when their only feedback is text or graphics on a screen.

Physical computing changes this relationship. When learners write code that makes an LED flash, a robot navigate a maze, or a sensor respond to temperature changes, they see the consequences of their programming decisions, and the connection between software and the physical world becomes visible and meaningful.

It also creates authentic contexts for teaching core computing concepts. Sequencing determines the order of a robot's movements. Selection allows a system to respond differently depending on sensor input. Iteration enables continuous monitoring and control. Variables store data gathered from the environment. Concepts that might otherwise seem abstract become purposeful because learners apply them to solve problems.

Learners also begin to think as system designers, recognising relationships between hardware, software, data, people, and environments. This way of thinking becomes increasingly important as computing becomes more connected.

Physical computing also makes iteration difficult to avoid, and that is a good thing. Projects rarely work perfectly the first time. Sensors need recalibrating, circuits need checking, and programs need debugging. As learners test, diagnose, and improve their ideas, they experience something fundamental to computing: successful solutions develop through experimentation and refinement.

Opening opportunities

Another strength of physical computing is its potential to engage a wide range of learners. While some young people are drawn to programming challenges, others are inspired by music, art, design, engineering, environmental science, or solving problems that matter to them.

Learners might build interactive artworks, create digital musical instruments, design wearable technologies, monitor biodiversity, or develop assistive devices. Although the projects differ, they rely on fundamental computing concepts. The context changes, but the learning remains rooted in computer science.

This flexibility can support inclusive classroom practice. Open-ended projects allow learners to demonstrate understanding in different ways and approach shared challenges from different starting points. They provide opportunities for peer learning, creativity, learner agency, and different levels of challenge.

Such experiences can also challenge narrow perceptions of computing and

WHY TEACH PHYSICAL COMPUTING?

Physical computing and robotics provide authentic contexts for teaching programming, data, systems, and computational thinking. They encourage creativity, collaboration, and resilience while helping learners understand how digital systems interact with the physical world. In doing so, they prepare young people not only to use technology, but to question it, improve it, and ultimately shape it.



> PHYSICAL COMPUTING CREATES AUTHENTIC CONTEXTS FOR TEACHING CORE COMPUTING CONCEPTS THAT OTHERWISE SEEM ABSTRACT

who it is for. Learners come to appreciate that the discipline involves creativity, collaboration, design, and problem-solving as well as programming.

Physical computing also lends itself to learning across subjects. Maths supports movement and measurement, science provides contexts for collecting and analysing data, and design and technology contributes to prototyping and product development. These connections reinforce an important message: computing is not an isolated discipline, but a way of understanding and shaping the world around us.

Looking ahead

The technologies shaping society increasingly combine software, hardware, data, and automation. Artificial intelligence is also being embedded within physical devices that can sense, predict, and respond to their environments. Understanding these technologies requires more than a knowledge of how to write code; it requires an understanding of how digital systems behave in the real world.

Schools need not prepare every learner for a career in robotics or embedded systems. They should, however, help every young person develop the knowledge and confidence to participate in a digital society.

Physical computing provides a powerful context for exploring automation, sensing, and intelligent systems while encouraging learners to consider their wider impact. Questions about safety, privacy, accessibility, sustainability, and ethics emerge naturally. Who benefits from a system? What data does it collect? What happens when a sensor is inaccurate? When should a decision be automated? These questions help learners recognise that technological decisions involve human values and choices.

Physical computing can also support progression across the 4–19 age range. Young children explore sequencing and cause and effect through programmable toys. Primary-aged learners investigate inputs and outputs using sensors and interactive projects. Older students can design more sophisticated robotic and embedded systems incorporating data, networks, and intelligent behaviours. The contexts change, but learners revisit and deepen their understanding of fundamental concepts.

The articles in this issue explore approaches to teaching physical computing and robotics across this age range. Together, they demonstrate that physical computing is more than an engaging classroom activity. It provides a powerful context for connecting computing concepts with the world around us while developing creativity, critical thinking, collaboration, and problem-solving.

As computing continues to evolve, our challenge as educators is not simply to teach learners how technology works today. It is to help them understand the systems shaping tomorrow and give them the knowledge, confidence, and agency to shape those technologies for the benefit of everyone. Physical computing and robotics remind us that computing is ultimately about ideas made real, and that remains a powerful learning experience. (HW)



ANDREW CSIZMADIA

Andrew is the Bebras competition manager at the Raspberry Pi Foundation. A former university lecturer and computing teacher educator, he has extensive teaching experience. His robotics interests include serving as a Student Robotics trustee and European Robotics Week ambassador.



LEGO and a handful of micro:bits is enough to start a lunchtime physical computing club

© Alesiakan/stock.adobe.com

PHYSICAL COMPUTING

How to start small

I am a passionate advocate for physical computing as a context for learning programming. I've spent much of my professional life helping others to build interesting projects from limited tools and materials. I have seen dinosaur-themed carousels, whoopee cushion sensors made from paper plates, motion-sensitive musical instruments, and so many robots made out of junk. It is genuinely inspiring to see what people can build with a little bit of experience, confidence, and joy.

When educators tell me they can't do physical computing, the reason usually falls into one of three areas. Confidence is a significant barrier, and logistics and classroom management are separate but related factors. Finally, cost is a key challenge, but one that can be mitigated.

Confidence comes first

Whenever I have run physical computing workshops with teachers or students, there is usually a literal light bulb moment: when learners discover that they can control a simple light bulb LED with a piece of code they have written, and it works, they are thrilled. Finding ways for you and your students to have quick wins breeds success and confidence.

Confidence comes from experience, practice, failure, and having curiosity and the right mindset. When learners come across the inevitable barriers in physical computing, such as reversed LED polarity or loose connections, they pick up some of the principles of debugging, and this is how they build confidence.

There is no single first step

Educators often ask what their first activity should be, but I don't think there is a single best answer. It depends on multiple factors, including the learning goals, the age of the learners, and the availability of equipment. However, there are a number of ways to start small. If understanding circuits is the focus, the first step might be to create a simple circuit out of cheap materials such as copper tape, tin foil, and LEDs, and this could start in a science lesson. In a computing lesson, you might program some LEDs to blink using low-cost devices

such as a micro:bit or a Raspberry Pi Pico, combined with some buttons or sensors, and embedded in a craft project to tell a story or teach another concept.

Even better, perhaps begin the other way round: find a context, challenge, or story that you and your learners are interested in, and look for an opportunity to embed interactivity or enrich learning. How might LEDs, buttons, or motors bring learning to life? Could your learners build it? Could they program it?

It doesn't have to be big or complicated. If you can get a learner to realise that their code can impact or react to the real world, that is a powerful starting point.

Kit, storage, and the classroom

Physical computing equipment adds a logistical challenge to your classroom and poses a number of questions about how it should be stored, set up, and packed away.

My approach has always been to involve the learners as much as possible. The first time I use new equipment with a group, we invest time setting up the kit, achieving something with it, and then carefully packing it down. Your first session might only include 20 minutes of programming activity, but building familiarity and confidence is a valuable investment. Each time the learners use the equipment, the process gets faster and they become more confident with getting it out, setting it up, and troubleshooting any issues.



Quick wins such as coding an LED light bulb to turn on can give you and your learners a boost in confidence

> IF YOU CAN GET A LEARNER TO REALISE THAT THEIR CODE CAN IMPACT THE REAL WORLD, THAT IS A POWERFUL STARTING POINT

Make them as responsible as you are for how the kit is organised, packed away, and looked after. Designate it as a role for your digital leaders, or rotate who is responsible each lesson or term. If you have learners who like to spend time in your classroom during breaks, get them to help organise things between lessons. There are learners who will happily do this, and it creates a shared sense of responsibility.

Breakages will happen. Some are frustrating, and you might need to replace parts, but LEDs are cheap. If one blows during a lesson, there is an opportunity to learn from that failure, and it should not cost much to replace, especially if you buy in bulk. You can mitigate some of this by choosing components that come already soldered and are easy to connect, although you will pay a premium over raw components, wires, and breadboards. Once you have invested in the kit, you want it to be looked after, so sensible storage is also worth some thought. Whether you are storing in shoe boxes, fancy plastic trays, or recycled food containers, include a list of parts, with images. This helps learners organise the kit while also supporting their vocabulary learning.

Working with your IT environment

Using school computer systems can present its own challenges. Many countries require schools to have web filtering; some schools will lock down USB ports, or otherwise limit what learners have access to. I have typically found IT support teams supportive and friendly, particularly when I have been able to convey a clear argument about the learning gains I perceive, or hope for, from an activity. That usually helps us to figure out a way of addressing the problem.

In one classroom, we ended up with our own small Wi-Fi hotspot. The devices could only connect to that hotspot. IT could isolate that subnet and add additional protections, and it could be activated only during certain times. In another school, I was able to get my hands on a few old machines that were due to be recycled; these sat at the back of my class, isolated from the network, and became physical computing workstations.

Many developers of educational physical computing devices are aware of these challenges and are working to make their tools as accessible as possible. In some markets, such as the USA, there is legislation about the accessibility of the hardware and software schools use, which is helpful in raising the standard, although it may present barriers when it comes to using some older tools.

All of this is a further argument for starting small. Test things out, find where the challenges are, work out whether you can mitigate them, and iterate.

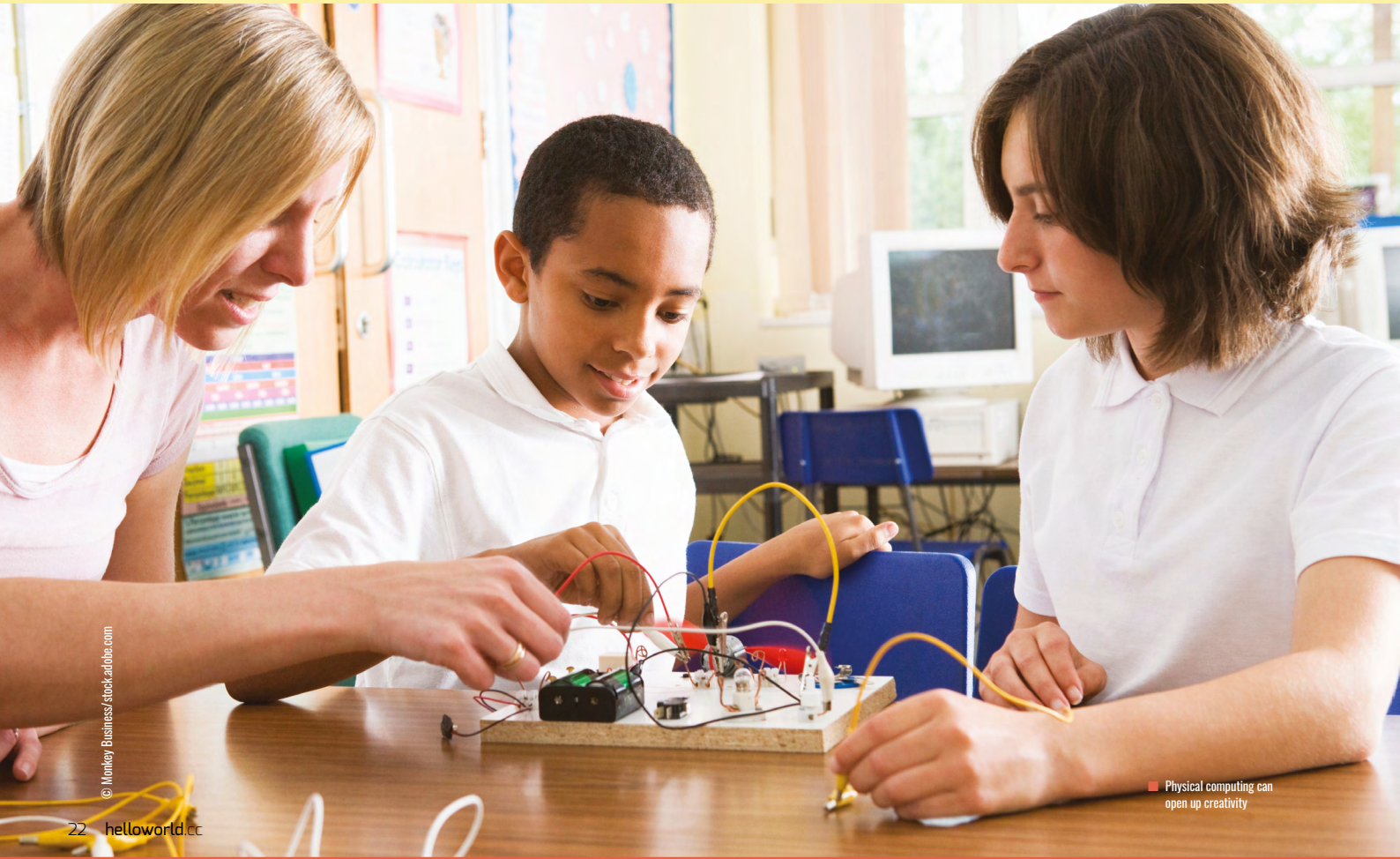
Starting small with little budget

My approach in the classroom has always been to start small, learn, and grow activities over time. The first physical computing experience I was able to bring to my classes was using old LEGO RCX robots in my first year of teaching. I got a handful on loan from the local authority, tested them with a small group of learners who came to a lunchtime club, and later used them in class. Fortunately, a set of emulated activities was available, and while half the class used them, the other half used the physical robots, and we rotated each lesson. Following a year or so of that experience, I was able to make the case to invest in the next generation of LEGO robots for the school. These became part of a curriculum unit as well as supporting a series of clubs and competitions.

Much later, having experimented with Arduinos and other microcontrollers, I got my first Raspberry Pi and brought it into my classroom. We passed it round, the learners did some activities on it, and we pushed code onto it remotely so that everyone could run their own code on it. Once I was able to demonstrate the impact of that to my head teacher, I was able to find pockets of funds from different budgets and different areas: enough for a handful, then a class set, then a class set of additional equipment and storage.

I asked older students taking a GCSE in computing to have their own Raspberry Pi so that we had a device we could program on and that they could take home. The school helped pay for some, but mostly the learners bought their own, which gave them a sense of ownership as well as easing the cost to the school.

There are so many tools and devices out there to help you and your learners get started that it can be hard knowing what's best for you. My advice is, do not fall for the shiny product that does one amazing thing really well. Once it has done that one amazing thing, it might not have a lot of longevity or lifespan in your classroom. I tend to go for lower-cost, more versatile



© Monkey Business/stock.adobe.com

Physical computing can open up creativity

hardware such as Raspberry Pi, micro:bit, or Arduino.

You don't have to use ready-made, off-the-shelf components, either. You may find that your science lab already has some of the equipment you need. We made buttons with bits of tin foil and folded cardboard, sensors with paper plates that squash together with tin foil inside, and switches from paper clips and fasteners. Conductive thread is useful and copper tape is excellent.

Finally, a word on clubs. Whether they are after-school or lunchtime, and whether they are official Code Clubs or not, clubs are a great space to test and evaluate devices and activities. You can have a single device, or test a selection of devices with your learners and see what works — what gives you the biggest range of activities, what is the most robust, and which ones learners gravitate to or find easier to program. Getting hands on with the equipment helps you understand the potential impact before you commit time and money to the project.



It is inspiring to see what students can build with a little bit of confidence, experience, joy, and junk

> DO NOT FALL FOR THE SHINY PRODUCT THAT DOES ONE THING REALLY WELL

What to teach

It is important to connect with and reinforce existing concepts through physical computing. For example, I have found that physical computing can help learners develop their concept of the notional machine, and their idea of how programs flow. Physical projects often rely on capturing some input, processing that data, making a decision, and producing an output. A physical computing device gives you a range of options for inputs and outputs. Yes, you could trigger an input event with a key press, but using a button, a gesture, a radio signal, an orientation, proximity, temperature, or sound opens up much more creativity. Likewise, a written message on a screen conveys information, but so does a sound, a light, a spin, a movement, or a transmission. And that's just the common built-in sensors and outputs — add some tin foil and sticky tape and there's no end to what you could make.

Knowing when not to

Physical computing is not the answer to everything. Just like any other context or modality, if you overuse it, or it becomes the default, it risks becoming stale, just as text on a screen would become repetitive and boring if you used it all the time to explain every concept. If you are having to spend significant time explaining a convoluted context in order to teach a simple concept, the payoff is questionable.

Sorting algorithms are a case in point. You are manipulating a bunch of data inside a data structure. You could do some of that through physical computing, visualising the sort for example, but I do not think it is actually doing anything to help with the construction of that concept. Operators and calculations are similar: the kind of thing you might end up using in a physical computing context, but probably best taught elsewhere and then applied where necessary. Program logic, control, decision-making, and flow, on the other hand, are good concepts to explore in this way.

Reflecting with learners on when technology is the right or wrong solution for a given problem is a healthy practice to have in any case. It's important to have a degree of criticality and openness, and a willingness to challenge the assumption that technology is the right tool for every situation.

Some resource costs are inevitable, but they can act as a prompt for critical reflection rather than deterring creativity. That may mean adopting a culture of repair, choosing sustainable alternatives, or simply deciding not to make at all when the cost outweighs the pedagogical value. (HW)



JAMES ROBINSON

James is a highly experienced computing educator and trainer with a background in computer science and ten years of classroom experience teaching a range of STEM subjects. James currently works to promote effective and evidence-informed practices in computing education pedagogy.

ROBOTICS FOR URBAN STEM

How physical computing helps engage students in underserved communities

Teaching computer science in urban neighbourhoods presents unique socioeconomic barriers. For a long time, the conversation focused entirely on what these schools lack — such as updated hardware or certified tech teachers. While acquiring laptops and a certified teacher is crucial, preparing an instructor to teach computer science is equally important.

Even the most advanced technology is ineffective if the teacher and the instructional framework fail to capture a learner's attention. By itself, learning to code can feel tedious and disconnected from reality. It is tough for basic loops and variables to compete with the fast-paced, high-stimulus technology students interact with every day.

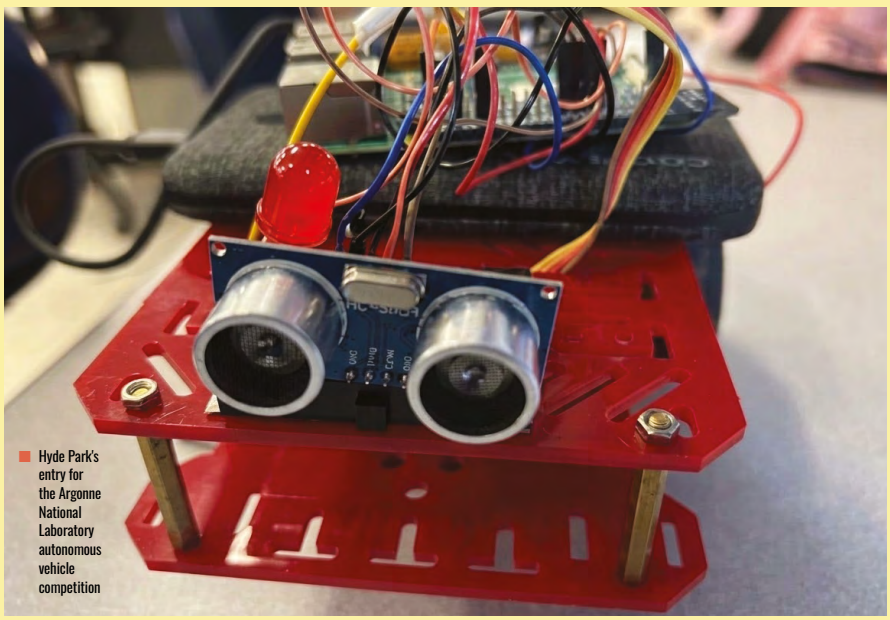
Teachers can engage their students by introducing physical computing: moving the text off the screen and into the real world, where students can see it working. When a student writes Python code and an actual robot moves, the mystery of technology disappears. As a result, they stop just consuming tech and start building, owning, and controlling it.

Closing the gap

To bridge this gap in access to computer science, Argonne National Laboratory implemented an autonomous vehicle competition in the Chicago area in the USA (**helloworld.cc/av-competition**). The core motivation was clear: give underserved communities access to the technology that is shaping the future. Rather than only teaching them technical skills, the programme also aimed to empower these students to step into leadership roles and actively shape the future of tech. More importantly, the organisation wanted Chicagoland students to realise that by mastering these skills, they wouldn't just be along for the ride — they would be the ones leading the charge and building the future of technology.

The competition was designed to show high-school students what real engineering looks like, taking them from a written proposal to a working final product. By building these projects from scratch, students get to experience the practical impact of physical computing first-hand. They are no longer simply practising syntax in a vacuum; they are managing a real-world project that requires critical thinking, hardware integration, and iterative troubleshooting. For participating Chicago Public Schools (CPS) students, this competition served as an entry point into the green economy and wider tech infrastructure, explicitly showing them how software engineering drives the movement of physical objects in the real world.

This passage from theory to practice is exactly what keeps early-stage college students engaged in engineering, and it works the same way for high-schoolers. When a student writes code that makes a vehicle move, they stop viewing computer science as an elite, abstract concept. The competitive element further sharpens their focus, giving them a concrete goal to work towards and a community of peers to collaborate with.



Hyde Park's entry for the Argonne National Laboratory autonomous vehicle competition

Conquering the obstacles

While the ideological benefits of physical computing are clear, implementing a robotics programme at an underfunded urban school throws up immense local hurdles. Bringing this project to my school, Hyde Park Academy, required us to simultaneously address financial barriers, technical knowledge gaps, and challenges related to student involvement. Being a Title I school (a school with a high number of



Hyde Park's robotics club won first place in the competition

students from low-income families) meant my school lacked both the funding and baseline student interest required to launch a fully functioning robotics team. Hardware is expensive, and equity is impossible without capital. Most of our students don't have $200 for a robot kit, nor do they have the patience to build a machine from scratch while simultaneously learning a new programming language.

Beyond the student experience, these systemic obstacles also permeated my professional development. I, and many other CPS teachers, began this journey with a limited understanding of robotics engineering. To remedy this knowledge gap, Argonne National Laboratory hosted an intensive six-hour training session for participating educators. The training took place at The National A. Philip Randolph Pullman Porter Museum, grounding our focus on technology in a space dedicated to historic labour rights and Black advancement. Having the training at the Pullman Museum reminded me of the struggles African Americans face, specifically in education, and spurred me on to implement an advanced STEM programme for a student population that is too often left out.

Empowering new innovators

After implementing the programme, I found that breaking down the barriers that keep young women of colour out of advanced STEM required me to push back against systemic gatekeeping.

Initially, my recruitment strategy was flawed. I assumed that a childhood interest in playing with robots would lead to a male-dominated pool of candidates to build them. However, the young men I recruited quickly lost excitement when faced with the approaching uncertainty of building an unfamiliar machine. The true turning point occurred when two young women overheard me talking about the robotics club and asked why they had not been allowed to join. In this way, I discovered that while girls might not traditionally play with robots, they still have a strong interest in constructing them. These two went on to become the most active members of Hyde Park's robotics club.

Once these two incredible students took over, they owned the entire engineering cycle from scratch — planning the timeline, building the frame, and wiring everything themselves. I only gave them a foundation in Python; the rest they learnt on their own.

For about a month, everything went well, but a few weeks before the competition, the girls hit a wall with some brutal mechanical issues and almost gave up. In my capacity as their mentor, I emphasised the immense progress they had made, reinforcing the idea that the journey of engineering is every bit as significant as the outcome. Reinvigorated by this perspective, the students dedicated their lunch breaks to rebuilding their robot, eventually engineering a vehicle that secured a first-place victory.

This experience underscores why a focused approach to physical computing in Title I schools is so vital. Structural inequalities and systematic gatekeeping in underfunded schools create a persistent opportunity gap that systemically discourages marginalised students from pursuing advanced computer science pathways. Urban students are often funnelled into passive computer literacy courses, while affluent schools offer rigorous engineering and programming tracks. Improving computer science education in urban schools requires adequate funding, teacher training, and increased access to advanced programmes; failing to do so will only widen the digital and achievement gaps for years to come. (HW)

> “STRUCTURAL INEQUALITIES AND SYSTEMATIC GATEKEEPING CREATE OPPORTUNITY GAPS



EUGENE POPE

Eugene is a computer science instructor with 23 years of expertise in cultivating the next generation of technical leaders. He uses enquiry-based learning and a culturally relevant curriculum to transform students from passive technology users into confident, independent problem-solvers. Eugene was named a 2026 Chicago Public Schools CS Hero.

MAKE CODING TANGIBLE

How unplugged resources are being used in South Africa to support a new coding and robotics curriculum

In 2025, South Africa's Department of Basic Education introduced an ambitious new coding and robotics curriculum to equip learners with essential 21st-century digital skills, foster problem-solving and creativity, and prepare them for future careers in a rapidly digitising economy. The intention is to build computational thinking, design thinking, and digital citizenship from an early age. However, many schools, particularly those in underserved communities, lack the reliable electricity, internet connectivity, and learner devices needed to deliver a digital curriculum. With fewer than 20 percent of Foundation Phase (ages 5–9) schools having access to this essential combination of infrastructure and technology, there is an urgent need for alternative approaches.



Learners use the Tangible app on the teacher's smartphone to check and debug their code

In response, Nelson Mandela University and the LEVA Foundation joined forces to develop Tangible, an affordable, tactile, and engaging unplugged resource that enables schools to introduce coding and robotics concepts without the need for computers or internet access.

Making coding tangible

Tangible (**helloworld.cc/tangible**) is a physical programming environment in which children construct, execute, and debug algorithms using interlocking coding tiles similar to jigsaw pieces. Each tile represents an instruction, allowing children to build programs by arranging the tiles into sequences. They then physically carry out the instructions themselves.

A typical Tangible activity works like this:

1. **Solve a challenge:** Each activity has a specific goal, such as navigating from a starting point to a destination while avoiding obstacles. Learners must determine which instructions, and in what order, will achieve the goal.
2. **Choose the tiles:** Each tile represents an instruction or coding command, such as 'move forward', 'turn right', 'repeat', and so on. Learners select the tiles they need to solve the challenge.
3. **Build a program:** The learners arrange the tiles in the correct order. The resulting sequence becomes their program — a physical representation of a set of instructions.
4. **Execute the program:** Instead of a computer executing the instructions, the children do it themselves. They move around, role-playing the sequence they have created.
5. **Debug the program:** If they do not reach the desired goal, they examine their sequence, identify where they went wrong, and modify the program. In this way, debugging is introduced as a natural part of problem-solving.
6. **Increase the challenge:** As learners progress, activities introduce more sophisticated concepts such as loops, procedures, conditional instructions, and optimisation, requiring increasingly complex programs.

