ECE 180DA

Team 12

Group member: Chunhoi Wong, Jeffery Yang, YathongYam, Chenyu Liu

Final Project Proposal

Intelligent Bowling

1. Introduction

1.1 The purpose of the project

As we know, bowling is an activity that takes up a lot of space. During this covid-19 pandemic, people are required to remain their social distance. Being at the bowling alley together with friends will be much more difficult than before. This Intelligent Bowling is designed for people who want to play bowling anywhere without physically being at a bowling alley. The design allows players to practice their skills at home while enjoying more fun with the game.

In the long term, this design can apply to professional players in their daily training. Players will only need a controller and a computer to play the game. Therefore, this saves a lot of space. To make the game more fun and diverse, players can use voice control to command the bowling direction. This game has two modes: Single-player and Multiplayer. Therefore, this will allow you to play with friends and actually interact with other players.

1.2 The Items List

Hardware:

- Controller
  - Power bank
  - Raspberry Pi
  - IMU
  - Protection
- Laptop
  - Internal microphone
  - Camera
  - Graphic Display

Software:

- Unity
  - Design the graphic and the bowling alley
  - Speech to text function
- MQTT
  - Set up the publisher and subscriber

2. Project Description

2.1 General Idea

Objective one is to get the controller to interface with the game itself and allow movement to be registered by the game. When the player stops his/her movement, the game should be able to register that the controller is in static status. Then the bowling ball will begin to roll.

2.2 Basic Logistics

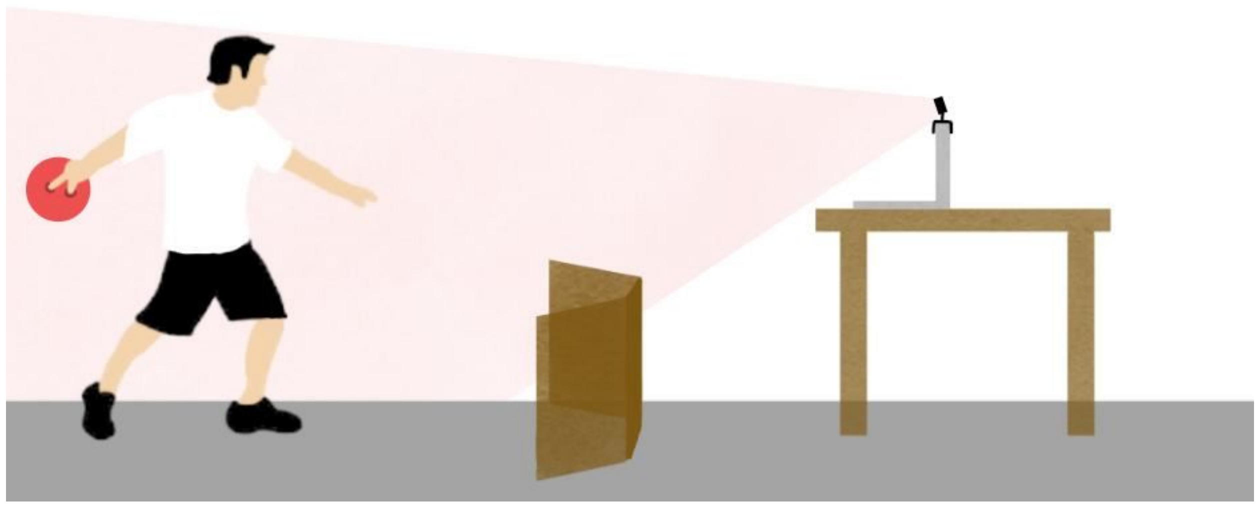


Figure 1: This graph shows the general logistics of our design and the game.

