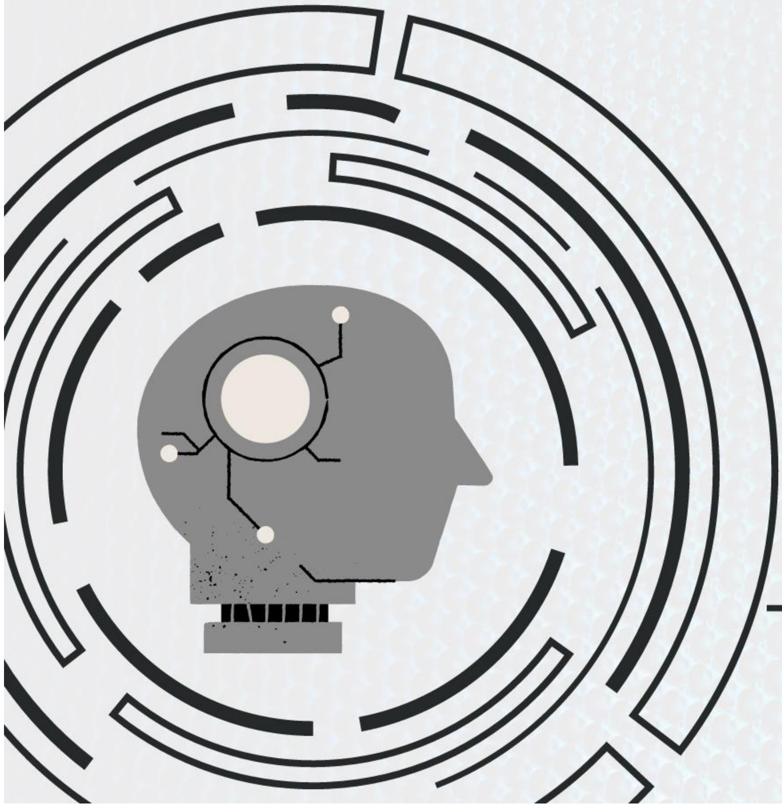
الكلية التكنولوجية بالقاهرة
Technological Faculty in Cairo

جامعة مصر التكنولوجية الدولية
MITU
MISR INTERNATIONAL TECHNOLOGICAL UNIVERSITY

BULBUL ROBOT

Intelligent medical assistant robot



AI
Department
Supervisor : Dr.wafaa
2024

Our Team Work
Mohamed Zain
Kirlos Ayad
Ahmed Nasser
Mohamed Amr
Ahmed sakr
Martin Emad
Mohammed Ahmed
Ahmed Amin
Hassan Mohamed
Ahmed Adel
Ahmed Abdelkader

Part 1(software)

1.1 Introduction

Hospitals are lifesavers, offering critical medical care during times of illness or injury. Yet, for many patients, the sterile environment and separation from loved ones can create a sense of isolation and loneliness. This isolation can have a negative impact on well-being and even hinder the healing process

1.2 Problem definition

Hospitals, while essential for treating illness and injury, can be sterile and isolating environments. Patients often find themselves confined to a bed, separated from loved ones and their daily routines. This lack of social interaction can have a significant negative impact on their well-being and hinder the healing process.

1.3 The Cascade of Consequences

Social isolation in hospitals can lead to a cascade of negative consequences, including:

- **Decreased Morale and Motivation:** Social isolation can rob patients of a sense of purpose and connection. The absence of familiar faces and social interaction can lead to feelings of helplessness and a lack of motivation to participate in recovery efforts.
- **Increased Anxiety and Depression:** The unfamiliar surroundings and lack of social interaction can heighten anxiety levels in patients. Coupled with feelings of loneliness, this anxiety can morph into depression, further hindering the healing process.
- **Poorer Sleep Quality:** Social isolation can disrupt a patient's sleep cycle. The lack of daytime social interaction can make it difficult to fall asleep at night, and feelings of loneliness or anxiety can lead to fragmented sleep. Insufficient sleep weakens the immune system and hinders the body's ability to heal.
- **Slower Recovery Times:** The negative emotional and physical consequences of social isolation can all contribute to a slower recovery for patients. Reduced motivation, poorer sleep quality, and heightened anxiety can impede healing and prolong hospital stays.
- **Increased Need for Pain Medication:** Social isolation can exacerbate pain perception. The emotional distress caused by loneliness can amplify feelings of pain, leading to a greater reliance on pain medication.

By addressing social isolation in hospitals, Bulbul has the potential to mitigate these negative consequences and promote faster recovery times, improved patient well-being, and a more efficient healthcare system.

Figure 1: The state and feeling of the patient in the hospital room

1.4 Who is Bulbul?

More Than Just a Machine: Bulbul - The Intelligent Assistant System

Bulbul is more than just a robot; it's a novel intelligent assistant system designed to completely transform the patient's experience. It transcends the limitations of a traditional machine, acting as a comforting and interactive companion specifically tailored for the hospital setting. Bulbul seamlessly combines entertainment features with crucial post-operative follow-up, ensuring patients feel supported and connected throughout their recovery journey.

Figure 2: Bulbul Robot

A Multifaceted Approach to Patient Care

Bulbul's functionalities extend far beyond basic entertainment. Let's explore the key features that make Bulbul such an innovative force in patient care:

- **Patient Interaction:** Bulbul fosters a sense of companionship by engaging in conversations on a variety of topics. This human-like interaction combats feelings of isolation that can often plague hospitalized patients.
- **Follow-Up Care:** Bulbul transcends its role as a companion by offering crucial post-operative support. It can be programmed to check in with patients about their well-being, ensuring medication adherence and identifying any potential concerns before they escalate. This proactive approach fosters a sense of security and promotes faster recovery.
- **Fulfillment of Needs:** Bulbul empowers patients by facilitating requests for basic needs like food and water. This promotes a sense of autonomy and control, which can significantly improve a patient's morale during their hospital stay.

Bulbul represents a significant leap forward in patient care. By combining entertainment, follow-up support, and need fulfillment, Bulbul transforms the hospital experience from sterile and isolating too supportive and interactive. This innovative system has the potential to revolutionize the healthcare industry, fostering faster recovery times and improved patient well-being.

2.1 Technical Architecture:

The Bulbul system operates in three distinct stages:

1-Audio Capture: The WO-Mic software installed on a mobile device captures the patient's voice input
.

2-AI Processing: The captured audio is transmitted to a central laptop server. Here, sophisticated artificial intelligence models analyze the audio and generate a tailored response
.

