MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE

SUMYSTATEUNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE

BACHELOR THESIS

On the topic

“Telepresence Robot”

Head of Department	**Dovbysh A.S.**
Student Group IH-65aн	**Bananga L. A.**
Supervisor	**Oleksiienko G.A.**

SUMY 2020

MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE

SUMYSTATEUNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE

Approved ____________

Head of department Dovbysh A.S.

"_________________” 2020

Task of Bachelor Thesis

Fourth-year student, group IN-65ан specialty "Informatics" Leeroy A Bananga

Topic: “Teleprsence Robot”

Approved by order of the Sumy State University

№________ of _________ 2020

Contents Explanatory Note: 1) An analytical review of the literature; 2) Statement of the problem; 3) The choice of methods for solving the problem; 4) Develop an information web resource; 5) Conclusions.

Date of issuance of the task	"___________" 2020
Supervisor	Oleksiienko G.A
Task adopted to be implemented by	Bananga L. A.

ABSTRACT

Note: 40 pages, 11 figures, 17 sources literature, 1 app.

Object of study – Intelligent Robotic System

Purpose – Developing a simple telepresence robot

Research methods – system, deductive and inductive analysis, computer experiment

Results – A working robotic system was developed which implements Virtual Reality functionality in it.

KEY WORDS

Telepresence robot

Telepresence

Raspberry pi

Python

Content

Contents

INTRODUCTION

Before technology came to be people faced a lot of limitations without it. But as time progressed people have become dependent on technology to an extent that they cannot or it’s difficult for them to live without technology. Today we see many technological achievements being achieved from allowing computers to learn on their own without explicitly being programmed (A.I), to self-driving vehicles becoming a norm. All this was not possible before our time say about 40 years ago. Technology has taken the human race to the next higher level, but nothing has taken human race to the advanced level of technology like robotics. In this era we see robots replacing many of the tasks which might be mammoth to a mere human being. We are seeing robots producing some of our everyday necessities, robots building cars and even to the extent of participating in research concerning other celestial bodies in our galaxy.

Telepresence is an experience which allows users to feel like they are physically in a remote location. Telepresence uses technology which is related to virtual reality. With the use of mediums such as video cameras, sensors, microphones to substitute for the actual human senses of the user. Telepresence technology can be used to integrate virtual things into the real environment for example holograms.

A Telepresence robot uses this technology to operate. A telepresence robot is a wheeled remote controlled device that has wireless connectivity capabilities usually Wi-Fi which allows it to be remotely controlled by the user and provide telepresence experience to the user in the process. The robot uses a tablet or a camera and speakers to provide video and audio functions. Telepresence robot are used in different industries and departments around the world. A telepresence robot can be used to stand for a teacher, the robot will move around the room and interact with students as if the teacher was physically present. Telepresence robots can be used in a scenario where a worker who is disabled or their physical location stops them from physically attending work but they can still have physical presence through the use of such robotic systems.

1. INFORMATION REVIEW

1.1 REVIEW OF KNOWN SOLUTIONS

The first similar solution is The InTouch Vita, also known as the RP-Vita, is a remote presence robot developed by iRobot and InTouch Health, and is assisting medical professionals in the exchange of information in healthcare environments and enhanced service to patients. Already being used in several hospitals, the Vita robot is helping to revolutionize the healthcare industry by allowing medical staff to monitor and advise patients from remote locations.

In addition to letting specialists advise colleagues and engage with patients and doctors and nurses, the RP-Vita can be used to access recorded information via data ports and can be connected to ultrasound imaging machines, digital stethoscopes, and more. The Vita can transmit information to doctors from almost any type of medical device that has USB connectivity.

The unit has a simple-to-use iPad interface; and its automatic docking function ensures that it stays charged in case of emergencies. The Vita is equipped with automated navigation functions to allow for easy mobility and safe travel without needing user guidance. This makes it suitable for retrieving information from various patients autonomously. Capable of retaining and processing enormous amounts of information, the Vita will facilitate doctors with data storage and retrieval, and assist medical professionals with assessing and recording information. Additionally, it uses the cloud-based SureConnect, which helps to maintain a reliable network connection.

The second solution most probably the commonly know is the Mars Exploration Rover, either of a pair of U.S. robotic vehicles that explored the surface of Mars from January 2004 to June 2018. The mission of each rover was to study the chemical and physical composition of the surface at various locations in order to help determine whether water had ever existed on the planet and to search for other signs that the planet might have supported some form of life. https://telepresencerobots.com/robots/intouch-health-rp-vita

RP-Vita By iRobot

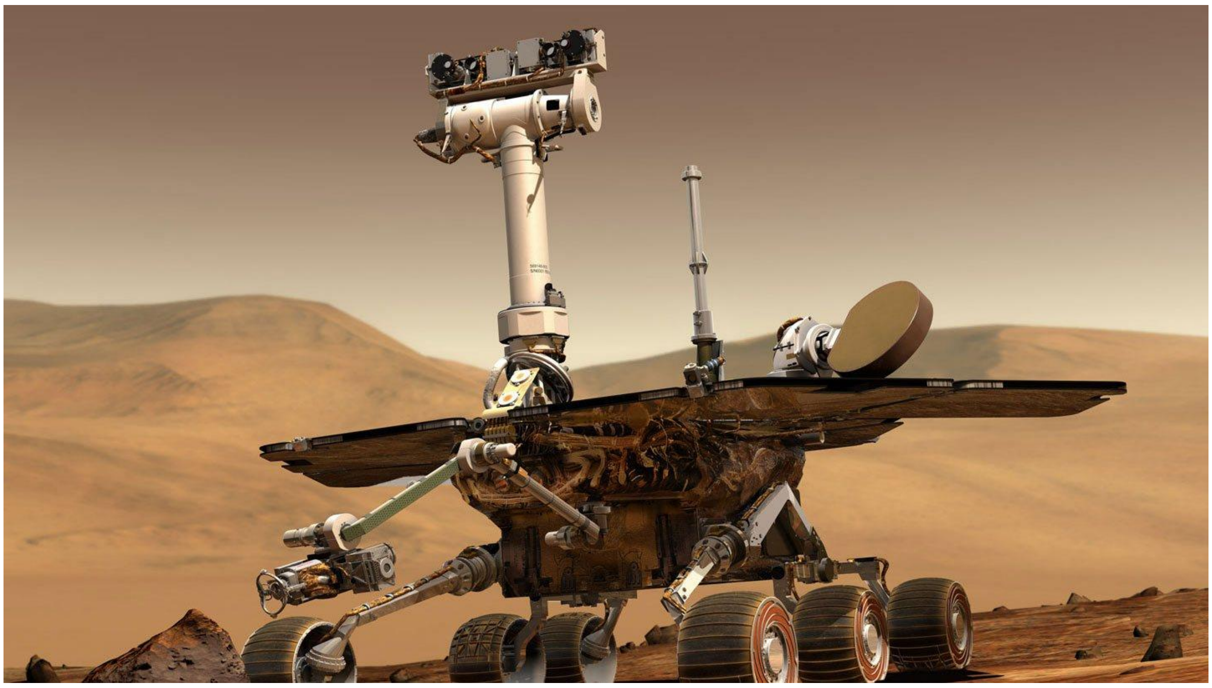
The twin rovers, Spirit and Opportunity, were launched on June 10 and July 7, 2003, respectively. Spirit touched down in Gusev crater on January 3, 2004. Three weeks later, on January 24, Opportunity landed in a crater on the equatorial plain called Meridiani Planum, on the opposite side of the planet. Both six-wheeled 174-kg (380-pound) rovers were equipped with cameras and a suite of instruments that included a microscopic imager, a rock-grinding tool, and infrared, gamma-ray, and alpha-particle spectrometers that analysed the rocks, soil, and dust around their landing sites.

The landing sites were chosen because they appeared to have been affected by water in Mars's past. Both rovers found evidence of past water; perhaps the most dramatic was the discovery by Opportunity of rocks that appeared to have been laid down at the shoreline of an ancient body of salty water.

Each rover was designed for a nominal 90-day mission but functioned so well that operations were extended several times. NASA finally decided to continue operating the two rovers until they failed to respond to commands from Earth.

