Connect2: A research platform for family computational tinkering

Jonathan Sanderson
Faculty of Engineering & Environment
Northumbria University
Newcastle-upon-Tyne, UK
jonathan.sanderson@northumbria.ac.uk

Abstract*—*In a recent engagement projectError! Reference source not found.***, families were supported to build and program highly personal mechatronic 'puppets'. While a well-received and successful project, families' responses to the coding components bear further investigation. This proposal outlines a 'version 2' hardware platform, which addresses electrical and practical limitations of the original devices, while offering potential for a wider range of programming approaches. We intend to use this platform as a research tool, to explore influences on the adoption of tinkering behaviours for coding activities. 'Computational tinkering' is an emergent field of practice and research, and we believe this work will offer insights for the nascent community.***

I. Background

Connect[1] was a family engagement project run by the researcher in partnership with the Making Studios team at the Life Science Centre, Newcastle-upon-Tyne. Delivered during and in the aftermath of the COVID-19 pandemic, the project worked with a total of 150 families. In a typical workshop, children of late primary age (10–11 years) worked with a parent or other carer to craft a servo and microcontroller-operated mechanical puppet of their own design. The family would code movement sequences which were triggered on receipt of 'mood' messages broadcast by other puppets. Families' creations were ambitious, whimsical, often ridiculous, and sometimes personally meaningful.

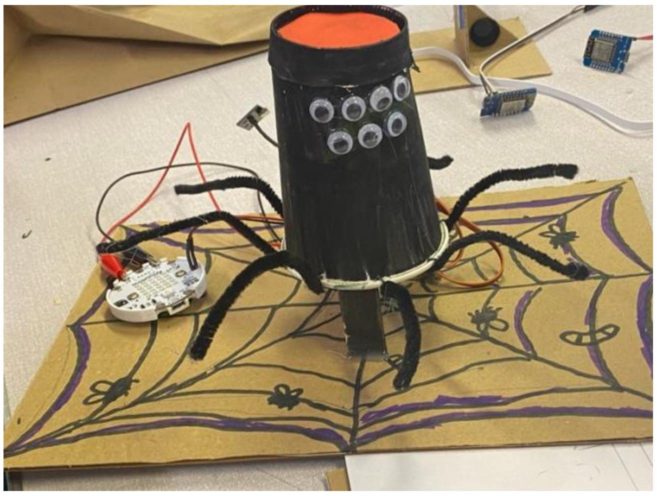

Fig. 1. A Connect puppet, in this case a mildly terrifying spider wearing an unexplained fez. The fez moved independently of the spider.

The hardware platform chosen for the project was Kniwwelino[2], a product of the Luxembourg Institute of Science & Technology (LIST). While inspired by and superficially similar to Micro:Bit, Kniwwelino has key differences which were of value to Connect. Firstly, it is based on ESP8266. This microcontroller integrates basic Wi-Fi support, enabling Connect puppets to continue to communicate with each other once taken home by participants – an idea of key appeal to participants, and to the project's funder.

Secondly, Kniwwelino development is based on the Arduino toolchain, including a LIST-developed, open-source Ardublockly[3] derivative[4]. For Connect, this facilitated a customised deployment which hid or entirely removed many language and hardware features deemed irrelevant or distracting, whilst integrating project-specific language additions. Since the ESP8266 can be flashed over-the-air, the entire toolchain could be hosted by the project and provided as a website, with nothing to install on the wide range of PCs, laptops, tablets and Chromebooks deployed in the schools, libraries and community venues which hosted workshops.

Fig. 2. A family during a school-hosted Connect workshop, programming their cat puppet. The whiskers mechanism was particularly elegant. Note the Chromebook, with no physical connection to the puppet.

The approach was practical and popular with both venues and participants. Almost without exception children were familiar with (and overwhelmingly enthusiastic about) Scratch-like environments and took the peculiarities of the project's implementation in their stride. Following a workshop, participants could also ‘pick up where they left off’: the server retained scripts and related them to individual devices via the hardware MAC address. Participants did not need to take their code with them, the web interface loaded their board’s last point automatically.