3-Response Delivery: The Space Disk program processes the AI-generated response and transmits it back to the patient's bedside via the Bulbul Robot's tablet screen, providing a visual and auditory output (Bulbul Response)

2.2 How connect audio with server?

When utilizing a mobile phone as a client device in conjunction with Wo Mic, it functions as an audio sensor, capturing sound from the environment. This recorded audio data is then transmitted wirelessly via Wi-Fi to the server, which is a laptop in this scenario.

2.3 Using a Laptop as a Server in the Client-Server Model

In the client-server model, a laptop acting as a server distributes information and services to other networked computers, known as clients. This architecture facilitates functionalities such as data sharing, resource allocation, and computational tasks. A single server can handle multiple clients, and conversely, a single client can interact with multiple servers. The client process can either reside on the same device or connect remotely over a network to access services provided by the server.

2.4 Connect Wo mic with space desk

Space desk allows a laptop to extend its display onto a tablet, providing an additional screen for expanded workspace or multitasking. Integrating Wo Mic via Wi-Fi enables the tablet to function as a wireless microphone, enhancing audio capabilities for virtual meetings and presentations.

Integrating Wo Mic with Space desk via Wi-Fi enhances the functionality of a laptop used as a server. Space desk extends the laptop screen onto a tablet, effectively transforming the tablet into an additional display. Concurrently, Wo Mic enables the tablet to serve as a wireless microphone or audio sensor. This configuration allows the laptop server to manage and process audio inputs received from the tablet, significantly enhancing the system's versatility for tasks

3.1 Speech

One of the most important parts of the robot project is to interact with the patient.

For example, if the patient needs to know information, deliver information to someone, search for information he wants or even entertain his time and talk to someone during the time of isolation from the health in his place of isolation.

There is supposed to be an interactive language between the helper robot and the patient to achieve all the above examples and much more.

To achieve this, audio libraries were used to make the robot able to listen to the patient and then respond to it with a voice that suits being a robot and in a calm manner for the quality of listening.

Now will explain how to complete this task in detail, step by step:

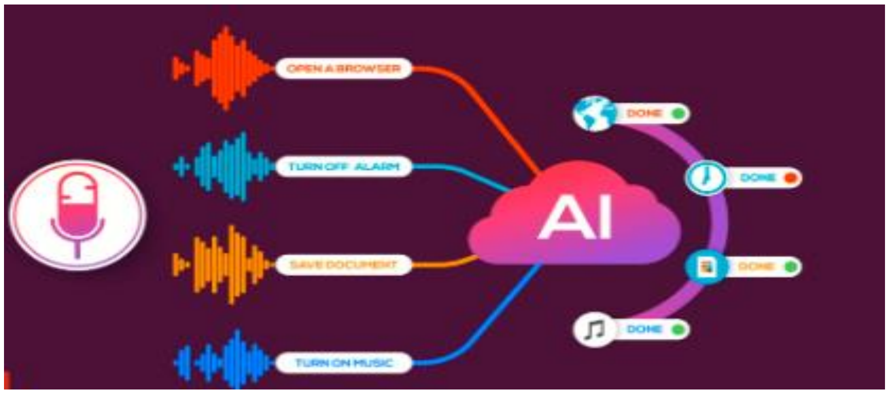


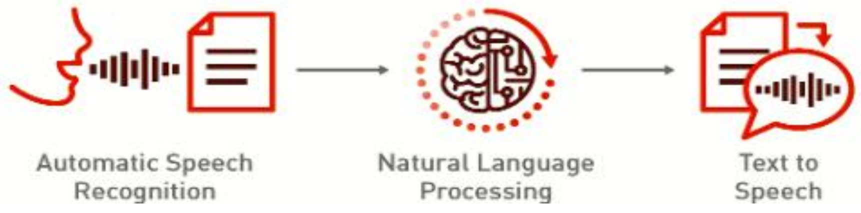
Figure 3: Speech

3.2 Why the robot with a voice?

- Sometimes the patient needs a friend who will comfort his loneliness and talk and discuss with him while he is in a state of individual isolation without anyone
- If the patient feels some fatigue or symptoms and does not know what to do in these cases, in this case he can talk to the robot and tell him how he feels to be able to describe or guide him.
- If the patient’s state is difficult and he feels tired or severe symptoms, in this case the patient will tell the robot and then the robot will realize that the situation is dangerous and he contacts the treating doctor, also, if the patient wants to send notes to the doctor, he can do this by listening to the robot.

3.3 Connect audio with the robot

- To connect listening and speaking with the robot, this was done by using Python language and its audio libraries inside.
- Initially, two very important libraries were used, the first of which was a library " Speech_Reconition". This library is used for voice recognition from the user (patient) and then converted into text.
- The second library is the "pyttsx3" library, The library used to convert the respond text from the AI model (answer to the patient) to a voice that the patient listens to.
- To set up a library pyttsx3 to be convert text to voice used "pyttsx3.init()", then to get all the votes available inside the library to use it was used "getProperty("voices")" and then select the right voice for the robot from the list of voices using "voices[1].id", also can select rate of speed of talking robot by "setProperty('rate', 130)"And can change rate of speed.
- To make a button when you press it, it will receive the audio, through a library "Tkinter" that displays GUI inside this button "speak" and another buttons, button "speak" it is used in a way that when pressed, it listens to the patient's voice
- When the text is recorded and a new sound is taken, the previous text is deleted by "delete ('1.0', Tk.end) when TK is summary to Tkinter library, then a word is written to listen
- Then prepare the patient's voice listening function by turning on the microphone "sr.Recognizer and sr.Microphone" when sr summary to Speech_Reconition library, and adjusting it against the ambient noise.
- Send the text to Google Translator to convert it to text in English "recognize_goggle(audio, language='en')".
- Then made a field in Arabic to translate the response into Arabic only and show it in the GUI and not pronounce it.
- In the end, when the text is pronounced with the user's voice the robot's voice by "say(audio)" when audio variable contain the text, then waiting to speak again after talking by "runandwait()".



Figar 4: Speech Recognition

3.4 Connect audio with motion command

- If the patient wants to send the robot to someone without holding the movement device, he can do it with a few small words, we can make it turn left or right, or walk forward or backward, simply and with a few simple words such as forward, backward, right and left.
- One of the best of those commands id to make him walk on the line prepared for him to reach the patient with a few simple words such as auto, go to the patient, go to the other side, go to the next side and turn around for example.

