Challenge 2 – Team 16

Daniël de Goederen
Eindhoven Technical University
1847775
d.x.d.goederen@student.tue.nl

Ami Gjoca
Eindhoven Technical University
2214326
a.gjoca@student.tue.nl

Frankie Whitehead
Eindhoven Technical University
2426323
f.a.whitehead@student.tue.nl

Katarzyna Mendel
Eindhoven Technical University
2185946
k.w.mendel@student.tue.nl

Introduction

The last quartile we have been working together on making a connected escape room in Yellowstone theme. In this report we will cover the connection of modules and its role in the story, show DIY guides for building our Challenge 1 modules and finally reflect on our experiences.

Use of Modules

For our escape room, all modules are connected to OOCSI and activate at the pace of the user. After each module is complete, a signal is broadcast to OOCSI which in turn unlocks and activates the next module. Throughout the experience, participants receive narrative guidance and task hints delivered directly to their phones via OOCSI-connected messaging, keeping them informed without breaking immersion.

The escape room begins in a dark setting simulating the night sky of Yellowstone National Park. Users must first light the campfire residing in the middle of the room to make light. This works through a reed switch within the fire that is mounted to contact magnets attached to the missing firewood. Once correctly placed, the circuit closes lighting up the LEDs and activating the speakers which emulate fire. This then sends a signal to OOCSI that the fire is lit allowing for the Colorado Cam to be activated for the next challenge. As a narrative development, the participant responsible for lighting the fire has burned their hands and is therefore unable to use them for the remainder of the experience.

In the flickering glow, participants see the pamphlets and finally make out the message printed out across them: A warning that the Yellowstone volcano is predicted to erupt within 24 hours. They now have to find a green bag amongst a pile of different ones and get their car keys to be able to go to the phone booth and contact emergency services. Participants scan the color of the bag using the Colorado Cam on their phones which is connected to OOCSI. When the camera detects the correct color, a signal is send to OOCSI in the form of color_scanned: "true". This triggers an audio message informing users that a bear has consumed the car keys, directing them to Module 3.

Figure 1. Bear for module 4

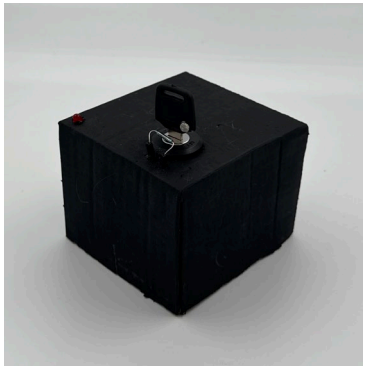


Figure 2. Key insert box for module 4

Figure 3. Steering wheel for module 5

In the midst of escaping, you need to kill the bear that ate your keys. Participants receive a prompt on their phones instructing them to kill the bear. This signal from OOCSI sequentially unlocks the next HTML screen. The bear has five lives, and participants must perform stabbing motions using the phones' accelerometer to access the next challenge. Proximity feedback is provided through the bear's growling sounds, with a final roar confirming its defeat. Upon completion, the result is published to OOCSI, bear_killed : ''true" which activates narrative prompt and the next module. Participants are then led to a plush bear on the ground (see Figure 1) which has a stomach cavity housing items symbolising the diet of a bear (canned fish, nuts, organs). They now have to scramble around the stomach to find the car key.

After the key is found, a message on the phone guides them to the Jeep display which contains a key ignition. Intuitively users insert and twist the key to initiate the driving sequence (see Figure 2). As the ignition switch itself was faulty, it works through magnets on the key which engage a reed switch only when the key is turned. Once triggered, the reed switch closes the circuit and publishes the data reed_active: "true" to OOCSI. This signal sequentially unlocks the driving game.

Upon receiving the signal from previous module, a message informs the need to put their phone into the cavity in the jeep wheel (see Figure 3). The driving game then starts on the laptop as a Processing game. Steering input is captured through the phones' built in accelerometer, with real time movement data transmitted to the game via the OOCSI server, translating physical wheel rotation into in-game directional control. The participant has three lives and needs to dodge animals. Failure to complete the course results in an unsuccessful escape attempt.

Once users successfully pass the animals in the driving game, they need to retrieve the phone from the wheel. A message is displayed "Finally you arrived to the phone booth. Put on the right frequency to call for help". Participants then adjust two potentiometers until the correct frequency is reached (see Figure 7). The potentiometer values are read continuously and transmitted to OOCSI; when the correct combination is detected, the signal freqCorrect: "true" is sent, simultaneously activating a green LED as visual confirmation. The final twist is then revealed*: ''E*mergency services are currently unavailable; the only way to escape is jumping on a raft from a nearby waterfall'' .

The final module is activated only after the right frequency is found. The balancing boat game which is displayed on the laptop as an HTML file and played with the left and right arrow keys. Users have three lives and need to dodge obstacles in the water stream to finally escape the danger. Successfully navigating the course to its conclusion signals the completion of the escape room experience. Unfortunately we didn't manage to make a physical balancing board in time (see Figure 8).

DIY Guides

See pictures on the next pages for stepwise instruction for the creation of modules 1 and 2.

