A

PROJECT REPORT

ON

"AgroWeeder"

is a partial fulfilment of the Diploma in **Information Technology** of
Maharashtra State Board of Technical Education, Pune during the academic year 2019-2020

By

Yash Sudhir Shirke
(170110015)
Udayraj Sambaji Gawade
(1807110399)

Guide by

Mrs. Jadhav D.S.

DEPARTMENT OF INFORMATION TECHNOLOGY

JSPM's JAYAWANTRAO SAWANT POLYTECHNIC

HADAPSAR, PUNE-411028.

(Academic Year: 2019-2020)

ACKNOLEDGEMENT

It is matter of great pleasure for me to submit this seminar report on AgroWeeder a part of curriculum for award of Maharashtra education of technical education's Diploma in information technology from JSPM's Jayawantrao Sawant Polytechnic.

Firstly, I would like to express my gratitude to my guide Mrs Deepali Suhas Jadhav, for their inspiration, adroit guidance, constant supervision, direction and discussion in successful completion of this Seminar.

I am thankful to Head of Department Prof. R. P. BEMBDE, for his valuable support and guidance.

I am thankful to my Principal Dr. S. M. DEOKAR and to all our staff members who encouraged me to do this Seminar.

Also, I extend my thanks to all my colleagues those who have helped me directly or indirectly in completion of this seminar and last but not least, I am thankful to my parents, who had inspired me with their blessings.

SUBMITED BY-

Mr. Yash Sudhir Shirke (1707110015)

Mr. Udayraj Sambaji Gawade (1807110399)

ABSTRACT

The pi-camera module along with an ultrasonic sensor is used to provide necessary data from the real world to the AgroWeeder which would then pass the data on to the raspberry-pi. The multipurpose Agroweeder is capable of removing the weeds from the farm, tilling of land, seeding, harvester, and many more thing can be perform with that. This will help farmers to reduce the labour cost, time, and fuel consummation.

The multipurpose Agroweeder is fully automated. It uses the AI and ML. using image processing we can give the security as well as object detection.

INDEX

List of Tables

List of Figures

Introduction

Identification and Justification of Problem:

Replacing human labour with automation is growing trend across the entire world.

According to survey, the world population is increasing tremendously. With increase in population there is need of increase in food production. Agriculture sector becomes inefficient due to traditional methods of farming.

To improve the quality and quantity of agriculture products and to minimize the human efforts, we need to make use of intelligent and automatic machinery for agricultural activities. So as modern citizens of country, we have deigned AgroWeeder: Self power weeder based on Raspberry Pi for farmers.

Village Study:

In our country farmers still use traditional methods of farming in an era where across the globe all other countries are using modern technology and equipment's. Due to lack of modern agriculture techniques and limited access to technology, farmers are facing huge loss in agriculture which results in number of farmer's suicides year by year. Also, due to use of traditional methods and equipment's in farming there is less crop production and cost of labour is high.

Problem Statement

AgroWeeder: Self power weeder based on Raspberry Pi.

Description of Problem:

As the AgroWeeder: Self power weeder working is based on Raspberry Pi; it is termed as Self power weeder. This design definitely becomes boom in agriculture sector. This will help in reducing the cost of labour as well as minimizes the human efforts.

Description of Innovative Solution:

We are implementing this prototype using Raspberry-pi as a processing chip. Central Controller that is raspberry-pi is mounted on the power weeder. The pi camera is placed at the top of the power weeder. The ultrasonic sensors would be placed on the front side of the power weeder. The motor-driver ICs are used for the operation of motors and the motion of the power weeder.

Thus, our system will minimize human efforts by automation in agriculture equipment's.

