Week 1: Introduction to Donkey Car and AI

- Introduction (5 minutes)
  - Welcome students and provide an overview of the lesson plan.
  - Explain the importance of AI in autonomous vehicles and its applications.

- What is Donkey Car? (10 minutes)
  - Explain the concept of Donkey Car as an open-source DIY self-driving platform.
  - Discuss the various components of a Donkey Car, such as the chassis, motor, camera, and microprocessor.

- Introduction to AI (15 minutes)
  - Define AI and its applications in various fields.
  - Discuss the role of AI in autonomous vehicles and its importance in Donkey Car.

- Activity: AI Examples (20 minutes)
  - Provide examples of AI applications in everyday life, such as voice assistants, recommendation systems, and image recognition.
  - Discuss how AI can be applied to Donkey Car to enable autonomous driving.

- Recap and Homework (10 minutes)
  - Summarize the key points covered in the lesson.
  - Assign homework: Research and find one real-world example of AI in autonomous vehicles.

Python Code Example:

```python
# Example Python code for controlling Donkey Car

# Import necessary libraries
import time

# Define a function to drive the car forward
def drive_forward():
    # Code to control the motors and drive the car forward
    print("Driving forward...")

# Call the function to drive the car forward
drive_forward()
```

MicroPython Code Example:

```python
# Example MicroPython code for controlling Donkey Car

# Import necessary libraries
import time

# Define a function to drive the car forward
def drive_forward():
    # Code to control the motors and drive the car forward
    print("Driving forward...")

# Call the function to drive the car forward
drive_forward()
```

Week 2: Selecting the Right Microprocessor for Donkey Car

- Recap and Discussion (10 minutes)
  - Review the homework and discuss the real-world examples of AI in autonomous vehicles.
  - Encourage students to share their findings and insights.

- Introduction to Microprocessors (15 minutes)
  - Explain the role of microprocessors in Donkey Car and their importance in AI applications.
  - Discuss the different types of microprocessors commonly used in DIY robotics.

- Factors to Consider (15 minutes)
  - Discuss the factors to consider when selecting a microprocessor, such as processing power, memory, compatibility, and cost.
  - Explain the trade-offs between different microprocessors and their impact on the performance of Donkey Car.

- Activity: Microprocessor Selection (20 minutes)
  - Provide a list of microprocessors commonly used in Donkey Car projects.
  - Assign students to research and compare the specifications, capabilities, and costs of different microprocessors.
  - Instruct students to select the most suitable microprocessor for their Donkey Car project based on their requirements and budget.

- Recap and Homework (10 minutes)
  - Summarize the key points covered in the lesson.
  - Assign homework: Write a short paragraph explaining their microprocessor selection and justification.

Python Code Example:

```python
# Example Python code for microprocessor selection

# Define a list of microprocessors
microprocessors = ["Raspberry Pi", "Arduino", "Jetson Nano", "ESP32"]

# Select the most suitable microprocessor based on requirements
selected_microprocessor = "Raspberry Pi"

# Print the selected microprocessor
print("Selected Microprocessor:", selected_microprocessor)
```

MicroPython Code Example:

```python
# Example MicroPython code for microprocessor selection

# Define a list of microprocessors
microprocessors = ["Raspberry Pi", "Arduino", "ESP32"]

# Select the most suitable microprocessor based on requirements
selected_microprocessor = "Raspberry Pi"

# Print the selected microprocessor
print("Selected Microprocessor:", selected_microprocessor)
```

Week 3: Building and Configuring Donkey Car

- Recap and Discussion (10 minutes)
  - Review the homework and discuss the students' microprocessor selections.
  - Encourage students to share their reasoning and discuss any challenges they encountered.

- Building Donkey Car (20 minutes)
  - Provide step-by-step instructions on assembling the Donkey Car components, including the chassis, motor, camera, and microprocessor.
  - Demonstrate the correct wiring and connections.

- Configuring Donkey Car (20 minutes)
  - Explain the process of configuring the microprocessor for Donkey Car.
  - Discuss the necessary software installations, libraries, and dependencies.
  - Guide students through the configuration process, ensuring they understand each step.

- Activity: Test Drive (10 minutes)
  - Allow students to test drive their Donkey Car in a controlled environment.
  - Encourage them to observe and document any issues or errors they encounter.

- Recap and Homework (10 minutes)
  - Summarize the key points covered in the lesson.
  - Assign homework: Troubleshoot any issues encountered during the test drive and document the solutions.

Python Code Example:

```python
# Example Python code for building and configuring Donkey Car

# Import necessary libraries
import RPi.GPIO as GPIO

# Set up GPIO pins for motor control
```

```
GPIO.setmode(GPIO.BCM)
GPIO.setup(17, GPIO.OUT)
GPIO.setup(18, GPIO.OUT)

# Function to drive the car forward
def drive_forward():
    GPIO.output(17, GPIO.HIGH)
    GPIO.output(18, GPIO.LOW)

# Call the function to drive the car forward
drive_forward()
```

MicroPython Code Example:

```python
# Example MicroPython code for building and configuring Donkey Car

# Import necessary libraries
from machine import Pin

# Set up GPIO pins for motor control
motor1 = Pin(17, Pin.OUT)
motor2 = Pin(18, Pin.OUT)

# Function to drive the car forward
def drive_forward():
    motor1.on()
    motor2.off()

# Call the function to drive the car forward
drive_forward()
```

Week 4: Testing and Implementing Code with Donkey Car

- Recap and Discussion (10 minutes)
  - Review the key points covered in the previous lesson about building and configuring Donkey Car.
  - Discuss any issues or challenges encountered during the test drive and troubleshooting process.

