ESCUELA DE INGENIERÍA

UNIVERSIDAD MODELO

INGENIERÍA BIOMÉDICA

LÓPEZ CUA ANDREA ELIZABETH

8VO SEMESTRE

ROBÓTICA

# REALIDAD VIRTUAL

## INTRODUCCIÓN

En el vasto campo de la ingeniería biomédica, dos tecnologías emergentes han estado revolucionando la forma en que interactuamos con el mundo que nos rodea: la Realidad Aumentada (RA) y la Realidad Virtual (RV). Estas tecnologías ofrecen oportunidades únicas para mejorar la atención médica, la formación de profesionales de la salud y la investigación biomédica. En este contexto, es fundamental comprender cómo la RA y la RV están siendo aplicadas en el ámbito biomédico para potenciar diagnósticos más precisos, tratamientos más efectivos y una comprensión más profunda del cuerpo humano.

En el mundo de la ingeniería biomédica, la integración entre la tecnología y el cuerpo humano ha dado lugar a avances significativos en el campo de la robótica. Uno de los desafíos más apasionantes es la creación de avatares humanoides que puedan interactuar de manera natural con su entorno, imitando los movimientos y acciones de un ser humano real. En este contexto, el presente proyecto se centra en la creación de un modelo virtual de un humanoide AVATAR, dotado de un sensor de aceleración y encoders en las articulaciones, con el objetivo de replicar fielmente los movimientos de un modelo real. Esta tarea no solo representa un desafío técnico, sino que también abre nuevas posibilidades en campos como la teleoperación, la rehabilitación y la asistencia médica.

**OBJETIVO GENERAL**

Desarrollar un modelo virtual de un humanoide AVATAR en el contexto de la ingeniería biomédica, equipado con un sensor de aceleración y encoders en las articulaciones, con el fin de replicar de manera precisa los movimientos de un modelo real y explorar su potencial aplicación en campos como la teleoperación y la rehabilitación.

**OBJETIVOS ESPECÍFICOS**

- → Integrar un sensor de aceleración y encoders en las articulaciones del modelo virtual del humanoide AVATAR, estableciendo una conexión bidireccional que permita la captura y transmisión de datos de movimiento en tiempo real.
- → Desarrollar algoritmos de fusión sensorial que integren los datos del sensor de aceleración y los encoders para estimar la posición y orientación del modelo virtual con alta precisión y bajo tiempo de latencia.
- → Validar el modelo virtual del humanoide AVATAR mediante pruebas de replicación de movimientos en comparación con un modelo real, evaluando la fidelidad y la sincronización de los movimientos en diferentes escenarios y condiciones de operación.

## MARCO TEÓRICO

***Matrices de Rotación y Cinemática de Robots:*** Las matrices de rotación son herramientas fundamentales en la cinemática de robots, permitiendo describir la transformación de coordenadas entre diferentes sistemas de referencia. En el contexto de un brazo robótico, estas matrices son esenciales para calcular con precisión la posición y orientación del extremo del brazo en relación con la base del robot. Este análisis cinemático es crucial para garantizar movimientos controlados y exactos del brazo durante el escaneo tridimensional, asegurando la alineación correcta del end effector con respecto al objeto escaneado.

***Imagenología y Procesamiento de Imágenes:*** La imagenología desempeña un papel vital en la captura de datos tridimensionales. Al emplear cámaras y sensores en el brazo robótico, se capturan imágenes que son sometidas a técnicas avanzadas de procesamiento. La detección de bordes, la segmentación de objetos y la correspondencia de características son procesos cruciales que permiten la reconstrucción precisa de modelos tridimensionales. La interpretación precisa de las imágenes garantiza una representación fiel del entorno escaneado.

***Integración de Software y Hardware:*** La eficiente integración de software y hardware es un componente esencial para el rendimiento óptimo del sistema digitalizador y capturador tridimensional. Implica el desarrollo y la optimización de algoritmos de control, procesamiento de datos y visualización de resultados. La selección y configuración adecuadas de componentes de hardware para el brazo robótico y los sensores utilizados en el sistema son fundamentales para garantizar una operación sin problemas y resultados de

alta calidad. La colaboración armoniosa de software y hardware maximiza la eficiencia y la confiabilidad del sistema en su conjunto..

