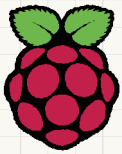
SOCIAL ACTION HACKATHON

FACILITATOR GUIDE



Raspberry Pi

INTRODUCTION

You're about to facilitate a Social Action Hackathon, an activity the Raspberry Pi Foundation team developed to give young people a broad, hands-on introduction to digital making using the Raspberry Pi computer, Agile development methodology, and beginner Python programming.

Your role

As a facilitator, you will support teams of young people through the ideation, design, and creation of an invention which will help beneficiaries of UK charities.

At times, you will be required to speak to the whole group of participants and lead them through discussions that will help them coalesce their ideas into a project. At other times, you will act as an assistant, offering your expertise and support to help teams overcome technical obstacles and to manage their motivation and behaviour.

At no time should you feel like a classroom teacher delivering wisdom and knowledge from on high, or like you are lecturing to the participants.

The hackathon structure

If you've played cooperative board games like Forbidden Island or Pandemic , or tabletop role-playing games like Dungeons and Dragons , you're already on your way to successfully running this Social Action Hackathon: if you treat the activity as a big cooperative game which you are refereeing, then you're on the right track!

First you will be leading sessions in which the participants consider the lives of the people they are aiming to help. You will ask the participants to watch videos about charity beneficiaries and then facilitate a discussion about:

- The relevance of digital making and technology in social action
- Ways in which technology can solve social issues and support the work of charities
- The issues these charities and their beneficiaries face
- Possible solutions the participants could create to address these issues

- The type/style of project that is feasible for teams to build within two days

Next, through discussion and playing a simple game, you will help the participants learn:

- How the Agile development cycle works, and what steps and systems are involved
- How participants will use Agile development practices to create a new product
- How the available technology works and can be used in the context of this hackathon

In the final sessions of the hackathon, the participants will work to create something amazing! During these sessions, your role will be to support teams as they work on their ideas, and to make sure they are motivated and organised. The hackathon will end with a 'pitch session' in which teams explain their process and product to the whole group.

Roles for participants

During each Agile development cycle, participants will take on one of four roles. Each role comes with a set of tasks and responsibilities, which are listed on the lanyards the participants will wear. At the beginning of each cycle, participants will swap lanyards and take on a different role.

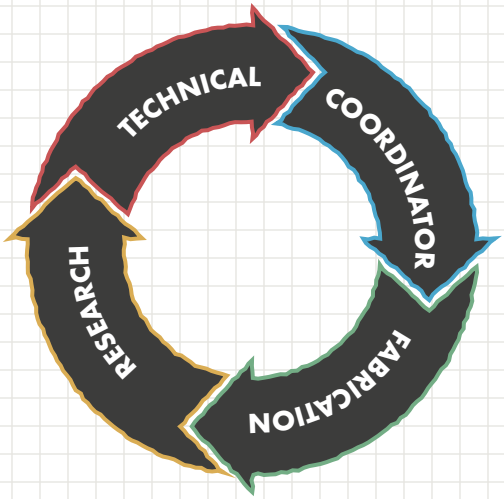
- **Coordinator:** Manages workflow and resources, as well as communication between the team and the facilitators; pitches in to support hands-on work where necessary. In-a-nutshell guidance: "Make sure everyone has what they need to succeed."
- **Fabrication Lead:** Creates the physical and artistic parts of the product that people will interact with, e.g. the case, stand, controls, colours, and artwork. In-a-nutshell guidance: "Make the interesting, colourful, and structural parts of the product that people see, touch, and use."
- **Research Lead:** Supports Technical and Fabrication Leads to find guidance and troubleshoot problems; provides support with pair programming, testing, and quality assurance; acts as

an extra pair of hands when needed. In-a-nutshell guidance: "Troubleshoot technical difficulties, find alternate solutions, and implement these."

- Technical Lead : Works on the internal functions of the product, including the code and the wiring and testing of electronic components. In-a-nutshell guidance: "Get the guts of the product working."

Participants need to form teams of four; if absolutely necessary, teams of five are OK, but teams cannot have less than four or more than five members. If there is a fifth team member, the team will use them as a second Research Lead, so that the Technical and Fabrications Leads each have a Research Lead to support them.

Roles MUST be swapped at the start of each development cycle, and there is a set order in which participants move from role to role:



This order allows participants to move between hands-on roles (Technical/Fabrication Lead) and supporting roles (Coordinator, Research Lead), providing a scaffold for them to understand every facet of the product's development.

1. SOCIAL ACTION HACKATHON INTRODUCTION (WHOLE GROUP)

1. Introduction (3 minutes)

- Get everyone's attention.
- Introduce yourself and the team by playing a 'two truths and a lie' icebreaker:
    - Each member of RPF staff should (in turn) introduce themselves by name and say three facts about themselves, two of which are true and one of which is untrue.
    - Ask the young people to discuss for 30 seconds, then have them guess which the untrue fact is.
    - Once they have guessed, move on to the next staff member.
    - Once every staff members has had a turn, move on.