In August 2005 Spirit reached the summit of Husband Hill, 82 metres (269 feet) above the Gusev crater plain. Spirit and Opportunity continued to work even after a significant Martian dust storm in 2007 coated their solar cells. Opportunity entered Victoria crater, an impact crater roughly 800 metres (2,600 feet) in diameter and 70 metres (230 feet) deep, on September 11, 2007, on the riskiest trek yet for either of the rovers. On August 28, 2008, Opportunity emerged from Victoria crater and set off on a 12-km (7-mile) journey to the much larger (22 km [14 miles] in diameter) Endeavour crater. https://mars.nasa.gov/mer/

Mars Exploration Rover



The third solution is the Double 3 which is a self-driving two-wheeled video conferencing robot from Double robotics . Double 3 enables telecommuters, remote workers, and students to feel more connected to their colleagues by giving them a physical presence where they can't be present in person.

The Double 3 has an array of 3D sensors which enables Double 3 to understand its environment, where it's safe to drive, and how to divert around obstacles to reach the destination. Obstacle avoidance means that completely untrained drivers can drive Double 3 without fear of bumping into walls or people.

1. Dots are drawn on the floor where Double 3 can safely drive. The driver can click anywhere on the floor and the robot will go there, avoiding obstacles along the way. Its two 13 Megapixel cameras provide an ultra wide field of view and multiple levels of zoom. A gearmotor enables both cameras to physically tilt up and down, using an advanced software algorithm it combines all of the movement into one seamless user experience. An advanced array of six microphones helps the driver hear people from far away and with less background noise. https://www.doublerobotics.com/

Double 3 by Double robotics

1.2STATEMENT OF PROBLEM

A robot is to be made which has wireless capabilities. The robot should be able to be controlled wirelessly by any means (remote or computer commands).

The robot will be equipped with a camera to provide a video feed of where it is. This video feed is to be streamed from the robot to a website where the user can connect and get the live video feed of the robot camera. The mounted camera must be able to move so that the robot pilot can look around the environment from which the robot is present, where ever they are using their head movements, meaning when they turn their head to the left the camera should face left as well and when they turn their head to the right the camera should face to the right as well. The same is to be done for looking up and down.

A virtual reality headset can be used to provide the pilot with a visual aid and to make them feel as if they are physically present where the robot is.

2. SELECTION OF INSTRUMENTS OF SOLVING

2.1 SELECTION HARDWARE

L298N motor driver is used to drive the 2 DC motors which turn the robot wheels for the robot to maneuverer around. This driver can control both spinning direction and speed of the wheels. Pulse-Width Modulation (PWM) is used to control the speed of the wheels and an H-bridge for controlling rotation directions (forward or backwards).

Pulse Width Modulation is a way of getting an analog result using digital ways. Digital control is used to toggle the signal between on and off and the on off pattern can replicate voltages between 5 volts which is on and 0 volts which is off. Various analog values are aquired by changing the pulse width which is the duration of the on time.

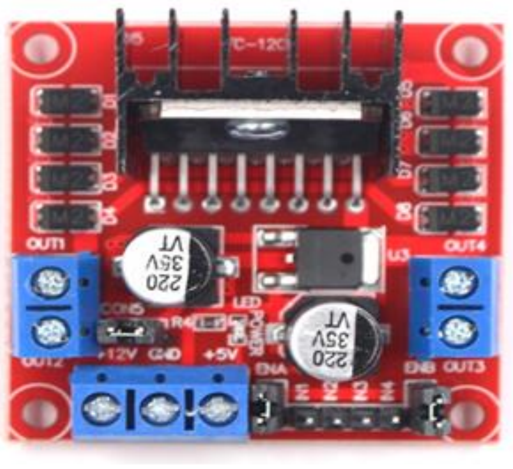


Fig 1.1– L298n motor driver

Pulse width modualation allows us to change how long the signal is HIGH in and analog type of way.

The speed of the motors can be controlled by varying the input voltage and Pulse Width Modulation is a common technique used for doing this. It is a technique where average value of the input voltage is adjusted by sending a series of ON-OFF pulses. The average voltage is proportional to the width of the pulses known as Duty Cycle.

When the signal is high it's called "on time" and to describe the amount of on time the duty cycle concept is used. Duty cycle is measured in percentage, the higher the duty cycle, the greater the average voltage being applied to the dc motor (High Speed) and the lower the duty cycle, the less the average voltage being applied to the dc motor (Low Speed).

By changing the polarity of the voltage, the spinning direction of the wheels can be changed and a common way to do that is through the use of an H-bridge found on the L298N motor driver. It has 4 switches with a motor in the middle. Closing 2 particular switches at the same time reverses the polarity of the applied voltage this causes a change in the spinning direction of the motor.

Raspberry Pi 3 model B+ as the brains of the robot was used because as it is, it is a complete fully functional computer with input and output capabilities and a processor as well. Raspberry Pi 3 model B+ has Wi-Fi and Bluetooth modules in-built in the computer which will make it easy for us to use the robot over Wi-Fi and send commands to the robot and there won't be need for us to look for a separate Wi-Fi module as it comes with the raspberry pi. Also raspberry pi is ideal for the project as there is need for a powerful processor with minimum power requirements to work with the robot and raspberry pi 3B seem to meet the requirements as its processor is powerful enough to process the robot commands and requires minimum power [1].

The Raspberry Pi uses the "Cortex-A53" processor which implements the ARMv8-A 64 bit instruction set. The ARMv8-A architecture include 64 bit data processor, a 64 bit general purpose registers and an extended virtual addressing.

The computer also has a 1GB LPDDR2 SDRAM

2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE. These network interface cards will allow us to control the robot using an Xbox pad connected to the computer via Bluetooth. The pad will be used to control the wheels of the robot. The Wi-Fi will be used to connect and communicate with the robot using SSH to start the necessary programs for the robot to operate.

CSI camera port for connecting a Raspberry Pi camera which will be used to provide a video feed for the robot.

The Debian operating system Ubuntu mate which is installed on a micro SD card to work with the raspberry pi is open source hence problems encountered will be easy to solve as there will be many solutions available. Raspberry Pi model 3B has 40 pin extended GPIO to enhance our real world project.

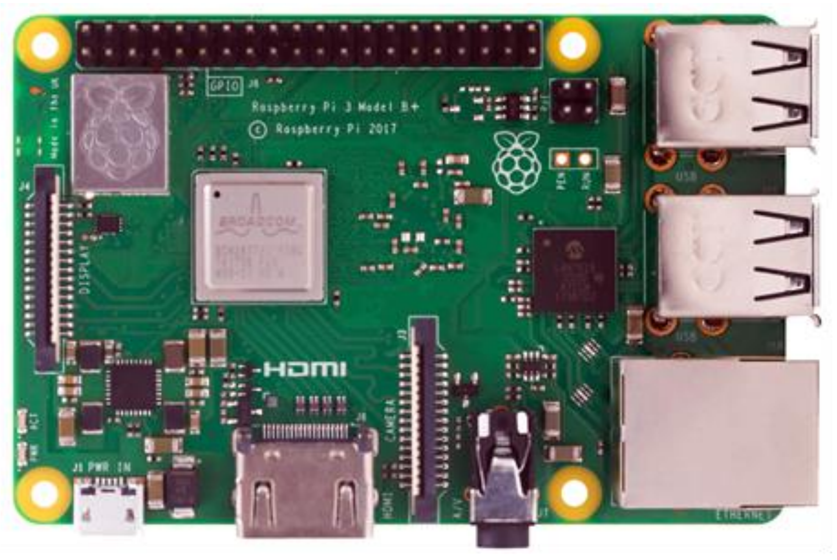


Fig. 1.2– Raspberry pi 3 B+

Virtual Reality headset. A make shift Virtual reality headset will be used to control the camera direction through head movements. A cell phone will be fitted in the headset so that we use the accelerometer readings to control the camera remotely with head movements.

Servos. The robot uses two Micro Servo motors to control the camera pan, one to turn the camera left and right and the other one to tilt the camera up or down. Data from the cell phone accelerometer is going to be used for the movement of the two servos based on the head position. The two servos will be connected to the GPIO of the robot computer.

Since the raspberry pi has Wi-Fi it will be possible to communicate with the servos using a Wi-Fi connection. A Java Script was developed to collect the accelerometer readings or the orientation of the connected device to control the two servos which in turn will change where the camera is facing. An application called flask which will be explained more in detail was used as a server to serve the site from which a device will connect and if connection is successful the device will be able to control the two servos.

One of the two servos gets its data from the x axis of the connected device and the other servo will get its data from the y axis of the device. The combined movements of the servos simultaneously allows the robot to look around the environment from which it is in.