The OpenCV will start tracking and collecting data once the player says “Start”.
The gyroscope inside the controller will decide the spin and the curve of the bowling ball. The camera from the computer will capture and detect the action so that the bowling ball's moving speed will be based on how fast the controller moves from the

bottom to the top. There are three wooden plates at the end of the track(left, middle, right). On each plate, there is a colored spot with different colors so the camera can determine which plate the ball hits. If the ball hits the left or right plate, the program would consider the ball to be off track. The user needs to throw the ball again. If the ball hits the middle one, the program will start and use the data of the OpenCV and IMU to simulate its movement in Unity3D.

2.3 Additional Features

In order to give better control to the players and make this game more diverse and interactive, voice control and multiplayer mode will be added to this project. Also, we may design an AI to play against players in single-player mode.

2.3.1 Voice Control

Additional features include having voice control, where the computer recognizes voice (left or right), and adjusts the position of the bowling ball accordingly. The game display will be designed by the software Unity. The player can say “left” or “right”, and the bowling ball will move along the track to the left or right with limited speed(minor-adjust). As a result, a player who is new to this game can do a better job.

2.3.2 Multiplayer mode

Multiplayer mode: This game allows two players to play together. After a player finishes his/her turn, it becomes the second player's turn. Both players will have their scores displayed on the left screen.

2.3.3 Future Version

Players vs AI: In the next quarter, we may design an AI to play against the players with different levels of difficulty.

Sound Effect: Adding more sound effects to this game, such as creating a voice guiding system and adding a judge who will verbally update the score.

3. System Description

Multiplayer: The MQTT will be used to communicate and share information between two players. It can help us code for the game and update the player’s score immediately.

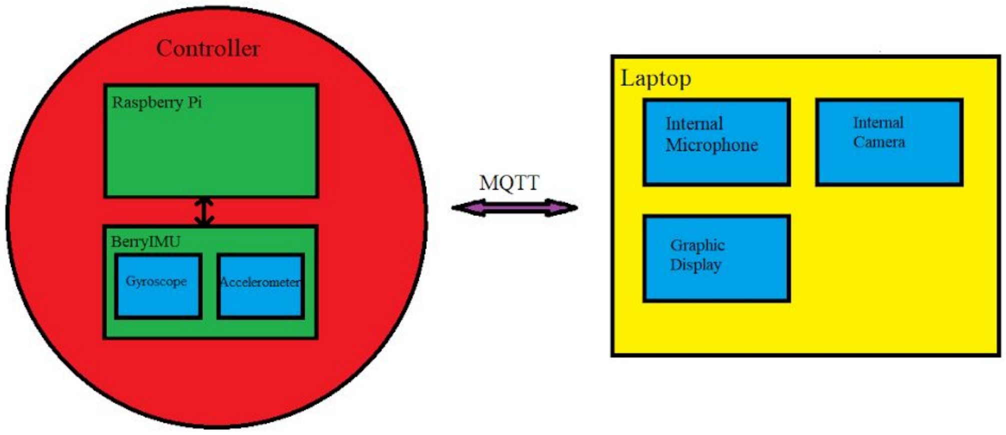


Figure 2.1 : The basic design for the setup.