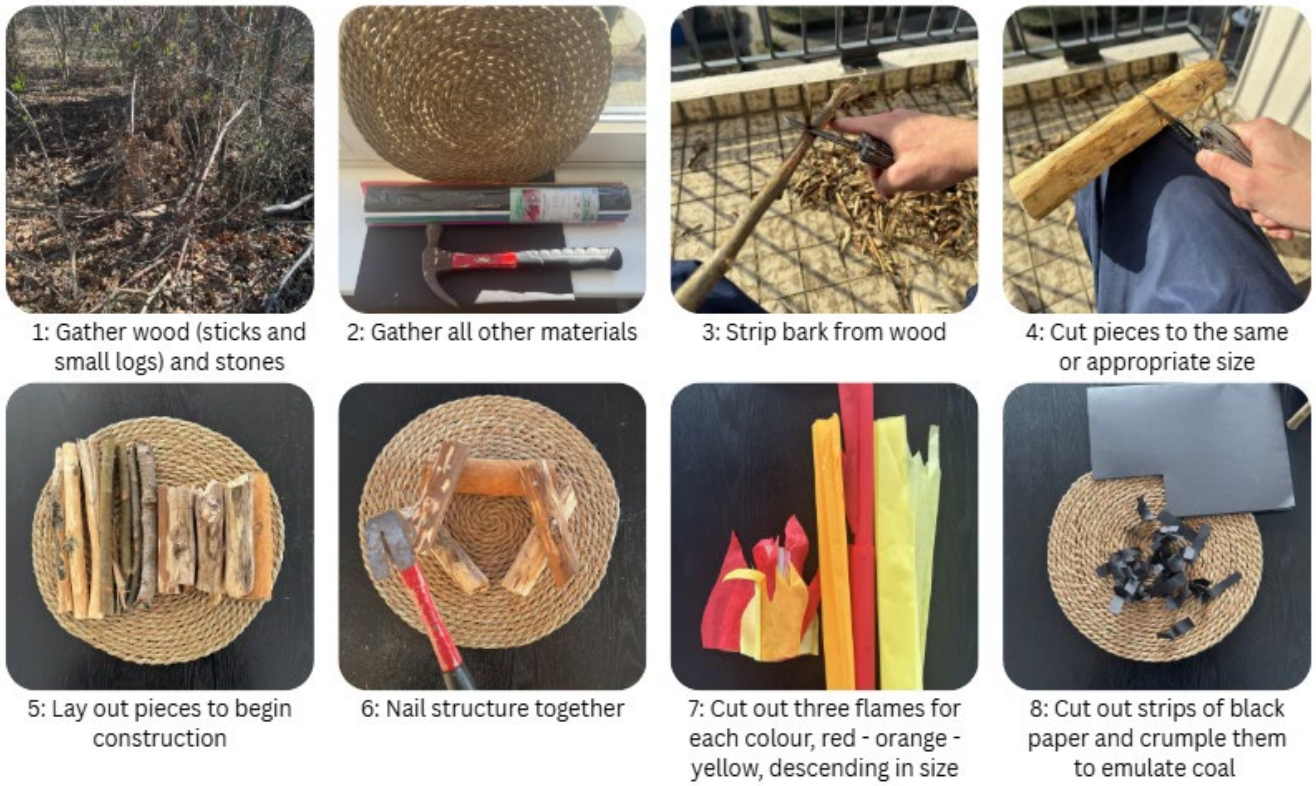


Figure 4. First 8 steps of DIY module 2

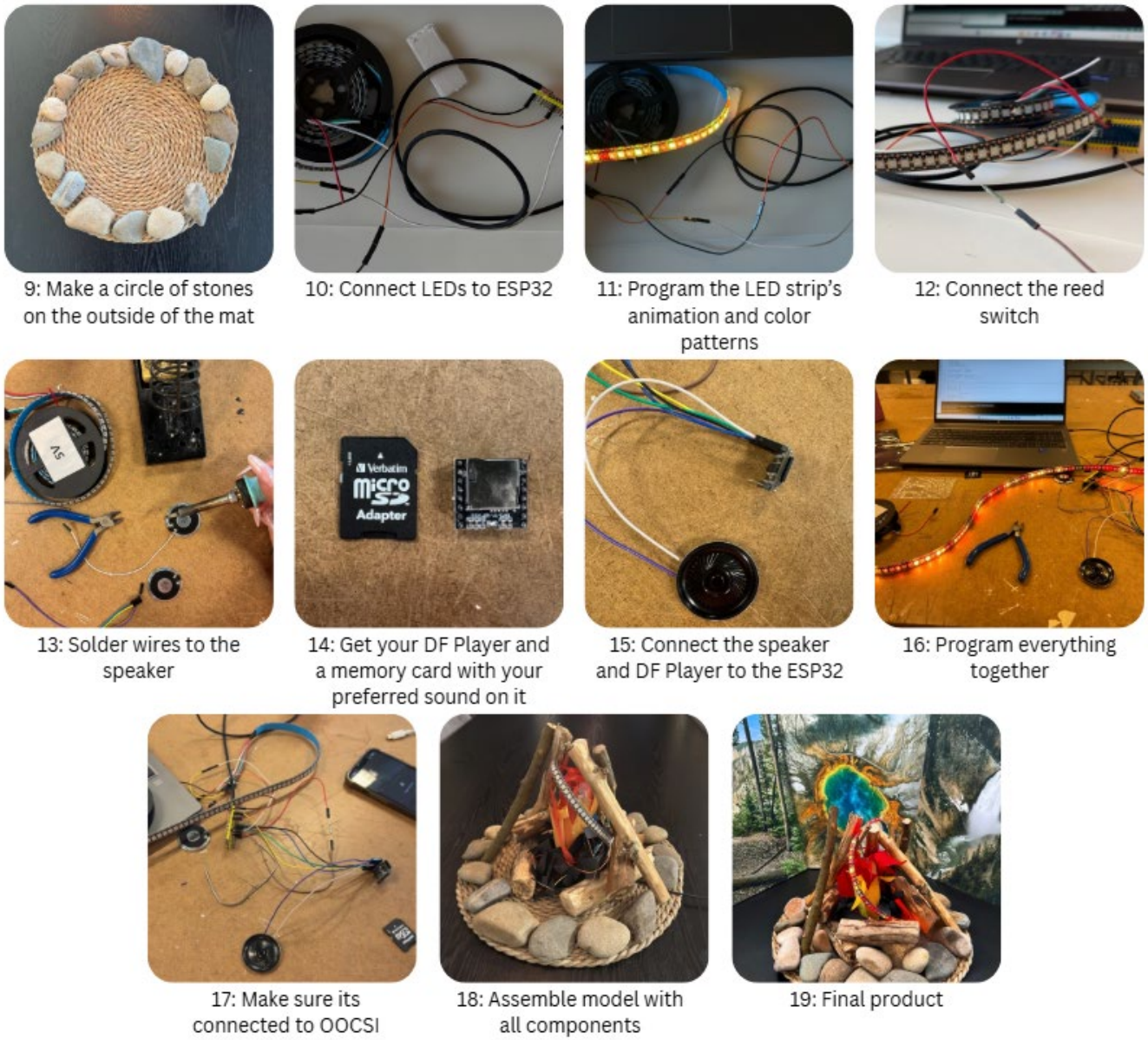


Figure 5. Last 11 steps of DIY module 2

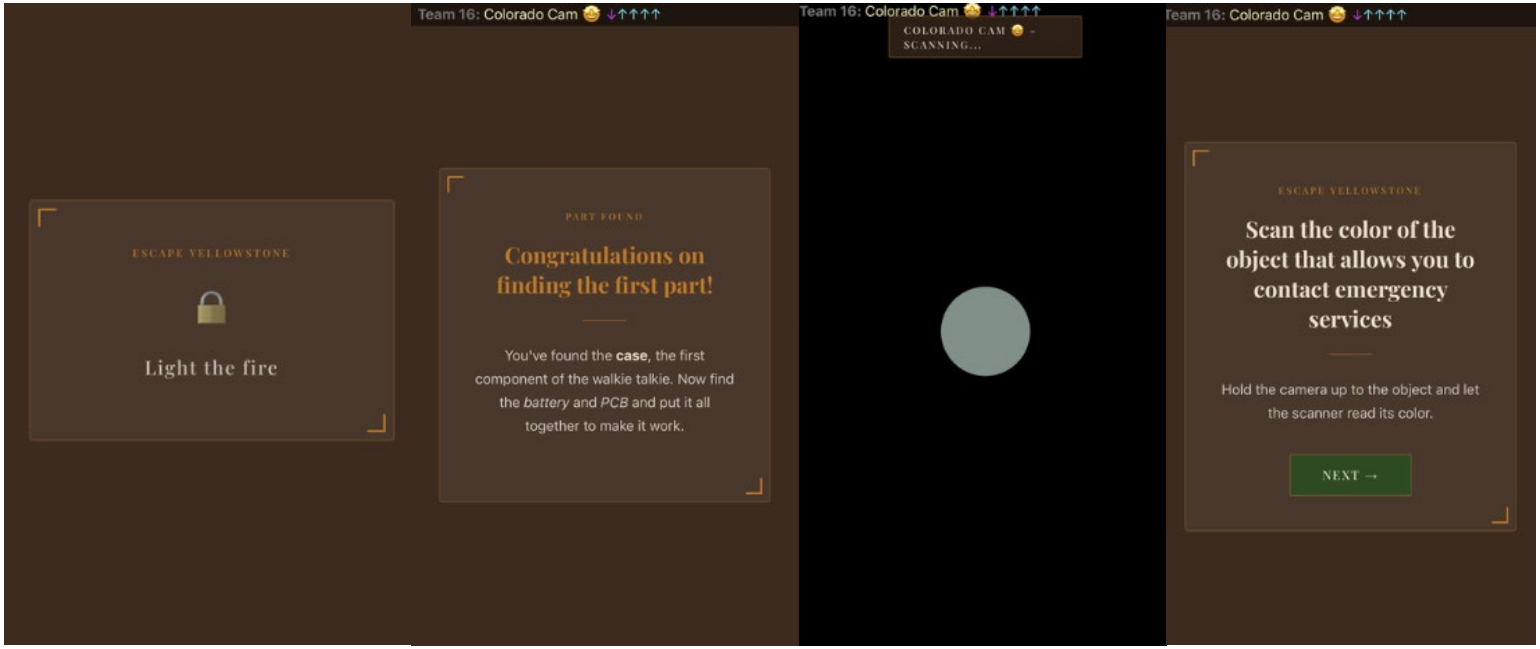


Figure 6. Screens for DIY guide module 1

DIY Guide module 1

Step 1: Host website on Data Foundry

Step 2: Download our modified Colorado Cam file

Step 3: Edit the code to your liking

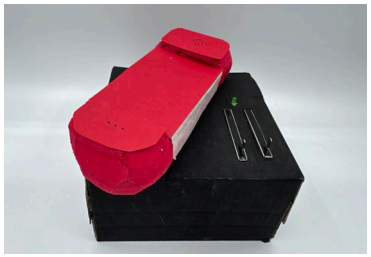


Figure 7. Emergency Services dial for module 6

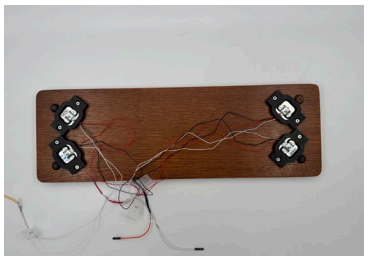


Figure 8. Discarded balancing board for module 7

Data Documentation

For the system diagram and static logical data canvas, see the appendix A. The images were too big to put in between text or in the side bar.

Group Reflection

Overall, we consider Challenge 2 a partial success. While we achieved most of our individual learning goals, the fact that the system did not function properly during the demo highlights a shortcoming in our execution. Although we developed valuable skills, the final result did not fully meet the of an entirely working, integrated system. Therefore, we would describe the project as successful in terms of learning, but a bit less in terms of delivery.

The demo day did not go as planned. Two modules failed due to power banks that were not turned on, and we forgot to place the wooden stick in the room, making part of the interaction impossible. These were avoidable mistakes and indicate a lack of preparation and structure. Although we felt caught off-guard, this was ultimately due to our own insufficient readiness. We only arrived one hour in advance, which proved inadequate for setting up and testing a complex, interconnected system. However, this issue goes beyond simply arriving earlier. The absence of a structured pre-demo checklist and clearly assigned responsibilities meant that crucial tasks, such as powering devices, were overlooked. Implementing such procedures would likely have prevented these errors.