Literature Review

Literature Review

Name of publication	Year	Advantages	Disadvantages
International Research Journal of Engineering and Technology (IRJET)	Mar 2019		
International Research Journal of Engineering and Technology (IRJET)	Apr-2018		
Journal of Emerging Technologies and Innovative Research	September 2018		
International Journal of Advance Research and Innovative Ideas in Education International Journal of Advance Research and Innovative Ideas in Education	2017		
IEEE Recent Advances in Intelligent Computational Systems (RAICS)	10-12 December 2015		

Requirement Analysis

Basic Requirements: -

For implementing this project, we need to install the python 3 and the dependencies in our systems which will help the code to execute successfully

Following are the dependencies to be installed they are

- Tensorflow
- NumPy
- Opencv-python

Hardware Requirements: -

- Raspberry pi 3b+
- Pi camera :- 5mp
- Arduino uno R3
- Motor driver
- Power bank
- 12v battery
- Jumper cables

Software Requirements: -

- Operating System: - Raspbian

  Editor: -Python2.7 IDE and Arduino IDE

2D Diagram

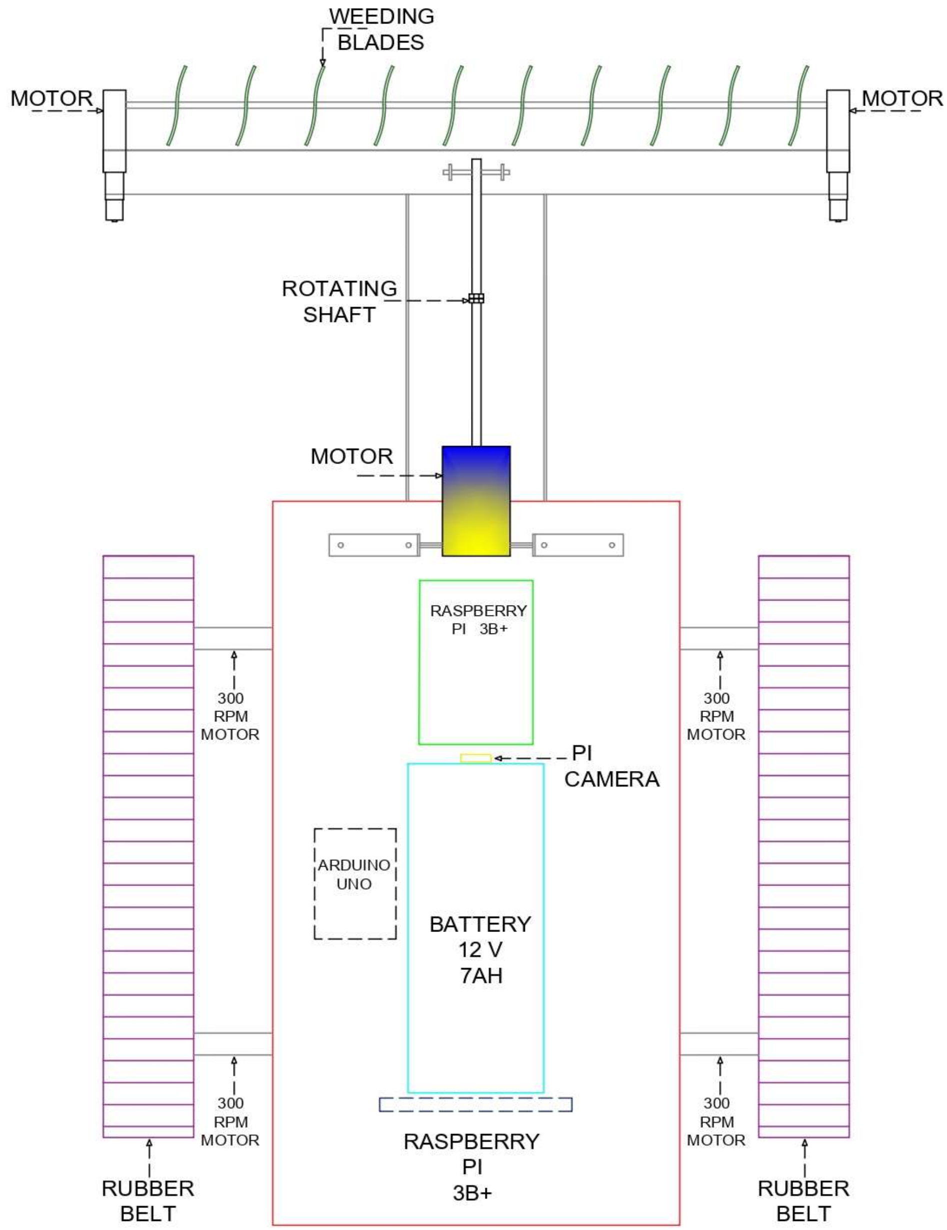
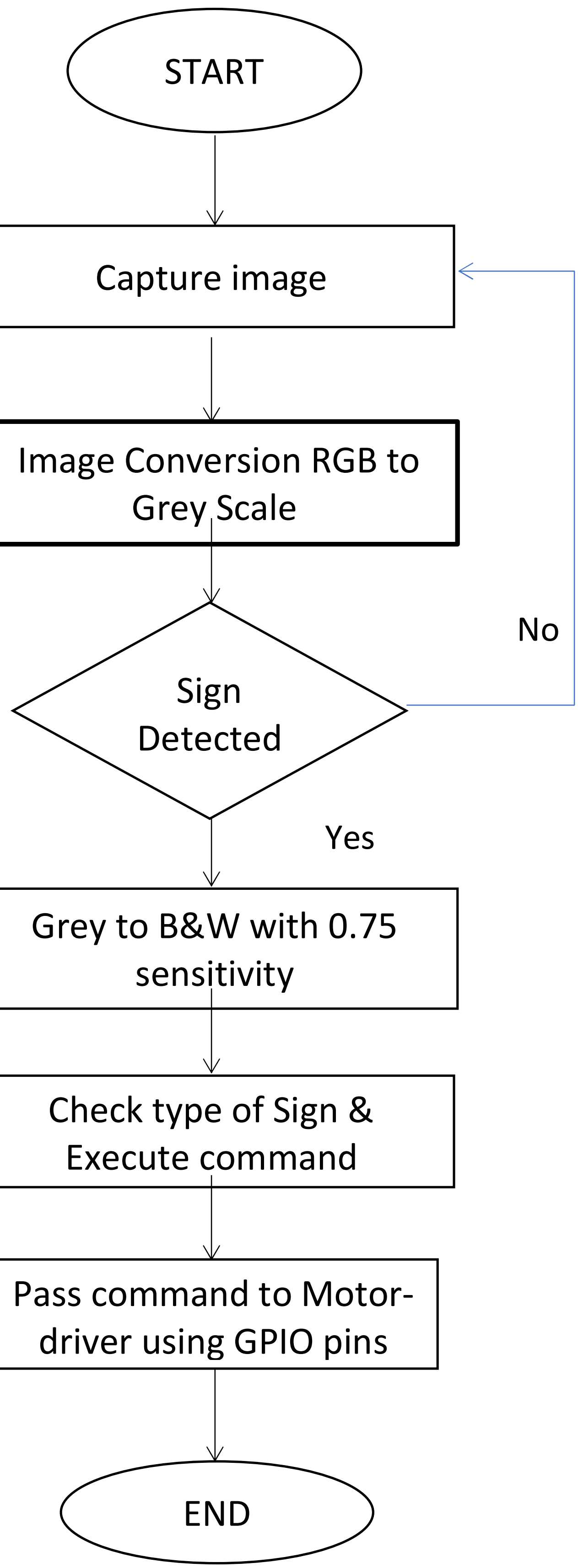
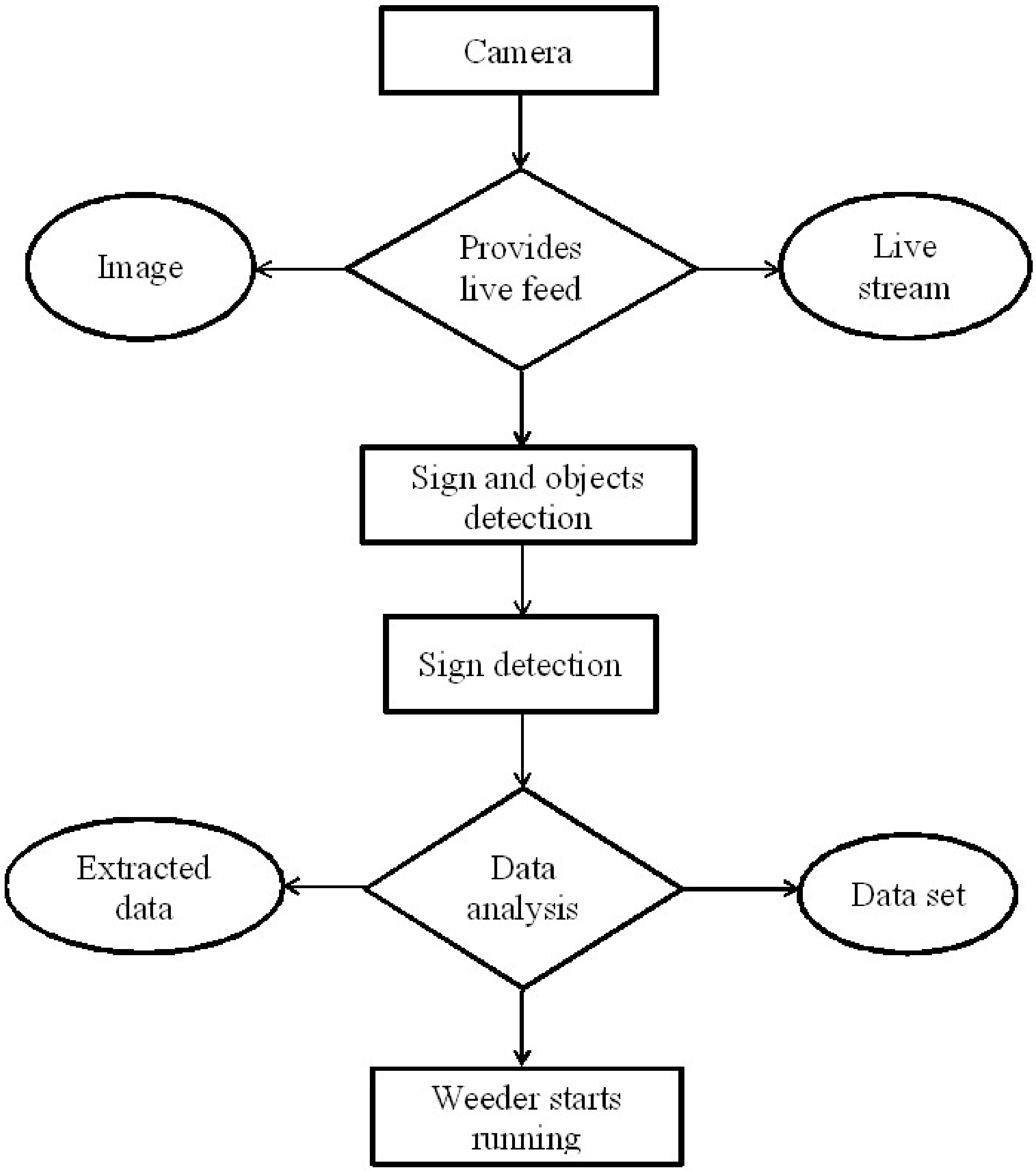