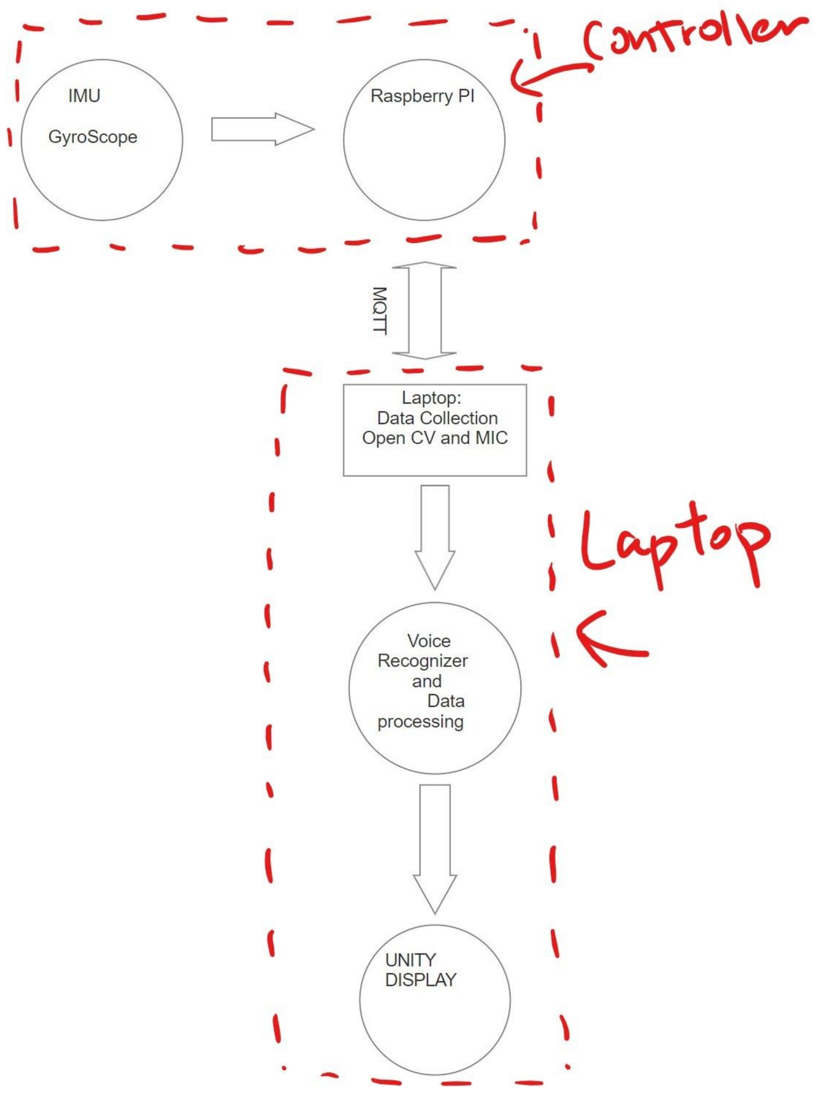

Controller
IMU
GyroScope
Raspberry PI
MQTT
Laptop: Data Collection Open CV and MIC
Laptop
Voice Recognizer and Data processing
UNITY DISPLAY

Figure 2.2 : The basic design of the block diagram.

3.1 Controller Design

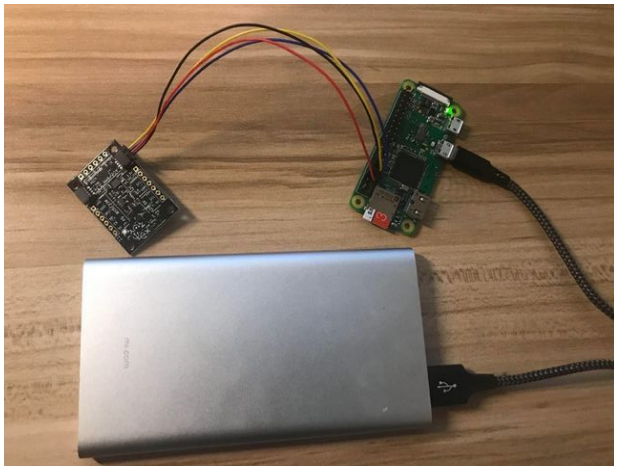


Figure 3: The hardware that we put inside the ball. The Raspberry Pi, the IMU, and the power bank will be included in the controller.

The controller will be designed with a spherical shape that mimics the shape of a real bowling ball. The controller is painted red and is filled with cotton and soft plastic material to protect the hardware. We will try our best to make the density of the sphere uniform so that the sphere would move uniformly.

Inside the sphere, we will install the Raspberry Pi and the IMU, and the power bank. The power bank will give power to the Raspberry Pi and the IMU to ensure it can wirelessly connect to our laptop.

