Centre For Innovation

Project VISION

-Team Sahaay

SAHAAY
Social Innovation Club

So Who are We ?

WE ARE ONE OF YOU !!
With just a different VISION and
roll numbers...

1. Harshit Raj - ME19B110
2. Vinayak Nishant - EE19B129
3. Yogesh Agarwala - EE19B130
4. Saroopa G - ME19B163
5. Anish Anand Pophale - CH20B012

Centre For Innovation

Index

1. Problem we see
2. Solution
3. Design
4. Concept
    a. Image recognition
    b. Triangulation
    c. Gripper
5. Progress
6. Future target

SAHAAY
Social Innovation Club

The Problem we see

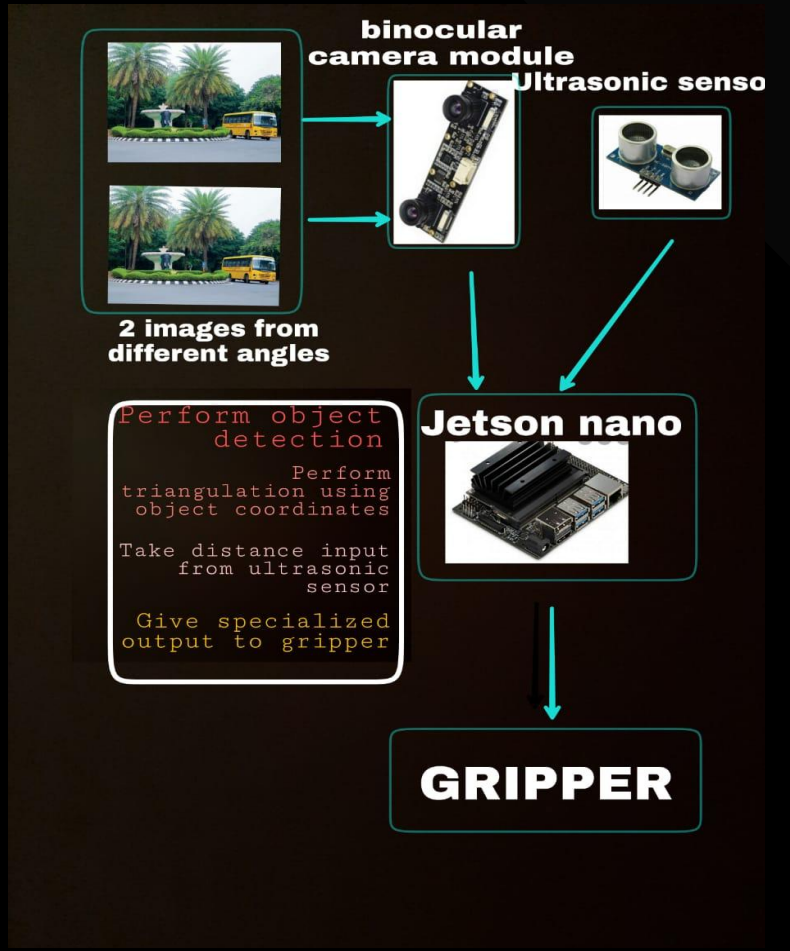
Independent mobility for a visually challenged person is a day-to-day problem. It is difficult for them to determine a safe path without colliding with over-hanging and protruding objects.

Such challenges may undermine their autonomy, cause emotional distress, and even expose them to injury. Project Vision attempts to solve mobility issues for visually impaired individuals.

Solution

We are solving this problem using **eep learning(YOLO)** and **computer vision** technology. The idea is to develop an Electronic Travel Aid for visually challenged peoples, to overcome the hurdles they face while using a traditional stick.

The project would act like artificial eyes for the visually challenged people, and the information regarding surrounding proximity will be conveyed on the user's fingertips with a special **gripper module**. It uses **ultrasonic sensors** and a **camera** to detect obstacles and their distance through computer vision and then notifies the user about it through vibrations on their fingertips.