In addition, the limited preparation time on the demo day was a symptom of a broader planning issue. Many modules were only finalized during the final days (Friday through Sunday), leaving little to no time for proper integration and testing. This suggests that we underestimated the complexity of combining individual components into a cohesive system. The problem was not only a lack of time, but also a lack of intermediate milestones focused on integration. A clearer planning strategy, made immediately after Challenge 1, should have included deadlines, dedicated integration moments, and buffer time to accommodate unexpected challenges. This would have reduced time pressure and improved the overall reliability of the system.

From a team perspective, these issues also point to shortcomings in collaboration and coordination. Responsibilities for testing and final setup were not clearly distributed, which contributed to oversights during the demo. In future projects, defining roles more explicitly and ensuring shared accountability for system readiness would improve both efficiency and outcome.

Regarding the personal goals we set in the beginning of the course we are satisfied with our learnings and how they contribute to us becoming better designers.

Daniël for example set the goals of learning how to write basic HTML/CSS, understand OOCSI and how to connect physical and digital prototypes, and how to use Data Foundry for the hosting of websites. He achieved these goals by taking in responsibility for the creation of the HTML application and two Processing games, for which he need to understand both HTML, OOCSI and Data Foundry. In addition, he also wanted to learn how to visualize data streams, which he realized by making the System Diagram.