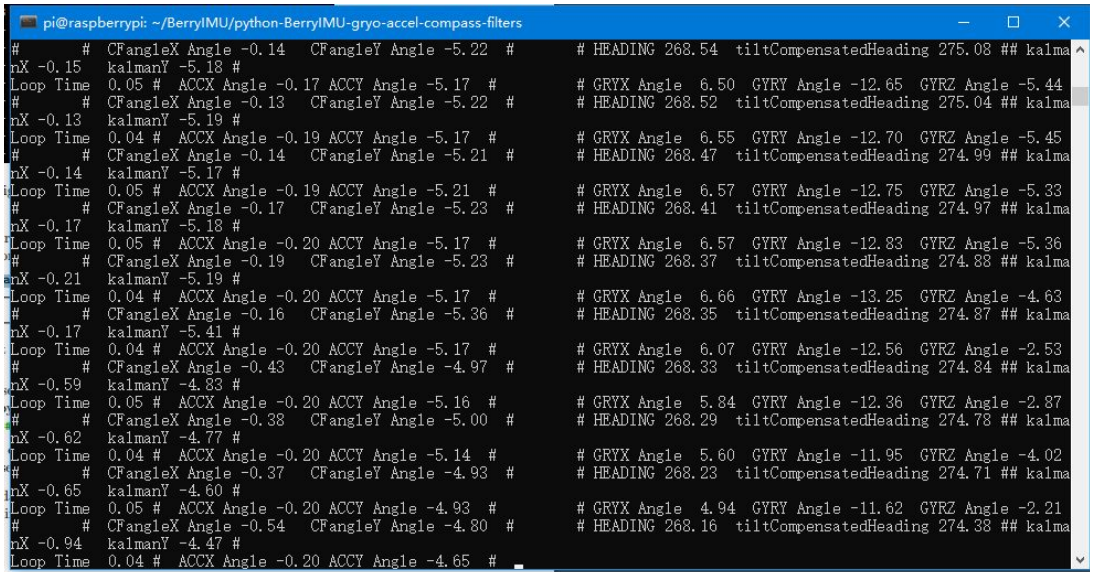
3.2 The Software Operation

In this section, we will cover the software that we use for this project and its mechanism.

3.2.1 Raspberry Pi

Raspberry Pi is programmed with python. It will be used for gathering all data from the IMU, addressing the received data from data to command, and sending commands to the laptop.