The option to use a website to control the servos was made mainly because it is easy to get the device orientation from a web browser than using API's like Core Motion which can be used to get accelerometer reading from iPhone devices which can be a little bit hard for someone to get their head around which also will make the robot camera control device depended in this case if we were to use Core Motion the robot will only be controlled by iOS devices only. Using a website to control the camera pan and tilt will allow the camera controls of the robot not to be platform depended meaning the camera can be controlled from any device which can connect to the internet and publishes its orientation data via the web browser meaning whether the connected device is an Android or iOS device it can still control the camera movements of the robot.

N.B. Only mobile devices can control the camera as not all laptops publish their accelerometer readings while some do not have accelerometers at all.

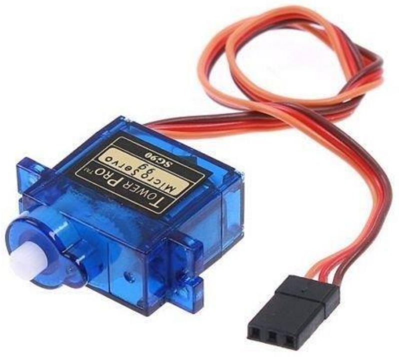


Fig1.3 – Servo

Camera. For video streaming. To be able to provide the user with the sense of vision where the robot is, a camera is used to give the user a video feed of the environment in which the robot is. A small 5 Megapixel camera will be used for this. It will be able to provide a 1080pixel high definition video recording at 30 frames per second for a pure video feed. A fisheye lens can be fitted in-front of the camera to provide a 3D visual of the area.

The camera will be mounted on the two above mentioned servos. The computer will stream the camera feed to a website as well just like the servos. This will allow the robot to be used by any mobile device from any platform making the robot platform independed like the servos. To get the VR experience an application called “WebDuo” can be downloaded on the mobile device which is to connect to the robot camera. WebDuo allows dual view which displays the same screen like a mirror. It will display two web pages at the same time on the same screen. With this feature, two tabs of the robot camera website can be opened simultaneously and the device placed in the VR headset to get the virtual presence feel.

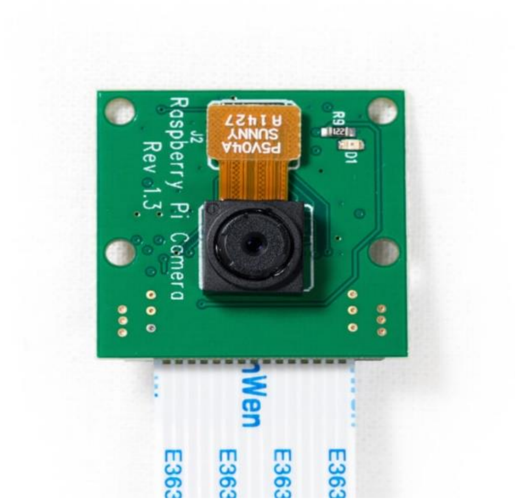


Fig1.4 – RPi Camera

Xbox Controller. Originally this is a game controller for the Xbox One gaming consoles by Microsoft. The controller is compatible with operating systems such as Linux, iOS, macOS, Android to mention a few. It contains a micro USB port which allows for wired use of the gamepad with the robot. The controller also features a new proprietary protocol which has a higher bandwidth essential for communication. This will allow us to add voice commands to control the robot using a microphone on headphones connected to the controller. The controller also features a Bluetooth module embedded in it which allows other devices with Bluetooth capabilities to connect to the controller as if it was just like any other device which has Bluetooth. A driver needs to be installed on the host computer to enable it to communicate and use the controller as just pairing with the controller using Bluetooth will not properly configure the controller to communicate with the robot hardware. The Bluetooth capabilities of the controller will enable wireless control of the robot. Connecting the controller with Bluetooth isn’t enough to control the direction of the robot, key mapping of the controller buttons has to be made. The button value of the pressed key or pushed analogue will have to make the robot do something for instance pushing the left analogue up will make the robot turn left and pushing the same analogue to the right will make the robot turn to the right.

Key mapping was made possible by a package called pygame which will be explained more in detail later on. This package is used by games to read button presses from gamepads and to reconfigure controller keys.

Fig1.5 – XboxOne Controller

N.B. not every button was used only the left analogue and trigger buttons were used.

2.2 SELECTION LANGUAGES

PYTHON. Is a high level programming language which has an object oriented approach. Python code and be dynamically typed.

It supports multiple programming paradigms which include procedural programming (structured programming), functional programming and object-oriented programming.

The main focus of the programming language is the ease of use. Python is an interpreted meaning it is not compiled into machine code rather a python interpreter will interpret the instructions at runtime. This allows the same code to be used on different machines.

Python is one of the most popular programming languages in robotics mainly because Python and C++ are the 2 main programming languages which ROS (Robot Operating System) recognises. Python also a large number of free libraries which will prevent us from reinventing the wheel when we are implementing some functionality in our robot.

JavaScript. JS is one of the programming languages used in the development of the telepresence robot. JavaScript is also an interpreted language just like python meaning that it does not produce an executable file rather instructions are interpreted at runtime. It is normally referred to as just-in-time compiled programming language.

JavaScript is lightweight and is well known as a scripting language for web pages though it can be used for non-browser environments for example Apache CouchDB. It supports object-oriented, imperative and declarative programming styles.

Since JavaScript runs on the client side it can be used to program the occurrence of event which is what we need in our robot to get the orientation of the mobile device so as to control the camera pan and tilt of the robot. JavaScript has the ability to get events in our case we can use javascript to get the accelerometer events in real time and relay them to the servos which control the camera in real time.

Despite some naming, syntactic, and standard library similarities, JavaScript and Java are otherwise unrelated and have very different semantics. The syntax of JavaScript is actually derived from C, while the semantics and design are influenced by the self and Scheme programming languages.

HTML. A little bit of HTML was used in the development of the project.

Hypertext Markup Language is the standard language for anything displayed by the web browser on the internet.

The beauty of HTML is that it can be used with other technologies such a s cascading style sheets (CSS), it can also be used alongside scripting languages like JavaScript mentioned above, basically we can embed code from JavaScript or PHP which is one of the programing languages widely used on the internet today to make interactive web pages and make websites look amazing. To use these on a website they have to be included inside an HTML tag because without it a script written in JavaScript or PHP will not run as HTML is the basic language needed to make web pages. That's how essential HTML is to the creation of websites.

Hypertext are links that are used to connect internet pages to each other between sites or in the same website. One can become an active participant on the World Wide Web simply by uploading on the internet and linking whatever they upload to web pages created by other internet users.

In this project HTML embedded with some JavaScript was used to make a website from which the user will connect to so that they can use the servos to control the camera pan and tilt.

2.3 SELECTION API

These are rules and features existing inside a program that allows us to interact with the program through software and not through human user interface. It is a contract between the application which is providing the API and other software or hardware using it.

FLASK. Is a micro Web Server Gateway Interface web application framework.

A web server gateway interface is a specification which describes how a web server communicates with other web applications and also how web apps can be chained to process one request

Flask is a micro framework because I does not require any tools or libraries and it has no form validation or database abstraction or any third party libraries which provide some common functions.

Although flask does not require any tools or third party libraries for common functionality it supports extensions which an application can use to add application features as though they were implemented by flask.

Flask API provides an implementation of browsable API like Django. The flask API because of its simplicity and lightweight it was a good option to use it on the robot to host and publish the website from which the servos will be controlled.

It will allow us to get the necessary readings to move the servos from the phone and send that same information over the network to the raspberry pi which will in turn move the physical servos.

PYGAME. These are python modules which are designed for video game programming. Because of its wide use pygame is compatible with every operating system and platforms.

Pygame is open source released under the LGPL licence and it can be used to create open source, shareware and free commercial games under the LGPL license.

It is a wrapper for the Simple DirectMedia Layer library (SDL library) which provides access to the system hardware components such as keyboard, joystick, video, sound and mouse which are used in gaming mainly.

Pygame has a cross-platform nature which means it can be used to write games and applications which can run on any platform which supports them.

Pygame has functions which allows a program to get button events from connected joysticks and manipulate the behaviour of a program just like a game. These functions where included in the software of the robot which enabled us to get values from the Xbox Controller buttons and change the behaviour of our robot in the process.

