PISTON
Journal of Technical Engineering:

Unreal Engine Based MCS Program Using Microcontroller Wemos D1 Mini, MPU-9250 Sensor and UDP Protocol

Frengki Simatupang[1,a)], Istas Pratomo Manalu[1]

[1]*Computer Technology, Vocational Faculty, Institut Teknologi Del Toba, 22381, Indonesia*

E-mail: [a)] frengki.simatupang@del.ac.id

Received: May 30, 2024 *Revision: July 22, 2024* *Accepted: July 29, 2024*

Abstract: The Motion Capture System (MCS) is a series of steps involving the capture, processing, and mapping of human, animal, or object movements into virtual characters. This process includes motion recording, data processing with computers, and mapping the results into digital models. This article discusses the use of MCS for realistically reconstructing and simulating behavior, which is valuable in various applications such as entertainment, animation, computer games, sports, and rehabilitation. MCS faces challenges in accurately capturing the diversity and complexity of body movements, requiring advanced hardware and software. Unreal Engine, a development platform known for its ability to produce high-quality graphics and realistic physics systems, is often used to integrate MCS. The MPU9250 sensor, a 9-axis inertial sensor combining an accelerometer, gyroscope, and magnetometer, is used in this project to detect linear motion, rotation, and orientation with high accuracy. This research aims to develop an MCS program based on Unreal Engine using the MPU9250 sensor and the UDP protocol, with the Wemos D1 Mini as the client module to process and transmit sensor data via Wi-Fi. This process involves collecting sensor data, transmitting data via UDP, and processing the data in Unreal Engine to control the movement of objects or characters. Sensor calibration is performed to obtain accurate reference values, and the collected data is used to calculate pitch, roll, and yaw values. This data is then sent to Unreal Engine to display animations corresponding to human movements. Testing shows that the generated movements align with those demonstrated, although there are minor acceptable errors. This research highlights the importance of combining sensor technology and software to produce an accurate and responsive motion control system, providing an interactive and realistic user experience.

Keywords: Motion Capture System, Unreal Engine, MPU-9250 Sensor, Wemos, UDP Protocol.

Abstrak: *Sistem Motion Capture (MCS) adalah serangkaian langkah yang melibatkan penangkapan, pemrosesan, dan pemetaan gerakan manusia, hewan, atau objek ke dalam karakter virtual. Proses ini mencakup rekaman gerakan, pengolahan data dengan komputer, dan pemetaan hasilnya ke dalam model digital. Artikel ini membahas penggunaan MCS untuk merekonstruksi dan mensimulasikan perilaku secara realistis, yang berguna dalam berbagai aplikasi seperti hiburan, animasi, permainan komputer, olahraga, dan rehabilitasi. MCS menghadapi tantangan dalam menangkap keragaman dan kompleksitas gerakan tubuh secara akurat, yang memerlukan perangkat keras dan perangkat lunak canggih. Unreal Engine, platform pengembangan yang dikenal karena kemampuannya dalam menghasilkan grafis berkualitas tinggi dan sistem fisika yang realistis, sering digunakan untuk mengintegrasikan MCS. Sensor MPU9250, sensor inersia 9 sumbu yang menggabungkan akselerometer, giroskop, dan magnetometer, digunakan dalam proyek ini untuk mendeteksi gerakan linear, rotasi, dan orientasi dengan akurasi tinggi. Penelitian ini bertujuan mengembangkan program MCS berbasis Unreal Engine menggunakan sensor MPU9250 dan protokol UDP dengan Wemos D1 Mini sebagai modul client untuk memproses dan mengirimkan data sensor melalui Wi-Fi. Proses ini melibatkan pengumpulan data sensor, pengiriman data melalui UDP, dan pemrosesan data dalam Unreal Engine untuk mengontrol gerakan objek atau karakter. Kalibrasi sensor dilakukan untuk mendapatkan nilai referensi yang akurat, dan data yang dikumpulkan digunakan untuk menghitung nilai pitch, roll, dan yaw. Data ini kemudian dikirimkan ke Unreal Engine untuk menampilkan gerakan animasi yang sesuai dengan gerakan manusia. Pengujian menunjukkan bahwa gerakan yang dihasilkan sesuai dengan yang diperagakan, meskipun terdapat sedikit kesalahan yang masih dapat diterima. Penelitian ini menunjukkan pentingnya kombinasi teknologi sensor dan perangkat lunak untuk menghasilkan sistem kontrol gerak yang akurat dan responsif, serta memberikan pengalaman pengguna yang interaktif dan realistis.*

