Head-motion controlled telepresence robot

Team Members

Aseel Arafeh

Ghaid Ewawi

Supervisors

Dr. Alaa Halawani

Dr. Zein Salah

Acknowledgment

In the name of "Allah", the most beneficent and merciful who gave us strength, knowledge and helped us to get through this project.

Working on this project has been an opportunity for gaining lots of experience. For that, we want to thank all the people who helped us through.

We would like to express our sincere appreciation to our graduation project supervisors Dr. Alaa Halawani and Dr. Zein Salah for their guidance, continuous encouragement, and support throughout the semester.

Furthermore, we have to show our gratitude to Eng. Wael Takrouri and Dr. Ayman Wazwaz for being there to answer our questions and provide us with valuable suggestions and tips which helped us in choosing some system components.

Moreover, it is our duty to thank our families for their generous encouragement and continuous support throughout our life. For our friends, we are truly grateful for all your support throughout the whole education stage.

At last, we would like to thank all the people who helped, supported, and encouraged us to successfully finish the graduation project.

Abstract

At the current age of corona, it became harder than ever to attend physically at a distant location. Flying between countries is closed most of the time, but still, people have their connections and events that they want to attend. Current media of communication only allow the participant to see and hear what is on the other side, with minimal control interaction methods with the distant location. In this project, we hope to add a level of involvement for the participants in their desired distant location. We would like to allow them to look around the room freely, perform some everyday gestures such as nodding, moving their head left and right, and other natural movements they do while communicating in their daily lives.

Our work was to create a system that contains two main parts each at different location. The two locations communicate with each other via the internet. At the first location, there is a user who would like to view and interact with the second location. This user is equipped with a headset mounted at his head, the headset contains sensors for measuring the orientation of his head. At the second location, there is a stand that contains three different motors controlled with a NodeMCU microcontroller. The purpose of this stand is to perform a realistic movement that mimics how the user's head moves. Each of the three motors is responsible for the movement around one of the three different axes.

Table of Contents

List of Tables

List of Figures

Chapter 1 : Introduction

1.1 Overview

Telepresence refers to a set of technologies that allow people to feel as if they were “there” even though they are in another location, via telerobotics. In lesser terms, telepresence must allow for two or more users to communicate as they would if they were face-to-face, representing posture, body language, and gestures – giving the feeling as if they were engaging in an in-person conference or meeting.

1.2 Motivation and Importance

The development of telepresence systems enhances the productivity of distant meetings. It allows better interaction, improved communication between the members, and reduced travel fees. They also maintain the freedom of the user and his ability to navigate the remote area with ease. Telepresence systems also help when a person has restrictions or conditions that intercept his ability to attend physically, such as a broken leg.

1.3 Problem Statement

The majority of the available telepresence devices allow limited movement. The user usually controls the movement of the distant robot manually using a remote control. This method of control results in limited movement. The robot also might crash into walls or bump into objects because they are not in the field of view of the user.

One other limitation of the current devices is that they do not improve the experience of the user. All a user can do is to control the distant robot as if he was controlling a toy car and that does not give him the feeling of involvement in the distant location.

The motion of the robots is usually slow since it depends on the remote-controlled movement and a fixed body-to-head which does not give the user the freedom to make an immediate turn or change of view.

1.4 Project Description

Our work is a system with two main sides. On the first side, the user starts a video call using his smart phone, while wearing a headset that contains a sensor which detects the orientations of his head. A system on the other side is controlled accordingly. This way, the user can see what exists in the other room based only on the orientation of his head, and that gives him the freedom to interact with a distant environment without having to physically attend.

On the second side, the telepresence device's main component is a stand that holds the smart-phone with the ongoing video call. The stand's posture has three degrees of freedom "3DOF" and has a natural-like movement that adapts to the head movement detected by the sensor on the user's head. The camera of the smart-phone captures the surrounding room and transmits it to the user.

Our solution enhances the experience of both parties. The user can control the angle he wants to see by changing the orientation of his head. The second party can see the video of the distant user with the right angle he is looking at which gives them the feeling of having him amongst them as a physical entity.

Figure 1.1 shows the components of our proposed solution.

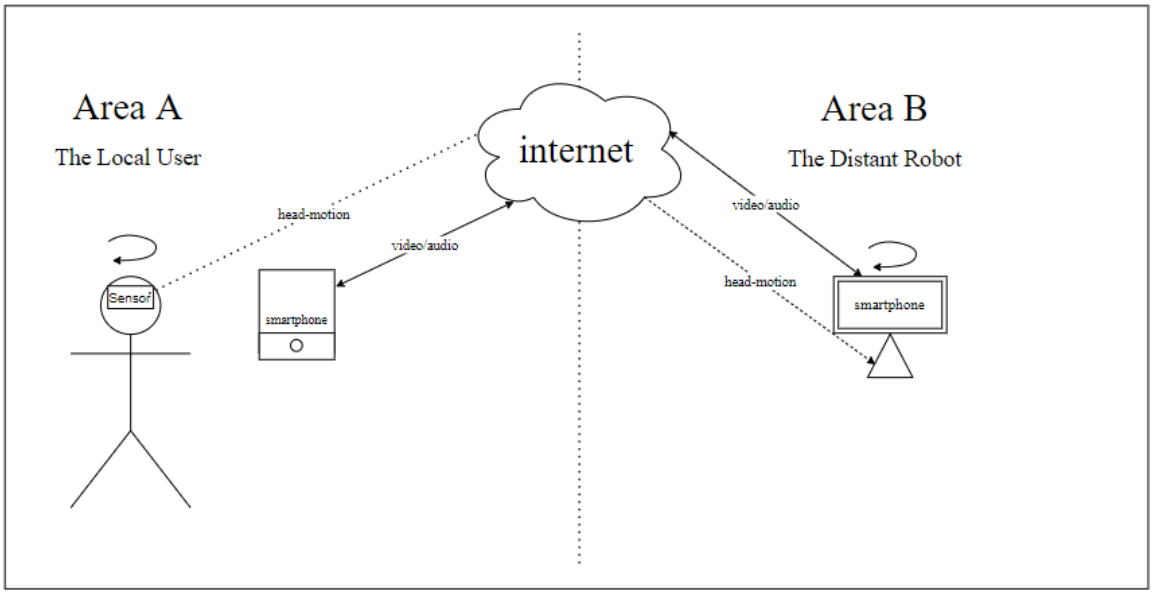

Figure 1.1: System Context Diagram

1.5 Objectives

- To detect the orientations of the head of the user through a sensor.
- To transmit the sensed orientations by the sensor to the internet through the microcontroller.
- To provide a model in the distant location that mimics/ reflects the received head orientation to the smart-phone with 3 degrees of freedom.

1.6 Requirements

- User can control a distant robot by his head motion
- User can receive video and audio from a distant location
- User can transmit his video and audio and display it in a distant location

1.7 Report Outline

This report is organized as follows: Chapter 2 introduces some literature review including available telepresence devices and related projects. It also goes briefly over the theoretical background of the project, hardware, and software components. Chapter 3 discusses the conceptual design of the system, block diagrams, pseudo-code, and detailed hardware connections. Chapter 4 discusses the hardware and software implementations of the project. Chapter 5 shows the testing which is made until we reach the final system design. Finally, Chapter 6 presents a summary and the future work.

Chapter 2 : Background

2.1 Overview

This chapter briefly describes the theoretical background of the project. Short description of the hardware and software parts which are used in the system is also introduced.

2.2 Theoretical Background

Telepresence

Telepresence refers to technologies that allow a user to appear to be present, feel like he is present, or have some effect in a space the person does not physically inhabit. Telepresence can include video teleconferencing tools, where video and audio stream is conveyed to a remote location, as well as more involved robotics installations that can actually help a user to accomplish tasks from a remote location [1].

2.3 Literature Review

Below, we will introduce some of the previously proposed telepresence devices and their features.

A. beam

Description:

beam, shown in figure 2.1, is a two-wheeled videoconferencing robot. It gives the user the ability to freely roam around faraway locations. [2].

Features:

It aims for uses such as monitoring and navigating the distant location without taking into consideration the interaction between the robot and other humans.

Limitations:

The screen that shows the video of the participant is fixed.