4.1 API connect and database

If the patient wants to talk to the robot spontaneously about medical matters or anything he wants, how will the robot be able to answer it clearly and accurately?
This step is done by two steps: making the database for the bulbul robot and then connecting the robot with an AI model as "Gemini" by connect with API.
Now the robot can be interactive with the Response and Request with the AI model and the database

4.2 What's database?

- A database of Bulbul Robot has been created that contains his private information to know who he is and what we want from him
- These examples include your name Bulbul, you are a doctor's assistant, you work in this hospital. What should you do when consulting? How to respond appropriately. And some of the many things that have been provided in the database to form the full character
- The database is linked with the AI model, in a way that makes it think that it is the robot. When the model receives the question from the patient who responds, the robot is Bulbul, not the model.

4.3 Connect API with the robot.

- The Gemini model has been used to connect with the robot and the database of Bulbul by 'google.generativeai library '.
- Each person has its own API key inside the model, that key is imported to be connect by the model and the robot then import and execution this key by "configure(api_key=google_api_key)".
- Then the version of the current available model and the linking method that will be carried out by "GenerativeModel('gemini-1.0-pro-latest')".
- Now after the user has said the message or question, will request this message to Gemini model by API reset by "model.generate_content(conversation) " when the conversation variable contains the message.

4.4 Full connect API, database, Robot

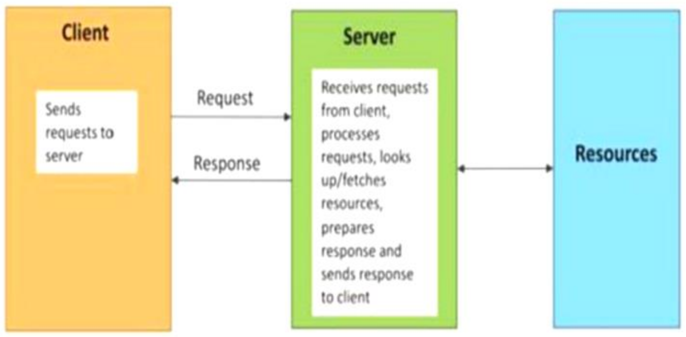

Client
Sends requests to server
Request
Response
Server
Receives requests from client, processes requests, looks up/fetches resources, prepares response and sends response to client
Resources

Figure 5 API Connect

- At first, the patient or the user will start asking for help though a question, information or advice.
- Second, this message will be request by API to the server in charge of the project which is Gemini model
- Thirdly, Gemini model relates to Bulbul database which in figure is resource, this connect it makes the server think it is now a robot Bulbul with your information and is ready for any question
- Fourthly, When the Gemini model (Bulbul now) request the message, it will respond appropriately and send this message to user or patient to be displayed by sound waves that the patient hears

5.1 Graphical user interface (GUI):

- Itis a common method of interaction between a user and a computer, software, or website. Graphical user interfaces provide a more intuitive, intuitive, efficient and attractive way to interact with computers, making them easier to use and reach a wider audience.
- Use the Tkinter Python library to perform virtual destination simulation with users.

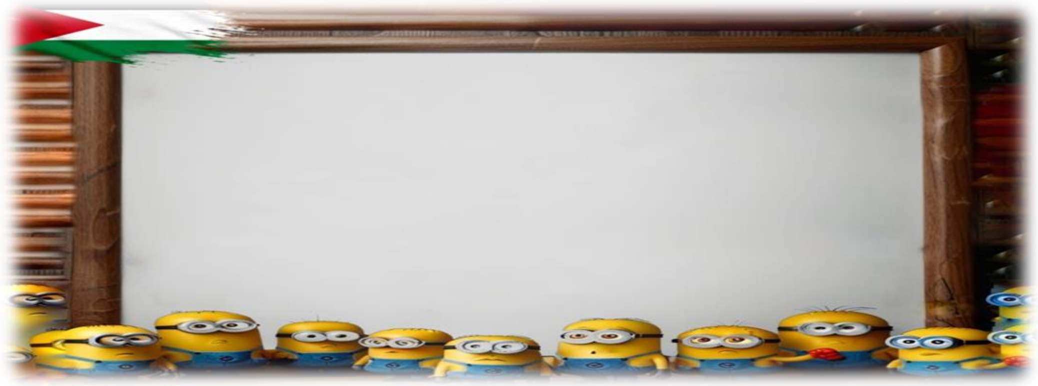


Figure 6: background

The graphical user interface (GUI) of the robot consists of two main sections:

5.2 To interact with the robot:

Chat button: Open the conversation window with the robot.

- ➢ Follow the conversation easily.
- ➢ Understand exactly what the robot is saying.
- ➢ Review the conversation later.

Robot text box: View what the robot says while it talks to you.

- ➢ Follow the conversation easily.
- ➢ Understand exactly what the robot is saying.
- ➢ Review the conversation later.

Text translation box: Translating robot texts into Arabic.

- ➢ Understand what the robot is saying easily with a translation from English to Arabic

5.3 To control the robot's movement:

Back button: Move the robot backwards.

- ➢ Move the robot away from an object or free up space.
- ➢ Return to a specific back point.
- ➢ Avoid colliding with obstacles.

Forward button: move the robot forward.

- ➢ Move the robot towards a specific destination.
- ➢ Reaching a specific place.

Left button: move the robot to the left.

- ➢ Change the robot's direction.
- ➢ Reach a place to the left of the robot.

Right button: move the robot to the right.

- ➢ Change the robot's direction.
- ➢ Reach a place to the right of the robot.

auto button: Move the robot to the line

- ➢ Direction to the line for easy rotation of the robot
- ➢ No obstruction occurs outside the line according to the decree

