Diabot: a friendly telepresence robot

Davide Nesi
UniFi
davide.nesi@stud.unifi.it
davide.nesi@stud.unifi.it

Abstract

The proposed system allow users to experience real telepresence on a human sized bot. The system is made to be inexpensive and easy to reproduce. The field of application of telepresence is very wide, I narrowed my research to social and security applications. I decided to make the system as flexible as possible to allow further modifications. Conducted tests show how exiting and fun to use this system is and how it can be improved.

1. Introduction

Diabot is a telepresence robot born from the necessity to keep in touch with friends living far from home and old friends. The name of the robot comes from the contraction of Diana (the first person to have tested the system) and robot. The major goal for this project is to provide a system that can be easy to build and use, providing both hardware and software simplifications. All hardware components are easy to find and assemble and most of the parts are 3D printed but can be manually crafted too. All parts are available as 3d object file in the project repository at github.com/DavideNesi/diabot. Software wise the system is using a combination of technologies in order to provide a very integrated user experience. Flask API and WebRTC are the most important tools used to provide a standard interface. The user is provided with an interface integrating both the video call and the robot control panel. From the webpage interface the user can connect to the robot using an API and automatically join a WebRTC call on a supported smartphone mounted as the head of the telepresence robot. We will later discuss all implementation details.

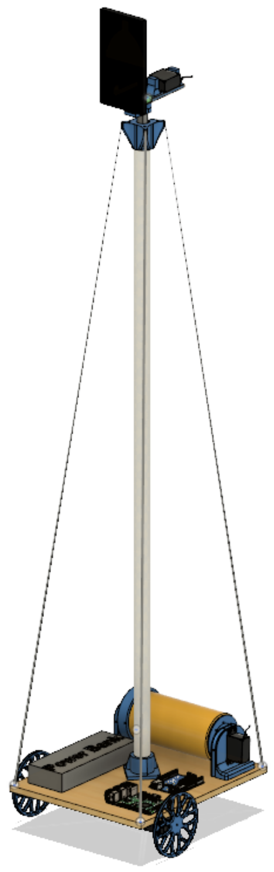


Figure 1. Rendering of Diabot

2. Personas

Telepresence, even if restricted to only social and security field, has a very wide applications field. For this reason I decided to narrow the research to two main areas that best fits my experience and I feel confident this project can be applied to.

2.1. Social interaction scenarios

- Gianna is an exchange student and she's studying in London for 10 months. She has a lot of new friends but she feels she's missing home. She could use Diabot to take part in birthday parties and other social occasions.
- Anselmo is a grandfather and he rarely see his grand-

sons and granddaughters. He lives in a nursing home far from the city his family lives. His grandsons and granddaughters can use Diabot to check Anselmo at his nursing home. He can use Diabot too (with some help from friends and family) to visit his house.

2.2. Security scenarios

- Patrizia is a business owner and sometimes she just want to check her office. She would like to interact with employee too. She already has a video surveillance system installed in her factory but she wants more flexibility. She can use Diabot for all her needs.

- Roberto owns a small company and he thinks the video surveillance system he has installed is not enough. He can program Diabot to do a patrol and record and/or process all the audio video stream searching for anomalies.

2.3. Implicit system requirements

From the previous subsection we can see how it is strongly needed a flexible but simple way to use system. In particular we need two interfaces: one to the most naive user, graphic, easy to use and intuitive, the other to expert user willing to explore and bring the bot out of standard applications. From the two areas of application proposed it is easy to see how the social interaction scenarios are much closer to real case applications and quite easy to see in near future. For this reason the social aspect of this project is the one most of the research was done. The security and patrolling scenario influenced some of the technical choices made, such as the decoupling of web interface and the robot using an API layer.

3. Needfinding

After searching for telepresence applications and research articles the next step was to search users needs. For this phase I decided to use Google Form and ask exchange students to compile a set of questions. The result obtained from this phase showed clearly that students living abroad often feels lonely and that even the family would have big beneficial effects from a closer contact. Apart from this not very surprising result the form showed how the vast majority of users are familiar with video games and most of them feel unconformable or weird during a video call. This two points have big repercussion on the project: the familiarity with computer games can be used to introduce a control system already used in other context by the users and the negative feels of video calls is important in relation to final user experience valuation of Diabot. In general from this phase it is quite clear how this project could help the researched users group.