***V REALM BUILDER:*** Es una herramienta de software que permite a los usuarios crear entornos de realidad virtual de una manera intuitiva y eficiente. Esta plataforma ofrece una variedad de herramientas y características que facilitan la creación de mundos virtuales, incluyendo la creación de terrenos, la colocación de objetos y la configuración de interacciones.

***Realidad Virtual:*** Es una tecnología que permite a los usuarios sumergirse en entornos simulados tridimensionales generados por computadora, interactuando con ellos de manera inmersiva. Abarca los siguientes puntos:

- Definición: La realidad virtual es una simulación de la realidad que se experimenta a través de dispositivos de visualización, como auriculares o pantallas, y se complementa con dispositivos hápticos y de seguimiento de movimiento para una experiencia más inmersiva.
- Interacción: La interacción del usuario con el entorno virtual se realiza a través de interfaces hombre-máquina que permiten acciones como la manipulación de objetos, la navegación y la comunicación con otros usuarios.
- Inmersión y Presencia: La inmersión se refiere al grado en que el usuario se siente sumergido en el entorno virtual, mientras que la presencia se refiere a la sensación de estar físicamente presente en ese entorno, aunque sepa que es una simulación.

- Aplicaciones: La realidad virtual se aplica en una amplia gama de campos, incluyendo la industria del entretenimiento, la medicina, la educación, la arquitectura, el diseño de productos, la capacitación y la simulación.
- Beneficios y Desafíos: Entre los beneficios de la realidad virtual se incluyen la mejora de la experiencia del usuario, la reducción de costos en entrenamiento y simulación, y la posibilidad de experimentar situaciones peligrosas o inaccesibles en la vida real. Sin embargo, existen desafíos como la necesidad de equipos costosos, la posibilidad de mareos o fatiga, y la falta de estándares y directrices claras para el diseño y desarrollo de experiencias de realidad virtual.

***Telerrobótica:*** La telerrobótica es un campo interdisciplinario que combina la robótica y las telecomunicaciones para permitir el control remoto de sistemas robóticos a través de redes de comunicación. Este enfoque permite que los robots realicen tareas en entornos peligrosos o inaccesibles, con la supervisión y el control de operadores humanos ubicados en lugares distantes. La telerrobótica encuentra aplicaciones en una amplia gama de industrias, incluyendo la medicina, la exploración espacial, la industria nuclear, la minería, entre otras.

*Realidad Aumentada (RA) en Ingeniería Biomédica*: La Realidad Aumentada combina elementos virtuales con el entorno físico, creando una experiencia inmersiva que mejora la percepción y comprensión de la realidad. En ingeniería biomédica, la RA se utiliza para superponer información médica relevante, como imágenes de resonancia magnética (RM), tomografías computarizadas (TC) o datos de sensores biométricos, sobre el campo de visión

del usuario. Esta tecnología permite a los profesionales de la salud visualizar de manera precisa y detallada estructuras anatómicas, patologías y dispositivos médicos durante procedimientos quirúrgicos, entrenamiento o planificación de tratamientos.

Además, la RA facilita la interacción con modelos anatómicos virtuales en tiempo real, lo que ayuda a mejorar la comprensión espacial y la destreza quirúrgica de los cirujanos. Asimismo, se está explorando su potencial en la rehabilitación física y cognitiva, donde la superposición de elementos virtuales puede motivar y guiar a los pacientes durante sus ejercicios terapéuticos.

**DESARROLLO**

***Materiales:***

- Arduino UNO
- Software Arduino
- Software MATLAB
- Protoboard
- Jumpers macho – macho
- Potenciómetros 100KOhms
- Muñeca
- IMU

Para la implementación del avatar virtual controlado por un modelo físico se empleó una muñeca barbie para replicar los movimientos requeridos, tales como la rotación, flexión y extensión de los hombros. Además de la Unidad de Medición Inercial (IMU) como sensor principal, se integraron 2 potenciómetros para capturar ángulos con precisión. Estos potenciómetros se montan en la muñeca mediante soportes diseñados a la altura de los hombros, asegurando una alineación precisa con el eje de rotación de las articulaciones.