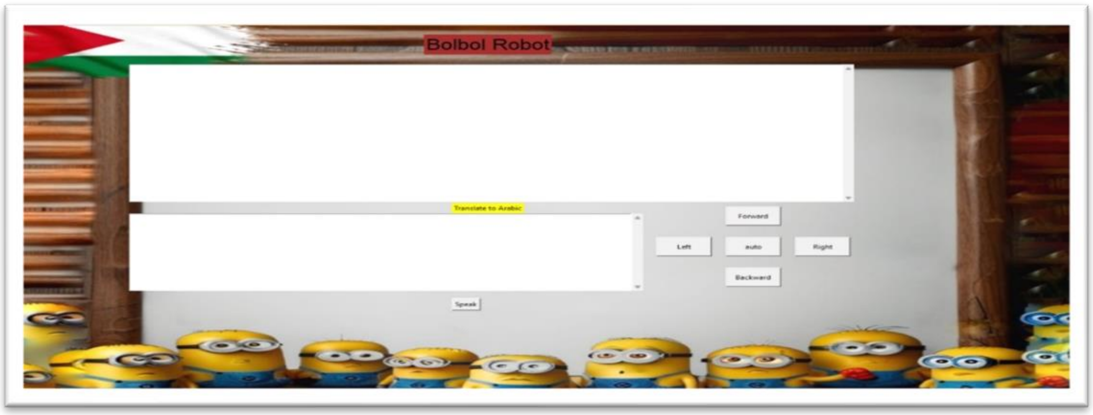


Figure 7: Background after adding

6.1 Connecting and controlling a Bulbul Robot via HC-05 Bluetooth Module.

In this guide, we will explore how to connect a computer to an HC-05 Bluetooth module and send commands to control a robot. This involves establishing a serial connection over Bluetooth and sending specific commands to move the robot in different directions.

6.2 Prerequisites:

Before we start, ensure you have the following:

- A computer with a Bluetooth adapter.
- An HC-05 Bluetooth module connected to the robot.
- Python installed on your computer.
- The pyserial library installed (pip install pyserial).

6.3 How the Bluetooth is connect and work?

Establishing the Bluetooth Connection.

To begin, we need to set up a serial communication link between the computer and the HC-05 module. The following steps detail the connection setup:

1.Identify the Bluetooth Serial Port:

- Determine the serial port that the HC-05 module is using. This is typically named something like COM5 on Windows.

2.Configure the Serial Port:

- Define the port and the baud rate. The baud rate for HC-05 is commonly set to 9600.

3.Open the Serial Port:

Establish the connection and allow some time for the connection to stabilize.

Sending Commands to the Robot:

Once the connection is established, we can send commands to the robot to control its movement. The robot will interpret specific characters as movement commands. We will define a method move that takes a direction and sends the appropriate command via the serial connection.

Define Movement Directions:

The robot can move forward, backward, turn left, or turn right. Each direction corresponds to a specific character command:

"F" for forward
"B" for backward
"L" for left
"R" for right

Send Commands via Bluetooth:

- The move method sends the command based on the specified direction.
- It uses the(write)method of the serial connection to send the command to the HC-05 module.

Putting It All Together:

With the serial connection established and the(move)method defined, you can now control your robot by creating an instance of(Robot Controller)and calling the move method with the desired direction.

6.4 Imports:

```
from machine import UART
from machine import Pin
```

These lines import two modules from the **`machine`** library, which is commonly used for microcontroller programming in Python environments like Micro Python.

- **`UART`:** This module provides functionality for interacting with Universal Asynchronous Receiver/Transmitter (UART) peripherals. UART is a serial communication protocol that enables data exchange between devices using a single transmit (TX) line and a single receive (RX) line.
- **`Pin`:** This module allows you to control individual pins on the microcontroller. Pins can be configured as inputs (to read data) or outputs (to control devices connected to them).

1. LED Pin Setup:

```
led = Pin(25, Pin.OUT)
```

This line creates a `Pin` object named `led` and configures it as an output pin. Here's a breakdown:

- **`led = Pin(25, Pin.OUT)`:**
  - **`led`**: This is the name you're giving to the pin object for easy reference in your code.
  - **`25`**: This specifies the physical pin number on the microcontroller (consult your board's documentation for pin mapping).
  - **`Pin.OUT`:** This sets the pin's direction to output, meaning you can use it to send a signal (usually high or low voltage) to control an external device.

UART Peripheral Initialization:

```
uart = UART(0, 9600)
```

This line creates a `UART` object named **`uart`** and initializes it for communication. Here's what it does:

- **`uart = UART(0, 9600):`**
  - **`uart`**: This is the name you're assigning to the UART object.
  - `0`: This indicates the specific UART peripheral you're using on the microcontroller (some boards might have multiple UARTs). Check your board's documentation for details.
  - `9600`: This sets the baud rate (communication speed) to 9600 bits per second. Common baud rates include 9600, 115200, etc. The baud rate needs to match the baud rate of the device you're communicating with.

The next code defines a loop that listens for serial commands sent through the UART connection and executes corresponding actions on the microcontroller.

Checking for Incoming Data:

```
if uart.any() > 0:
```

- This line checks if there's any data available in the UART receive buffer using the `uart.any()` method. It returns a value greater than 0 if there's at least one byte of data waiting to be read.

Reading Incoming Message:

```
message = uart.readline()
```

- If data is available, **`uart.readline()`** reads a line of text (up to a newline character) from the receive buffer and stores it in the `message` variable. However, it's important to note that **`readline()`** might return `None` if no complete line is available yet.

Handling Potential **`None`** Message:

```
if message is not None:
```

- This check ensures that we only proceed if a valid message was read. If `readline()` returned `None`, this block gets skipped.

Decoding the Message:

```
message = message.decode("utf-8")
```

- Assuming the data is sent in UTF-8 encoding (a common text encoding format), this line decodes the raw bytes received into a human-readable string.

Printing the Received Message:

```
print(message)
```

- This line prints the received message to your console or serial terminal, which can be helpful for debugging purposes.

Control Flow Based on Received Commands:

```
if message == 'B' :
    move_backward()
if message == 'F' :
    move_forward()
# ... similar checks for L, R, H, J, S
```

- The code then checks the content of the `message` variable against predefined characters:
  - 'B': Triggers a function `move_backward()` that you'll need to define to control your device's backward movement.
  - 'F': Triggers a function `move_forward()` that you'll need to define to control forward movement.
  - Similar checks are performed for 'L' (turn left), 'R' (turn right), 'H' (potentially turn on LED and call `auto()` - function unclear without further context), 'J' (potentially turn on LED and call `returnline()` - function unclear), and 'S' (stop).

2. **Important Considerations:**
   - Make sure you have defined the functions `move_backward(), move_forward(), turn_left(), turn_right(), auto(),` and `returnline()` elsewhere in your code to provide the desired behavior based on the received commands.
   - The code assumes **UTF-8** encoding for the messages. If the sending device uses a different encoding, you'll need to adjust the decoding step accordingly.

6.5 Code Functionality of (Raspberry pi Pico 2040)

This part explains the functionality of a Python script designed for a microcontroller-based robotic system. The code manages motor control, IR sensor input, and UART communication to navigate and respond to various commands. The primary components used include the Pin and PWM classes from the `machine` module, along with UART communication and time management utilities.