Furthermore, Katarzyna achieved her goal of strengthening programming skills and learning how to use OOCSI by taking on responsibilities for the creation of a Processing game and connect it accordingly. She also improved her ability to translate creative ideas into working prototypes and became more proficient in applying affordances [1] in designs by for example making the steering wheel for module 5.

Similarly, Frankie worked towards his learning goals by actively engaging with the technical and collaborative aspects of the project. He strengthened his teamwork

skills by closely collaborating with group members, contributing to shared decisions and supporting the integration of different project elements. Frankie also focused on working with electronics, gaining hands-on experience with components such as ESP32 microcontrollers and load cells, which helped him understand how individual parts function within a larger system. This knowledge supported his goal of creating a connected system, where physical and digital elements communicate effectively.

Similarly, Ami worked towards her learning goals by gaining hands-on experience with ESP32 and Arduino, improving her ability to install libraries, configure boards, and connect devices to Wi-Fi and OOCSI. Through this, she developed stronger troubleshooting skills and a deeper understanding of how systems function and how data flows between connected modules. She also gained practical hardware experience, including soldering, and learned the importance of early component testing. Additionally, Ami improved her ability to work under pressure, adapt to challenges, and find alternative solutions when needed.

References

1. S. A. G. Wensveen, J. P. Djajadiningrat, and C. J. Overbeeke. 2004. Interaction frogger. Proceedings of the 5th conference on Designing interactive systems: processes, practices, methods, and techniques: 177–184.