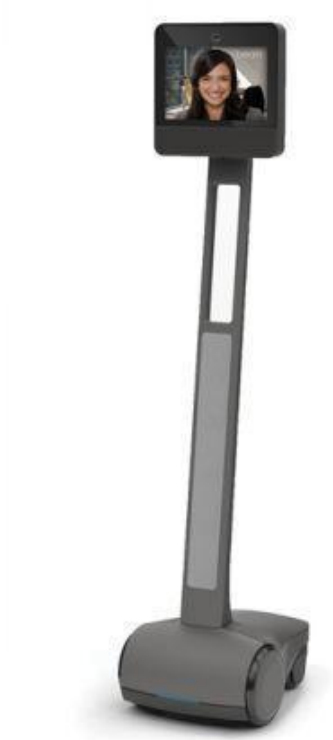

Figure 2.1: beam telepresence robot, [2]

B. VGo

Description:

VGo, shown in figure 2.2, was released after beam. One main enhancement is that its camera is not fixed as the one in beam but has one DOF. This allows for extra flexibility and control over what the participant sees [3].

Features:

This device has an advantage over "beam" since its camera has one DOF.

Limitations:

It is widely discussed how missing non-verbal gestures such as head orientation are compromising interpersonal communication, i.e., controlling the device manually distracts the user from focusing on communication with other participants next to the robot [4].

Figure 2.2: VGo Telepresence Robot, [3]

Better products for telepresence have been investigated by several researchers. A main problem is that the distant participant is busy controlling different aspects of the robot which deprives him of paying attention to the meeting and participants. Our aim is to reduce the effort in controlling the device so that the participant focuses on the social aspect of attending remotely. In table 1, brief comparison between the mentioned projects and ours is shown.

Table 1: Comparison of different devices

Comparison	Head DOF	Head control
beam	0DOF “Fixed”	No control
VGo	1DOF “tilt”	Manually tilt the head
This project	3DOF “pan, tilt and roll”	Automated “reflecting real head-motion”

2.4 Components

This section contains a short description of the hardware and software components used in our system. It also contains a brief justification for certain choices.

2.4.1 Hardware

NodeMCU ESP8266

A microcontroller is an integrated circuit that is housed within each component that it needs to perform the necessary operations and that can perform a particular task routinely without requiring another boom. It contains a microprocessor, memory units and input-output interfaces, analog-to-digital conversion (ADC), pulse width modulation (PWM) and various control and communication modules [5]. The microcontroller we will use in our project is NodeMCU ESP8266.

The NodeMCU (Node MicroController Unit), shown in figure 2.3, is an open-source software and hardware development environment built around an inexpensive System-on-a-Chip (SoC) called the ESP8266. It contains CPU, RAM, networking (WiFi), and even a modern operating system and SDK [6].

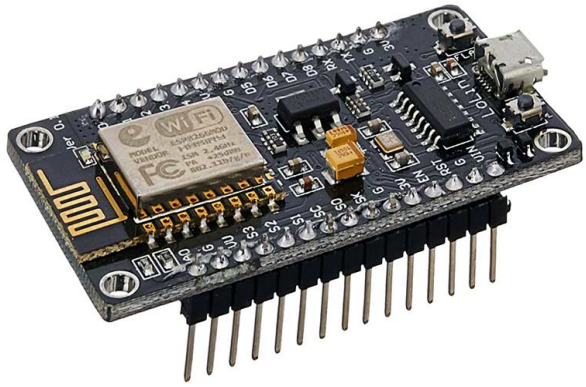

Figure 2.3: NodeMCU ESP8266, [7]

NodeMCU ESP8266 was our microcontroller of choice because it contains a self-contained wifi connection module. Having a NodeMCU with its built-in module is better on the economic scale than an Arduino or Raspberry Pi with wifi module add-on. At the processing scale, the 80 MHz is an advantage over the 16 MHz provided by the Arduino and this processing power is sufficient for our work; the extra processing power introduced by a Raspberry Pi is not needed [8][9].

Sensing head orientation:

This includes raw sensors; an accelerometer, gyroscope and magnetometer attached into a wifi-based microcontroller.

We did our research about the ICs that included the sensors we need, we finally picked MPU9250 to be the sensor we use in our system [10]. Table 2 shows the difference between sensor options.

Table 2: List of sensor options

	MPU6050	MPU9250
Sensor contained in	accelerometer and a gyroscope	accelerometer, gyroscope, and a magnetometer
Interface	I2C	I2C
Yaw drifting	Yes	No
Availability	Yes	Yes

Our initial pick after looking for usages and examples was MPU6050. We found many applications where the readings of a gyroscope and accelerometer were used together in calculating the angles around the three different axes. The values for pitch and roll were stable and correct, but the values of the yaw kept drifting. We tried many techniques for correcting the values but eventually found out that those two sensors were not enough. A magnetometer was also needed for getting accurate yaw measurements.

Bringing in a magnetometer could be done by two different methods. The first was to buy a separate magnetometer and handle the coordination between its readings with what we already got from MPU6050. While the second method was buying MPU9250 which is a chip that contains all three sensors and has an already built library that handles the coordination. Since the implementation of the yaw value correction was already implemented in the MPU9250 library, we decided to start using it instead of dealing with the tedious work of buying a separate module.

- **MPU9250**

MPU-9250, shown in figure 2.4, is a 9-axis MotionTracking device that combines a 3-axis gyroscope, 3-axis accelerometer, 3-axis magnetometer and a Digital Motion Processor (DMP) all in a small 3x3x1mm package [11].

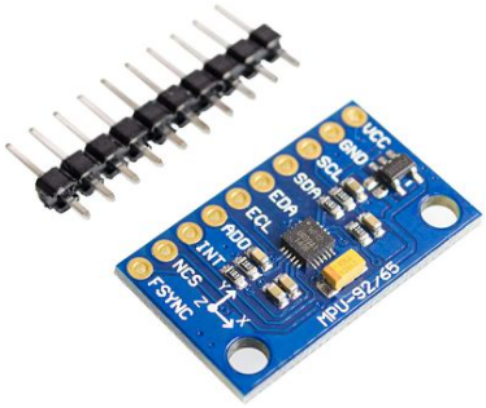

Figure 2.4: MPU9250, [10]

- **Wemos D1 R1**

Wemos D1 R1, shown in figure 2.5, is actually a Wi-Fi development board based on ESP8266-12E, which looks like an Arduino Uno board but the core part is the ESP-12E chip. Surprisingly boards purchased almost at the same time had different looks! In some boards, WEMOS D1 is printed, and in some other boards, only D1 Wi-Fi is printed [12].

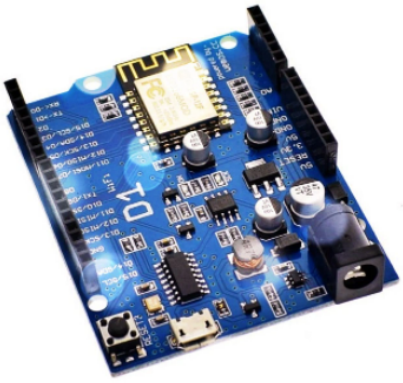

Figure 2.5: WEMOS D1 R1, [12]

MG996r servomotor

A servomotor, shown in figure 2.6, is a linear actuator or rotary actuator that allows for precise control of linear or angular position, acceleration, and velocity. It consists of a motor coupled to a sensor for position feedback. It also requires a relatively sophisticated controller, often a dedicated module designed specifically for use with servomotors [13].

Figure 2.6: MG996r servomotor, [14]

Stepper motor

Stepper motors, shown in figure 2.7, are DC motors that move in discrete steps. They have multiple coils that are organized in groups called "phases". By energizing each phase in sequence, the motor will rotate, one step at a time [15].

Figure 2.7: Stepper motor, [16]

Between servo and stepper motors, we decided to use servomotors since their movement is smoother than stepper motors, and this serves our purpose of reflecting natural-like movements.

5V 2A Power Adapter

A power adapter, shown in figure 2.8, has the ability to convert (100-240V/ 0.35A) AC to (5V/ 2000mA) DC. We use it to power our NodeMcu (which requires 3.3V operating voltage) and 3 servo motors (each one requires 5V operating voltage).

Figure 2.8: 5V 2A Power Adapter, [18]

9V Battery

We use a 9V battery, shown in figure 2.9, to power our Wemos D1 R1.

Figure 2.9: 9V Battery, [19]

Volt Battery Snap

Volt battery snap, shown in figure 2.10, is designed to snap onto the leads on the terminal end of any standard 9V battery [17].

