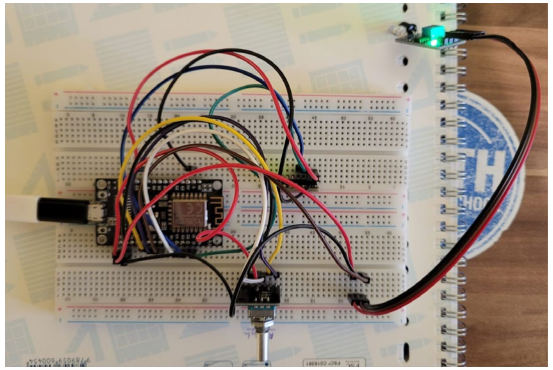
Project Overview

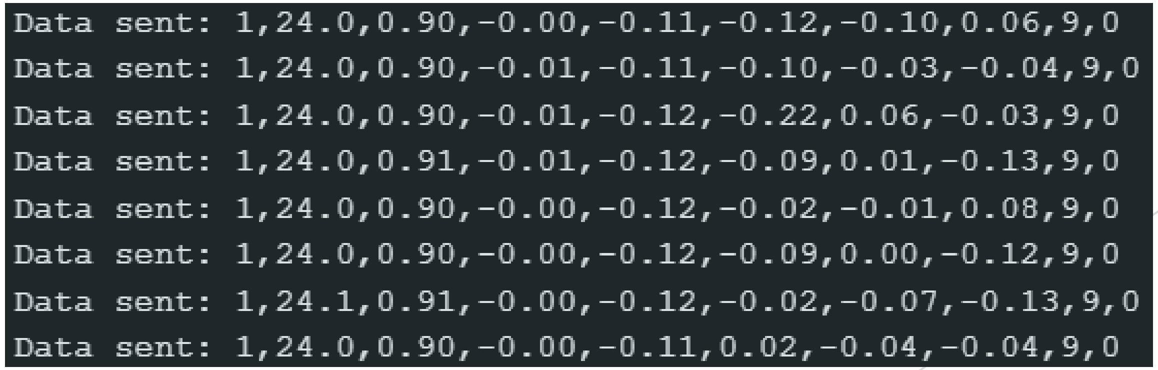
Sensor Integration with Unity Game Environment

- **Goal:** Seamless integration of hardware sensors with a Unity game environment.
- **Hardware Components:** ESP8266, MPU6050, rotary encoder, proximity sensor.
- **Software Components:** Unity Engine, Arduino IDE, and C# scripts.
- **Core Features:**
  - Real-time data transfer using User Datagram Protocol (UDP) protocol.
  - Interactive control of a 3D environment using sensor inputs.

ESP8266 Setup

- ESP8266 Sensor Integration to capture data from sensors and transmit it to Unity.
- Reads proximity, MPU6050 (temperature, acceleration & angular velocity), and rotary encoder data.
- Sends data in a comma-separated format via UDP.
- **Highlights**:
  - Optimized for low-latency communication.
  - Modular code for adding/removing sensors.
  - setup(): Initializes sensors and Wi-Fi connection.
  - loop(): Continuously collects sensor data and sends it using Wi-Fi UDP.