Fig. 2d diagram of AgroWeeder

Flow chart



START
Capture image
Image Conversion RGB to Grey Scale
Sign Detected
No
Yes
Grey to B&W with 0.75 sensitivity
Check type of Sign & Execute command
Pass command to Motor-driver using GPIO pins
END


Camera
Provides live feed
Image
Live stream
Sign and objects detection
Sign detection
Data analysis
Extracted data
Data set
Weeder starts running

Working

- The Agroweeder is a farm based machine which is used for tilling the land and the removing weeds from the land.
- The Agroweeder works on a simple small board processor raspberry pi 3b+. When the Agroweeder gets start the pi camera captures the live streaming of the field and sends the information to the raspberry pi.
- Then the raspberry pi detects the signs.
- Then raspberry pi process the video, and sends the command to the arduino using GPIO pins.
- Then the arduino process the command and send the command to both the motor drivers.
- The first motor is use for running of Agroweeder.
- And the second one is used for controlling the weeding tools which is fixed in front of the Agroweeder.

Target Beneficiary group:

This system is beneficial for all the farmers for weeding purpose. Instead of working with machine by physically presenting, we can operate the machine by seating at one place. Mobility problems can be solved using this system.

Expected Outcomes/Outputs:

Low Cost power weeder contains various facilities like sign detection, weed removing from gardens and lawns. It can also be used for digging the soil.