4. Implementation

In this section we will build Diabot from the ground up, going from hardware components to software implementation. In this process I'll discuss the implementation choices made during the prototyping phase. All parts used in the system are available in the project repository (links and stl files).

4.1. Initial prototypes

During this project I've experimented with different implementation choices. The first iteration of this project was powered only by Raspberry. In that configuration RaspberryPi was connected to a small display, to a webcam, a small speaker and to Arduino. The load on Raspberry was too high and the max fps of the system was around 2. This problem with Raspberry shown how it can not be used to stream bidirectional audio video over WiFi. So I decided to just handle the video call problem using a separated device. The most reasonable way was to use an old smartphone. Most of recent but considered old smartphones are more than capable to support a video call over WebRTC. Both tablets and augmented reality visors available in laboratory were experimented with this system. The available tablet was very heavy and did not support the WebRTC framework used for video calls. The augmented reality glasses (Optinvent Ora2) are not suited to bidirectional audio video streams.

4.2. Hardware

In this section I'll show all components of the proposed system. All components used in Diabot are quite common and easy to assembly.

4.2.1 Frame

The frame used to build Diabot is composed by common material and jointed by 3d printed parts and screws. The main body where most of the components are fixed is a MDF board of 30 cm x 40 cm. The telescopic mechanism is simply a squared tube inside a normal tube with a 3D printed adapter.

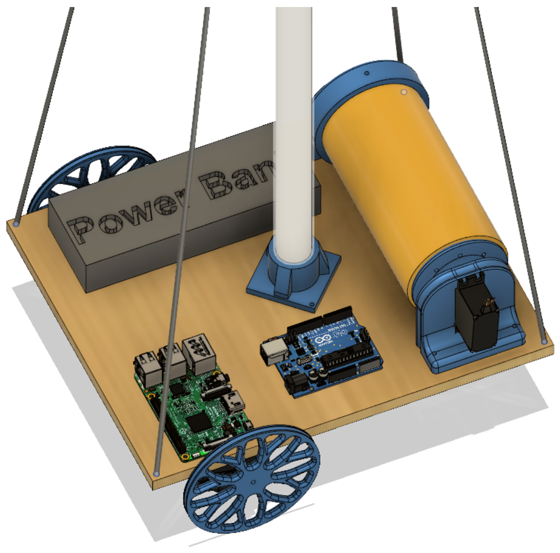


Figure 2. Detail of Diabot with component placing

4.2.2 3D printed parts

The frame is kept together by 3D printed parted parts. The head piece is 3D printed and easy to modify for multiple smart phones. Doing so the user can use an old smart phone as video call platform. The height regulation mechanism is made from 3D printed parts too.

4.2.3 Power

The power for the whole system comes from a standard powerbank. The powerbank needs to have at least two USB ports and at least 2A of discharge rate in order to work as main power source for Diabot.

4.2.4 Motors

Diabot uses 4 motor: 2 to move in space, 1 to extend and retract the telescopic axis (sitting and standing up heights) and 1 to pitch the head, used as a neck. This setup allows great freedom of movement. I decided to use servomotors because they are very cheap, easy to use and directly compatible with Arduino.

Figure 3. Servomotor

There are different types of servomotors and I decided to use 5V ones because they are common and easy to power. In the system I used 3 continuous rotation servos and 1 positional (neck position).

4.2.5 Telescopic mechanism

The telescopic mechanism allows the bot to change the camera height. This feature is implemented in the bot to simulate difference height of a person sitting and standing up. The mechanism is implemented with a rope winding and unwinding on a servocontrolled cylinder. The other end of the rope is linked to the inner square tube of the mechanism. When the rope wind around the cylinder the inner square tube rises.

4.2.6 Endstops

To protect the system and have a better control over the telescopic sitting and standing up mechanism I decided to use 2 endstops. The decision to have a closed loop over the actuation of height came from observation of the most critical aspects of the system. Without endstops the height actuation was not reliable enough.

