EE 321
Project Final Report

HITRoViF

(A Haptics Implemented Teleoperated Robot with a Vision-Based Feedback System)

By

Ayhan Alp Aydeniz & Kemal Akçay

Introduction

This project final report considers a robotic system built for the final project of the course of EE 321. This project is a robotic arm controlled through haptics and it has a vision based feedback system showing whether the grasping mechanism holding the ball or not.

The word of “robot” first used in a play written by Karel Capek and it is used as a term for a manmade worker. Its origin was the Slavonic word “robota” meaning arduous work. In practice, since Unimate which is the first industrial robot, robots were used and still being used to do arduous works. Teleoperated robotics is a field of robotics to work in the environments in which humans are needed, but cannot be situated, through robots. In the control part, through haptics technology, robots can be actuated according to human body motion.

At the beginning of the project, our purpose was to build a robotic system which can be seen as a prototype of haptics implemented robotics.

This project includes a 3 degree-of-freedom (DoF) robotic arm and a gripping mechanism on the arm working through an Arduino and a camera system with which we will build the feedback mechanism. The whole system was built on a platform and the haptics mechanism is a wearable mechanism including an Arduino Uno, a potentiometer, and a joystick.

Khatib et al. (2016)

In the picture above, a state of art in the haptics implemented robotics can be seen. This robot is named as Oceanone and it used for the missions conducted in deep ocean. This robot is built and designed by a team led by Oussama Khatib at Stanford University. The main purpose of this robot, conducting underwater missions with humans requires to face some constraint like not having enough oxygen in tubes and time constraints. Therefore, haptics implemented robots can bring mesmerizing solutions to these problems.

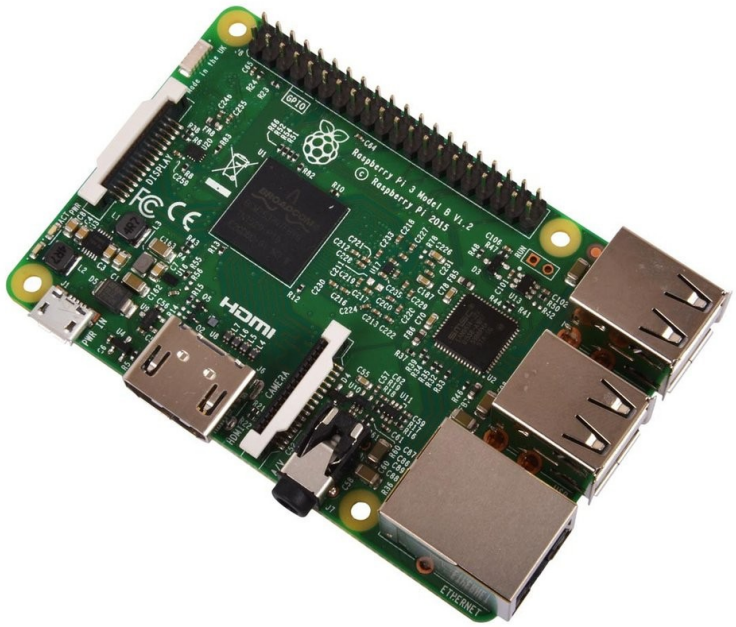
Design and Implementation

In this project, we have used both a microcontroller and a microprocessor for the whole system for task requiring different complexities. An Arduino Uno is used as a microcontroller and a Raspberry Pi 3 as a microprocessor.

Arduino Uno is a microcontroller having open source hardware and used as a microcontroller in the projects. It is based on the Atmega328P and it has 14 digital input/output pins. It is programmed through its own Integrated Development Environment (IDE).

Arduino Uno (Microcontroller)

Raspberry Pi 3 is a microprocessor which can run several operating systems, but, in this project, its own operating system called Raspbian is used. Through Raspberry Pi, we were able to implement some complex tasks like scanning all pixels values in a frame seen by a webcam.