A. *Example of target behaviour*

In one workshop, a child and her carer had built a cat puppet, one mechanical component of which was a small cartoon poo which emerged from the appropriate location. The following exchange, overheard between child and carer, illustrates the sort of family learning and exploration dynamic the Connect system was intended to prompt and support:

> *“That looks good.”*
> *“Yeah, but it just comes out.* [mimes with hands] *I think it should turtle a bit… then plop.”*
> *“OK, let’s work out those* [servo] *angles and make some changes.”*

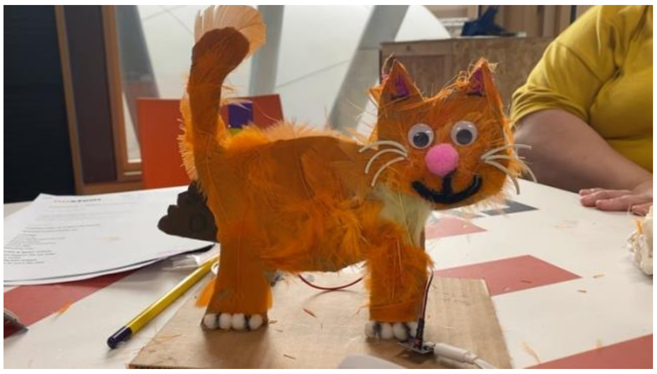
Fig. 3. Another (of many) puppet cats, this one uniquely featuring animated bodily functions (just visible, frame left).

B. *Limitations and challenges*

While successful on its own terms, the first version of Connect had its problems. Powering and controlling 5 Volt devices like servo motors from 3.3 Volt microcontrollers is a challenge with which practitioners and makers will be familiar. A typical microservo has a stall current of perhaps 600mA, while microcontroller power supply is often constrained. ESP8266 VSYS maximum power delivery is not specified[6], but is usually taken to be around 300mA.

In practice, an ESP82666 can commonly drive one servo without issue. Adding a second, however, will often lead to undervolting and/or timing imprecision. To the user, this presents as ‘jittering’: jerky servo movement.

For Connect, we required servo movement to be deterministic, repeatable, and smooth – our servo animation library integrated easing algorithms to allow more natural, organic and expressive puppet movements. While occasionally creatively interesting and even funny, servo jitter was not generally acceptable.

Our eventual solution was to break out a 5V USB supply via a miniature ‘power distribution board.’ Servo and Kniwwelino controller power were drawn from this supply, with grounds tied. While electrically successful – relieved of power supply duties, three servos could be controlled without issue – the resulting wiring was confusing, aesthetically challenging, prone to accidental disconnection, and difficult for participants to reassemble. The distribution boards were also hand-made from stripboard and header pins, taking considerable practitioner time to prepare.

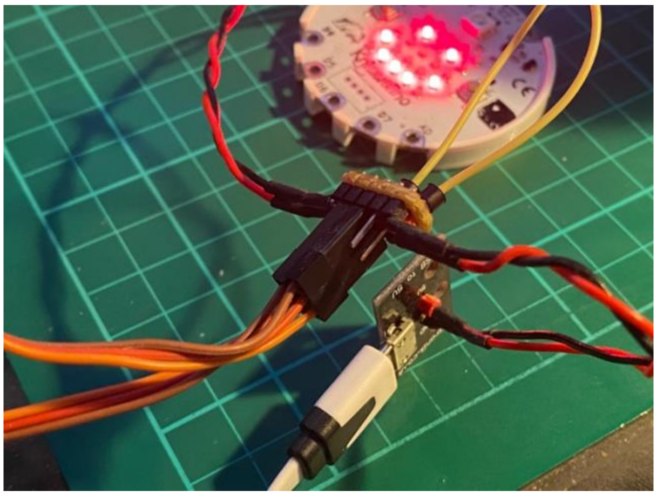
Fig. 4. Kniwwelino/servo wiring harness, as deployed.