3.2.2 IMU

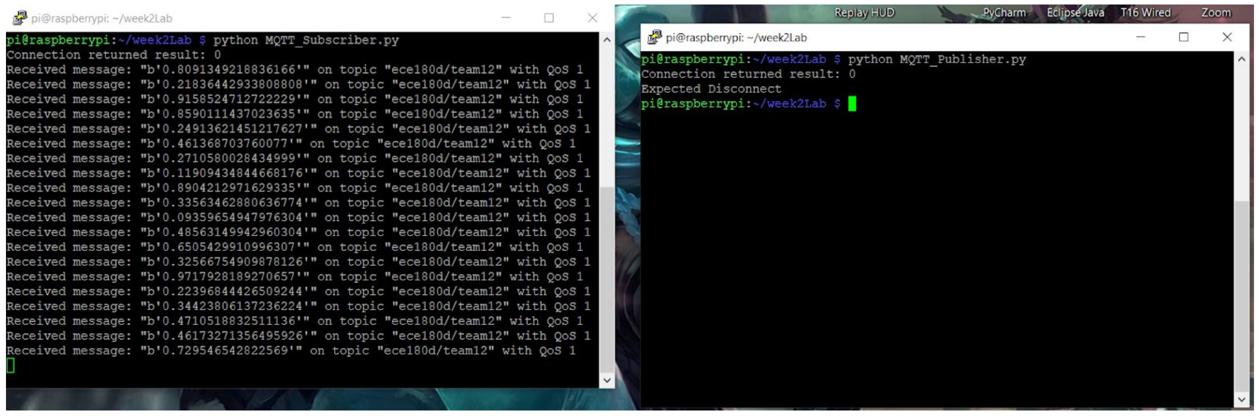


```
pi@raspberrypi: ~/BerryIMU/python-BerryIMU-gryo-accel-compass-filters
#       #  CFangleX Angle -0.14   CFangleY Angle -5.22  #       # HEADING 268.54  tiltCompensatedHeading 275.08 ## kalma
nX -0.15   kalmanY -5.18 #
Loop Time  0.05 #  ACCX Angle -0.17 ACCY Angle -5.17  #       # GRYX Angle  6.50  GYRY Angle -12.65  GYRZ Angle -5.44
#       #  CFangleX Angle -0.13   CFangleY Angle -5.22  #       # HEADING 268.52  tiltCompensatedHeading 275.04 ## kalma
nX -0.13   kalmanY -5.19 #
Loop Time  0.04 #  ACCX Angle -0.19 ACCY Angle -5.17  #       # GRYX Angle  6.55  GYRY Angle -12.70  GYRZ Angle -5.45
#       #  CFangleX Angle -0.14   CFangleY Angle -5.21  #       # HEADING 268.47  tiltCompensatedHeading 274.99 ## kalma
nX -0.14   kalmanY -5.17 #
Loop Time  0.05 #  ACCX Angle -0.19 ACCY Angle -5.21  #       # GRYX Angle  6.57  GYRY Angle -12.75  GYRZ Angle -5.33
#       #  CFangleX Angle -0.17   CFangleY Angle -5.23  #       # HEADING 268.41  tiltCompensatedHeading 274.97 ## kalma
nX -0.17   kalmanY -5.18 #
Loop Time  0.05 #  ACCX Angle -0.20 ACCY Angle -5.17  #       # GRYX Angle  6.57  GYRY Angle -12.83  GYRZ Angle -5.36
#       #  CFangleX Angle -0.19   CFangleY Angle -5.23  #       # HEADING 268.37  tiltCompensatedHeading 274.88 ## kalma
nX -0.21   kalmanY -5.19 #
Loop Time  0.04 #  ACCX Angle -0.20 ACCY Angle -5.17  #       # GRYX Angle  6.66  GYRY Angle -13.25  GYRZ Angle -4.63
#       #  CFangleX Angle -0.16   CFangleY Angle -5.36  #       # HEADING 268.35  tiltCompensatedHeading 274.87 ## kalma
nX -0.17   kalmanY -5.41 #
Loop Time  0.04 #  ACCX Angle -0.20 ACCY Angle -5.17  #       # GRYX Angle  6.07  GYRY Angle -12.56  GYRZ Angle -2.53
#       #  CFangleX Angle -0.43   CFangleY Angle -4.97  #       # HEADING 268.33  tiltCompensatedHeading 274.84 ## kalma
nX -0.59   kalmanY -4.83 #
Loop Time  0.05 #  ACCX Angle -0.20 ACCY Angle -5.16  #       # GRYX Angle  5.84  GYRY Angle -12.36  GYRZ Angle -2.87
#       #  CFangleX Angle -0.38   CFangleY Angle -5.00  #       # HEADING 268.29  tiltCompensatedHeading 274.78 ## kalma
nX -0.62   kalmanY -4.77 #
Loop Time  0.04 #  ACCX Angle -0.20 ACCY Angle -5.14  #       # GRYX Angle  5.60  GYRY Angle -11.95  GYRZ Angle -4.02
#       #  CFangleX Angle -0.37   CFangleY Angle -4.93  #       # HEADING 268.23  tiltCompensatedHeading 274.71 ## kalma
nX -0.65   kalmanY -4.60 #
Loop Time  0.05 #  ACCX Angle -0.20 ACCY Angle -4.93  #       # GRYX Angle  4.94  GYRY Angle -11.62  GYRZ Angle -2.21
#       #  CFangleX Angle -0.54   CFangleY Angle -4.80  #       # HEADING 268.16  tiltCompensatedHeading 274.38 ## kalma
nX -0.94   kalmanY -4.47 #
Loop Time  0.04 #  ACCX Angle -0.20 ACCY Angle -4.65  #
```

Figure 4: The Gyroscope data example

The gyroscope and accelerometer will determine the initial rotation factor of the ball. After that, we can determine how the movement would change due to the ball and add the feature into the Unity3D engine.