Import Statements and Initial Setup

The script begins with the necessary import statements:

```python
from machine import Pin, PWM   Importing Pin and PWM classes
import time   Importing time module
import utime   Importing utime module
from machine import UART   Importing UART class
```

These imports enable the use of GPIO pins, PWM for motor control, and UART for serial communication.

GPIO Pin Configuration

The code sets up GPIO pins for LEDs, motors, and IR sensors:

```python
led = Pin(25, Pin.OUT)
uart = UART(0, 9600)   Setting up UART with baud rate 9600

 Defining motor pins
motor1 = Pin(15, Pin.OUT)
motor2 = Pin(11, Pin.OUT)
motor3 = Pin(12, Pin.OUT)
motor4 = Pin(13, Pin.OUT)

 Defining enable pins and PWM objects
enable1 = PWM(Pin(6))
enable2 = PWM(Pin(7))

 Defining right and left IR digital pins as input
right_ir = Pin(2, Pin.IN)
left_ir = Pin(3, Pin.IN)
```

Here, the LED is set as an output on pin 25, and UART is initialized on port 0 with a baud rate of 9600. The motor pins are defined for controlling four motors. Two enable pins are set up with PWM objects for speed control, though their frequency and duty cycle settings are commented out. The IR sensors are set as inputs to detect obstacles or line tracking.

Motor Control Functions

Several functions are defined to control the movement of the robot:

```python
 Forward
def move_forward():
    motor1.low()
    motor2.high()
    motor3.high()
    motor4.low()

 Backward
def move_backward():
    motor1.high()
    motor2.low()
    motor3.low()
    motor4.high()

 Turn Right
def turn_right():
    motor1.high()
    motor2.high()
    motor3.low()
    motor4.low()

 Turn Left
def turn_left():
    motor1.low()
    motor2.low()
    motor3.high()
    motor4.high()

 Stop
def stop():
    motor1.low()
    motor2.low()
    motor3.low()
    motor4.low()
```

These functions control the direction of the robot by setting the states of the motor pins to achieve forward, backward, left turn, right turn, and stop actions.

Autonomous Movement Based on IR Sensors

The `auto` and `returnline` functions enable the robot to move autonomously based on IR sensor inputs

```python
def auto():
    right_val = right_ir.value()
    left_val = left_ir.value()
    print(str(right_val) + "-" + str(left_val))
    if right_val == 0 and left_val == 0:
        move_forward()
    elif right_val == 1 and left_val == 0:
        turn_left()
    elif right_val == 0 and left_val == 1:
        turn_right()
    else:
        stop()

def returnline():
    while True:
        right_val = right_ir.value()
        left_val = left_ir.value()
        print(str(right_val) + "-" + str(left_val))
        if right_val == 1 and left_val == 1:
            move_forward()
        elif right_val == 1 and left_val == 0:
            turn_left()
        elif right_val == 0 and left_val == 0:
            move_forward()
        elif right_val == 0 and left_val == 1:
            stop()
```

The `auto` function reads the values from the IR sensors and adjusts the robot's movement accordingly. The `returnline` function keeps the robot moving forward while monitoring the IR sensors to adjust the direction if needed.

Main Loop and UART Communication

The main loop continuously checks for UART messages and IR sensor values to control the robot:

```python
while True:
    right_val = right_ir.value()
    left_val = left_ir.value()
    if uart.any() > 0:
        message = uart.readline()
        if message is not None:
            message = message.decode("utf-8")
            print(message)
            if message == 'B':
                move_backward()
            if message == 'F':
                move_forward()
            if message == 'L':
                turn_left()
            if message == 'R':
                turn_right()
            if message == 'H':
                led.value(1)
                auto()
            if message == 'J':
                led.value(1)
                returnline()
            if message == 'S':
                stop()
```

The loop checks for incoming UART messages and executes corresponding functions based on the command received. It also reads the IR sensor values to adjust the robot's movement.

This script effectively integrates motor control, sensor input, and UART communication to control a robotic system. The robot can move in various directions, respond to IR sensor inputs, and receive commands via UART to perform specific actions, making it suitable for applications like line following or obstacle avoidance.

7.1The design of the chassis :

is one of the initial steps of the robot, everything is built on it. Initially, we were thinking as a group to get an ideal chassis shape and a suitable type of wood to be lightweight on the engines .The blackwash Wood was chosen and the chassis was designed, but it turned out that The Shape of the wood was not the best and we mentioned that it was not the best choice.

This is the shape of the chassis.

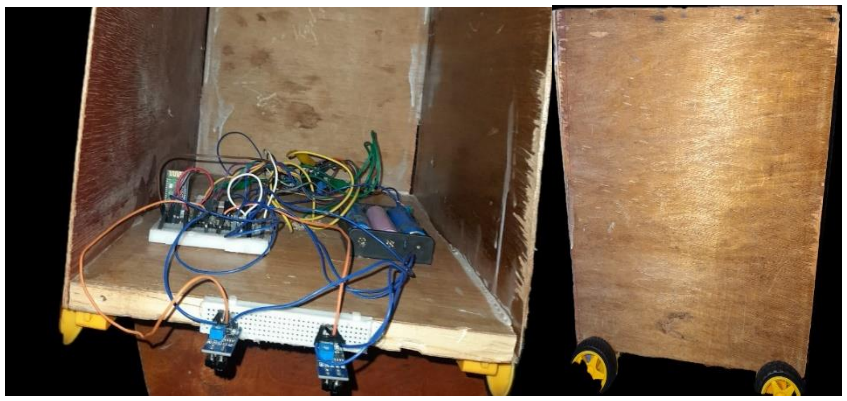


Figure 8: First shape

So, we decided on a new chassis design, where the old chassis was replaced by a new chassis that is better in terms of shape and weight on the engines, using the previous chassis design. The chassis includes a good medium internal space for placing medicines and the like, there is a place at the top to install the Monitor on the chassis, and the monitor is oriented at an angle for the patient's convenience while looking at it, rather than placed vertically. The size of the chassis 35 * 35 * 35 cm (length, width, height).

And this is the shape.