A second issue related to the remote compilation/over-the-air flash update workflow. This met its design objectives, but introduced a problem: tinkering approaches often revolve around rapid feedback, trial-and-error, or variations on the predict-run-investigate-modify-make**Error! Reference source not found.** cycle. Mistakes should be easily spotted and quick to rectify. Unfortunately, remote compilation and over-the-air flashing imposes an unavoidable latency to the cycle of around two minutes. Informal observations appear to support the practitioners’ concern that this latency affects participants’ willingness to iterate their code. That is: compilation and flashing latency reduces the ‘tinkerability’ of the system.

The proposed project aims to provide a hardware platform to support research into alternative coding environments, and their impact on participant behaviour and outcomes.

II. Related Work

Berland (2016)[8] explored the relationship between tinkering methodologies and computational thinking, but ‘computational tinkering’ as a phrase and research movement emerged during or shortly before the pandemic.

A significant strand of work grew out of ongoing collaborations between MIT Media Lab (specifically current and former members of the Scratch team) and the Tinkering Studio at the Exploratorium in San Francisco – the latter a nexus of work on tinkering methodologies and practice.

Considerable research interest has focussed on barriers to the integration of computational components into making and tinkering activities in informal learning spaces. For example, Moreno et al (2022)[10] articulated participants’ association of Scratch with school learning, and their expressed wish to learn different computing skills; frustration with artificial or overly-constructed technical challenges, for example the use of a programming environment in place of a simple light switch; and facilitators’ discomfort with computing.

This last continues to be explored in greater detail, for example by Hayden and Roque (2024)[11], who focus on the space as infrastructure, its role in providing a rich, equitable learning environment, and in shaping facilitators' preferences and constraints and hence their practice.

Literature about specific computational tinkering interventions or activities is scant. It's at present unclear if this is a result of the field's recent emergence, project and publication delays resulting from the pandemic, the research community's interest in practitioner challenges, or practitioners' lack of access to research publication routes. Most practical activity appears centred around Scratch, and increasingly the OctoStudio mobile app[12].

A computational tinkering interest group is coordinated by the Exploratorium Tinkering Studio along with members (and former members) of the Scratch/OctoStudio team at MIT Media Lab. Membership of the group extends across at least four continents, with roughly monthly Zoom calls.

III. Prototype

A. *Clarification of problems*

This project has three objectives:

1. Hardware which addresses the servo power/wiring problem.
2. A board which facilitates research exploration of multiple programming approaches, particularly including low iteration latency.
3. Researcher skills development: building confidence and at least minimal competence to design custom boards and commission short-run manufacture would allow more ambitious future projects.

We propose to reimplement a breadboard prototype (see figure 5) on a custom PCB, retaining the entire microcontroller board as a discrete component. This approach meets objective 3 above, with minimal technical risk.

Possible extension objectives are outlined below.

B. *Controller choice & servo wiring*

ESP8266-based devices, including Kniwwelino, typically supply only a 3.3V VSYS line of limited current capacity, as noted. Some other boards, notably including many ESP32-based devices, support a 5V VBUS line, usually derived from the USB input. A similar approach is taken by the Raspberry Pi Pico W and Pico 2 W controllers, which are widely available, including from institutionally-approved suppliers within the EC/UKCA regulatory area.

While no maximum power draw is documented for Pico VBUS[13], in our testing we've seen few issues driving at least three servos from a single controller, assuming the specific servos are tolerant of 3.3V control signals and the USB supply is adequate.

Our proposed circuit design is, therefore, initially rather simple: a carrier for a Pico 2 W board, integrating power circuitry for at least three servos along with other components needed for the Connect system. These include and an off-the-shelf 5x5 LED matrix breakout[14] controlled via I2C, and two physical buttons.