Statement of AI

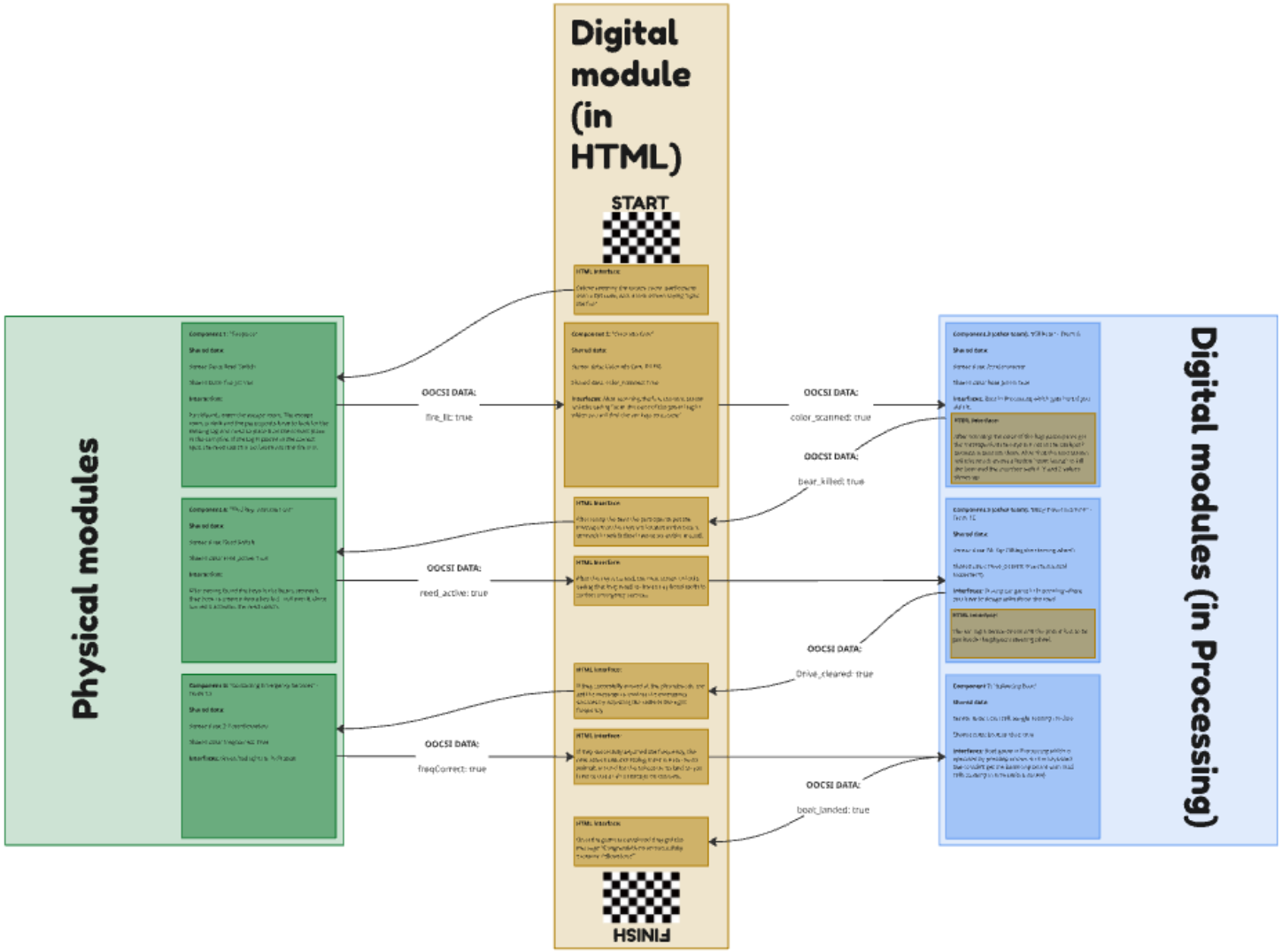
We used AI in the debugging and step wise instruction for the creation of code for the HTML, Processing and Arduino IDE code.