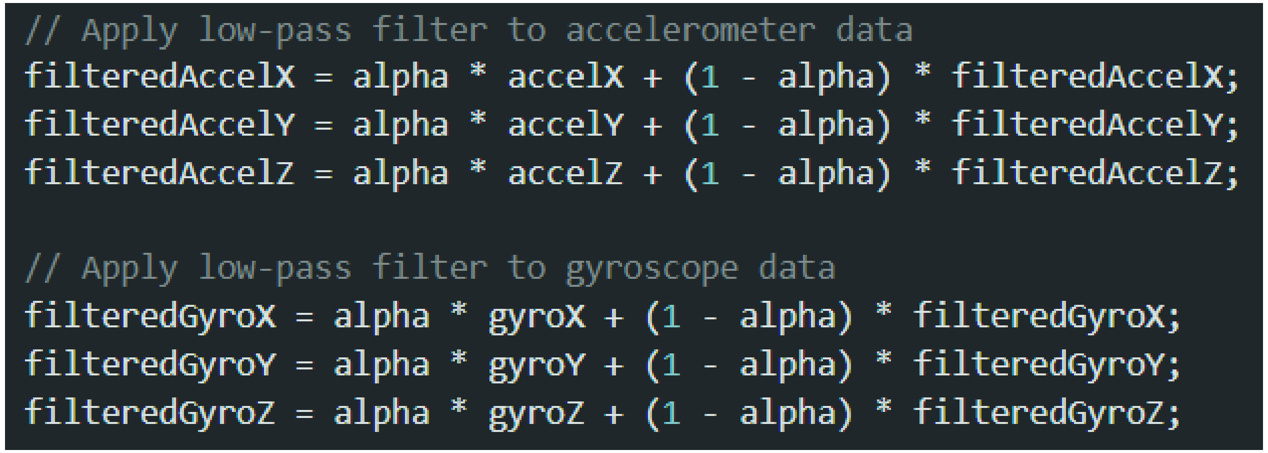


```
Data sent: 1,24.0,0.90,-0.00,-0.11,-0.12,-0.10,0.06,9,0
Data sent: 1,24.0,0.90,-0.01,-0.11,-0.10,-0.03,-0.04,9,0
Data sent: 1,24.0,0.90,-0.01,-0.12,-0.22,0.06,-0.03,9,0
Data sent: 1,24.0,0.91,-0.01,-0.12,-0.09,0.01,-0.13,9,0
Data sent: 1,24.0,0.90,-0.00,-0.12,-0.02,-0.01,0.08,9,0
Data sent: 1,24.0,0.90,-0.00,-0.12,-0.09,0.00,-0.12,9,0
Data sent: 1,24.1,0.91,-0.00,-0.12,-0.02,-0.07,-0.13,9,0
Data sent: 1,24.0,0.90,-0.00,-0.11,0.02,-0.04,-0.04,9,0
```

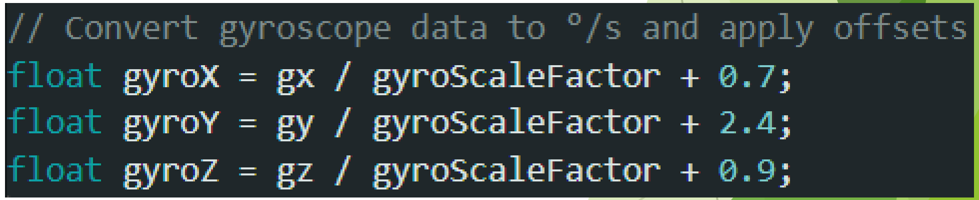
UDP

Low-pass filter and zero-error correction

- Implemented a low-pass filter for noise suppression.
- **Exponential Moving Average:**
  - smoothes out the noise by giving more weight to recent data while gradually "forgetting" older data.
  - filteredValue = alpha * newValue + (1 - alpha) * previousFilteredValue
  - Alpha controls the fractional contribution of the newValue.
- Found significant zero errors in the gyroscope readings. Applied offsets to nullify.



