Raspberry PI Project – Callum Holman-Hathaway, Joshua Kear, Liam Hateley

Table of Contents

Plan

We are going to make a 2D flappy bird redesigned game! This game is going to be made to make people move around while playing an exciting game that is popularly known. The game will include pressure sensors so that the bird can be moved and that it is easily accessible for everyone to play.

Roles & Responsibilities of each team member

Callum – Overall Team Manager, Designer and Coding

Josh – Coding

Liam – Coding

Concept

The general concept of the game is that we would have a start menu that leads to the game, where the player can then control a pixelated bird character, using a pressure sensor to help control the movement of the said character. The main goal of the game is to have the player navigate the bird through green pipes without hitting them, as if the player hits one of these pipes it will cause them to lose within the game. Whenever the player hits the pressure sensor it would allow them to instantly go up, but if no pressure is put upon the sensor, then gravity will take over causing the bird to fall until it hits the bottom of the map, which will also game over the player. This is meant to be a fun and challenging game with all age groups in mind, meaning that it can be fun for all, while also testing how good their reflexes are! Each pipe the player passes through will earn them a point, that then is tallied up and displayed at the end of the game on the game scoreboard.

Gameplay

When you launch the game up, it prompts you to press any key to start. Once you've pressed a key, the game will start, and you are able to play the game. Basically, you have to keep pressing the space bar, so the bird jumps (as to fly) and you have to fly through in between the green pipes. Once you have flown through them, you will receive a point, and it'll be the same as the game progresses. It generates them endlessly, so the game will never end unless you mess up.

Why are we making the game?

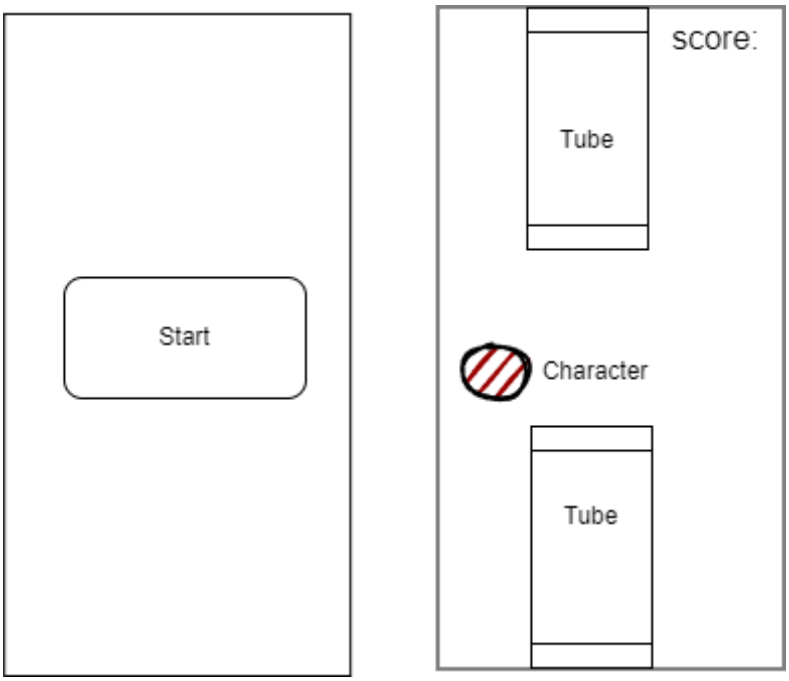
We are making the game because we want to showcase our skills at Cheltenham Science Festival since we have adequate skills in programming and game design. Cheltenham Science Festival is one of the popular events to go to for everything to do with Technology and General Sciences. The goal of our project is to show case a game which relates to the Subject of Internet of Things (IoT).

Our project will showcase not only our technical abilities but also our understanding of the Internet of Things (IoT) and its applications in a cyber-awareness context. This aligns with the festival's goal of inspiring people to explore science and technology, making our game both educational and engaging for players aged 8 to 18.

Additionally, creating this game allows us to contribute meaningfully to the local emphasis on cybersecurity, a field of increasing importance given recent cyber-attacks in the region. By developing a functional prototype, we aim to demonstrate how IoT can be used innovatively and effectively to promote awareness and inspire future solutions in technology.