Raspberry Pi 3 (Microprocessor)

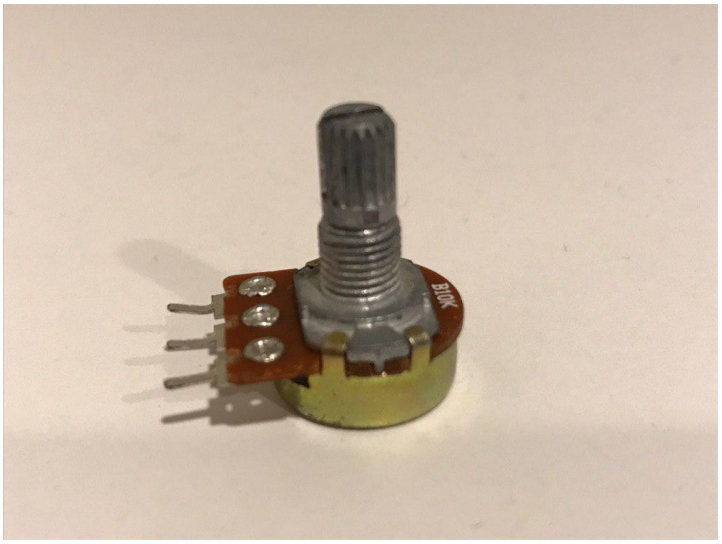
Servo motors were used to actuate the arm and three of them were used for 3 DoF mechanism; however, one of them will be used to grib the objects.

Servo Motor - MG 996

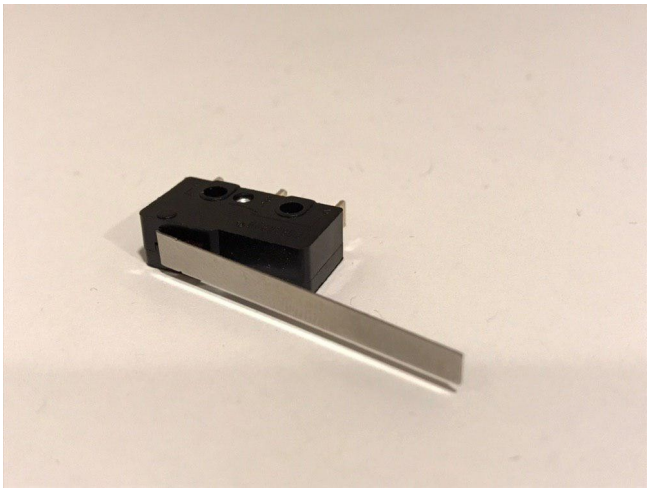
An image showing the servo motor which is used in this project can be seen above.

In order to actuate the servos according to the angles of the joints of human arm, we have used a potentiometer and a joystick. The potentiometer was responsible for the angle of our elbow, and the joystick which have the ability of moving in two directions was responsible for the angles of our wrist. We have used a limit switch to send the data to start gripping.