Kata kunci: *Sistem Motion Capture, Unreal Engine, Sensor MPU-9250, Wemos, UDP Protokol.*

INTRODUCTION

The Motion Capture System (MCS) is generally defined as a series of steps involving the capture, processing, and mapping of human, animal, or object movements into virtual characters. This process includes motion recording, data processing with computers, and mapping the results into digital models. In this article, MCS [1]–[3][4] is used to record human movements with the primary goal of realistically reconstructing and simulating the behavior of humans, animals, or objects. This is highly useful in various applications such as entertainment, animation, computer games, sports, and rehabilitation. The development of MCS faces significant challenges due to the diversity and complexity of body movements. To capture movements accurately, various devices have been developed. For example, laser scanners are used to reconstruct the geometry of human forms, while optical sensor-based motion capture equipment tracks player movements. Based on the type of equipment used, motion capture is categorized into two: image-based and sensor-based motion capture. An effective and accurate MCS requires a combination of advanced hardware and software. There are two main types of tools for MCS: marker-based and markerless tools [5]. Markerless MCS does not require equipment on the actor but uses technologies like camera modules to capture movements without specific markers. Conversely, marker-based MCS uses special equipment or clothing with magnetic sensors or flex sensors to determine the human joint points [4]. Motion control systems (MCS) are critical components in various modern applications, particularly in gaming, simulation, and robotics. MCS is used to detect and control the movement of objects or characters within applications, allowing for more natural and responsive interaction. In games and simulations, MCS is often used to control character or object movements based on user input, such as hand, head, or body movements. In robotics, MCS is used to control the movements of robots based on received sensor data. One of the main challenges in developing MCS is ensuring high accuracy and responsiveness. High accuracy is necessary to ensure that the movements detected by the system correspond to the actual movements, while high responsiveness is needed to ensure that the system can quickly respond to movement inputs without significant delays. This is crucial in real-time applications, such as VR games and simulations, where delays in response can degrade the user experience.

In recent decades, the gaming and simulation industries have experienced rapid development. One of the main factors driving this growth is advances in graphics and data processing technology. Unreal Engine [6], developed by Epic Games, is one of the leading platforms in this industry. First introduced in 1998, Unreal Engine has been used to develop various types of applications, from games, films, and training simulations to virtual reality (VR) and augmented reality (AR) applications. Unreal Engine is a highly popular game development software and is frequently used in the entertainment industry. It has powerful capabilities for producing high-quality graphics and supports various features, including motion capture. Unreal Engine is known for its ability to generate high-quality graphics and realistic physics systems. The latest version, Unreal Engine 5, introduces new features such as Nanite and Lumen, allowing for highly detailed object rendering and more realistic dynamic lighting systems. These capabilities make Unreal Engine the primary choice for many application developers needing high-quality graphics and animations.

The Wemos D1 Mini is a microcontroller board that uses the ESP8266 chip [7][8][9]. This microcontroller is known for its built-in Wi-Fi capability, enabling easy and efficient network connectivity. The Wemos D1 Mini is very popular among electronics enthusiasts and IoT developers due to its compact size, ease of use, and flexibility. To achieve high accuracy and responsiveness in MCS, sensors capable of accurately detecting movements are required. One widely used sensor is the MPU9250, developed by InvenSense (now part of TDK). The MPU-9250 [10] is a 9-axis inertial sensor commonly used in electronics and robotics projects. It combines an accelerometer, gyroscope, and magnetometer in one package [4]. This sensor can detect linear motion, rotation, and orientation with high accuracy. The accelerometer in the MPU9250 [11] can measure acceleration along three axes (x, y, z), allowing for the detection of linear movements. The gyroscope can measure rotational speed around three axes, which is very useful for detecting orientation changes. The magnetometer is used to measure the Earth's magnetic field, which can be used for orientation calibration and drift compensation in the gyroscope. The combination of these three sensors enables the MPU9250 to provide highly accurate and comprehensive motion data. The MPU9250 also has a small size and low power consumption, making it suitable for various portable and wearable applications. Additionally, this sensor supports multiple communication protocols, such as I2C and SPI, making it easy to integrate with microcontrollers and other computer systems.

In real-time applications like MCS, the speed and efficiency of data transmission are crucial. The User Datagram Protocol (UDP) is one of the communication protocols often used in real-time applications [12]. UDP is a connectionless communication protocol, meaning it does not require a fixed connection between the sender and receiver. This makes UDP faster and more efficient compared to the Transmission Control Protocol (TCP), which is more complex and ensures data delivery. One of the main advantages of UDP is its low latency, which is very important in real-time applications where the speed of data transmission directly impacts system responsiveness. Although UDP does not guarantee data packet delivery, its speed and efficiency make it a suitable

choice for applications like MCS, where some data packet loss can be tolerated as long as the system can respond quickly to inputs.