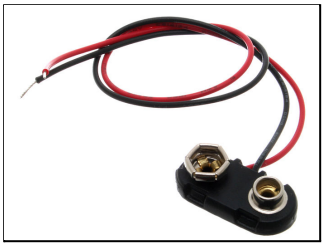

Figure 2.10: Volt Battery Snap, [17]

2.4.2 Software

Arduino IDE

Arduino IDE is an environment that could be used for writing the code and uploading it into NodeMcu.

PubNub

PubNub is a global Data Stream Network and real-time Communication Platform that provides real-time publish/subscribe messaging. [20]

We chose this platforme due to many reasons:

- It is extremely fast. Routing a message through the PubNub network takes as little as 4-9 milliseconds, often resulting in an end-to-end transmission of 30-40ms for clients on a broadband connection [21].
- It supports a library for Arduino IDE, and it is used in IoT applications.
- It provides (1 million Transactions) for free [22].

The connection with a PubNub channel contains two parts; a publisher and a subscriber. The publisher is the party responsible for uploading data into the channel. In the second part, a subscriber acts as a listener for new events. Each time data gets published on the channel, the subscriber is notified and provided with the data.

Video Communication Service/Application

Both video and audio of the participant will be transmitted and received using an online medium such as google meet.

Chapter 3 : Design

3.1 Overview

This chapter discusses the overall design of the system and the way its components are integrated together, showing the block diagram and schematic diagram for the design, in addition to some details about the algorithms we are going to use.

3.2 Detailed Design

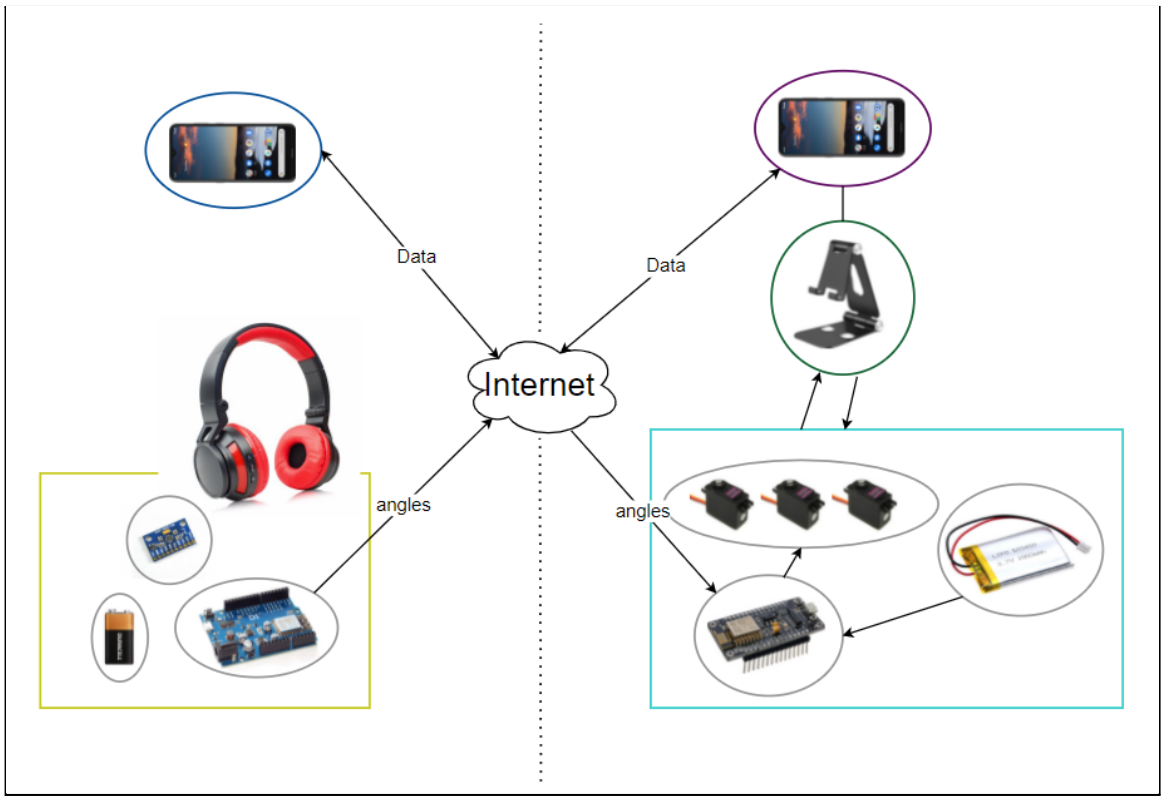

Figure 3.1: System Design

The system design shown in Figure 3.1 illustrates a general overview of the system's main components and the connections between them. In the local area, the user wears a customized headset, and it contains the needed sensors for detecting the orientation and motion of his head. The measurements get transmitted via Wifi into an online server. The participant's smart-phone has an ongoing video call with another smart-phone in a distant location. The measurements are sent via the internet and then received by the NodeMCU, which then controls the rotation of three different servomotors to reflect the local participant's head motion.

3.3 Block Diagrams

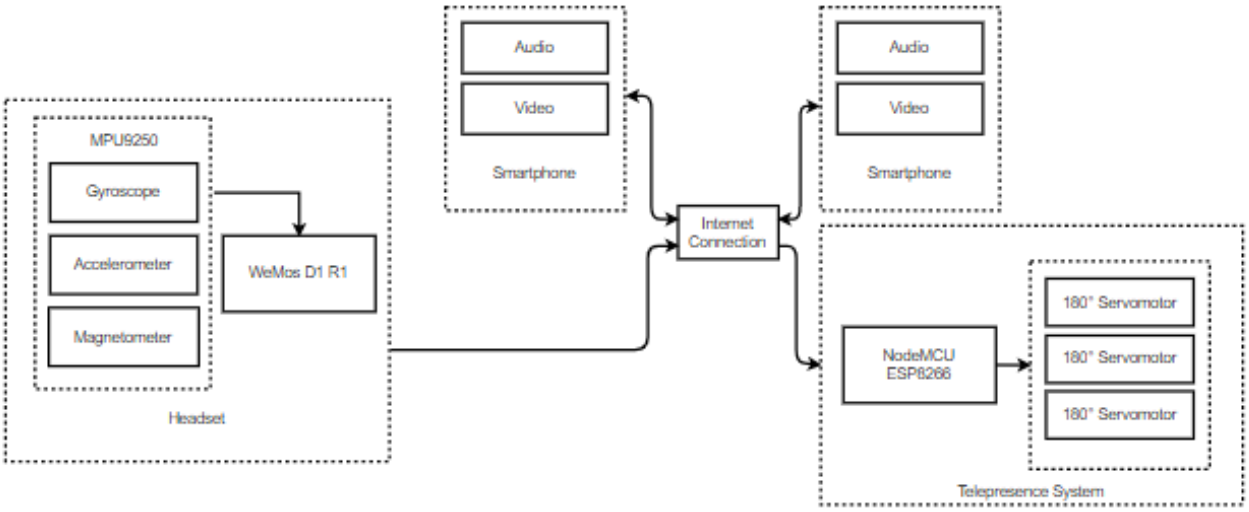

Figure 3.2: System Block Diagram

The system diagram in figure 3.2 shows how the measurements get transmitted from the headset into the Wemos in one direction. Which in turn uploads the measurements via wifi into the PubNub's channel so it can be reachable from the NodeMCU. At the distant location, the Telepresence System is represented by the NodeMCU that controls three servomotors. Those three servomotors are responsible for 3 DOF movements as follows: Each servomotor is responsible for the movement along one of the 3-axis of movement. While the system is in work, i.e., the stand moves realistically mimicking the measurements received from the headset, The two smartphones are connected by two-way video/audio call using a bidirectional channel which allows both sides to see and hear the other.

The uses of the system are illustrated in the following points:

1. The user puts on the headset and establishes a connection between the headset and PubNub's channel.
2. The sensor reads head-motion and transmits the measurements into the internet.
3. The NodeMCU is notified and the measurements are sent into it.
4. While the connection is maintained, the NodeMCU controls the three servomotors according to the measurements received.

5. The servomotors rotate resulting in the stand moving realistically.

While the mentioned system is connected, a video call between the user's smartphone and the smartphone on the stand that is being controlled by the servomotors is established.