The joystick



The potentiometer

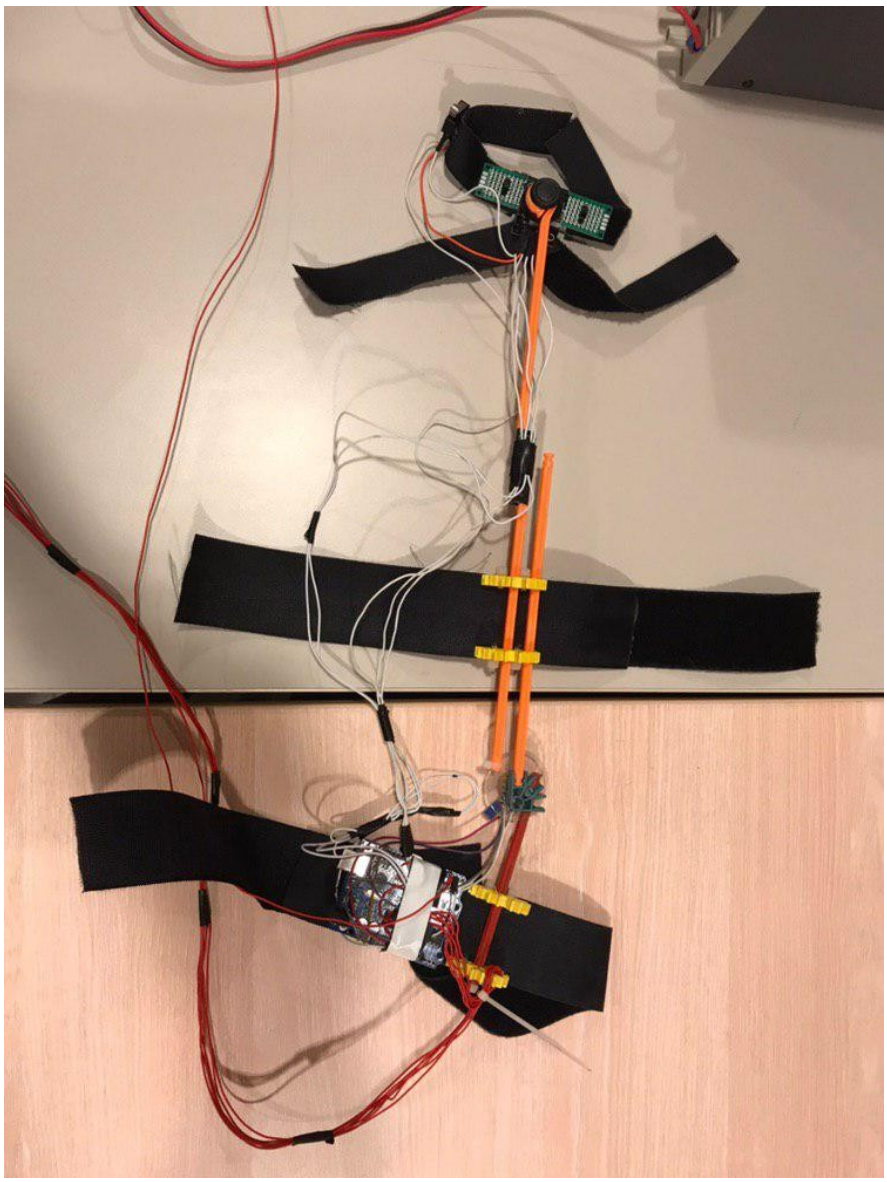


The limit switch

The pictures of the joystick and the potentiometer can be seen above.

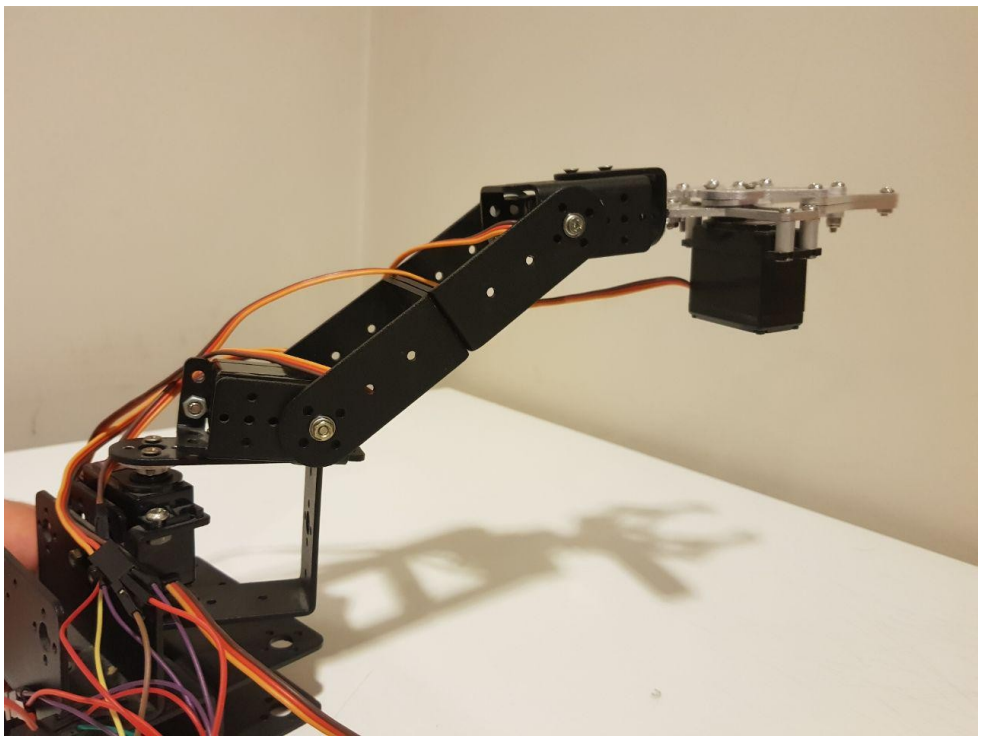
These parts are used to actuate the robotic arm and these are used as input data to the Arduino. The joystick and the potentiometer were used, because they can send continuous data. At the beginning, we were going to use rotary encoder; however, rotary encoder does not sen continuous data to the Arduino, so that we have switched the system branches to a potentiometer and to a joystick.