Integrating the MPU9250 sensor with Unreal Engine via the UDP protocol using the Wemos D1 Mini is a critical step in developing an accurate and responsive motion control system. This process involves several steps, from collecting sensor data to processing the data in Unreal Engine to control movements.

This research aims to develop a MCS program based on Unreal Engine that uses the MPU9250 sensor and the UDP protocol. The program is expected to detect and control movements accurately and responsively, providing an interactive and realistic user experience.

METHODS

Rotation Matrix Roll, Pitch, Yaw

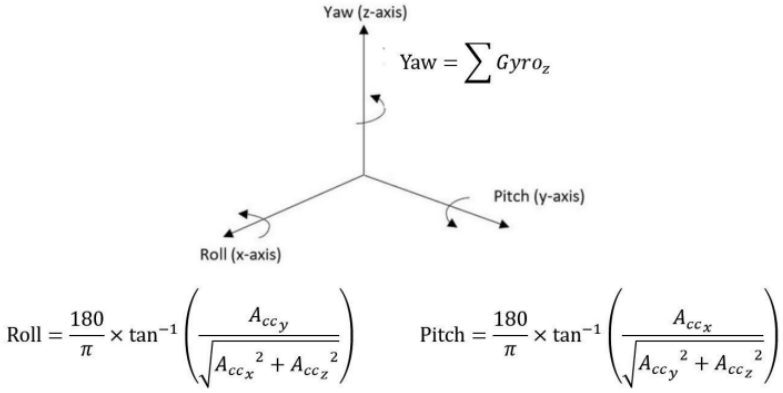

Figure 1. Illustration of Pitch, Roll, and Yaw

The values of Roll, Pitch, and Yaw can be obtained from the directional movements of the MPU-9250 sensor [4]. The roll value is derived from the sensor's movement to the right and left, the pitch value from the sensor's movement forward and backward, and the yaw value is obtained from the sensor's rotation from one side to the other. A rotation matrix is a matrix used to describe rotational transformations in three-dimensional space. The rotation matrix is often used to represent rotation in Euler coordinates, where the rotation is represented by three angles: roll (rotation around the X-axis), pitch (rotation around the Y-axis), and yaw (rotation around the Z-axis). The rotation matrix for roll (α), pitch (β), and yaw (γ) are as follows: Roll (α): Describes the rotation around the X-axis. This matrix depicts a clockwise rotation when viewed from the positive end of the X-axis. Pitch (β): Describes the rotation around the Y-axis. This matrix depicts a clockwise rotation when viewed from the positive end of the Y-axis. Yaw (γ): Describes the rotation around the Z-axis. This matrix depicts a clockwise rotation when viewed from the positive end of the Z-axis.

Workflow System

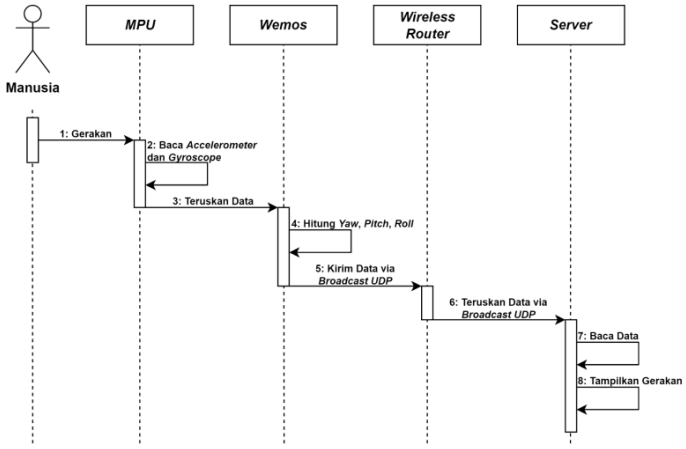

Figure 2. Workflow System

The workflow of the client and server system in this study is as follows: The MPU9250 sensor detects human motion. The accelerometer within the MPU9250 measures linear acceleration values along three axes (x, y, z). Simultaneously, the gyroscope in the MPU9250 measures angular velocity values along the same three axes.

Data from the accelerometer and gyroscope are used to calculate the yaw position (rotation angle around the z-axis), pitch position (rotation angle around the x-axis), and roll position (rotation angle around the y-axis). The Wemos then sends data packets via the Broadcast UDP Protocol through a wireless router to the server. The server receives data consisting of the ID of each sensor device and the yaw, pitch, and roll information. The server then displays the animation results corresponding to the human motion.