Finally, competing against other teams and collaborating with North Green Security provides us with a platform to refine our project management skills while delivering a tangible, impactful result.

Basic Design of the game

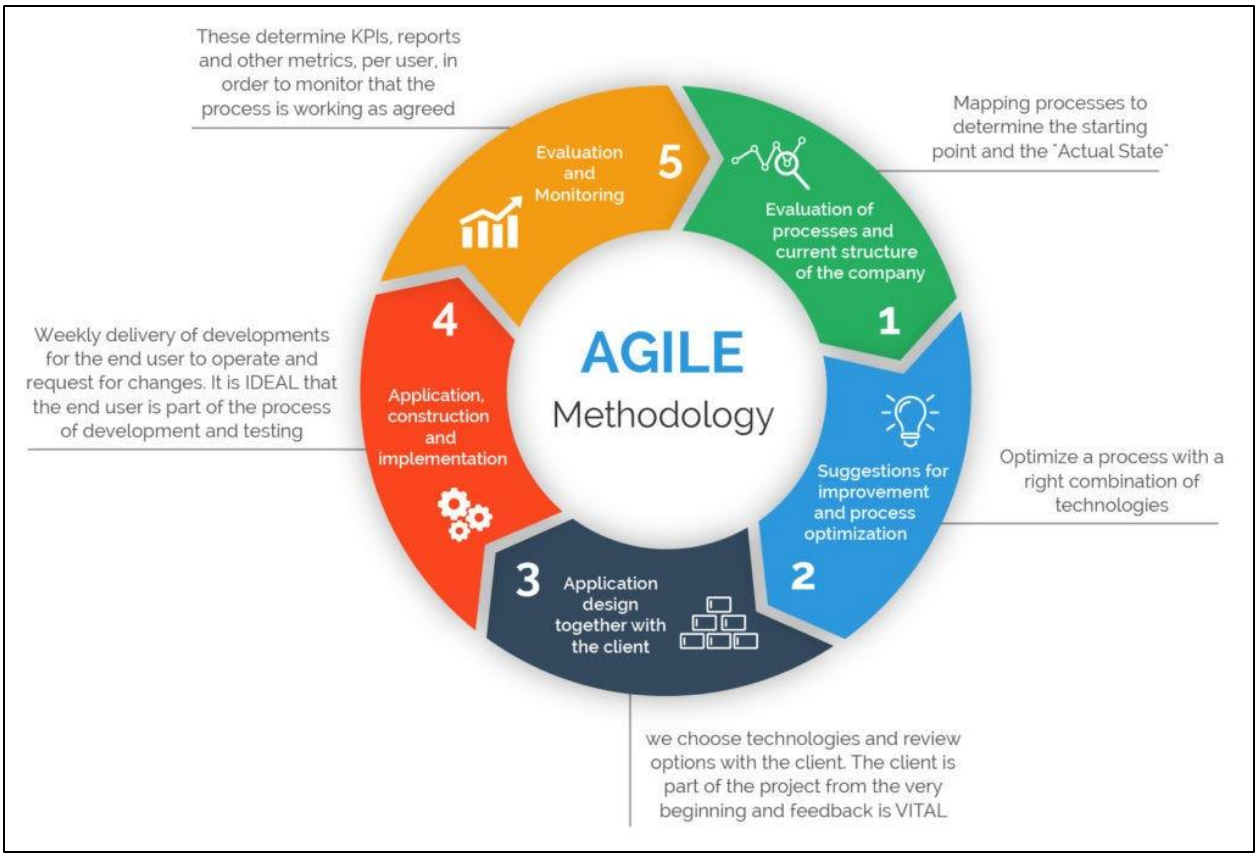


- This is the basic wireframe design for the game
- We will utilise different sounds for the game, so it doesn't infringe copyright of the original game.

https://github.com/LeonMarqs/Flappy-bird-python

Methodology Used within our project

We are using the Agile methodology; we will be including incremental upgrades to the game and having collaborative sessions with developers and the client. This is a common feature of Agile to share feedback with others to ensure that every requirement is met by us so that it meets the needs of the client.