2. Detail the aims for the hackathon (2 min)

- Over two days, participants will:
    - Form teams to work together like tech start-up companies.
    - Find a problem that they want to solve for a charity organisation.
    - Design a product that can help solve that problem.
    - Build and program the product to make it a reality.
    - Present their product, and how it solves the problem, to the whole group.

3. Explain what is about to happen (5 min)

- Shortly, the group will split into two halves; one half will do a social action workshop; the other half will do the technical workshop to learn about electronics and coding. At the start of the workshops, participants will form the teams in which they will work for the rest of the hackathon.
- The two halves will swap over mid-morning, so that everyone can do both workshops.
- After lunch, the two halves will reunite as a big group, and teams will start to work on their projects.
- Teams will work on their products until after lunch on the second day, when they'll begin putting together presentations to get ready to pitch their product at the end of the day.

4. Split the group into two halves

5. RPF staff split up and lead the two beginners workshops: Technical Workshop and Social Action Start-Up

2. SOCIAL ACTION START-UP WORKSHOP (HALF GROUP)

1. Introduction (5 minutes)

- Put up slide show 2. Social Action Hackathon
- Introduce yourself and the team by name again, to refresh the young people's memories.
- Explain:
    - Participants will find out how other digital makers have been able to help their communities in the past
    - Everyone will discuss their charity partners (or a charity case study) to come up with ideas about how to help people in need using technology.
    - After participants have settled on a problem they want to address, they'll play a game to get an introduction to a project management technique called Agile, which real tech start-ups to create new technology.
    - **If this is the group's first session:**
      Right now, participants need to form teams of four.
      Note: there can be some teams of five, but only if it is absolutely necessary — this programme works **far** better with teams of four.
- Once the young people are in teams and seated, hand out in pairs:
    - 2 tablets per group
    - Splitters and headphones (encourage the young people to use their own headphones if they have them)
    - One 'Starter pack' per group

2. Discussion: social action through digital making (15 minutes)

- Explain to the young people that they are about to watch a video that shows how technology benefits people with special needs.
- Ask the young people to watch the video on their tablets and see if they can work out how technology is helping people who are in difficult situations or have specific needs.
    - Young people should write down all the different ways in which they can see technology helping the beneficiary in this video.
- Show the video: The near future — a better place
    - Young people have headphones and tablets and can watch in pairs.
    - If a projector is available, you can show the video to the whole group if you prefer.
- Discussion: how can technology help people with special needs?
    - This discussion should show the young people that:
        - Technology is most helpful when it automates tedious, difficult, or uncomfortable tasks.
        - There are many ways in which technology can be used to improve the quality of life or situation of people with special needs.
        - Some solutions require human control and interaction to operate, and other solutions are designed to be passive or invisible to the user.
- Show the slide show of previous projects from NCS workshops, explaining very briefly what each one is.

3. Action: who are we helping? (20 minutes)

Participant guide sections needed

- Reflection questions
- Project plausibility checklist

If young people have not met with their charity partner (this is most likely), each team should watch one of the case study videos (provided on their tablets) about possible beneficiaries of their start-up, and discuss the video using the reflection questions. **Ensure they record the name and function of their charity at this time.**

If young people have visited their charity partner, they should call to mind what they found out and what the experience was like, and then use the reflection questions in their participant guide to discuss their charity partner interaction as teams.

During this time, facilitators and mentors should:

- Maintain young people's focus and provide conversation prompts to help the young people identify the needs of their charity partner.
- If necessary, maintain/manage realistic expectations for project ideas using the project plausibility checklist.
- Ensure that participants are taking good notes and engaging in the design process.
- Ensure young people have access to all handouts/resources.

4. Plenary: decide on an issue to tackle (5 minutes)

Looking at their answers to the reflection questions , teams should now discuss what issue they'd like to tackle using digital making, and develop some broad ideas about what sort of solution they'd like to make.

Before moving on, ensure that teams have a fairly clear idea about what problem they are going to address to help their beneficiaries, and about the type of product they could build for this purpose.

3. INTRO TO AGILE DEVELOPMENT

1. Introduction (5 minutes)

Load slide show 3. Intro to Agile Dev

Explain to the young people:

- Over the next couple of days, they are going to be engaging in product development. That's the process of going from an idea, to a design, to making something they can demonstrate.
- To do that, they are going to use a development methodology called Agile. Agile is the standard method of development in many tech industries and used by most software and technology companies the world over.
- Here is a video showing what it means to undertake Agile development.

2. Discussion (15 minutes)

- Show the video, and then guide participants' discussion and answer questions about the structure and format of the hackathon.
- Answer any questions from YP regarding the Agile process as we will be engaging in it.
- If **Necessary**: Explain the four roles teams will be using during the development cycle, showing the relevant slide:

What roles are part of this Agile process?