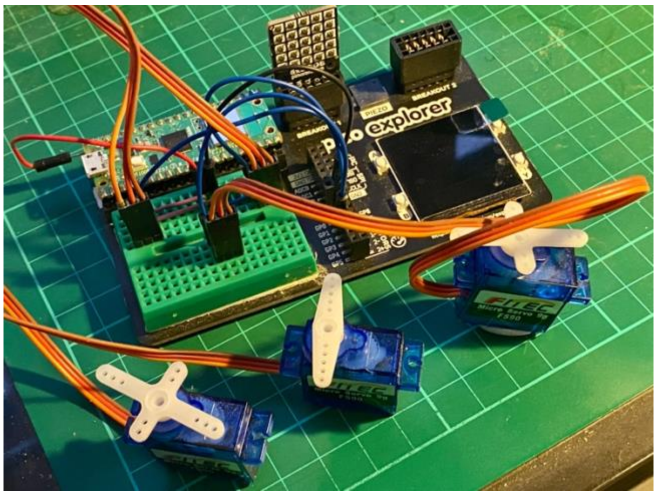

Fig. 5. Representative, partly-functional breadboard prototype. Note the VBUS jump wire (top left) and 5x5 LED matrix breakout (top centre).

C. *Software status*

While not yet ready for workshop use, a basic Micropython implementation of the Connect system exists, running on Pi Pico devices. MQTT is used for message exchange, and the other main component is a custom animation library for servo control. This has been ported from Arduino to Micropython, and while further work is needed the current version is adequately functional. A functional Micropython-based Connect v2 system presents a low development risk.

D. *Coding environment(s)*

As an interpreted language, Micropython immediately avoids the compile/download/flash stages of Arduino development.

Our intention is to test implementations of several alternative coding environments in workshop circumstances, and to observe participants' responses to their respective affordances and challenges. Categories of coding environment might include:

1) *Text-based*

Thonny[15] (desktop application) or ViperIDE[16] (web-based) provide purely text interfaces to Micropython. Considering workshop participants' age, we'd expect this approach to be problematic. However, the benefits may outweigh the challenges. We may also be underestimating children's abilities and resolve, and the impact of the family dynamic: "You type, I'll tell you what to write" might go a long way.

2) *Block-based*

BIPES[17], Microblock IDE[18], and other platforms provide block-based environments. The potential for customisation will require further investigation, however BIPES appears likely to leverage our existing experience of modifying and deploying Blockly environments.

3) *Frame-based*

Strype[19] is a new web-based Python editor, which explores a middle ground between block and text coding. A fully block-based environment is a higher research priority, given participants' familiarity with Scratch. However, a common observation of practitioners designing and facilitating tinkering activities is that over-simplifying elements isn't always productive. While the objective might be to smooth participants' progress, they're typically invested in the process, less averse to problem solving than we might expect, and keen to develop a sense of mastery.

E. Extensions: towards version 3

We envisage the Pico W-based board to be a developmental/test/research tool. As such, it's a deliberately simple board: a reasonable 'first board design' starting point for the researcher. Two possible extensions are apparent:

The breadboard test piece incorporates a 5x5 LED matrix controlled over I2C via a IS31FL3731 driver[20]. This is available as an off-the-shelf breakout from a popular UK manufacturer/retailer. Integrating a similar circuit directly into the board design would be an obvious first enhancement.

Meanwhile, the Connect partners are actively seeking funding to continue the project. If successful, we'd be looking for potentially a few hundred production units. At that scale, it would likely be cost-effective to consider a fully bespoke board. This would entail replacing the Pico W itself with a fully integrated implementation using a bare RP2040 or RP2350 controller, along with a Wi-Fi module such as the Raspberry Pi RM2 (which appears to be the Infineon CYW43439[21]). While considerably more ambitious, we note that a reference board design is available for RP2350[22].

IV. Responsible innovation

One criticism of 'digital making,' and by extension computational tinkering, is that while the digital components can be very cheap, they represent significant embodied energy use. For Connect we considered it important that participants could take their puppets home, complete with the associated microcontroller. While our telemetry data suggest that a reasonable proportion of devices were subsequently connected to the network, two years on from that project it's likely that many devices are either already in or destined for landfill.