The clever part of this approach is that code becomes something children can see, manipulate, and discuss, working physically and collaboratively, rather than something they simply type into a computer. This makes the normally abstract concept of an algorithm tangible. Learners can:

- Physically construct a program
- See the order in which instructions will be executed
- Change individual instructions
- Compare different solutions
- Collaborate with other children
- Immediately test whether their program works

The Tangible activities are built around challenges, problem-solving, and algorithm design rather than programming-language syntax. This makes the resource particularly suited to Foundation Phase Coding and Robotics, as young children can explore fundamental computational concepts through movement and play before being expected to work with conventional programming languages. Importantly for

underserved schools in South Africa, they now have a resource that enables them to deliver important aspects of the coding and robotics curriculum.

Tangible provides a complete suite of resources for teachers, including lesson plans and training videos (**helloworld.cc/tangible-training-video**).

Not just for underserved schools

It would be a mistake to think of Tangible as simply a solution for schools lacking access to computers. I have used Tangible in my computing classes in the UK, where every child has a Chromebook. I found it particularly effective as a way of getting children out of their seats and away from their screens, and they loved it. This is an important point. Unplugged coding should not necessarily be seen as a second-best alternative to computer-based programming. It can be a powerful teaching approach in its own right.

environments. They also provide opportunities for movement, discussion, collaboration, and experimentation that can be difficult to achieve when every learner is sitting in front of a screen. Tangible demonstrates that coding does not have to begin with a keyboard and a screen. It can begin with a challenge, a group of children, a handful of physical instructions, and the question: 'Can you work out how to make it happen?'

The Tangible app

For teachers or parents with a smartphone, there is a Tangible app that can be downloaded for free. The app features a series of interactive educational challenges. Once the learners have used the tiles to build a program, the teacher or parent uses their phone to scan the tiles. The app will then automatically animate the code's output on the screen, providing the learners with a visual representation of the code.

> UNPLUGGED CODING CAN BE A POWERFUL APPROACH TO PROGRAMMING

Physical activities can help children understand computational concepts before they encounter the complexity of programming syntax and software



A Coding4Mandela judge marking a team's code

Community coding with Coding4Africa

Competitions are popular across Southern Africa as a way of engaging communities, developing teamwork, and motivating children to work together to solve problems. One such initiative is the Coding4Mandela Challenge. It began as a one-day event on Mandela Day, when South Africans give 67 minutes to service in honour of Nelson Mandela's legacy, and was initially simply a coding competition intended to inspire learners.

Over time, this evolved into Coding4Africa, a much larger initiative that runs across an extended period (**helloworld.cc/Coding4Africa-2025**). The intention shifted from a symbolic Mandela Day activity to a sustained programme. It is now becoming a movement that aims to make coding education accessible to children regardless of their background, location, or access to digital technology.

The Coding4Africa initiative brings together young learners from different countries and communities around a shared vision of empowering young Africans through digital skills and innovation. As the movement continues to grow, each event demonstrates how coding can be used not only to develop coding skills, but also to build confidence, teamwork and aspiration.

United Siyafunda, a nonprofit which promotes STEAM education in South Africa, has participated proudly in the competition for two consecutive years. This year, the organisation took the bold step of facilitating the event in Gauteng, a province in South Africa, with an ambitious target of reaching 2000 learners. United Siyafunda organised events across Gauteng, giving children from underserved communities the opportunity to showcase their computational thinking skills. These platforms highlight the learners' creativity and problem-solving abilities and demonstrate the transformative power of coding and robotics in shaping Africa's future innovators.

Together, approaches such as Tangible and initiatives such as Coding4Africa are helping to create a future in which every child has the opportunity to develop the computational thinking, creativity, and problem-solving skills they need to shape the world around them. (HW)



GRAHAM HASTINGS
Graham is an education consultant for United Siyafunda, a nonprofit promoting STEAM education in South Africa.

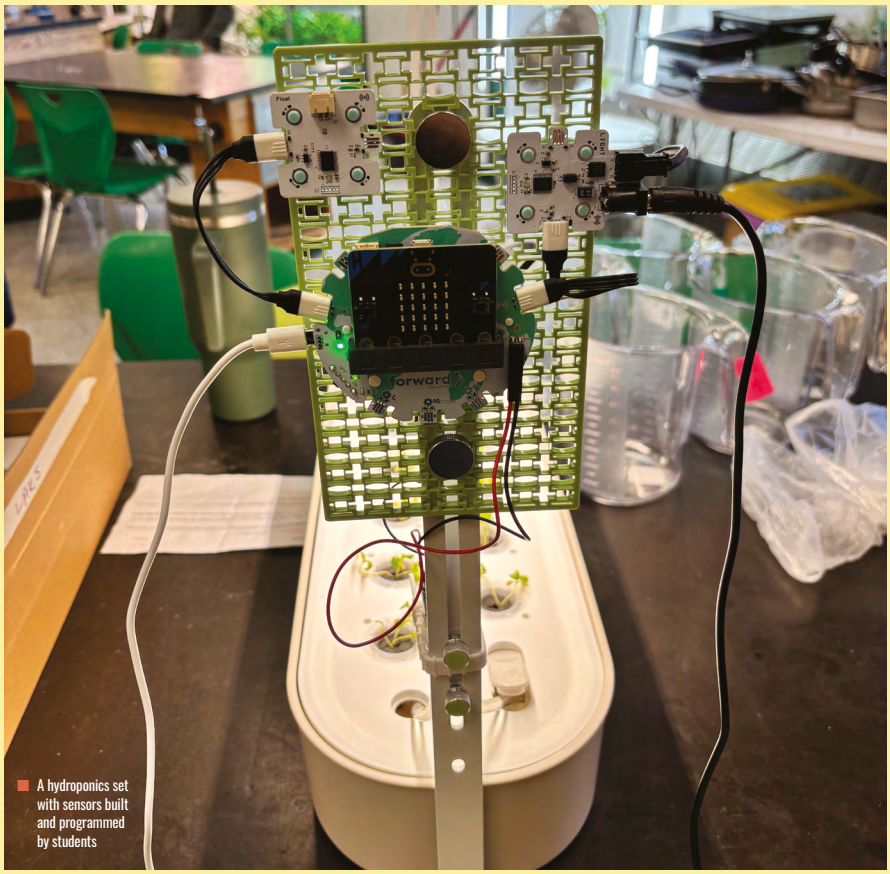
FROM DIRECT INSTRUCTION TO DISCOVERY

Shifting to hands-on physical computing empowers students to become more collaborative, independent, engaged learners

Before I became a computer science teacher five years ago, I had been teaching maths for 16 years. As a maths teacher, I often used a direct instruction method in which I began lessons with a mini-lesson, then gave a guided practice, and ended the lesson with some time for independent practice. This worked well for students, as I was introducing specific skills that they would later apply to problems on the state exam. Because I was comfortable with this structure, when I first started teaching computer science, I used a similar direct teaching method, with a mini-lesson, a codealong, and then independent or pair programming. However, that was about to change.

Rethinking my teaching

Although direct teaching worked for some lessons, I often found that my students were not trying to write code on their own, because they knew I would show them how to do it. A large component of computer science is debugging, and I wanted students to have the independence to figure out solutions without always relying on me. However, releasing control of the lesson to the students has always been difficult for me. It requires me to take a step back and give them the freedom to make mistakes. This can be challenging, because many of my students are afraid to make mistakes. I don't like watching them getting frustrated and struggling, so I often step in rather than giving them time to figure it out themselves.



A hydroponics set with sensors built and programmed by students

This year, after more than 20 years of teaching, I decided to take a risk and try something new. At the CSTA Conference 2025, I learnt about a company, Forward Education, that makes Climate Action Kits with micro:bits to create prototypes that model real-world technology (**helloworld.cc/climate-kit**). As part of the mission of my school is to promote environmental integrity through hands-on learning, these kits seemed like a good way for me to try out some new teaching strategies.

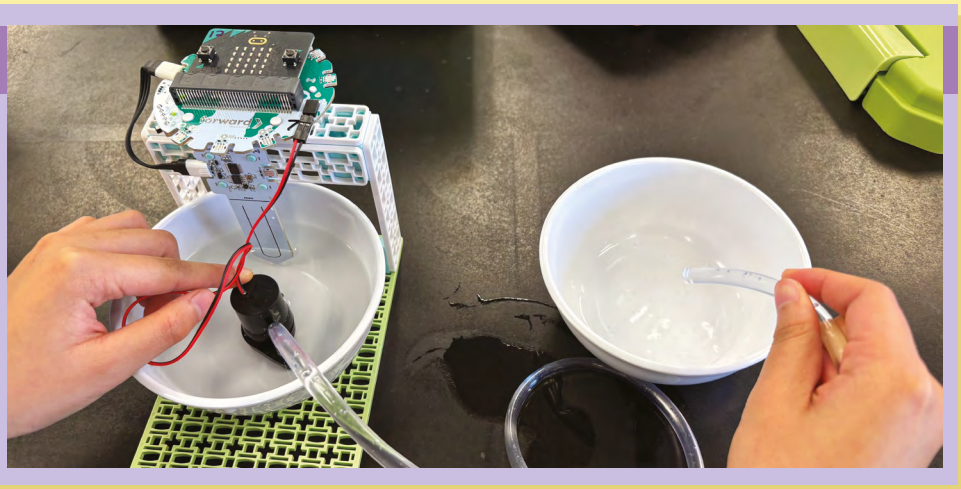
Exploring physical computing

On the first day, I handed out the micro:bits and students explored on their own. They chose projects on the MakeCode website and followed tutorials to help them write programs for their micro:bits. That night, I chose a lesson for the next day from the Forward Education website and built the model — about wildlife crossings — myself. It was the first time I had tried physical computing.

The next day, we began by learning about

TIPS FOR CHANGE

- Try out projects on your own first
- Designate students to help others
- Be vulnerable — it's OK if you don't know everything
- Give students extra time for exploration
- Take risks — it's OK to make mistakes
- Show students how you work through problems
- Charge the necessary components



wildlife crossings, and students then worked in groups to make a working prototype of a crossing using the Climate Action Kit. I immediately saw more engagement in the activity than usual, especially from students who would not usually participate in lessons. When students were stuck on something, they had to work together to figure it out; I could not tell them what to do, as it was all new to me too.

The prototype included a solar sensor, a sonar sensor, and servo motors, but it used a button on the micro:bit to control the movement of the servo, rather than the sensors. As I was walking around the room, I observed two students, both of whom had learning difficulties and one of whom was also an English language learner, clapping next to their model. When I asked them about what they were doing, one said she had wondered whether they could get the gate to move using the sonar sensor, and then the other said he had figured out how to write the code to do that. On the first day of using the Climate Action Kit, the students were already working together to problem-solve on their own.

We continued the next few lessons in a similar way, with a short introduction to an environmental issue and then a model build. Each day, the students got better and more engaged with their prototypes. They were able to troubleshoot within their groups and would then help other groups of students who were struggling. Throughout the lesson, rather than standing at the front of the room doing a codealong or spending a significant amount of time working with individual students, I circulated and helped groups of students figure out solutions on their own. I had time to check in with all the students in the class, rather than just a few. Additionally, I felt energised after these lessons, rather than exhausted, as I usually was after lessons using direct instruction.

Reflecting on change

For the final project, the groups each chose a project related to an environmental issue they were interested in and presented to the class. Students researched their environmental issues using reliable sources and built a prototype related to the issue. The presentation included an explanation of the topic, how the prototype addressed the issue, a demonstration of the prototype, and an explanation of the micro:bit code. Students enjoyed demonstrating how their prototypes worked and felt comfortable speaking about their projects to the class.

After we finished this unit, I asked the students to fill in a survey. The answers were very positive. Students liked the hands-on projects related to real-world issues. This helped them to understand computer programs better, as lines of code were used to control physical components. They liked working with partners and helping each other figure out how to build models that moved and used sensors. Some of the students wanted to go even further with the micro:bits and Climate Action Kits and work with larger groups to build their own models related to environmental issues. Overall, they felt more engaged than in previous units.

Experimenting with physical computing made me rethink how I teach. Moving forward, regardless of the content, I plan on making lessons more hands-on and giving more freedom to students during class. I feel more comfortable releasing control and am less worried about students not being able to help themselves. When you provide students with more opportunities to work together and solve problems on their own, they become more engaged, understand better, and enjoy lessons more. (HW)



MARISA LAKS

Marisa is a computer science teacher in New York City, USA. She facilitates citywide professional learning, serves as NYC regional vice president for CSTA New York, was a CSTA Impact Fellow, and presented at CSTA 2024 and 2025. Marisa is passionate about equitable CS education.

DISABILITY IN THE MAKERSPACE CLASSROOM

Using accommodations in physical computing spaces

In every classroom, educators encounter learners with a wide range of needs. In the United States, 15 percent of public-school students receive services under the Individuals with Disabilities Education Act, with Specific Learning Disability (SLD) representing the largest category, at 32 percent (National Center for Education Statistics, 2024). SLD includes students with language-based learning disabilities (LBLD), a population I have worked with exclusively for 16 years at Landmark High School in Beverly, MA, in the United States. For the past eleven years, I have taught computer science in a makerspace setting, continually adapting instruction to meet diverse learning profiles.

SLD manifests in many ways. Students may struggle with reading comprehension, fluency, and decoding, as well as oral and written expression. Executive function challenges, such as organising materials, managing time, and processing information, are also common. Working memory and processing speed can further impact how students engage in learning. While these challenges are often associated with diagnosed disabilities, many students can benefit from support in these areas.

Differentiation is essential to meeting these varied needs without overwhelming educators. By breaking units into flexible, student-centred lessons, teachers can provide access points that allow each learner to engage meaningfully with the content.

Reasonable accommodations

The makerspace environment, particularly in computer science and physical computing, naturally supports inclusive practices. Multimodal instruction is foundational in my classroom. Presenting information visually, auditorily, and in writing allows students to choose how they engage with the material. Providing clear models and exemplars is equally important. When students can interact with a finished product and reference a rubric, expectations become more concrete, especially when learning new platforms such as Arduino.

Scaffolding is another key strategy. By gradually taking on more responsibility, students build the skills they need for independent project work, including breaking tasks into manageable steps, iterating on ideas, and incorporating feedback. This process not only develops competence but also helps to strengthen students' confidence and identity as creators and problem-solvers.



The makerspace environment naturally supports inclusive working practices

FURTHER RESOURCES

AccessComputing: An organisation focused on increasing the participation of people with disabilities in computing (helloworld.cc/access-computing)

The DO-IT Center: Working to empower people with disabilities through technology and education (helloworld.cc/do-it)

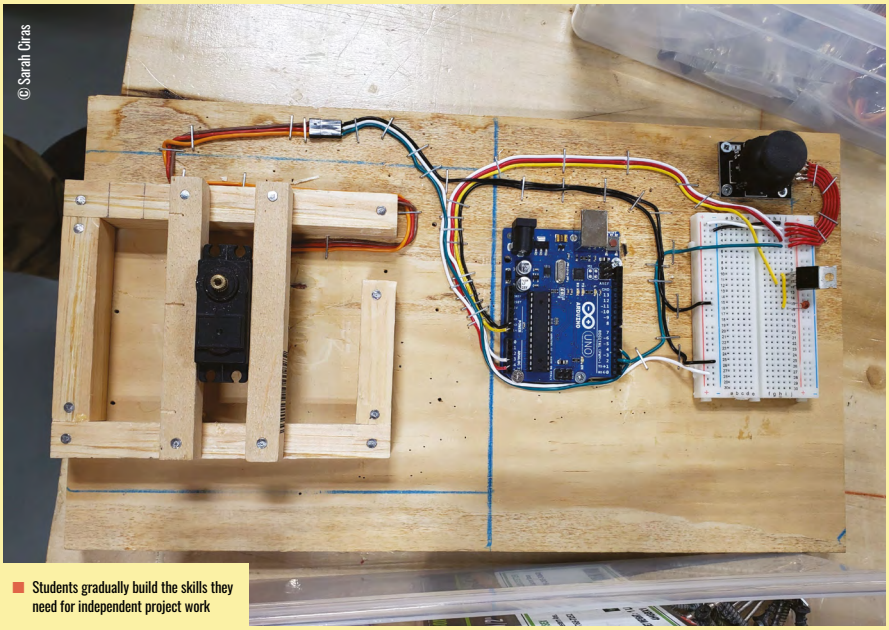
Computer Science Teachers Association: A community of educators with regular conference and professional development opportunities (csteachers.org)

Universal Design for Learning: Learn how to make your classroom work for all learners (helloworld.cc/udl-guidelines)

Supporting executive function is critical in a busy makerspace filled with tools and materials. Organisation systems such as labelled storage, clear bins, and pegboards help students quickly locate and return items, reducing cognitive load. Built-in clean-up time ensures that organisation is maintained without stress or haste.

Time management is another important consideration. In project-based environments, it is easy for both students and teachers to lose track of time. Using interval timers helps build awareness and pacing. Students also benefit from opportunities to process their ideas verbally or through strategies like the rubber duck method. Flexible timelines further reduce pressure, allowing students to produce higher-quality work.

An often overlooked strategy is intentional waiting time. Giving students time to write and organise their thoughts before responding promotes more equitable participation. In my classroom, students use journals and graph paper to capture ideas, sketch designs, and plan projects, providing accessible tools for thinking and iteration.



Students gradually build the skills they need for independent project work

Project-based learning as differentiation

In our makerspace, known as the Steamworks, students follow a structured curriculum in the first half of the year before transitioning to independent or small-group projects. While this shift increases engagement, it also introduces challenges. Some students struggle to settle on an idea, instead jumping between concepts. To address this, we encourage persistence through uncertainty and provide curated resources such as Instructables (**instructables.com**), Makezine (**makezine.com**), and Hackaday (**hackaday.com**) for inspiration.

When students need additional support, I guide them through a narrowed set of project options based on their interests. Limiting choices can reduce overwhelm, while still preserving student agency.

Ownership is central to this model. As a facilitator, I rarely say no. Instead, I prompt students to consider feasibility, cost, and timelines. Students take on the role of project managers by developing proposals, setting milestones, sketching designs, and tracking materials through spreadsheets. This process builds both technical and executive function skills.

Our classroom frequently utilises platforms such as Arduino and Raspberry Pi. One notable project involved students designing and building a full-size arcade cabinet. They collaborated to construct the physical structure using power tools, wired controls for two players, and configured Raspberry Pi systems with multiple emulators. Despite its complexity, the project resulted in a tangible product that served the broader school community, demonstrating just how much students can achieve when given the right support.

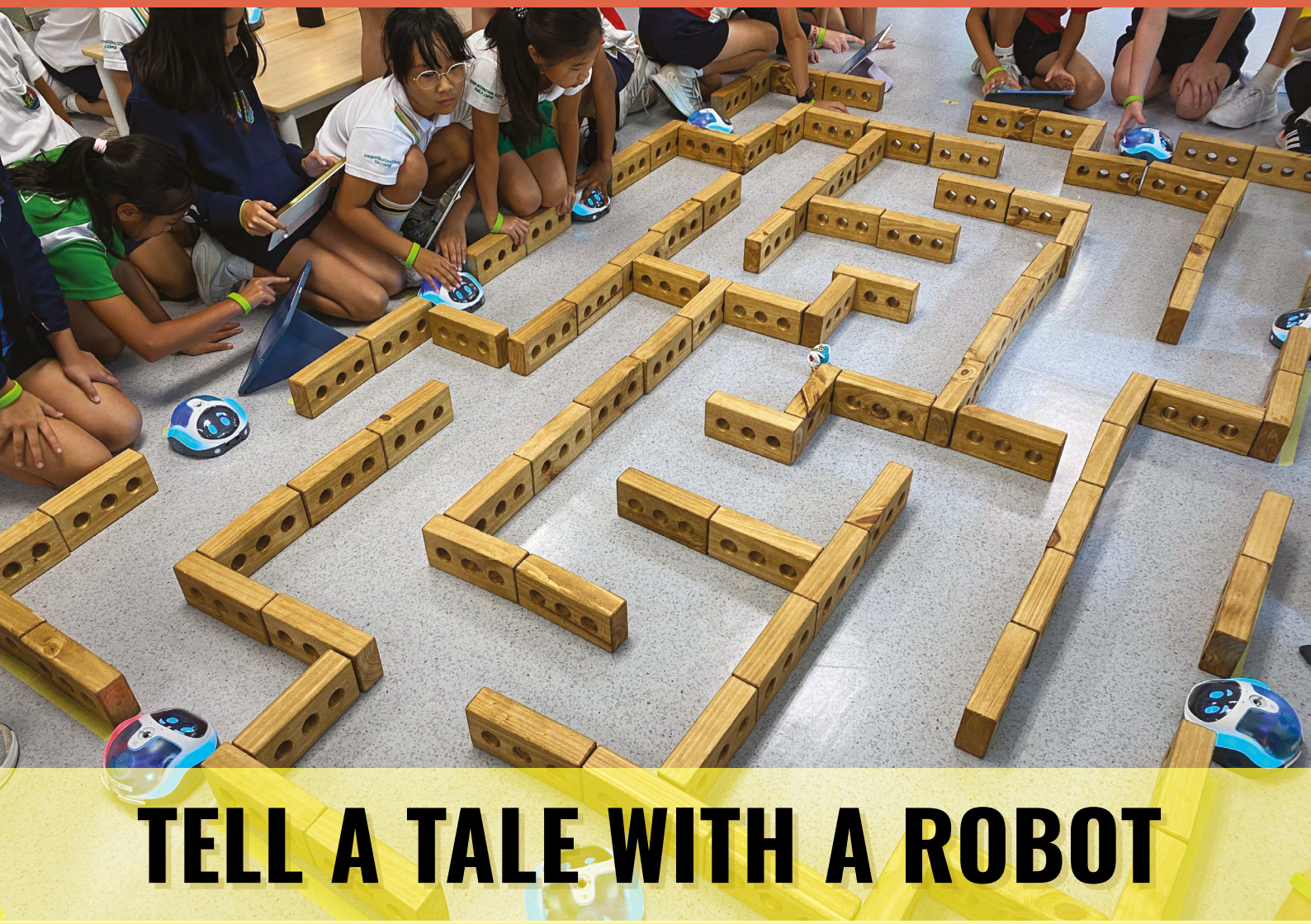
Project-based learning in an interdisciplinary makerspace allows students to see beyond their challenges and recognise their capabilities. By embedding Universal Design for Learning principles, including multiple means of engagement, representation, and expression, we can create rigorous, inclusive environments in which all students thrive. (HW)

> “SUPPORTING EXECUTIVE FUNCTION IS CRITICAL IN A BUSY MAKERSPACE FILLED WITH TOOLS AND MATERIALS



SARAH CIRAS

Sarah is a special educator and computer science teacher at Landmark High School in Beverly, MA, USA. She is a CSTA Impact Fellow alum and was the 2025 Learning Disabilities Association of America Sam Kirk Educator of the Year.



TELL A TALE WITH A ROBOT

James Abela and **Sethi De Clercq** share how story mats, costumes, and a pedagogy-first approach turn primary robotics into purposeful programming, problem-solving, and imaginative design

From ages 5 to 8, robotics can sometimes be introduced as a set of buttons to press: forward, left, right, go. Children quickly learn the mechanics, but the learning becomes much richer when the robot has a reason to move. A mat turns the floor into a world. A costume turns the robot into a character. Together, they give pupils a story that makes algorithms, sequences, and debugging feel purposeful. Crucially, it shifts the focus away from the hardware itself; by teaching the wider thinking, rather than just the tool, we ensure children are developing true computational fluency rather than just learning which buttons to press. We want them to see past the plastic, past the 'toy', and understand the logic beneath it.

This is especially powerful with younger learners, because they are still switching between physical experience, spoken language and drawing for communication, and developing their abstract thinking. Before a child can confidently describe an algorithm on paper, they can often act it out, point to a square, place a card, or say, "The robot needs to go to the treasure." This natural, spoken approach mirrors the way modern developers interact with technology today. When a child uses their voice to direct a peer or a robot, they are engaging in the foundational steps of vibe coding, using natural, descriptive language to express logical intent before translating it into formal syntax. It makes computer science accessible by meeting the children exactly where their language skills are. For example, in one class, pupils began as

'human robots', following commands exactly and noticing why precise instructions matter. They then moved to programming a Tale-Bot (**helloworld.cc/tale-bot**), using command cards to plan a route, testing it on a mat, and fixing mistakes when the robot did not reach its goal.

Worlds on mats

A good mat is more than decoration. It is a shared problem space. A number mat can support counting, position, and directional and orientation language: move from three to seven, visit the even numbers, avoid the multiples of five, or find three different routes to the same target. A story mat can become a forest, town, museum, beach, castle, space station, or school. In each case, the computational thinking is the same, but the motivation changes.

For lower primary, a mat can give pupils an immediate goal: take the doctor to the patient, guide the bee to the flower, move the recycling robot to the correct bins, or help the lost child find the classroom. The teacher can ask, "What does the robot need to do first? What should it do next? How many times will it move forward?" Pupils can build the sequence with physical cards before pressing any buttons, reducing cognitive load and encouraging talk. By physically laying out these tangible command blocks on the floor, children construct a visual script before committing to the execution or 'acting'. This concrete manipulation bridges the gap between spoken instructions and symbolic programming. We can even introduce basic loop concepts early on by asking questions such as, 'Instead of placing three forward cards, how else might we show this?' This shifts the activity from simple button pressing to foundational architectural design, keeping the cognitive demands developmentally appropriate yet conceptually rich.

For upper primary, the same idea can become more demanding. Pupils might design a rainforest rescue mat with restricted paths, animal habitats, and hazards. They could compare two algorithms and decide which is more efficient, add rules such as 'the robot cannot cross water', or create a route that includes loops. A historical mat could ask pupils to guide a museum robot through artefacts in chronological order. A geography mat could ask a climate robot to visit renewable energy sources and avoid areas at risk of flooding. The mat gives context, but it also creates constraints, and constraints are where good problem-solving begins. When we introduce these constraints, we aren't just giving learners a harder game; we are introducing them to data structures and rule-based logic in an entirely visual, friction-free sandbox.