- Testing Forward Driving (15 minutes)
  - Explain the code for driving the car forward and its implementation using the selected microprocessor.
  - Demonstrate how to test the forward driving functionality of Donkey Car.
  - Instruct students to test the forward driving code on their own Donkey Car and observe the behavior.

- Testing Turns (15 minutes)
  - Introduce the code for making turns and its implementation using the selected microprocessor.
  - Demonstrate how to test the turning functionality of Donkey Car.

- Instruct students to test the turning code on their own Donkey Car and observe the behavior.

- Activity: Implementing Remote Control with Autonomy (20 minutes)
  - Explain the concept of remote control with autonomy, where the car can be controlled remotely but also has autonomous capabilities.
  - Provide code examples for implementing remote control and autonomy using the selected microprocessor.
- Instruct students to modify the code to add remote control functionality to their Donkey Car.

Here are some examples of implementing remote control with autonomy in Donkey Car:

1. Remote Control Mode:
   - Allow the user to control the Donkey Car remotely using a joystick or a smartphone app.
   - Implement code to receive input from the joystick or app and translate it into motor control commands.
   - The Donkey Car will move according to the user's input, allowing for manual control.

2. Autonomous Mode:
   - Implement code for autonomous driving using AI algorithms, such as deep learning or computer vision.
   - The Donkey Car will use its sensors and camera to perceive the environment and make decisions on its own.
   - The car can follow a predetermined path, avoid obstacles, or perform specific tasks like object detection or lane following.

3. Hybrid Mode:
   - Combine remote control and autonomy to create a hybrid mode.
   - Allow the user to switch between manual control and autonomous driving at any time.
   - The Donkey Car can be controlled remotely when needed, and then switch to autonomous mode for specific tasks or when the user wants to take a break.

4. Safety Overrides:
   - Implement safety features that allow the user to regain control of the Donkey Car in case of emergencies or unexpected situations.
   - For example, the user can press a specific button or trigger a command to immediately stop the car or switch to manual control.

5. Telemetry and Feedback:
   - Provide real-time telemetry and feedback to the user during remote control and autonomous driving.
   - Display information such as speed, sensor readings, camera feed, and motor control commands on a dashboard or a smartphone app.

These examples demonstrate how remote control and autonomy can be combined to create a flexible and interactive experience with Donkey Car. The user can enjoy the control and excitement of manual driving while also benefiting from the intelligence and capabilities of autonomous driving.

- Recap and Homework (10 minutes)
  - Summarize the key points covered in the lesson.

  - Assign homework: Modify the code to add additional functionalities to Donkey Car, such as obstacle detection or line following.

Python Code Example:

```python
# Example Python code for testing and implementing code with Donkey Car

# Import necessary libraries
import RPi.GPIO as GPIO

# Set up GPIO pins for motor control
GPIO.setmode(GPIO.BCM)
GPIO.setup(17, GPIO.OUT)
GPIO.setup(18, GPIO.OUT)

# Function to drive the car forward
def drive_forward():
    GPIO.output(17, GPIO.HIGH)
    GPIO.output(18, GPIO.LOW)

# Function to turn the car left
def turn_left():
    GPIO.output(17, GPIO.LOW)
    GPIO.output(18, GPIO.LOW)

# Function to turn the car right
def turn_right():
    GPIO.output(17, GPIO.HIGH)
    GPIO.output(18, GPIO.HIGH)

# Call the function to drive the car forward
drive_forward()
```

MicroPython Code Example:

```python
# Example MicroPython code for testing and implementing code with Donkey Car

# Import necessary libraries
from machine import Pin

# Set up GPIO pins for motor control
motor1 = Pin(17, Pin.OUT)
motor2 = Pin(18, Pin.OUT)

# Function to drive the car forward
def drive_forward():
    motor1.on()
    motor2.off()
```

```
# Function to turn the car left
def turn_left():
    motor1.off()
    motor2.off()

# Function to turn the car right
def turn_right():
    motor1.on()
    motor2.on()

# Call the function to drive the car forward
drive_forward()
```

Week 5: Sensors and Camera Selection for Machine Vision

- Recap and Discussion (10 minutes)
  - Review the key points covered in the previous lesson about testing and implementing code with Donkey Car.
  - Discuss any modifications or additional functionalities added to Donkey Car by the students.

- Introduction to Sensors (15 minutes)
  - Explain the role of sensors in Donkey Car and their importance in machine vision.
  - Discuss the different types of sensors commonly used in DIY robotics, such as ultrasonic sensors, infrared sensors, and line sensors.