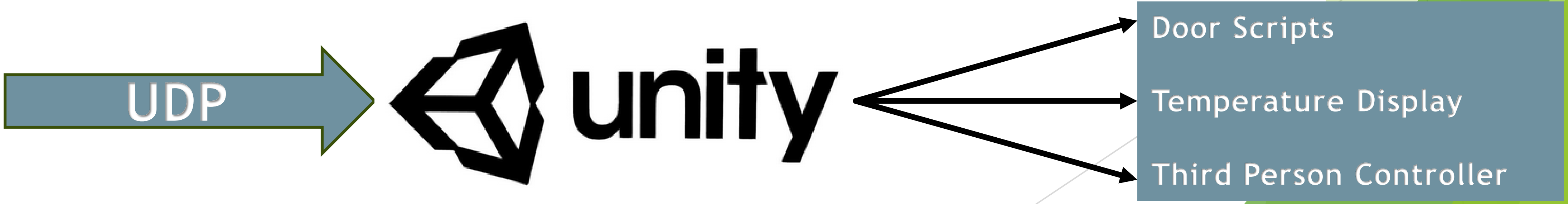
```
// Apply low-pass filter to accelerometer data
filteredAccelX = alpha * accelX + (1 - alpha) * filteredAccelX;
filteredAccelY = alpha * accelY + (1 - alpha) * filteredAccelY;
filteredAccelZ = alpha * accelZ + (1 - alpha) * filteredAccelZ;

// Apply low-pass filter to gyroscope data
filteredGyroX = alpha * gyroX + (1 - alpha) * filteredGyroX;
filteredGyroY = alpha * gyroY + (1 - alpha) * filteredGyroY;
filteredGyroZ = alpha * gyroZ + (1 - alpha) * filteredGyroZ;
```



```
// Convert gyroscope data to °/s and apply offsets
float gyroX = gx / gyroScaleFactor + 0.7;
float gyroY = gy / gyroScaleFactor + 2.4;
float gyroZ = gz / gyroScaleFactor + 0.9;
```

UDP Receiver in Unity

- Receives sensor data from the ESP8266 and processes it in Unity.
  - Listens on port 12345 for UDP packets.
  - Parses comma-separated values into meaningful data points.
  - Real-time updates to Unity game objects (e.g., temperature display)
- **Highlights**:
  - OnUDPDataReceived(): Receives data and calls ParseReceivedData().
  - ParseReceivedData(): Converts data into variables for temperature, proximity, encoder position and MPU readings (e.g., Gyroscope)
  - SendDataToScripts(): Updates other Unity scripts (e.g., doors, temperature display).



UDP

unity

Door Scripts

Temperature Display

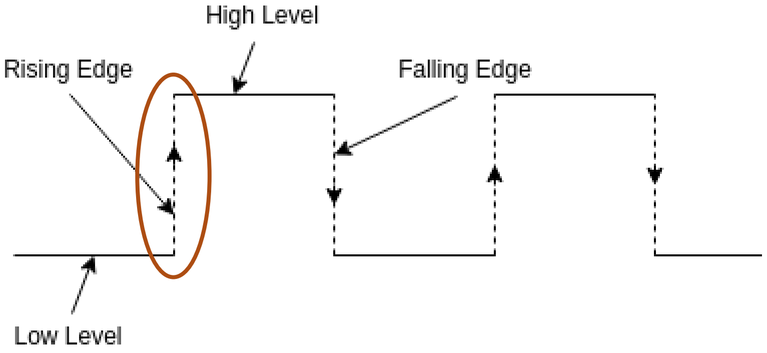
Third Person Controller

Doors Mechanism in Unity

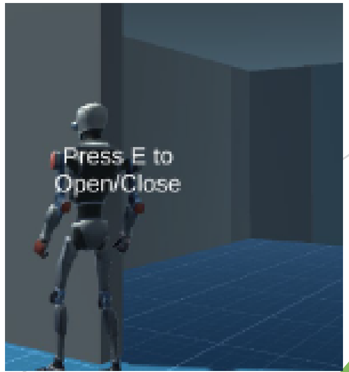
- Controls the toggle action of the door using proximity sensor output.

- **Door Scripts**:
  - Each door is controlled using a dedicated script (e.g., Door1, Door2, Door3).
  - CheckHigh1(): checks for highprox signals which is a rising edge detector of the proximity sensor.
  - Proximity detection toggles door only if the player is in range of that particular door.
  - Door functionality has been taken from the original door script.