4.2.7 Arduino

Arduino is the spinal cord of Diabot. It receives orders from Raspberry Pi using the serial usb port and it sends signals to servomotors according to the order received. It also collect the end stops state signal and actuate the height servomotor accordingly.

4.2.8 Raspberry

Raspberry is the brain of the whole system. It is connected to Arduino using a standard USB cable. This allows Raspberry to communicate using serial protocol with the rest of the robot.

4.3. Software

During the implementation of this project I experimented different software solutions. The first doubt I had was about the video call service to use. Telepresence is not a new born field and I had the opportunity to look around and understand what is typically used in this situations. Excluding proprietary embeddable solutions the most used video call service is Skype. This implementation choice makes the hole system less compact and integrated. I decided to use WebRTC and explore a control and video call integrated GUI. Given the requisites of flexibility and the different use case scenarios I decided to decouple the bot interface and the main web interface. Doing so I was able to both have an easy to use and always available GUI service and an advanced user interface using an API. The most important implication of using WebRTC is that every smart phone produced in the last 5 years is suitable to be used as Diabot primary human interface module. This design choice lowered the costs, assuming the common user has an old smart phone laying around.

4.3.1 Arduino

The robot runs on Arduino and Raspberry Pi. Arduino was programmed to check the end stops and execute order from the serial port connected to Raspberry. The most important thing programming Arduino for this application is to reduce the serial communication introduced lag.

```
void setup() {
  Serial.begin(9600);
  Serial.setTimeout(50);

  leftServo.attach(LEFT_SERVO_PIN);
  rightServo.attach(RIGHT_SERVO_PIN);
  neckServo.attach(NECK_SERVO_PIN);
  heightServo.attach(HEIGHT_SERVO_PIN);

  pinMode(UPPER_ENDSTOP_PIN, INPUT_PULLUP);
  pinMode(LOWER_ENDSTOP_PIN, INPUT_PULLUP);
}
```

Figure 4. Arduino fragment including serial delay reduce

For this reason I had to add a specify serial timeout for the communication with Raspberry Pi. The command arrive to Arduino as a string and it has to be parsed. After parsing the command the servos are actuated accordingly. To control servomotor I used the Arduino servo library. Continuous and positional servos are different as working mechanism but identical for Arduino: they both receive a 0 to 180 value. Positional servos use the 0-180 as the position of the shaft rotation relative to the servo body, continuous servos have 90 as hold position, 0 as full speed clockwise and 180 as full speed counterclockwise.

4.3.2 Flask

Arduino is the spinal chord of the robot but RaspberryPi is both brain and ears. RapsberryPi is connected to Arduino with a standard USB cable and it is connected to a WiFi. Raspberry runs a Flask script running an API.

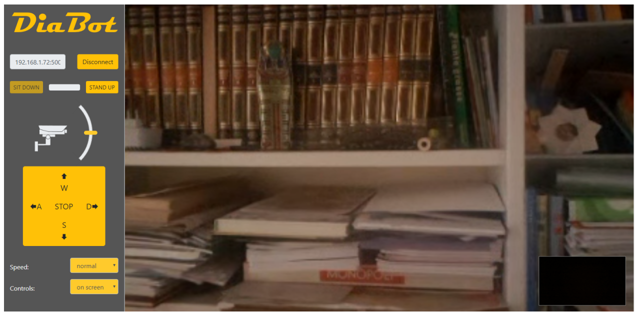
```
@app.route('/status', methods=['POST'])
def status():

        if not request.json or not 'order' in request.json:
                return jsonify({'order':"not_valid_format"}),201

        r = re.compile('.*;.*;.*;.*')
        receivedOrder = request.json.get('order',"")
        if r.match(receivedOrder) is None:
                return jsonify({'order':"not_valid_format"}),201

        v = receivedOrder.split(';')
        if not (0<=int(v[0])<=5 and 0<=int(v[1])<=5 and
                0<=int(v[2])<=5 and 0<=int(v[3])<=2):
                return jsonify({'order':"invalid_parameters"}),201

        #left,right,neck,height
        #L[1:5] R[1:5] N[1:5] H[1:2]
        lastOrder = receivedOrder
        ser.write(lastOrder.encode())
        result = {
                'received': receivedOrder,
                'status': 'executing'
        }
        print(result)
        return jsonify({'order':result}),201
```