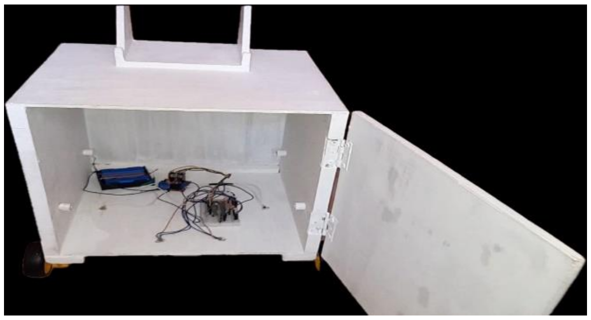


Figure 9: Second shape

Wooden pieces were placed under the chassis to place the wheels and avoid contact of the wires with the ground, which facilitates the movement of the robot. The colors of the chassis were selected with the consent of all members of the group after several selections,
and these colors were selected.

The chassis design was drawn on reality, this is the final shape of the chassis (the final image of the robot).

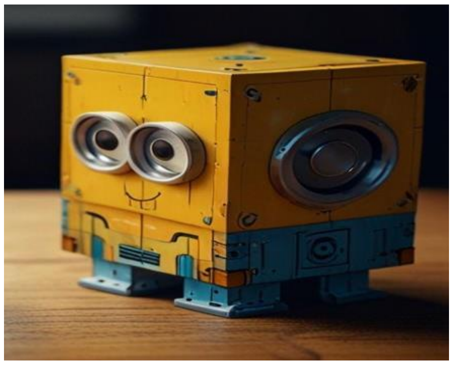


Figure 10: Final Design

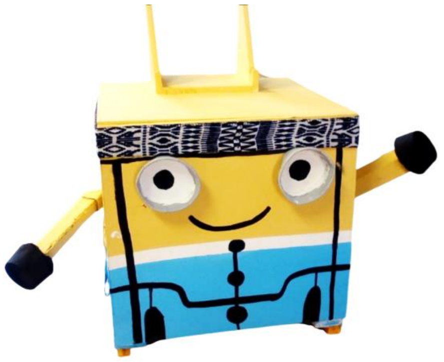
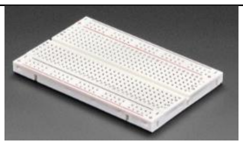
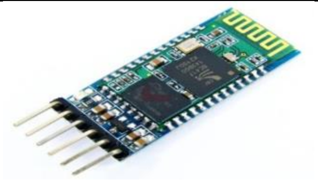


Figure 11: Final shape

7.2 Components

Rechargeable battery	
Battery case holder 3 cells	
wires	
DC gearbox motor	
Raspberry pi pico	
Bread board	
Wires	
Driver motor	
Bluetooth module HC-05	

Steps

- We used rechargeable batteries to be easy to use.
- They are 1500 MP batteries. They can be recharged in 30 minutes and last for 8 hours.
- We used 6 batteries, fixed on a strong wooden base in Two battery case holder and gathered the wires (in-out) to the motors Connect the 6 motors with wheels to the driver motor.
- The motors are 150 RPM each - total power is 900 RPM.
- They are well-fixed to the base.
- The base is a strong wood rectangle for this model only.
- The driver motor is fixed to the front of the base to control the 6 wheeled motors. The machine moves automatically on a black line on the Floor from the initial point to each patient room with the medicine.
- Then the patient speaks to the machine to go back to the initial point.
- The breadboard is a microchip to connect the driver, Bluetooth and raspberry pi together.
- The Bluetooth chip is used to connect to the tablet which speaks. to the patient with the machine to connect software and hardware together.
- We used the raspberry pi because this chip is programmed with python.

7.3 The project connections

After selecting the component, we begin to test every component to check any problem then we found some problems:

- we test 2 motors and freewheel, but it couldn't move the robot because the robot is so heavy on 2 motors only

solve: we removed the freewheel and add 2 motors to make the movement more easy

- we found the left side slower the right-side motors the made the robot can't move straight

solve: we programmed the driver controller

- after testing the tablet on the robot that made the movement more hard

Solve: we added another 2 motors

At the end

- install the motors under the robot and connect the wires through the center hole to the driver (L298N Motor Drive)
- install the driver inside the robot on the center of the base
- install the Raspberry inside the robot to protect from any damage or lose any wire that could cause turn off any part or the robot at all
- install the battery between the driver and the Raspberry to connect them with electric
- we install the switch outside the robot to turn on or turn off easily
- the infrared sensor
- the Bluetooth module for connection with the tablet send and receive the orders

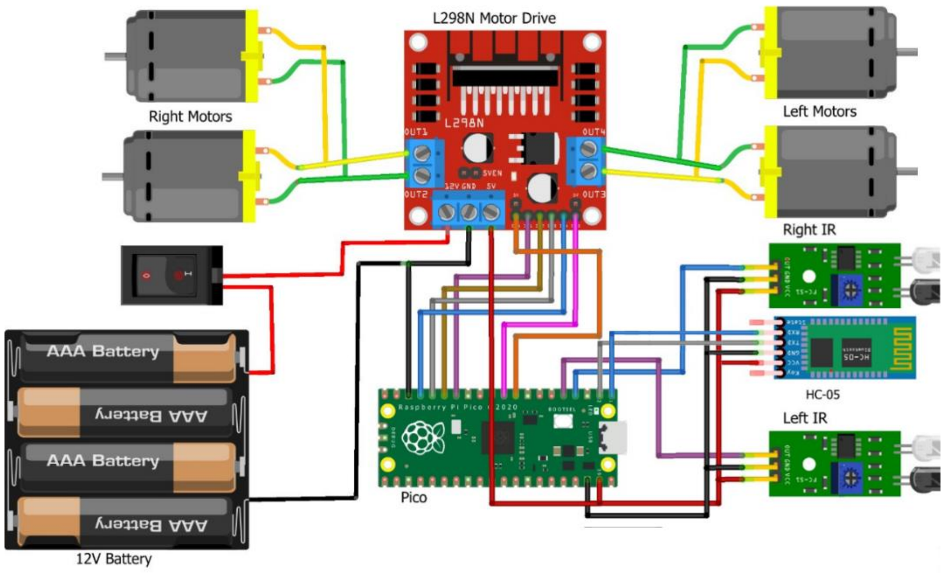


Figure 12: project connections

7.4 Respberry pi pico connection

How I get the idea of the connection:

I search on the internet to find how to connect the component we select it to use and I found a two pictures I can use

Use HC-05:

I found this connection. This connection is for controlling a car using Bluetooth module .. but don't need this only. So I needed another way of connecting most of the components.