MAT IDEA

Mats do not have to be expensive. Try masking tape for a fun maze, a laminated grid, a story map drawn on sugar paper, numbered carpet tiles, printed classroom locations, or a reusable transparent mat over different pictures. Add command cards, small obstacles, role cards, and one deliberate bug. The same robot can become a guide, cleaner, explorer, delivery bot, or museum assistant. By keeping the materials low-tech and physical, the focus stays exactly where it belongs: on the core computational concepts and the children's own creative problem-solving.



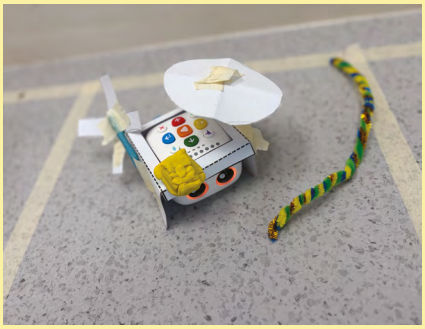
> "WE WANT LEARNERS TO SEE PAST THE PLASTIC, PAST THE 'TOY', AND UNDERSTAND THE LOGIC BENEATH IT



Costumes create purpose

Costumes and accessories make the robot's job visible. A paper umbrella turns it into a sun-protection robot. A small arrow sign makes it a guide. A cardboard scoop makes it a cleaner. A message pouch makes it a delivery robot. In our planning work, children considered school problems such as messy classrooms, hot weather, getting lost, busy teachers, and wasted electricity. Their ideas included sweeping floors, sorting recycling, holding a fan, giving directions, projecting arrows, handing out worksheets, and turning off lights.

This matters because the costume links the program to a need. The question is no longer 'Can you move the robot to B4?' but 'Can your guide robot help a new pupil reach the library?' or 'Can your energy patrol robot check the lights in each classroom?' The movement still requires sequence, direction, and debugging, but the task now includes empathy, design, and explanation. Pupils have to think about who the robot is helping, what the problem is,





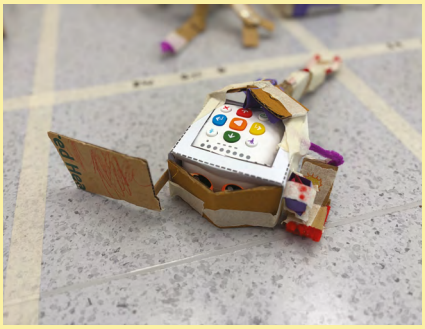
and whether the prototype solves it. See more about design thinking in Hello World issue 24, pages 58–59 (**helloworld.cc/24**).

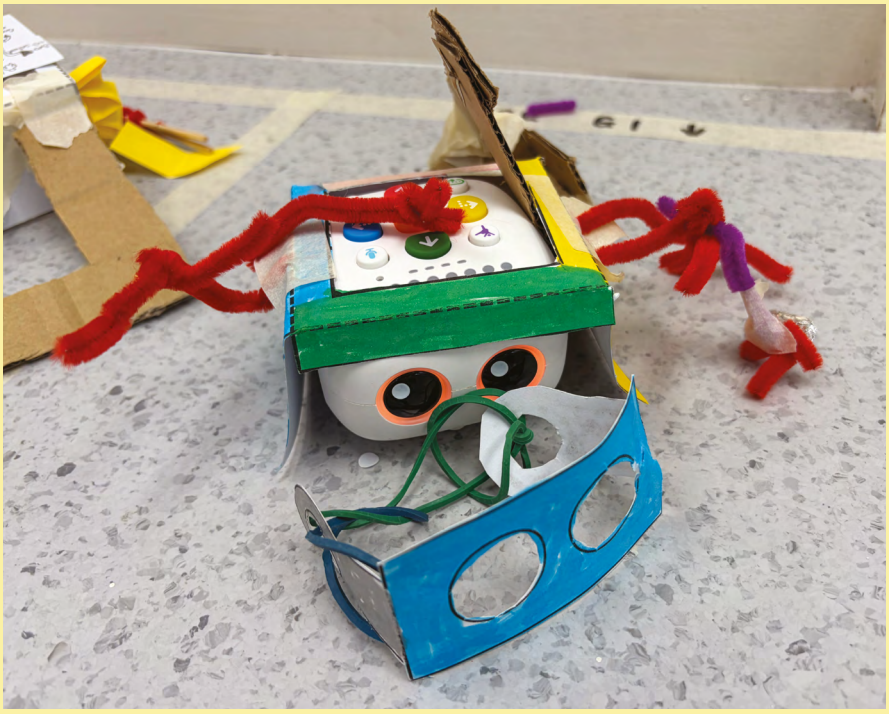
Costumes also invite children to test their ideas realistically. A top-heavy accessory may fall off when the robot turns. A large scoop may block movement. A sign may need to face the right way for the story to make sense. These are not failures; they are design feedback. In the classroom, it is useful to remind pupils that the invention is a mini prototype, not a life-size machine. This keeps the task achievable while keeping the imagination alive.

Debugging the story

Story mats make debugging easier to see. If the robot stops at the wrong square, pupils can point to the exact moment where the story broke: it missed the patient, crossed the river, delivered the letter to the wrong teacher, or turned away from the treasure. This makes debugging less abstract. To anchor this structured exploration, we look to the PRIMM framework (Predict–Run–Investigate–Modify–Make). Before the robot even moves, we ask pupils to Predict the path based on their physical command cards. They Run the program







on the mat, Investigate any deviations together, Modify the card sequence based on what they observed, and finally Make their new, functional story a reality. And again, children can trace the cards, replay the route with a finger, predict the outcome, test the program, and then change one instruction at a time.

One useful activity is to ask a group to create a route with one deliberate bug, then swap mats with another group. The second group has to find the bug, explain what went wrong, and repair the sequence. This encourages careful reading of code and respectful collaboration. Another variation is to ask pupils to solve the same story in three different ways, comparing routes for length, clarity, or reliability.

Mats and costumes can also support inclusion. When pupils rely only on spoken instructions, unfamiliar language, unclear images, or missing resources can make a task feel unfair. A well-designed mat provides visual clues, shared vocabulary, and a common reference point. Children can gesture, rehearse directions, use command cards, and work with a partner. This helps multilingual learners, pupils who need more processing time, and pupils who find abstract instructions difficult.

Ultimately, by lowering the barrier to entry through physical narratives, we introduce complex computational architecture early, without the syntax fatigue, long before screens are introduced, and through one of the most powerful pedagogical tools we have access to as educators: storytelling! Design thinking is all about encouraging empathy and trying to answer the question, 'How did your invention improve someone's life?' Pupils might build a classroom cleaner, a weather helper, a school navigator, or a calm-walk companion. They plan the mat, design the accessory, write the sequence, test the tool, and debug it. By the end, they have not just pressed buttons. They have told a story, solved a problem, and experienced computing as something human. (HW)

> "DESIGN THINKING IS ABOUT EMPATHY AND ANSWERING THE QUESTION, HOW DID YOUR INVENTION IMPROVE SOMEONE'S LIFE?

SETHI DE CLERCQ AND JAMES ABELA



Sethi is head of Key Stage 1 and the computing lead at Rugby School Thailand. He is also the founder of the Flipped Classroom Tutorials YouTube channel and co-founder of **ReadySetCompute.com**, focusing on a pedagogy-first approach to educational technology.

James is director of digital learning and entrepreneurship at Garden International School in Kuala Lumpur, Malaysia. He is founder of the South East Asian Computer Science Teachers Association and co-founder of **ReadySetCompute.com**.

MACHINE LEARNING IN PHYSICAL MOTION

How Digital<all> and micro:bit's CreateAI tool help young people explore machine learning through movement

I hope we have all experienced the magic of literally putting coding into our students' hands with micro:bits — the best way to bring computing to life. At Digital<all>, we use micro:bits all the time for exactly this reason. We love to make coding and STEM education meaningful, purposeful, creative, and engaging, and the micro:bit is the perfect vehicle for this. In our most recent workshops and resources we have gone a step further, using the micro:bit's CreateAI tool (createai.microbit.org) and our own bespoke interfaces to create powerful student experiences.

CreateAI and machine learning

The CreateAI function is not new, but it continues to wow our students and is a great example of how physical computing is the best medium for helping them make a meaningful connection between coding and the real world.

For the uninitiated, CreateAI is a free tool that makes it easy for students to explore AI through movement and machine learning. This means you can connect your micro:bit to the CreateAI web page, record your movements using the micro:bit, and then use machine learning to train the micro:bit to independently recognise categories of movements. And the very best feature? Programming your micro:bit to make decisions based on the recognition of those movements.

Landslides, big waves, and journeys into space

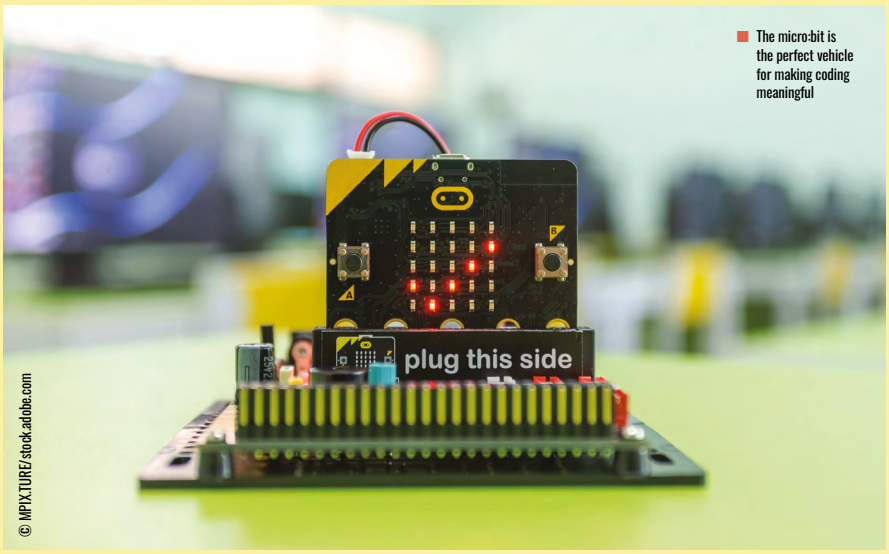
In all our outreach, we try to take real-world scenarios and show how coding and technology relate to them, with the ambition of showing young people what different careers could be open to them and what sort of skills they might need to access those careers, while of course having fun as well. Here is a selection of our favourite CreateAI workshops.

Landslide monitor

In our Coding for Climate Action scheme of work, winner of the Third Sector Awards Sustainable Initiative of the Year, we include a lesson on using the CreateAI function to develop a landslide monitoring system. Students train their micro:bits to recognise when the land is stable, when there's a slight vibration, and when there is a landslide. The micro:bit is then coded to emit a warning signal based on the movements it detects. A possible extension

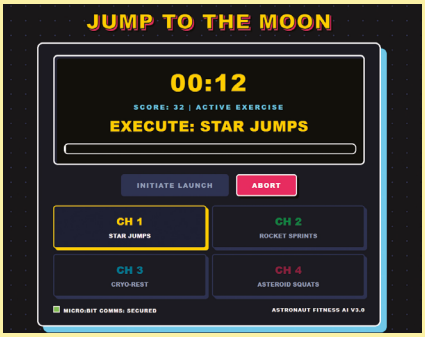
DIGIT<ALL>

At Digital<all> we specialise in bringing the computing and STEM curriculum to life. We offer teacher training, resources, and workshops that help young people develop digital and AI literacy and see how these skills can directly apply to their lives, whether that's working towards a career, developing personal skills, or tackling local and global challenges such as climate change. In the last three years, our work has reached over a million young people. We were awarded Small Charity of the Year and Sustainable Initiative of the Year by the Third Sector Awards.



The micro:bit is the perfect vehicle for making coding meaningful

© MPIX.TURE/stock.adobe.com



Learners can program the micro:bit to recognise fitness activities

would be to build a distributed system in which the micro:bit that detects the landslide sends a radio signal to an alarm micro:bit. After all, why would an alarm going off under a load of mud make sense?

Jump to the Moon

As part of our space and science outreach programme, our workshops and resources help young people see the value of digital skills, STEM technologies, and space exploration. Astronauts need to stay fit, and the Jump to the Moon workshop is a great way of demonstrating that while also showing how machine learning works. Again, the students train the micro:bit to recognise a series of actions — in this case students train their micro:bit to recognise star jumps, rocket sprints, cryo-rest, and asteroid squats. Learners can then program the micro:bit to request different fitness activities. This is a great way of teaching programming fundamentals such as variables, assigning and using random numbers, iteration, and selection.

In our workshops we have taken things a step further: students can program their micro:bit to interact with our bespoke user interface, which will tell them which actions to do when a timer counts down.

Stick Surfer

The Stick Surfer workshop is similar to the fitness session, but instead of focussing on fitness, it helps train people's surfing skills. This one is my personal favourite! While training the micro:bit to recognise how to paddle, pop up, and balance, students get to practise their surfing actions and perfect their micro:bit's ability to recognise them when they happen. After going through the iterative process of training, testing, and retraining their machine learning model, students code the micro:bit to feed back when they complete each action correctly. This is vital, as the workshop ends with using the micro:bit to send data to our live Stick Surfer game, where students see who can surf a big wave without wiping out.

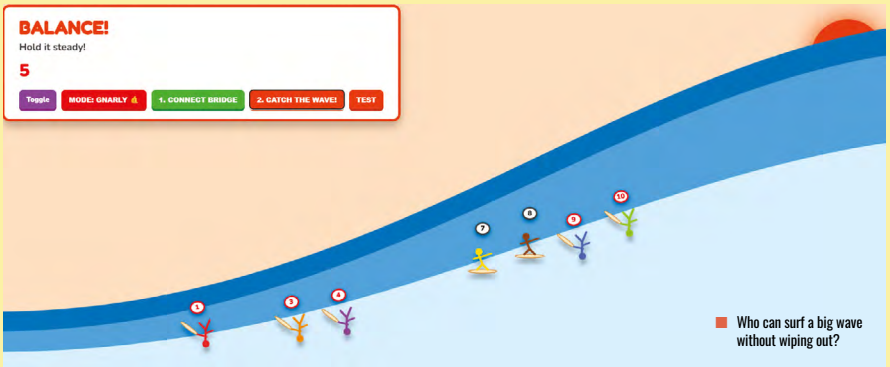
> "OUR WORKSHOPS AND RESOURCES HELP YOUNG PEOPLE SEE THE VALUE OF DIGITAL SKILLS AND STEM TECHNOLOGIES

Making learning purposeful

Digital education should be bigger than the sum of its parts. It's great to study programming fundamentals, iterative processes, networks, and AI literacy, but they are even better when embedded in contexts that make people want to get up, run around, or feel able to make a difference to their lives, their community, and the planet. That is at the heart of what we do. Judging from the smiles, feedback, and repeat visitors, we are confident we are making a difference. And if this reads like a love letter to micro:bits, that's because it is!

The Micro:bit Educational Foundation has a host of free resources, such as a sports data logger and storytelling friend — do check them out. We use our own resources and materials, which are all free and can be downloaded from our website or booked as a workshop if you're in the UK (**digitall.charity/resources**).

If you would like to know more about the work we do and how we have reached over a million young people in the last three years, there is plenty more to see on our website: **digitall.charity**. (HW)



Who can surf a big wave without wiping out?



MATT HEWLETT

Matt is a computer science teacher and education and community lead for Digital<all>. He has worked with schools in both the primary and secondary sectors, supporting curriculum development, student engagement, and raising attainment. A passionate advocate of environmental causes, when not working for Digit<all> he is usually on Dartmoor or at the beach.

COMPUTER SCIENCE LEARNING BEYOND THE COMPUTER LAB

Five ways to bring physical computing and AI to rural learners



What if the best place to begin computer science education isn't in the computer lab, but in the community?

In Montana, we have been exploring what happens when computer science and AI learning begin not with the technology itself, but with the learners and communities who will use it.

The Center for Applied Special Technology (CAST, **cast.org**) is an educational research and development organisation that originally created, and continues to lead, the Universal Design for Learning (UDL) framework. The AI Education Project (aiEDU, **aiedu.org**) is an organisation that focuses on K–12 AI literacy and AI readiness. Together, we worked with educators across rural Montana to explore how AI literacy could become relevant, accessible, and connected to the places where young people live and learn.

Rather than asking educators to become computer science experts overnight, we focused on creating approachable ways for them to experiment with AI and computing and connect these technologies to learning experiences they were already providing.

This matters because expanding computer science in rural communities doesn't necessarily mean starting with more computer science courses. It can begin by helping educators recognise the computing that is already embedded in the learning experiences, problems, industries, and places around them, and giving them

the confidence and tools they need to build from there.

Rural computer science education should not simply be a smaller version of what happens in a well-resourced urban or suburban computer science classroom. Rural communities bring their own challenges, cultures, and ways of knowing. Agriculture, natural resources, healthcare, advanced manufacturing, outdoor recreation, and community infrastructure can all provide authentic contexts for computational thinking.

The question then shifts from 'How do we bring computer science to rural learners?' to 'How can learners use computing to create, understand, and solve problems that matter in their communities?'

Our experience with our AI in the Big Sky project has helped us identify five places where educators can begin.



1. Start with the people, not the technology

When a new technology emerges, it can be tempting to begin with the tool: How can we get AI into classrooms? How can we teach students to code? What physical computing devices should we buy? Instead, start by asking, Who are our learners and educators? What do they need in order to be successful?

This was particularly important at AI in the Big Sky. Many participating educators did not enter the project identifying as computer science or AI educators. Nearly 80 percent had never taught students about AI, even though many already believed that understanding AI would be important for the young people they served. "Adults and students both need time to explore, experiment with, and learn about new and emerging technology, like AI. This time to explore is really vital for building confidence in using and teaching about technology," said Connor Mulvaney, local AI educator and now National Programs Lead at aiEDU.

Many of the educators we worked with at AI in the Big Sky were from the Boys & Girls Clubs, and are preservice teachers or young adults who are interested in becoming teachers. They may not have formal training in pedagogy yet, but they bring other important skills and experiences to their work. They know the young people they serve. They build relationships, recognise when something is not working, find ways to keep young people engaged, and genuinely care about creating meaningful experiences for them.

Starting with what educators already know and do well, while giving them tools to strengthen their practice, can create better entry points into computer science for educators and learners alike.

> "HOW CAN LEARNERS USE COMPUTING TO UNDERSTAND AND SOLVE PROBLEMS THAT MATTER IN THEIR COMMUNITIES?

2. Build from what is already working

Not every learning experience needs

WHAT IS UNIVERSAL DESIGN FOR LEARNING (UDL)?

UDL is a framework for designing learning experiences which recognises that learners are all different. Learners bring different strengths, interests, experiences, backgrounds, and support needs to any learning environment.

Instead of designing for an 'average' learner and making adjustments later, UDL encourages educators to anticipate this variability from the start. This means thinking about different ways to engage learners, make information accessible, support learning, and provide options for learners to demonstrate what they know.

In computer science and physical computing, UDL might mean providing written and visual directions, modelling a task before learners try it, offering different roles during a group project, or giving learners multiple ways to explain what they created.

The goal is not to make learning easier, but to remove unnecessary barriers so that more learners can participate in meaningful and challenging learning.

to become an AI lesson, or a computer science lesson. Instead, look at what educators and learners are already doing and ask, Where could computing meaningfully extend this experience?

Hilary Lozar, STEM Coordinator at Flathead Boys & Girls Club (**flatheadbgc.org**), told us, "We used familiar, hands-on activities as entry points for exploring new technologies. Educators experimented with drones, coding, light-up cards, and AI-supported activities. The technology wasn't introduced simply because it was new. It became another tool for creating, investigating, problem-solving, and extending existing learning experiences."

This approach can be particularly valuable in rural communities because computer science can connect naturally to the work already happening around learners. Sensors can help investigate environmental conditions. Drones can connect coding with agriculture, mapping, wildfire response, or natural-resource management. Data collection can help learners explore questions about their own communities. Start with something learners already care about, then ask what computing makes possible.

3. Make computing tangible

Computer science can feel abstract when learners encounter it primarily through a screen. Physical computing changes that relationship. Drones fly. Lights illuminate. Sensors detect changes in the environment. Motors move. A piece of code suddenly produces an observable result in the physical world. These experiences create immediate feedback that lets the learner know something happened because of what they created.

"For many of our young people, technology becomes much more meaningful when they can hold it, build with it, and see what it can do. When a drone takes off or something they programmed actually works, computer science stops feeling like an abstract idea and becomes something they can imagine themselves doing," Hilary told us.

> "PHYSICAL COMPUTING RARELY WORKS PERFECTLY THE FIRST TIME. NEITHER DOES CODING. AND NEITHER DOES AI

They also create authentic opportunities for computational thinking. Learners make predictions, test ideas, encounter unexpected results, debug problems, make changes, and try again.

Physical computing can also provide multiple entry points into the same learning experience. One learner may be interested in writing the code; another may want to construct the device, document the process, troubleshoot the system, analyse the data, or explain the team's solution. Providing those different pathways is a practical application of UDL. The learning goal can remain rigorous and consistent while learners have different ways to engage, access information, contribute to the work, and demonstrate their understanding.

The goal is not to make computer science easier, but to remove unnecessary barriers so more learners can participate in the complex thinking, creativity, iteration, and problem-solving that make computer science valuable.

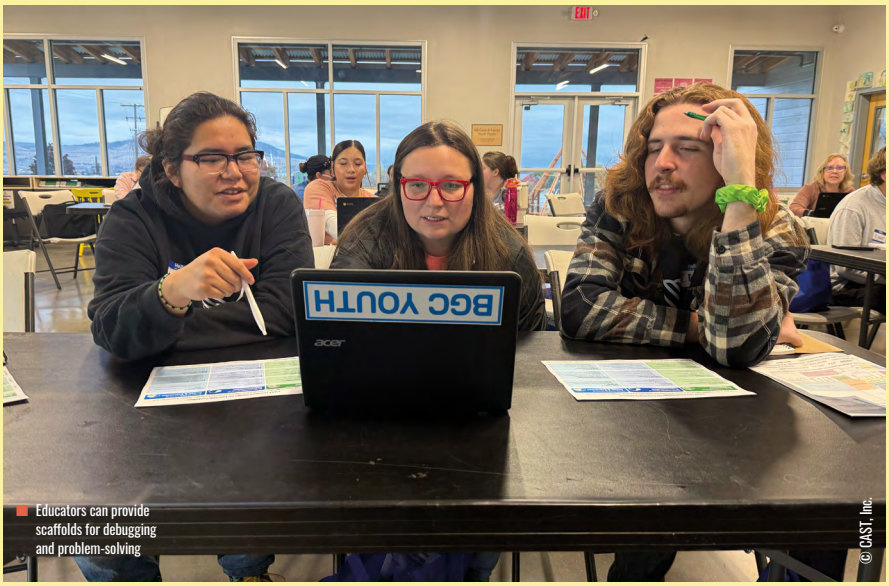
4. Design for variability from the start

Simply putting technology into a learning environment does not make that environment accessible. Physical computing can introduce its own barriers: unfamiliar technical vocabulary, complicated instructions, fine-motor demands, inaccessible interfaces, limited background knowledge, fear of making mistakes, or

LEARN MORE

Explore these free, downloadable resources designed to help educators integrate physical computing and AI into learning environments:

- **AI for Homework Help Protocol: Guidance on how to help youth use AI as a tutor, not a shortcut (helloworld.cc/ai-homework-help)**
- **Flying Orb Protocol: An activity to build teamwork, belonging, and STEM skills through drone-assisted play (helloworld.cc/flying-orbs)**
- **Light Up Pop-up Card: An introduction to electronic circuits and using AI to modify instruction (helloworld.cc/light-up-pop-up)**



Educators can provide scaffolds for debugging and problem-solving

© CAST, Inc.

simply the assumption that everyone should approach a problem in the same way.

UDL encourages educators to anticipate those barriers during the design process. That might mean pairing written directions with visuals or demonstrations. It could mean providing examples learners can examine before starting independently. Learners might work collaboratively at some points and independently at others. Educators can provide scaffolds for debugging and problem-solving and allow learners to explain what they created through writing, speaking, demonstrations, diagrams, video, or other forms of communication that align with the learning goal.

AI also adds new possibilities. Educators can use AI to help develop examples, adapt explanations, create locally relevant scenarios, support translation, troubleshoot code, or generate additional scaffolds. Learners may also use AI as a thinking partner as they brainstorm, code, revise, and problem-solve. But the technology should support the thinking — not replace it. The goal remains helping learners understand what they are doing, make decisions, test their ideas, evaluate results, and learn through the process.

> “THE GOAL IS NOT TO MAKE CS EASIER, BUT TO REMOVE UNNECESSARY BARRIERS TO PARTICIPATION

5. Give educators and learners permission to experiment

One of the most important things we learnt through AI in the Big Sky had less to do with any particular technology and more to do with confidence. Hilary reported, “Seeing the light of discovery on the faces of educators who were initially hesitant to jump into the AI Field Kits was incredibly powerful. It was a reminder that, no matter our age or experience, we all find joy in exploring, experimenting, and experiencing success with new technology.”

Educators talked about trial and error, patience, changing how they approached problems, and becoming less afraid of AI. Over the course of the project, they became more willing to use AI themselves and introduce it to young people.

That willingness to experiment matters, because physical computing rarely works perfectly the first time. Neither does coding. And neither does AI.

Something doesn’t fly. The light doesn't turn on. The sensor returns an unexpected value. The code produces the wrong result. The AI provides an answer that doesn't make sense. Those moments are not interruptions to the learning experience – they are the learning experience.

Computational thinking requires learners to break problems apart, test possible solutions, analyse what happened, revise their approach, and try again. Creating environments in which experimentation is expected can help educators model the same mindset we want learners to develop: Trial and error isn’t failure. It’s computational thinking.

Start local

The future of CS education isn’t about putting more devices into classrooms, or requiring every school to offer more computer science courses. It’s about expanding where computing happens, whose expertise is valued, and who gets to see themselves as someone capable of using technology to solve problems. For rural communities, that means beginning with the tremendous assets already there: local knowledge, community relationships, and the ingenuity used every day to solve problems.

Physical computing gives learners a way to connect those assets to technology. AI creates new ways to support educators and learners as they design, code, create, and troubleshoot. UDL helps ensure those opportunities are designed from the start in a way that includes the whole range of people who will participate in them.

