CRK RACING

BY:

SANDHYA MALLA and Team

Unit Test

- Camera: To test the camera's scanning functionality and quality correctly.
- OpenCV: To verify that OpenCV recognizes and interprets QR codes accurately.
- QR Code Scanning: To test the entire QR code scanning starting from detection to response.
- Car Connection: To test the connectivity and communication between the app and the car.
- Controller App: To test whether the app correctly sends control signals to the Kart or not.
- Hit and Stop: To validate the precise execution of the hit-and-stop feature.
- Vehicle Communication: The communication between vehicles will result in an almost instant execution of the power-up against opponents.

Integration Test

Bottom-Up Approach:

- Camera and OpenCV: Integrate and test video processing functionality.
- QR Code Scanning: Add QR scanning capabilities to the camera.
- Car Connection: Connect sensors and Raspberry Pi with the app.
- Controller App: Combine the app and car connection for end-to-end functionality.
- Hit and Stop: Integrate the hit-and-stop feature with the main logic.
- Vehicle Communication: Connect multiple vehicles for the purpose of communicating information.

Big Bang Testing:

- This involves testing all modules together as a fully integrated system in a live racing environment to ensure that they interact seamlessly and meet performance expectations.

Business Model Canvas

The Business Model Canvas

Designed for: CRKR

Designed by: Group 2

Date: 11/17/24

Version:

Key Partnerships

1. Sponsor: Tarick Walton
2. Resources Provided: RC Cars, Keyboard, Mouse, Track
3. Software Libraries: Open CV, Socket IO

Key Activities

1. Designing Power-up features
2. Integrating Cameras for QR code scanning
3. Communication between cars

Key Resources

1. Materialistic Resources: RC Cars
2. Human Skills: Python, Computer Vision, Raspberry Pi Hardware
3. Funding

Value Propositions

1. Unique Features: RC Cars that have features such as the Mario Kart Game
2. Low Competition
3. Scalable to bigger vechicles
4. Enhances traditional RC car racing with unique features

Customer Relationships

1. Direct interaction with users through demos
2. Engaing with users on social media and gaming platforms
3. Feedback collection to improve features

Channels

1. Racing Events
2. Trade Shows
3. Software Installation

Customer Segments

1. Nintendo Fans
2. Gamers looking for real-life MK experiences
3. Curious SWE or Students
4. Event organizers looking for unique attractions
5. More sponsors

Cost Structure

1. Salaries and benefits
2. Buying “Cars”
3. Marketing and Sales Expenses
4. Licensing or Technology fees
5. Legal/Copyright cost
6. Software/ hardware Maintenance

Revenue Streams <u>Options</u>

1. Subscription Fees
2. Software Sales
3. Service Charges| Updates fees
4. Licensing or franchising revenue
5. Hosting Gaming competitions or tournaments

PROGRAM UNITS

connect():

- Purpose: The method establishes a websocket connection with the opposing kart. It then maintains this connection for sending and receiving messages. It spends the majority of its time waiting for messages. When a message is received, it sets a bool flag to trigger changes in the game logic.
- Planned Clients: Websocket connections, gameLogic method
- Dependencies: Asyncio, Threading, Websockets

gameLogic:

- Purpose: The method gameLogic contains the logic required for the game to run. It checks for whether or not the kart is hit and performs the appropriate action. It also calls the sendMessage function to tell the other kart when it has been hit.
- Planned Clients: Kart control system, Kart communication system, connect method
- Dependencies: Asyncio

sendMessage(String message):

- Purpose: The method sendMessage sends a string to the opposing kart using a websocket connection.
- Planned Clients: gameLogic method
- Dependencies: Asyncio, Threading, Websockets

3RD PARTY PROGRAM UNITS

Movement Control Program Units

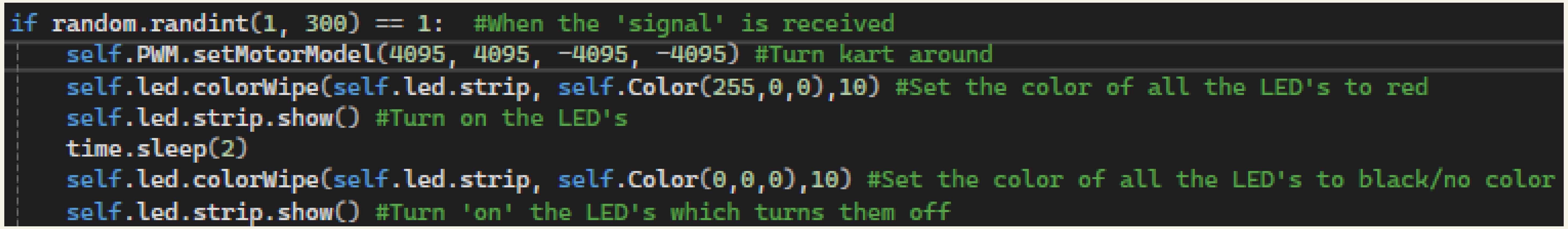
- on_btn_ForWard() - Moves forward.
- on_btn_BackWard() - Moves backward.
- on_btn_Turn_Left() - Sharp turn left.
- on_btn_Moveleft() - Moves left.
- on_btn_Dialeft() - Moves forward and left.
- on_btn_Diad_left() - Moves backward and left.
- on_btn_Turn_Right() - Sharp turn right.
- on_btn_Moveright() - Moves right.
- on_btn_Diaright() - Moves forward and right.
- on_btn_Diad_right() - Moves backwards and right.
- on_btn_Stop() - Stops the kart.

Other Program Units

- mywindow() – The main class that enables the mobile app to connect to the kart and control it.
- on_btn_Video() – Turns the camera on so it can start sending video data to the mobile app.
- on_btn_Connect() – Allows someone with the mobile app to connect and control this respective kart.

Software in Action

Kart Being Hit



```
if random.randint(1, 300) == 1:  #When the 'signal' is received
    self.PWM.setMotorModel(4095, 4095, -4095, -4095) #Turn kart around
    self.led.colorWipe(self.led.strip, self.Color(255,0,0),10) #Set the color of all the LED's to red
    self.led.strip.show() #Turn on the LED's
    time.sleep(2)
    self.led.colorWipe(self.led.strip, self.Color(0,0,0),10) #Set the color of all the LED's to black/no color
    self.led.strip.show() #Turn 'on' the LED's which turns them off
```

THE END

Any Questions?