To build the wearable mechanism, we have used parts for Kinex. The parts of Kinex were better than the 3D printed parts. The wearable mechanism includes a potentiometer, a joystick, and an Arduino Uno.



The wearable control mechanism of Haptics system

A picture showing the control mechanism can be seen above and this mechanism was connected directly to the robotic arm. The data is sent through cables to the servo motors.

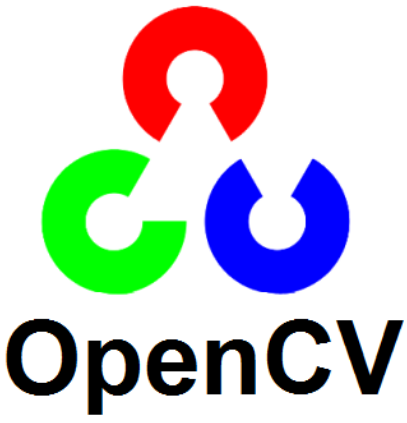


The 3-DoF robotic arm with a gribbing mechanism

The robotic arm is shown in the figure above. The parts of the robotic arm was already available with us, we have only assembled the robotic arm. The combination of the robotic arm and the wearable mechanism is the complete haptics implemented robotic system.

As mentioned in the section used as introduction, we have implemented a vision based feedback system. The vision based feedback system and the wearble mechanism are the parts that makes our whole system special. Because in the other projects available on the internet does not use a feedback system and they use only basic parts to control the robotic arm.

The feedback consists of a webcam, a Raspberry Pi 3, and 2 LEDs with a 220 ohm resistor. The vision system uses OpenCV as the computer vision library. At the beginning, we have used Processing to implement vision system; however, processing is not advanced and it is not as comprehensive as OpenCV.