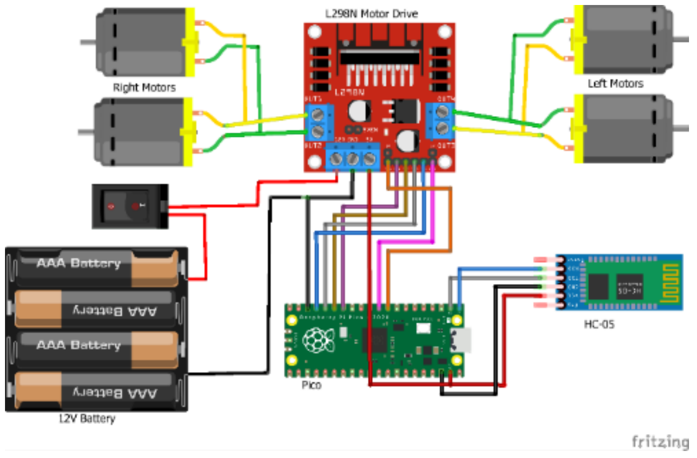


Figure 13 Use HC-05:

Using TCRT5000:

This photo is a connection of car robot to be a line follower. This way was useful but also not all what we need.

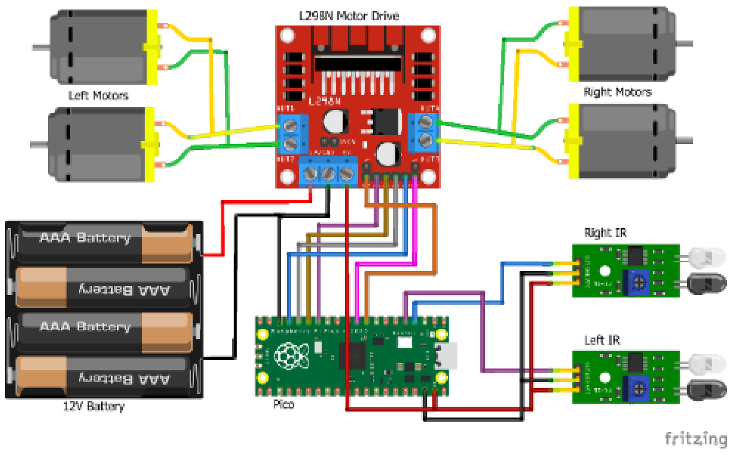


Figure 14 Using TCRT5000:

The new connection :

I decide to concatenate the two connection above and make the car to be line follower and also must be with Bluetooth to use it by sending a messages to it. The most popular application for the Bluetooth car is to make an application and control the car using the application. But when I talk with the software team I know that they want to use the car to be response by sound. So I found this is the most suitable thing to connect laptop with the robot.

How I concatenate the two connections above:

I decide to study all the ports on raspberry pi pico to found a way to concatenate the two circuits.

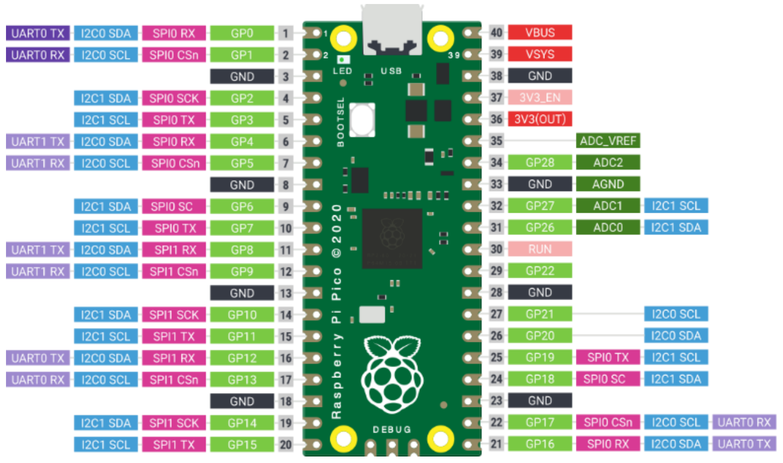


Figure 15 new connection:

7.5 What the pins on pico means:

There are 28 pins we can use it to in or out the date. These names are:
GP0, GP1, GP2, GP3, GP4, GP5, GP6, GP7, GP8, GP9, GP10, GP11, GP12, GP13, GP14, GP15, GP16, GP17, GP18, GP19, GP20, GP21, GP22, GP23, GP24, GP25, GP26, GP27, GP28.

Also there are 2 pins to output voltage which are:
VBUS, VSYS

Also, there are 6 pins as a ground.

7.6 What I do in my circuit:

Positive and negative
I connect the two terminal of the voltage and the ground to the basic positive and negative in the bread bored to use it in all components that use voltage easily

The control pins of motor driver
I connect the four wires from the output of motor driver with the pins: GP10, GP11, GP12, GP13
to control the four motors by the motor driver.

The voltage of the motor driver.
I connect the positive of the driver with the positive of the battery. And the negative with the negative of the battery and the negative of the negative in the bread bored and the output voltage (5V) with the VSYS in the raspberry pi pico

Connection of motor in the driver:
I split the wires of motor to the right side motors and the left side motor
I connect all positive terminals of motor with the right positive side in the driver (I get this by experiment) and I also do for the left side

Connect sensors :
The positive terminal of all sensors connected with the positive terminal in bread bored and the negative terminal of all sensors connected with the negative terminal in bread bored and the data terminal for right sensor connected with GP2 and the data terminal for left sensor connected with GP3

Bluetooth module HC-05:
The positive terminal of Bluetooth module connected with the positive terminal in bread bored and the negative terminal of bluetooth module connected with the negative terminal in bread bored and the TX terminal of the bluetooth module with GP0 and the RX with GP1

8.1 Conclusion: A Future Transformed by Bulbul - A Vision of Comfort and Connection

A Hospital Room Reimagined

Imagine a hospital room that feels less sterile and more like an extension of home. A comforting presence sits beside the bed – not a loved one, but Bulbul, the intelligent assistant system. Bulbul engages the patient in conversation, offering a friendly distraction from discomfort and loneliness. This is not a scene from science fiction; it's a glimpse into the future where Bulbul operates in hospitals worldwide.

A Network of Care

With Bulbul widely implemented, patients can expect a more holistic and supportive hospital experience. Imagine a network of Bulbul robots seamlessly integrated with hospital systems. Follow-up care becomes effortless, with Bulbul prompting patients about medication adherence and proactively identifying potential complications. Basic needs are fulfilled with ease, as Bulbul facilitates requests for food and water, empowering patients and reducing their dependence on busy nurses.