Why are we using the agile methodology?

The reasoning behind our choice for the utilisation of the agile methodology within our project:

- **Adaptive and flexible** - Due to its adaptive and flexible nature it allows us to respond quickly to changing requirements within our project as well as improvements we make based off customer feedback, which helps us produce that high quality final product that we are aiming for.
- **Customer satisfaction/target audience satisfaction** - This key agile strength grants us the ability to create a product that better meets the customers/target audience needs and wants, achieved through the methodology's iterative development, continuous feedback abilities and frequent delivery of value.
- **Continuous improvement** - This agile process allows us too constantly refine and perfect our project and its service, through the methodology's iterative design and feedback loops, that help us to improve the overall quality, efficiency, and customer satisfaction of our final product.

Break the project down based on the methodology

So, we will be working on the game, which is flappy bird, but we've got to plan it so based on the documentation there'll be a flappy bird game.

Timeline

Work on documentation, make game, publish game and put it on Cheltenham science festival.

User Requirements

The user requirements for our project will consist of:

- One of the most important user requirements our project considers is the fact that the project works as intended, as well as on top of that it works alongside the IOT (Internet of Things) device we will be utilising for the project.
- An additional user requirement that is significant to the project is that the project not only works as intended but also is built to run quickly and smoothly for its end users, helping us to create a robust and refined application for our target audience.
- Another user requirement of our project is that the project will be designed with the target audience in mind, meaning their wants and needs will be considered, and the project will be built around these principles.

Target audience

The target audience of our project will range anything between the ages 8 – 18, however the game can still be played by any age group who wish to challenge themselves and play it, nonetheless the project will accommodate to the target audience we have in mind, meaning the other age groups many not like the games as much as the target audience will as their own needs and wants won't have been met. The reasoning for a target audience decision was that the younger generations (ages 8 to 18) will be at the Cheltenham science festival, where we are presenting our game/project at, so by creating the game to accommodate to their requirements it allows us to build a game that is considered both engaging and fun for them, as well as a bit of challenge that they can test their reaction time with.

Why are you doing it this way

We have created the plan this way since it aligns with our project timeline and IoT roll-out strategy, helping us in creating a well-designed and optimized workflow. The plan was designed on the basis of stakeholders' requirements, resource availability, risk analysis, and industry best practice. By executing these requirements, we can effectively manage our time, costs, and quality, with the possibility of being integrated alongside existing infrastructure and scaled up for future business growth.

It also facilitates teamwork, which prevents the scope creep possibility, and decreases the possibility of unexpected issues. As for the IoT, it provides effective and secure data communication among the devices, reliability of the system, and network performance. While also providing improved monitoring, maintenance, and debugging, long-term sustainability, and system security.

Lastly, this strategy facilitates effective project delivery with the efficiency, security, and agility of IT project management and the adoption of IoT.

Risk Assessment

We are assessing our game based on the risks that are likely to occur and how'd we mitigate them to ensure that everything works on the day of the Cheltenham science festival, where we will be showcasing our game/project.

Hazard	Risk that could occur	Likelihood	Severity	Mitigation Measures
Electrical Hazard	Risk of electric shock from exposed wires	Low	High	Ensure all wiring is insulated, use low-voltage components, and secure connections. Also to ensure all cables are PAT Tested.
Overheating	Components overheating and causing burns	Low	Medium	Use heat sinks, proper ventilation, and avoid overloading circuits.
Trip Hazards	Loose wires causing trips and falls leading to accidents such as broken leg	Medium	High	Clear all pathways, secure cables and ensure they are covered with cable covers.

Component failure	Sensors could potentially fail mid use	Medium	Medium	Regular testing before the event, have backup components available.
Software bugs	Game freezing or malfunctioning	Medium	Low	Implement thorough testing into the games and know reset commands in case of a required reset.
Cybersecurity Risks	Unauthorized access to IoT devices	Low	High	Use secure connections, encrypt data, and disable unnecessary network access.