Pygame has key mappings in it which are general for all games. These key mappings are in a dictionary which can be used to customise how the buttons of different game pads do different functions other than the general functionalities. In this case the left analogue of the Xbox controller was customized to be used to turn the robot left and right and the two trigger buttons to move the robot forward and backwards.

2.4 SELECTION DRIVER

A driver is software or part of a software that allows the operating system to communicate with a device connected to it wired or wireless. A driver is written by the manufacturer of the device and it knows how to communicate with the hardware device to get data from it.

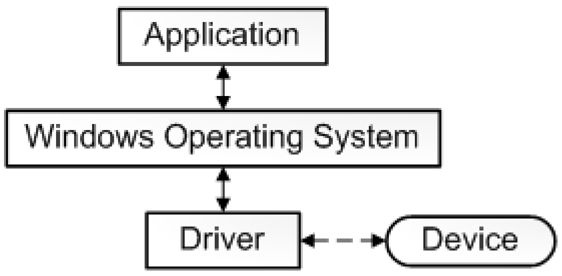


Fig 2.1 – Driver diagram

XBOXDRV. Made for LINUX systems xboxdrv is a gamepad driver which allows xbox 360 controller to work in the userspace. Besides from the xbox related stuff the driver has support for other gamepads such as the Thrustmaster Dual Power 3 gamepad.

The driver can be used on top of the linux basic joystick driver. It provides a lot of configuration options which allows for keyboard and mouse simulation events.

Remapping of the Xbox one controller keys was made possible by this driver as it allows remapping of buttons and axes and also it enables to invert the axis to normal as by default the y axis is inverted and it can be a little confusing when working with the inverted y axis

Apart from it being a pure driver, xboxdrv has a set of configuration tools which gives the ability to tweak virtual input devices that xboxdrv will create and while it’s possible to read input data directly from an event device it allows configurability of the driver on regular computers joysticks.

2.5 OpenSSH

It is a tool for remote connectivity and login with the SSH protocol. Communication with the raspberry pi 3 is done through SSH. The computer will be operated using the terminal and passing it commands to execute specific tasks is done through SSH.

SSH stands for secure shell, a shell from which the robot will be controlled from. SSH encrypts its traffic to avoid hacking, connection hijacking amongst a lot of other attacks.

SSH is easy to install and configure and it has several authentication methods for secure connections. It is made under a BSD-style license and is included in many commercialised products.

OpenSSH Daemon or sshd or server is the deamon program which runs in the background for the ssh client. It provides an encrypted communication between two hosts over an untrusted or unsecure network like the internet.

3 ROBOT REALIZATION

3.1 ROBOT MODELLING

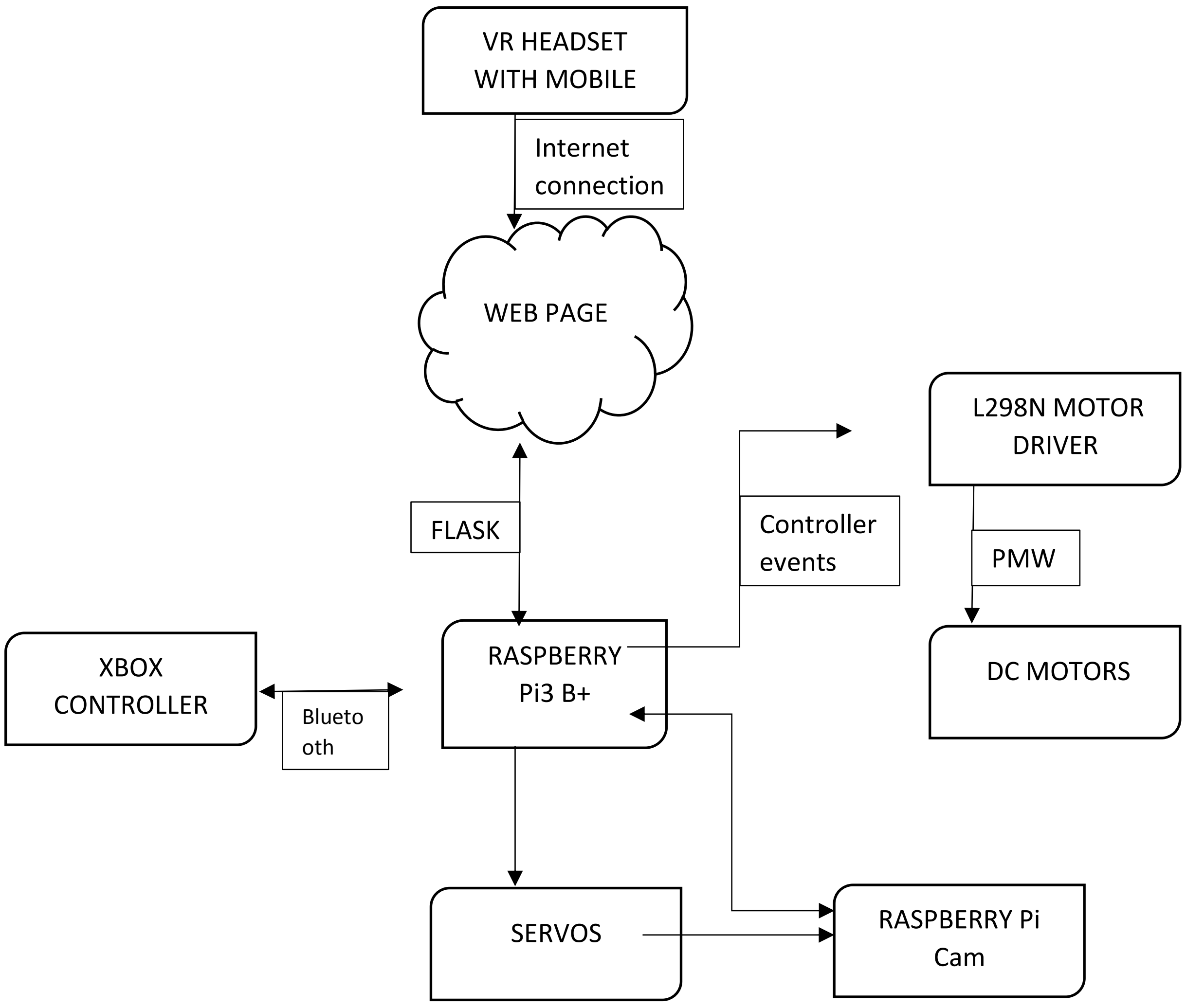


Fig 3.1 – Robot Model

The above is a hardware model of how the robot is connect and how it works. The mobile phone inside the VR headset will connect to a site hosted by the robot using flask application.

The website will request orientation information from the connected device. The orientation data is sent to the raspberry pi computer. The computer will then send the orientation values of the connected device to the servos connected to the gpio pins of the raspberry pi. These values will be using for camera tilt and panning in simple terms the servos will move the camera according to the accelerometer values the raspberry pi is receiving from the connected device.

The Xbox Controller will connect and communicate with the raspberry pi via a Bluetooth connection. Real time controller key event values are sent to the raspberry pi computer which it will interpret them using the xboxdrv driver and pygame.

The signals are sent to the L298N motor driver which is able to move the dc motors according to the command it receives from the computer which itself would have received from the xbox controller.

During the development of the robot, camera panning and tilt proved to be not so easy to accomplish with the game controller because Duty Cycle does not accept negative values and if we were to use the controller the robot would not be able to look to the left as far left is -100%.

A decision to use the accelerometer values of a mobile device was made.

ACCELEROMETER. An accelerometer is a device which measures changes in velocity in one of the 3 axis of a device. It is used to get the device orientation in order to control some applications or some hardware components of the device.

Accelerometer values are measured in increments of gravitational acceleration with a value of 1.0 representing 9.8 meters per second. Depending on the direction of acceleration the produced values maybe positive or negative.

Most smartphones today have these accelerometers preinstalled in the devices by the manufactures. An accelerometer is not a software it’s a small hardware chip which measures and transmits orientation information of the underlying device.

The easiest way to get these accelerometer values is through a web browser as the web browser automatically uses these accelerometer value for websites which have a device motion detection functionality in them on their web pages. VR uses these accelerometer values as well.

Using a website proved easy than to us API's which help us get these accelerometer values because of the main reason that if we use the web browser the robot will not be platform dependent meaning it won't only work on one type of device for example iOS devices if an apple API like Core Motion was used to get accelerometer values from iPhone devices the servos will only work with iPhone devices and not compatible with Android devices. If another device is to be used, it's API to get accelerometer values will have to be added to the robot making the robot software long and complicated for such a simple project.