Product design

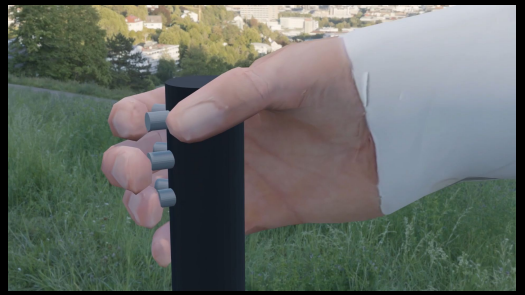
Centre For Innovation

The idea is to develop a Smart Belt that the user can wear and a specialized gripper, for output, attached to his usual walking stick.

- **The Belt**
  - Two cameras at a separation capturing images and an ultrasonic sensor
  - A Jetson nano microprocessor
    - Object detection using deep learning algorithm
    - Distance calculation for every object using triangulation algorithm (much like parallax you read in school)
    - Sending the output signal to the gripper using bluetooth

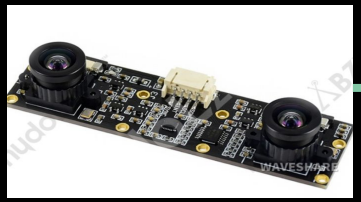
- **Gripper**

Attached on the walking stick it will create a 3D map using the hand as 2D plane and add the depth dimension using specialised cam - follower assisted pins protruding from the gripper.



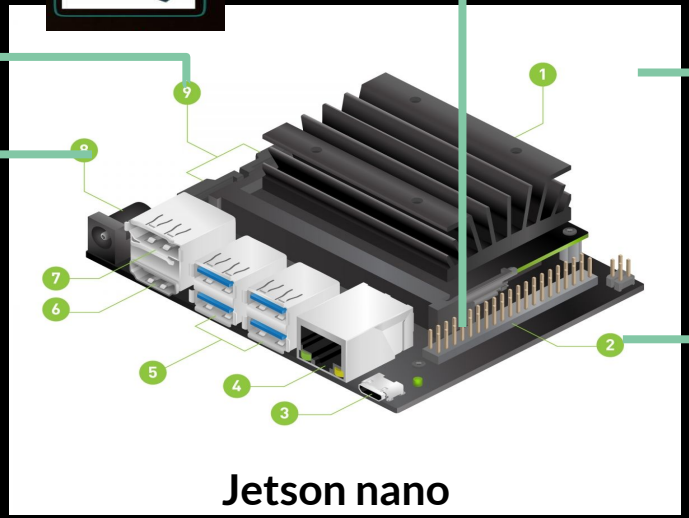
Circuit design

Centre For Innovation



Camera module is connected to MIPI CSI Camera connector.

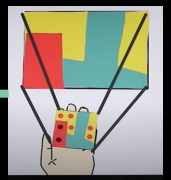
Ultrasonic sensor



Jetson nano

The Algorithm goes in an sd card which will go inside the microSD card slot.

Power Source



Output through cam-follower

Concepts

1. Object detection using YOLO
2. Distance calculation using Triangulation
3. Gripper module

IMAGE RECOGNITION PROGRESS

Centre For Innovation

Image Recognition model is applied using **YOLO v3 (You Only Look Once)** algorithm utilizing deep learning as it is the fastest compared to CNN and other methods

YOLO is trained on **COCO dataset** which has 80 different labels.

We apply a single neural network to the full image. This network divides the image into regions and predicts bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities.

SAHAAY
Social Innovation Club

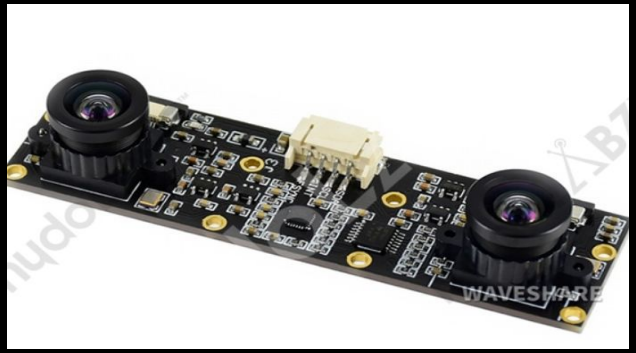
THE TRIANGULATION METHOD

Centre For Innovation

The Concept is just like our eyes !!

The two slightly different images are used to find the depth of the image ie, the distance of the object from the camera.