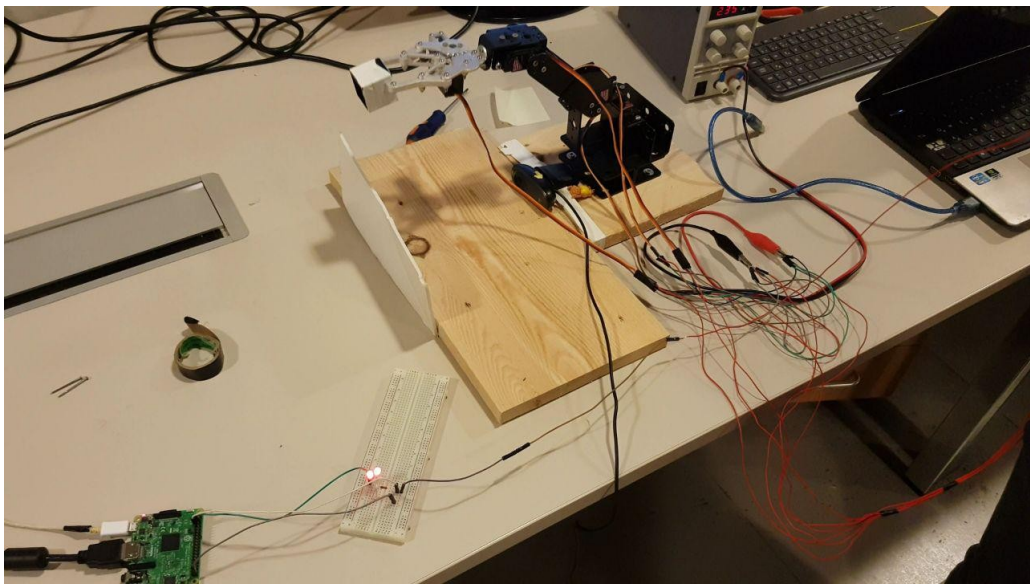
The logo of OpenCV

OpenCV is a comprehensive computer vision library used in many computer vision applications by the research community. The installation of OpenCV into Raspberry Pi 3 takes several hours. We have faced some problems, during the installation. Because OpenCV is a comprehensive computer vision library, it requires huge free space in the memory. We have firstly used an SD card which have 8 GB of memeory, but installation and compiling OpenCV failed for 2 times, then we have used an SD Card having 32 GB of memory. To compile OpenCV, the raspberry was working for 6 hours. However, at the end, we have installed OpenCV into a virtual environment in the Raspbian, so that the installation was done succesfully.

In the first try, we have purchased a Raspberry Pi camera; however, it did not function, then we found that it has a hardware problem, then we have used a Logitech webcam.

The Logitech Webcam we used in the feedback system

The feedback mechanism always check the frames including the objects that are available in front of the robotic arm. When the gripping was successful, webcam perceives a black point in the frames, then it turns the LEDs ON. The Raspberry Pi 3 also takes as inputs the limit switch in the system. When the switching turns 1 and the black points appear in the system, the Raspberry Pi 3 outputs through the LEDs.



The feedback mechanism

The feeedback mechanism can be shown in the figure above. The webcam stands in front of the robotic arm.

The parts were used in the project were our own parts, we have only purchased a Raspberry Pi Camera, but we have not used it in the project.

Similar Projects and Comparison

As the similar projects, there are various haptics implemented robotic arm projects.
Youtube videos of two projects that are most relevant can be found via the links below

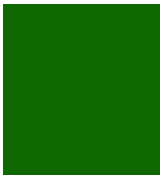
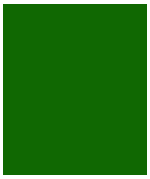
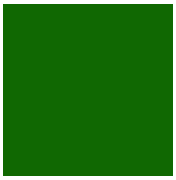
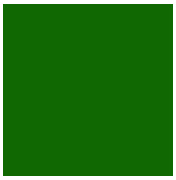
- https://www.youtube.com/watch?v=2YwYnAYH6Pk
- https://www.youtube.com/watch?v=zUgE1CyHiPA

As the innovative and the unique parts of our project, we have implemented a feedback system which is not available in other related projects and we have designed and build our own unique wearable control system.

Youtube Link

- https://youtu.be/npqIy8wsHLw

Gantt Chart

Tasks	Week 9	Week 10	Week 11	Week 12	Week 13	Week 14
Design and Schema (Kemal and Alp)						
Finding the required parts (Kemal and Alp)						
Assembly of Robotic Arm (Kemal and Alp)						
Implementation of Wearable Mechanism (Kemal)						
Implementation of the Vision System (Kemal)						
Combining the subsystems (Kemal and Alp)						