Test Plan

Testing Methodology

1. **Unit Testing** - Individual components such as the pressure sensor, movement mechanics, and scoring system will be tested separately.
2. **Integration Testing** - The interaction between the pressure sensor, game mechanics, and display will be tested for smooth functionality.
3. **User Acceptance Testing (UAT)** - Testing with users in the target age range (8-18) to ensure ease of play and engagement.
4. **Performance Testing** - Ensuring the game runs smoothly on the Raspberry Pi without latency.
5. **Security Testing** - Checking for vulnerabilities that may arise with IoT integration.
6. **Bug Tracking & Fixing** - Documenting issues and iterating improvements based on feedback.

Feedback from others

To do – SEND GITHUB LINK to peers for review.

Feedback from Logan McAvoy -

Advantages

- Good gameplay
- Tracks score and time

Disadvantages (all visual)

- Low framerate
- Strong screen tearing (Vsync fix)
- Gaps in terrain
- No animation when hitting something (DEAD)
- Score is delayed (but still tracking)

Progress Tracking Logs

What have we done today? 21/10/24

- Josh – I have set up a GitHub repository to allow users to view our code and also made the whole game using a referenced source.
- Callum – I have helped Josh set up GitHub repository and set up documentation and expanding on it.
- Liam – I have researched different sensors so that we will find one that is suitable for our project.

04/11/24

Liam & Callum – Added to documentation and wireframes for the game

Josh – Made improvements to the code and fixed bugs

06/01/2025

Callum – Off for sickness

Josh & Liam - Imported main game code into Raspberry PI's

13/01/2025

Exam week for IT Service Delivery.

20/01/2025

Exam prep for Cybersecurity unit – skipped Client Code and Server code until next week.

27/01/2025

Josh – Coding in the client server

Liam – Working on the server code

Callum – Working on the documentation

03/02/2025	Integrated pressure sensor with game controls; tested input accuracy and stability.
10/02/2025	Sprint Review: Conducted internal playtesting; identified lag and frame drop issues.
17/02/2025	Optimised game performance; added basic animations and smoother transitions.
24/02/2025	Mid-project demo for tutor; began implementing feedback (UI improvements, sound tweaks).
03/03/2025	Developed server-side scoreboard and real-time update features; tested networking code.
10/03/2025	Sprint Review: Improved server reliability and UI layout; fixed delayed scoring bug.
17/03/2025	Conducted User Acceptance Testing (UAT) with peers aged 8–18; collected feedback.
24/03/2025	Applied UAT feedback: enhanced UI colour scheme and added animated death sequence.
31/03/2025	Created promotional materials and posters for Cheltenham Science Festival.
07/04/2025	Easter Break – no development (paused).
14/04/2025	Finalised game features; conducted full integration test with Raspberry Pi hardware.
21/04/2025	Began final documentation writing and evidence collection (screenshots, logs, videos).
28/04/2025	Final internal QA test; verified stability, responsiveness, and sensor accuracy.
05/05/2025	Prepared project showcase: packaged game, tested setup procedure for event deployment.
12/05/2025	Peer testing session: invited classmates to play and report bugs or usability issues.
19/05/2025	Applied last-minute polish: improved instructions screen, updated scoreboard visuals.

26/05/2025	Created final presentation slides and backup materials for event.
02/06/2025	**Cheltenham Science Festival presentation** – live demonstration of final game product.

Sensor Research

Different types of sensors that we considered using for our project:

"**Motion sensors**" could be used to detect the movement of the players or an object held by the player, perhaps allowing them to jump within the game.

A "**capacitive touch sensor**" could also be used as this sensor detects the presence of things, such as a finger, which could then be utilised to allow the player to jump within the game.

Another sensor that could work is a "**proximity sensor**", as this device is capable of detecting objects without contact and when it detects something it will generate a signal, which could be then used to allow the player to jump within the game.

A "**pressure sensor**" could be used to control the height of the player's jump (the lower the pressure the lower the sprite will go, and the higher the pressure the higher the sprite will go).

Research conclusion:

In conclusion, we will be using a pressure sensor for our project, as it best fits the game we have in mind. This will allow the user to move the sprite based on how much pressure they put on the sensor, such as if the user applies more pressure to the sensor, the higher up the sprite will go in game, and the less pressure the user applies the lower down the sprite will go.