3.4 Pseudo-Code and Algorithms

Pseudo-Code

The algorithm shown below runs at the wemos. It contains a loop that reads data from the sensor and publishes it into PubNub's channel as a JSON object.

```
// Publisher
// Setup Part
1  :  INITIALIZE network connection
2  :  CONNECT with network
3  :  INITIALIZE PubNub connection
4  :  INITIALIZE MPU9250 connection
5  :  WAIT until the readings of the angles stabilize
6  :  SET the current angles as the reference points.
        ReferenceX, ReferenceY,   ReferenceZ
// Loop Part
9  :  IF mpu reads a new data
10 :      PROCESS readedYawAngle, readedPitchAngle, readedRollAngle
11 :      IF readedYawAngle belongs to [-90, 90]
12 :          yawAngle = readedYawAngle
13 :      IF readedPitchAngle belongs to [-70, 90]
14 :          pitchAngle = readedPitchAngle
15 :      IF readedRollAngle belongs to [-45, 45]
16 :          rollAngle = readedRollAngle
17 :     ENCODE data to JSON object
18 :     SEND data into PubNub Channel
19 :  END IF
```

The algorithm shown below is what runs at the NodeMCU. The algorithm contains a loop that keeps receiving the different x,y and z measurements from PubNub channel. After that, it controls the orientation of the corresponding servomotor, i.e., the x measurement is reflected by one servomotor that is responsible for yaw, the y measurement is reflected by the pitch servomotor and the z measurements by the roll servomotor.

```
// Subscriber
// Setup Part
1  :  INITIALIZE network connection
2  :  CONNECT with network
3  :  INITIALIZE PubNub connection
4  :  INITIALIZE all servos' pins as output
5  :  INITIALIZE servo1 as yaw
6  :  INITIALIZE servo2 as pitch
7  :  INITIALIZE servo3 as roll
// Loop Part
8  :  IF connection failed
9  :     continue
10 :  ELSE
11 :     RECEIVE JSON Object
12 :     Decode JSON to get the angles
13 :     SET angle of yaw at x degrees
14 :     SET angle of pitch at y degrees
15 :     SET angle of roll at z degrees
16 :  END IF
```

Schematic Diagram

In figure 3.3, the schematic diagram represents the components of the system and their connection with the microcontroller at a local location.

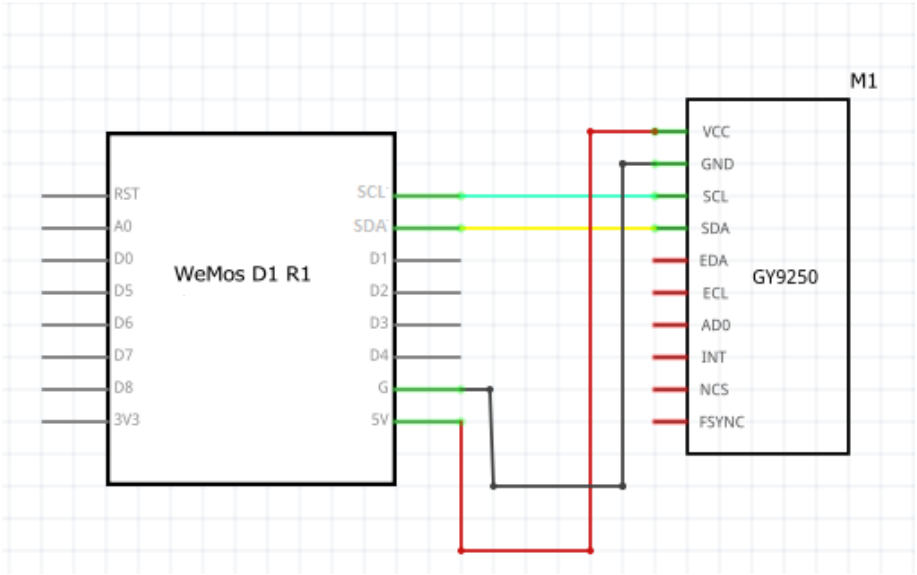

Figure 3.3: System Schematic Diagram - local location

In figure 3.4, the schematic diagram represents the components of the system and their connection with the microcontroller at a distant location via analog output pins.

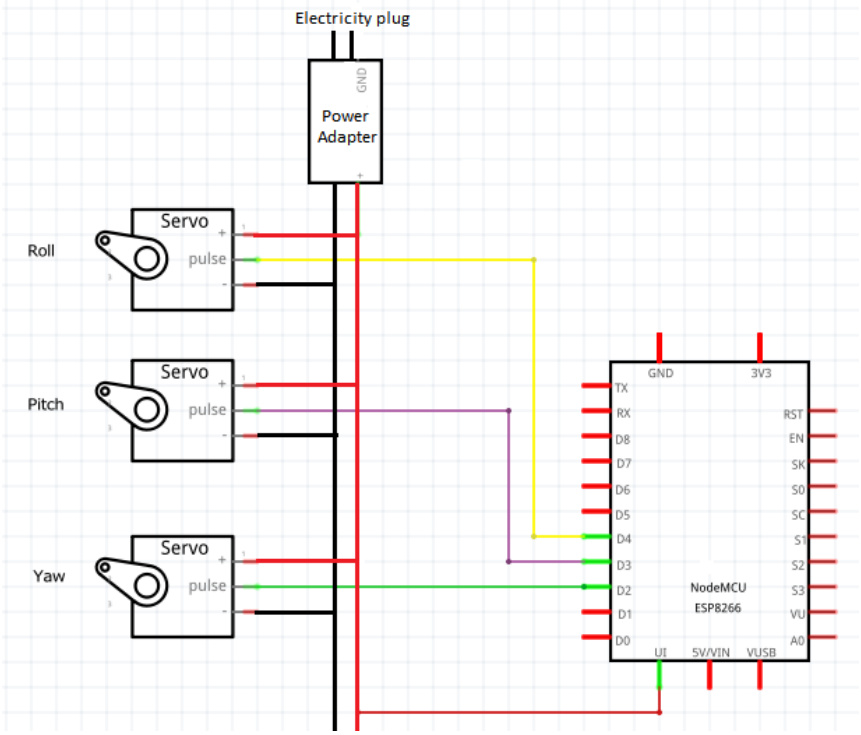

Figure 3.4: System Schematic Diagram - Distant location

3.5 Hardware Setup

At the local site, the user wears the headset as shown in figure 3.6, the sensors inside the headset measure three angles that represent the orientation of the user's head. The three angles represent pan, tilt and roll as shown in figure 3.5, and the measurements are transferred via wifi into PubNub's channel

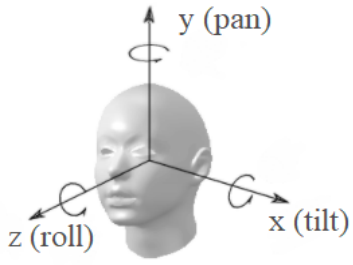

Figure 3.5: Head Rotation Angles

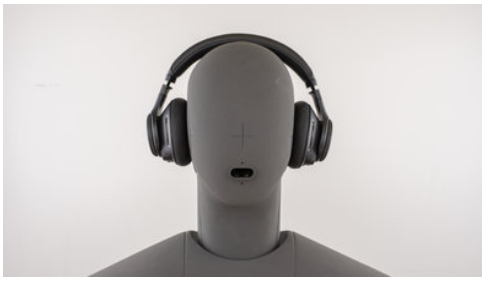

Figure 3.6: Headphone Placement, [23]

At the distant location, figure 3.7 shows an initial sketch of the proposed stand that contains three servo motors controlled by the NodeMCU; the actual design is shown in the implementation section. The movement of each axis is independent from the other two. At the same time, whenever a servomotor moves, the smartphone is surely affected by its movement. As shown in the figure, when the servomotor at y moves, the whole stand moves at y direction causing the pan movement. When the servomotor at x moves, the parallelogram moves along the x axis causing the smartphone to move up and down causing tilt movement. When the servomotor at z moves, it causes the smartphone to roll around the z axis.