Appendix

[1]

```
'''
   Ayhan Alp Aydeniz - Kemal Akçay
   Adapted from the original code written by Adrian Rosebrock
     Visit original post: https://www.pyimagesearch.com/2016/05/09/opencv-rpi-gpio-and-gpio-zero-on-the-raspberry-pi/
'''
from __future__ import print_function
from imutils.video import VideoStream
import argparse
import imutils
import time
import cv2
import RPi.GPIO as GPIO

redLed = 21
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(redLed, GPIO.OUT)

GPIO.setmode(GPIO.BCM)
GPIO.setup(4, GPIO.IN)

ap = argparse.ArgumentParser()
ap.add_argument("-p", "--picamera", type=int, default=-1,
       help="whether or not the Raspberry Pi camera should be used")
args = vars(ap.parse_args())

vs = VideoStream(usePiCamera=args["picamera"] > 0).start()
time.sleep(2.0)

colorLower = (0, 0, 0)
colorUpper = (90, 90, 90)

GPIO.output(redLed, GPIO.LOW)
ledOn = False

while True:
   frame = vs.read()
   frame = imutils.resize(frame, width=500)
   hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

   mask = cv2.inRange(hsv, colorLower, colorUpper)
   mask = cv2.erode(mask, None, iterations=2)
   mask = cv2.dilate(mask, None, iterations=2)

   cnts = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL,
         cv2.CHAIN_APPROX_SIMPLE)
   cnts = cnts[0] if imutils.is_cv2() else cnts[1]
   center = None

   if len(cnts) > 0:
      c = max(cnts, key=cv2.contourArea)
      ((x, y), radius) = cv2.minEnclosingCircle(c)
      M = cv2.moments(c)
```

```
        center = (int(M["m10"] / M["m00"]), int(M["m01"] / M["m00"]))

        if radius > 10:
            cv2.circle(frame, (int(x), int(y)), int(radius),
                (0, 255, 255), 2)
            cv2.circle(frame, center, 5, (0, 0, 255), -1)

            if not ledOn and GPIO.input(4)==0:
                GPIO.output(redLed, GPIO.HIGH)
                ledOn = True
            elif GPIO.input(4)==0:
                GPIO.output(redLed, GPIO.LOW)
                ledOn = False

    elif ledOn or GPIO.input(4)==0:
        GPIO.output(redLed, GPIO.LOW)
        ledOn = False

    cv2.imshow("AAA - KA", frame)
    key = cv2.waitKey(1) & 0xFF

    if key == ord("q"):
        break

print("\n Demo of the project is done - EE 321 \n")
GPIO.cleanup()
cv2.destroyAllWindows()
vs.stop()
```

[2]

```
#include <Servo.h>

  Servo myservo1;
  Servo myservo2;
  Servo myservo3;
  Servo myservo4;

int mv1,mv2;

int pos1 = 50;
int pos2 = 50;
int pos3 = 0;
int pos4 = 30;

int maxspd=5;
int dly=100;

void setup(){
  Serial.begin(9600);

      myservo1.attach(9);
      myservo2.attach(11);
      myservo3.attach(5);
      myservo4.attach(3);

     pinMode(13,OUTPUT);//output for raspberry
```

```
}
void loop(){

  mv1= map(analogRead(A0),0,1023,(-maxspd),(macspd+1));
  mv2= map(analogRead(A1),0,1023,(-maxspd),(maxspd+1));

  pos1 += mv1;
  pos2 += mv2;
  pos3= map(analogRead(A2),0,1023,0,180);

  if(pos1>180)pos1=180;
  if(pos2>180)pos2=180;
   if(pos1<0)pos1=0;
   if(pos2<0)pos2=0;

  if(digitalRead(2)==HIGH){
     pos4=130;
     digitalWrite(13,HIGH);
    }
    else{
      pos4=30;
      digitalWrite(13,LOW);

      }


    Serial.print(pos1);
      Serial.print("\t");
 Serial.print(pos2);
      Serial.print("\t");
 Serial.print(pos3);
      Serial.print("\t");
 Serial.println(pos4);

  myservo1.write(pos1);
  myservo2.write(pos2);
  myservo3.write(pos3);
  myservo4.write(pos4);

delay(dly);


}
```