Figure 5. Flask API fragment showing POST

The POST method to submit a command to Flask API is shown in Figure 5. It was necessary to filter out bad requests and not formatted request's bodies. The most important piece of code here is the serial write to send the order received and checked to Arduino. RaspberryPi always assigns serial port named ”ttyACM0” to Arduino, so the code forward the command to that port. Other endpoint implemented in Python are just used to be sure the robot is ready before starting to send commands. The script starts when RaspberryPi boots up, so usually it takes at most a ten seconds.

4.3.3 Web page

The webpage is the real end user interface. This interface is the most important one in the entire project. The main



Speed: normal

Controls: on screen

Figure 6. Interface while connected to a robot

goal here is to allow the user to smoothly control the robot without losing precious screen space for the video call. In order to do this I decided for a clean and essential look that can efficently deliver all the informations needed to control the robot to the user.

As we can see in Figure 6 the whole interface can be divided in two sections: the dashboard and the video call area. The dashboard is on the side to be less intrusive. The video call area is in the center of the screen to maximize the scene immersion. Developing this project the need to have a full video call service was clear. In order to avoid to develop a brand new video call service or the set up of a signaling server we decided to use WebRTC as provided from appr.tc website. This website is an example on how to use WebRTC, in this application it became the video call engine.

```
function standingUp() {
    sendOrder("3;3;0;2");
    document.getElementById("standBtn").disabled = true;
    document.getElementById("sitBtn").disabled = true;
    document.getElementById("forwardBtn").disabled = true;
    document.getElementById("leftBtn").disabled = true;
    document.getElementById("stopBtn").disabled = true;
    document.getElementById("rightBtn").disabled = true;
    document.getElementById("backwardBtn").disabled = true;
    document.getElementById("controlsSelector").disabled = true;
    document.getElementById("speedSelector").disabled = true;
    $("#arc-slider").roundSlider("disable");
    setTimeout(reenableAfterStanding, 3000);
    animateProgressBar(1);
    function reenableAfterStanding() {
        document.getElementById("standBtn").disabled = true;
        document.getElementById("sitBtn").disabled = false;
        var e = document.getElementById("controlsSelector");
        var selectedControl = e.options[e.selectedIndex].value;
        if (selectedControl != "keyboard") {
            document.getElementById("forwardBtn").disabled = false;
            document.getElementById("leftBtn").disabled = false;
            document.getElementById("stopBtn").disabled = false;
            document.getElementById("rightBtn").disabled = false;
            document.getElementById("backwardBtn").disabled = false;
        }
        document.getElementById("controlsSelector").disabled = false;
        document.getElementById("speedSelector").disabled = false;
        $("#arc-slider").roundSlider("enable");
    }
}
```

Figure 7. Standing up command handling

The dashboard is composed by different items. Buttons and selectors are the most used widgets. The robot is quite light and moving very fast while changing the height position of the robot may cause instability. For this reason I added animations and delays to the height actuation. Doing so the user has a clear feedback of the action requested and it was immediately clear that during that animation no other action can be performed. The bot takes at most 3 seconds to stand up. So I decided to set the animation duration to 3 seconds. In Figure 7 is shown the standing up handling function, implementing all the animation we discussed above.

5. User test

User test was conduced at MICC. The pool of available users was very fitting with the possible end user and adopter of the system. All participants were computer enthusiast and most of them were gamer or at least familiar with moving an avatar inside a virtual environment. In order to keep the test as short as possible the form used in the test is quite short. I left however free suggestion fields on the end. I need to thank all MICC for participation and patience.

5.1. Protocol

The protocol was designed to test the web interface, the robot controls and social interactions using the video call service. The test was split in different phases:

1. Introduction: the user is presented with the system. The user is required to perform single actions without guidance on how to perform them.

2. Timed course: the user is required to complete a course in the least time possible

3. Quest: the user is require to investigate on a document. The research will require the user to autonomously interact with an actor and to stand up to find the document on a shelf.