Coordinator

- In charge of managing planning and workload requirements and external communications.
- Responsible for chairing the startup at the beginning of a sprint.
- Responsible for managing workload and resource allocation for the sprint.
- Responsible for seeking outside help and communicating issues to facilitators.
- Should lend a hand and get involved in making the project if the startup is experiencing time pressure or complications.

Technical Lead

- In charge of wiring, coding, and creating the internal workings of the device, according to the plan.
- Responsible for bringing any technical issues or questions to the Coordinator.
- Responsible for working closely with the Research Lead to find new solutions and work through the backlog of tasks.

Research Lead

- In charge of finding new solutions and answers to technical questions using online searches and resources.
- Responsible for helping Technical Lead implement complex solutions where necessary.
- Responsible for liaising between Technical and Fabrication Leads to ensure that the interface/housing will work and fit the device.
- Can be used as flexible workforce if the startup is experiencing time pressure or complications.

Fabrication Lead

- In charge of creating the physical interface and the art-and-craft portions of the project.
- Responsible for collecting and organising materials and components necessary for the project.
- Responsible for working closely with the Research Lead to make sure that what they are building is fit for purpose.

3. Action (25 minutes)

- Explain that the young will now play a game demonstrating how the Agile development process works: the airplane game. (Instructions included as Appendix A)
- Give prizes/stickers to the team that wins the airplane game.

4. Plenary (5 minutes)

Take questions from the young people about the process and clarify any misunderstandings or answer queries. Remind the young people that they will all have set roles during each Agile sprint cycle, which will change each sprint — everyone will get a turn at every role.

Let the young people discuss for two minutes which roles they feel good/uncomfortable about undertaking so that they have an idea of which role they'd like to try first. Then send them on to the next part of their day.

4. TECHNICAL INTRODUCTION WORKSHOP (HALF GROUP)

1. Introduction (5 minutes)

Load slide show 4. Physical Computing with Python

- Introduce yourself and the team by name again, to refresh the young people's memories.
- Explain:
    - We are about to do some coding and make some machines!
- **If this is the group's first session:**
    - Right now, participants need to form teams of four.
    - **Note:** there can be some teams of five, but only if it is absolutely necessary — this programme works far better with teams of four.
- Once the young people are in teams and seated, hand out to each team:
    - 2× pi-tops
    - 2× mouses

2. Discussion: What is physical computing? (10 minutes)

- Explain that physical computing is about using computers and code to control things in the real world.
- Give a few simple physical computing examples, such as:
    - Digital thermostats to control the temperature of a room
    - Motion-sensing CCTV systems
    - Timed or motion-sensing night-lights
- Have participants think of some other examples.

- Show participants the range of components available (show slide 2), and give a brief intro for each:
    - LED — small, low-power light bulbs you can flash or illuminate
    - Button — allows physical, interactive control by humans to start/stop functionalities of a product
    - Buzzer — makes a noise
    - PIR — motion sensor
    - UDS — distance sensor, uses ultrasonic waves to detect how far away something is (range is less than 70 cm)
    - LDR — detects whether it's light or dark
    - Raspberry Pi Camera Module -- takes photos and video
    - Servo motor — moves/turns attachments small distances
    - Sense HAT — an add-on board that has an LED array, a joystick/button and a whole array of sensors to detect environmental/motion conditions
    - Tell the young people that, on their tablets, they can find videos that explain each component in more depth.

3.What are Raspberry Pi, Python, and Mu? (10 minutes)

Show slides 3–6

Discuss the Raspberry Pi, the pi-top, the GPIO pins, and Python.

- Participants can now boot up their Raspberry Pis. Encourage them to explore the menu to see the range of applications on the OS, similar to any other computer they might have used. **During this time, Facilitators should be troubleshooting any issues that arise with the machines.**
- Explain that 'GPIO' stands for 'general purpose input and output', and that the Raspberry Pi's GPIO pins can be used to control output components (e.g. LEDs, buzzers, motors, etc.) and to read input components (e.g. buttons, LDRs, and motion sensors).
- Show the participants how to start the Mu editor, write a 'hello world' program in Python, and run this command:

```
print("hello world!")
```

4. LEDs, buzzers and buttons (30 minutes)

Show slides 7–9

- Briefly explain the breadboards and hand out LEDs, jumper cables, and buttons.
- Show the slide with the code snippet that lights up an LED.
- Support the young people to use the jumper cables to connect components to specific pins on the Raspberry Pi, and then work towards using Python code to control the LED.
- Add a button to the board and control the LED with the button.