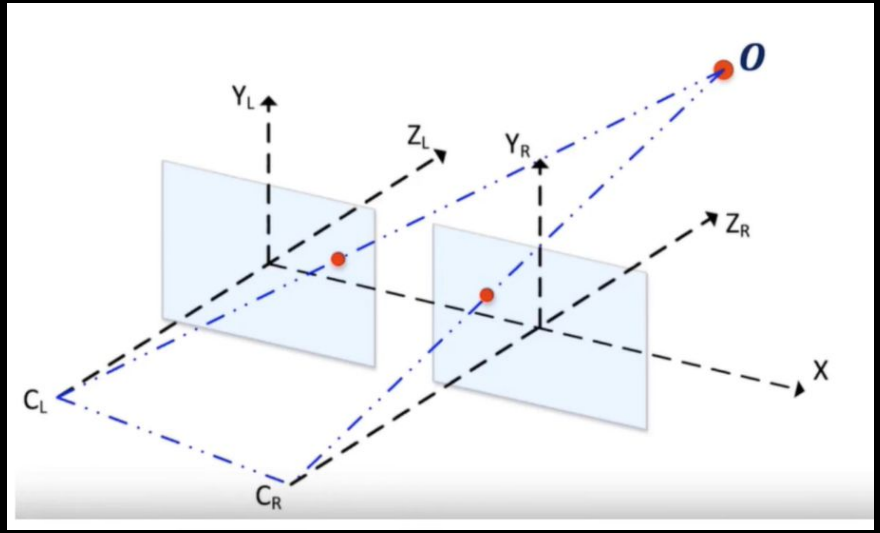
We use trigonometry to achieve this !!
Let us look in through each step.



SAHAAY
Social Innovation Club

Finding the Disparity Map

- From the two images from the cameras, we use trigonometry to find the disparity of pixels in each image
- Disparity is the difference in the location of the same 3d image from 2 different camera angles



Both cameras have the parallel X and Y axis

Finding the Disparity and Depth Map

- From the disparity values, We now find the depth(Z) parameter of each pixel using the formulas.
- Using the X,Y and Z coordinates of each pixel, we can create the intensity depth map.
- Finally, locate the coordinates of the detected object and find the depth… Eureka !

Middlebury 2001 data set.

$$Z = \frac{fb}{d}$$

$$X = \frac{Zx_L}{f}$$

$$Y = \frac{Zy_L}{f}$$

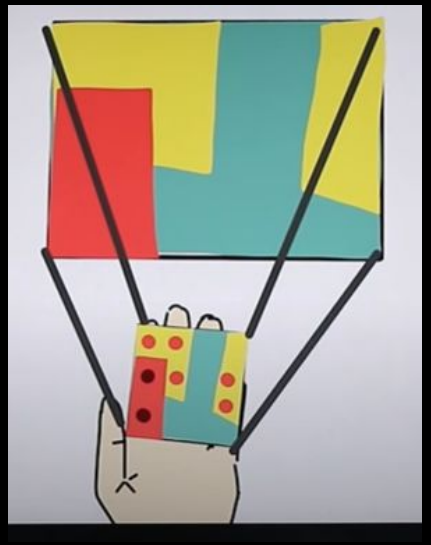
THE GRIPPER CONCEPT

The gripper is our way of displaying the output to the blind person.

We map the whole intensity-depth map detected by the Camera into his hand !!!!