We don’t have to wait for every rural school to have a computer science lab, or for every educator to become an AI expert. We can start with the people, with the problems they care about, and with the communities they call home. (HW)





AMANDA BASTONI

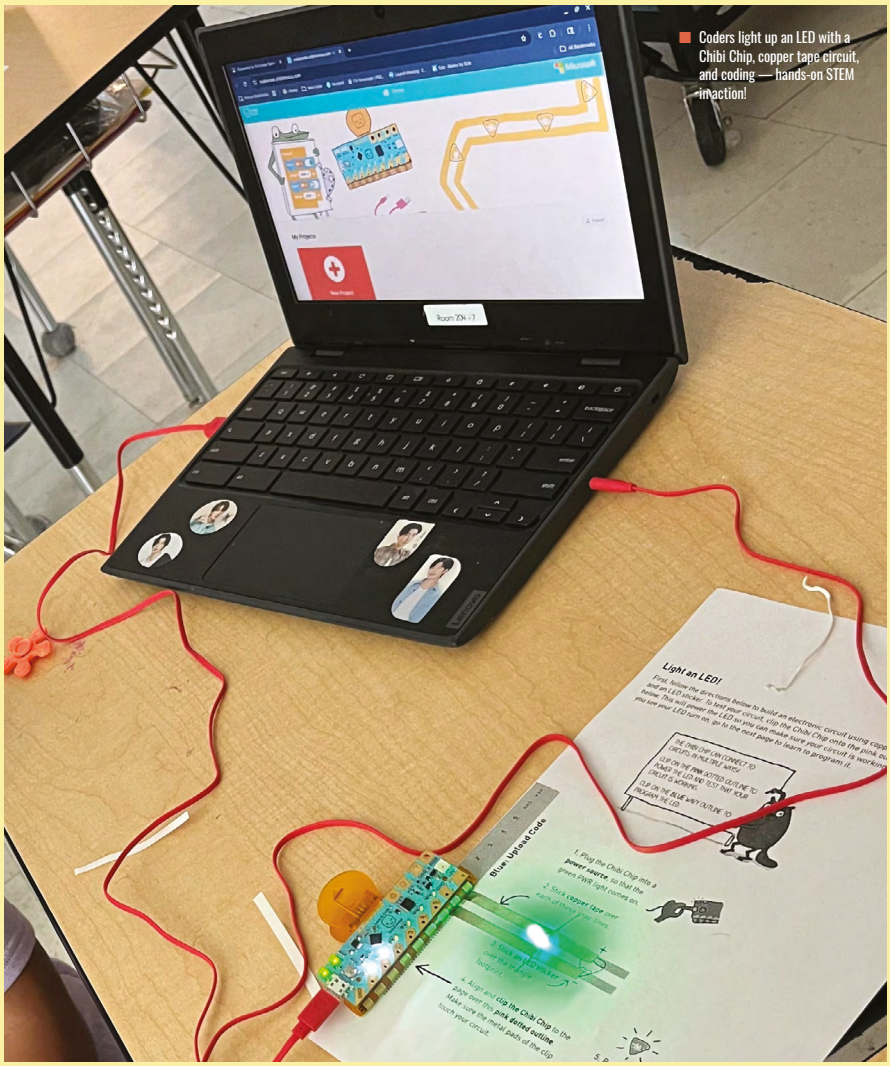
Dr Amanda Bastoni is senior director of postsecondary and workforce implementation at CAST. With over 20 years of experience in education and research, she focuses on Universal Design for Learning, career and technical education, and expanding access to STEM learning opportunities, particularly in rural communities.

HOW A COMIC BOOK LED ME TO PHYSICAL COMPUTING

Trakeisha Howard introduces Chibitronics to help get students hooked on physical computing

The first time I pressed a button stitched into my own comic book and heard it play back, 'You win again, technology', I was drawn into the possibilities of creativity within physical computing — the practice of building hardware that senses and responds to the physical world through buttons, sensors, and lights.

I had created an original Ratchet & Clank comic book in Microsoft Publisher and taught myself Arduino's web-based IDE so I could program an Arduino board. I then sewed conductive thread through the pages by hand to wire it all together, along with conductive buttons. I programmed the buttons to give a sound output for some of the catchphrases and sound effects of the game. Some of the buttons were programmed to say, 'I'm glad you're back, old friend', a blaster sound, and the chime of collecting gears.



Coders light up an LED with a Chibi Chip, copper tape circuit, and coding — hands-on STEM inaction!

That comic was my college capstone project, but because I was so engaged, it never felt like an assignment. After years of programming in Java, assignments, and readings, this project stuck with me because it let me start from something I actually wanted to play and then work backwards into the physical and software engineering to make it real. That stuck with me as an educator: play gets students to lean in, in a way that a worksheet rarely does — and this is why the micro:bit is such a hit in classrooms.

But there's a tool that works well as a lead-up to micro:bit, and most teachers haven't heard of it: the Chibi Chip. This is one of the products from a company called Chibitronics. The Chibi Chip is the paper-friendly programmable microcontroller behind Chibitronics' Love to Code system. It is programmable through a MakeCode-based editor, and fills a gap that is easy to miss when a curriculum jumps straight from 'no circuits at all' to 'here's a full microcontroller board'.

What is Chibitronics?

Chibitronics (**chibitronics.com**) make copper tape peel-and-stick LED stickers, and the Chibi Chip, a paper-friendly programmable

microcontroller, that turns arts-and-crafts projects into working electronics. Physical computing can simply mean students folding or drawing a circuit path onto paper, sticking copper tape along it, connecting a battery, and making the LED light up when the circuit is complete. No software to install, no wiring to get wrong, just a closed loop and an immediate result. This is a version of physical computing that works even for classrooms that do not have laptops for every student, which is more common than curriculum planning sometimes assumes.

The Chibi Chip adds a bit of rigour and excitement. It drops into that same paper circuit, same tape, same battery, same LEDs, but students now code the Chibi Chip using **makecode.chibitronics.com**. Students can code the Chibi Chip to control how the lights behave: blinking, fading, and responding to touch. By the time they are ready for a micro:bit, they have already handled the two hardest parts of physical computing — building a working circuit and writing the code that controls it. And since Microsoft MakeCode is also the coding environment for the micro:bit, nothing about the transition feels like starting over.

Building the progression

I teach this as a staircase: when you skip a physical step, you may stumble or fall, and no step works without the one before it. Chibitronics paper circuit materials come first, as even young students can complete a working circuit in minutes, with no code required. The Chibi Chip comes next, once students trust their circuit enough to start writing code for it. Once they are prepared for the basics of physical computing using the paper circuits, I introduce the micro:bit. This is a tiny programmable computer about the size of a credit card, made by the Micro:bit Educational Foundation, designed specifically for teaching coding and physical computing in schools. Each stage builds confidence for the next one. Students who have struggled to get an LED to light up on paper are more patient debugging code later, because they have already learnt that circuits fail quietly, and that checking your connections comes before blaming the software.

My favourite way to run the early stages is by making a light-up Christmas card. First I use a circuit sketcher, which is a free online tool for planning and printing paper circuits.



TRAKEISHA HOWARD

Trakeisha is a licensed CS educator in Tennessee, USA, and a K–12 instructor at CodeCrew, teaching computer science and other core subjects since 2018. She holds a CS degree from LeMoyne-Owen College and a master's in education from Rhodes College.

> WHEN YOU SKIP A PHYSICAL STEP, YOU MAY STUMBLE OR FALL, AND NO STEP WORKS WITHOUT THE ONE BEFORE IT

Most students struggle when their copper tape crosses over itself and the lights stop working, so now they have to plan their path first before laying down the circuit tape. It's a small constraint, but it gets them thinking like engineers rather than just decorating. Once the circuit works, they wire in the Chibi Chip and code the lights to blink, fade, or twinkle, then take the finished card home to share with family. I still remember the look on one student's face the first time her tree lit up — it was the exact look I imagine I had on my face the moment my hand-built comic book worked for the first time.

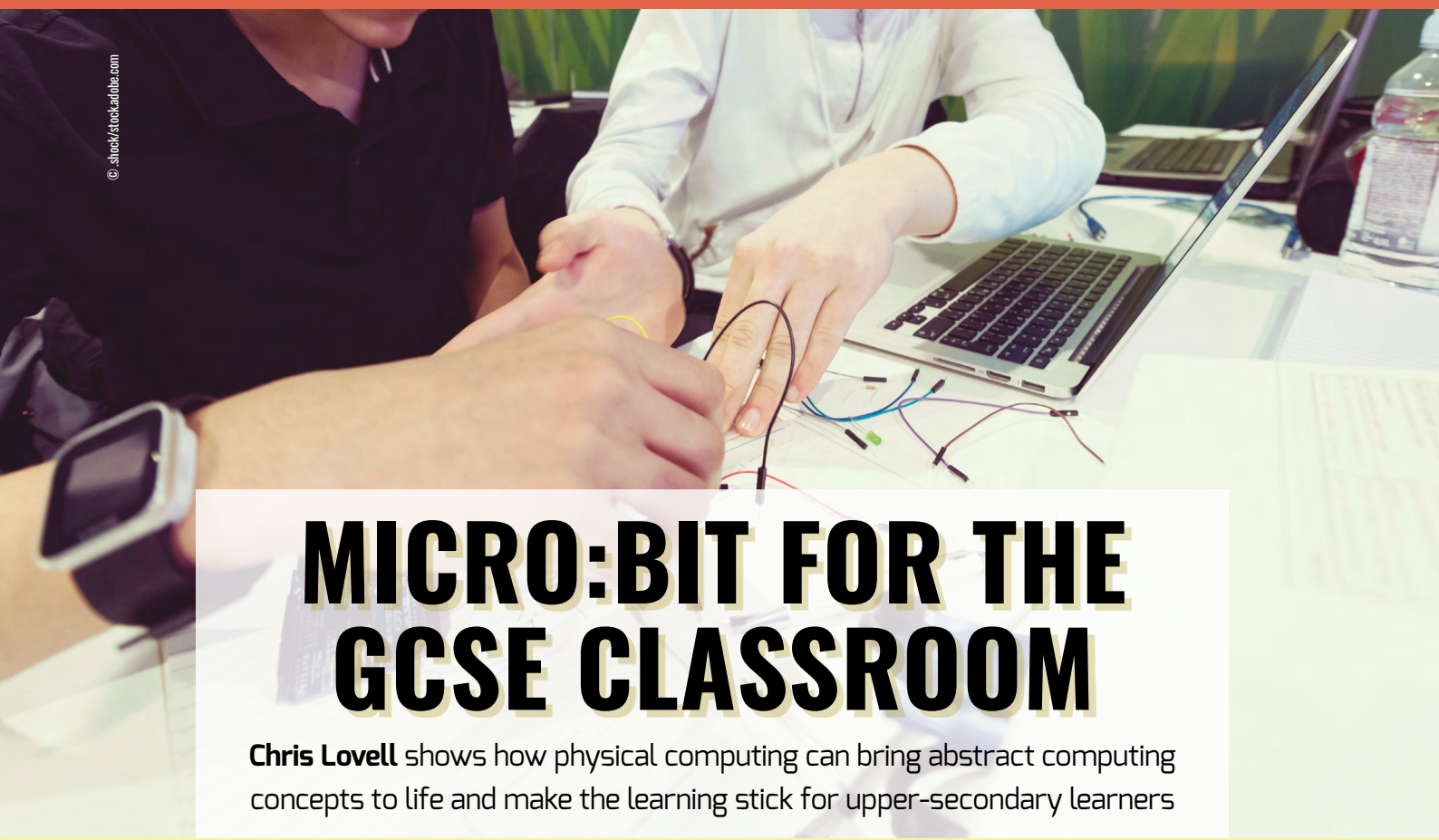
Play, AI, and access

Once students have that circuits-and-code confidence, the next step is to introduce AI. The free micro:bit CreateAI tool (**createai.microbit.org**) lets students collect their own movement data from the micro:bit's accelerometer, train a simple model to recognise gestures like a wave, and run it directly on the micro: bit. It turns machine learning from a buzzword into something students build and test with their own hands, mistakes included, and when the model gets a gesture wrong, that is often a more useful conversation than when it works perfectly the first time. None of this needs a fully stocked classroom, either. You can start with nothing more than copper tape, an LED, a coin-cell battery, and paper. There are plenty of paper-circuit activities built around exactly that combination, and it costs very little to try one.

If even those materials are hard to come by, it's worth checking whether there is a local library system, maker space, or educator resource centre that lends kits. Structure has its place in a CS classroom, and students do need time to practise syntax and problem-solving. But if you want students who genuinely care about loops, conditionals, and code, give them something worth playing with first, and let the learning follow. (HW)

RESOURCES TO EXPLORE

Chibitronics (chibitronics.com) offers copper tape, LED stickers, and the Chibi Chip microcontroller. Circuit Sketcher (sketcher.chibitronics.com) is a free tool for planning paper circuits before printing. Microsoft MakeCode (makecode.com) powers both the Chibi Chip and the micro:bit with block-based coding. MakeCode Arcade (arcade.makecode.com) turns micro:bit projects into playable retro-style games, while micro:bit CreateAI (createai.microbit.org) lets students train simple gesture-recognition models.



© shock/stock.adobe.com

MICRO:BIT FOR THE GCSE CLASSROOM

Chris Lovell shows how physical computing can bring abstract computing concepts to life and make the learning stick for upper-secondary learners

The micro:bit has been a classroom staple for over a decade. My students (aged 8–14) use the device to build everything from Space Invaders-inspired games to step counters and wearable projects, and we all know how well it engages them. But can it earn its place with upper-secondary learners preparing for exams by helping them retain the abstract concepts of the GCSE computer science specification? (GCSE is the qualification taken by students aged 14 to 16 in the UK, and this article will reference the AQA GCSE Computer Science specification, helloworld.cc/AQA-CS.) Here's what I've learnt about physical computing at upper secondary level.



```
pet.py

from microbit import *
import audio

timer = 0
display.show(Image(
    "00000:"
    "09090:"
    "00000:"
    "99999:"
    "00000"))
audio.play(Sound.HELLO)

while True:
    if pin_logo.is_touched():
        timer = 0
        display.show(Image.HAPPY)
        audio.play(Sound.HAPPY)
    elif accelerometer.was_gesture('shake'):
        timer = 0
        display.show(Image.SURPRISED)
        audio.play(Sound.GIGGLE)
    else:
        sleep(500)
        timer += 0.5
        # sleep for half a second only to make it react more quickly to logo touch &
shake

    if timer == 20:
        display.show(Image.SAD)
        audio.play(Sound.SAD)
    elif timer == 30:
        display.show(Image.ASLEEP)
        audio.play(Sound.YAWN)
    elif timer == 40:
        display.show(Image.SKULL)
        audio.play(Sound.MYSTERIOUS)
        break
```

Figure 1 Extract from the pet project code

To understand the needs of computing teachers, I asked my teacher connections on social media how they use physical computing at upper-secondary level. 42 percent said it was core to their curriculum, clear evidence that the micro:bit can work at GCSE. But 31 percent barely use it, and 27 percent want to use it but don't know how. Those latter two groups represent the real opportunity, and this article is for them.

The micro:bit is a palm-sized computer designed specifically for the classroom. It includes a 5×5 LED display, two programmable buttons, a speaker, a microphone, motion sensors, and radio communication, and it is powered by USB or batteries. For upper-secondary lessons, I use the Python editor (**python.microbit.org**), in which learners write Python code in the browser before 'flashing' it to the micro:bit. To manage a class, I use the micro:bit classroom tool (**classroom.microbit.org**), which you can use to push code and examples out to learners and review their work as they go. For planning, the Micro:bit Educational Foundation (**microbit.org**) hosts a library of lesson ideas, videos, handouts, and wall displays for educators.

The micro:bit pet project

I have found that by making a pet with the micro:bit (**helloworld.cc/pet-project**, **Figure 1**), learners create a tangible artefact while also covering a significant proportion of the AQA GCSE computer science specification. Learners code the pet to respond to inputs: the capacitive gold logo tickles it, while a shake surprises it. To implement their ideas, learners must use an 'if–elif–else' construct. The project may be extended through functions, giving learners a natural

introduction to decomposition. Variables hold the pet's hunger and happiness, updated with each interaction and checked by a 'while true' loop that keeps the pet listening for input rather than stopping after the first event.

With the micro:bit, these concepts are made visible. A learner does not have to trust that a variable changes; they watch the pet's display alter as its happiness rises and falls. The abstract becomes concrete, and physical computing tasks become their own reward. I have learnt that some learners get lost in tweaking their pet without ever making the connection to the exam specification. To make that connection secure, I pause mid-lesson for a quick knowledge check, point to the specification statements, and ask learners to explain in plain words which construct their code uses and why. I have found that the micro:bit makes the ideas tangible, and I then recall the exam specification so the knowledge sticks.

> WITH THE MICRO:BIT, ABSTRACT EXAM-LEVEL CONCEPTS ARE MADE VISIBILE

The bridge to exam-level Python

A question teachers often ask me is whether MicroPython on the micro:bit is 'exam-level' Python. The honest answer is that the language is the same. According to micro:bit's own documentation, MicroPython is a full implementation of Python 3 designed to run on microcontrollers (**helloworld.cc/micropython**). The constructs the AQA exam assesses (variables, selection, iteration, functions) behave identically on the micro:bit and on a desktop. What differs is the outer layer: the standard library is a subset, and there is a micro:bit module that allows access to the hardware, such as `display.show()` function and reading the buttons. Those device-specific lines won't appear in an exam.

I never use the micro:bit as a replacement for exam practice, but as a bridge to it. I ask students to take a working pet program and translate it into exam-standard Python, then into AQA-style pseudocode. I have found that the act of translating their code is what makes those core ideas stick with learners.

Beyond programming

I have found that the micro:bit earns its place beyond programming. In cybersecurity lessons I use a password-checker project (**helloworld.cc/password-project, Figure 2**): students build the code themselves, storing a secret password in a list and comparing it, element by element, against the one they enter with the buttons. The task often crops up in the AQA exam as both a programming question and, as part of Paper 2, a question about what makes a secure password. The device gives the cybersecurity strand a physical home while allowing learners to practise the array traversal that examiners reward.

Data representation may also be explored with the micro:bit. The 5×5 display is a bitmap: each LED is a bit, lit or unlit, which is exactly the 'convert a bitmap into binary' question the AQA specification includes. And teachers have told me they use the micro:bit's radio communication to teach networking too, exchanging data wirelessly between devices.

Getting started

If you are new to the micro:bit, you don't need to begin with the pet or the password-checker. A single micro:bit and a willingness to tinker is enough to start, and a classroom set is a bonus, not a requirement. The editor at **python.microbit.org** runs entirely in the browser, so learners write and flash real Python with nothing to install, and the built-in simulator means a lesson can run even where devices are scarce.

I recommend starting small with one short task that works first time, then building up. The Micro:bit Educational Foundation's resources are a reliable scaffold, and a vibrant community shares ideas freely online. What matters is matching the device to where your students are, not covering every feature.

Across programming, cybersecurity, and data representation, the micro:bit has earned its place in the GCSE classroom. It complements the activities that support learning GCSE computer science content, making abstract concepts concrete and helping knowledge stick. (HW)

Figure 2 Extract from the password checker code

```
from microbit import *

# The secret password: a sequence of A and B presses
SECRET = ["A", "B", "B", "A"]

# What the learner has entered so far (incomplete login attempt)
entered = []

def add_press(letter):
    """Add a button press to the attempt and show it."""
    entered.append(letter)          # store it in the list
    display.show(letter.lower())    # small feedback: a or b
    sleep(400)

def check_login():
    """Compare the attempt to the secret, element by element."""
    if len(entered) != len(SECRET):
        display.show(Image.NO)
        sleep(500)
        entered.clear()             # reset the attempt
        return

    match = True
    for i in range(len(SECRET)):
        if entered[i] != SECRET[i]: # compare one position at a time
            match = False
            break

    if match:
        display.show(Image.YES)     # login succeeded
    else:
        display.show(Image.NO)      # login failed

    sleep(800)
    entered.clear()                 # reset for the next attempt

# Show the secret (for the teacher's demo) on startup
display.scroll("LOGIN")

while True:
    if button_a.was_pressed():
        add_press("A")

    if button_b.was_pressed():
        add_press("B")

    # Shake to submit the password
    if accelerometer.was_gesture("shake"):
        check_login()
```



CHRIS LOVELL

Chris is a secondary-school computing teacher and a micro:bit champion.

THINKING GLOBALLY, ACTING LOCALLY WITH PHYSICAL COMPUTING

Tonya Coats and **Omar Shepherd** share how combining physical computing with state park partnerships creates a scalable, interdisciplinary framework for environmental data science

Computer science does not exist in a vacuum, yet it is frequently taught as an isolated subject. To keep students engaged, we need to teach code as a tool for human impact, not just a technical skill. Our team at the San Bernardino County Superintendent of Schools (SBCSS) designed a hands-on learning model that blends physical computing and environmental science through design thinking. Funded by the California Math, Science, and Computer Science (CAL-MSCS) grant, this model of professional learning demonstrates how local geographic resources can be leveraged to address the United Nations Sustainable Development Goals. More importantly, by partnering with the California State Parks Online Resources for Teachers and Students (PORTS) programme, we have established a model that can be easily adapted by educators everywhere.

Accessibility also shaped our tool choices. We wanted educators to leave with ideas they could realistically bring back to their classrooms, so we focused on common classroom supplies and low-cost, classroom-friendly tools such as micro:bits. This mattered because equitable computer science implementation depends on more than student interest. Teachers also need tools that are affordable, adaptable, and manageable within real school settings.

Parks as classrooms

Our journey began with a simple question: How can we move beyond traditional, screen-based coding exercises and turn environmental science into a hands-on, minds-on, high-tech adventure? The answer can be found in physical computing tools such as micro:bits, which encourage students to design solutions that solve real-world problems through coding.

For us, it was important to bring the outdoors in. Not every teacher or student has easy access to a state park, nature preserve, or outdoor learning space. Through the PORTS programme and our station-rotation model, we created an experience in which teachers could connect with California's natural and cultural resources without leaving the room. This helped educators see how outdoor learning can happen through virtual field trips, storytelling, physical computing, and local environmental investigations. Through a live virtual session with California State Parks interpreters, educators experience how natural and cultural sites can become powerful entry points for student learning. Through this experience, teachers were given a model for bringing meaningful virtual learning experiences to students who might never visit those locations in person.

From there, educators rotated through a series of stations, each inspired by a California state park and connected to an environmental principle and concept. At the Anza-Borrego Station: Soil Quest, teachers



© Image courtesy of SBCSS Digital Learning Services

Educators collaborating at a state park-themed learning station

THE PORTS PROGRAMME

The Parks Online Resources for Teachers and Students programme (**helloworld.cc/ports-program**), known as PORTS, is a distance-learning initiative from California State Parks. Through free, live, interactive virtual field trips led by park interpreters, PORTS expands access to natural and cultural resources, giving students from a wide range of geographic and socioeconomic backgrounds opportunities to engage with environmental science, history, and climate literacy.



© Image courtesy of SBCSS Digital Learning Services

Connecting educators with California's natural and cultural resources to see how outdoor learning can happen virtually

> HOW CAN WE TURN ENVIRONMENTAL SCIENCE INTO A HIGH-TECH ADVENTURE?

created moisture sensors to investigate land degradation and soil conditions. At the Silverwood Lake Station: Water Lab, they programmed temperature alerts to monitor aquatic ecosystems. At the Wildwood Canyon Station: Habitat Watch, educators created temperature sensors to study different habitat zones and determine which areas were safest and healthiest for wildlife. Each station positioned technology as a tool for investigation.

By exploring data collected across stations, teachers bridged the gap between physical computing and environmental science. They didn't just observe; they adapted the code to analyse real-world environmental conditions. This hands-on experience empowered them to apply these concepts directly to designing sustainable systems and developing real-world solutions.

The global–local lens

At the heart of the workshop was the connection between global sustainability goals and local environments, encouraging educators to 'think globally, act locally'. By anchoring each station to a specific UN Sustainable Development Goal (such as Clean Water and Sanitation, or Life on Land), alongside a California Environmental Principle and Concept, the tasks became more relevant.

Teachers began to see that the environmental challenges faced locally are connected to issues happening around the world. This allowed educators to see coding as a useful skill for solving problems both in their own communities and beyond.

This interdisciplinary model provides an accessible entry point for students of all backgrounds to shift their perspective of computer science from an intimidating, abstract discipline to a powerful, creative toolkit for social and environmental good.

An international blueprint

For the workshop to have lasting impact, teachers needed to leave with more than ideas. They needed time to turn what they experienced into something they could use with students.

During the closing reflection, educators worked in cross-district teams to adapt the station activities into classroom-ready lesson plans. Using a design thinking lesson template, they identified student needs, connected the activities to standards, and considered how physical computing could support learning in their own classrooms.

These lesson drafts were gathered into an open-access digital library so educators could download, adapt, and build on one another's work. This shared resource helped extend the learning beyond a single workshop and gave teachers practical examples they could bring back to their schools.

The model can be adapted in many communities. Educators might partner with a state park, a local conservation group, a community garden, or an urban green space. The process stays the same: identify a local environmental resource, connect it to a global sustainability goal, pair it with accessible physical computing tools, and guide students toward meaningful problem-solving. (HW)



OMAR SHEPHERD AND TONYA COATS

Tonya and Omar are digital learning and computer science project specialists at San Bernardino County Superintendent of Schools (SBCSS), dedicated to expanding equitable CS access and interdisciplinary AI/STEM integration statewide through the CAL-MSCS grant.





BREAKING DOWN BARRIERS

Professional development for physical computing confidence

When it comes to teaching physical computing, many computing educators are hampered by a lack of devices and hardware — but even if these are available, they are often soon gathering dust in the back of someone's classroom because implementing new initiatives takes knowledge, resources, and the sustained time to apply them.

In India, the Atal Tinkering Labs (ATL) programme was set up by the Atal Innovation Mission (AIM) to give school students across India hands-on access to tools like basic electronics, 3D printers, and robotics. The goal of the project is to let young people experiment and develop their creativity and problem-solving skills. Each lab is a space where a student can turn a curious idea into a working prototype, and in doing so, build the confidence to tackle real-world problems.

As of today, 10,000 labs have been set up across the country, spanning both government and private schools, and more than 10 million students are involved. But a lab full of equipment is not the same as a lab full of learning. Many schools received devices, wiring kits, and microcontrollers, but only a small number of them are now fully functional. The equipment sits unused, not because students lack curiosity, but because the teachers responsible for the labs often lack the confidence and technical knowledge needed to introduce physical computing to their classrooms.