5. Sense HAT (30 minutes)

Show slides 14–21

- Talk through the range of sensors on the Sense HAT, and the LED matrix, and then explain how to attach the HAT to the Raspberry Pi.
- Participants can explore the range of ways that the LED matrix can be used to communicate information to a user. This can include scrolling messages, illuminating specific pixels in a range of colours, and displaying specific symbols or images.
- Give participants an opportunity to discuss how and why it might be useful to communicate information through such a simple medium, and what its strengths and drawbacks might be.
- Participants can explore the various atmospheric and orientation sensors, then use them to display real-time information on the LED matrix. Ask them to use the LED matrix to respond to changes in the environment, such as a sudden change in orientation, or an atmospheric reading reaching a critical value.
- Give participants an opportunity to discuss where and when it might be useful to monitor the atmosphere or the motion of an object.

Show slide 22: Recipes

- Give participants an opportunity to discuss how these techniques/components could be incorporated into real-world solutions to specific problems. Also provide them with a list of other available components, and ask them to think about how these could be used in physical computing solutions.

6. Plenary: pack down (5 minutes)

Show slide 23

Have the young people save all their code, and demonstrate how to shutdown the pi-tops so these are ready for the next session. Then send them on to the next part of their day.

5.INITIAL AGILE SETUP (WHOLE GROUP)

1. Introduction (5 minutes)

Load slide show 5. NCS Agile Development Setup

- Get the young people into their teams and explain that it's now time for them to decide what their social action product is going to be, to prepare their resources, and to get ready to build.
- Before they can start building, they need to:
    - Choose a single idea for their product.
    - Break it down into individual tasks.
    - Set up their backlog to keep track of their work.
    - Choose roles for the first sprint.
    - Have their first stand-up meeting.

2. Design (20 minutes)

Participant guide sections needed

Project plausibility checklist

Product planning sheet

1. In their teams, the young people should now use the notes and ideas they recorded during the social action start-up workshop to come up with three ideas that could help solve the problem they identified.
2. Using the product plausibility checklist in the Developer Guide, the team should discuss the relative merits of each idea and settle on the one they want to move forward with.
3. Once the team has chosen an idea, they should draw their product on the product planning sheets and answer the following questions:

- What functionality will your product have? What will it do?
- How will your product help solve the problem you have identified?

- What components will your product require?
- Will your product need an internet connection to function?
- What are you going to call your product?What components will your product require?
- Will your product need an internet connection to function?
- What are you going to call your product?

3. Kanban board setup (25 minutes)

Participant guide sections needed

Product planning sheet

Backlog breakdown — user stories

Explain:

- Teams will now break their project design down into all the tiny jobs required to get from the product idea to a working product they can demonstrate to the whole group.
- This is not a complex process, but it will require you to work together, get into the shoes of your beneficiaries or users, and think about what is really important to your product.
- By telling themselves what professional product developers call **user stories**, they will look at how they want their product to work and then start creating the necessary functionalities, tying them together, and testing them.

Show the slide about user stories (slide 5)

- Talk the young people through the steps on the slide and explain the following user story.

A start-up wants to create a selfie booth for elderly people in a care facility, and their ideas for the product are:

1. The booth should stand in the common room of the care facility, ready for elderly people and the visitors they'd like to take a picture with.

2. The machine should be displaying simple instructions on a screen that explain how to take a selfie.

3. When the elderly person pushes a button, the machine should count down from 3 on the screen, then take a picture and post it to the facility's Snapchat account.

4. The machine should display the selfie on the screen with a thank you message to let the elderly person know they have successfully taken a photo.

Explain:

- Looking at this user story, someone creating the product can begin to think about the build in practical terms by picking apart all the functionalities required.

- Teams should look at all the things they want the machine to do — not just the user-facing functions! There are a few extra things the designers need to implement that, if they work, a user will never see, in this case connecting the product to the internet, and linking to the Snapchat account.

For this selfie booth build, the list of necessary tasks is:

1. Create a stand/housing/interface for the product so that it looks inviting and is strong enough to withstand daily use.

2. Create user instructions that will appear on the screen when the machine boots.

3. Wire up a button that can be pushed to make the machine take a picture.

4. Test that the button is wired correctly and works.

5. Connect the camera that will take the picture.

6. Test that the camera is connected correctly and takes images the right way round (i.e. not upside down).

7. Connect the button push to the image capture.

8. Test that a photo is taken every time the button is pressed.

9. Connect the product to the internet using WiFi.

10. Install the software needed to access Snapchat.

11. Connect the product to the facility's Snapchat account.

12. Test that images are being sent to the account when the button is pressed.

13. Create a thank you message that will be displayed along the captured image.

14. Make sure the photobooth restarts its program once it has taken a picture.

Remind the young people:

- For every task they want their product to perform, they need to not only complete the job but also make sure that the functionality they're building works the way they want it to, every time.
- They need to test every component they add to their product, once before they link it into the rest of the system, and once again after they have linked it to the system.

- The teams should now use their notes on the product planning sheet to tell themselves a user story for their product, and write down on their user story sheet all the tasks required to create the product.
- Next, they need to split the tasks into two categories: technical and manufacturing.
- Then they should grab sticky notes — with two different colours for the two different categories — and write each task on a single sticky note according to its category.