Appendix A

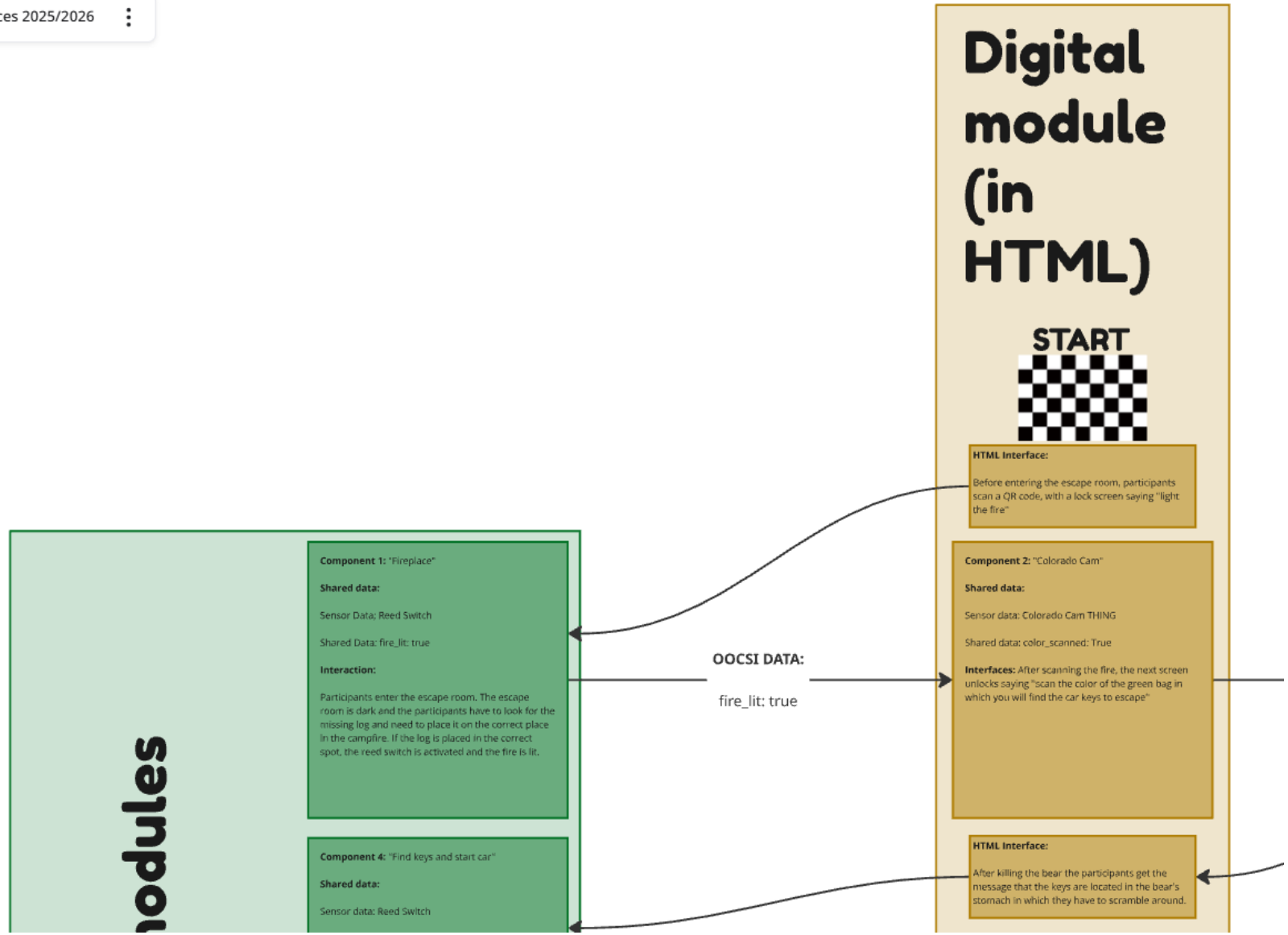
What	Daniël	Katarzyna	Frankie	Ami
HTML application all modules	X			
Physcical campfire module 1			X	X
Processing module 3	X			
Processing module 5	X	X		
Processing module 7	X			
Physical frequency slider module 6				X
Physical steering wheel module 5		X		
Physical bear module 4			X	
Electronics module 4		X		X
Electronics balancing board module 7			X	
Video				X
Flyer	X			
Posters		X		
Report	X	X	X	