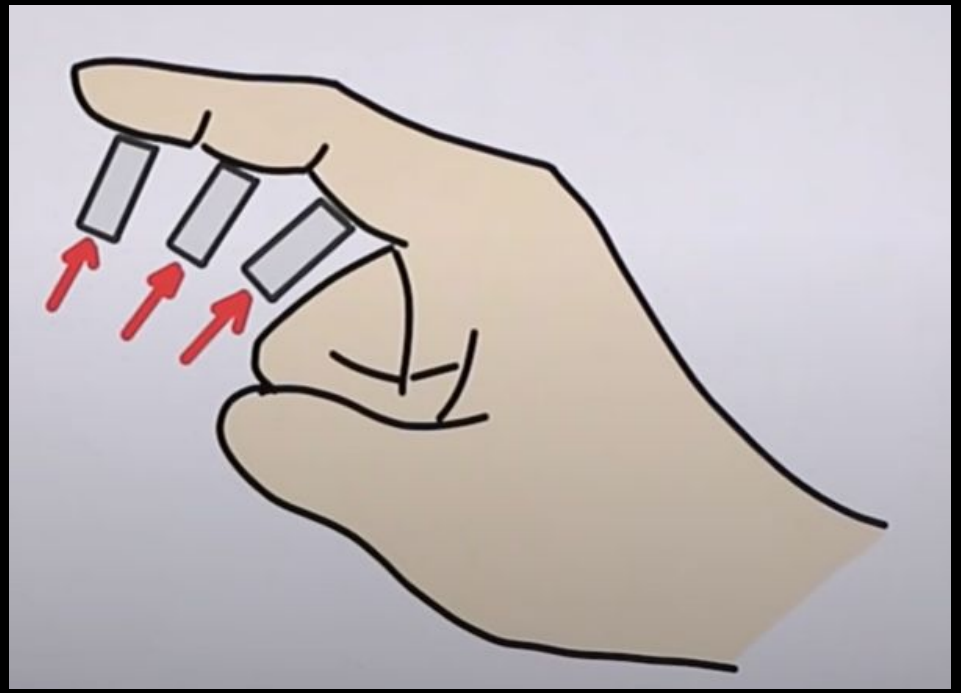
How we do it ??

Using tiny haptic sensors/linear actuators,
We push the parts of his fingers with more pressure on more nearer object and there by Conveying him the information about the obstacles



*For more info on gripper idea check this amazing youtube video **stuff made here** : https://youtu.be/8Au47gnXs0w*

THE GRIPPER CONCEPT



*For more info on gripper idea check this amazing youtube video **stuff made here** : https://youtu.be/8Au47gnXs0w*

THE GRIPPER CONCEPT : Simulation

Centre For Innovation

Progress

Completed Work

1. You Only Look Once (YOLO) Image Recognition Code
2. Camera Calibration and Triangulation Code
3. The Gripper Module and Prototype Circuitry