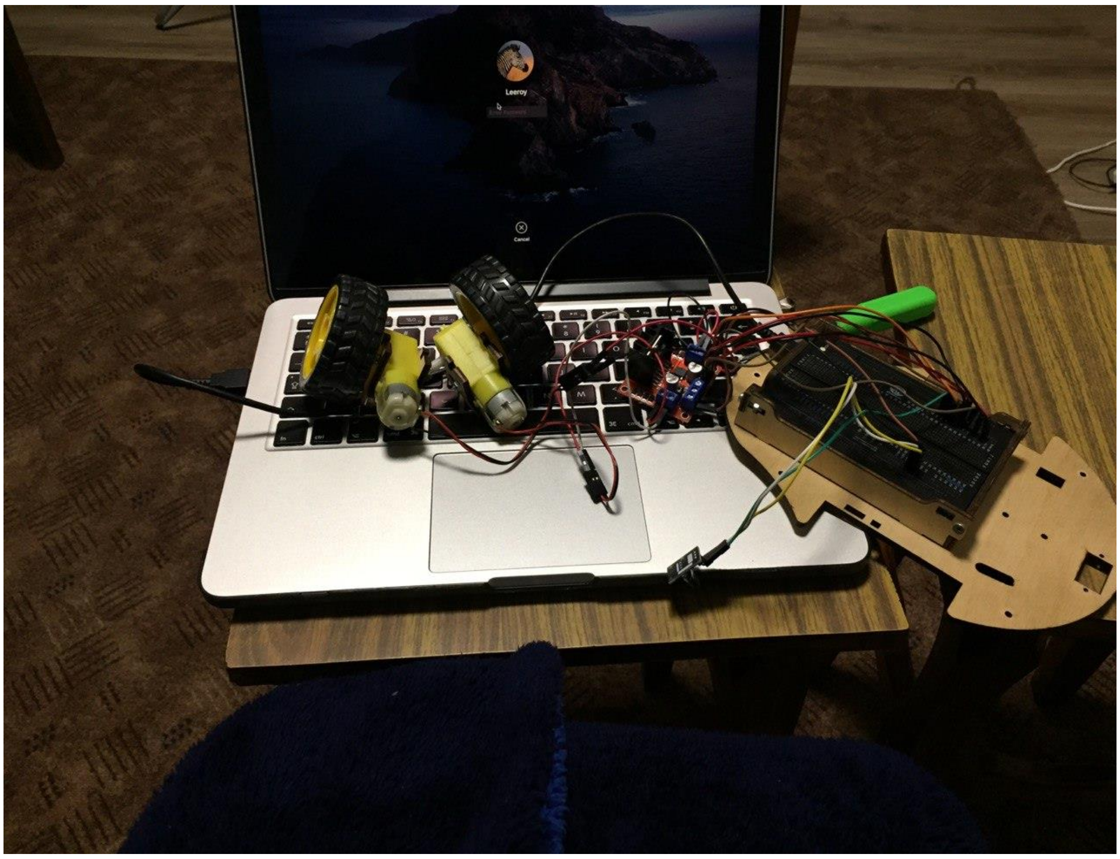


Fig. 3.2 – dc motor arduino test

The above picture shows the wheels of the robot being tested on an Arduino board. Arduino board was used to test the wheel movement before connecting them on a raspberry pi because of power requirement. The wheels need 9 volts to so it wasn't practical to connect them direct to the raspberry pi as there was a high risk of damaging the computer because of the power requirements of the wheels.

Below is a picture of the wheels powered on using the Arduino breadboard.

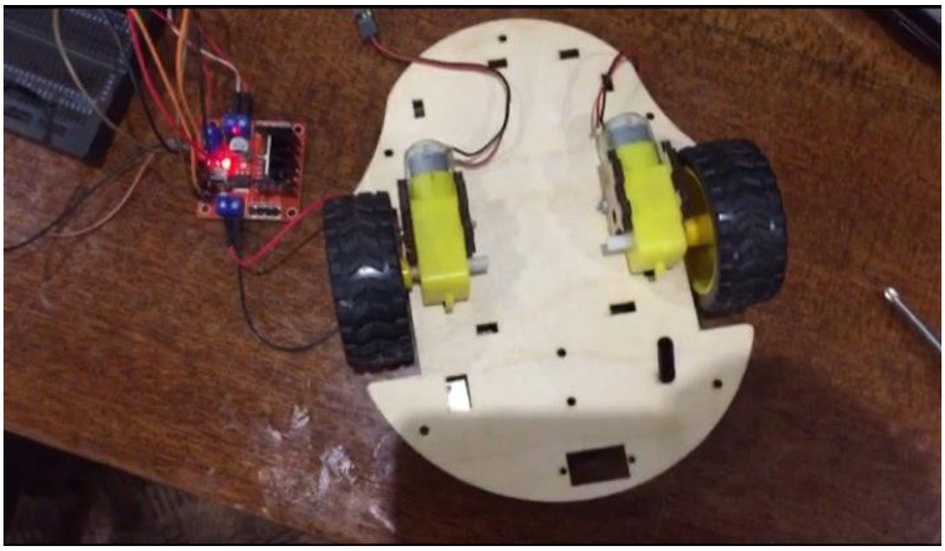


Fig. 3.3 – wheel mounting

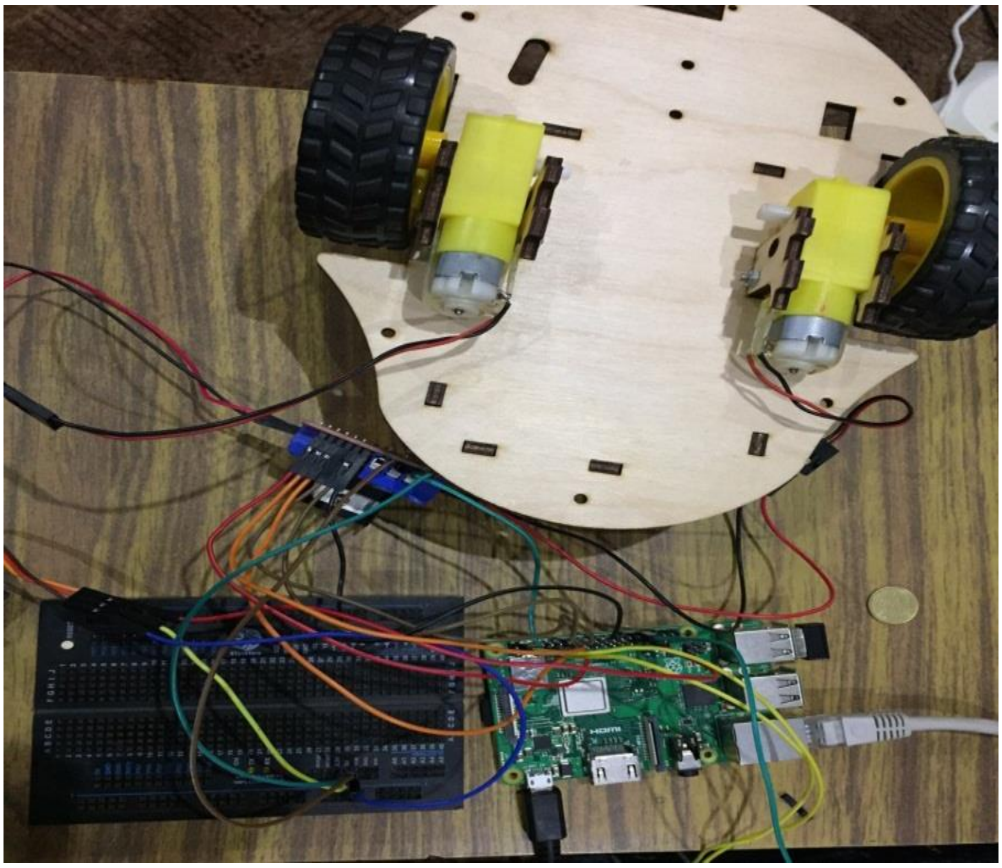


Fig. 3.4 – dc motor raspberry pi connection and test

Fig 3.4 the wheels now being controlled by the raspberry pi but getting their power supply from the Arduino breadboard.

