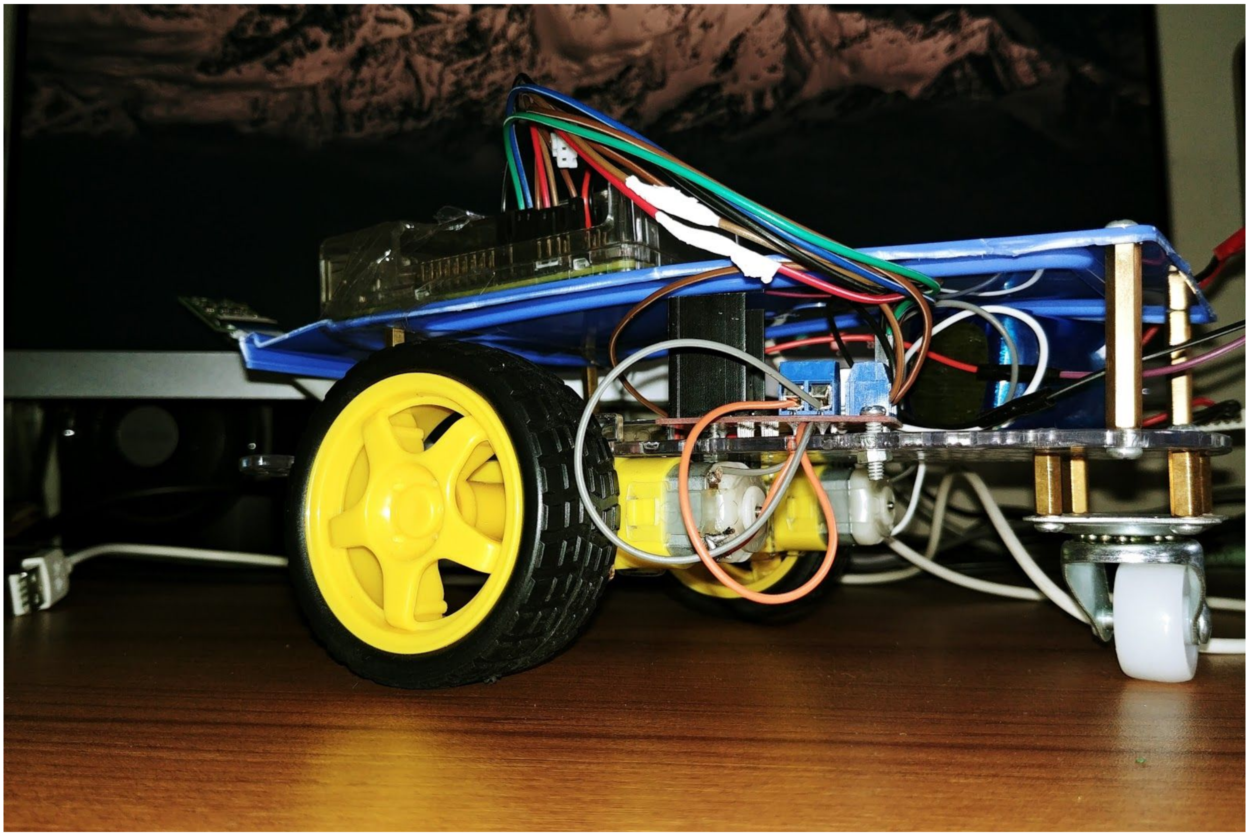


Smart Robot

07.08.2019

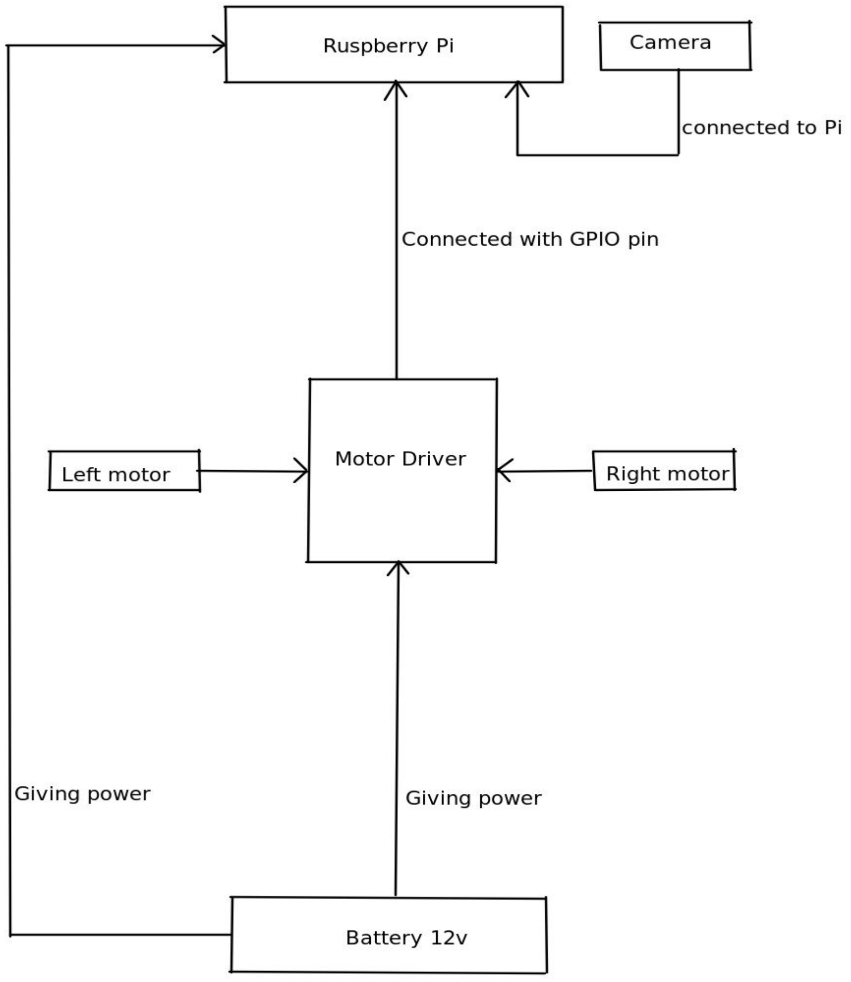
Name	ID
Faisal Abdullah	17-33471-1
Fokrul Islam Bhuiyan	17-33535-1
Hedaetul ISlam	17-33554-1

Project Description

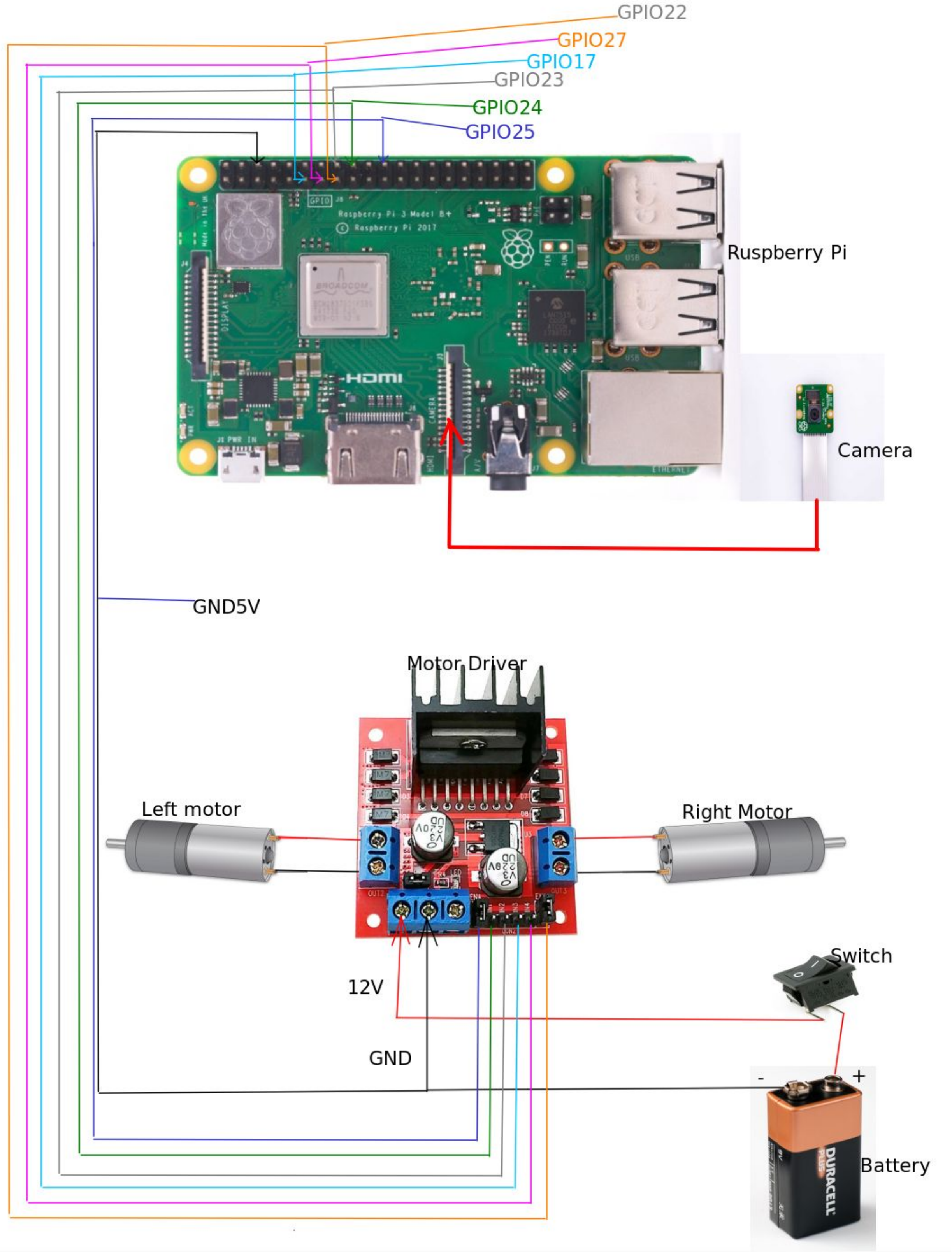
We all know the Raspberry Pi is a wonderful developing platform with its high computational power and development options it can work out wonders in hands of electronics hobbyists or students. So, we supposed to build an image-processing robot using Raspberry Pi. This robot is capable of travel one point to another point. What the robot do is that it clicks picture of a view from upper and detects all the obstacles in the picture and also calculate the distance of objects from one to another. Then the robot can travel given source to destination by following the shortest path. But we could not reach that much. In the project, we build an image-processing line follower robot and try to add shortest path functionality. We used the Pi Camera to capture a video stream and applied computer vision algorithms to follow the line. Using camera has some big advantages over sensors like sensor follow the line using edge detection rather than grayscale thresholding which is virtually immune for shadows and grey zones in the image. The robot will work in such a way that first it will capture the frame then it will define the ROI(region of interest) after that identify all of the black regions in this ROI(region of interest) and accordingly will follow the line.