Proposed Design

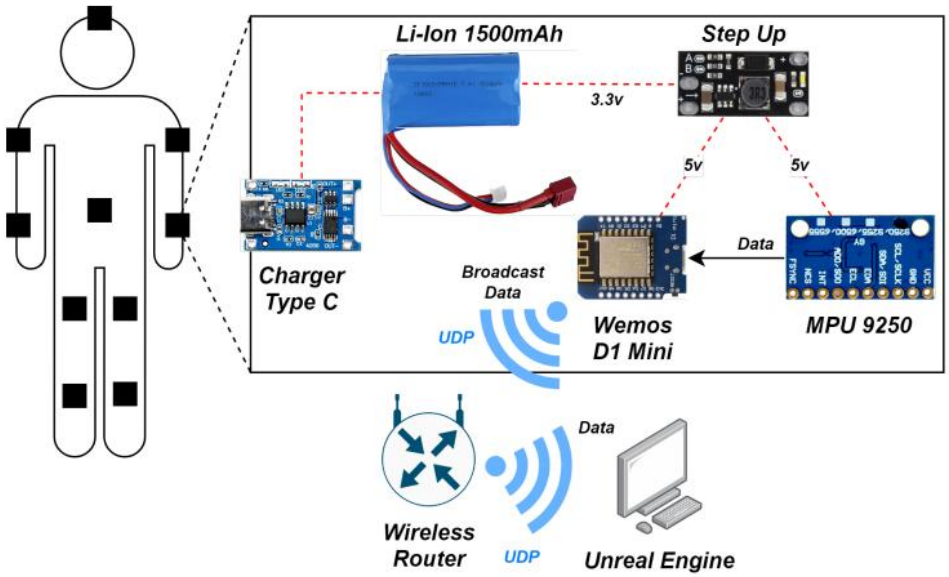

Figure 3. Proposed Design

The Type C 18650 Charger Module is used to facilitate the charging process of a 1500mAh Li-Ion battery. The 1500mAh Li-Ion battery has high energy capacity, relatively light weight, and low self-discharge rate. The battery is used as the power source to supply the Wemos D1 Mini and the MPU9250 sensor. A step-up converter from 3.7 Vdc to 5 Vdc is used to increase the voltage from the battery's 4.2 Vdc to 5 Vdc to supply the input to the Wemos D1 Mini. The Wemos D1 Mini is used as a client module to process sensor data and transmit it via Wi-Fi. The MPU9250 sensor has 9 degrees of freedom (DOF) output data, which includes accelerometer, gyroscope, and magnetometer data. This data is processed to display the yaw, pitch, and roll values. The MPU module is used to generate the yaw, pitch, and roll angle values as input data for Unreal Engine. Unreal Engine is used to process the yaw, pitch, and roll coordinate data sent by the MPU sensor. This software will display animations corresponding to human movements. The integration of the MPU9250, UDP, Unreal Engine, and Wemos D1 Mini consists of the following processes:

a. Sensor Data Collection: Data from the MPU9250 sensor is collected and processed using the Wemos D1 Mini. The Wemos D1 Mini reads data from the sensor via the I2C protocol and transmits it over the network using UDP.
b. Data Transmission via UDP: Data collected by the Wemos D1 Mini is sent via the UDP protocol to a computer or device running Unreal Engine.
c. Data Processing in Unreal Engine: Unreal Engine receives the data via UDP and processes it to control the movement of objects or characters in the application. Unreal Engine provides various tools and APIs to process input data and implement accurate and responsive motion control