So if the ATL programme comprehensively removed one access barrier by providing devices, how could we work with educators to develop their knowledge and support them to apply it in their classrooms?

Why one-off training wasn't the answer

We know from research, theories of educator development, and our work in schools and with teachers that a one-off training day does not embed changes in practice. A single day outside the classroom, with no context and limited practical application, means there is limited knowledge transfer and almost no chance for educators to apply their learning in their own setting.

Using our professional development principles, we knew that we needed to design learning that was:

- Sustained — so that learning is revisited as they begin to apply it in their classrooms and cascade it to their peers
- Collaborative — teachers are experts, and working with colleagues in a low-stakes environment allows for knowledge sharing and experimentation
- Adaptable to context — making sure that we worked with the exact devices that teachers would be using in their schools

We decided on a hybrid approach, designed to allow in-person hands-on learning as well as the ongoing revisiting of knowledge, and with examples that would allow teachers to continue to learn as they returned to their classrooms. Through this combined method, we hoped to see an increase in teacher confidence, which we know is a significant barrier that needs to be addressed before teachers change their practice.

A hybrid approach

In India, we have partnered with the Bhubaneswar City Knowledge Innovation Cluster Foundation (BCKIC) and the Indian Institute of Information Technology and Management – Kerala (IIITM–K) to deliver teacher training on the ground. BCKIC and IIITM–K are the official ATL Sarthi partners working in collaboration with the Atal Innovation Mission (AIM), NITI Aayog. Through this initiative, they support Atal Tinkering Labs by strengthening their



For the in-person part of the hybrid model, participants were trained on basic circuit design, design thinking, and using Raspberry Pi computers

operations, building educators' capacity, and fostering innovation among young people.

We began with an in-person day of training using the ATL devices. Our most recent session took place at KIIT International School in Bhubaneswar, where 21 schools took part, each sending one teacher and one student, providing some exciting ways to cascade this learning back to schools. In this in-person session, participants were trained on basic circuit design, design thinking, and using Raspberry Pi devices.

Before arriving at the session, teachers had access to the first of four online courses specifically designed for this programme, an introduction to physical computing and electronic circuits aimed at beginners. This course can then be cascaded in schools, with videos available to show colleagues (and students) how to connect a breadboard.

We will soon release a further three courses, all designed to suit the needs of teachers running a Tinkering Lab in their schools. These will address how to use an Arduino device working with C++, how to use a Raspberry Pi with Scratch, and the pedagogy of project-based learning and how it can be used to support physical computing.

As there was limited time to release teachers for external training, and given that an expert can't be in the room with a teacher as they deliver their first physical computing lesson, online courses that can be revisited at any time can be an ideal means of offering the expert guidance and modelling a teacher needs.

What we have learnt

We will soon be entering the next phase of the rollout, which will extend training to more schools and partner institutions and continue to pair the Training Hub's self-paced courses with in-person sessions on the ground.

While it's still early days, initial signs suggest that teachers are already feeling more confident that they can plan physical computing lessons, work with the devices, and demonstrate their use to learners. Our team in India is looking forward to seeing the impact of the project in classrooms, where devices are no longer gathering dust, but are now enabling students to experiment, turning a curious idea into a working prototype, and in doing so, building the confidence they need to tackle real-world problems.

Beyond India

Although we designed this training specifically for Indian educators, the lessons learnt are relevant across many areas of the world:

- Educators need time to learn from an expert, and with peers, in a low-stakes environment before they can introduce physical computing into their classrooms
- Teachers build confidence by gaining knowledge which they can then replicate and repeat in their own teaching, and a hybrid approach means teachers have support once they go back to their schools

If there are unused physical computing devices gathering dust in a cupboard in your setting, what would it take for you to bring together a few colleagues and use our online courses to start building your knowledge and confidence?

There are more than 20 professional development courses for you to explore on the Training Hub: **helloworld.cc/training-hub**. (HW)



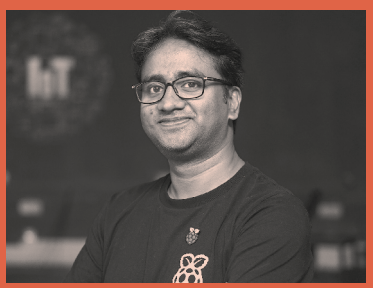
EMMA KING

Emma is the senior online courses manager at the Raspberry Pi Foundation. She oversees the Training Hub which has a range of courses to support educators in classrooms, clubs, and all settings where adults help young people learn computing!



MAMTA MANAKTALA

Mamta is deputy country director and senior learning manager at the Raspberry Pi Foundation in India. A former computer science teacher, she leads curriculum development and teacher professional development across India.



RAJIB SARKAR

Rajib is an accomplished educationalist with extensive experience in teaching and curriculum development across prestigious institutions and organisations. Specialising in engaging, learner-centric programme design, he is dedicated to transforming education.



If we want students to develop and exercise their critical-thinking skills, what does that mean for how we use AI professionally?

© Rawpixel.com/stock.adobe.com

SURVIVING THE EDUCATOR AI PRESSURE COOKER

It's OK to pause the AI adoption mania and consider what's truly best for your students

When a new educational technology tool comes along, there are frequently some early adopters who choose to test the tool in their classrooms immediately — even when there may be no evidence that the tool improves student learning. This isn't necessarily bad; teachers can often rely on their experiences and training to determine whether or not a particular tool is a good fit for the classroom.

Tool developers, especially those from for-profit companies, may have their own motivations for increasing their presence in your classroom. Profit, whether it's measured in dollars, data sets, or devotees, may at times be at odds with our classroom goals. And tools that have no or limited evidence of increased learning, or those which may even show evidence of decreased learning, seem to have become more pervasive.

Even Sal Khan, founder of Khan Academy, knows this. In a recent post on LinkedIn, Khan wrote, "Almost every ed tech vendor says its research shows the product works. Some of that research is strong. Much of it isn't. Decision makers need to know the difference, because a tool that hasn't met a real evidence bar shouldn't receive public money or classroom time."

And yet, in 2026, teachers all over the world are being asked to incorporate AI in their classrooms — even when the "real evidence bar" has not been met. You can't turn anywhere today without being inundated with AI products and being told to start using AI for the sake of the kids. After all, they will need to use AI if they are to be great employees later.

AI use by teachers

It's not just that AI tools have proliferated across nearly every facet of our lives; it's also that we are constantly coming across messaging telling us to use them. We're told over and over again that we should be using AI, and that our students need to be taught how to use AI. Recently, I heard teachers describe how they use AI to write parent emails. One teacher described how she preserves her voice by simply asking the AI to write it in such a way that it

REDUCED PERSISTENCE AND INDEPENDENCE

Recent research found that even one 10–15-minute session using AI may reduce your ability to perform similar tasks independently and decrease your persistence. What does that mean for students, who are still learning?

sounds like her.

As I tried to unpack this, I was left with many questions: Do you want to give AI your 'voice' so that it can emulate you? What are the implications of doing so, particularly with AI agents already pretending to be people through emails? Do your students' parents want to read AI-written emails rather than teacher-written emails? When should AI use be disclosed (in this situation, for example, to the recipients of the email)? What happens when parents respond with AI-written emails, and there is a back-and-forth conversation led by AI? Does this benefit your students? How will AI tools use the data they have collected, either about the parent, the student, or you? More critically, do we want to turn teacher-parent interactions over to a machine?

> "IT TAKES TIME TO SHAPE OUR VALUES AROUND AI TOOLS AND DECIDE WHETHER, WHEN, AND HOW TO BRING THEM INTO THE CLASSROOM

What the evidence shows

In a hot-off-the-press article, researchers found that adults using AI tools for tasks hinders independent performance and decreases persistence (**helloworld.cc/ai-independence**). They state that "Although AI assistance improves performance initially, people's performance drops sharply once AI is removed. More strikingly, participants in the AI condition also give up more frequently on the tasks compared to control participants. This pattern is robust and replicates across domains. These effects emerge after a 10–15 minute session, raising the question of what prolonged AI use does to human capability over time."

If even one (ONE!) 10–15 minute session using AI can reduce your ability to perform a task independently and decrease your persistence, imagine what that does to a student who uses AI frequently, over many years, as their learning habits are developing.

This evidence adds to the mountain of research showing that using AI tools can negatively impact knowledge, independent performance, and persistence. While it is also important to begin to ascertain the ways in which AI may help learning, the evidence for that remains less clear. Given the divergent voices by companies and researchers, it is challenging to know to whom we should be listening.

Weighing pros and cons

It takes time to shape our values with respect to AI tools and decide whether, when, and how to bring them into the classroom. Often, teachers are not given the luxury of the time needed to explore and shape their own values around AI.

We know that data centres are a huge drain on natural resources and contribute to climate change. We know that data privacy is a huge risk when using AI systems. We also now know that AI use contributes to our own de-skilling: the loss of the ability to conduct tasks that we once knew how to do on our own. Should those facts be considered before determining whether, when, and how you want to use AI tools and have students use them?

There are a myriad other reasons to be wary of mass AI adoption at this early stage, and to be cautious about the promotion of AI as a solution to students' learning challenges. As Natasha Singer notes in her new book *Coding Kids*, one of the key missions of public schools is to teach students "to think independently and question assumptions".

It's critical that we give educators like you the time to consider whether AI use promotes or hinders that mission, to weigh the underlying motivations of those selling or giving away their wares, and to determine how you want to approach and use AI, both personally and professionally in teaching students. (HW)

This article was written and reviewed by humans.



MONICA MCGILL

Monica is the founder and president of the Institute for Advancing Computing Education (**iacomputinged.org**). She leads a team dedicated to studying the impacts of computing (and AI) education on learners. She is a former computer science professor in the USA.

BEFORE THE FIRST LINE OF CODE

Why design thinking should come first

There's a moment I keep coming back to. It happened at a student showcase organised by the Welspun Foundation, at which young learners were presenting projects they had built in Scratch. The room was loud with that particular kind of pride kids have when they have made something — animated stories, quizzes, simulations, games — all of them the product of weeks of tinkering.

One project, a car parking simulator, stopped me. The student walked me through it with real enthusiasm: navigate a car through a lot, avoid the obstacles, park in the marked space. He pointed out the scoring system, the collision detection, the timer, how the difficulty ramped up. You could tell he had spent hours getting the details right. And the thing worked — it really did!

Then he grinned, stepped back, and handed me the keyboard. I had no idea what I was doing. Which keys moved the car? Did clipping the yellow line count as a crash? Was I supposed to park right away, or rack up points first? Every time I paused, he leaned in to help: "Reverse first", "Don't touch the yellow line", "Park inside the white box."

He wasn't trying to take over; he just wanted me to enjoy it. But I noticed that I wasn't learning the simulator from the simulator. I was learning it from him. The code worked, but I couldn't figure out how to play it.



GEETANJALI KHATRI

Geetanjali is a curriculum and content developer with eleven years of experience in K–12 education. Her work focuses on shaping meaningful learning experiences in computing and AI in India. Passionate about how young people learn and create, she loves seeing their ideas come to life.

When working code isn't enough

That one exchange altered how I think about computing education. For a long time I assumed that moments like this were just part of learning to code — the normal friction of a first build. I don't see it that way anymore. I think it points to something we've got backwards. We've got so good at teaching students how computers think that we've stopped teaching them how people think.

None of this is a knock on programming itself. It's still one of the best tools we have for building logical reasoning, creativity, and computational thinking (breaking a problem down, spotting patterns, turning an idea into something that runs). Those skills aren't going anywhere.

But outside the classroom, nobody judges an app by how clean its code is. People come back to technology because it solves something real, because it feels obvious to use, because it was built with them in mind, not because the variables were named well. You don't recommend a game to a friend because of its architecture.

And the strange thing is, the technology that gets this right almost never starts with code. It starts somewhere else entirely.

The gap in computing education

Think about how many computing lessons open: 'Today we're building a game.' Within ten minutes, kids are picking sprites and backdrops and writing logic. Almost nobody stops to ask who any of it is for, what problem it's actually solving, or how a stranger is supposed to figure out what to do the first time they touch it.

Product teams that take user-centred design seriously ask those questions before they build anything: they talk to users, sketch rough prototypes, watch people struggling with early versions. They have learnt that understanding people isn't something you tack on at the end —it's where you start.

In the classrooms I've watched, it's usually the opposite. Kids are encouraged to start building almost immediately, long before they've thought much about who is going to use what they make.

The kid with the parking simulator did exactly what most of us would have done in his position. He focused on making it work.



Writing feedback on student projects

Nobody had ever asked him to sit back and watch someone play it for the first time. That's not a gap in what he did. It's a gap in what we taught him.

Design thinking before the first line of code

This is more or less what design thinking is trying to fix, though the term gets flattened pretty quickly into a five-step diagram (Empathise, Define, Ideate, Prototype, Test) that people memorise and move past without really absorbing. Underneath the diagram, it's a much simpler idea. Before you build anything, ask who you are building it for (**helloworld.cc/design-thinking**).

Picture the same student inviting five people to try his simulator before calling it finished. He wouldn't have found more bugs. His code already worked fine. He would have found the gaps in the experience: where the instructions weren't clear, where a visual cue was missing, where the first level was harder than it needed to be. Not because someone handed him a list of features to add, but because he watched a real person get confused.

That's the actual difference between designing with people and just building for them, and it's not a new idea, even inside Scratch. Mitchel Resnick and the team at MIT Media Lab built Scratch around the Creative Learning Spiral, moving through Imagine, Create, Play, Share, and Reflect before starting the cycle again, because they wanted kids to improve their ideas by putting them in front of other people and hearing what happened (**helloworld.cc/creative-learning-spiral**). It's design thinking, essentially, just under a different name: make something, watch how people experience it, refine, repeat.



Students showcasing their inventions at a Welspun Foundation event

> MAKING SOMETHING WORK AND MAKING IT WORK FOR SOMEONE ELSE AREN'T THE SAME THING

Maybe the real learning doesn't happen when a student finishes their code. Maybe it happens the first time someone else uses it.

Why AI makes human empathy essential

I think AI raises the stakes on all this rather than lowering them. AI coding assistants can now turn a plain-language prompt into working code. They can explain concepts, catch bugs, or even build a first draft of an app. Writing the software is no longer the hard part. The hard part, the part AI can't do for you, is deciding what's worth building, why it matters, and who it's actually for. That takes empathy and observation and judgement, none of which show up in a prompt. If anything, AI makes design thinking more essential, not less.

I still wonder sometimes whether that student went back and reworked his simulator after the showcase. Maybe he added clearer instructions. Maybe he simplified the first level. Or maybe he just walked away having learnt something no lesson plan could have handed him: that making something work and making it work for someone else aren't the same thing.

He probably thought I was there to judge his programming. Really, he taught me something bigger: that teaching computing isn't just about teaching code. It's about teaching kids to notice people, question their assumptions, and understand who they are building for. Because the real work of innovation starts well before anyone opens an editor. It starts with a kid handing you a keyboard and watching you get it wrong. (HW)

WHEN NATIVE AMERICAN HISTORY BECOMES THE CS LESSON

Inspiring and empowering Native youth to see themselves in computer science

The choices we make right now — including the curriculum we build — should serve not only the students in front of us today, but also the communities and futures they will help to shape. Indeed, in many Native American and Indigenous communities, decisions are taken through a lens of the effect they will have on the next seven generations. For me, that belief has become a way of thinking about computer science education. It is not simply a set of technical skills, but a way for Native youth to understand their histories, strengthen their identities, and build tools for the future.

Too often, even high-quality computer science curricula are culturally neutral by default. This means students are not able to see their communities, histories, or responsibilities reflected in the work. I started Relentless Feather (**relentlessfeather.org**) because I was thinking about the future of Native youth in my community and asking myself whether they were being invited into the opportunities and possibilities being offered by the digital future.

I wanted to create something that would inspire Native youth to consider a future in computer science. A Native-focused, Native-centred computer science curriculum has since been developed that is based in Native history, showing Native youth that they are represented in computer science education and that they have every right to be coders, software developers, and computer science professionals.

Growing up on the reservation, I saw no shortage of talent, curiosity, or creativity. What was often lacking, though, were opportunities and pathways that helped young people connect those strengths to future possibilities. This is the gap Relentless Feather is trying to close: creating computer science learning that begins with Native youth, reflects their communities, and treats their futures as something worth planning for with care. In that sense, teaching computer science is also part of the work of protecting the next seven generations.

What I mean by Native-centred

When I talk about Native-centred computer science, I'm not talking about computer science with a Native theme placed on top of existing content. I do not mean swapping a name in a word problem, or putting a Tribal flag on a slide. Native-centred CS begins with the learner's community, history, identity, and responsibilities, and then builds real technical skills from there.

Native-centred CS comes down to three ideas and how they work together. The first idea is that the content connects to the student. They are learning computer science by building projects tied to their own land, their own history, and their own identity and community, not someone else's. The second idea is that history is the focus. Students learn what happened here, what is still happening, and why that history matters in the world they are growing up in. And the third idea is that the skills are real. Students are still learning variables, conditionals, loops, events, coordinates, debugging, and design — the same computer science concepts taught in any other classroom — but through content that reflects who they are and where they come from.

A lesson about Minnesota's treaties

One example of this approach is a Scratch project called 'Treaties That Shaped Minnesota', which I taught in March 2026 to about 250 sixth-grade students at Olson Middle School in Bloomington, Minnesota, with the support of teachers Alexandra Holter, Ashly Tritch, and Amy Kienberger.

The project begins with a historical question: How did Minnesota become Minnesota? Between 1805 and 1866, a series of treaties between the US government and the Dakota and Ojibwe nations transferred specific areas of land. Piece by piece, those treaties shaped the area we now recognise as the state of Minnesota.

In the Scratch project, the land areas affected by each treaty or historical event are represented as sprites. Students begin with 13 irregularly shaped pieces placed beside an empty outline of Minnesota. They drag each piece towards its correct location on the map and its position is checked. When a piece is close enough, students are prompted with a question about the treaty or event connected to that land. If they answer correctly, the piece snaps into place and a sound plays. Some of the sprites are already



coded, and students are responsible for completing the remaining pieces by adding the logic, questions, and interactions that make the project work.

Learning the history of Native Americans in Minnesota is not separate from computer science. The project uses variables, to keep track of position and whether a piece has been locked into place; and conditionals, to decide whether an answer was right. Loops, events, coordinates, collision detection, user input, operators, sound, debugging — all of it sitting inside one Scratch project, and all of it directed at one goal. What makes the lesson Native-centred is that the treaty history is the reason the project exists. Students are learning Scratch, but they are also learning how Minnesota became Minnesota, how treaties shaped the land beneath their feet, and how code can help them study, question, and retell that history.

Where this idea keeps showing up

The same design principles — content connected to students' lives, Native American history as part of the learning, and rigorous technical skill development — show up across Relentless Feather's other projects:

- Rez Dog Tracker App: Unplugged introduces younger students to data collection, organisation, and app design through a community-based public health question: How might an app help people understand and respond to free-roaming dogs on reservations?
- RF Code is a 27-unit web design curriculum in HTML, CSS, and JavaScript in which Native history is the foundation for every unit. Students build websites while learning about sovereign Tribal nations, language revitalisation, design, and eras of Federal Indian policy.
- Student portfolios become more than a collection of technical projects. They become evidence that students can build with code while also studying their histories, communities, and futures.

The work ahead

Across these projects, the goal is not simply to make computer science more engaging, although engagement matters. The deeper goal is to show Native youth that technical learning can begin with who they are, where they come from, and what responsibilities they carry into the future. When students build with their own histories and communities in mind, computer science becomes more than a skill set. It becomes a way to study, preserve, question, and create.

The next seven generations of Native youth are not going to be served by a curriculum that treats their identity as decoration, their history as supplementary, or their future as someone else's responsibility to plan. They are going to be served by computer science, by AI literacy, by every technical skill the modern world is asking for, taught in classrooms where their nations, their languages, their lands, and their stories are the starting point.

In the words of the Lakota writer Joseph M. Marshall III, "Stories of virtue are at the core of cultural renewal for each new generation." What we are building at Relentless Feather is one way of doing that. Yours, if you are doing this work too, is going to look different, and it should. The work of each generation is to carry the story forward in whatever medium the times have given us. (HW)



FEATHER LAROCHE

Feather is an enrolled member of the Kul Wicasa Oyate (Lower Brule Sioux Tribe) and the founder and executive director of Relentless Feather, a Native-led nonprofit developing coding and AI education grounded in Native history and Tribal knowledge and identity. His work focuses on preparing Native youth to become creators, entrepreneurs, and future Tribal leaders in a world increasingly shaped by technology.

CREATIVITY AFTER HOURS

How Boys & Girls Clubs of Palm Beach County are building innovators and tinkerers through their STEM programme

When the school day ends, the learning at Boys & Girls Clubs of Palm Beach County, Florida, is just getting started. At clubs across the county, members are designing and racing solar-powered cars, building mini generators from cardboard, wire, and magnets, practising bricklaying and job-site safety, and creating apps that tackle real community needs. Each experience turns curiosity into hands-on STEM learning, and each one is made possible by partners who believe in what young people can do.



Hands-on learning with drone technology

Fourth–fifth-grade: exploration

Beginning in the fourth grade (ages 9–10) in preparation for the fifth-grade science assessment, this foundational stage is designed to build curiosity and interest through hands-on, problem-solving learning projects. Our Solar Fare event is a six-week experience focused on solar energy, engaging over 200 students in hands-on STEM learning. Club members participated in the Junior Solar Sprint, designing and racing solar-powered cars while also learning how renewable energy reduces greenhouse gases (**helloworld.cc/junior-solar-sprints**). The competition highlighted creativity, engineering design, and teamwork, with participating elementary sites earning top placements through support from the Florida Solar Energy Center (FSEC), FINFROCK design-build firm, and Florida Power & Light.

The Solar Cook-Off expanded the impact of the programme by involving additional club members who built solar box cookers from recyclable materials to explore how sunlight is absorbed, reflected, and transmitted. Partnerships with the FSEC, the University of Florida IFAS (4-H), and the Florida Department of Agriculture & Consumer Services strengthened the programme across all participating elementary sites in the county.

Sixth–eighth-grade: connections

Learning deepens as students begin to connect their interests to civic responsibility through mentorship from STEM community partners. This season, middle-school teams (ages 11–14) harnessed that same inventive spirit to earn spots in regional competitions by winning top placements in robot design and showcasing creative builds enhanced by in-house 3D printing, thereby strengthening essential STEM skills including engineering, coding, teamwork, and problem-solving.

Ninth–twelfth-grade: career exposure

Internships, apprenticeships, and scholarships are now the goal, and we encourage students to participate in career fairs, speak with industry professionals, and take tours of facilities. Through a generous STEM & Work-Based Learning grant, Boys & Girls Clubs of Palm Beach County is expanding career-connected learning by empowering teens to lead, mentor, and innovate.

As part of this initiative, robotics club members serve as STEM mentors while also developing technology-driven solutions to real community needs, for example creating an app that tracks and maps the accumulation of rubbish around local waterways to support targeted clean-up efforts. High-school robotics teams (ages 14–18) have advanced through qualifying matches to secure places in regional competitions, while our older members showcased innovative ideas for accessibility — such as AI-powered tools that interpret sign language gestures — at student innovation events hosted by science centres and universities.

Hands-on problem-solving

Club members who were named Youth of the Year (YOY) were extended special invitations to attend partner STEM Initiative Youth Energy Academies, where the day

EXAMPLE PROJECT

Boys & Girls Clubs STEM Workshop: Create a community app and showcase

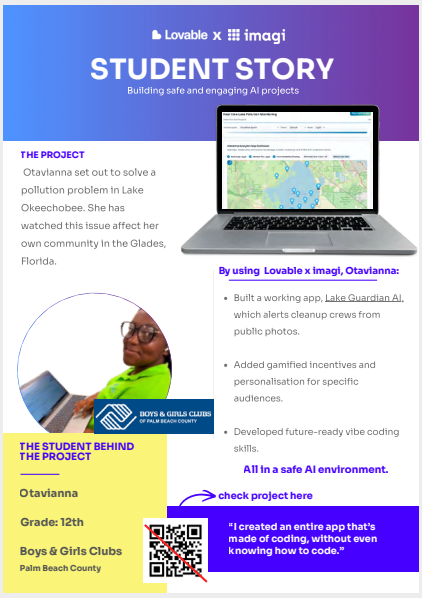
AIM

Build a functional prototype or tracker to address a local environmental or community challenge, and collaborate with industry partners for mentorship and to showcase the prototype.

MATERIALS

Computer or tablet, access to no-code development tools such as Lovable, local mapping data, and guidance from partner networks.

1. **Identify a need and connect with partners:** Research a local issue within the school. Small actions can create big steps. Some examples include food waste, littering, and equitable access to AI training. There is a lot of talent and business in your backyard. Contact your local chamber of commerce to find organisations and businesses to partner with.
2. **Real-world connection:** Invite mentors from local colleges, universities, or school sites to break down the complexity of the problem. Tie in your town's redevelopment plan to give context on how your local area is addressing the problem.
3. **Test and iterate:** Share the application with peers and mentors, gather feedback, and refine the code or layout to improve performance.
4. **Follow-up:** Move with intentionality and strategy. The individuals you engaged with throughout the process can become your biggest advocates. With approval from your school or location, invite these mentors to a class presentation or showcase. The goal is to send a follow-up — a thank you letter with pictures or a heartwarming email to show the progression of your work. Invite local community leaders so students can see the intersection between plans and policy in their city or state.





Club member Otavianna used Imagi x Lovable to design a website that tracks trash near Lake Okeechobee using real-time user data

concluded with a hands-on challenge building mini model generators from cardboard, wire, and magnets to measure voltage and put engineering concepts into practice. At regional teen centres, Project Build brought a trades workshop offering practical exposure to construction fundamentals such as bricklaying, using spirit levels, and learning safety procedures. In seasonal sessions, club members across the county also engaged with drone technology and robotics through hands-on inquiry, mentorship, and real-world problem-solving.

Industry insight

Experiences at aerospace and defence facilities showed how world leaders use virtual reality and remotely operated vehicles to defuse threats and keep communities safe. Environmental partnerships and conservation summer camps are committed to leading the next generation of changemakers through advocacy, youth voice, and civic engagement, earning the organisation recognition for its dedication to environmental stewardship.