- The teams should then set up their Kanban board, taking a whiteboard and dividing it into three columns: 'To do', 'Doing', and 'Done'.
- Finally, the young people should stick all their sticky notes in the 'To do' column of their Kanban board

- The teams should then set up their Kanban board, taking a whiteboard and dividing it into three columns: 'To do', 'Doing', and 'Done'.
- Finally, the young people should stick all their sticky notes in the 'To do' column of their Kanban board

4. First stand-up (10 minutes)

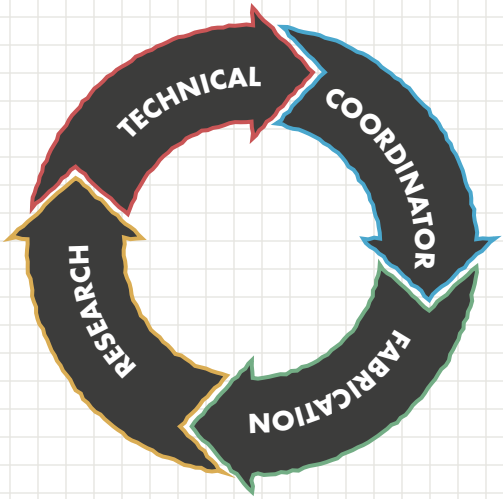
See 'Sprints' facilitator notes

5. First development phase (30 minutes)

See 'Sprints' facilitator notes

6. SPRINTS (WHOLE GROUP)

NOTE: All young people **MUST** change roles each sprint, and the role order is:



It doesn't matter with which role they begin, but they must follow this order in their sprint cycle.

1. Stand-up (10 minutes)

Participant guide sections needed

- Stand-up checklist

Every sprint **must** begin with a stand-up meeting in which the team members use the stand-up checklist to:

- Take stock of their progress in the last sprint as a team
- Hand over their role to the next person who will undertake it, with an explanation of the state of the current task they are completing
- Receive a handover explanation of their new role and where they need to begin work from the person previously in that role

- Swap lanyards over to denote their new role in the team
- Understand what is most important to achieve in the current sprint

During the team's stand-up meeting, facilitators and team leaders should spend some time with their teams to:

- Assist the Coordinator to run the meeting according to the stand-up checklist
- Ensure that discussions are moderated and productive
- Manage behaviour and motivate team members to participate
- Provide advice and expertise when requested
- Maintain realistic expectations of team members about the feasibility of their product/workload

The stand-ups should become more streamlined as the teams get used to the process, possibly allowing for more working time each session. Don't make the team take ten minutes if you feel they have achieved everything they need to, but don't allow them to skip parts of the process either. **Without a stand-up, a sprint will be largely unproductive.**

2. Development phase (80 minutes)

This is the teams' time to work through their backlog of tasks. Make sure that Coordinators and Research Leads are not idle and instead supporting their team and lending a hand to the primary workers wherever necessary.

During this phase of the sprint, facilitators should be in a wholly supportive role by:

- Helping teams (on request) to find solutions to technical problems through their own knowledge and supporting/directing online research
- Managing behaviour of participants to ensure that everyone is on track and working without disrupting other teams
- Checking in with teams to ensure that they produce a product prototype, and that workload is being distributed sensibly and fairly

- Ensuring that Kanban tickets are being moved from the backlog as team members undertake and complete tasks (suggest that moving tickets can be done by **either** the relevant Lead **or** the Coordinator, but by not both)
- Checking in with teams to, if needed:
- Suggest iterations and advancements in functionality to Technical Leads
- Encourage the Fabrication Leads to make improvements to the aesthetic or user interface of the project
- Assist groups with user testing and quality assuring their work
- Support teams to get materials and components they require to do their work

NOTE: Make sure that **only the Coordinator** comes away from their work to speak with you or get your attention on behalf of their team. **This is one of the dedicated jobs of the Coordinator** and should only be done by them. Of course, once you are with their team, speak with whoever needs assistance.

FINAL SPRINT (WHOLE GROUP)

1. Introduction (1 minute)

NOTE: at the end of this session, teams MUST have a product they can demonstrate during their pitch. If they need to remove or stop work on functionality in order to have a Minimum VIable Product, this is their call to make.

Gain attention of the room and display the final sprint slide on the main screen.

Explain that during this stand-up, the teams will need to:

- Discuss the content of their pitch and how they will deliver it
- Create tickets for all the tasks required to create the pitch, and add these to their Kanban board
- Assign the new workload to team members to complete alongside any remaining work on the product itself

2. Final stand-up (15 minutes)

Participant guide sections needed

Pitch questions

Stand-up checklist

This is the last sprint before the pitch session. As such, the stand-up is a little longer to allow teams to add new tasks for creating the pitch and to assign workforce to handle the final backlog.