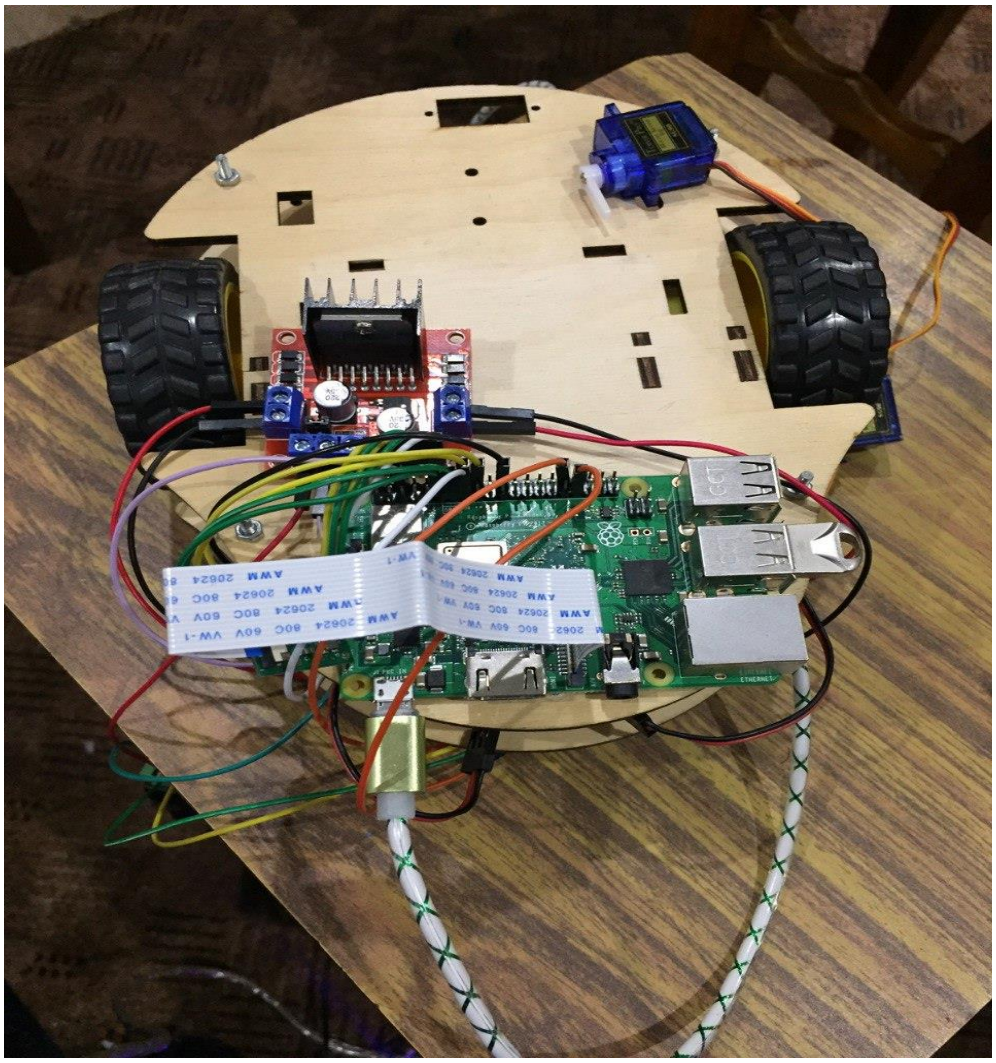
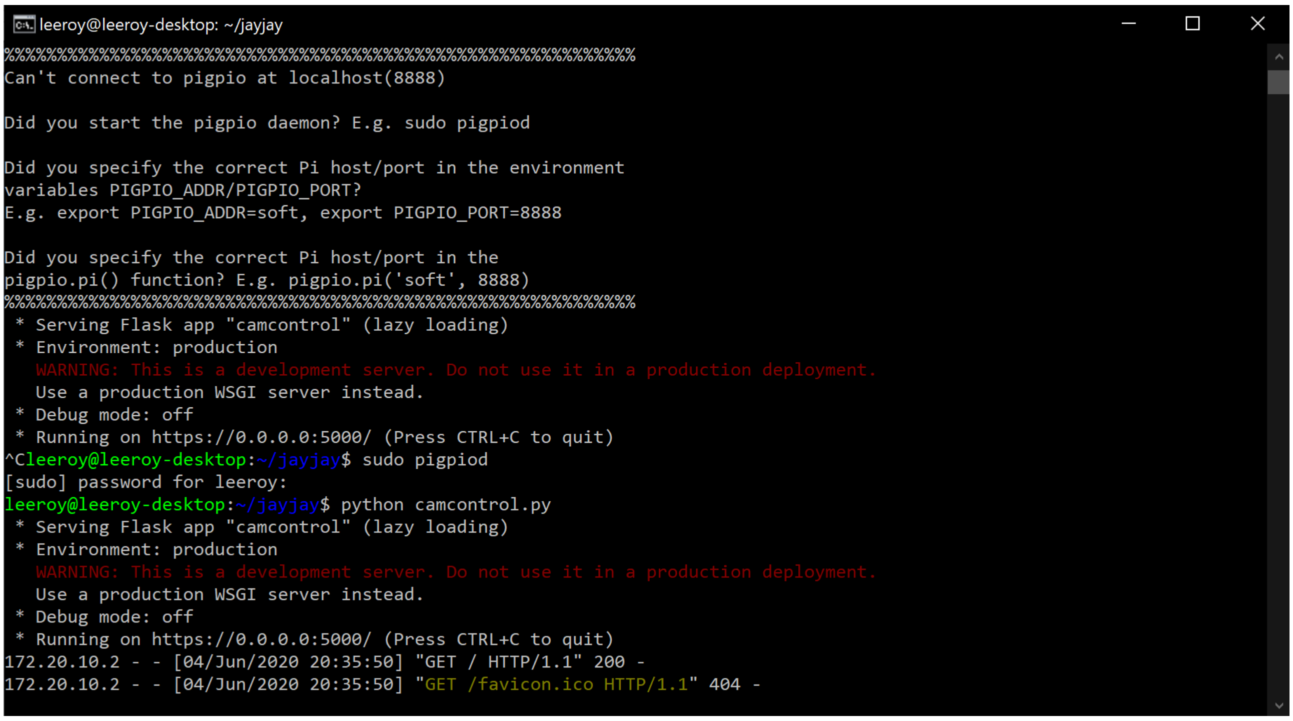


Fig. 3.5 - full assembly with camera attached

Full realization of the robot is seen on the picture above with the camera attached to it now.

Now the robot is running on batteries to power on the wheels and the raspberry pi computer is being powered by a powerbank.



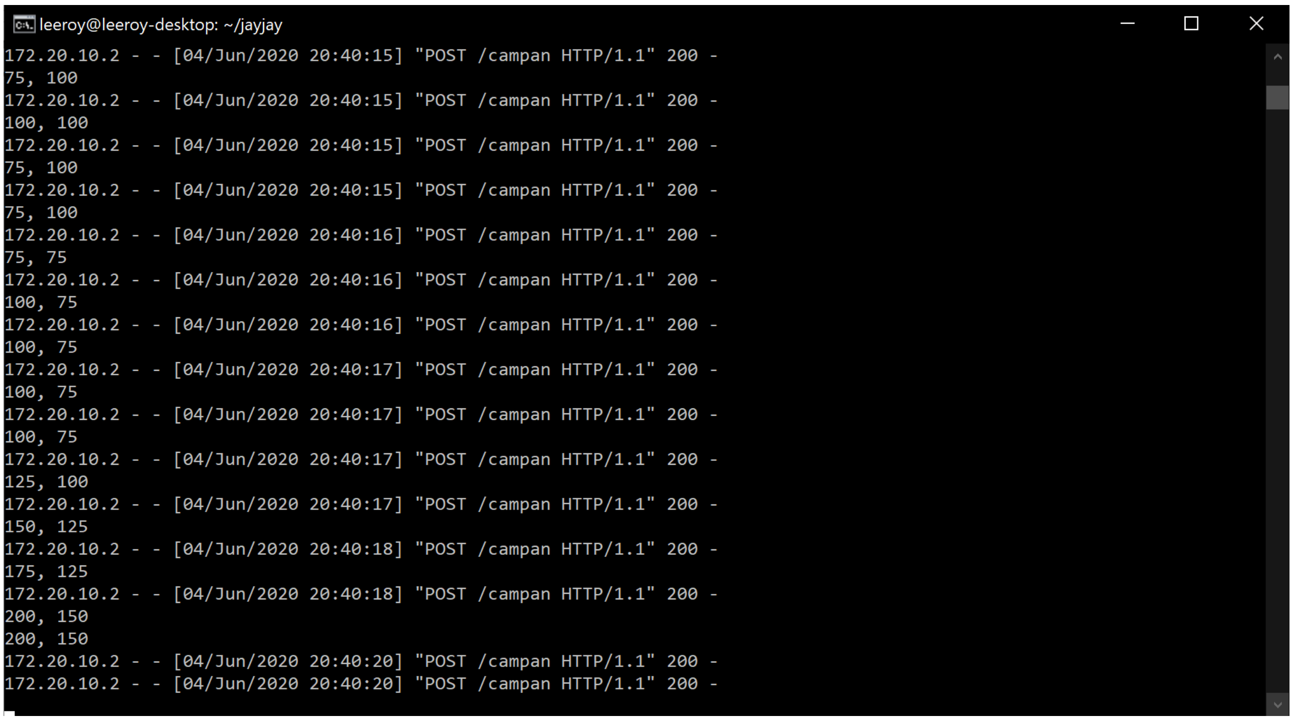
leeroy@leeroy-desktop: ~/jayjay

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Can't connect to pigpio at localhost(8888)