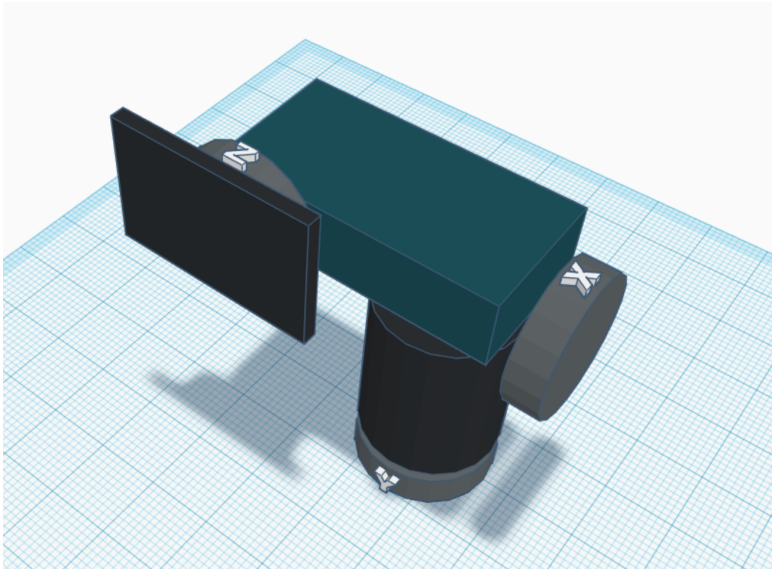

Figure 3.7: 3D sketch for the stand in the distant location

Chapter 4 : Implementation

4.1 Overview

This chapter describes the implementation part of our project with more details. It dives deep into the different hardware components of the system and its software with all of its modules.

4.2 Hardware Implementation

This section presents the process of assembling the hardware components of the two parts of our system.

The first part of our system consists of an MPU9250 sensor connected to a Wemos W1 R1 microcontroller. The microcontroller was powered using a 9V battery.

The first step on this side was to find a way to measure the participant's head orientation without causing him disturbance because the sensor should be somewhere on the head to measure the orientation. We figured that gluing our components on a headphone would suit our needs. So we brought a non-working headphone and arranged our system on it.

The microcontroller was fixed on the side of the headphone with the battery next to it. Then we placed the sensor at the top, since it was the most appropriate place to get accurate measurements for head orientations. Then, we wired the components together and the first part of our system was ready.

The final arrangement of the hardware components in this part is illustrated in figure 4.1.

For the second part of our system, the first step was to get an appropriate design that suits the movements we need to reflect. After searching online, we found an appropriate design. But it was not built for the same purpose so we needed to adjust the dimensions until the model reflected our needs.

Figure 4.1:Headset Implementation

Each component was designed while taking into consideration its placement within the other components. All model components are shown in Figure 4.2.

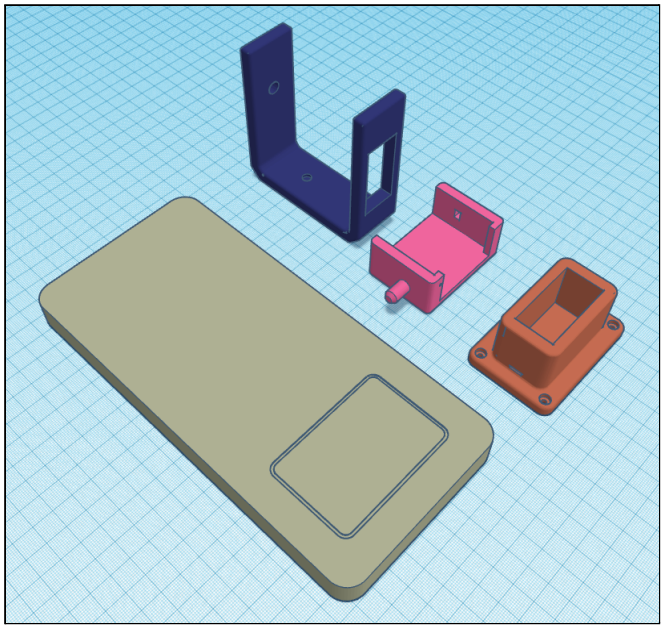

Figure 4.2: 3D model component

We used Tinkercad (a simple online software for 3d visualizations similar to Paint3D) for visualizing the stand and putting all components together, as shown in figure 4.3. The actual dimensions were used in tinkercad, and we also brought a 3D model of our servo motors. We virtually put all pieces together and studied their movement and measurements.

After we got satisfied with the result of the visualization, we sent the components for printing. The components turned out as expected. We brought them all together with the motors and we did not face any problems at that stage. Figure 4.4 shows the model after assembly.

After the assembly, we started testing the movement of each of the three motors individually to check if the parts moved as expected.

The next step was to wire all three motors so they can move simultaneously. A 5V power adapter was used to power the motors which were connected to it in parallel. For the movement control of the motors, each was connected into a different output pin at the NodeMCU. The NodeMCU was also powered by the same power adapter.