Proximity Sensor data



Toggle Door

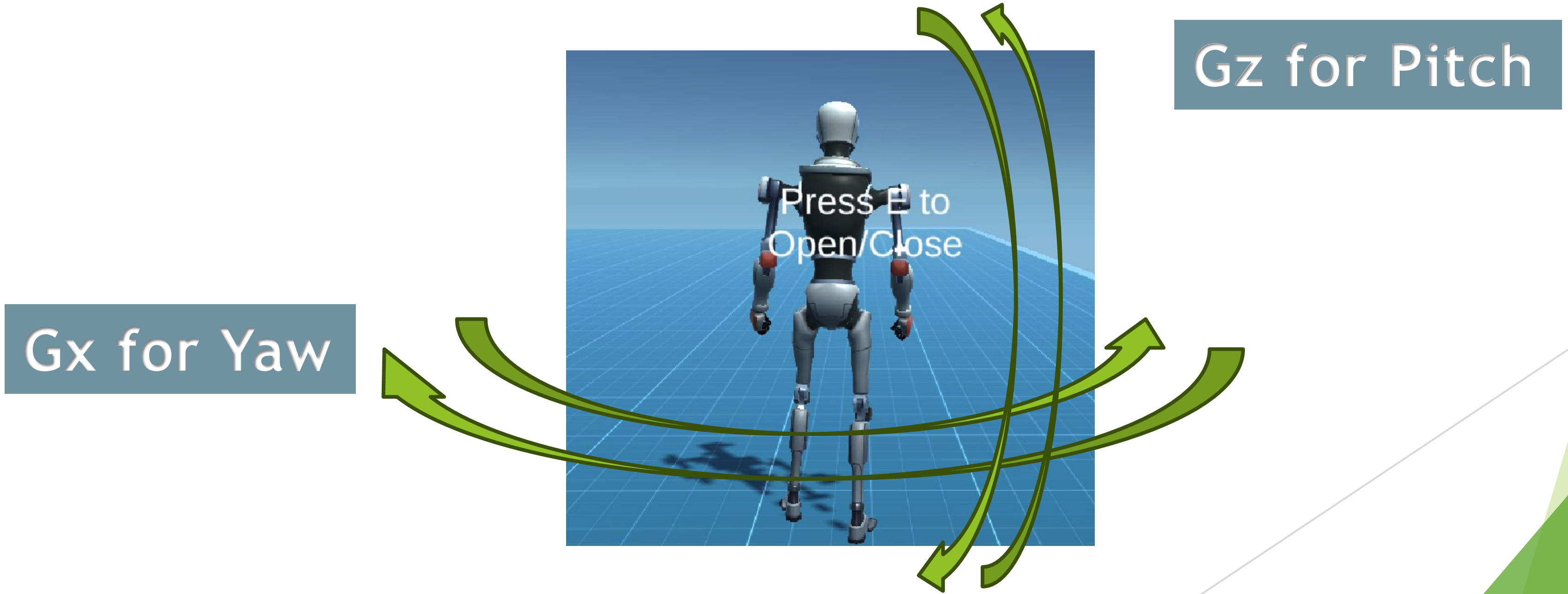

Press E to Open/Close

Temperature Display

- Displays real-time temperature in the Unity environment.
- TemperatureDisplay script defines a function updates a function UpdateTemperature()
- Receives temperature values and updates the UI element TextMeshPro in the game environment.