Did you start the pigpio daemon? E.g. sudo pigpiod

Did you specify the correct Pi host/port in the environment
variables PIGPIO_ADDR/PIGPIO_PORT?
E.g. export PIGPIO_ADDR=soft, export PIGPIO_PORT=8888

Did you specify the correct Pi host/port in the
pigpio.pi() function? E.g. pigpio.pi('soft', 8888)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 * Serving Flask app "camcontrol" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on https://0.0.0.0:5000/ (Press CTRL+C to quit)
^Cleeroy@leeroy-desktop:~/jayjay$ sudo pigpiod
[sudo] password for leeroy:
leeroy@leeroy-desktop:~/jayjay$ python camcontrol.py
 * Serving Flask app "camcontrol" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on https://0.0.0.0:5000/ (Press CTRL+C to quit)
172.20.10.2 - - [04/Jun/2020 20:35:50] "GET / HTTP/1.1" 200 -
172.20.10.2 - - [04/Jun/2020 20:35:50] "GET /favicon.ico HTTP/1.1" 404 -
```



leeroy@leeroy-desktop: ~/jayjay

```
172.20.10.2 - - [04/Jun/2020 20:40:15] "POST /campan HTTP/1.1" 200 -
75, 100
172.20.10.2 - - [04/Jun/2020 20:40:15] "POST /campan HTTP/1.1" 200 -
100, 100
172.20.10.2 - - [04/Jun/2020 20:40:15] "POST /campan HTTP/1.1" 200 -
75, 100
172.20.10.2 - - [04/Jun/2020 20:40:15] "POST /campan HTTP/1.1" 200 -
75, 100
172.20.10.2 - - [04/Jun/2020 20:40:16] "POST /campan HTTP/1.1" 200 -
75, 75
172.20.10.2 - - [04/Jun/2020 20:40:16] "POST /campan HTTP/1.1" 200 -
100, 75
172.20.10.2 - - [04/Jun/2020 20:40:16] "POST /campan HTTP/1.1" 200 -
100, 75
172.20.10.2 - - [04/Jun/2020 20:40:17] "POST /campan HTTP/1.1" 200 -
100, 75
172.20.10.2 - - [04/Jun/2020 20:40:17] "POST /campan HTTP/1.1" 200 -
100, 75
172.20.10.2 - - [04/Jun/2020 20:40:17] "POST /campan HTTP/1.1" 200 -
125, 100
172.20.10.2 - - [04/Jun/2020 20:40:17] "POST /campan HTTP/1.1" 200 -
150, 125
172.20.10.2 - - [04/Jun/2020 20:40:18] "POST /campan HTTP/1.1" 200 -
175, 125
172.20.10.2 - - [04/Jun/2020 20:40:18] "POST /campan HTTP/1.1" 200 -
200, 150
200, 150
172.20.10.2 - - [04/Jun/2020 20:40:20] "POST /campan HTTP/1.1" 200 -
172.20.10.2 - - [04/Jun/2020 20:40:20] "POST /campan HTTP/1.1" 200 -
```

Fig. 3.6 – raspiberry pi getting accelerometer data from mobile phone

CONCLUSION

In conclusion the approach was useful to show that there is a wide range of telepresence technology implications. After installing and assembling the hardware properly and launching the software, it was shown that telepresence robots can be implemented and controlled using mobile devices which will lead to easy platform independent robots controlled from anywhere.

LIST OF REFERENCES

1. https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/
2. https://developer.arm.com/ip-products/processors/cortex-a/cortex-a53
3. https://stackoverflow.com/questions/2939685/accelerometer-api-for-laptops
4. https://learn.sparkfun.com/tutorials/pulse-width-modulation/all
5. https://developer.mozilla.org/en-US/docs/Web/JavaScript
6. https://en.wikipedia.org/wiki/JavaScript
7. https://en.wikipedia.org/wiki/HTML
8. https://developer.mozilla.org/en-US/docs/Web/HTML
9. https://developer.mozilla.org/en-US/docs/Glossary/API
10. https://flask.palletsprojects.com/en/1.1.x/
11. https://palletsprojects.com/p/flask/
12. https://realpython.com/pygame-a-primer/
13. https://www.pygame.org/wiki/about
14. https://docs.microsoft.com/en-us/windows-hardware/drivers/gettingstarted
15. https://github.com/xboxdrv/xboxdrv
16. https://www.cyberciti.biz/faq/ubuntu-linux-install-openssh-server/
17. https://www.stuffaboutcode.com/2014/10/raspberry-pi-xbox-360-controller-python.html

ADDITITION 1. Wheel control code

```
import pygame

from pygame.locals import *

import os, sys

import threading

import time

import RPi.GPIO as GPIO


class XboxController(threading.Thread):

class XboxControls():

    LTHUMBX = 0

    LTHUMBY = 1

    LB = 10

    RB = 11

        class PyGameAxis():

    LTHUMBX = 0

    LTHUMBY = 1

    RTRIGGER = 4

    LTRIGGER = 5


class PyGameButtons():

    LB = 4

    RB = 5


AXISCONTROLMAP = {PyGameAxis.LTHUMBX: XboxControls.LTHUMBX,

                 PyGameAxis.LTHUMBY: XboxControls.LTHUMBY,

TRIGGERCONTROLMAP = {PyGameAxis.RTRIGGER: XboxControls.RTRIGGER,

                    PyGameAxis.LTRIGGER: XboxControls.LTRIGGER}
```

```
def __init__(self,
            controllerCallBack = None,
            joystickNo = 0,
            deadzone = 0.1,
            scale = 1,
            invertYAxis = False):

threading.Thread.__init__(self)
self.running = False
     self.controllerCallBack = controllerCallBack
     self.joystickNo = joystickNo
     self.lowerDeadzone = deadzone * -1
     self.upperDeadzone = deadzone
     self.scale = scale
     self.invertYAxis = invertYAxis
     self.controlCallbacks = {}

self.controlValues = {self.XboxControls.LTHUMBX:0,
                    self.XboxControls.LTHUMBY:0,
                    self.XboxControls.RTRIGGER:0,
                    self.XboxControls.LTRIGGER:0,




self._setupPygame(joystickNo)

@property
  def LTHUMBX(self):
     return self.controlValues[self.XboxControls.LTHUMBX]

  @property
  def LTHUMBY(self):
        return self.controlValues[self.XboxControls.LTHUMBY]
    @property
    def RTRIGGER(self):
        return self.controlValues[self.XboxControls.RTRIGGER]

    @property
    def LTRIGGER(self):
        return self.controlValues[self.XboxControls.LTRIGGER]


def _setupPygame(self, joystickNo):
pygame.init()
pygame.joystick.init()
joy = pygame.joystick.Joystick(joystickNo)
joy.init()

def run(self):
        self._start()
def _start(self):

        self.running = True
while(self.running):
            for event in pygame.event.get():
                        if event.type == JOYAXISMOTION:
                if event.axis in self.AXISCONTROLMAP:
                    yAxis = True if (event.axis == self.PyGameAxis.LTHUMBY or event.axis ==
self.PyGameAxis.RTHUMBY$
self.updateControlValue(self.AXISCONTROLMAP[event.axis],
                                    self._sortOutAxisValue(event.value, yAxis))
                if event.axis in self.TRIGGERCONTROLMAP:
                    self.updateControlValue(self.TRIGGERCONTROLMAP[event.axis],
                                    self._sortOutTriggerValue(event.value))
```