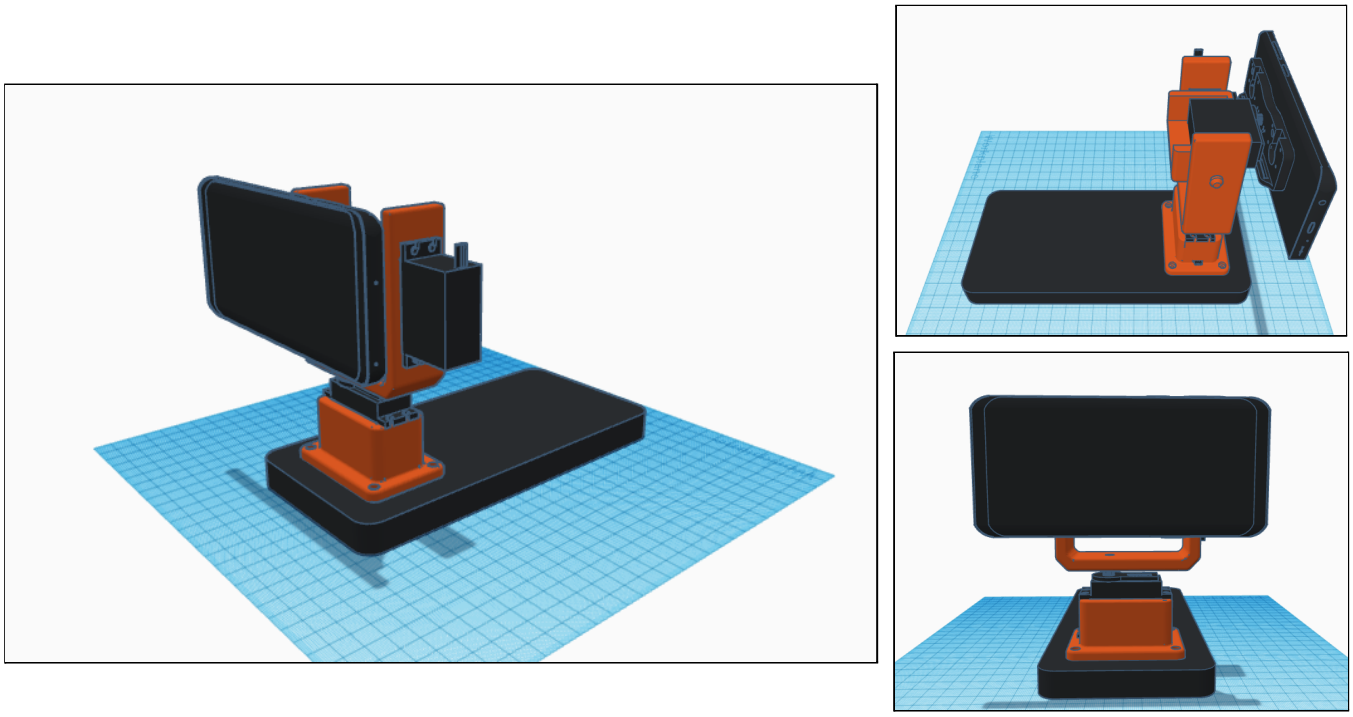

Figure 4.3:3D model

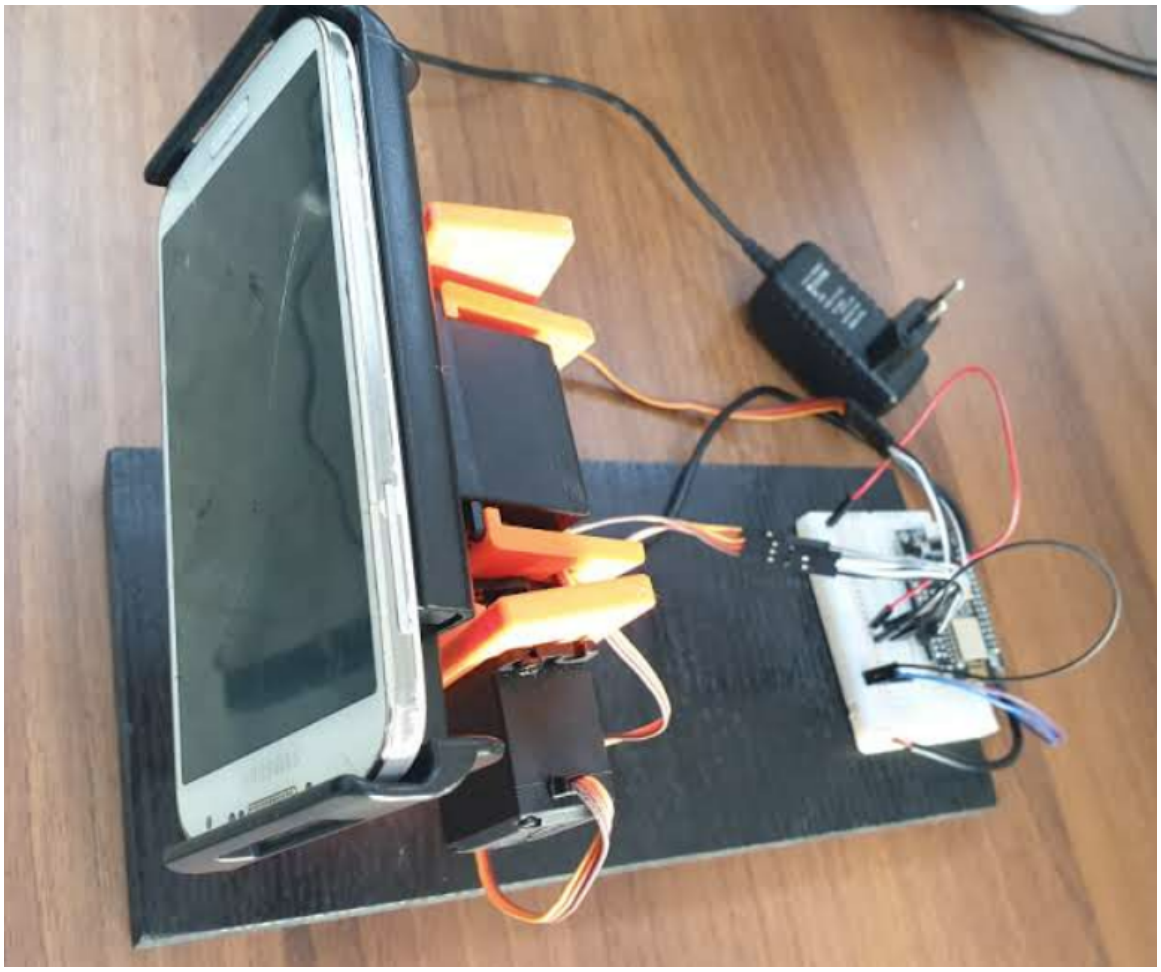

Figure 4.4: Model after assembly

4.3 Software Implementation

This section describes the implementation details of software components of the system. It also explains the features and functions of each software component in the system.

4.3.1 Control Libraries Installation

Both parts of our system were put into work using wifi-based microcontrollers. Node MCU connected to the motors and Wemos R1 D1 connected to the sensor.

For the Wemos R1 D1, we installed “MPU9250” library which is responsible for reading the data from the sensor.

For the NodeMCU side, we installed “Servo” library which allows for controlling the angle of rotation for the servo motors we used.

For the communication between the two parts, “PubNub” library was installed and the data was transferred as JSON objects.

4.3.2 Wifi Connection Module and data transmission

To allow for the communication between the two parts of our system a channel on PubNub website was reserved. Both sides communicate using this channel. The Wemos gets the measurements from the sensor, processes and converts them into angles, and then publishes them into the PubNub channel as a JSON object described in figure 4.5.

```
[
{
    "yaw" : 0,
    "pitch": 0,
    "roll": 0
},
{
    "yaw" : 45,
    "pitch": 90,
    "roll": 45
}
]
```

Figure 4.5: JSON object format

The NodeMCU subscribes into the same channel and gets the JSON objects, decodes the readings and reflects them each on the appropriate motor.

4.3.3 Theory

This section describes the process the measured angles go through until they are reflected properly on the motors.

The sensor has the ability to measure orientation 360° around each axis of movement. The range of the measured values are in the range (-180, 180). The measured angles refer to the sensor's own points of reference, i.e. a user might start at a position that is at angle 45° according to the sensor, see figure 4.6 (a). That means the angles cannot be immediately sent to the motors. The preparation process goes as follows:

First, we introduce the angle correction process which means that the start angle of the user gets marked as our point of reference, see figure 4.6 (b). Each measurement afterwards gets evaluated

according to that initial point of reference. Each reading gets referenced into start position. The correction process is done according to the following equation:

```
resultAngle = ( mod ( 180 + measuredAngle, 360 ) - referenceAngle ) - 180
```

Second, we trim the values into the set of possible human head movements, see figure 4.6 (c). For example, a person cannot move his head sideways more than 45° [24].

Finally, the angles are encoded and sent by wifi. Keep in mind the same process is applied along the 3 different axes of movement.

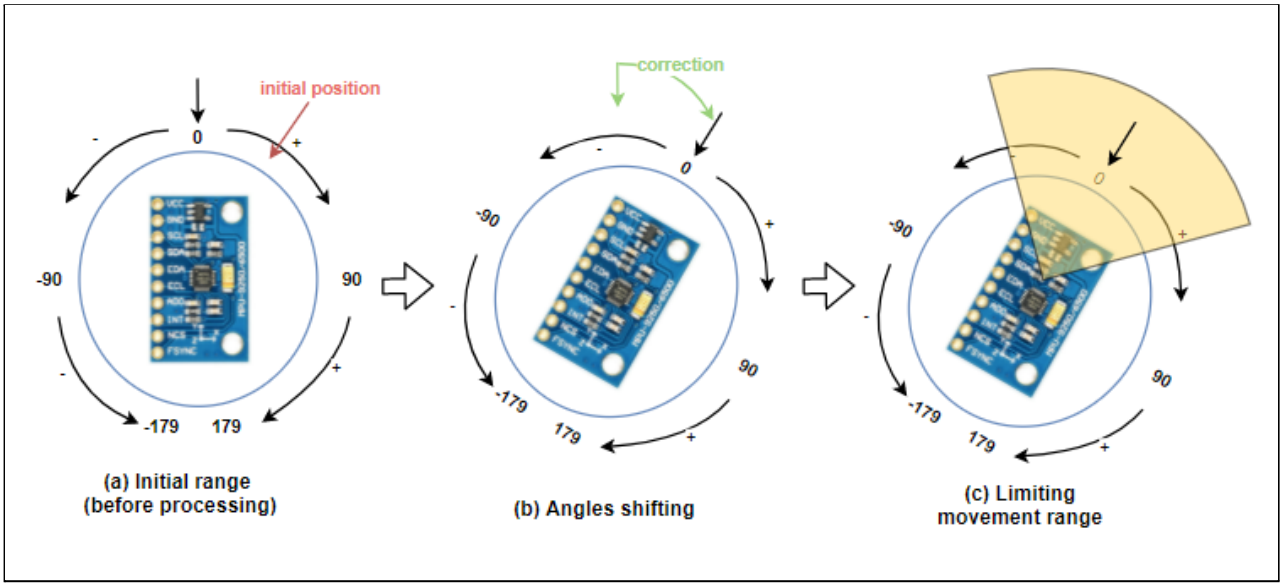

Figure 4.6: Steps for processing sensor angles along roll axis

At the receiver side, one more step is needed for the angles to be ready. The step is shifting the received values by 90°. This is needed because the motor understands angles within the range [0, 180] which means the center of the motor is 90, not 0. It is guaranteed that all angles fall within the correct range because of the trimming and processing that was previously done.

```
// map receivedAngle from range [-90, 90] to [0, 180].
motorAngle = receivedAngle + 90;
```

4.3.4 Coding

In this section, we will describe setup and loop functions for the two microcontrollers.

Publisher Part (Wemos)

Setup:

First, the microcontroller connects to the internet and waits until the process succeeds. When the connection is established, the microcontroller connects into PubNub channel that was already reserved.

After the connection with the channel is established, the microcontroller initializes the communication with the MPU9250 sensor. After the connection with the sensor is established, a wait time of 5 seconds is needed until the readings stabilize. The position of the sensor at that point is regarded as the reference position. Which means change in angles will all be relative to that position.