RESULT AND DISCUSSION

The Implementation on the Client Side

The device calibration program allows for initiating the calibration of pitch, roll, and yaw by sending messages 'cal_pitch', 'cal_roll', and 'cal_yaw' to the Arduino device. Calibration is performed to obtain reference data or set the necessary values so that the device or sensor can provide accurate results or meet the requirements. Below is the pseudocode for sensor calibration.

```
// Pseudocode to handle calibration commands from packetBuffer
function handleCalibrationCommand(packetBuffer, len, SENSOR_ID):
    // Check if the packetBuffer starts with "cal pitch"
```

```
        if (strncmp(packetBuffer, "cal pitch", 9) == 0):
            if (len == 9):
                print("cal_pitch")
                calibration.pitch = true
            else:
                sensorId = atoi(packetBuffer[10:])
                if (sensorId == SENSOR_ID):
                    print("cal_pitch")
                    calibration.pitch = true
        // Check if the packetBuffer starts with "cal_roll"
        elif (strncmp(packetBuffer, "cal_roll", 8) == 0):
            if (len == 8):
                print("cal_roll")
                calibration.roll = true
            else:
                sensorId = atoi(packetBuffer[9:])
                if (sensorId == SENSOR_ID):
                    print("cal_roll")
                    calibration.roll = true
        // Check if the packetBuffer starts with "cal_yaw"
        elif (strncmp(packetBuffer, "cal_yaw", 7) == 0):
            if (len == 7):
                print("cal_yaw")
                calibration.yaw = true
                yaw = 0
            else:
                sensorId = atoi(packetBuffer[8:])
                if (sensorId == SENSOR_ID):
                    print("cal_yaw")
                    calibration.yaw = true
                    yaw = 0
// Call the function to handle the calibration command
       handleCalibrationCommand(packetBuffer, len, SENSOR_ID)
```

The calculation process for pitch, roll, and yaw is based on accelerometer (AccX, AccY, AccZ) and gyroscope (GyroZ) data. The pitch calculation is computed using trigonometric formulas Eg.1 based on accelerometer data.

$$Pitch = \quad 180 * \text{atan}\left(\frac{AccX}{sqrt(AccY*AccY+AccZ*AccZ)}\right)/M\_PI \qquad (1)$$

The roll calculation can be determined using trigonometric formulas based on accelerometer data, Eg.2.

$$Pitch = \quad 180 * \text{atan}\left(\frac{AccY}{sqrt(AccX*AccX+AccZ*AccZ)}\right)/M\_PI \qquad (2)$$

The yaw calculation uses Eg.3 below.

$$Yaw+= \quad GyroZ * 2 \qquad (3)$$

The atan() function is used to calculate the arctan angle of the ratio of AccX to the square root of the sum of the squares of AccY and AccZ, and to calculate the arctan angle of the ratio of AccY to the square root of the sum of the squares of AccX and AccZ. The results are converted from radians to degrees by multiplying by 180 and dividing by the value of pi (M_PI). Below is the pseudocode for calculating Roll, Pitch, and Yaw.

```
// Pseudocode to calculate pitch, roll, and yaw
// Main function to calculate pitch, roll, and yaw
function calculateOrientation(AccX, AccY, AccZ, GyroZ):
    // Initialize yaw variable if it hasn't been initialized
    if yaw not initialized:
        yaw = 0
    // Calculate pitch
    pitch = 180 * atan(AccX / sqrt(AccY * AccY + AccZ * AccZ)) / M_PI
    // Calculate roll
    roll = 180 * atan(AccY / sqrt(AccX * AccX + AccZ * AccZ)) / M_PI
    // Update yaw based on GyroZ
    yaw += GyroZ * 2
    // Ensure yaw stays within the range -360 to 360 degrees
    if yaw > 360:
```

```
        yaw = yaw - 360
    if yaw < -360:
        yaw = yaw + 360
    // Return pitch, roll, and yaw values
    return pitch, roll, yaw
// Call the function to calculate orientation
pitch, roll, yaw = calculateOrientation(AccX, AccY, AccZ, GyroZ)
```

The calculated results for Roll, Pitch, and Yaw are then sent to the server. The sensor data is transmitted to the server using the SensorData object. The SensorData object, from the SensorData class, is used to store the sensor data that will be sent to the server. The sensor data is input into the SensorData object as follows: SENSOR_ID (representing the identification of the sensor sending the data), yaw value (yaw angle), pitch value (pitch angle), and roll value (roll angle).

The Implementation on the Server Side

The server receives data from the client using the UDP protocol. The process of receiving sensor data involves function blocks derived from the Blueprint Class, each serving a specific purpose: Get Actor of Class Function Block: This function block creates a target or reference for the specified 3D object or animation (illustrated in Figure .4). Start UDPReceiver Function Block: This function block is used to receive data via the UDP protocol. It includes declarations for inputs such as the SSID (Wi-Fi name) and the Port used to receive data from UDP. Is New Data Ready Function Block: This block ensures whether the data is available or not. If no data is incoming, it displays "No Data Ready"; if data is received, it shows the string data from the IMU sensor values. Get Data Function Block: This block receives data conditioned on the output from the Is New Data Ready block. If the Is New Data Ready block returns true, Get Data will activate and display the sensor values, which are then stored in the AnimBP variable. Break UDPData Function Block: This block splits the incoming data into multiple parts: float1, float2, float3, and a unit representing the roll, pitch, yaw values, and a unit as the identity value for each sensor.