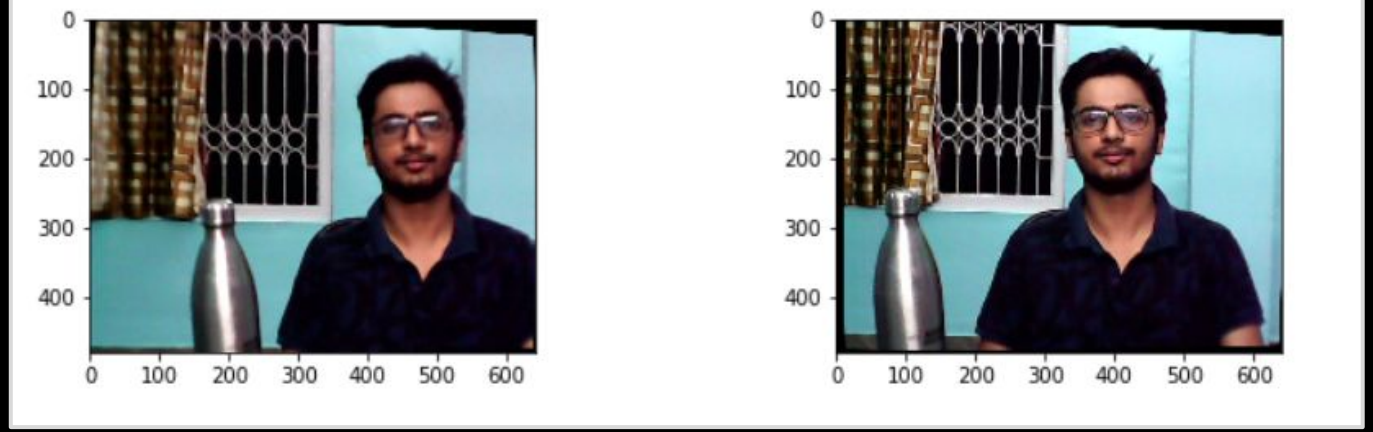
SAHAAY
Social Innovation Club

The Calibration code

```
In [6]: """
        Undistortion and Rectification part! Undistorts and Rectifies the images using the
        """
        leftMapX, leftMapY = cv2.initUndistortRectifyMap(
            K1, D1, R1, P1, (width, height), cv2.CV_32FC1
        )
        left_rectified = cv2.remap(
            leftFrame, leftMapX, leftMapY, cv2.INTER_LINEAR, cv2.BORDER_CONSTANT
        )
        rightMapX, rightMapY = cv2.initUndistortRectifyMap(
            K2, D2, R2, P2, (width, height), cv2.CV_32FC1
        )
        right_rectified = cv2.remap(
            rightFrame, rightMapX, rightMapY, cv2.INTER_LINEAR, cv2.BORDER_CONSTANT
        )

        print("After rectification: ")
        #plotting
        f, ax = plt.subplots(1,2, figsize=(12, 3))
        ax[0].imshow(cv2.cvtColor(left_rectified, cv2.COLOR_BGR2RGB))
        ax[1].imshow(cv2.cvtColor(right_rectified, cv2.COLOR_BGR2RGB))
        plt.show()
```

The Rectified Images

After rectification:



The Disparity Map code

```
In [7]: """
        disp_matrix = depth_map(gray_left, gray_right)
        """
        #We need grayscale for disparity map.
        gray_left = cv2.cvtColor(left_rectified, cv2.COLOR_BGR2GRAY)
        gray_right = cv2.cvtColor(right_rectified, cv2.COLOR_BGR2GRAY)

        disp_matrix = depth_map(right_rectified,left_rectified)  # Get the
        #print(disp_matrix)

        print("Depth map:")
        plt.figure(figsize=(15, 4.5))
        plt.imshow(cv2.cvtColor(disp_matrix, cv2.COLOR_BGR2RGB))
        plt.show()
```

Note: All the camera parameters, disparity matrix and calibrations are done wrt to the cheaper Cameras we ordered in the last tenure.

With better camera on the way, we can achieve better and accurate calibration, quality and hence the disparity & Distance