El proceso de desarrollo de software comenzó con la selección de un avatar en la plataforma VRealm. Posteriormente, se identificaron las articulaciones de interés y se procedió a su movilización utilizando MatLab y Arduino. Los ángulos medidos por los potenciómetros y la

IMU se asignaron para controlar las articulaciones del avatar virtual. Se implementaron medidas para filtrar los datos relevantes de la IMU y garantizar una lectura continua en el monitor serial, teniendo en cuenta que la IMU registra no solo cambios en los ángulos, sino también otros datos.

**RESULTADOS**

***Código Arduino***

```
#include "REG.h"

#include "wit_c_sdk.h"


/*

Test on MEGA 2560. use WT901CTTL sensor


WT901CTTL     MEGA 2560

VCC <--->  5V/3.3V

TX  <--->  19(TX1)

RX  <--->  18(RX1)

GND <--->  GND

*/

#define ACC_UPDATE    0x01

#define GYRO_UPDATE   0x02
```

```
#define ANGLE_UPDATE  0x04

#define MAG_UPDATE    0x08

#define READ_UPDATE   0x80


static volatile char s_cDataUpdate = 0, s_cCmd = 0xff;

static bool startDataCollection = false; // Variable para indicar si se debe comenzar a
tomar los datos


static void CmdProcess(void);

static void AutoScanSensor(void);

static void SensorUartSend(uint8_t *p_data, uint32_t uiSize);

static void SensorDataUpdata(uint32_t uiReg, uint32_t uiRegNum);

static void Delayms(uint16_t ucMs);


const uint32_t c_uiBaud[8] = {0,4800, 9600, 19200, 38400, 57600, 115200, 230400};
```

```
void setup() {

// put your setup code here, to run once:

Serial.begin(115200);

WitInit(WIT_PROTOCOL_NORMAL, 0x50);

WitSerialWriteRegister(SensorUartSend);

WitRegisterCallBack(SensorDataUpdata);

WitDelayMsRegister(Delayms);

AutoScanSensor();

}



int i;

float fAcc[3], fGyro[3], fAngle[3];



void loop() {

while (Serial1.available()) {

WitSerialDataIn(Serial1.read());
```

```
}


// Espera la recepción de la letra 's' desde el puerto serial
if (Serial.available() > 0) {
char receivedChar = Serial.read();
if (receivedChar == 's') {
startDataCollection = true; // Marca que se debe comenzar a tomar los datos
}
}


// Si se debe comenzar a tomar los datos, realiza la lectura del sensor
if (startDataCollection) {
CmdProcess();
if (s_cDataUpdate) {
for (i = 0; i < 3; i++) {
fAcc[i] = sReg[AX + i] / 32768.0f * 16.0f;
```

```
fGyro[i] = sReg[GX + i] / 32768.0f * 2000.0f;

fAngle[i] = sReg[Roll + i] / 32768.0f * 180.0f;

}



if (s_cDataUpdate & ANGLE_UPDATE) {

Serial.print(fAngle[0], 3);

Serial.print(',');

Serial.print(fAngle[1], 3);

Serial.print(',');

Serial.print(fAngle[2], 3);

Serial.print("\r\n");

s_cDataUpdate &= ~ANGLE_UPDATE;



}



s_cDataUpdate = 0;
```

```
    }

  }

}


void CopeCmdData(unsigned char ucData) {

static unsigned char s_ucData[50], s_ucRxCnt = 0;


s_ucData[s_ucRxCnt++] = ucData;

if (s_ucRxCnt < 3) return; // Menos de tres datos devueltos

if (s_ucRxCnt >= 50) s_ucRxCnt = 0;

if (s_ucRxCnt >= 3) {

if ((s_ucData[1] == '\r') && (s_ucData[2] == '\n')) {

s_cCmd = s_ucData[0];

memset(s_ucData, 0, 50);

s_ucRxCnt = 0;

} else {
```

```
s_ucData[0] = s_ucData[1];

s_ucData[1] = s_ucData[2];

s_ucRxCnt = 2;

}

}

}


static void ShowHelp(void) {

}


static void CmdProcess(void) {

switch (s_cCmd) {

case 'a':

if (WitStartAccCali() != WIT_HAL_OK) Serial.print("\r\nSet AccCali Error\r\n");

break;

case 'm':
```

```
if (WitStartMagCali() != WIT_HAL_OK) Serial.print("\r\nSet MagCali Error\r\n");

break;

case 'e':

if (WitStopMagCali() != WIT_HAL_OK) Serial.print("\r\nSet MagCali Error\r\n");

break;

case 'u':

if (WitSetBandwidth(BANDWIDTH_5HZ) != WIT_HAL_OK) Serial.print("\r\nSet
Bandwidth Error\r\n");

break;

case 'U':

if (WitSetBandwidth(BANDWIDTH_256HZ) != WIT_HAL_OK) Serial.print("\r\nSet
Bandwidth Error\r\n");

break;

case 'B':

if (WitSetUartBaud(WIT_BAUD_115200) != WIT_HAL_OK)

Serial.print("\r\nSet Baud Error\r\n");
```

```
else {
Serial1.begin(c_uiBaud[WIT_BAUD_115200]);
Serial.print(" 115200 Baud rate modified successfully\r\n");
}
break;
case 'b':
if (WitSetUartBaud(WIT_BAUD_9600) != WIT_HAL_OK)
Serial.print("\r\nSet Baud Error\r\n");
else {
Serial1.begin(c_uiBaud[WIT_BAUD_9600]);
Serial.print(" 9600 Baud rate modified successfully\r\n");
}
break;
case 'r':
if (WitSetOutputRate(RRATE_1HZ) != WIT_HAL_OK)
Serial.print("\r\nSet Baud Error\r\n");
```

```
else Serial.print("\r\nSet Baud Success\r\n");

break;

case 'R':

if (WitSetOutputRate(RRATE_10HZ) != WIT_HAL_OK) Serial.print("\r\nSet Baud
Error\r\n");

else Serial.print("\r\nSet Baud Success\r\n");

break;

case 'C':

if (WitSetContent(RSW_ACC | RSW_GYRO | RSW_ANGLE | RSW_MAG) !=
WIT_HAL_OK)

Serial.print("\r\nSet RSW Error\r\n");

break;

case 'c':

if (WitSetContent(RSW_ACC) != WIT_HAL_OK) Serial.print("\r\nSet RSW Error\r\n");

break;

case 'h':
```

```
ShowHelp();
break;
default:
break;
}
s_cCmd = 0xff;
}

static void SensorUartSend(uint8_t *p_data, uint32_t uiSize) {
Serial1.write(p_data, uiSize);
Serial1.flush();
}

static void Delayms(uint16_t ucMs) {
delay(ucMs);
}
```