Teams should begin this stand-up by using the pitch questions in order to:

- Decide upon the content of their pitch and any visual aids required
- Discuss how they will present their pitch in the next session
- Create new backlog tickets for all of the new tasks related to the pitch creation

Then the teams should go through the familiar stand-up process using the stand-up checklist (and your guidance, if needed) to decide upon their **Minimum Viable Product** for this final sprint. The MVP is the most advanced, **stable** iteration of a product with all current functionality completed. In this last sprint, teams need to:

- Include as many stable functions as they can in the product version they will pitch
- Remove (or hide) any unstable or unfinished functions or features before the pitch
- Abandon the addition of any new features if these would be too complex to finish in time
- Direct any spare workforce towards:
- Assembling the necessary resources for the pitch
- Tying up loose ends on the Kanban board

3. Final development phase (70 minutes)

This time is for the teams to work through the final tasks on their Kanban board to create their MVP and prepare for their pitch. They and you should proceed as in previous sprints.

If it looks like a team will be unable to finish their product to the level they are aiming for:

- Have them prioritise the creation of their pitch content and visual aids over new functionality or changes to the look of the project
- Talk to the Technical Lead and Coordinator about what exactly the MVP (the most advanced stable build) is that they will be able to present at the end of this session, and have them stop work on any further development beyond this
- Ensure that teams do not feel bad about being unable to achieve their planned outcome, and encourage them to discuss in their pitch why they couldn't manage it

4. Consolidation of work (5 minutes)

Teams should use this time at the end of the final sprint to consolidate their work so that it is ready to demonstrate during their pitch, and to pack away any equipment they won't need in the pitch session. Any teams that have not finished their at this time **may not work through the pitch session.** The time constraint is an important one, because it forces teams to decide upon a minimum viable product at the beginning of the final sprint. It would also be disruptive, and disrespectful to other teams, to work while others are pitching.

THE PITCH (WHOLE GROUP)

This session is exclusively intended for teams to:

- Present their finished product to the whole group in a two-minute pitch.
- Reflect upon their process of design and creation over the last two days.
- Receive feedback on their work and pitch.
- Provide feedback to other teams on their work and pitches.

1. Introduction (5 minutes)

Get the attention of the room, show the pitch slide, and explain:

- This session allows teams to give and receive positive feedback on their work and process.
- Participants are not allowed to work on their product or pitch while others are pitching.
- Every team member must say something during the team's presentation — no silent participants!
- After each team's presentation, everyone will be able to ask them questions. Participants should be respectful to the presenting team — they'll have to answer questions on their turn too!
- Instruct participants to pay attention to all the other groups' presentation, as they wish the others to do while they themselves are presenting.

2. Pitching (70 minutes)

Select teams to come to the front and present their pitch to the group. Each team has two minutes to explain and demonstrate their product and tell the team's journey during its development; if needed, allow them to run over a little, but stop them at three minutes.

After each group has pitched:

- Take three questions from the audience about the project and pitch, and moderate the discussion of the answers.
- Ask two audience members for 'two stars and a wish': they should name two positive elements of the product and one which could use improvement.
- Ask the next group to present.

3. Closing (15 minutes)

Once all teams have presented their pitches:

- Show the closing slide.
- Thank participants for joining the hackathon.
- Explain that there are lots of opportunities for them to grow their digital making skills at CoderDojos, which are free, local clubs where young people come together to make cool things with code.
- Tell young people you are available after the session to answer questions about CoderDojo and further opportunities for digital making in your local area.
- Have participants pack away and return all equipment.
- Dismiss the participants.
- Refill starter packs with necessary supplies, including Developer Guides.
- Pack everything away ready for storage between this hackathon and the next.

APPENDIX A: AGILE AIRPLANE GAME – INSTRUCTOR NOTES

Step 0: Materials prep

Show the Airplane Game slide (slide 12)

- Prepare points spreadsheet on screen — input team names, adjust document zoom
- Pass out supplies (paper, scissors, coloured markers)
- Pass out plane plan booklets to each team: 'your initial setup'
- Hand out note cards to mentors/facilitators to remind them of judging criteria

Step 1: Set the stage

Everyone here is part of the Agile Aviation Company; their job is to produce the best paper planes in the world.

The AAC produces planes of three levels of difficulty, and more difficult ones are sold for more money.

You will be working in 5-minute sprints to make as many planes as you can: 1 minute of discussion time where you will make a plan for the sprint, and 4 minutes of plane-making time. **You cannot begin construction of any planes during planning time.**

We will stop at the end of each sprint to assess your production and award points based on how many planes which fulfil the user requirements you have created.

Step 2: Starting details — the rules

- Customers across the country have placed orders for planes, and we need to fulfill them as quickly as we can.
- You must colour-code all planes before handing them in (based on difficulty: blue = beginner, green = intermediate, red = advanced) — a stripe or symbol is fine.
- Sprints are 5 minutes long: 1 minute of planning time, during which nobody is allowed to do any work on the planes, and 4 minutes of actual making time.