3.2.3 MQTT



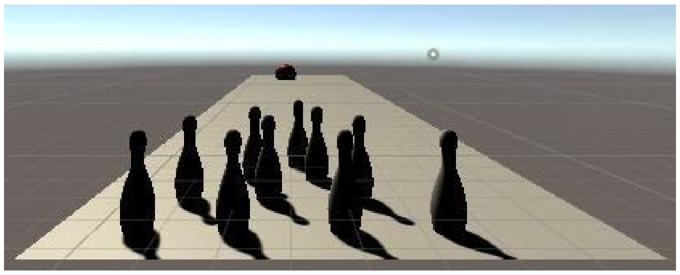
```
pi@raspberrypi:~/week2Lab $ python MQTT_Subscriber.py
Connection returned result: 0
Received message: "b'0.8091349218836166'" on topic "ece180d/team12" with QoS 1
Received message: "b'0.21836442933808808'" on topic "ece180d/team12" with QoS 1
Received message: "b'0.9158524712722229'" on topic "ece180d/team12" with QoS 1
Received message: "b'0.8590111437023635'" on topic "ece180d/team12" with QoS 1
Received message: "b'0.24913621451217627'" on topic "ece180d/team12" with QoS 1
Received message: "b'0.461368703760077'" on topic "ece180d/team12" with QoS 1
Received message: "b'0.2710580028434999'" on topic "ece180d/team12" with QoS 1
Received message: "b'0.11909434844668176'" on topic "ece180d/team12" with QoS 1
Received message: "b'0.8904212971629335'" on topic "ece180d/team12" with QoS 1
Received message: "b'0.33563462880636774'" on topic "ece180d/team12" with QoS 1
Received message: "b'0.09359654947976304'" on topic "ece180d/team12" with QoS 1
Received message: "b'0.48563149942960304'" on topic "ece180d/team12" with QoS 1
Received message: "b'0.6505429910996307'" on topic "ece180d/team12" with QoS 1
Received message: "b'0.32566754909878126'" on topic "ece180d/team12" with QoS 1
Received message: "b'0.9717928189270657'" on topic "ece180d/team12" with QoS 1
Received message: "b'0.22396844426509244'" on topic "ece180d/team12" with QoS 1
Received message: "b'0.34423806137236224'" on topic "ece180d/team12" with QoS 1
Received message: "b'0.4710518832511136'" on topic "ece180d/team12" with QoS 1
Received message: "b'0.46173271356495926'" on topic "ece180d/team12" with QoS 1
Received message: "b'0.729546542822569'" on topic "ece180d/team12" with QoS 1
```

```
pi@raspberrypi:~/week2Lab $ python MQTT_Publisher.py
Connection returned result: 0
Expected Disconnect
pi@raspberrypi:~/week2Lab $
```

Figure 5: An example of using MQTT to deliver and receive messages.

The laptop will open the MQTT subscriber and the controller will open the MQTT publisher. Once the controller moves, the raspberry pi will start receiving the data from the IMU, MQTT will deliver that data to the laptop. We then can adjust the movement on Unity.