Loop:

In the loop function, angles update when the sensor reads a new data, the data is encoded as a JSON object and gets uploaded into PubNub channel. Any value beyond the scope of natural head movements gets discarded.

Subscriber Part (NodeMCU)

Setup:

In the setup function, the internet connection is established. Then, the microcontroller connects into PubNub channel as a subscriber. Then, the control pin of each of the three servomotors gets specified.

Loop:

In the loop function, the listener object is established. Then it accepts all receiving angels via the reserved channel. They are received as JSON objects which contain three different angle measurements. Each measurement gets reflected on the appropriate motor. with the shift of 90°.

Chapter 5: Validation and Results

5.1 Overview

In this chapter we will discuss the testing of all component of the system and the results obtained. We test all the parts to ensure that all of the functions work as expected and without errors.

5.2 Hardware Testing

This section discusses the testing process of each of our hardware components.

5.2.1 Testing MPU9250 Sensor

We connected the sensor with a NodeMCU microcontroller and tried to test the example code that comes with the MPU9250 library to measure the values. The sensor was wired as shown in figure 5.1. We realized that the measured angles change rapidly and that they keep drifting. After debugging the problem, we found out that the library implementation of the sensor uses 64 bits addressing while the NodeMCU uses 32 bits addressing, we looked for another working library but it turned out most of them are based on 64 bits addressing. As a result, we decided to change into a microcontroller that uses the same addressing and is not a downgrade from the processing power of the NodeMCU so we chose Wemos R1 D1.

After bringing in the Wemos, the angles displayed were reasonable. So we started performing predefined movements to see how the sensor responds. We tested the movement across the three different axes of movement shown in Figure 5.2. Each axis was tested separately as follows:

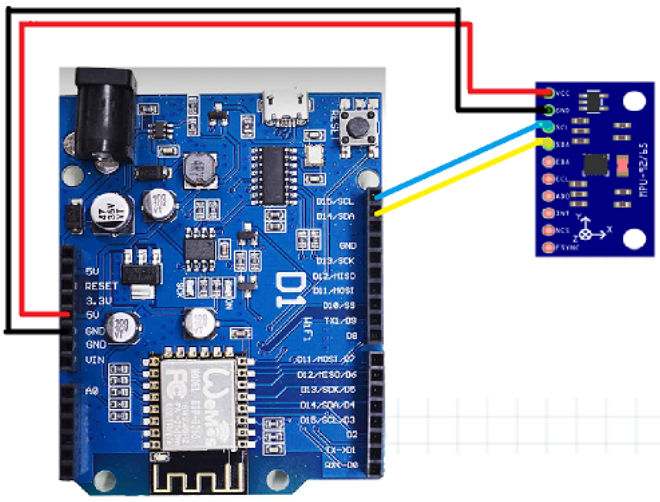

Figure 5.1: Testing MPU9250 sensor

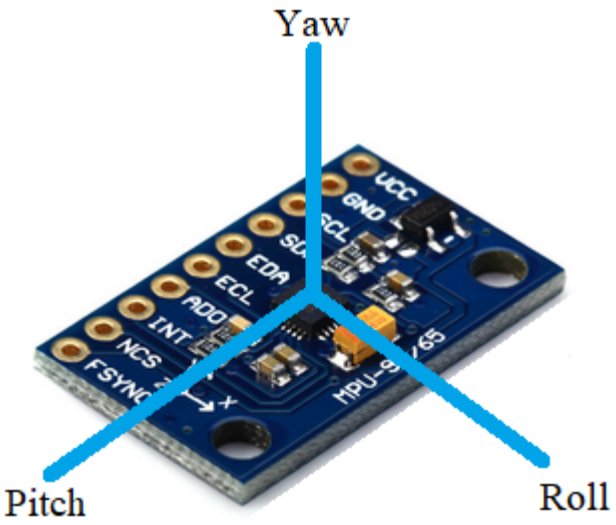

Figure 5.2: Three axes of movement

1. **Testing pitch values.**
   We rotated the sensor along the Pitch axis, 90 degrees in both directions. Then we did the same rotations but gradually. Finally we returned to the initial position to make sure the measured angle stays the same without drifting. The sensor passed our tests and reflected the angles as expected.

2. **Testing roll values.**
   We did the same test done for pitch, but this time along the roll axis. We checked for drifting and monitored how the angles change and the measured angles were as expected.

3. **Testing yaw values.**
   Again we moved the sensor 90 degrees in both directions along the yaw axis and the results were as expected.

4. **Testing combinations of multiple movements**
   The final test was to see the readings when multiple rotations were applied at the same time. The measurements were fine when each rotation was introduced individually, but we noticed a small difference of yaw when pitch and roll were combined. We debugged and searched for the problem and realized it occurred because the magnetometer of the sensor was not calibrated. The calibration process is done by waving the sensor in a figure eight for 30 seconds. The offsets are stored at the microcontroller's EEPROM and get used by the sensor. After the calibration, the drifting effect became minimal and did not cause any further problems.

5.2.2 Testing Wemos D1 R1

We did multiple tests to ensure that the Wemos is working correctly. First we uploaded a simple code that blinks the builtin LED and this way we made sure we were able to compile and upload the code with no problems. Then we tested the wifi module by uploading a code that lists available networks and then connected to one of them. After that, we tested input pins by connecting the MPU9250 sensor and uploading the code that reads data from it.

5.2.3 Testing Mg996r servo motor

We had three different servo motors. Each one of them was connected individually to the NodeMCU microcontroller, as shown in Figure 5.3, and we did two tests. One where we sent the degrees 0, 90, 180 separated by a 5 seconds interval to check the range of the motors. The second test was going gradually from 0 to 180 and back to 0, this test was done to check the smoothness of the movement.

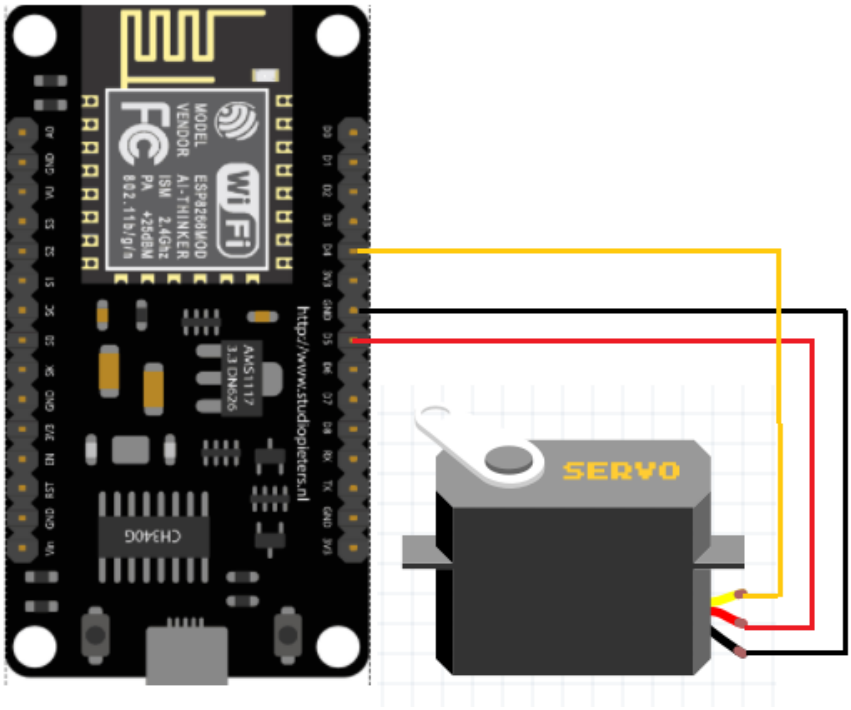

Figure 5.3: Testing Mg996r servo motor

5.2.4 Testing NodeMCU

The NodeMCU was tested in a similar fashion to the Wemos test. We uploaded the code for blinking the builtin led, tested the wifi module, and the output pins that we wanted to use.

5.2.5 Testing model movement

Our main concern with the movement of the model was checking if the natural movements of a human head was possible. So we were not focused on 180° along the three different axes. We tested

if the stand can do all required rotations without the components bumping and restricting the movement of each other as follows:

- Yaw movement: we tested for a range of 180° since the human head can turn 90 degrees in both directions (right and left).
- Pitch Movement: We assumed the start position to be parallel to the ground surface. The human head can move 90 degrees upwords and 70 degrees downwards so we tested movement of the pitch for those angles [24].
- Roll movement: We tested the stand for 45 degrees of rotation in both directions [24].