8.2 Continuous Development: Expanding Bulbul's Capabilities

The future of Bulbul holds even greater promise. Here are some potential advancements to consider:

- **Advanced Emotional Intelligence:** Bulbul's ability to recognize and respond to a patient's emotional state could be further refined, allowing it to offer tailored support and de-escalate anxieties.
- **Integration with Medical Records:** Secure access to patient medical records would allow Bulbul to provide personalized care instructions and medication reminders.
- **Multilingual Support:** Expanding Bulbul's language capabilities would ensure a wider range of patients can benefit from its companionship.

8.3 A Brighter Future for Patients

Bulbul's potential to revolutionize patient care is undeniable. By fostering companionship, providing proactive support, and empowering patients, Bulbul can transform the sterile hospital environment into a space that promotes healing, reduces recovery times, and improves overall well-being. As Bulbul continues to evolve, we can look forward to a future where every patient experiences a more supportive and connected journey to recovery.

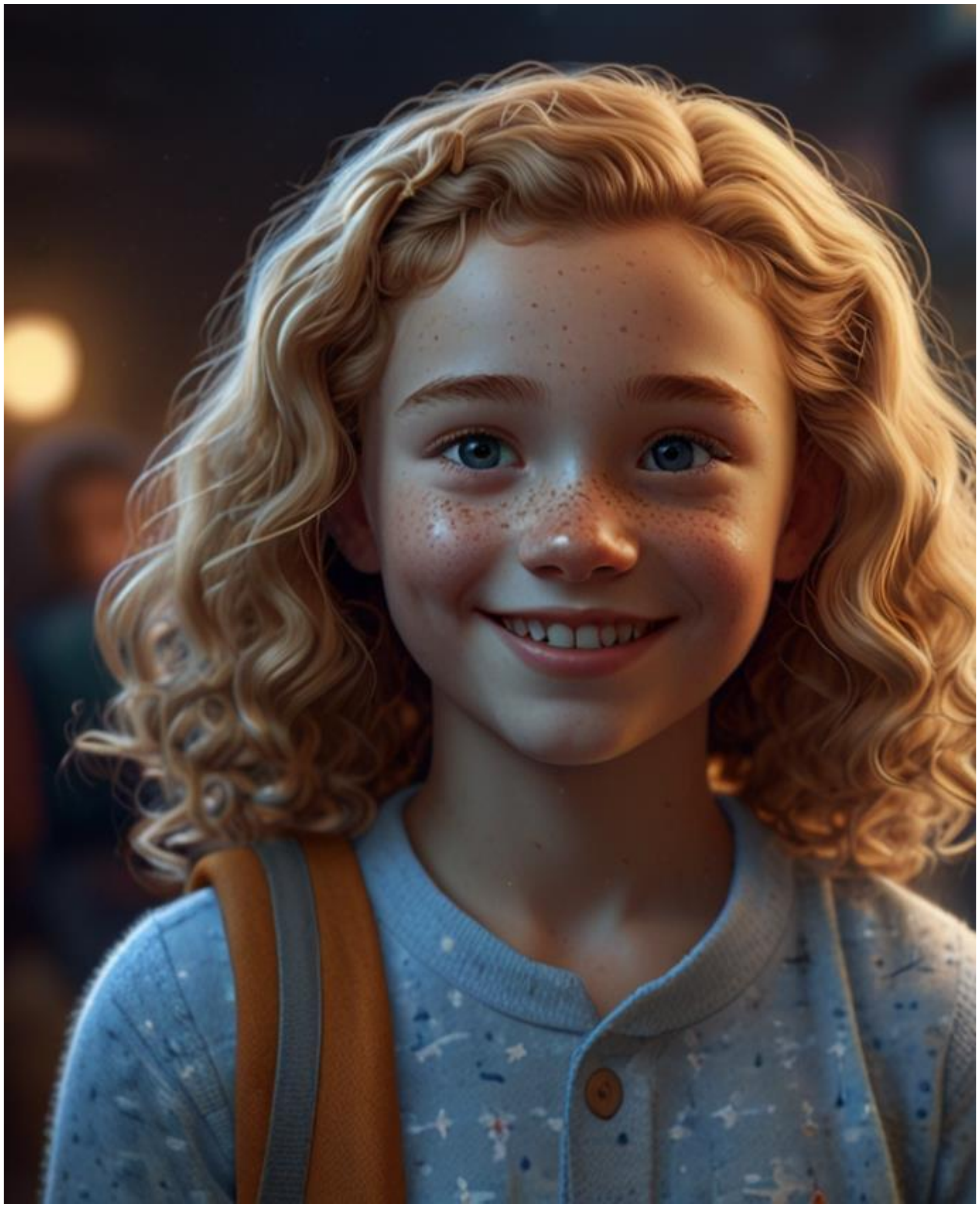


Figure 16: The patient’s feeling after applying the bulbul robot in hospitals

References

9.1 Research Papers:

- **"The Role of Social Interaction on Patient Outcomes: A Review of the Literature"** by Kathleen M. Sodergren in *Journal of Nursing Scholarship* (2001): https://journals.sagepub.com/doi/10.1177/1948550613502990 This paper explores the well-established link between social isolation and poorer patient outcomes.
- **"Robots for Enhanced Healthcare Delivery: Ethical Considerations"** by Matthew A. Chapman et al. in *The American Journal of Bioethics* (2016): https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9989998/ This article examines the ethical considerations surrounding the use of robots in healthcare, offering a framework for responsible development and implementation.

9.2 News Articles:

- **"Robots Are Coming to Hospitals, and They're Here to Help"** by Tara Parker-Pope in *The New York Times* (2018): https://www.nytimes.com/2022/10/07/opinion/machines-ai-employment.html This article explores the growing trend of robots being used in hospitals for various purposes, including patient care.
- **"Can Robots Help Reduce Hospital Readmissions?"** by NPR (2019): https://www.npr.org/2022/05/24/1100985010/are-robots-the-solution-to-understaffed-nursing-homes This NPR story discusses a pilot program using robots to help patients manage their care after discharge, potentially reducing readmission rates.

9.3 Websites:

- **The National Institutes of Health (NIH) - Robots in Healthcare**: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9589563/ This NIH webpage provides a general overview of robots used in healthcare settings, including potential benefits and challenges.
- **The Robot Report - Healthcare Robots**: https://www.therobotreport.com/ This website offers news and information specifically on robots designed for use in healthcare, including robots for patient interaction and support.