Where curiosity becomes innovation

From solar-powered cars and box cookers to career tours, trades workshops, robotics regionals, and protective environmental apps, every experience shares the same spark. At Boys & Girls Clubs of Palm Beach County, creativity doesn't clock out when the final bell rings. It grows after hours, driven by the belief that it takes a village to raise a child. By engaging industry professionals, academics, and volunteers to build a thriving STEM community, mentors help young people turn big ideas into real-world impact across clubs throughout Palm Beach County and envision themselves within a future STEM career pathway. (HW)



REBECCA HAYNES

As STEM programme manager at the Boys & Girls Clubs of Palm Beach County, Rebecca designs learning experiences that build communication, collaboration, and digital literacy using robotics and coding, makerspace engineering, and AI. Rebecca grows professional communities through CSTA and weekly staff coaching so more youth can see themselves in STEM.

TEACHING LOOPS ON THE PICRAT GRID

Andy Perryer places one loops lesson across the PICRAT grid and argues for choosing a cell on purpose

The PICRAT model by Kimmons, Graham, and West (**helloworld.cc/kimmons**) is a framework for evaluating and improving how digital tools are integrated into classrooms. PICRAT is a matrix tool that allows teachers to reflect on how effectively they are using technology. PIC runs along the y axis, and RAT along the x axis (**helloworld.cc/picrat-quick-read**).

Computing lessons have a place in every part of the PICRAT grid, but more cells does not mean more technology. Sometimes a lesson needs a worked example on the board and nothing else. The skill lies in choosing the PICRAT cell on purpose and being able to explain why.

What PICRAT asks

PICRAT asks two questions about classroom technology. The first is what learners do with it. Passive students watch or read. When they answer questions or debug someone else's code, they are Interactive. Creative is the row at which they make something of their own.

The second question is what effect the technology has on the teaching. Replace means it does the job of whatever you used before, which in computing is often other technology. If the task stays the same but works better, usually because feedback is faster, that is Amplify. Transform describes tasks that could not exist without the technology.

Teaching loops four ways

I taught a 60-minute Year 8 (ages 12–13) lesson on 'for' loops in Python. The aim of the lesson was that students should be able to explain what the loop counter does and predict a short loop's output. **Figure 1** places it in all nine cells.

Here are four, in teaching order for this aim; a different aim would start elsewhere.

Figure 1. One loops lesson, nine placements

Year 8, for loops in Python. Outlined cells are the placements discussed in the article.

	Replace	Amplify	Transform
Creative	**CR** Creative Replace Write a loop of your own design in a plain text editor instead of the exercise book, with nothing to run it in.	**CA** Creative Amplify Build a five-round micro:bit reaction game with a for loop, testing it on the device after every change.	**CT** Creative Transform Build a micro:bit step counter the PE department wears for a week, then improve the code from their data.
Interactive	**IR** Interactive Replace Pair-debug a broken loop in a shared editor, doing what two students at one keyboard would do.	**IA** Interactive Amplify Solve a Parsons problem in an online IDE and run it for instant feedback. Or pair-debug while the teacher watches every pair's code live.	**IT** Interactive Transform Change the loop count and growth rate in a population simulation that runs 1,000 generations, and explain what happens.
Passive	**PR** Passive Replace Watch a recorded worked example of the teacher live-coding the loop, in place of the demo at the board.	**PA** Passive Amplify Step through the loop in a code visualiser, watching the counter and output change on every pass.	**PT** Passive Transform Watch a turtle loop draw a 10,000-step pattern as the teacher changes one number at a time.

A worked example: Passive replace

I recorded myself live-coding a loop that prints the seven times table, and the class watched on laptops (see **Figure 2**). It did the job of my demonstration at the board, and students saw the shape of a loop before having to write one. Being able to pause and rewind nudged this activity towards Amplify, but the task hadn't changed, and the real gain was that I could help students while it played. Also, the first time, several students copied the loop into their editor without reading it.

A Parson's problem: Interactive amplify

This was a loop lesson rebuilt as a Parson's problem in an online IDE, in which students put jumbled lines in order and then ran them. It answered the copying issue, but a paper Parson's problem is still the same task. Paper can't tell a student that their loop stops one short. The online version checks order and indentation without the blank page, so it suits the slot after the worked example.

```
for i in range(1, 13):
  print(i, " x 7 =", i * 7)
```

Figure 2. The seven times table loop

The worked example in the video, and the loop in the Parsons problem once it's solved. The bug planted for the debugging pairs is range(1, 12), which stops at 11 × 7.

```
for i in range(1, 13):
    print(i, "x 7 =", i * 7)
```

TRY IT NEXT WEEK

Take one lesson you are teaching next week. Place it on the grid and write the cell at the top of the plan, with one sentence on why. Then pick one activity and move it one cell on purpose, and note what changed for the students. For more examples, 32 computing lessons are placed on the grid, each with its reasoning, at **helloworld.cc/picrat-examples**.

Pair debugging in a shared editor: Interactive replace or Interactive amplify
Pairs fix a loop with planted bugs, talking as they go, where off-by-one errors surface. If the shared editor only does what two students at one keyboard would, that is Interactive replace. Watch every pair's code update live, step in when a pair stalls, and the same activity becomes Interactive amplify. The tool is the same. The cell comes from how you use it.

A micro:bit reaction game: Creative amplify
Students write a 'for' loop that lights the display after a random pause and times how long the player takes to press a button (see **Figure 3**). Using a loop in something they designed shows whether the idea has stuck, and one pair added a score and a sound nobody asked for, which is the Creative row doing its job. I have had this cell argued up to Transform. The class could write the same game in a simulator; the device in their hands adds feedback, which amplifies the task. A step counter the PE department wears for a week, improved from real users' data, would be Creative transform, because the task has changed.

```
from microbit import *
import random

times = []
for turn in range(5):
  display.show(turn + 1)
  sleep(random.randint(1000, 4000))
  display.show(Image.HEART)
  button_a.was_pressed()
  start = running_time()
  while not button_a.was_pressed():
    pass
  times.append(running_time() - start)
  display.scroll(str(times[-1]))
  sleep(1000)
display.scroll("Best " + str(min(times)))
```

Choosing on purpose

Passive replace often receives undeserved criticism. For a novice meeting a loop for the first time, it is usually the honest choice. John Sweller showed in the 1980s that beginners learn a new structure better from worked examples than from solving problems unprepared (**helloworld.cc/sweller**), and PRIMM, from Sue Sentance and colleagues, builds that in by asking students to predict and investigate before they make anything (**helloworld.cc/primm**).

The risk is when every lesson sits there. One video recap before a hard problem-solving lesson is a choice. Weeks of lessons like this is a habit.

The easiest way to spot a teaching habit is to put a week of lessons on a wall. Print the grid as large as possible and ask each teacher to stick three of last week's lessons on it. Every time I have run this, the cluster has landed in Interactive amplify, a perfectly good place to be. The best moment I have seen at the wall was a teacher who put one lesson in two cells, held out against the room, and was right, because it changed cells halfway through. If the cluster is a habit, each teacher picks one lesson and the group agrees the smallest move to shift it by one cell. A prediction question every couple of minutes makes a video Interactive; letting students choose what their loop makes turns a set project Creative.

Resist setting a target for how many lessons sit where, because once the grid becomes a score, people place lessons to hit it. It works better as a planning note. Write the intended cell beside each lesson in a unit, with a reason, so colleagues can argue with it. (HW)

Figure 3. The micro:bit reaction game

MicroPython. The for loop runs five rounds, showing the round number, then the best time.

```
from microbit import *
import random

times = []
for turn in range(5):
    display.show(turn + 1)
    sleep(random.randint(1000, 4000))
    display.show(Image.HEART)
    button_a.was_pressed()
    start = running_time()
    while not button_a.was_pressed():
        pass
    times.append(running_time() - start)
    display.scroll(str(times[-1]))
    sleep(1000)
display.scroll("Best " + str(min(times)))
```



ANDY PERRYER
Andy is director of education innovation at Cognita, an international schools group. He is based in Dubai. A former head of primary computing, he built PICRAT Suite (**picrat.com**), free tools for planning and reflecting on classroom technology.

MISSION SPACE LAB

The European Astro Pi Challenge is an ESA Education project run in collaboration with the Raspberry Pi Foundation and implemented by ESEROs at a national level. It gives young people the chance to learn how to code and conduct real space science in orbit on the Astro Pi computers aboard the International Space Station (ISS).

ABOUT ASTRO PI



The challenge offers two missions: Mission Zero and Mission Space Lab. Mission Zero is designed for beginners. Individuals or groups write a simple Python script to display a personalised image for the astronauts to see. Mission Space Lab is designed for advanced coders. Teams write a Python program to log sensor or camera data to explore life on Earth or orbital mechanics.

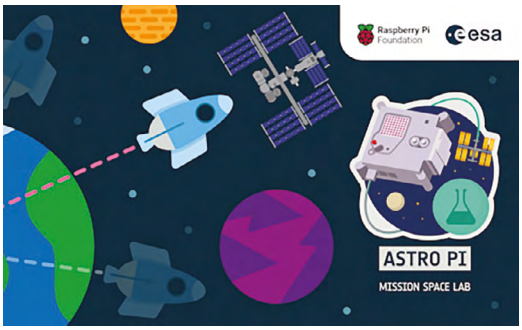
Enter the European Astro Pi Challenge 26/27 by visiting **astro-pi.org** to select a mission.

AGE RANGE

12–19 years

ELIGIBILITY CRITERIA

- Working in a team of 2 to 6 young people
- Supervised by a mentor who will register the team (mentors cannot supervise more than ten Mission Space Lab teams each)
- Enrolled full time in a primary or secondary school in an ESA member state, Canada, Cyprus, Latvia, Lithuania, or Slovakia; or a member of a group such as a Scout group or after-school club, or certified home school located in an ESA member state, Canada, Latvia, Lithuania, or Slovakia
- Part of a team in which at least 50 percent of team members are nationals of an ESA member state, Canada, Cyprus, Latvia, Lithuania, or Slovakia
- Submit completed programs by 22 February 2027 12 noon CET



Run your own code aboard the International Space Station

Mission Space Lab invites young people to run their own code on the International Space Station (ISS), giving them an unforgettable opportunity to conduct real science in space. Teams write a Python program that captures sensor data from Raspberry Pi computers on the ISS for use in their very own space science projects.

Every eligible program is given a ten-minute slot on the ISS. Once the mission is complete, successful teams will receive official certificates showing the exact time and geo-coordinates when their code ran. They'll also receive their very own data and image sets to download, keep, and analyse.

This Mission Space Lab worksheet is to use in your classroom or Code Club. It includes the complete worked code example alongside guided prompts for young people to predict, investigate, and modify their programs. The full Astro Pi Mission Space Lab Creator Guide can be found here: **helloworld.cc/msl-guide.**

REQUIREMENTS

- Any macOS, Microsoft Windows, or Linux computer to write and test a Python program on
- Astro Pi Replay tool, which works on Chrome, Safari, and Mozilla Firefox browsers (**helloworld.cc/astro-pi-replay**)

Astro Pi Mission Space Lab

A worked code example for teams of students to use with the Mission Space Lab Creator Guide.

Team name:_____ Date:_____

THE CODE

```
from picamzero import Camera
from sense_hat import SenseHat
import csv

camera = Camera()
sense_hat = SenseHat()
# take a photo and save it in a file called image.jpg
camera.take_photo("image.jpg")
# create a variable to store the sensor readings
data = []

# repeat 5 times
for i in  range(5):
    # read the raw x,y,z data from the magnetometer.
    # The contents of magnetometer_reading will be a dictionary
    # that looks like this:
    #
    # {
    # "x": 7.907470226287842,
    # "y": 5.466001033782959,
    # "z": 12.868783950805664
    # }
    magnetometer_reading = sense_hat.get_compass_raw()
    # Take the numeric values from the magnetometer_reading
    # and put them in a list called magnetometer_values
    magnetometer_values = list(magnetometer_reading.values())
    # append the values to the data list to save later
    data.append(magnetometer_values)
    # open a file called data.csv for writing

with open("data.csv", "w") as csvfile:
    writer = csv.writer(csvfile)
    # write a header to explain what the columns mean
    writer.writerow(["magnetometer_x",
    "magnetometer_y","magnetometer_z"])
    # write all 5 readings from the data list to data.csv
    writer.writerows(data)
```

PREDICT

Read the code with your team. What will it save? How many sensor readings will the program capture? Which sensor is being used? What will be the output?

Our prediction:

RUN

Upload the code to rpf.io/replay. Was your prediction correct? What outputs did you get?

What we saw:

INVESTIGATE

Find the loop. How many times does it repeat? What is inside the "data" list at the end?

Our answers:

Astro Pi Mission Space Lab

MODIFY – Change the code

1. Capture a sequence of photos

The code takes only one photo. Change it to take 10 photos, one after another. Use a loop, like the loop for the sensor data. Give each photo a new name, for example `image_0.jpg, image_1.jpg, image_2.jpg`.

2. Take more sensor data

The code reads only the magnetometer. Add code to read another sensor too – for example, the temperature sensor (`sense_hat.get_temperature()`) or the accelerometer (`sense_hat.get_accelerometer_raw()`). Save the new data in the CSV file, with appropriate headers.

3. Run the program for 10 minutes

The code repeats 5 times. Change it so the program runs for up to 10 minutes – the full time your team has on the ISS. Hint: Use the datetime library. Record the start time, then use a while loop that continues until 10 minutes have passed.

MAKE – Personalise your program

Try changing the code to capture specific data to conduct a science experiment of your choice. What data will you need to capture? Think about how much data you will need, and how often you will need to capture it.

TEST AND SUBMIT YOUR PROGRAM

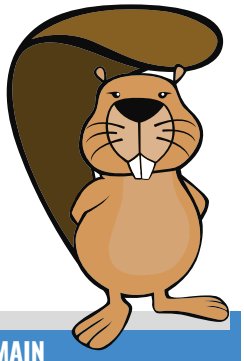
Before you submit your final program, test it with the Astro Pi Replay Tool. You can find more details on how to test it and send it to us here: rpf.io/msl-creator

THE BEBRAS PUZZLE PAGE

Each issue, **Andrew Csizmadia** shares a computational thinking problem for your students based on the work produced by the International Bebras Community

ABOUT BEBRAS

Bebras is organised in over 90 countries and aims to get students excited about computing and computational thinking. Last November, over 525,000 students participated in the UK annual challenge. Our archived questions let you create your own automarking quizzes at any time during the year. To find out more and to register your school, head to **bebras.uk**.



DOMAIN

Algorithms, logic, and programming

SKILLS

Algorithms

AGE

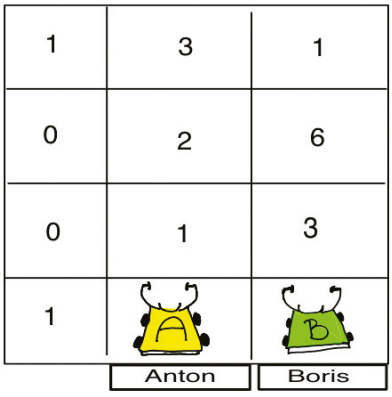
8–12 years

DIFFICULTY RATING

Age 8–10: medium
Age 10–12: easy

THE PROBLEM: RUBBISH ROBOTS

The picture below shows a map of a park divided into rectangles.



The number in each rectangle represents the number of pieces of litter that have been left there by visitors to the park.

The park keepers have two robots, Anton and Boris, who collect all the litter they find in every rectangle they enter.

The robots were given the following instructions:

First, robot Anton was sent: ↑ = upwards ↑ = upwards ← = left

Then, robot Boris was sent: ↑ = upwards ↑ = upwards ← = left

Question:

How many pieces of litter will Boris collect?

3 | 9 | 11 | 12

Background

In computer science and robotics, it is often important to know in which order instructions in a program are processed (or how machines do their jobs).

We need to think about how the actions of our program or robot affect the environment. This task is about how many pieces of litter will be left in each square of the park after Anton and Boris have followed the given instructions. Another goal might be to create a more efficient program — for example, by not sending a robot to collect litter from a square where there is no litter to be collected.

This puzzle was developed by the Bebras team in Slovakia, and reviewed and modified by members of the Bebras community. The solution is on page 79.

KEYWORD SPOTLIGHT: SEQUENCE

Defining everyday words and phrases in computer science

In computer science, a sequence is a set of instructions that are executed in a specific order, one after another. It is one of the three fundamental control structures in programming, alongside selection and iteration. In a sequence, each step is carried out exactly as written, and the order is crucial; changing the order can change the outcome or cause errors.

For example, when writing a program to calculate an average, the data must be input first, then processed through calculations, and finally the result is displayed. If these steps are out of order, the program will not work correctly.

Sequence is often one of the first concepts students encounter in programming. It helps learners understand how algorithms are constructed and why precision matters. Using real-world examples, such as recipes or instructions, can make the concept of sequence more accessible and highlight the importance of correct ordering. (HW)



Beneath a personalised cyberdeck exterior there is a lot of hardware and software at work

ALL HANDS ON DECK

How and why to join in the cyberdeck trend

It all began with two phone calls. The first was a kid asking our CEO how you would build a cyberdeck. The second was the CEO asking me, "Could you look into this?"

HOW TO BUILD A CYBERDECK

A cyberdeck needs four things: brains, input and display, power, and a case. A Raspberry Pi covers the first — the rest is up to you.

Our free guide walks through the parts, the software, wiring buttons and lights to the GPIO pins, and building an enclosure. Top tip: make a cardboard version first — cardboard is cheap, and mistakes can be expensive.

helloworld.cc/cyberdeck

I said yes because I only half knew the answer and I was curious. I read and watch a lot of science fiction, so I knew what a cyberdeck was. But we don't have neural implants in the real world, so what exactly was it that people were calling cyberdecks?

I went looking for a definition and couldn't find one; I went looking for a function and found too many — and these are precisely the reasons people build them.

What is a cyberdeck?

Ask r/cyberdeck and you'll get a hundred answers, but the consensus runs like this: it's a homemade portable machine built around a single-board computer, with a screen and a keyboard. It does the few things its owner actually needs, in a case they chose, with the buttons they wanted. It looks like it escaped from the set of a Ridley Scott movie or a Saturday morning cartoon.

Mine is at the humble end of the scale. In my study I had an old component box, the sort with a hinged lid and moulded compartments, plus a spare Raspberry Pi. I dropped the Raspberry Pi in and bought two things: a tiny Bluetooth keyboard, because nothing I owned would fit, and speakers, because it has to be loud.

It is not beautiful, it runs for about 45 minutes on a battery, and the lid doesn't quite close over the cables. But it plays music, it runs a terminal, and it is unmistakably mine, which is the whole point.

Over to the team

I set my team the same challenge: build your own cyberdeck — no brief, no budget, no rules. The two decks looked nothing alike, and neither resembled mine, but







Becca's Knitdeck is a small cyberdeck built to read knitting patterns; it shows a pattern on an e-ink screen, and you can turn the pages with buttons

both of them started in exactly the same place. I asked my colleagues the same three questions.

What were your first thoughts when I asked you to build a cyberdeck? Did you have preconceived ideas about what one would look like?

Mr C: I thought of William Gibson's *Neuromancer*. His decks are sleek Japanese machines, somewhere between a games console and sound equipment. Mine was never going to be that polished, but I wanted the same cyberpunk vibe. The key constraint was portability: it had to close up, travel safely and be ready to operate anywhere. A rugged Peli case gave me that maker-built briefcase form, while illuminated controls and laser-cut panels supplied the science-fiction theatre. I began with the aesthetic, then let it decide what the machine needed to become.

> A CYBERDECK NEEDS FOUR THINGS: BRAINS, POWER, INPUT AND DISPLAY, AND A CASE — THE REST IS UP TO YOU

Becca: I also began with the aesthetic. My feed is full of crafty makers posting cyberdecks that are fluffy and bejewelled, cobbled together with glue and card, built into a Polly Pocket or some other cute retro toy case— basically everything you wouldn't normally think of as a computer.

Computers are inherently so personal. Our whole lives are embedded in them, and yet they look so generic and so boring. How things look matters a lot to me: I love fashion, I love doing hair, and I want the things around me to fit with that. I've always made computers look different with electronic textiles or beads sewn into my phone. A handmade personal computer that actually looks personal to you was what drew me in.

Talk me through your creative process. What did you decide to build, and how did you realise your idea?

Mr C: Once I'd settled on the Peli case, I raided my boxes of abandoned technology: a laser keyboard from an old Kickstarter, leftover arcade buttons, tablets once used at events. Those constraints drove the design. I planned a laser-cut acrylic control

GIBSON'S DECK

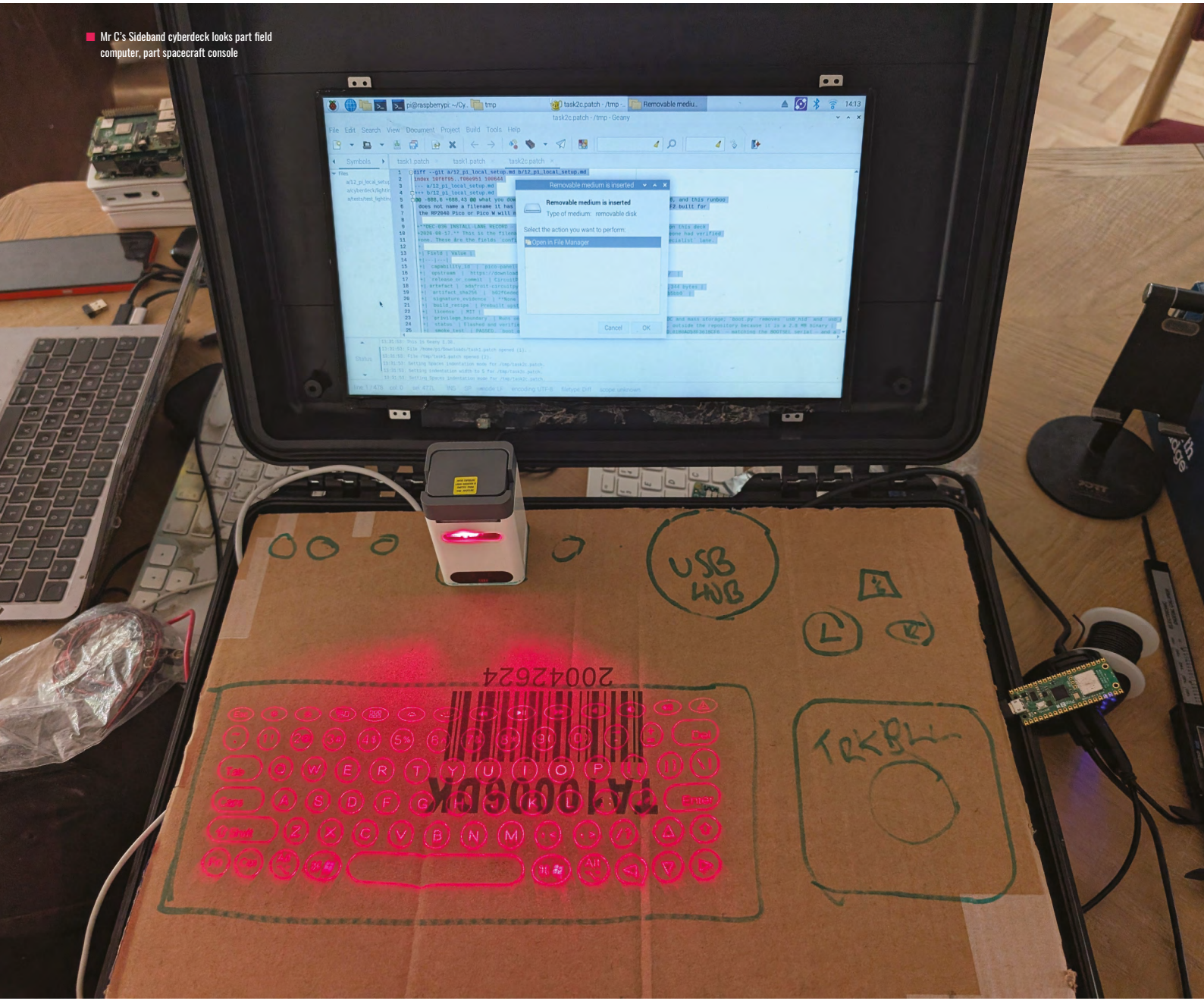
The name comes from William Gibson's 1984 novel *Neuromancer*, in which a 'console cowboy' called Case jacks into the matrix using a cyberspace deck. Gibson never explains how it works, which is exactly why makers have found it so easy to reinvent.

He wrote this book, which gave us the word 'cyberspace', on a 1927 manual typewriter, and didn't own a computer until after the royalties arrived.

panel, then turned the tablets into wireless auxiliary displays connected through the Raspberry Pi. I didn't know how to build much of the networking or software, so I effectively vibe-coded those parts, planning the architecture first, then running an agent on the Pi itself.

Becca: From the start I wanted my cyberdeck to do something specific, rather than being a catch-all device, so I went with a knitdeck. I have hundreds of patterns and I knit every day, following them on my tiny phone screen. I wanted the device I knit with to feel like part of the knitting world.

Finding the enclosure defined the whole design: how big it could be and how it felt. I got the case second-hand and it has a certain noughties grandma aesthetic, which I added to with charity-shop fabric, beads, and wadding for softness. Instead of a keyboard, I used pastel-coloured keys and a couple of arcade buttons, because I wanted it tactile, and everyone loves the sound of mechanical keys.



Mr C's Sideband cyberdeck looks part field computer, part spacecraft console

What tips do you have for anyone building their own cyberdeck? What mistakes did you make?

Mr C: My mistake was scope creep. Every feature suggested three more until an enjoyable build ended up pretty stressful. Start with the simplest definition: connect a Raspberry Pi, screen, and controls, make them portable, and you already have a cyberdeck. Everything beyond that should earn its place. The distinctive part isn't technical complexity; it's the aesthetic and the user experience. Use neglected parts and old tech, and build around whatever interests you most. Set a viable build, finish it, then add the glowing controls, displays, and fruity parts that make it unique.

Becca: Keep the deck simple. My mistake was making mine too complicated, what Mr C calls scope creep. I've only got six buttons, but that is a lot of wiring, a lot of soldering, and a lot of additions to the code.

I wanted it to be low energy and run on a battery, so I planned an e-ink display with a Pi Zero. That brought compatibility problems and much more software. I ended up using a full Raspberry Pi 5, so it doesn't run on a battery at all. If you can, add one or two buttons max, and use a normal keyboard and monitor.