5.3 Software Testing

Most aspects of software testing were introduced in the testing of the related hardware components. In this section we will discuss what is left, which is the testing of PubNub library.

As discussed in the implementation section of PubNub, the Wemos connected with the sensor publishes data into the PubNub channel and the NodeMCU subscribes into the channel to get the data. For measuring the time it takes to receive the data between the two ends, we printed the timestamp at send and receive time and compared them to each other. As we can see, the measurements of {90, 50, 70} were sent at {06:656} sec and received on the other side at {06.860} sec with a difference of 0.204 seconds as shown in figure 5.4.

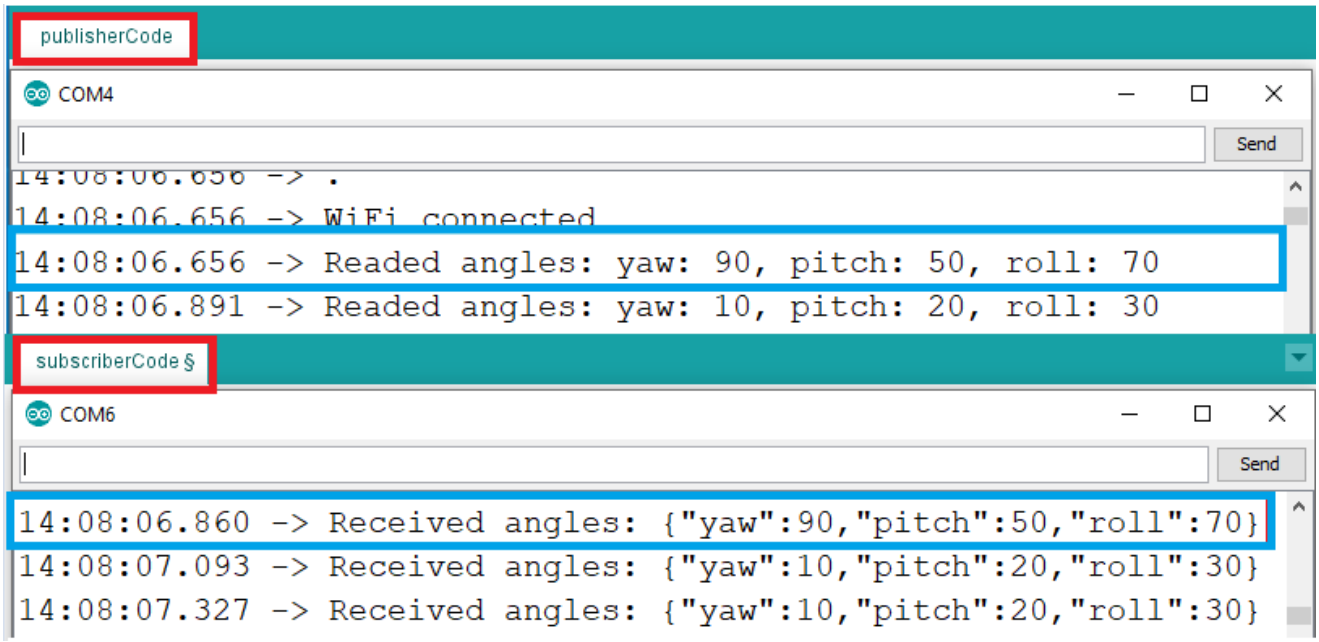

Figure 5.4: Sending and receiving time between publisher and subscriber

Chapter 6: Conclusion and Future Work

6.1 Conclusion

In this project, we have addressed the limitation of the traditional communication methods in transmitting certain nonverbal cues and have proposed a telepresence model that enables the participant to interact with a distant environment.
The model we build allows the participant to look around a distant environment by moving his head around. The model is represented by a stand that holds a smartphone and is attached to motors that are controlled by a microcontroller. The microcontroller receives the head orientations measured by sensors located on the participant's head via the internet. Then, the stand reflects those movements automatically, without having the user to explicitly control the stand.

We were able to build the system successfully and achieved our goal of enhancing the user experience of using remote video conferences. We managed to put together a wearable headset that measures head orientations. And constructed a stand that has 3 DOF movement that mimics head motion. As well as, we were able to transfer head movements between the two environments in a very short duration which allows for real-time interaction between the two far away parties.

6.2 Future Work

Ultimately, with this project, we aim to enhance the experience for both distant and local environments as much as we can. An enhancement over the current model would be adding mobility features. That will provide distant users with an extra level of freedom in navigating the distant location.

Another area of improvement would be adding a camera on the side that contains the sensor and building an application that supports starting and ending the session and allows for controlling the mobility of the stand in all four directions.

References

[1] Telepresence Definition, “https://www.techopedia.com/definition/14600/telepresence”, 18/11/2020.

[2] beam Telepresence Robot, “https://suitabletech.com/products/beam”, 18/11/2020.

[3] VGo Telepresence Robot, “http://www.vgocom.com/”, 18/11/2020.

[4] Stahl, Christoph & Anastasiou, Dimitra & Latour, Thibaud. (2018). Social Telepresence Robots: The role of gesture for collaboration over a distance. 409-414. 10.1145/3197768.3203180.

[5] Güven, Yılmaz & Coşgun, Ercan & Kocaoğlu, Sıtkı & Gezici, Harun & Yilmazlar, Eray. (2017). Understanding the Concept of Microcontroller Based Systems To Choose The Best Hardware For Applications. Research Inventy: International Journal of Engineering And Science. 7. 38.

[6] NodeMCU ESP8266, “https://developer.ibm.com/tutorials/iot-nodemcu-open-why-use/“, 19/11/2020.

[7] Picture of NodeMCU ESP8266, “https://components101.com/development-boards/nodemcu-esp8266-pinout-features-and-datasheet”, 19/11/2020.

[8] Parihar, Yogendra Singh. (2019). Internet of Things and Nodemcu A review of use of Nodemcu ESP8266 in IoT products. 6. 1085.

[9] Louis, Leo. (2018). Working Principle of Arduino and Using it as a Tool for Study and Research. International Journal of Control, Automation, Communication and Systems. 1. 10.5121/ijcacs.2016.1203.

[10] Accelerometer, Gyroscope and IMU Sensors in Robotics, “https://www.intorobotics.com/accelerometer-gyroscope-and-imu-sensors-tutorials/”, 13/03/2021.

[11] MPU9250, “https://invensense.tdk.com/wp-content/uploads/2015/02/PS-MPU-9250A-01-v1.1.pdf”, 20/03/2021

[12] WeMos D1 R1, “https://www.electroschematics.com/d1-wi-wi-board/”, 20/03/2021.

[13] MG996r servomotor, “https://www.electronicoscaldas.com/datasheet/MG996R_Tower-Pro.pdf” , 19/11/2020.

[14] Picture of Servomotor, “https://www.jsumo.com/mg996r-servo-motor-digital”, 19/11/2020.

[15] Stepper motor, “https://learn.adafruit.com/all-about-stepper-motors”, 19/11/2020.

[16] Picture of stepper motor, “https://www.ato.com/4-wire-nema-17-bipolar-stepper-motor-6v-0-8a-1-8-degree”, 19/11/2020.

[17] Volt Battery Snap, "https://evandesigns.com/products/battery-snap", 20/05/2021.

[18] Picture of 5V base 2A Power Adapter, "https://www.banggood.com/5V-2A-EU-Power-Supply-Micro-USB-AC-Adapter-Charger-For-Raspberry-Pi-p-949349.html?cur_warehouse=CN", 20/05/2021.

[19] Picture of 9V battery, "https://thomaselectricaldistributors.co.uk/wp-content/uploads/2019/06/GP-Ultra-PP3-Alkaline-9V-Battery.jpg", 20/05/2021.

[20] PubNub, "https://www.pubnub.com/docs/", 25/3/2021.

[21] What is Latency and Why Does it Matter?, "https://www.pubnub.com/blog/what-is-latency-and-why-does-it-matter/", 25/3/2021.

[22] PubNub free transactions, "https://support.pubnub.com/hc/en-us/articles/360051973971-Can-I-use-PubNub-for-free-", 25/3/2021.

[23] Picture of headphone placement, "https://www.rtings.com/headphones/learn/over-ear-vs-on-ear-vs-in-ear-vs-earbuds-comparison", 20/5/2012

[24] Natural movement of human neck, "https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1250253", 25/3/2021.