Preparing the PI & Other modules

We will be using a Raspberry PI to host our game on along with a Client and a Server.

Code – Main Game file [flappy.py]

import pygame, random, time from pygame.locals import * import sys import os import logging from datetime import datetime

Create "logs" folder if it doesn't exist

if not os.path.exists('logs'): os.makedirs('logs') if not os.path.exists('logs/game'): os.makedirs('logs/game')

Create a log file with timestamp

timestamp = datetime.now().strftime('%Y%m%d-%H%M%S') log_filename = f"logs/game/{timestamp}.txt"

Set up logging to file

logging.basicConfig(filename=log_filename, level=logging.DEBUG, format='%(asctime)s - %(levelname)s - %(message)s',)

Redirect stdout and stderr to log file

sys.stdout = open(log_filename, 'a') sys.stderr = open(log_filename, 'a')

Log the initial startup message

logging.info("Game started")

Initialize pygame

pygame.init()

Get the screen resolution

screen_info = pygame.display.Info() SCREEN_WIDTH = screen_info.current_w SCREEN_HEIGHT = screen_info.current_h

VARIABLES

SPEED = 10 GRAVITY = 0.4 GAME_SPEED = 10 FRAME_RATE = 60

GROUND_WIDTH = 2 * SCREEN_WIDTH GROUND_HEIGHT = 100

PIPE_WIDTH = 150 PIPE_HEIGHT = SCREEN_HEIGHT

BIRD_WIDTH = 34 BIRD_HEIGHT = 24 BIRD_SCALE = 2

Load sounds

wing = 'assets/audio/wing.wav' hit = 'assets/audio/hit.wav' point = 'assets/audio/point.wav'

pygame.mixer.init() wing_sound = pygame.mixer.Sound(wing) hit_sound = pygame.mixer.Sound(hit) point_sound = pygame.mixer.Sound(point)

Set display mode (start in windowed mode)

screen = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT), pygame.RESIZABLE) pygame.display.set_caption('Flappy Bird')

Load images (scaled once)

BACKGROUND = pygame.image.load('assets/sprites/background-day.png').convert() BACKGROUND = pygame.transform.scale(BACKGROUND, (SCREEN_WIDTH, SCREEN_HEIGHT)) GROUND_IMAGE = pygame.image.load('assets/sprites/base.png').convert() GROUND_IMAGE = pygame.transform.scale(GROUND_IMAGE, (GROUND_WIDTH, GROUND_HEIGHT))

is_fullscreen = False

def toggle_fullscreen(): global is_fullscreen, screen if is_fullscreen: screen = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT), pygame.RESIZABLE) else: screen = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT), pygame.FULLSCREEN) is_fullscreen = not is_fullscreen

class Bird(pygame.sprite.Sprite): def **init**(self): pygame.sprite.Sprite.**init**(self) self.images = [pygame.transform.scale(pygame.image.load(f'assets/sprites/bluebird-{flap}.png').convert_alpha(), (int(BIRD_WIDTH * BIRD_SCALE), int(BIRD_HEIGHT * BIRD_SCALE))) for flap in ['upflap', 'midflap', 'downflap']] self.speed = 0 self.current_image = 0 self.image = self.images[self.current_image] self.mask = pygame.mask.from_surface(self.image) self.rect = self.image.get_rect() self.rect[0] = SCREEN_WIDTH / 6 self.rect[1] = SCREEN_HEIGHT / 2

```
def update(self):
    self.current_image = (self.current_image + 1) % 3
    self.image = self.images[self.current_image]
```

```
    self.speed += GRAVITY
    self.rect[1] += self.speed

def bump(self):
    self.speed = -SPEED
```