- Selecting the Most Helpful Sensors (15 minutes)
  - Discuss the factors to consider when selecting sensors for Donkey Car, such as the desired functionality, accuracy, range, and cost.
  - Provide examples of sensor applications in Donkey Car, such as obstacle detection, line following, and distance measurement.

- Camera Selection for Machine Vision (15 minutes)
  - Explain the importance of a camera in machine vision and its role in Donkey Car.
  - Discuss the different types of cameras commonly used in DIY robotics, such as USB cameras, Raspberry Pi cameras, and AI cameras.
  - Provide guidance on selecting the most suitable camera for Donkey Car based on the desired image quality, resolution, and compatibility.

- Recap and Homework (10 minutes)
  - Summarize the key points covered in the lesson.
  - Assign homework: Research and select the most suitable sensors and camera for their Donkey Car project based on their requirements and budget.

Python Code Example:

```python
# Example Python code for sensor and camera selection
# Define a list of sensors
sensors = ["Ultrasonic Sensor", "Infrared Sensor", "Line Sensor"]

# Select the most suitable sensor based on requirements
selected_sensor = "Ultrasonic Sensor"

# Print the selected sensor
print("Selected Sensor:", selected_sensor)

# Define a list of cameras
cameras = ["USB Camera", "Raspberry Pi Camera", "AI Camera"]

# Select the most suitable camera based on requirements
selected_camera = "Raspberry Pi Camera"

# Print the selected camera
print("Selected Camera:", selected_camera)
```

MicroPython Code Example:

```python
# Example MicroPython code for sensor and camera selection

# Define a list of sensors
sensors = ["Ultrasonic Sensor", "Infrared Sensor", "Line Sensor"]

# Select the most suitable sensor based on requirements
selected_sensor = "Ultrasonic Sensor"

# Print the selected sensor
print("Selected Sensor:", selected_sensor)

# Define a list of cameras
cameras = ["USB Camera", "Raspberry Pi Camera", "AI Camera"]

# Select the most suitable camera based on requirements
selected_camera = "Raspberry Pi Camera"

# Print the selected camera
print("Selected Camera:", selected_camera)
```

Week 6: Saving Sensor Logs and Training Data

- Recap and Discussion (10 minutes)
  - Review the key points covered in the previous lesson about sensor and camera selection.
  - Discuss the students' sensor and camera selections and their reasoning.

- Importance of Saving Sensor Logs (10 minutes)
  - Explain the importance of saving sensor logs in Donkey Car for data analysis and training purposes.
  - Discuss the types of data that can be logged, such as sensor readings, motor control commands, and GPS coordinates.

- Methods to Save Sensor Logs (15 minutes)
  - Introduce different methods to save sensor logs in Donkey Car, such as writing to a file, sending data to a cloud server, or using a data logging module.
  - Discuss the advantages and disadvantages of each method and their impact on the performance of Donkey Car.

- Training Data and Its Reusability (15 minutes)
  - Explain the concept of training data in Donkey Car and its role in machine learning.
  - Discuss the process of collecting and labeling training data for different tasks, such as object detection, lane following, and obstacle avoidance.
  - Highlight the reusability of training data and its potential for improving the performance of Donkey Car.

- Recap and Homework (10 minutes)
  - Summarize the key points covered in the lesson.
  - Assign homework: Implement a method to save sensor logs in their Donkey Car project and collect training data for a specific task.

Python Code Example:

```python
# Example Python code for saving sensor logs and training data

# Import necessary libraries
import csv

# Function to save sensor logs to a CSV file
def save_sensor_logs(sensor_data):
    with open('sensor_logs.csv', 'w', newline='') as file:
        writer = csv.writer(file)
        writer.writerow(["Timestamp", "Sensor Data"])
        for data in sensor_data:
            writer.writerow([data["timestamp"], data["sensor_data"]])

# Define a list of sensor data
sensor_data = [
    {"timestamp": "2022-01-01 10:00:00", "sensor_data": 10},
    {"timestamp": "2022-01-01 10:01:00", "sensor_data": 15},
    {"timestamp": "2022-01-01 10:02:00", "sensor_data": 20}
]

# Call the function to save sensor logs
save_sensor_logs(sensor_data)
```

MicroPython Code Example:

```python
# Example MicroPython code for saving sensor logs and training data

# Import necessary libraries
import ucsv as csv

# Function to save sensor logs to a CSV file
def save_sensor_logs(sensor_data):
    with open('sensor_logs.csv', 'w', newline=") as file:
        writer = csv.writer(file)
        writer.writerow(["Timestamp", "Sensor Data"])
        for data in sensor_data:
            writer.writerow([data["timestamp"], data["sensor_data"]])

# Define a list of sensor data
sensor_data = [
    {"timestamp": "2022-01-01 10:00:00", "sensor_data": 10},
    {"timestamp": "2022-01-01 10:01:00", "sensor_data": 15},
    {"timestamp": "2022-01-01 10:02:00", "sensor_data": 20}
]

# Call the function to save sensor logs
save_sensor_logs(sensor_data)
```