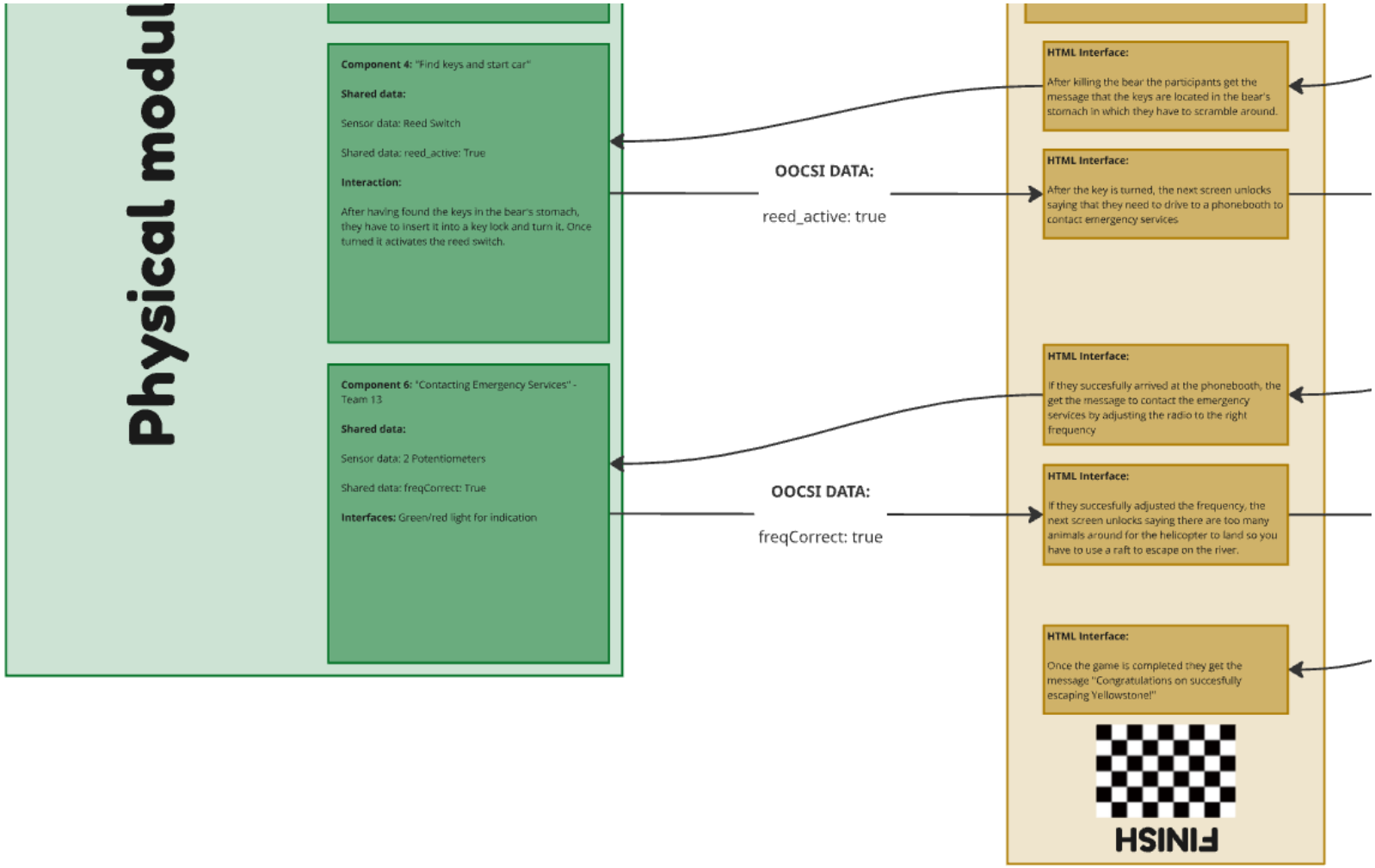
Figure 9. Group member task division

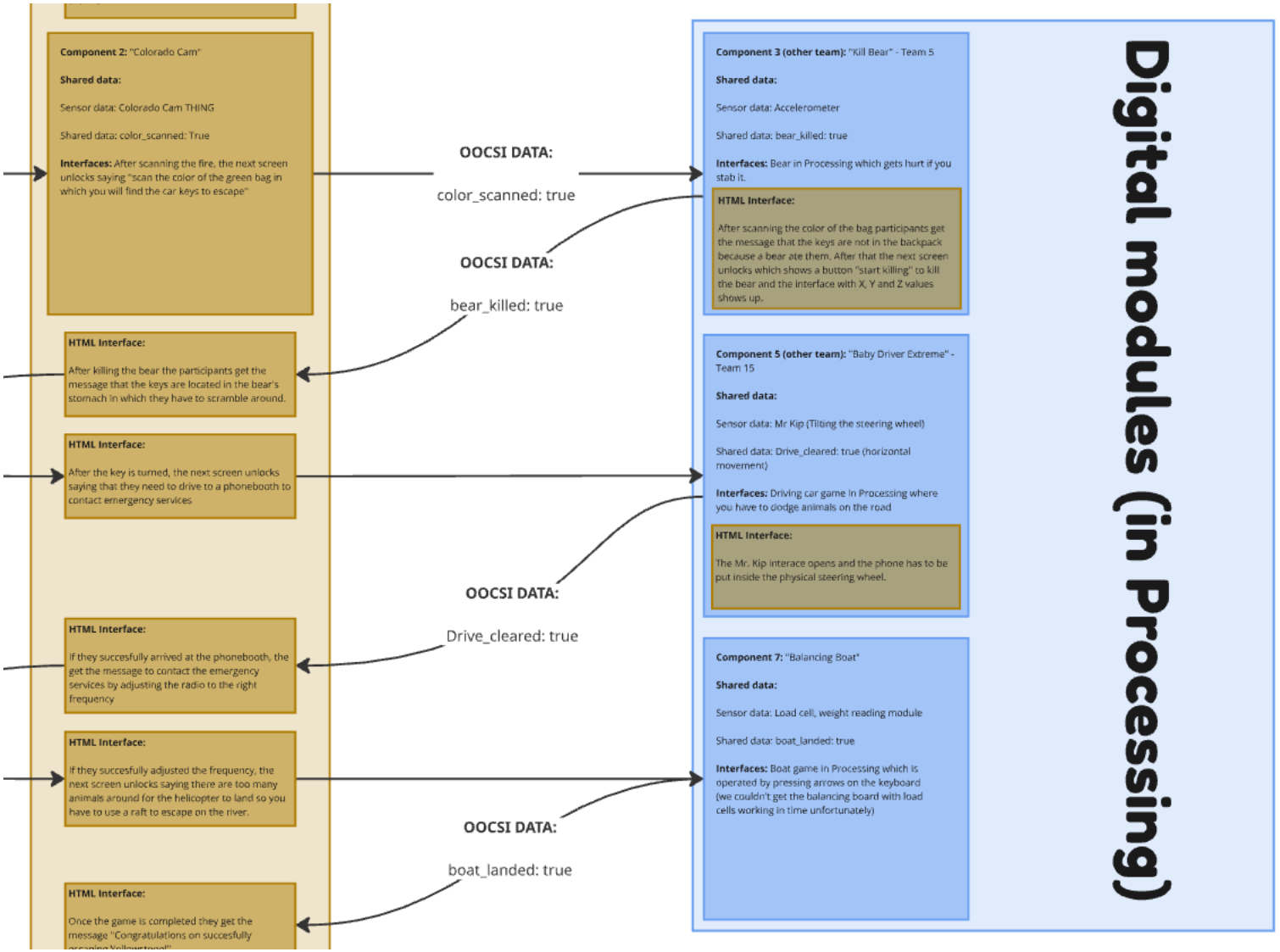
System Diagram displayed in Physical and digital modules