Implemented code

```
import cv2
import numpy as np
from picamera.array import PiRGBArray
from picamera import PiCamera
import time
import RPi.GPIO as GPIO
from imutils.perspective import four_point_transform
#from imutils import contours
#import imutils

cameraResolution = (320, 240)

# initialize the camera and grab a reference to the raw camera capture
camera = PiCamera()
camera.resolution = cameraResolution
camera.framerate = 32
camera.brightness = 60
camera.rotation = 360
rawCapture = PiRGBArray(camera, size=cameraResolution)

# allow the camera to warmup
time.sleep(2)

def findTrafficSign():
    '''
    This function find blobs with blue color on the image.
    After blobs were found it detects the largest square blob, that
must be the sign.
    '''

    # define range HSV for blue color of the traffic sign
    lower_blue = np.array([90,80,50])
    upper_blue = np.array([110,255,255])

    while True:
        # The use_video_port parameter controls whether the camera's
image or video port is used
        # to capture images. It defaults to False which means that the
camera's image port is used.
        # This port is slow but produces better quality pictures.
        # If you need rapid capture up to the rate of video frames, set
this to True.
        camera.capture(rawCapture, use_video_port=True, format='bgr')

        # At this point the image is available as stream.array
        frame = rawCapture.array

        frameArea = frame.shape[0]*frame.shape[1]

        # convert color image to HSV color scheme
        hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

        # define kernel for smoothing
        kernel = np.ones((3,3),np.uint8)
        # extract binary image with active blue regions
        mask = cv2.inRange(hsv, lower_blue, upper_blue)
```

```
        # morphological operations
        mask = cv2.morphologyEx(mask, cv2.MORPH_OPEN, kernel)
        mask = cv2.morphologyEx(mask, cv2.MORPH_CLOSE, kernel)

        # find contours in the mask
        cnts = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)[-2]

        # defite string variable to hold detected sign description
        detectedTrafficSign = None

        # define variables to hold values during loop
        largestArea = 0
        largestRect = None

        # only proceed if at least one contour was found
        if len(cnts) > 0:
            for cnt in cnts:
                # Rotated Rectangle. Here, bounding rectangle is drawn
with minimum area,
                # so it considers the rotation also. The function used
is cv2.minAreaRect().
                # It returns a Box2D structure which contains following
detals -
                # ( center (x,y), (width, height), angle of rotation ).
                # But to draw this rectangle, we need 4 corners of the
rectangle.
                # It is obtained by the function cv2.boxPoints()
                rect = cv2.minAreaRect(cnt)
                box = cv2.boxPoints(rect)
                box = np.int0(box)

                # count euclidian distance for each side of the
rectangle
                sideOne = np.linalg.norm(box[0]-box[1])
                sideTwo = np.linalg.norm(box[0]-box[3])
                # count area of the rectangle
                area = sideOne*sideTwo
                # find the largest rectangle within all contours
                if area > largestArea:
                    largestArea = area
                    largestRect = box

        if largestArea > frameArea*0.02:
            # draw contour of the found rectangle on  the original
image
            cv2.drawContours(frame,[largestRect],0,(0,0,255),2)

            # cut and warp interesting area
            warped = four_point_transform(mask, [largestRect][0])

            # show an image if rectangle was found
            #cv2.imshow("Warped", cv2.bitwise_not(warped))

           # use function to detect the sign on the found rectangle
            detectedTrafficSign = identifyTrafficSign(warped)

            #print(detectedTrafficSign)
```

```
                # write the description of the sign on the original image
                cv2.putText(frame, detectedTrafficSign, (100, 100),
cv2.FONT_HERSHEY_SIMPLEX, 0.65, (0, 255, 0), 2)

        # show original image
        cv2.imshow("Original", frame)