Find the case first and let it define the build, because that's where the personality comes from. Let the complexity be in the design, rather than in how many buttons you've got. (HW)

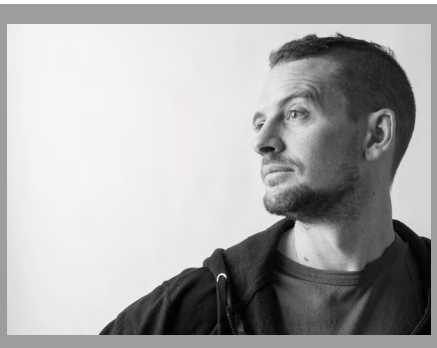


Marc's Toolbox terminal cyberdeck is built into an old component box which plays music and runs a terminal



MARC SCOTT

Marc spent 15 years teaching secondary-school science, computer science, and electronics before joining the Raspberry Pi Foundation, where he has worked for a decade developing engaging Code Club projects for young people.



MARK 'MR C' CALLEJA

Mr C is a teacher, hacker, surfer, D&D Dungeon Master, thrower of bladed implements, and maker of useful things. During the day he makes educational resources for the Raspberry Pi Foundation, which you can find on the Foundation's website.



BECCA ROSE

Becca is a learning manager on the Code Club projects team at the Raspberry Pi Foundation. She is maker and educator and loves anything craft related. She enjoys teaching and learning computing with arts and crafts.

COMPUTING EDUCATION THROUGH PLAY

Insights from educators at the Digital Schoolhouse Playful Computing Conference

The Playful Computing Conference is an annual event based in the UK. It is hosted by Digital Schoolhouse and brings together educators and digital industries to transform computing education through play. We asked educators who attended this year's online conference to share their experiences and tell us how it will help their practice.

What was your main motivation for attending the Digital Schoolhouse Playful Computing Conference this year?

I attended in 2025 and it was one of the best conferences I have been to. I love the focus on play and video games. I have always been a huge advocate for teaching children to code through making games, so this conference really leans into my interests!

I would definitely recommend the conference to fellow computing educators. I feel that senior leaders would also benefit from looking at how to introduce more playful learning across the school so it doesn't stop in the early years.

I believe strongly in learning through play, but unfortunately, teaching through play often stops at reception. Wouldn't it be great if it continued through all years, including secondary education?

NICKI COOPER

Nicki is a former teacher with over 20 years' experience as a secondary-school computing teacher, advanced skills teacher, and head of department. Now she develops computing resources and delivers creative computing workshops for U Can Too (**ucantoo.org.uk**).

How would you describe the overall atmosphere of the conference to a teacher who didn't attend?

The overall atmosphere was incredibly positive, welcoming, and inspiring. Despite the event moving online at short notice because of the extreme heat, the organisers created an engaging and well-structured experience that retained a strong sense of community. The sessions were practical and thought-provoking and were delivered by passionate educators, researchers, and industry experts who were keen to share ideas that could be implemented immediately in the classroom.

What stood out was the shared enthusiasm for making computing creative, inclusive, and accessible for every learner. There was a genuine willingness to exchange ideas and celebrate innovative practice, which made the conference feel collaborative rather than simply a series of presentations.

The day exceeded my expectations. I came away with fresh ideas, practical strategies, and renewed enthusiasm for developing our computing curriculum. Although I would have enjoyed the face-to-face networking opportunities had the conference been able to go ahead in person, the online format still provided an excellent professional learning experience and reinforced the value of being part of such a supportive education community.


NICKI COOPER

OLIVIA RODRIGUES

Olivia is the assistant head teacher for technology and innovation at Donhead Preparatory School in Wimbledon, London. She teaches primary computing, AI, digital literacy, and physical computing, supporting tech integration across the wider curriculum.

Can you describe one or two playful computing resources that you are excited to try in your own classroom?

The conference provides you with information about the raft of new tech available and how to adapt it for your own setting.

I have used Flock XR (**flockxr.com**)


OLIVIA RODRIGUES

already, but would like to promote it more with the schools I am involved in as an alternative, or extension, to Scratch. The lesson plans are easy to follow and adapt, so I want to share this with staff so that they could explore and deliver them themselves and build their subject confidence.

Sniff and Snails (**sniffandsnails.com**) is a great resource that could be combined with other areas such as design and technology. I would like to develop the Mario Kart idea more and link it directly to the Junior Esport competition.

Get Directing (**helloworld.cc/get-directing**) could be used within English lessons and linked to the VR headsets. I have previously used DelightX (**delightex.com/edu**) to create scenes but never to code movement, so I was excited to try this and then allow the children to see it on the ClassVR headsets.

JOANNE HODGE

Joanne is a Digital Schoolhouse regional and international lead teacher based at Our Lady of Lourdes Primary. She delivers fun free computing workshops to schools all over the North West of England.

How will the ideas gained at the conference influence your teaching?

My favourite part of the day was the thought-provoking advice and ideas shared by Dr Doug Specht in his 'Digital Creativity for Everyone' presentation. His emphasis on digital creativity being not just about tools, but about thinking and purpose, resonated with me. It made me reflect on my lesson planning and consider how I can build in more time and space for pupils to think deeply and develop their own solutions.

I also enjoyed Sheena Peckham's session from Internet Matters, particularly her up-to-date statistics on how children are actually using AI, as well as her introduction to the SIFT tool for evaluating the reliability of AI-generated content.

Although Professor Michael Kölling's session on using Strype for Python programming ran at the same time as Sheena's, I was able to catch part of it. I was excited to see how this system bridges the gap between block-based and text-based programming. One challenge I often see with Year 6 pupils is mastering correct indentation in Python, and Strype's frame-based structure helps overcome this by handling indentation automatically, allowing learners to focus on developing their understanding of commands. I also explored Strype's demo video, which shows how to create a simple game in six minutes — something I would highly recommend to anyone interested in using the platform (**strype.org**).

FIONA KUIJPERS

Fiona teaches primary computing at the British School in the Netherlands. She has taught for over 17 years in both the UK and overseas, working as a primary teacher and a computer science specialist.

What was your takeaway from the conference?

The conference has reinforced the importance of making learning more hands-on, kinaesthetic, creative, and fun. It reminded me that computing should not just be about listening, writing notes, or following instructions on a screen. Learners need opportunities to explore, build, test, collaborate, and play with ideas.

If I had to leave fellow educators with one message, it would be this: stop trying to make computing look like every other subject.

Some of the most powerful learning happens when pupils physically make, move, test, and demonstrate ideas. We sometimes forget that computing does not always have to begin on a screen. Old bits of card, string, paper, counters, blocks, or classroom materials can be used to model algorithms, networks, data, logic, and problem-solving in a way that students can see and touch.

This hands-on, crafted approach can make abstract concepts much more concrete. It gives learners the chance to play with an idea before they are expected to code it.

As AI changes the skills our students will need, creativity, problem-solving, and computational thinking matter more than ever. Playful computing is not a gimmick. It is a powerful way to make learning stick.

GARETH SHAW

Gareth has been teaching for 24 years. His work now spans a wide range of teaching and training roles across Northern Ireland, from teaching IT at Ballyclare High School to delivering digital skills and AI training for businesses, organisations, and adult learners.

Learn more about the Playful Computing Conference and other free resources from Digital Schoolhouse at **digitalschoolhouse.org.uk**. (HW)


JOANNE HODGE


FIONA KUIJPERS


GARETH SHAW

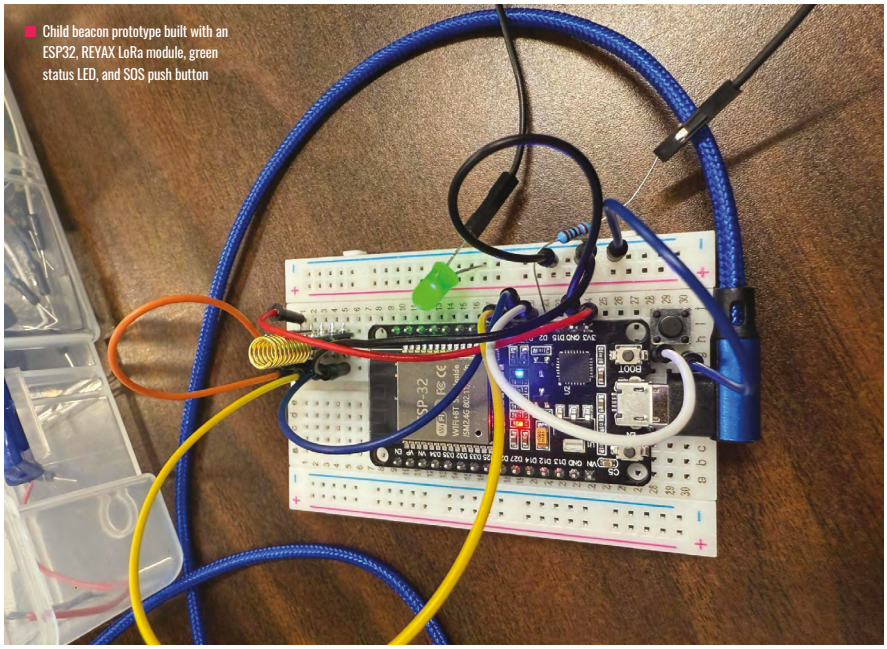
NO SIGNAL? NO PROBLEM

Roza Shaimurat shares how students designed an off-grid safety system

I rarely begin a project by telling my students what to build. Instead, I begin with a simple question: 'What problem can we solve together?' One of those conversations eventually led to ProxyConnec.

We started by asking, 'What happens if a child becomes separated from their family somewhere without mobile service?' Whether it's a hiking trail, a campsite, or a crowded festival, losing cellular coverage is more common than many people realise. Most tracking devices rely on mobile networks, nearby smartphones, or subscription services, and without those networks, many simply stop working.

Rather than accepting these limitations, my students and I challenged ourselves to build something different — a child safety system that could communicate completely off the grid. The long range offered by LoRa made it a natural choice: no towers, no infrastructure, just a direct signal between two devices.



Child beacon prototype built with an ESP32, REYAX LoRa module, green status LED, and SOS push button

ProxyConnec

As the project developed, students decided it deserved its own identity. They combined the words 'proxy' and 'connection' to create the new name ProxyConnec, representing a communication link that continues to work even when traditional networks are unavailable. Naming the project seemed like a small step, but it transformed the way students talked about their work. ProxyConnec was no longer a class assignment — it was something they had created together.

How the system works

ProxyConnec uses two ESP32 microcontrollers and REYAX LoRa radio modules to create a private, long-range communication link between a wearable child beacon and a parent monitoring station — no Wi-Fi, cellular network, or monthly subscription required.

The idea behind the system is simple. The child beacon sends a small PING message every second. As long as communication is active, the beacon's green LED remains on, reassuring the child that they are still connected to the parent station. At the same time, the parent's blue status LED confirms that heartbeat messages are being received. If no message arrives for five seconds, the red warning LED turns on and the buzzer alerts the parent immediately.

One of my favourite moments of this project came when students stopped asking, 'How do we make this work?' and started asking, 'What happens if it doesn't?' That question completely changed the way we thought about the project.

Designing for failure

Students realised that many existing safety devices depend on a child pressing an emergency button. But what if the child is frightened, injured, or simply too young to react? Instead of waiting for an SOS signal, they redesigned the system so that losing communication became the warning itself. In ProxyConnec, silence becomes meaningful. If the heartbeat stops, the parent knows something isn't right.

The beacon also includes an SOS button. When pressed, it immediately sends an SOS message to the parent station, which responds with a faster alarm pattern. Students liked having two layers of protection — both automatic monitoring and a manual SOS option.

Students learnt to program ESP32 boards, communicate using LoRa radios, and build embedded systems. But the biggest lessons came while debugging. Loose wires, missed messages, incorrect pin assignments, and timing issues taught them that engineering is an iterative process. Every problem became an opportunity to improve the design, rather than a reason to start over.



ROZA SHAIMURAT

Roza is a master/lead computer science teacher and district computer science curriculum specialist at LISA Academy in Arkansas, USA. She is a CSTA Impact Fellow alumna, a Presidential AI Challenge state winner, and an advocate for project-based learning which empowers students to use computing to solve authentic real-world problems.

Testing and future updates

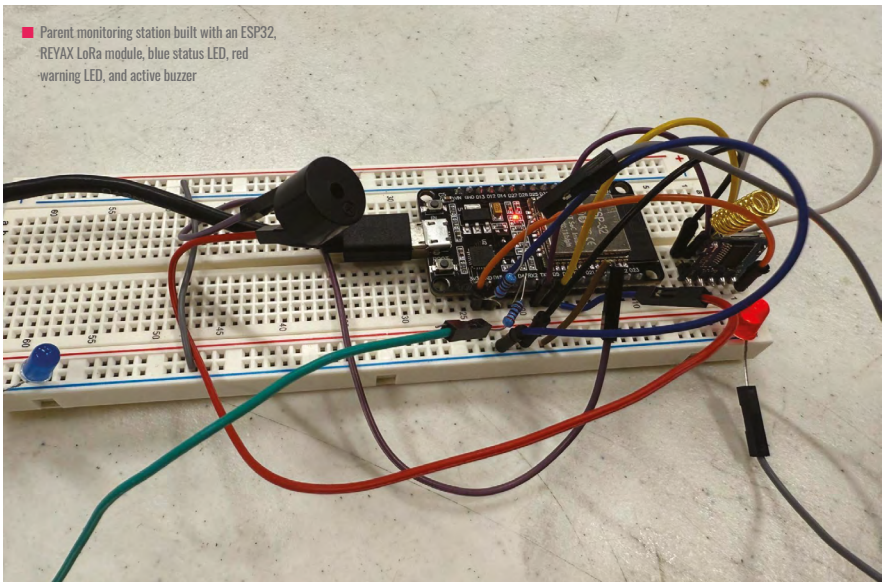
One afternoon, we took the prototype outside to test how it performed in a real environment. We simulated a connection failure by powering off the child beacon and disconnecting the battery. Within five seconds, the parent station responded exactly as expected — the red warning LED illuminated and the buzzer sounded. Students were excited to see the system react correctly, even when communication stopped unexpectedly. That moment reminded us that good engineering isn't about preventing every failure; it's about designing systems that respond safely when failure happens.

As students refined the prototype, they naturally began imagining future improvements, including GPS for location awareness, biometric sensors to detect signs of stress, and Bluetooth connectivity that would allow parents to monitor the system from a phone while still keeping communication completely independent of the internet.

Watching students take ownership of the project reminded me why I enjoy teaching through project-based learning. Once they believed they were building something that could genuinely help people, they stopped thinking about grades and started thinking like engineers. (HW)

LEARN MORE

Want to explore ProxyConnec in more detail? ProxyConnec was presented as a research poster at the 2026 Arkansas Tech University Scholars Symposium. Scan the QR code to view the complete poster and learn more about the project's design, hardware, and future development.



Parent monitoring station built with an ESP32, REYAX LoRa module, blue status LED, red warning LED, and active buzzer

EXAMPLE PROJECT

ProxyConnec: Build an off-grid child safety communicator

AIM

Build a two-device communication system in which a wearable beacon sends continuous heartbeat messages to a parent base station. If communication is lost, the parent station immediately alerts the user.

MATERIALS

- 2 × ESP32 development boards
- 2 × REYAX RYLR998 LoRa modules
- Push button
- Blue LED
- Red LED
- Green LED
- Active buzzer
- Breadboards
- Jumper wires
- 3.7 V LiPo batteries

ACTIVITY

1. Assemble the child beacon and parent station using ESP32 boards and LoRa modules.
2. Program the beacon to transmit a PING message every second and an SOS message when the push button is pressed.
3. Program the parent station to monitor incoming messages and respond with LED and buzzer alerts.
4. Test the system during normal operation, then simulate signal loss by moving the beacon out of range or disconnecting its power supply.
5. Verify that the warning activates after five seconds without a heartbeat and that pressing the SOS button triggers the emergency alarm.
6. Challenge students to extend the design with features such as GPS, biometric sensors, or Bluetooth connectivity.

DISCUSSION

- How is automatically detecting a lost connection different from waiting for someone to press an emergency button?
- How does designing for failure make technology more reliable?
- What other community problems could be solved using a similar approach?





COOLEST PROJECTS

From Bradford to Georgia: Young makers finding their community

Every year, young people from around the world bring their ideas to life and celebrate them together, online and in person, at Coolest Projects. This year was the biggest yet, with over 15,000 young people entering the global online showcase alongside a packed calendar of in-person events across Ireland, the UK, the US, and more.

Coding and creativity go hand in hand, and Coolest Projects exists to celebrate both. Whether a project is a first attempt or a polished creation, what matters is the curiosity and effort behind it.

Events around the world

There's something special about walking into a room full of young makers buzzing with excitement to show what they have built. This year, the Raspberry Pi Foundation hosted in-person events in Dublin, Ireland; Bradford, England; Minnesota,USA; Hyderabad, India; and for the first time, Georgia, US, buzzing with more young-creator energy than ever before.

"Coolest Projects is all about giving young people the chance to share what they've made and be part of a creative community. Bringing the event to Georgia for the first time is a way to reach new areas, meet new creators, and show that opportunities to learn, code, and make should be available wherever young people are."
(Sofia Mohammed, US executive director, Raspberry Pi Foundation)



Meet the creators

The heart of every event is the creators themselves, and their experiences show exactly why Coolest Projects matters.

Gabriel came to Coolest Projects UK in Bradford with Fossil Roulette, a Scratch game inspired by a board game he spotted at the Fossil Festival in Lyme Regis.

"I found the event very exciting because at one point I had a queue of people wanting to try out my game!"
(Gabriel)

ZaLayaa arrived at Coolest Projects Minnesota having built FIND ME ASAP in just 15 days. It's an AI-powered emergency platform that helps parents act quickly when a child goes missing, with features designed around children with invisible disabilities. Beyond the project itself, it was the atmosphere on the day that stayed with her:

"It was really cool seeing all the different projects and what other kids built, and being in a room where it was all kids creating and sharing their ideas."
(ZaLayaa)





> THE ONLINE SHOWCASE OPENS COOLEST PROJECTS UP TO YOUNG CREATORS, NO MATTER WHERE THEY LIVE

For Aarabhi, taking part in Coolest Projects Ireland was about more than showing off her Raspberry Pi Pico project Mystical Waterfall, a nature scene complete with running water and LED lights — it was about living out a dream.

"I watched a Netflix series, *The InBESTigators*, where children showcase their science projects. After watching it, it was my dream to do the same!"
(Aarabhi)

For some creators, the events are about celebrating a shared passion for making with family and friends. After attending Coolest Projects Ireland in 2025 and being desperate to enter again, Leo did just that in 2026. Here's his highlight from the event:

"I went to Coolest projects last year with my cousins and thought it was cool, so I wanted to apply this year. I was very late applying so only had three days to create a project before the deadline, it was pretty intense! My highlight was being at the event with my cousin, Noah. It was great fun!"
(Leo)

A global stage for every idea

The online showcase opens Coolest Projects up to young creators everywhere, no matter where they live or whether an event is taking place near them. This year, a record number of creators from 40 countries took part, each one choosing to put their work out into the world and be part of something bigger.

The judges' favourite picks were revealed in a livestream celebration hosted by science presenter Greg Foot. VIP judge Ben shared some powerful advice with the creators:

"Keep building. Keep experimenting. Keep trying things. Always stay curious. Always try to stay learning. And, the most important thing: it's OK to fail. Trying new things is difficult and you're always going to make mistakes, but that's the best way to learn."
(Ben Powley, senior developer, Jagex)

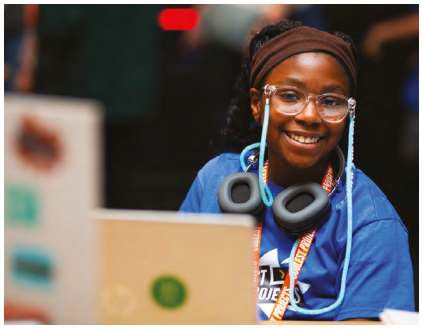
Looking forward

We are excited to share that the Coolest Projects 2027 online showcase will open for entries in January.

We will also be running in-person Coolest Projects events in Ireland, the UK, India, the US, South Africa, and Kenya next year, with further in-person events due to be run by partners around the world. We will announce the full calendar of in-person events soon at **coolestprojects.org**.

Thank you to our sponsors

Coolest Projects is free for young people to take part in because of the generosity of our sponsors. Their support makes the online showcase and in-person events possible, and ensures that any young person with an idea has a place to share it and see it come to life. A huge thank you to our Champion Sponsors: Broadcom Foundation, Amazon Future Engineer, Avnet, and Qube Research and Technologies. (HW)





HELEN GARDNER
Helen is the programme manager for Coolest Projects at the Raspberry Pi Foundation.



Students at Agastya International Foundation in India try out NeuroBridge

THE CAREER JOURNEY OF A YOUNG INNOVATOR

Making neuroscience accessible with NeuroBridge

In past issues, we have often interviewed people who are firmly established in their careers. For this issue, we wanted to speak with a young person who is starting out on their path in computer science and ask them what inspired them to engage in this field.

Hi Aditya! Tell us a bit about yourself.
Hi! I am Aditya Mittal, a 17-year-old from Singapore. I have always been interested in the intersection of neuroscience and engineering, particularly in how technology can make solutions more accessible.

We heard about you because of your invention. Can you tell our readers what you have created?
I built a low-cost portable EEG acquisition system called NeuroBridge. Electroencephalography, or EEG, allows the measurement of electrical activity in the brain. However, commercially available systems cost thousands of dollars.

I made NeuroBridge because I realised how important this technology is in fields including epilepsy monitoring, brain–computer interface research and development, and education in the field of neuroscience. Barriers to accessibility, including cost and a need for specialists, make EEG inaccessible in many underserved communities.

This made me wonder whether I could build an affordable system using commercially available components that could still record meaningful EEG signals — which brings us to NeuroBridge.

How much of your inspiration came from seeing a real-world need and how much came from a technical interest in the challenge?
It was definitely a mix of both. The engineering challenge certainly drew me in, as EEG signals are of extremely small magnitudes, so acquiring them properly means dealing meticulously with noise, electrodes, connections, and signal processing.

The side regarding the real-world need was extremely important in keeping me going. I have been working with the Agastya International Foundation in India for the past three years, an organisation that brings hands-on STEM learning to students who may not have access to it otherwise, and seeing how much students can do when given the right technology made the problem feel a lot more real.

Since then I have also been able to deploy this technology directly at Agastya's main campus. Students were able to visualise and understand their own brain signals for the first time after learning about neuroscience and computer science. NeuroBridge has now been adopted in their curriculum, with over 2000 students already having used it for their learning. The idea of this potential impact gave the project a purpose from the beginning.

Can you break down your hardware and software for us?
The system is built around a Raspberry Pi 5 as the main computing platform. An analogue-to-digital signal converter is used alongside it, in addition to a custom-made cap and electrodes for signal

acquisition. It can record eight EEG channels continuously as a standalone, portable system that can acquire, process, visualise, and store data without the need for an external computer.

On the software side, I mostly used Python. The primary libraries I used were spidev for SPI communication, gpiod for controlling the Raspberry Pi's GPIO pins, SciPy for digital signal processing, and Matplotlib for live EEG visualisation. Part of the reason I chose Python was the fact that it allowed me to run everything as part of one environment centred around the Raspberry Pi and utilise existing libraries that fit my needs.

How did you collaborate with medical professionals or biology teachers to make sure your code was clinically relevant?

My interest in the project came out of several years of epilepsy and neuroscience research, so I was fortunate enough to have researchers and clinicians around me whom I could learn from. Conversations with them helped me understand what actually matters in an accessible EEG device so I could prioritise those things in my system. After completing a summer internship at the Mount Sinai BioDesign lab, I gained a much better understanding of what biomedical technology should look like and was able to ask my mentor at the lab for guidance in the hardware part.

For the research side of the project, which has also been substantial and includes posters at the European Epilepsy Congress and the European Academy of Neurology, as well as a presentation at the Congress of Neurological Surgeons, I have been fortunate enough to work with a neuroscience professor from Columbia University who mentored me and supervised my work. He gave me feedback and reviewed my abstracts and findings to ensure I was on the right track. Speaking with other professionals at conferences and elsewhere has also been the source of many ideas and suggestions.

Did skills such as communication, pitching your idea, or writing technical documentation turn out to be just as important as writing code?

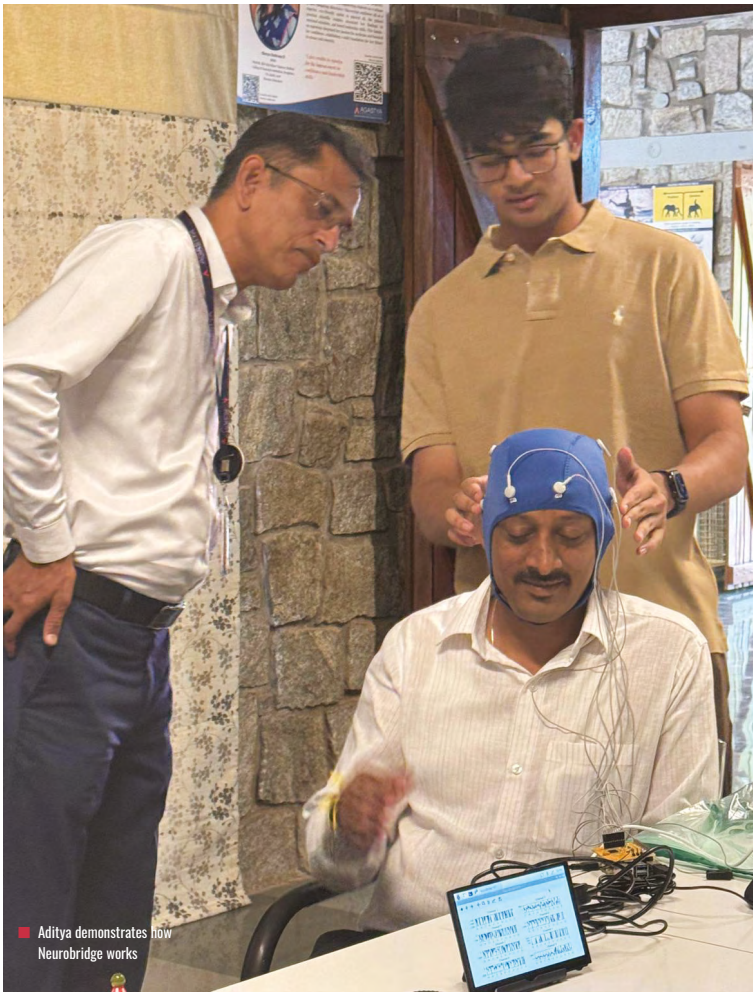
This came as a surprise to me. I thought that the difficult part would be putting the hardware and software together, but communication became its own problem once I wanted other people to use the system. I had to write set-up instructions and troubleshooting guides in different languages and make the interface usable for someone with no experience with these devices.