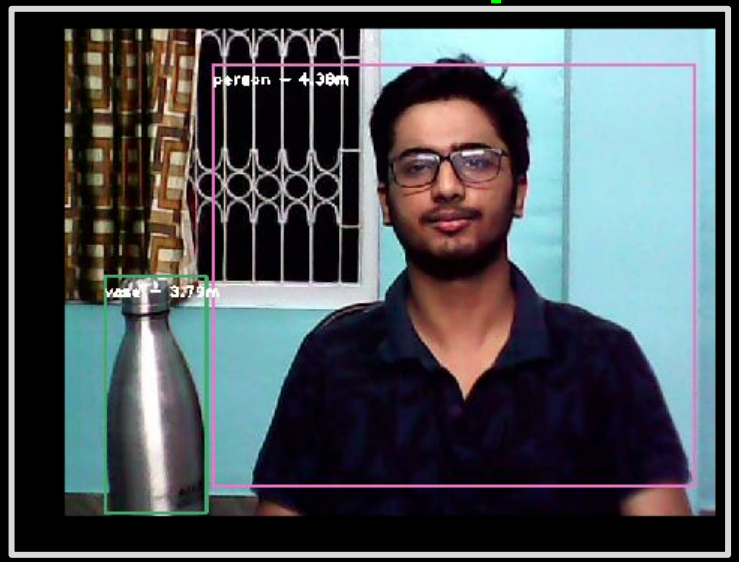
The Disparity Map

The YOLO and Distance code

```
"""
Model Input
"""
net.setInput(blob)
output_layers_names = net.getUnconnectedOutLayersNames() #finiding the unconnected layers
layerOutputs = net.forward(output_layers_names) #returns an array of output layers


"""
Detecting the rectangle with max confidence
"""
boxes = [] #stores the top left corner index of the box and the h and w
confidences = [] #storesthe max conf of the box
class_ids = [] #stores the index of max conf

for output in layerOutputs:
    for detection in output:
        scores = detection[5:] #first 5 are the dimensions of box and if object is present or not
        class_id = np.argmax(scores)
        conf = scores[class_id]
        # Setting confidence threshold = 0.1
        if conf > 0.1:
            cx = int(detection[0]*width) #center x of the box
            cy = int(detection[1]*height) #center y of the box
            w= int(detection[2]*width)
            h=int(detection[3]*height)

            bx = int(cx-w/2) #Left corner x
            by = int(cy-h/2) #Left corner y

            boxes.append([bx,by,w,h])
            confidences.append((float(conf)))
            class_ids.append(class_id)

#Non-max supression
indexes = cv2.dnn.NMSBoxes(boxes , confidences ,0.3 ,0.4)

#font and different colours
font = cv2.FONT_HERSHEY_PLAIN
colors = np.random.uniform(0,255 , size=(len(boxes),3))
```

```
"""
distance_matrix = (base offset x focal length)/disp_matrix
"""
distance_matrix = []
base =  0.07 # 1 / Q[3, 2] base offset (distance between the
focal = Q[2, 3] # Focal Length of the cameras

infi = 10e15;count =0
for i in range(disp_matrix.shape[0]):
  for j in range(disp_matrix.shape[1]):
    if disp_matrix[i][j] == 0:
      disp_matrix[i][j] = (1/infi)

distance_matrix = (base*focal)/disp_matrix
print(distance_matrix)
```

The final output



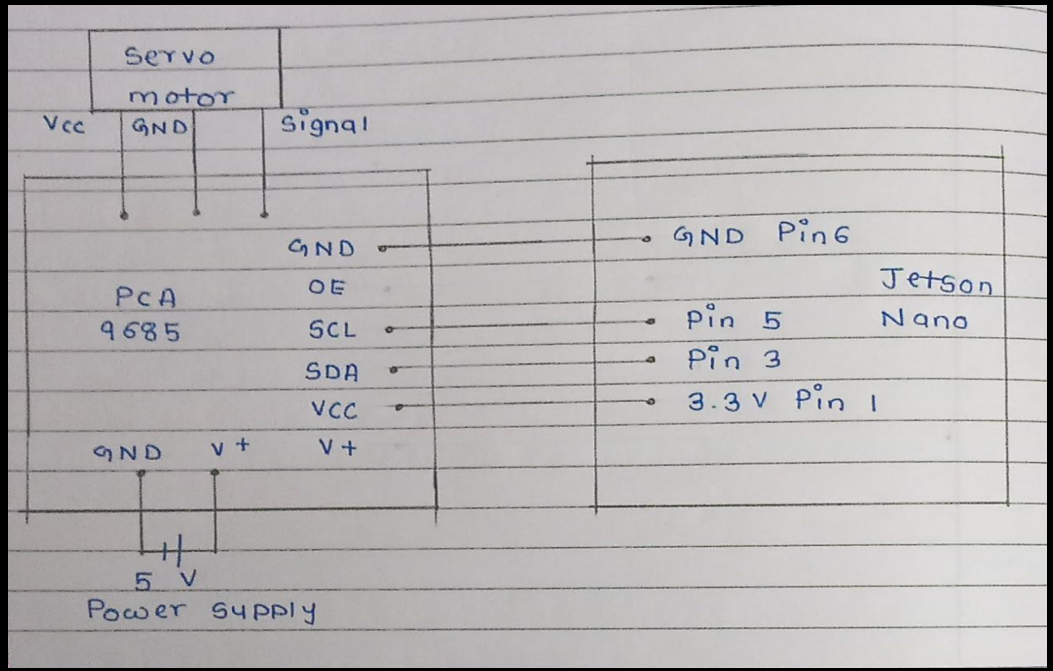
THE GRIPPER MODULE : Progress and Future Plans

- We have learnt I/O and Pulse Width Modulation on Jetson Nano and have circuits for the same on paper.
- We are now working on combining our knowledge to deliver instructions and commands to 4 different servo motors for rotating at different speeds.

FUTURE PLANS

- We plan to use bluetooth module on Jetson Nano and arduino in order to connect them wirelessly. Jetson Nano being attached to to user's waist and arduino to gripper module.

THE GRIPPER MODULE : Circuit Diagram



To control the servo motors using Jetson Nano we will use the PCA 9685 board.
The advantage of using this board is that multiple servo motors can be connected through the channels on the board .One such channel is shown in the circuit diagram.
This is not possible directly with the Jeston Nano as it has limited PWM pins.