4. Timed course: the user is required to complete again the same course.

5. Form: the user is required to complete a form regarding the experience with the system

6. Reward: the user is rewarded with candies.

The whole test takes around 10 minutes to complete. The timed course gives a measure of the confidence of the user controlling the robot. For this reason it is repeated 2 times: doing so I can measure how much few minutes with the system can actually improve the confidence in controlling the robot.

N	Question	Mean	
1	I found the system experience interesting	1.7	1.9
2	Some guidance on how to use the bot was required	4.1	1.9
3	I found easy to control the robot	2.1	1
4	Interacting with people was difficult	6	1.4
5	I found the interface intuitive	2.6	1.6

Table 1. Form results (1:strongly agree, 7:strongly disagree)

N	First timed test	Second timed test	Percentage difference
1	125	84	-33
2	103	75	-27
3	142	138	-3
4	183	160	-13
5	146	64	-56
6	67	72	7
7	170	105	-38
Mean value	134	100	-23

Table 2. Timed test results

5.2. Testing process

The user test protocol allowed the user to express them self and enjoy the experience. The timed course and the quest are very motivating tasks and all the users was engaged. I was able to conduct seven tests.

5.3. Results

The users started the test with a small introduction to the system. The main goal of this preliminary phase was to understand if the interface was intuitive enough. The user was asked to change the height of the robot, follow an object with the camera and to move in specific directions without indications on how to do it. In this phase all the user was able to perform at least two of the requests without any help. The two major issues I found in this phase were: the height regulation feature was not something expected and the inclination slider for the camera was very often only dragged and never released. The timed lap was a very enjoyable experience for all the users. Some of them really felt the need to do the absolute best time and that gave them a great motivation.

In Table 2 the results show how almost all the users got a better timing in the second test. The mean improvement was -23% from the first timed test. This result is very positive because it shows how even few minutes with with the systems can really make a difference on how comfortable the user is to use it. The quest was completed by all users and it was a very good test bench on complex interaction with both environment and people. The form results were very interesting. One of the most requested feature was to have a wide angle camera on the robot to better perceive objects and generally have better camera and/or internet connection. This request was often together with the request to have a more flexible neck joint. The user very often preferred the keyboard controls over the on screen controls. A very good suggestion proposed by users was to map the head pitching control to the mouse scroll wheel. The overall level of satisfaction and enjoyment after the test was very high.

6. Improvements

After the user test phase and various hours of usage I found different points where the system can be improved. In this section I will cover all the possible modifications and their impact on the system.

6.1. Hardware

The most requested modification is to change the type of camera lens. A possible solution for this problem is just use a smartphone with an integrated wider lens on the front facing camera. But it can be quite expensive. The best solution for this problem is probably just use a clip lens for smartphone. They are inexpensive and they work on almost every smartphone.

Figure 8. Clippable smartphone lens

The maximum angle of movement of the neck can be easily modified off centering the servomotor responsible for that actuation. Other hardware improvements are about the quality of video call and about the robot clothing. The quality of the video call was outside our research area, however it never was an obstacle for requested tasks. Maybe Raspberry could be replaced with something capable of withstanding the 2 way bidirectional audio video stream. Doing so the whole project would become more expensive and complex to build, but changing the whole video call engine the whole system could be even more integrated.

6.2. Software

After the user tests the preferred implementation of the control was the one using the keyboard to control the robot. The most requested feature was to map the camera inclination to the mouse scroll wheel or to ”+” and ”-” on the keyboard. This is a very simple modification of the web interface. The on screen buttons were for some users counter intuitive at first. Some users expected to hold the button pressed down to keep moving. Most of them found the two control system (on screen and keyboard) complementary.

7. Conclusions

The proposed system allows the user to experience the telepresence tools and mechanism in a simple environment with an easy to use and integrated interface. The user tests were all positive and nobody was frustrated by the usage of the bot. Actually all the user enjoyed the experience a lot. Thanks to low cost components and open source hardware and software the project is as simple to build as it can get. I am satisfied with this project. The possible improvements are minor changes to the system and this means that even the prototype realized up to this point already has all the key features desired implemented in the right way.