        # clear the stream in preparation for the next frame
        rawCapture.truncate(0)

        # if the `q` key was pressed, break from the loop
        if cv2.waitKey(1) & 0xFF is ord('q'):
            cv2.destroyAllWindows()
            print("Stop programm and close all windows")
            break

def identifyTrafficSign(image):
    '''
    In this function we select some ROI in which we expect to have the
sign parts. If the ROI has more active pixels than threshold we mark it
as 1, else 0
    After path through all four regions, we compare the tuple of ones
and zeros with keys in dictionary SIGNS_LOOKUP
    '''

    # define the dictionary of signs segments so we can identify
    # each signs on the image
    SIGNS_LOOKUP = {
        (1, 0, 0, 1): 'Turn Right', # turnRight
        (0, 0, 1, 1): 'Turn Left', # turnLeft
        (0, 1, 0, 1): 'Move Straight', # moveStraight
        (1, 0, 1, 1): 'Turn Back', # turnBack
    }

    THRESHOLD = 150

    image = cv2.bitwise_not(image)
    # (roiH, roiW) = roi.shape
    #subHeight = thresh.shape[0]/10
    #subWidth = thresh.shape[1]/10
    (subHeight, subWidth) = np.divide(image.shape, 10)
    subHeight = int(subHeight)
    subWidth = int(subWidth)

    # mark the ROIs borders on the image
    #cv2.rectangle(image, (subWidth, 4*subHeight), (3*subWidth,
9*subHeight), (0,255,0),2) # left block
    #cv2.rectangle(image, (4*subWidth, 4*subHeight), (6*subWidth,
9*subHeight), (0,255,0),2) # center block
    #cv2.rectangle(image, (7*subWidth, 4*subHeight), (9*subWidth,
9*subHeight), (0,255,0),2) # right block
    #cv2.rectangle(image, (3*subWidth, 2*subHeight), (7*subWidth,
4*subHeight), (0,255,0),2) # top block

    # substract 4 ROI of the sign thresh image
    leftBlock = image[4*subHeight:9*subHeight, subWidth:3*subWidth]
    centerBlock = image[4*subHeight:9*subHeight, 4*subWidth:6*subWidth]
```

```
    rightBlock = image[4*subHeight:9*subHeight, 7*subWidth:9*subWidth]
    topBlock = image[2*subHeight:4*subHeight, 3*subWidth:7*subWidth]

    # we now track the fraction of each ROI
    leftFraction =
np.sum(leftBlock)/(leftBlock.shape[0]*leftBlock.shape[1])
    centerFraction =
np.sum(centerBlock)/(centerBlock.shape[0]*centerBlock.shape[1])
    rightFraction =
np.sum(rightBlock)/(rightBlock.shape[0]*rightBlock.shape[1])
    topFraction =
np.sum(topBlock)/(topBlock.shape[0]*topBlock.shape[1])

    segments = (leftFraction, centerFraction, rightFraction,
topFraction)
    segments = tuple(1 if segment > THRESHOLD else 0 for segment in
segments)

    cv2.imshow("Warped", image)

    if segments in SIGNS_LOOKUP:
        print (SIGNS_LOOKUP[segments])

        if SIGNS_LOOKUP[segments] == 'Move Straight':
            return (forward(2))
        elif SIGNS_LOOKUP[segments] == 'Turn Back':
            return (uturn(2))
        elif SIGNS_LOOKUP[segments] == 'Turn Right':
            return (right(5))
        elif SIGNS_LOOKUP[segments] == 'Turn Left':
            return (left(5))


    else:
        return None

def init():
    GPIO.setwarnings(False)
    GPIO.setmode(GPIO.BCM)
    #GPIO.setmode(GPIO.BOARD)
    GPIO.setup(7,GPIO.OUT)
    GPIO.setup(8,GPIO.OUT)
    GPIO.setup(9,GPIO.OUT)
    GPIO.setup(10,GPIO.OUT)



def left(tf):
    print('abc')
    init()
    GPIO.output(7,GPIO.LOW)
    GPIO.output(8,GPIO.HIGH)
    GPIO.output(9,GPIO.HIGH)
    GPIO.output(10,GPIO.LOW)
    time.sleep(tf)
    GPIO.cleanup()

def right(tf):
```

```
    print('abc')
    init()
    GPIO.output(7,GPIO.HIGH)
    GPIO.output(8,GPIO.LOW)
    GPIO.output(9,GPIO.LOW)
    GPIO.output(10,GPIO.HIGH)
    time.sleep(tf)
    GPIO.cleanup()

def uturn(tf):
    print('ddd')
    init()
    GPIO.output(7,GPIO.LOW)
    GPIO.output(8,GPIO.HIGH)
    GPIO.output(9,GPIO.LOW)
    GPIO.output(10,GPIO.HIGH)
    time.sleep(tf)
    GPIO.cleanup()

def forward(tf):
    print('ccc')
    init()
    GPIO.output(7,GPIO.LOW)
    GPIO.output(8,GPIO.HIGH)
    GPIO.output(9,GPIO.LOW)
    GPIO.output(10,GPIO.HIGH)
    time.sleep(tf)
    GPIO.cleanup()

def back(tf):
    print('sss')
    init()
    GPIO.output(7,GPIO.HIGH)
    GPIO.output(8,GPIO.LOW)
    GPIO.output(9,GPIO.HIGH)
    GPIO.output(10,GPIO.LOW)
    time.sleep(tf)
    GPIO.cleanup()


def main():
    findTrafficSign()
    #identifyTrafficSign()




if __name__ == '__main__':
    main()
```