3.2.4 Unity



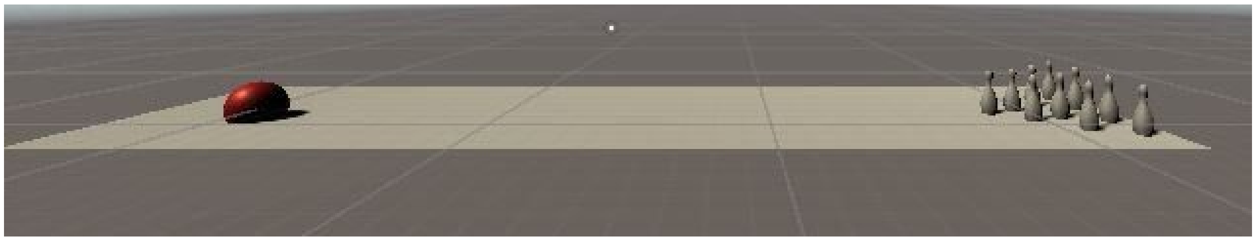


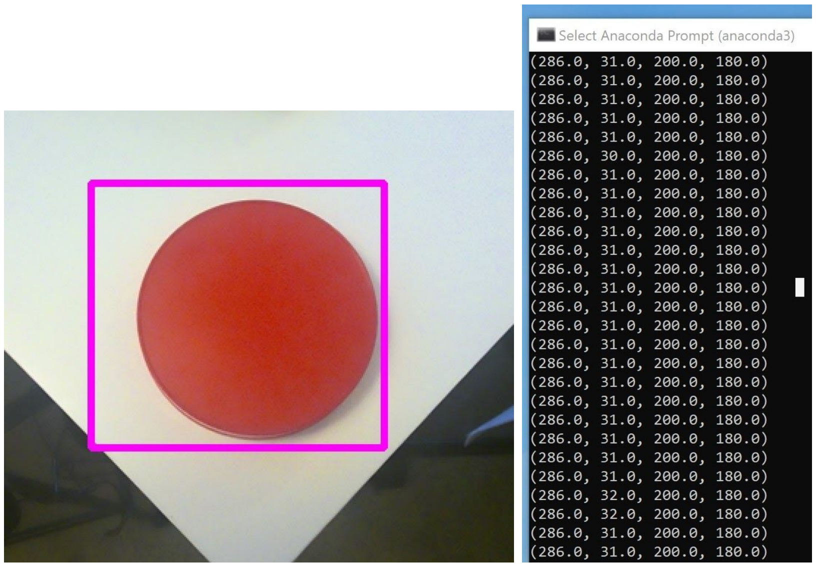
Figure 6: The bowling alley and the graph display in Unity.

The Unity game engine will be used to implement the bowling lane, ball, and pins. The user will be able to bowl the virtual bowling ball at the pins to see how many pins he/she can knock down. Unity has its own physical engine. This will help the player roll the bowling ball on the screen according to the roll in real life. The physical engine will determine the bowling ball speed and direction. Also, we will use the speech recognition library inside Unity. We will use the voice recognizer feature in Unity to determine the left or right movement directed by speech.

4. Implementation of the Design

This section explains how we implement our design, and introduces what tasks will be done in order to make our product work.

4.1 Object Capture



```
(286.0, 31.0, 200.0, 180.0)
(286.0, 31.0, 200.0, 180.0)
(286.0, 31.0, 200.0, 180.0)
(286.0, 31.0, 200.0, 180.0)
(286.0, 31.0, 200.0, 180.0)
(286.0, 30.0, 200.0, 180.0)
(286.0, 31.0, 200.0, 180.0)
(286.0, 31.0, 200.0, 180.0)
(286.0, 31.0, 200.0, 180.0)
(286.0, 31.0, 200.0, 180.0)
(286.0, 31.0, 200.0, 180.0)
(286.0, 31.0, 200.0, 180.0)
(286.0, 31.0, 200.0, 180.0)
(286.0, 31.0, 200.0, 180.0)
(286.0, 31.0, 200.0, 180.0)
(286.0, 31.0, 200.0, 180.0)
(286.0, 31.0, 200.0, 180.0)
(286.0, 31.0, 200.0, 180.0)
(286.0, 31.0, 200.0, 180.0)
(286.0, 31.0, 200.0, 180.0)
(286.0, 31.0, 200.0, 180.0)
(286.0, 31.0, 200.0, 180.0)
(286.0, 31.0, 200.0, 180.0)
(286.0, 32.0, 200.0, 180.0)
(286.0, 32.0, 200.0, 180.0)
(286.0, 31.0, 200.0, 180.0)
(286.0, 31.0, 200.0, 180.0)
```

Figure 7: Using OpenCV to capture the controller and receive data.