```
def stop(self):
    self.running = False

def doCallBacks(self, control, value):
if self.controllerCallBack != None: self.controllerCallBack(control, value)
if control in self.controlCallbacks:
      self.controlCallbacks[control](value)
def setupControlCallback(self, control, callbackFunction):
self.controlCallbacks[control] = callbackFunction

if __name__ == '__main__':

  LWP = 24
  LWN = 23
  RWP = 17
  RWN = 27

  LEN = 25
  REN = 22

  GPIO.setmode(GPIO.BCM)

  GPIO.setup(LWP,GPIO.OUT)
  GPIO.setup(LWN,GPIO.OUT)
  GPIO.setup(RWP,GPIO.OUT)
  GPIO.setup(RWN,GPIO.OUT)
  GPIO.setup(LEN,GPIO.OUT)
  GPIO.setup(REN,GPIO.OUT)

  L=GPIO.PWM(LEN,1000)
  R=GPIO.PWM(REN,1000)
```

```
    Pygame negative is up - wier$    xboxCont = XboxController(controlCallBack, deadzone = 10, scale = 100,
invertYAxis = True)

    #setup the left thumb (X & Y) callbacks
    xboxCont.setupControlCallback(xboxCont.XboxControls.LTHUMBX, leftThumbX)
    xboxCont.setupControlCallback(xboxCont.XboxControls.LTHUMBY, leftThumbY)
    xboxCont.setupControlCallback(xboxCont.XboxControls.RTRIGGER, rt)
    xboxCont.setupControlCallback(xboxCont.XboxControls.LTRIGGER, lt)
    try:
        xboxCont.start()
        print "xbox controller running"
        while True:

            except KeyboardInterrupt:
        print "User cancelled"

    except:
        print "Unexpected error:", sys.exc_info()[0]
        raise

    finally:
        xboxCont.stop()
```

Xbox Pad API. The above is the key configuration of the xbox controller. The default configuration is for the xbox 360 but we are using an xbox one controller so the pygame was used and xboxdrv to change the controller configurations so that they can be compatible with our robot.

```
AXISCONTROLMAP = {PyGameAxis.LTHUMBX: XboxControls.LTHUMBX,
                PyGameAxis.LTHUMBY: XboxControls.LTHUMBY,
```

The above is the key mapping of controller's left analogue to the pygame configurations. This configuration is the one which was used to control the wheels of the robot to turn left or write using the left analogue.

The commands received from the controller are interpreted as if the controller was used for playing a game by the computer but instead of moving variables in a game the commands will be used to control the movement of the robot.

```
TRIGGERCONTROLMAP = {PyGameAxis.RTRIGGER: XboxControls.RTRIGGER,
                 PyGameAxis.LTRIGGER: XboxControls.LTRIGGER}
```

Above is the dictionary of the two trigger buttons on the controller the left and right trigger buttons.

```
self.XboxControls.LTHUMBX:0,
                   self.XboxControls.LTHUMBY:0,
                   self.XboxControls.RTRIGGER:0,
                   self.XboxControls.LTRIGGER:0,
```

setting up controller properties.

```
@property
  def LTHUMBX(self):
    return self.controlValues[self.XboxControls.LTHUMBX]

  @property
  def LTHUMBY(self):
    return self.controlValues[self.XboxControls.LTHUMBY]
  @property
  def RTRIGGER(self):
    return self.controlValues[self.XboxControls.RTRIGGER]
  @property
  def LTRIGGER(self):
    return self.controlValues[self.XboxControls.LTRIGGER]
```

The above are properties of the buttons we are going to use. @property in python simply are setters and getters so in this case we are getting controller buttons and setting them up.

```
def leftThumbX(xValue):
    if xValue>=0 and xValue<=100:
        GPIO.output(RWP,GPIO.HIGH)
        GPIO.output(RWN,GPIO.LOW)
        GPIO.output(LWP,GPIO.LOW)
        GPIO.output(LWN,GPIO.HIGH)

        L.start(25)
        R.start(22)
        L.ChangeDutyCycle(xValue)
        R.ChangeDutyCycle(xValue)
        print "RIGHT TURN {}".format(xValue)

  def leftThumbY(yValue):
    if yValue>=0 and yValue<=100:
        GPIO.output(RWP,GPIO.LOW)
        GPIO.output(RWN,GPIO.HIGH)
        GPIO.output(LWP,GPIO.HIGH)
        GPIO.output(LWN,GPIO.LOW)

        L.start(25)
        R.start(22)
        L.ChangeDutyCycle(yValue)
        R.ChangeDutyCycle(yValue)
```

```
            print "LEFT TURN {}".format(yValue)

   def rt(RTRIGGER):

      GPIO.output(RWP,GPIO.LOW)
      GPIO.output(RWN,GPIO.HIGH)
      GPIO.output(LWP,GPIO.LOW)
      GPIO.output(LWN,GPIO.HIGH)
      print "FORWARD {}".format(RTRIGGER)

      L.start(25)
      R.start(22)
      L.ChangeDutyCycle(RTRIGGER)
      R.ChangeDutyCycle(RTRIGGER)

   def lt(LTRIGGER):

      GPIO.output(RWP,GPIO.HIGH)
       GPIO.output(RWN,GPIO.LOW)
      GPIO.output(LWP,GPIO.HIGH)
      GPIO.output(LWN,GPIO.LOW)
      print "REVERSE {}".format(LTRIGGER)

      L.start(25)
      R.start(22)
      L.ChangeDutyCycle(LTRIGGER)
      R.ChangeDutyCycle(LTRIGGER)
```

Customization of the controller controls is done here. The values being received from the respective button presses are doing something for example

```
def leftThumbX(xValue):
```

the above function will turn the robot to the right.

ADDITITION 2. CAMERA PAN AND TILT CODE

```
from flask import Flask, render_template_string, request

from time import sleep

import pigpio


ORTN = '''

<html>

  <body>

    <button id="connectbtn" onclick="requestPermission()">Connect</button>

    <div id="outputdiv"></div>

  </body>

  <script>

    var output = document.getElementById("outputdiv");

var btn = document.getElementById("connectbtn");

var socket = false;


var vertical = 0;

var horizontal = 0;


function sendToFlask()

{

    const xmlhttpreq = new XMLHttpRequest();

    const data = new FormData();


    data.append("updown", vertical);

    data.append("leftright", horizontal);


    xmlhttpreq.open("POST", "campan");

    xmlhttpreq.send(data);

}
```

```
function handleRotation(event)
{
   vertical = Math.round(event.gamma);
   horizontal = Math.round(event.beta);
}


function requestPermission()
{
   if (typeof(DeviceMotionEvent) !== 'undefined' && typeof(DeviceMotionEvent.requestPermission) ===
'function')
   {
            DeviceMotionEvent.requestPermission().
then(response => {
         if (response === 'granted') {
            window.addEventListener('deviceorientation', handleRotation);
         }
         finishRequest();
      }).catch(console.error);
   }
   else
   {
      window.addEventListener('deviceorientation', handleRotation);
      finishRequest();
   }
}
function finishRequest()
{
   setInterval(Flask, 200);
}



// schedule the first one.
```

```
setTimeout(processVideo, 0);



  </script>
</html>
'''
sideservo = 26
upservo= 12

sideservoc = 1750
upservoc = 1500

app = Flask(__name__)

gpio.set_servo_pulsewidth(sideservo, 0)
   gpio.set_servo_pulsewidth(upservo, 0)
def setServoDuty(servo, duty):
   gpio.set_servo_pulsewidth(servo, duty)

def clamp(num, minimum, maximum):
   return max(min(num, maximum), minimum)



@app.route("/")
def serveRoot():
   return render_template_string(ORTN)

@app.route("/campan", methods=["POST"])
def campan():

   horizontal = 25 * int(request.form["updown"])
   vertical = 25 * int(request.form["leftright"])
```

```
    print(str(horizontal) + ", " + str(vertical))


    setServoDuty(sideservo, clamp(sideservoc - horizontal, 500, 2500))
    setServoDuty(upservo, clamp(upservoc - vertical, 500, 2500))


    sleep(0.5)


    gpio.set_servo_pulsewidth(sideservo, 0)
    gpio.set_servo_pulsewidth(upservo, 0)


    return

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000, ssl_context='adhoc')
```