- **Beginner planes are worth 3 points, intermediate planes are worth 5 points, advanced planes are worth 10 points.**
- At the end of each sprint, we will see how many planes fit the user requirements and award points to each team.
- **Only half of your team can be folding planes each sprint,** but the others are allowed to lend support by colour-coding the planes, cutting paper, or making sure the planes are '*done* done' and ready for testing.

Step 3: First order, begin sprinting

- Give teams 1 minute to plan the sprint — **no making allowed during this time!**
- Run sprint for 4 minutes.

Step 4: End of sprint 1

- See if planes fit the user requirements (instructors/facilitators — depending on your group size, this may require Team Leaders to help assess half the group and report scores).
- Only tell teams about user requirements (=acceptance criteria) if/when they ask.
- Reject planes (by spectacularly ripping them) that don't meet **acceptance criteria:**
    - All the plane's folds line up with 3mm tolerance. (Don't be too stringent.)
    - The plane is properly colour-coded.
    - The plane is folded properly (no inverted folds, ...)
    - The plane must fly over a 6ft (2m) platform (a table usually works fine) with minimal effort
- Update points chart for sprint 1 based on completed planes that passed the test

Step 5: Run sprint 2

- Rotate team roles — the other half now folds the planes!
- Give teams 1 minute to reflect and plan
- Give them 4 minutes of making time
- Test finished planes
- Update points chart for sprint 2 based on completed planes that passed the test

Step 6: Run sprint 3 — change in demand

- Rotate team roles
- **Remove the most commonly made types of plane (likely Arrow and Condor) from production — safety recall on those models!**
- Give teams 1 minute to reflect and plan
- Give them 4 minutes of making time
- Test finished planes
- Update points chart for sprint 3 based on completed planes that passed the test

Step 7: Run sprint 4

- Rotate team roles
- Give teams 1 minute to reflect and plan
- Give them 4 minutes of making time
- Test finished planes
- Update points chart for sprint 4 based on completed planes that passed the test
- See who won!

Plenary — discussion

Observe the graph on the scoresheet and ask the group:

- What was the best move? Making lots of easy planes, or one hard one? Why? (Answer: Making lots of easy, small ones, because opportunity cost is less on failure.)
- Did anyone use their support team (the non-making half) to cut paper or do anything to take up available workload?
    - **If yes, that's good.** Encourage this as a good way to improve efficiency and keep things moving along in your workflow during the hackathon.
    - **If no, that's bad.** What were you doing with your time when you could have been helping? Cheerleading? Not a good use of your very limited time.
    - **Advice:** During a hackathon sprint, whenever you're a Coordinator or Research Lead, find things you can do to support your team — get extra bits of work completed, find better solutions, or add more twinkly lights!
- Did you notice your productivity going up across the sprints as you got more efficient and into a flow/rhythm?
- Did you feel it when the goalposts were moved, and the easy planes were taken away? How did it affect your rhythm/flow?

NOTES FOR INSTRUCTORS/HELPERS:

Customer requirements:

- All the plane's folds line up with 3mm tolerance.
  (Don't be too stringent.)
- The plane is properly colour-coded.
- The plane is folded properly (no inverted folds, ...)
- The plane **must** fly over a 6ft (2m) platform (tables or chairs)

Test infrastructure:

Setup a stable surface (tables or chairs) about 6ft (2m) wide for the planes to fly over.

Points:

At the end of each sprint, after plane testing and acceptance/rejection, add the points in the spreadsheet — the graph will auto-populate accordingly.

APPENDIX B: COMMON BUGFIXES & ERRORS

1. Code

Diagnosis:

a. RTFM — what does the error log tell you? (Look below for some common error messages you may not have seen before.)

b. Check for syntax/typos — encouraging pair programming with a typist and an overseer can help you avoid this.

c. Pin numbering — are the right pins addressed in the code? Is something wired to the wrong pin?

d. Check for duplicate code — has the team entered the import line repeatedly each time?

e. Filename — What have they named their file? Naming it the same thing as an existing library (e.g. picamera.py, python.py, pygame.py) can confuse their computer.

f. Add print commands through your script in key places — like at the top of functions or after significant unseen actions like sending messages or alerts. This will tell you how far your program got before failing and help you narrow down the options where something is going wrong.

Common error messages:

- **EOL while scanning string literal** — you forgot quote marks at the start or end of a string.
- **expected an indented block** — you haven't put an indent after a colon somewhere
- **unexpected indent** — you have an extra indent somewhere, or your indentation isn't always 4 spaces
- **Can't convert 'int' object to str implicitly** — you've tried to print or use a variable as a string somewhere. You need to stringify it with the command `str(variable_you_are_using)`.