The presentations I have given at international neuroscience conferences have also required me to explain and justify the project to a wide range of people in a short amount of time. Communication is just as important as engineering in spaces like these!

For a younger student sitting in an introductory CS class today, how does an ambitious project like this bridge the gap between high-school assignments and a viable, fulfilling career in tech?

Speaking from recent experience, high school can give you all the tools you need to start working on a problem you care about. An introductory CS class would teach you loops, functions, arrays, and debugging, and combining these concepts with skills or knowledge from other fields like maths, physics, or biology would allow you to reach many more fields.

My advice would be to find a problem you have a genuine interest in and build a specific and attainable solution to it. It doesn't have to be a new invention or something groundbreaking — it just has to be something you care about and will put effort into. Use what you have learnt in computer science and other classes or activities as tools to solve a real problem. (HW)

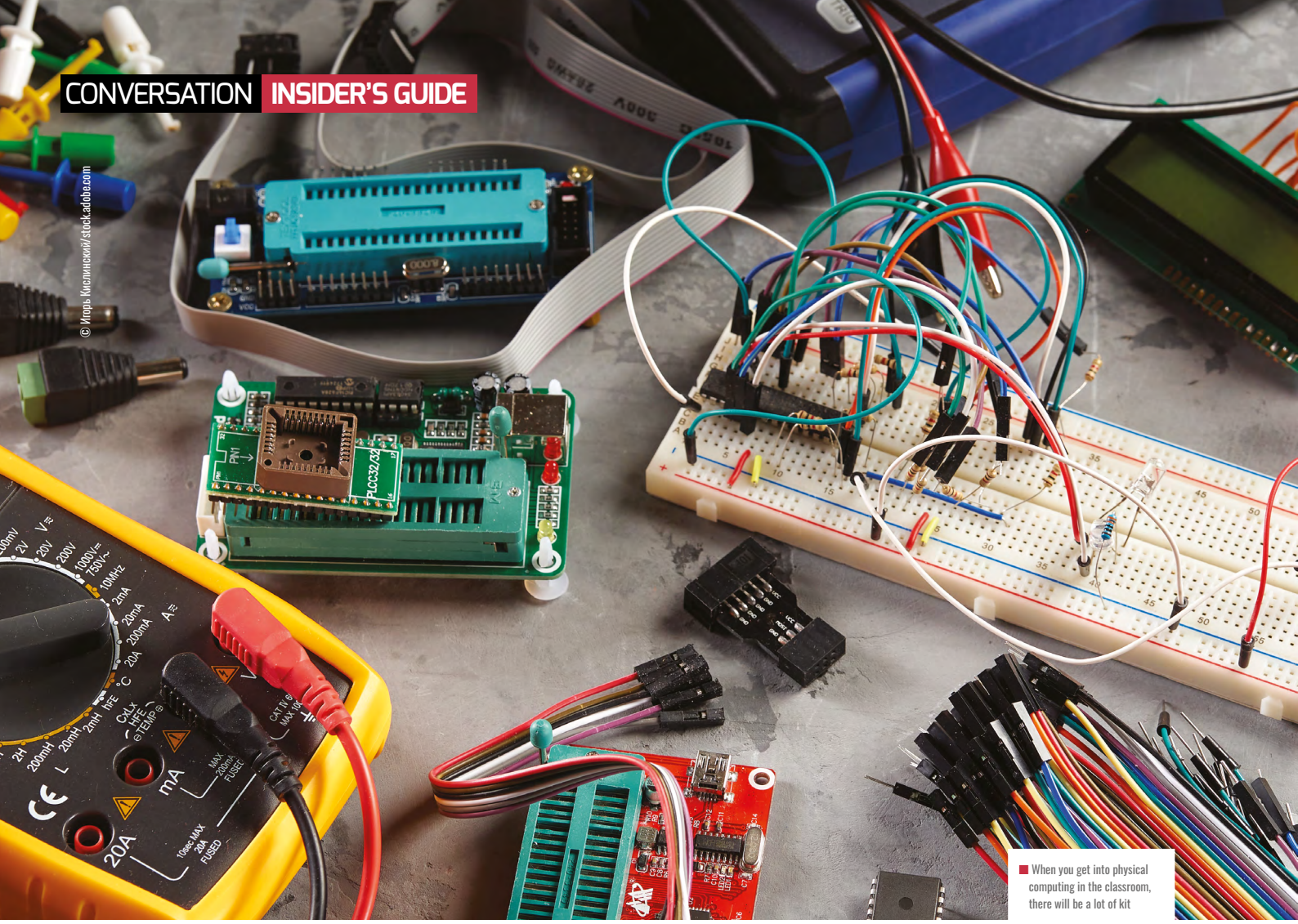


Aditya demonstrates how Neurobridge works



ADITYA MITTAL

Aditya is a 17-year-old student with an interest in the intersection of neuroscience and technology. He developed NeuroBridge, and has had his research presented at international scientific conferences and published in peer-reviewed journals. He hopes to continue learning about and developing accessible neurotechnology.



© Игорь Кислинский/stock.adobe.com

When you get into physical computing in the classroom, there will be a lot of kit

PHYSICAL COMPUTING SURVIVAL TIPS

Lessons from a decade of physical computing in the classroom

As CS educators, we are constantly told how enriching physical computing can be for our students. What we don't hear so much about is the extra headaches it brings for the teachers running the lesson or club. I've been a secondary-school teacher in the UK for over a decade, and in that time I've learnt some hard truths about dealing with pesky wires, temperamental electronics, and large numbers of exuberant students. I hope these cautionary tales can help prevent you from making the same mistakes I did.

I qualified as a CS teacher in 2014, just as the Raspberry Pi was taking off. A mentor suggested I start a Raspberry Pi club, so I got my own board, only to discover that the HDMI would not play nicely with my older classroom monitors. Sadly, that club never got off the ground. The following year, micro:bits arrived, one per Year 7 pupil, which felt like a step forward and gave me my first taste of how engaging physical computing lessons could be. The trouble was, it raised its own questions: did the pupils own the devices, or did the school? If pupils owned them, would they remember to bring them in? If you know Year 7 pupils, you already know the answer to that question.

That's the pattern with physical computing: it promises so much, and it delivers, but only once you have got over a few hurdles. Here's what I wish someone had told me.

1. Always have spares

During school activity week I once ran a five-day workshop with a group of students building Raspberry Pi-powered robot cars. I had planned for every contingency I could think of. What I hadn't

planned for was two students staging an impromptu 'Robot Wars' tussle with their creations and snapping the SD card in one of the cars. Cue a mad lunchtime dash into town for a replacement, followed by 20 minutes reformatting it. Now I never run a session without spares of everything.

2. Plan for mess and set rules early

Physical computing is messy. Small components end up everywhere and pupils are rarely enthusiastic about tidying up. A colleague once unveiled a big new box of Lego kit at an after-school club, only for the school bus to arrive and the children to vanish, leaving her alone with a floor full of tiny colourful pieces. Enforce time limits and tidy-up rules from day one, or you'll be the one picking up the pieces afterwards.

3. Ringfence your time

Physical computing is a time sink, and most teachers are already stretched thin. I found this out while running teams for a national Raspberry Pi competition that was open to all age groups. I ended up managing six teams with wildly different experience levels, and had just one after-school club slot a week. It was a stressful experience and I often ended up working on it at other times of my day so that I could help the pupils finish their work. Decide upfront how much time you can genuinely spare, and hold that line.

4. Gently curb their enthusiasm

Students' ambitions can often outpace reality. In that same competition, I had more than one conversation in which an enthusiastic pupil would suggest something along the lines of building an autonomous robot that would help Granny with her shopping. It's a lovely idea, but not on a school budget, and not in one after-school session a week. These reality checks are hard to deliver, but ultimately kinder than letting a project collapse under its own ambition.

5. Don't take on more than you can support

Some of my teams did brilliantly, reaching the competition finals through sheer determination, and I was genuinely proud of the teamwork and project management they demonstrated. When I spent time looking at the projects submitted by others, it became clear that the strongest teams were the ones who had put in serious hours outside school, often through dedicated code clubs, and had bigger budgets and more expertise than a time-poor, budget-poor teacher like me could ever offer. Know what you can realistically support, and be honest with students about it.

6. Get help

You don't have to do this alone. Parents, older students, or even local university students can all be brilliant extra resources, whether that means running a session, prepping kit, or just keeping an eye on a group while you troubleshoot elsewhere.

> "ENFORCE TIME LIMITS AND TIDY-UP RULES, OR YOU WILL BE THE ONE LEFT PICKING UP THE PIECES

7. Steer away from gimmicks

Not every bit of kit deserves classroom time just because it's shiny and new. I inherited a giant set of traffic lights that only worked with one specific piece of software, and frankly, what pupil wants to play with traffic lights anyway? I got far more mileage from spending a bit of the budget on two small Python-programmable refurbished drones, which were a great hit at school open days and perfect for end-of-term lessons.

8. Consider your storage space

When you get into physical computing in the classroom there will be a lot of kit, and it all needs somewhere to live. Work this out before you commit to buying anything, not after it arrives and takes over a cupboard (like those giant traffic lights).

9. Ground work in real problem-solving

Physical computing works best when it is tied to a practical scenario students can get invested in, rather than being tech for tech's sake. For example, one engaging Year 7 lesson starter is to ask how we can use a micro:bit as an intruder alarm to stop people from stealing sweets from a desk drawer (answer: use the light detection sensor to trigger the speaker to make a sound).

10. Start small and know yourself

Still want to try physical computing? My top tip is to avoid the urge to dive into complex hobbyist projects that need soldering or expensive kit, at least to begin with. The Raspberry Pi Projects site (**helloworld.cc/physical-projects**) is a fantastic place to start.

Be honest about your own confidence and interest, because you do need to enjoy this stuff to teach it well. Fair warning, though: it can be addictive, and I still have a drawer full of my own abandoned physical computing projects to prove it. (HW)



LAURA JAMES

Laura is a learning manager at the Raspberry Pi Foundation, creating engaging content for the Ada Computer Science platform. With over a decade of classroom experience, she brings practical insight from UK secondary schools to her work.



THE FUTURE OF STUDYING COMPUTING

© Seventyfour/stock.adobe.com

With advances in AI, will there still be career opportunities in computing?

The number of students choosing to study computing in school has decreased in the past couple of years (helloworld.cc/computing-drop). While some of this may be a result of the ongoing problem of schools not being able to offer the subject due to lack of specialist teachers, it may also reflect what students have heard about recent trends in AI.





TRACY GARDNER & MICHAEL CONTERIO
Tracy is a computer scientist, tech industry professional, and technology educator, and a co-founder of Flip Computing. Michael is an educator, science communicator, and former physicist, who worked on the courses for the Raspberry Pi Foundation's Training Hub.

Negative messages reaching students

There have been significant and visible media studies about how software development has now been 'solved', suggesting we won't need software developers in the future, and this has been matched by a decrease in the availability of software development roles, particularly at the entry level. Young people are hearing these stories, and they may deter them from choosing to study computing or computer science.

This is certainly a difficult time for students to be making decisions about their future, but we caution against concluding that the study of computer science will not lead to employable skills and enjoyable careers.

Changing roles

Clearly, in many companies the way software is developed has changed dramatically in recent years. Some leading software developers say they rarely interact with code any more, and advanced cybersecurity-focused AI tools have helped to identify a range of previously undetected software vulnerabilities.

This doesn't mean that professionals are doing less technical work, though. Instead, their focus has shifted to thinking about requirements specifications, threat models, and having the right tests in place. There is a move towards product engineer roles that

> WHATEVER HAPPENS NEXT, THE COMPUTING INDUSTRY WILL STILL REQUIRE DIVERSITY OF THOUGHT

combine end-user understanding with higher-level technical work using AI agents to create software. Software developers in these roles are supervising, evaluating, and taking responsibility. Some software developers who enjoyed the previous work may find less enjoyment in this new way of working — but there is work for those who are focused on user needs and business outcomes.

Whenever we move to higher levels of abstraction in software development, we still have some people working at the lower levels. These include people building the tools that the higher levels rely on, highly specialised or innovative solutions, safety- or mission-critical software that needs a high level of human involvement and responsibility, and some areas of academic research. This means that for those who enjoy these aspects of computer science, some work at this level will remain available. Work is being done in the industry to make these new ways of working more about engineering with high-level tools, rather than 'vibe coding'.

AI is affecting multiple subjects

Students may think it's safer to choose another subject, but recent work in the field of maths suggests this is not the best approach. OpenAI has recently been in the news for using AI to brute-force solve many open problems. Mathematicians are now dealing with the fallout from this and trying to work out what it will look like to be a mathematician in the future (**helloworld.cc/math-controversy**). Fields such as law, medicine, and business operations are also being affected, and fields such as physics and engineering could be next to see the introduction of high levels of automation.

How should we respond?

Whatever happens next in the development of AI, the computing industry will still require diversity of thought. We will still need people who understand business and user requirements, as well as those focused on the technical detail of coding. This means that we should still be encouraging students to choose subjects they enjoy and have an aptitude for. There is no guarantee that jobs will remain in any field, so switching subjects purely for employment reasons may not work out well.

Technology is becoming more and more central to modern life. We need more people, not fewer, who can understand it and influence it. If students find computing satisfying now, choosing a career in computing will put them in a good place to drive and direct it in the future. (HW)

AN EXAMPLE FROM PHYSICS



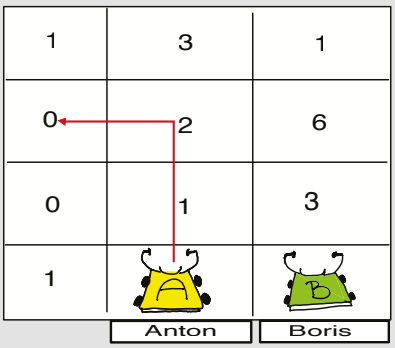
Exoplanets regularly cause the light from their star to dim as they pass between the star and the telescope measuring the light. Human researchers first discovered exoplanets by plotting and investigating these light measurements.

Nowadays, machine learning algorithms can process large quantities of data to accurately identify potential exoplanets, but checking these results means spending time with a sensitive telescope pointing at that star — and that resource is limited. Humans still need to apply their judgement to decide how to allocate limited scientific resources.

BEBRAS PUZZLE

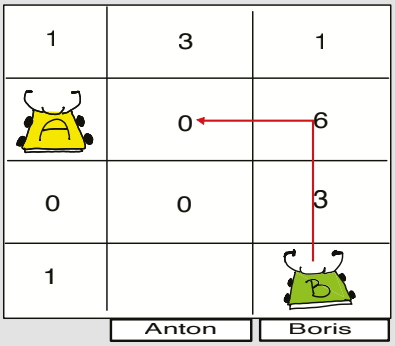
BEBRAS PUZZLE SOLUTION: RUBBISH ROBOTS (PAGE 63)

The answer is 9.



Explanation

- Anton will go first and using the instructions given he will collect 1 + 2 + 0 pieces of litter, that is, 3 altogether.
- Next, using the instructions given Boris will collect 3 + 6 + 0 pieces of litter, that is 9 altogether.
- Note that in the last square on Boris's path, there used to be 2 pieces of litter before both robots started, but Anton has already cleaned this square.



TURNING LEARNERS INTO THE MENTORS OF THE FUTURE IN IRAQ

Dr Alyaa Abdulhussein shares what inspired her to found the Abda3 Skills Club in Najaf, Iraq

My journey into teaching children to code began in 2020. I established a Code Club at the Engineering Technical College at Al-Furat Al-Awsat Technical University in Najaf, where I work. We gradually expanded from basic programming into educational robotics, micro:bit, and Arduino. At around the same time, I started teaching programming to my own children.

A dream that grew bigger

My children, Maryam and Mohammed, began participating in local robotics competitions before the age of ten. After discovering the excitement of coding and robotics, they wanted other children to experience the same joy of learning, creating, and achieving. This inspired us to ask: What if children who have learnt to code could inspire and teach other children?

With the support of our family, Abda3 Skills Club was founded in Najaf in 2025 as a free, volunteer-led educational initiative. Through Abda3, we launched the 1001 Digital Kids initiative, creating a welcoming environment where children could experiment, collaborate, explore technology, and develop their talents regardless of their financial circumstances. Abda3 began as a family learning experience, but grew into a much bigger dream.



Former learners become young mentors

© Dr Alyaa Abdulhussein

Code, build, think

A typical session begins with an exciting idea or a simple challenge that sparks curiosity. We introduce the concepts through accessible explanations before moving on to hands-on activities such as creating games with Scratch, programming micro:bit boards, building Arduino projects, or exploring robotics and AI.

Our approach combines three elements: CODE, for programming skills; BUILD, for hands-on projects; and THINK, for creative thinking and problem-solving. Children collaborate in small groups, test different solutions, and learn from their mistakes. One particularly rewarding aspect is seeing Maryam, Mohammed, and other experienced young learners supporting their peers. At the end of a session, children showcase their projects and share what they have discovered. For us, success means seeing children leave proud of what they have created and excited to explore their next idea.

For me, one of the most beautiful aspects of our club is the sense of belonging. More experienced students help beginners, and everyone has something valuable to contribute. We often see children move from asking, 'How does this work?' to 'Could I create something like this myself?' I imagine that many of our students would describe Abda3 as a place where friendship, creativity, and discovery come together, a place where they realise they are capable of much more than they once believed.

Beyond the trophies

Looking back at our first sessions in 2020, the most remarkable growth has been in children's confidence, independence, and creativity. Many began by following simple instructions. Over time, they started asking their own questions, experimenting with solutions, developing projects, and helping others learn. And the club has developed from small coding sessions to structured programmes covering programming, robotics, artificial intelligence, and cybersecurity.

One of our proudest achievements was winning first place in the Advanced Innovation category at the Arab Robotics and Artificial Intelligence Championship for three consecutive years: Jordan in 2024, Tunisia in 2025, and Egypt in 2026. Winning first place in the Technoxian World Robotics Championship was another highlight.

But some of our most meaningful memories have nothing to do with trophies. They happen when a child successfully runs their first program, proudly presents a project, or begins helping a younger student overcome a challenge. Seeing former learners become



Maryam and Mohammed as members of Abda3 Skills Club with their first place achievement at the Technoxian World Robotics Championship

© Dr Alyaa Abdulhussein


Abda3 is a place where friendship, creativity, and discovery come together

© Dr Alyaa Abdulhussein


Our students learn about coding concepts before moving on to hands-on activities

© Dr Alyaa Abdulhussein

young mentors has been particularly special. I am also grateful to the volunteers, trainers, and families whose dedication makes our free educational activities possible. As a mother, educator, and club supervisor, I believe our greatest achievements are not the trophies on our shelves, but the confidence, creativity, and kindness growing in our children.

Lessons from young learners

Working with children has taught me invaluable lessons. The most important is that children's abilities often exceed our expectations when we give them trust, encouragement, and the freedom to experiment. Every child learns differently. Some excel at coding, while others shine in robotics, design, problem-solving, or team leadership. Our responsibility is not to make every child the same, but to help each one discover their strengths.

I have also learnt how powerful peer learning can be. Watching children become mentors has shown me that teaching is not always about adults sharing knowledge with younger learners. An encouraging environment, in which mistakes are accepted as part of learning, can be more valuable than the most advanced equipment. I began this journey to teach children coding, but soon discovered that the children were teaching me to see education and creativity in a new way.



DR ALYAA ABDULHUSSEIN

Alyaa specialises in information technology and AI and has 20 years of experience teaching at Iraqi universities. She supervises Abda3 Skills Club in Najaf, Iraq, and is responsible for its registration with the Raspberry Pi Foundation.

Advice for starting out

My advice is simple: Don't wait for perfect conditions. When I started teaching my children coding in 2020, I never imagined that those small steps would grow into a volunteer-led initiative reaching hundreds of children. You do not need an advanced laboratory to make a difference, or expensive equipment. Start with a small group, use free educational resources such as Scratch and those provided by the Raspberry Pi Foundation, and create an environment in which children feel encouraged to explore. Don't worry if you don't know every answer. Learning alongside children and discovering solutions together can be a wonderful experience.

Remember that your goal goes beyond teaching code. You are helping children develop confidence, curiosity, and creativity. You may begin with one computer and a handful of children, but you never know how many big dreams will grow from that small beginning.

Even bigger dreams

My greatest hope is to see the Abda3 experience reach more children, especially those who have talent and enthusiasm but limited access to technology education. What began as a family learning experience has grown into a community initiative. Our next ambition is to develop a sustainable educational model that can gradually expand from Najaf to other schools and communities across Iraq. We hope to establish more hands-on learning opportunities in coding, robotics, AI, and cybersecurity, encouraging children to develop solutions to real-world problems. We also dream of building a network of young mentors who return to teach and inspire new generations. Partnerships with educational institutions and supportive organisations could help us achieve this while preserving our commitment to free, volunteer-led education.

Abda3 began with two children who wanted to share their knowledge with friends. We dream of a future in which thousands of children can share knowledge and create opportunities for others. That is the vision behind our motto, 'Building a Generation That Codes the Future.' (HW)



AI tools help students organise their thoughts instead of doing the thinking for them

© EFStock/stock.adobe.com

EMBRACING AI IN EDUCATION

A teacher's perspective

When I was growing up, teachers worried when we used the internet too much for our assignments. If we wrote a paper, we were allowed only one internet reference. Now, my students don't know how to use any resources that are not on the internet. They have never had to go to a library and find encyclopedias, magazines, or books to research a topic. They just go on the internet and find all the sources needed, and teachers think nothing of it.

AI as a learning tool

Teachers of today are worried about AI tools in the same way that teachers of the past worried about the internet. We need to teach students how to use AI products as tools and not cheats. Students need to learn that AI can be useful as a tutor or a template, but it is not a replacement for work.

AI can allow more in-depth learning to occur, just as the internet did for my generation. It gives access to more information and lets the focus shift from routine tasks to deeper learning.

AI tools such as MagicSchool (**magicschool.ai**) help students organise their thoughts without doing the work for them. For this reason, high-school students don't like it as much as they like ChatGPT. My students use it to help organise their notes — it doesn't take the notes for them. I love the insights tool that lets me see how the students are using MagicSchool, as I can then use this information to help direct them towards better research.

Enhancing student engagement

Another AI tool I am in love with is Gemini Notebook (previously called Notebook LM; **notebook.google.com**). I love to teach, and I want to help all my learners as best I can, but I cannot teach every lesson in the best way for every student's learning style. Gemini Notebook lets me take a lesson I have created and turn it into a podcast, a slide deck, a video, a set of flashcards, or a study document. So when a student asks, 'Hey, I am not quite understanding that lesson you taught and was wondering if you might have an infographic I could look at', I can produce one in minutes by simply uploading the resources of the lesson I have already created.

Streamlining teacher workflows

As a teacher, using AI tools lets me focus more on my teaching and less on mundane teacher tasks. I am often asked to write letters of recommendation for my students. MagicSchool lets me input the information I want to share and generates a letter in minutes, instead of taking a large portion of my planning time. I also use it to take an idea for a lesson and turn it into a fully fledged lesson plan.

Embracing the future of learning

AI is an evolving toolset that can support how we teach and how students learn, just as the internet was when I was a student. I hope you try some of these tools out and find that they enhance your teaching. (HW)

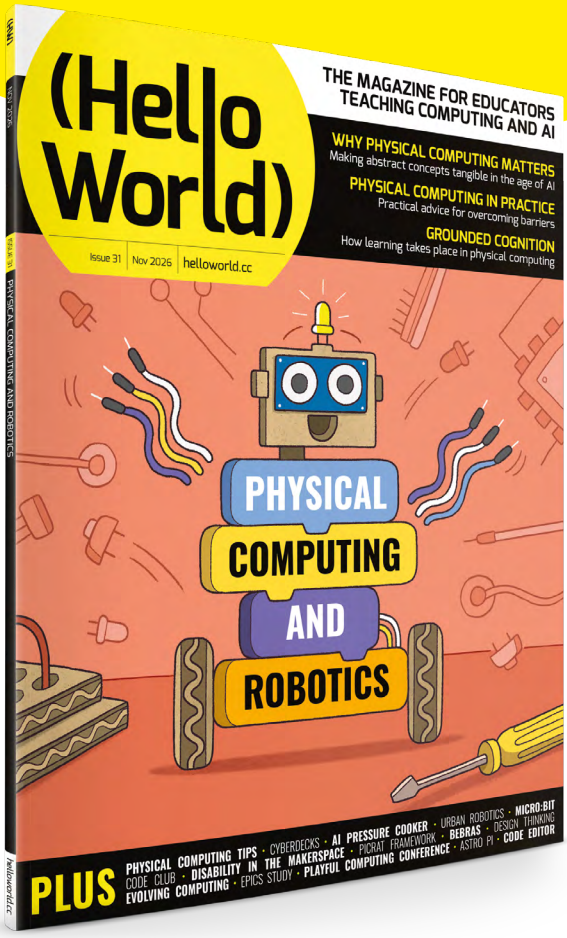


KALA GRICE-DOBBINS

Kala, a teacher in the Madison County School District in Huntsville, Alabama, USA, was recognised as Teacher of the Year for Madison County Schools in 2024–2025 and by the Alabama State Association in 2025. She also earned the distinction of first runner-up nationally for the Air & Space Forces Association's Teacher of the Year award (**LinkedIn: Kala Grice-Dobbins, kgricedobbins@madison.k12.al.us**).

SUBSCRIBE TODAY

FREE IN PRINT for UK-based educators



Why subscribe?

- Never miss an issue of Hello World
- Get notified about our latest podcast episodes
- Exclusive news, the latest research findings, and in-depth features
- Free, convenient, and full of practical ideas you can use straight away

TO SUBSCRIBE VISIT:

helloworld.cc/subscribe

Continue the conversation

Hear the voices behind the stories on our podcast. Search 'Hello World Podcast' on your favourite podcast app, or watch them on YouTube or Spotify.

helloworld.cc/pod

POD SERIES Listen or watch



(Hello World)

helloworld.cc



Raspberry Pi
Foundation

Finished reading?
Please share this magazine with others.