class Pipe(pygame.sprite.Sprite): def **init**(self, inverted, xpos, ysize): pygame.sprite.Sprite.**init**(self) self.image = pygame.image.load('assets/sprites/pipe-green.png').convert_alpha() self.image = pygame.transform.scale(self.image, (PIPE_WIDTH, PIPE_HEIGHT)) self.rect = self.image.get_rect() self.rect[0] = xpos

```
   if inverted:
        self.image = pygame.transform.flip(self.image, False, True)
        self.rect[1] = -(self.rect[3] - ysize)
    else:
        self.rect[1] = SCREEN_HEIGHT - ysize
    self.mask = pygame.mask.from_surface(self.image)

def update(self):
    self.rect[0] -= GAME_SPEED
```

class Ground(pygame.sprite.Sprite): def **init**(self, xpos): pygame.sprite.Sprite.**init**(self) self.image = GROUND_IMAGE self.rect = self.image.get_rect() self.rect[0] = xpos self.rect[1] = SCREEN_HEIGHT - GROUND_HEIGHT

```
def update(self):
    self.rect[0] -= GAME_SPEED
    if self.rect[0] <= -GROUND_WIDTH:
        self.rect[0] = SCREEN_WIDTH
```

def get_random_pipes(xpos): gap_size = random.randint(200, 400) # Randomize gap size size = random.randint(100, SCREEN_HEIGHT - gap_size - 100) # Randomize pipe height return Pipe(False, xpos, size), Pipe(True, xpos, SCREEN_HEIGHT - size - gap_size)

def display_score(score, screen, font): score_surface = font.render(f"Score: {score}", True, (255, 255, 255)) screen.blit(score_surface, (10, 10))

def show_start_screen(): screen.fill((0, 0, 0)) font = pygame.font.SysFont(None, 80) text = font.render("Press SPACE to Start", True, (255, 255, 255)) text_rect = text.get_rect(center=(SCREEN_WIDTH // 2, SCREEN_HEIGHT // 3)) screen.blit(text, text_rect) pygame.display.update() waiting = True while waiting: for event in pygame.event.get(): if event.type == KEYDOWN and event.key == K_SPACE: waiting = False

def game_over_screen(score): screen.fill((0, 0, 0)) font = pygame.font.SysFont(None, 80) text = font.render(f"Game Over! Score: {score}", True, (255, 255, 255)) text_rect = text.get_rect(center=(SCREEN_WIDTH // 2, SCREEN_HEIGHT // 3)) screen.blit(text, text_rect) pygame.display.update() time.sleep(2)

def game_loop(): show_start_screen() font = pygame.font.SysFont(None, 40) bird = Bird() bird_group = pygame.sprite.GroupSingle(bird) ground_group = pygame.sprite.Group(Ground(0), Ground(GROUND_WIDTH)) pipe_group = pygame.sprite.Group()

```
score = 0
clock = pygame.time.Clock()
passed_pipes = set()
pipe_spawn_delay = 3000
last_pipe_spawn_time = pygame.time.get_ticks()

while True:
    clock.tick(FRAME_RATE)
    current_time = pygame.time.get_ticks()

    if current_time - last_pipe_spawn_time > pipe_spawn_delay:
        pipe_group.add(*get_random_pipes(SCREEN_WIDTH))
        last_pipe_spawn_time = current_time

    for pipe in pipe_group:
        if pipe.rect.right < bird.rect.left and pipe not in
passed_pipes:
            passed_pipes.add(pipe)
            if len(passed_pipes) % 2 == 0:
                score += 1
                point_sound.play()

    for event in pygame.event.get():
```

```
        if event.type == QUIT:
            pygame.quit()
            sys.exit()
        if event.type == KEYDOWN and event.key == K_SPACE:
            bird.bump()
            wing_sound.play()

    screen.blit(BACKGROUND, (0, 0))
    pipe_group.update()
    pipe_group.draw(screen)
    ground_group.update()
    ground_group.draw(screen)
    bird_group.update()
    bird_group.draw(screen)
    display_score(score, screen, font)
    pygame.display.update()

    if pygame.sprite.spritecollideany(bird, pipe_group) or
pygame.sprite.spritecollideany(bird, ground_group):
        hit_sound.play()
        game_over_screen(score)
        return


game_loop()
```

Server Code – 2dflappyservergui.py

import socket import threading import tkinter as tk from tkinter import ttk

Server setup

HOST = '0.0.0.0' PORT = 12345

Store top 10 highest scores as (name, score) tuples

top_scores = []