We will use the contour form in OpenCV to check if the tracker can always identify and locate the controller (Figure 6). To implement this, we will use the following method to enhance the accuracy of tracking the controller:

- Try to track the controller in different color backgrounds.
- Picking red as the controller color.
- Try different swinging speed

This will decrease the possibility of detection errors.

As mentioned, we also need three different colors to determine whether the ball is off the track. For each plate, there is a colored point behind it. If the ball hits any of the plates, that specific color would move and OpenCV can detect it.

4.2 Data Accuracy

In order to perform better and make this game more realistic, we need to ensure the accuracy of the data that we receive. Since the data depends on the IMU, the following tests will be done during the implementation:

- Swing the controller at the same speed and same direction to check whether the data would be the same. To observe the data differently, we can adjust tolerance by recording the Raspberry Pi.

4.3 Voice Control

We will use Unity to translate speech to text. We will decide on a few keywords, such as “left”, “right” to help control. The following tests will be done to increase its performance:

- Try the feature out in different places, including places with background noise, to check if it is still able to recognize our voice commanding/directing the bowling
- Ensuring the computer receives the correct message.

- Monitor the displacement from the screen to ensure the bowling ball is moving smoothly in the correct direction.

5. Weekly Schedule

Week 1:

- Learn to use Github, OpenCV, and Python script.
- Practice using OpenCV to make an object tracker.

Week 2:

- Install the software and set up the virtual environment for RaspberryPi.
- Brainstorm ideas for the group project.
- Discuss the group proposal.

Week 3:

- Practice using MQTT.
- Finished up the draft project proposal.
- Finalize the hardware design for the project.

Week 4:

- Learn and practice using the IMU.
- Revise the draft proposal and have the final proposal ready.
- Learn to use the gyroscope.

Week 5:

- Purchase all materials and the items for the project.
- Try to use Unity to display the gameplay.

Week 6:

- Finish up the setup for the controller.
- Ensure the gameplay can be displayed smoothly.

Week 7:

- Complete the coding for tracking objects.
- Implementing and testing the accuracy of the object tracking function.

Week 8:

- Wirelessly connect the controller with the laptop
- Testing if the game can be played for single player
- Add a voice control system.

Week 9:

- Testing the voice control system
- Added multiplayer and the score sheet
- Examine our protocol, increase the stability such as Object tracking function, efficiency, and checking the protocol performance.

Week 10:

- Finalized the protocol
- Final testing for the protocol
- Have the protocol ready

6. User Safety

6.1 Safety Design

All safety issues are our concern when we are designing this. We will introduce our design in the safety aspect below; these are also considered as risk items.

6.1.1 Controller

The controller will be designed with soft and plastic material in order to make it lighter and easier for users to handle. The controller inside will be filled with cotton to protect the hardware to prevent the chances of breaking the hardware when the controller hits the plate.

6.1.2 Plate Design

Figure 8: shows the shape of the plate.

This shape limits the controller landing position once the user throws the controller. It prevents the controller from hitting or breaking any surrounding items such as glasses or vases.

6.1.2 Power Bank

The power bank can ensure the Raspberry Pi function for at least 12 hours. However, the power bank itself may potentially have electricity leakage. Therefore, we have to make sure the power bank is safe to use before the game begins.

Reference

https://www.youtube.com/watch?v=29vyEOgsW8s (Unity)

https://learn.adafruit.com/mqtt-in-circuitpython/connecting-to-a-mqtt-broker (MQTT)

https://www.youtube.com/watch?v=AsDHEDbyLfg (MQTT)

https://www.youtube.com/watch?v=GluSLXFGfJ8&t=487s (google cloud speech to text)

https://www.youtube.com/watch?v=Eu24gIbPXoY (unity)