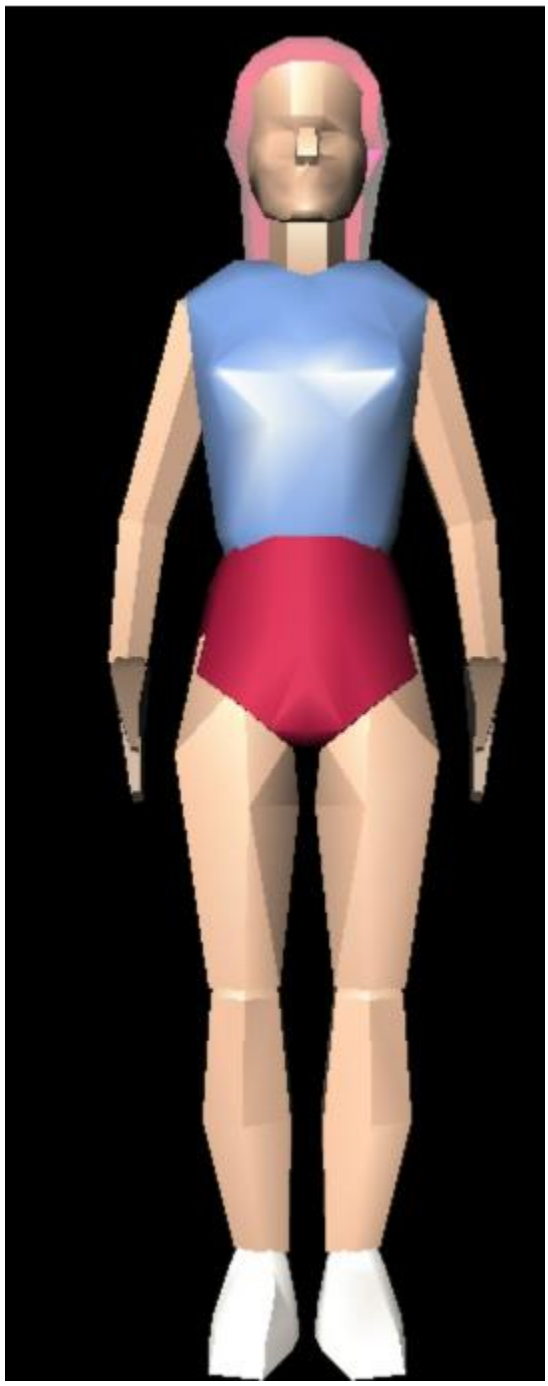
```
static void SensorDataUpdata(uint32_t uiReg, uint32_t uiRegNum) {

int i;

for (i = 0; i < uiRegNum; i++) {

switch (uiReg) {

case AZ:

s_cDataUpdate |= ACC_UPDATE;

break;

case GZ:

s_cDataUpdate |= GYRO_UPDATE;

break;

case HZ:

s_cDataUpdate |= MAG_UPDATE;

break;

case Yaw:

s_cDataUpdate |= ANGLE_UPDATE;
```

```
                    break;

                default:

            s_cDataUpdate |= READ_UPDATE;

                    break;

                }

            uiReg++;

            }

        }


    static void AutoScanSensor(void) {

        int i, iRetry;


    for (i = 0; i < sizeof(c_uiBaud) / sizeof(c_uiBaud[0]); i++) {

        Serial1.begin(c_uiBaud[i]);

        Serial1.flush();

        iRetry = 2;
```

```
s_cDataUpdate = 0;

do {

WitReadReg(AX, 3);

delay(200);

while (Serial1.available()) {

WitSerialDataIn(Serial1.read());

}

if (s_cDataUpdate != 0) {

//Serial.print(c_uiBaud[i]);

//Serial.print(" baud find sensor\r\n\r\n");

ShowHelp();

return;

}

iRetry--;

} while (iRetry);

}
```

```
Serial.print("can not find sensor\r\n");

Serial.print("please check your connection\r\n");

}
```