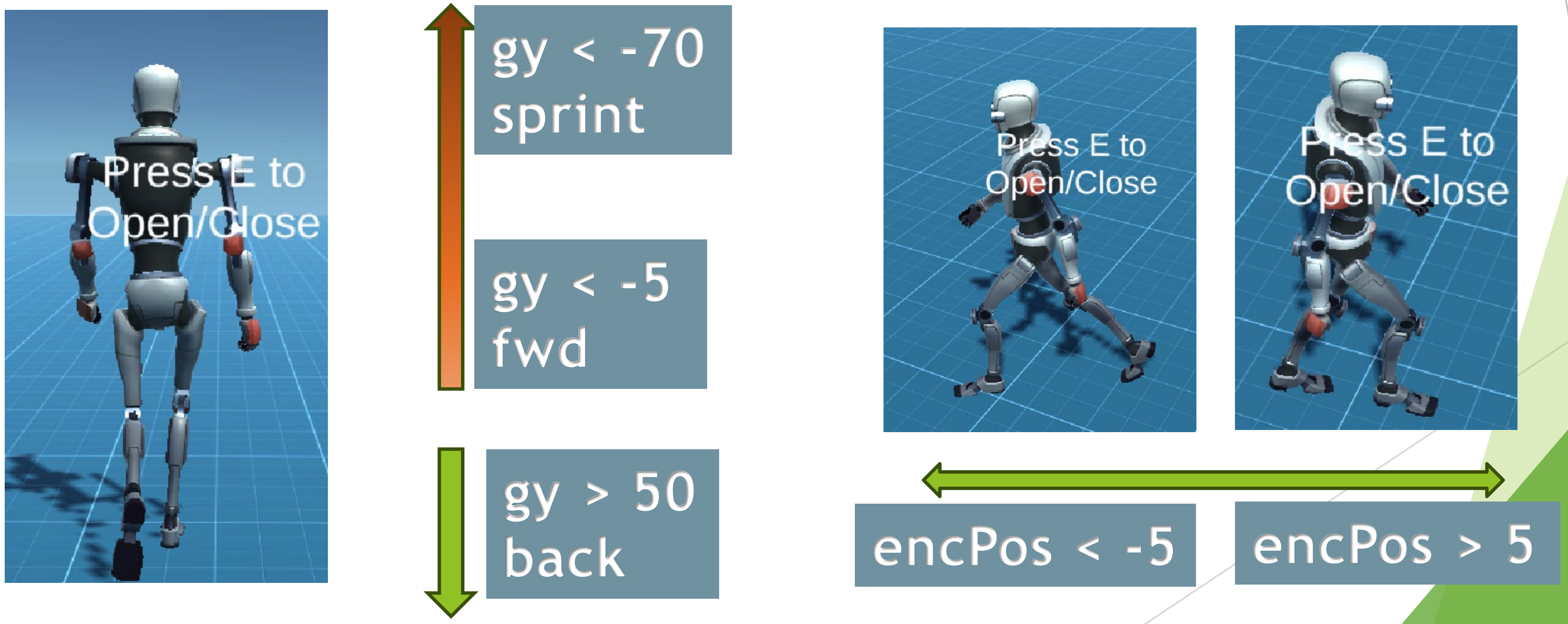
Camera and Third Person Controller

- Enables user control of the camera and the robot character in the Unity environment.
- Modified CameraRotation() for camera control:
  - Uses gyroscope values UDPReceiver.gx and UDPReceiver.gz for the Yaw and Pitch respectively.
  - Converts the angular velocities to angular displacements by multiplying with the delay interval set in Arduino code.



Camera and Third Person Controller

- Modified Move() for Robot Movement:
  - Uses gyroscope values UDPReceiver.gy for the forward, backward and sprint movement based on certain threshold values.
  - Uses UDPReceiver.encoderPosition for left and right movement based on certain threshold values.



gy < -70
sprint

gy < -5
fwd

gy > 50
back

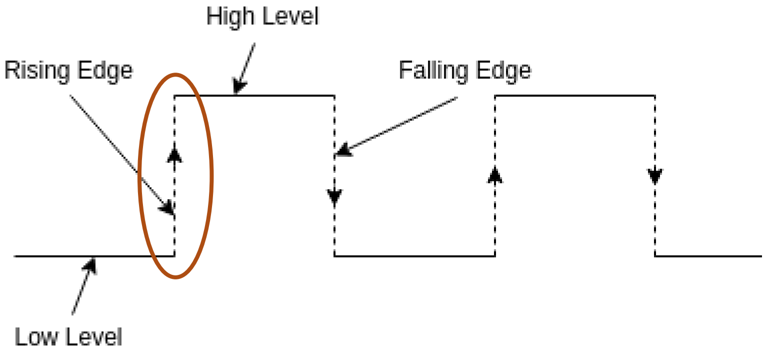
encPos < -5

encPos > 5

Camera and Third Person Controller

- highencswitch variable acts as the rising edge detector for the encoder switch.
- Modified JumpAndGravity():
  - Checks for highencswitch and activates the jump action when value is true.

Encoder Switch data



Jump