THE GRIPPER MODULE : Code for Jetson Nano

Taking inputs

```
In [ ]: #Pins are named alternatively on jetson nano. Odd pins are outward. Even pins are inward

In [ ]: ## TAKING INPUTS

In [ ]: #importing library for talking to board. RPi library works fine with  jetson nano as well
import RPi.GPIO as GPIO

#sets jetson nano board configuration of pins as default
GPIO.setmode(GPIO.BOARD)

#accepting input at pin 15 on jetson nano
GPIO.setup(15,GPIO.IN)

#setting variable x as input for pin 15
x = GPIO.input(15)

#outputting x will give 0/1 depending on weather the input is ON OR OFF
#x

#Will clear 15th pin's configuration. i.e pin 15 will no more accept any input
GPIO.cleanup(15)

#will clear the complete board. Like a reset button
GPIO.cleanup()
```

Sending output

```
In [ ]: ## OUTPUTS

In [ ]: #importing library for talking to board. RPi library works fine with  jetson nano as well
import RPi.GPIO as GPIO

#sets jetson nano board configuration of pins as default
GPIO.setmode(GPIO.BOARD)

## difference
#sending output via pin 7 on jetson nano
GPIO.setup(7,GPIO.OUT)

#Will set output as true
GPIO.output(7,True)

#Will set output as false
GPIO.output(7,False)

#Will clear 7th pin's configuration. i.e pin 15 will no more accept any input
GPIO.cleanup(7)

#will clear the complete board. Like a reset button
GPIO.cleanup()
```

THE GRIPPER MODULE : Code for Jetson Nano

Controlling Servos

In [ ]:
```
#in servoControl.py file
from adafruit_servokit import ServoKit
myKit = ServoKit(channels=16)
myKit.servo[0].angle = 0 #will change motor's movement on running program
myKit.servo[0].angle = 180 #will change motor's movement on running program
myKit.servo[0].angle = 0 #will change motor's movement on running
```

In [ ]:
```
#in servoControl.py file
import time
from adafruit_servokit import ServoKit
myKit = ServoKit(channels=16)
#will rotate the motor
for i in range(0,180,1):
    myKit.servo[0].angle=i
    time.sleep(.1)

for i in range(180,0,-1):
    myKit.servo[0].angle=i
    time.sleep(.1)
```

In [ ]:
```
#Pins are named alternatively on jetson nano. Odd pins are outward. Even pins are inward
```

In [ ]:
```
## TAKING INPUTS
```

In [ ]:
```
#importing library for talking to board. RPi library works fine with  jetson nano as well
import RPi.GPIO as GPIO

#sets jetson nano board configuration of pins as default
GPIO.setmode(GPIO.BOARD)
```

Setting up PCA9685 for PWM

]:
```
#try this link to set up config for PWM
#https://www.youtube.com/watch?v=eImDQ0PVu2Y&list=PLGs0VKk2DiYxP-ElZ7-QXIERFFPkOuP4_&index=59

import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BOARD)

GPIO.setup(33,GPIO.OUT)
my_pwm = GPIO.PWM(33,100)
my_pwm.start(5)
my_pwm.ChangeDutyCycle(1)
my_pwm.ChangeDutyCycle(99)

#in case any errors come do
#sudo reboot
```

Timeline

Work done (May'21 - Oct'21)

1. Implemented YOLO algorithm to detect objects from the live stereo camera feed.
2. Camera calibration and Triangulation code to find out the distance of detected objects
3. Did the initial setup of Jetson nano (writing image to microSD & network connection)
4. 3D model of the gripper module.
5. Prepared schematic diagram of jetson nano and gripper circuit (PCA9685 board).
6. I/O code for controlling servo motors through jetson nano.

Future goals (Nov'21 - May'22)

1. Test the YOLO, Triangulation, Gripper codes on Jetson Nano
2. Figure out how to wirelessly send the signal to the gripper circuit from the jetson nano
3. Additional features - Audio Translation using NLP, GPS integration

Our Journey and The ENDGAME

Centre For Innovation

The Pandemic hasn't stopped us from designing, learning and finally building the first prototype.

Once we have our working model, we plan to improve the accuracy and add features like navigation, smartphone compatibility and our very own Google voice Assistant.

and **Someday** see the invention bringing smile in other lives.

SAHAAY
Social Innovation Club

Centre For Innovation

Thank You

- Team Vision

SAHAAY
Social Innovation Club