As counterbalance, we must consider the benefits to the participant families. Connect workshops were carefully designed and facilitated to support families in overcoming the (considerable) challenges, to attempt something together which felt beyond their individual comfort zones, and to prompt positive memories around individual and family agency, digital technologies, coding, and creativity. While less tangible, the project partners remain convinced of the importance of this work. Host schools and community groups also noted the modelling of response to failure by carers, and the encouragement of children to discuss moods and emotions, as particularly valuable.

These costs and benefits are not directly comparable. Accordingly, we have a responsibility to minimise resource use and maximise participant benefit.

A. Hardware lifespan & reuse

The Kniwwelino boards used for the original Connect could be repurposed, though their relative obscurity in the UK is a limiting factor. Designing hardware for reuse, and fully documenting it, would be important. So too would be preparing additional learning materials. One limitation of Connect was for families responding, "That was awesome – what do we do next?" The original project offered regrettably little answer.

Extension materials and resources would prolong the product life, extracting more value from the hardware, for relatively limited outlay. The Pi Pico ecosystem is particularly attractive in this regard, with an increasing range of hardware, tutorials and learning materials surrounding the board. Ensuring compatibility with that ecosystem would be a positive step.

B. Research & hardware relevance

As a new field, computational tinkering is under-researched. We believe our investigation of coding environments and participant responses would be useful within that researcher and practitioner community, helping to inform future work.

The hardware itself might also be helpful: when Connect was presented to the community (August 2024), addressing the servo power problem was of significant interest to several attendees. While some Pico-based boards do offer servo power supply, these tend to be geared towards more advanced robotics use. As a result, they're relatively costly and Wi-Fi is typically not supported. While we're wary of introducing yet another board to the market, there does appear to be a gap in provision.

V. Author bio / experience

I'm an Assistant Professor in the Faculty of Engineering and Environment at Northumbria, working across the faculty as part of the multi-award-winning NUSTEM widening participation research and practice group. I also teach in the Department of Computer & Information Sciences.

My research interests, pointing towards a some-way-in-the-future PhD by publication, centre on novel and playful pedagogies in computing education, particularly in informal contexts. I'm currently supervising my third PhD student, who's investigating responses to failure during making and computational tinkering activities.

My early career was in television, producing and series producing documentaries, engineering challenge shows, and particularly children's science programming. I subsequently ran a small science communication consultancy, with clients across the UK and internationally. Our work included a popular series of science teacher training films; jump-starting the Royal Institution's in-house video efforts[23]; performer training on three continents; consulting for McKinsey; and assembling a network of several hundred academics to support BBC science production.

I've been active as a digital maker for a little over ten years, building theatre props, household devices, and overly-ambitious installation pieces. The latter have been exhibited at a Raspberry Pi Birthday Party, the Great Exhibition of the North, and Maker Faire UK. Connect grew out of that work, via a pilot 'Digital Making for Families' course in 2018.

A theme through my work since around 1990 has been extending opportunities to children to experience wonder, to notice that the world is comprehensible, and to feel that their actions matter – that they have agency over their lives and surroundings.

VI. Acknowledgements

The Connect project was funded by the North of Tyne Combined Authority STEM & Digital Skills Initiative. It was a collaboration between Northumbria University and Life Science Centre.

My ongoing computational tinkering work is supported by the NUSTEM group, and the Digital Learning Lab in the Department of Computer & Information Sciences, both at Northumbria University.

VII. References

[1] NUSTEM (2020), *Connect: Connecting communities through ridiculous Internet of Things devices*. Available at: https://nustem.uk/connect/ (Accessed 7 May 2025).

[2] Kniwwelino (2017). Available at: https://www.kniwwelino.lu/en/ (Acccessed 7 May 2025).

[3] Ardublockly (2015). Available at: https://ardublockly.embeddedlog.com/index.html (Accessed 7 May 2025).

[4] Kniwwelino development environment (2018). Available at: https://code.kniwwelino.lu/ (Accessed 7 May 2025).