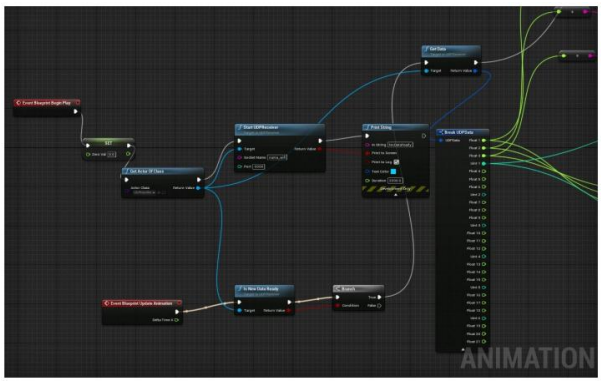

Figure 4. UDP Handler

Parsing Rotator Data to 3D Animation Body Parts: In this section, the values received from the UDPHandler are parsed into rotator values based on the data structure sent from the client. The values float1, float2, and float3 represent the Roll, Pitch, and Yaw, respectively. This allows the movements captured to accurately reflect those performed by the actor (human).

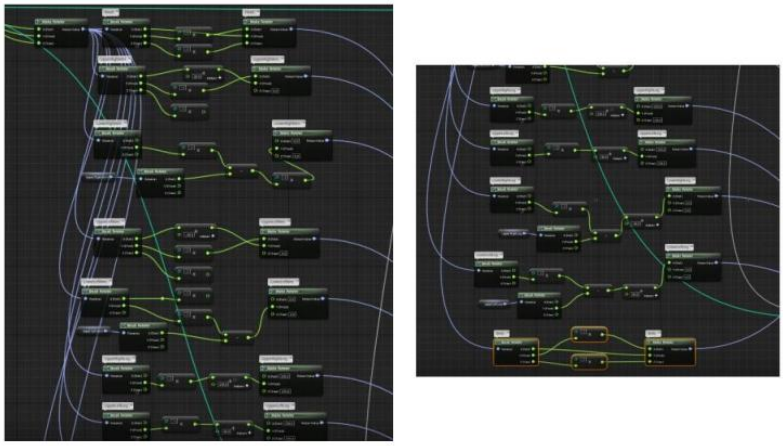

Figure 5. Conversion of Rotator Values

Sending Data to 3D Animation Body Part Variables, each received roll, pitch, and yaw value will not be directly assigned to the body part variables. Instead, they will be directed first using a Switch on Int based on the incoming unit value, ensuring that the sensor values are assigned to the intended body part or point, as shown in Figure 6.

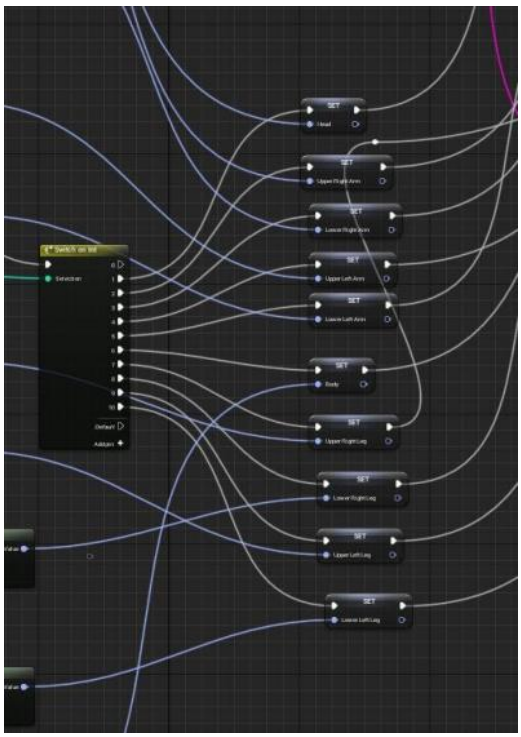

Figure 6. Set Value Rotator to Third PersonAnimBP

The function block of the print string is used to display the declared data as shown in Figure. 7. Here, the displayed values are the roll, pitch, and yaw data that have been successfully transmitted and serve as inputs for the available body part variables.