Future Scope

- To enhance the performance and ensure practicality of the car, the efficiency and processor speed need to be raised.
- A camera of better resolution would also be required as the scenes keep changing rapidly in the real world.
- To obtain precise and accurate results LASER sensors are required. These kinds of sensors are pretty much expensive. In future works, if the laser sensors named "LIDAR" is used; surely the results will have very less errors.

Conclusion

- The autonomous AgroWeeder would surely prove out to be a boon in the automation industry and would be preferred over many traditional techniques.
- They could be used for patrolling and capturing the images of the offender. As they won't require any drivers.
- Therefore our proposed idea will help farmers in saving their time and can earn more amount of money. They can operate our Agroweeder from home as well.

Reference

[1] https://howtomechatronics.com/tutorials/arduino/arduino-dc-motor-control-tutorial-l298n-pwm-h-bridge/

[2] https://mytectutor.com/dc-motor-control-using-l298n-motor-driver/

[3] https://www.cmi.ac.in/~madhavan/courses/prog2-2015/docs/python-3.4.2-docs-html/tutorial/index.html

[4] https://en.wikipedia.org/wiki/Arduino_IDE

[5] IEEE, 2014. www.ieee.org. [Online] Available at: http://www.ieee.org/about/news/2014/14_july_2014.html [Zugriff am 29 April 2015].

[6] Broggi, A. et al., 2013. Extensive Tests of Autonomous Driving Technologies. IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, 14(3).

[7] J.M.A. Alvarez, A.M. Lopez & R. Baldrich, IlluminantInvariant Model-Based Road Segmentation. Intelligent Transportation Systems, IEEE Transactions on, 12, 2008, pp 184–193.Bar Hillel, R. Lerner, D. Levi, & G. Raz. Recent progress in road and lane detection: a survey. Machine Vision and Applications, Feb. 2012, pp. 727–745

[8] Shahzeb Ali Department of Electronic Engineering, Mehran University of Engineering & Technology, Jamshoro, 2016 IEEE.

[9] K. R. Memon, S. Memon, B. Memon, A. R. Memon, and M. Z. A. S. Syed, "Real time Implementation of Path planning Algorithm with Obstacle Avoidance for Autonomous Vehicle," in 3rd 2016 International Conference on Computing for Sustainable Global Development", New Delhi, India, 2016 .