Lock for thread-safe score updates

score_lock = threading.Lock()

def update_scores(name, score): with score_lock: top_scores.append((name, score)) top_scores.sort(key=lambda x: x[1], reverse=True) if len(top_scores) > 10: top_scores.pop() update_ui()

def update_ui(): for widget in score_frame.winfo_children(): widget.destroy()

heading = tk.Label(score_frame, text="SCORE BOARD", font=("Arial", 20, "bold")) heading.grid(row=0, column=0, columnspan=2, pady=(0, 10))

name_header = tk.Label(score_frame, text="Name", font=("Arial", 14, "bold")) name_header.grid(row=1, column=0, padx=10, sticky='w') score_header = tk.Label(score_frame, text="Score", font=("Arial", 14, "bold")) score_header.grid(row=1, column=1, padx=10, sticky='e')

for idx, (name, score) in enumerate(top_scores, start=2):
    name_label = tk.Label(score_frame, text=name, font=("Arial", 12))
    name_label.grid(row=idx, column=0, padx=10, sticky='w')
    score_label = tk.Label(score_frame, text=str(score), font=("Arial", 12))
    score_label.grid(row=idx, column=1, padx=10, sticky='e')

def handle_client(conn, addr): with conn: print(f"Connected by {addr}") data = conn.recv(1024).decode() if data: print(f"Received message: {data}") try: # Expecting format: "name:score" cleaned_data = data.strip().replace("'", '').replace('"', "") if ':' not in cleaned_data: raise ValueError("Invalid format") name, score_str = cleaned_data.split(':', 1) score = int(score_str.strip()) name = name.strip() update_scores(name, score) conn.sendall("Score received!".encode()) except ValueError: conn.sendall("Invalid format. Use name:score".encode())

def start_server(): with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as server_socket: server_socket.bind((HOST, PORT)) server_socket.listen(5) print(f"Server is listening on port {PORT}...") while True: conn, addr = server_socket.accept() client_thread = threading.Thread(target=handle_client, args=(conn, addr)) client_thread.start()

Tkinter GUI setup

root = tk.Tk() root.title("Top 10 Highest Scores") root.geometry("400x500")

score_frame = tk.Frame(root) score_frame.pack(pady=20)

Start server in a separate thread so the GUI remains responsive

server_thread = threading.Thread(target=start_server, daemon=True) server_thread.start()

root.mainloop()

Code – Client Code [Client.py]

import socket

HOST = '0.0.0.0' PORT = 65432

while True: player_name = input("Enter player name: ") score = int(input("Enter player score: "))

```
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.connect((HOST, PORT))
    s.sendall(f"{player_name},{score}".encode())
    data = s.recv(1024)
    print('Received', data.decode())
```

Evaluation

Our project was all about making a 2D Flappy Bird game using a Raspberry Pi and a pressure sensor to bring in an IoT (Internet of Things) perspective. We are hoping to display this at the Cheltenham Science Festival, aiming to engage kids and teens from 8 to 18 years old. We built the game with Python and Pygame packages, adding in features like gravity, collision detection, and a scoring system. The game runs smoothly, with fun sound effects and pixel-style graphics that make it easy on the eyes and also entertaining towards the audience. Working together, we split up tasks, so everyone had a clear role within the project, and we followed the Agile approach when we initiated our project. This helped us stay organised, handle unexpected bugs, and tweak things based on feedback. The game works well and is fun to play, but we also saw areas where we could improve, like fixing the screen tearing and making the animations smoother as Logan mentioned and background coloring as William has mentioned. We did good planning and testing to make sure everything was safe and reliable for the event and planned how we were working and also

what equipment we needed. Overall, we're proud of what we achieved. The project hit all our goals and gave us a solid chance to show off our tech skills and how well we work as a team.

Feedback (continued)

William Sanders:

"I think that the game itself is really good and does not need improvements as the gameplay is functional and runs smoothly.

However, the user interface is overly bland and needs more colour and design elements to be appealing to the user."

We took Wills feedback into consideration and added a background to the Start Screen and tested it and it worked.