[5] Connect development environment (2022). Available at: https://connect.nustem.uk/ (accessed 7 May 2025).

[6] ESP8266 Hardware Design Guidelines (2024). Available at: https://www.espressif.com/sites/default/files/documentation/esp8266_hardware_design_guidelines_en.pdf (Accessed 14 May 2025)

[7] Sentence, S. and Waite, J. 2017. PRIMM: Exploring pedagogical approaches for teaching text-based programming in school. In Proceedings of the 12th Workshop on Primary and Secondary Computing Education (WiPSCE '17). Association for Computing Machinery, New York, NY, USA, 113–114. https://doi.org/10.1145/3137065.3137084

[8] Berland, M. 2016. Making, Tinkering and Computational Literacy, in Makeology, Routledge 2016 ISBN 9781138847811

[9] Bulovic, K., Bentley, Z. and Rusk, N. 2024. Designing for Tinkerability and Accessibility: Developing the OctoStudio mobile app to engage blind and visually impaired learners in creating with code. In Proceedings of the 23rd Annual ACM Interaction Design and Children Conference (IDC '24). Association for Computing Machinery, New York, NY, USA, 882–886. https://doi.org/10.1145/3628516.3659411

[10] Moreno, C., Hladik, S., Roque, R., & Hayden, R. (2022). Challenges in Facilitating Computational Experiences in Informal Learning Environments. Connected Learning Summit 2022.

[11] Hayden, R. & Roque, R. (2024) Lines of Infrastructuring: Revealing and Tracing Educators' Infrastructuring Practices Across Design Implementations. In Proceedings of the International Conference of the Learning Sciences (ICLS '24).

[12] Bulovic, Katarina, Zoë Bentley, and Natalie Rusk. 'Designing for Tinkerability and Accessibility: Developing the OctoStudio Mobile App to Engage Blind and Visually Impaired Learners in Creating with Code'. In Proceedings of the 23rd Annual ACM Interaction Design and Children Conference, 882–86. IDC '24. New York, NY, USA: Association for Computing Machinery, 2024. https://doi.org/10.1145/3628516.3659411.

[13] RP2350 Datasheet (2023). Available at https://datasheets.raspberrypi.com/rp2350/rp2350-datasheet.pdf (Accessed 14 May 2025).

[14] 5x5 RGB Matrix Breakout (2025). Available at https://shop.pimoroni.com/products/5x5-rgb-matrix-breakout?variant=21375941279827 (Accessed 14 May 2025).

[15] Thonny – Python IDE for beginners (2015). Available at https://thonny.org/ (Accessed 14 May 2025).

[16] ViperIDE (2024). Available at https://github.com/vshymanskyy/ViperIDE?tab=readme-ov-file (accessed 14 May 2025).

[17] BIPES – Block-based Integrated Platform for Embeded Systems (2021). Available at https://bipes.net.br/ide/ (accessed 14 May 2025).

[18] microBlock-IDE (2020). Available at https://github.com/microBlock-IDE/microBlock-IDE (accessed 14 May 2025).

[19] Strype: A Frame-Based Approach to Python (2025). Available at https://strype.org/ (accessed 14 May 2025).

[20] IS31FL3731 Audio Modulated Matrix LED Driver (2024). Available at https://www.lumissil.com/assets/pdf/core/IS31FL3731_DS.pdf (accessed 14 May 2025).

[21] Infineon CYW43439 1x1 Single-band Wi-Fi 4 + Bluetooth 5.4 combo device (2024). Available at https://www.infineon.com/cms/en/product/wireless-connectivity/airoc-wi-fi-plus-bluetooth-combos/wi-fi-4-802.11n/cyw43439/ (accessed 14 May 2025).

[22] Hardware Design with RP2350 (2024), Raspberry Pi Foundation. Available at https://datasheets.raspberrypi.com/rp2350/hardware-design-with-rp2350.pdf (accessed 14 May 2025).

[23] RiChannel. (2011). Available at: https://web.archive.org/web/20121017012121/http://richannel.org/ (Accessed 2 May 2025).