eriences 2025/2026




Physical modul
Component 4: "Find keys and start car"
Shared data:
Sensor data: Reed Switch
Shared data: reed_active: True
Interaction:
After having found the keys in the bear's stomach, they have to insert it into a key lock and turn it. Once turned it activates the reed switch.
Component 6: "Contacting Emergency Services" - Team 13
Shared data:
Sensor data: 2 Potentiometers
Shared data: freqCorrect: True
Interfaces: Green/red light for indication
OOCSI DATA:
reed_active: true
OOCSI DATA:
freqCorrect: true
HTML Interface:
After killing the bear the participants get the message that the keys are located in the bear's stomach in which they have to scramble around.
HTML Interface:
After the key is turned, the next screen unlocks saying that they need to drive to a phonebooth to contact emergency services
HTML Interface:
If they succesfully arrived at the phonebooth, the get the message to contact the emergency services by adjusting the radio to the right frequency
HTML Interface:
If they succesfully adjusted the frequency, the next screen unlocks saying there are too many animals around for the helicopter to land so you have to use a raft to escape on the river.
HTML Interface:
Once the game is completed they get the message "Congratulations on succesfully escaping Yellowstone!"
FINISH


Component 2: "Colorado Cam"
Shared data:
Sensor data: Colorado Cam THING
Shared data: color_scanned: True
Interfaces: After scanning the fire, the next screen unlocks saying "scan the color of the green bag in which you will find the car keys to escape"
OOCSI DATA:
color_scanned: true
Component 3 (other team): "Kill Bear" - Team 5
Shared data:
Sensor data: Accelerometer
Shared data: bear_killed: true
Interfaces: Bear in Processing which gets hurt if you stab it.
HTML Interface:
After scanning the color of the bag participants get the message that the keys are not in the backpack because a bear ate them. After that the next screen unlocks which shows a button "start killing" to kill the bear and the interface with X, Y and Z values shows up.
OOCSI DATA:
bear_killed: true
HTML Interface:
After killing the bear the participants get the message that the keys are located in the bear's stomach in which they have to scramble around.
HTML Interface:
After the key is turned, the next screen unlocks saying that they need to drive to a phonebooth to contact emergency services
Component 5 (other team): "Baby Driver Extreme" - Team 15
Shared data:
Sensor data: Mr Kip (Tilting the steering wheel)
Shared data: Drive_cleared: true (horizontal movement)
Interfaces: Driving car game in Processing where you have to dodge animals on the road
HTML Interface:
The Mr. Kip interace opens and the phone has to be put inside the physical steering wheel.
OOCSI DATA:
Drive_cleared: true
HTML Interface:
If they succesfully arrived at the phonebooth, the get the message to contact the emergency services by adjusting the radio to the right frequency
HTML Interface:
If they succesfully adjusted the frequency, the next screen unlocks saying there are too many animals around for the helicopter to land so you have to use a raft to escape on the river.
Component 7: "Balancing Boat"
Shared data:
Sensor data: Load cell, weight reading module
Shared data: boat_landed: true
Interfaces: Boat game in Processing which is operated by pressing arrows on the keyboard (we couldn't get the balancing board with load cells working in time unfortunately)
OOCSI DATA:
boat_landed: true
HTML Interface:
Once the game is completed they get the message "Congratulations on succesfully
Digital modules (in Processing)

Communication

Entry

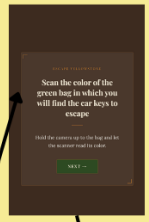
scan the qr code

Component 1 (physical): Campfire

Sensor: Reed switch
Actuator: LEDs

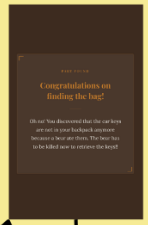
fire_lit: true

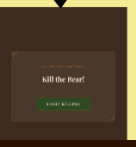
Component 2 (digital): Colorado cam



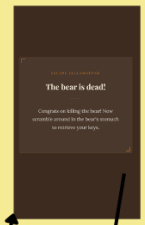
2. button 'next' pressed

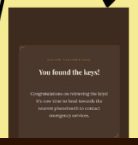
Component 3: (team 5) Kill bear



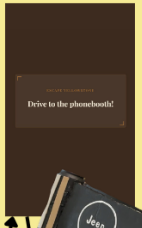


Component 4 (physical): Find keys and start car

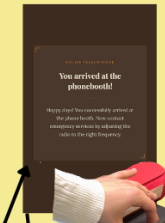




Component 5 (team 15): Baby driver extreme



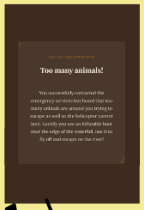
Component 6 (team 16): Contact emergency services



Sensors : 2
Potentiometers
Actuator: LED

freqCorrect: true

Component 7 (digital): Balancing Boat



boat_landed: true

DIY Digital Component

How to build the digital component DIY:

1. Host website on Data Foundry
2. Download our modified Colorado Cam code
2. Edit the code to your liking