2. Hardware

a. General:

i. Are the connections in the same row on the breadboard?

ii. Are the connection on the right pins?

iii. Are the jumper cables dodgy/poorly plugged in?

iv. Is the protoboard pushed all the way into the hub?

v. Is the gender-changer pushed all the way into the protoboard header?

vi. Check specific errors of component, as detailed below.

vii. If all else fails: swap it out, sticker it, and send the faulty bit back to the office with the SD cards for testing and replacement.

b. LEDs:

i. Need to be the right way around — long leg to positive!

ii. Check for dark spots in the LED — cloudy means it's been blown. Are they using a resistor?

iii. Dim LEDs may mean weak connections inside the protoboard, or a dodgy jumper cable.

c. Buttons:

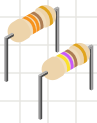
i. Are they pressed far enough into the holes on the protoboard? Sometimes you need to make sure there is a bit of a 'clunk' or positive contact when pressing the buttons in. It may take a bit of force.

ii. Are the wires connected to the right legs of the button? (Same side — both either top or bottom!)

iii. Test whether the contact inside the button is gone by touching the two jumper cables connected to the button to one another. If the thing works, the button's internals are broken, so swap it out.

iv. Can you feel a positive click when you press the button? If not, swap it out.

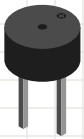
d. Resistors:



i. Too large a resistance will dim the LED. 100 or 150 ohm is fine.

ii. Make sure the resistor isn't shorting your circuit because it's touching something else.

e. Buzzers:

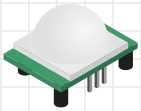


i. Are they the right way around? You can see the tiny positive symbol on the top — it also has the same 'long leg/short leg' setup as an LED.

ii. Listen closely — if everyone is using their buzzer, yours may be drowned out.

iii. Test the buzzer by connecting the positive terminal to GPIO 1.

f. PIR:

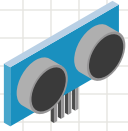


i. Are the connections going to the right pins?

ii. Are both the dials turned all the way anticlockwise?

iii. Cover the PIR with a jumper/box before you run the script. Let the PIR normalise for a few seconds (10–20s) before you try to test it. Is it still firing anyway? Then swap it out.

g. UDS:

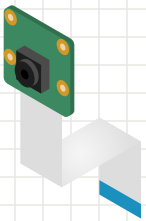


i. Are the connections going to the right pins?

ii. Are your resistors the right impedance? Too much may not give a proper reading, or may stop the UDS from working.

iii. Are the resistors making contact inside the breadboard? Sometimes they crumple instead of going into the port properly.

h. Camera:



i. 'Out of resources' error: You're already using the camera somewhere else. Kill all instances of your running scripts, edit cron (if you've got a boot script set up) by commenting out the command line at the bottom and then reboot.

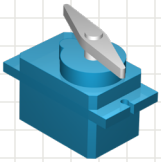
ii. 'Could not initialise' error: Check your connections, especially the sunny clip under the lens and the clip from the camera to the ribbon. Both sides need to be pushed in tight.

iii. Camera **not** enabled: Shouldn't happen; but if it does, run sudo `raspi-config` in the terminal. Go to I**nterfaces>Camera>Enable** and reboot.

iv. 'Could not enable/failed to enable' camera error:

1. Check that the sunny clip under the lens is tight.
2. Check that the clip connecting the board to the ribbon is tight.
3. Check the ribbon is not frayed, snapped, or crumpled.
4. Reboot the Pi.

i. Servos:



i. Servo judders/twitches:

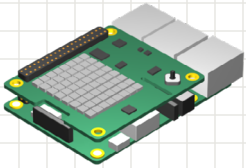
1. Start the pigpio daemon:
   a. Open a terminal window
   b. At the prompt type:
   `sudo systemctl enable pigpiod`
2. Set the GPIO Zero pin factory to `pigpio`:
   a. Open a terminal window

b. At the prompt type: `leafpad.profile` (opens a notepad editor)

c. At the bottom of the file, add the line `GPIOZERO_PIN_FACTORY=pigpio`

d. Save the file and exit

3. Reboot the Pi



j. SenseHAT:

i. Weird/dim/blank LED array

1. Check the headers are connected properly.

a. Are all the pins covered?

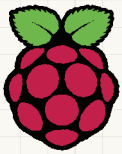
b. Is the board on the right way around — ribbon fold toward user, with Sense HAT on top?

3. Running headless

a. Code is not running at boot:

i. Have you specified the full path to the file in cron? (e.g. `/home/pi/filetorun.py` — the slash before home at the beginning is critical!)

ii. Is the code just running and then exiting instead of waiting for input? Don't forget that Mu and IDLE keep the Python shell active, whereas when you run Python from the command line, the interpreter will execute each line of the file in sequence and then terminate. To keep a program that sets threaded callbacks running, you need to add `import signal` at the start and `signal.pause()` as the final line.



Raspberry Pi