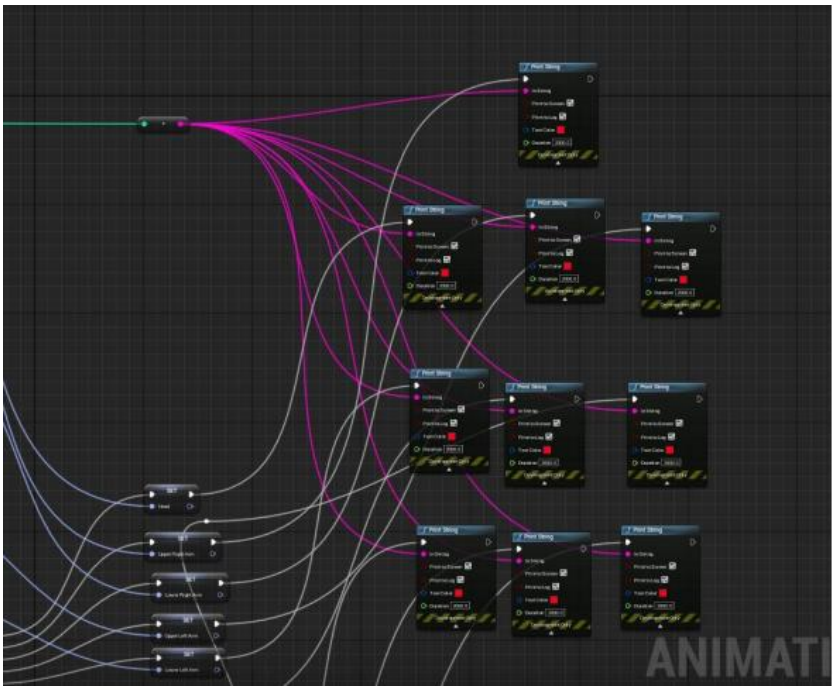

Figure 7. Print Value Rotator Third PersonAnimBP

Figure.8 shows the test for measuring the values of Pitch, Roll, and Yaw using the accelerometer and gyroscope sensors. The test results display a visualization of the calculated values from the accelerometer and gyroscope sensors. The data processing involves converting the initial X, Y, and Z values into Yaw, Pitch, and Roll angles. All the information is then transmitted via the serial protocol using the Wemos D1 Mini.

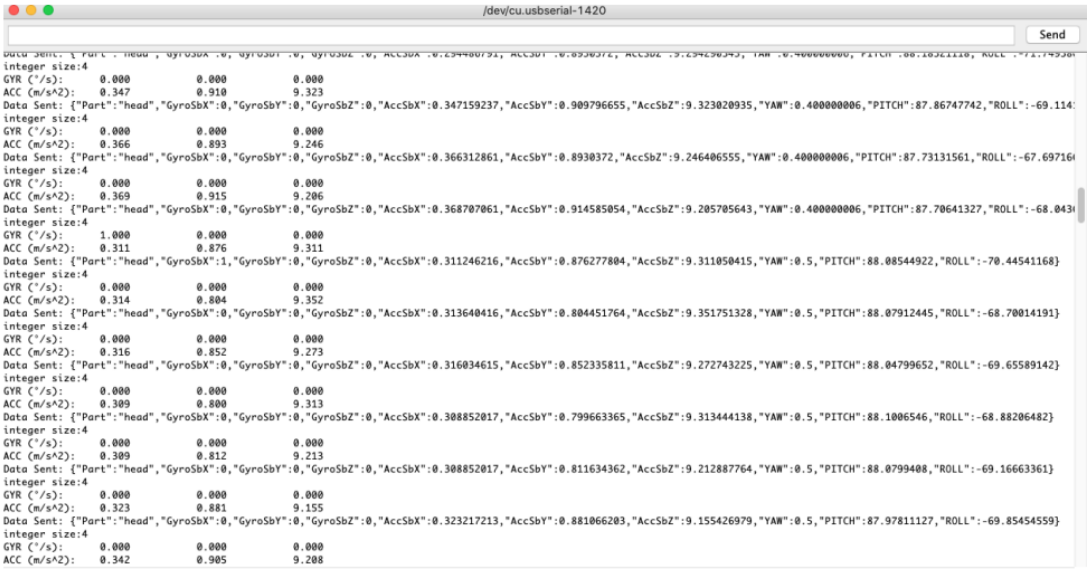

Figure 8. Display of sensor measurement results on the client

Testing can be conducted by having an agent perform several movements to observe the output image or movement generated through Unreal Engine. Based on Figure.8, the movements correspond to those demonstrated by the agent, although there are still some errors that are not significantly different from the expected results.