***Escaneo***



## CONCLUSIÓN

En conclusión, el desarrollo de un modelo virtual de un humanoide AVATAR con sensor de aceleración y encoders representa un avance significativo en el campo de la robótica biomédica. Este proyecto no solo tiene aplicaciones prácticas en áreas como la teleoperación y la rehabilitación, sino que también contribuye al avance del conocimiento en biomecánica y control robótico. Sin embargo, es importante destacar que aún quedan desafíos por superar, como la mejora de la precisión y la robustez del sistema, así como la integración de interfaces de usuario intuitivas para facilitar su uso en entornos clínicos y de investigación. En última instancia, este proyecto sienta las bases para futuras investigaciones en la interacción entre humanos y robots, promoviendo el desarrollo de tecnologías que mejoren la calidad de vida y la salud de las personas.

**AGRADECIMIENTOS**

Quiero externar el agradecimiento a mi equipo, los terminators, por su contribución en esta práctica. Primeramente quiero agradecer a Meliton Esquiliano, el cual se encargó de apoyarnos con el diseño en Fusion 3D de los brazos robóticos, así como con el armado de los mismos, así como la simulación en realidad virtual. Por otra parte, quiero agradecer a Fernanda Cetina, quien nos facilitó las fuentes y la información para la realización de este reporte. En la figura 8 podemos observar la foto del equipo

**BIBLIOGRAFÍA**


- S. Sánchez-López, J. M. de Agar, E. Cervera, and J. C. Torres, "Design of a 3D Scanner Based on a Robot Manipulator with Touch Sensor," Procedia Engineering, vol. 63, pp. 497-504, 2013.
- N. Correll, J. S. Clune, J.-B. Mouret, and H. Lipson, "Robot Self-Modeling: Mechanisms and Control," Autonomous Robots, vol. 30, no. 4, pp. 407-428, 2011.
- Crespo, M. M. Silva, and R. Carelli, "3D Scanning for Cultural Heritage: Evaluating the Quality of Scans," Journal of Cultural Heritage, vol. 13, no. 2, pp. 125-129, 2012.
- J. Y. Lee, S. Y. Seol, and C. W. Park, "Sensing Technologies for Robot Perception and Interaction," Sensors, vol. 19, no. 5, 2019.