Figure 9. Testing on Unreal Engine (Server-side)

CONCLUSIONS

This research aims to develop an MCS program based on Unreal Engine utilizing the MPU9250 sensor and UDP protocol. The program is expected to accurately and responsively detect and control movements, providing users with an interactive and realistic experience. The findings of this study are anticipated to contribute to the advancement of MCS technology, making it more sophisticated and efficient, applicable across various fields of application. The research holds several significant implications. Firstly, technically, it is expected to advance MCS technology, integrating the MPU9250 sensor, UDP protocol, and Unreal Engine to produce a system with high accuracy and responsiveness, crucial for real-time applications. Secondly, in terms of applications, the research outcomes can be applied in diverse fields such as game development, simulation, VR, animation, and robotics. Accurate and responsive motion control systems can enhance the quality and realism of these applications, thereby improving user experience. Thirdly, academically, this research will contribute to scholarly literature on the development of motion control systems based on Unreal Engine and the utilization of the MPU9250 sensor. The findings are expected to serve as a reference for other researchers interested in similar technological advancements.

REFERENCES

[1] A. S. Reuter and M. Schindler, "Motion Capture Systems and Their Use in Educational Research: Insights from a Systematic Literature Review," *Educ. Sci.*, vol. 13, no. 2, 2023, doi: 10.3390/educsci13020167.

[2] S. Salisu, N. I. R. Ruhaiyem, T. A. E. Eisa, M. Nasser, F. Saeed, and H. A. Younis, "Motion Capture Technologies for Ergonomics: A Systematic Literature Review," *Diagnostics*, vol. 13, no. 15, pp. 1–16, 2023, doi: 10.3390/diagnostics13152593.

[3] M. Menolotto, D. S. Komaris, S. Tedesco, B. O'flynn, and M. Walsh, "Motion capture technology in industrial applications: A systematic review," *Sensors (Switzerland)*, vol. 20, no. 19, pp. 1–25, 2020, doi: 10.3390/s20195687.

[4] F. Gifari, A. Siswo, R. Ansori, and F. C. Hasibuan, "Pengembangan Motion Capture Lima Jari Dengan Algoritma Complementary Filter Development Of Five Finger Motion Capture With Complementary Filter Algorithm," *Bandung Univ. Telkom*, vol. 8, no. 6, pp. 12039–12047, 2021.

[5] R. Haratian, "Motion Capture Sensing Technologies and Techniques: A Sensor Agnostic Approach to Address Wearability Challenges," *Sens. Imaging*, vol. 23, no. 1, pp. 1–21, 2022, doi: 10.1007/s11220-022-00394-2.

[6] A. Vershinina, "Applications of Game Engine technologies : Unreal Engine 5," no. March, 2023, doi: 10.2139/ssrn.300333022.

[7] R. Ramdan, L. Lasmadi, and P. Setiawan, "Sistem Pengendali On-Off Lampu dan Motor Servo sebagai Penggerak Gerendel Pintu Berbasis Internet Of Things (IoT)," *Avitec*, vol. 4, no. 2, p. 211, 2022, doi: 10.28989/avitec.v4i2.1317.

[8] W. Aditya and S. Subektiningsih, "*Schedule Cat Feeder* Berbasis *Internet of Things* Menggunakan Wemos D1 Mini dan Telegram," *J. Teknol. Inf. dan Ilmu Komput.*, vol. 11, no. 1, pp. 183–190, 2024, doi: 10.25126/jtiik.20241117847.

[9] W. Suryono, A. Setiyo Prabowo, Suhanto, and A. Mu'Ti Sazali, "Monitoring and controlling electricity consumption using Wemos D1 Mini and smartphone," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 909, no. 1, 2020, doi: 10.1088/1757-899X/909/1/012014.

[10] M. A. Nursyeha, R. H. Saputra, H. Aprillia, and F. R. Irwansyah, "Implementasi Sensor Inertial Measurement Unit untuk Sistem Odometri Kendaraan Otonom," *SPECTA J. Technol.*, vol. 7, no. 2, pp. 556–565, 2023, doi: 10.35718/specta.v7i2.784.

[11] B. Darmawan, D. Syauqy, M. Hannats, and H. Ichsan, "Rancang Bangun View controller Menggunakan Sensor Akselerometer Pada Game Bus Simulator Sebagai Sarana User Experience (UX) Berbasis Sistem Embedded," *Pengemb. Teknol. Inf. dan Ilmu Komput.*, vol. 3, no. 2, pp. 1883–1891, 2019.

[12] A. D. Fakhrudin, N. Fahmi, A. Hakim, A. Heri, and S. Budi, "Implementasi Protokol TCP dan UDP pada Sistem Monitoring dan Otomasi Rumah Jamur Berorientasi WSN Implementation of TCP and UDP Protocols on WSN-Oriented Mushroom House Monitoring and Automation System," *Telka*, vol. 9, no. 2, pp. 130–144, 2023.