Technical

Block Diagram



Circuit Diagram



System Description

OS :

Raspbian Buster with Desktop and recommended software (LINUX)

Version: july 2019

Kernel Version: 4.19

Programming Language :

The programming language we have used is Python.

Library :

In our project, we use some python's library and Raspberry Pi's library:

1. RPi.GPIO - use for Raspberry Pi's pin.
2. Picamera - use for Raspberry Pi's camera.
3. Opencv - use for image processing.
4. Numpy - use for array processing.
5. Time - use for motor delay.

Modification/Implementation

In the existing system, line following robot is made using the infrared sensors, in some project image-processing used too. So, our project is to modify this robot which can find shortest path from source to destination. This project can be done in two ways. One is image-processing and another is sensor system(IR sensor). We intend to do this by using image-processing. The robot will follow lines with video stream and travel all paths then finds out shortest path. The challenging part of that project is to find the shortest path between two points.

Source

I. Robot Assembly

https://www.youtube.com/watch?v=3a-bE1VlaU8

II. Motor controller RPI

https://www.youtube.com/watch?v=bNOlimnWZJE&t=159s

III. Camera setup

https://www.youtube.com/watch?v=xA9rzq5_GFM&t=176s

IV. Open CV

http://jollejolles.com/easy-install-opencv-for-python-on-mac-ubuntu-and-raspberry-pi/

V. Image Processing

https://www.udemy.com/image-processing-on-raspberry-pi/

VI. Shortest Path

https://www.youtube.com/watch?v=5yJDyflv3VA

Code Segment

Line Following Using Image-processing:

```
import RPi.GPIO as GPIO
from time import sleep

##################camera
from picamera.array import PiRGBArray
import time
import cv2
import picamera
import numpy as np

#l####################eft motor
in1 = 24
in2 = 23
en = 25
temp1=1
#####################rightmotor
in3 = 17
in4 = 27
enB = 22

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
#####################left motor
```

```
GPIO.setup(in1,GPIO.OUT)
GPIO.setup(in2,GPIO.OUT)
GPIO.setup(en,GPIO.OUT)
p=GPIO.PWM(en,1000)

#####################rightmotor
GPIO.setup(in3,GPIO.OUT)
GPIO.setup(in4,GPIO.OUT)
GPIO.setup(enB,GPIO.OUT)
p2=GPIO.PWM(enB,1000)

####################motor start
p.start(15)
p2.start(15)
print("\n")
print("working properly by camera.....")
print("\n")



#camera
camera = picamera.PiCamera()
camera.resolution =(192,108)
camera.framerate = 20
rawCapture = PiRGBArray(camera,size=(192,108))
time.sleep(0.1)



for frame in camera.capture_continuous(rawCapture, format="bgr", use_video_port=True):
    # Display camera input
```

```
image = frame.array
cv2.imshow('img',image)

# Create key to break for loop
key = cv2.waitKey(1) & 0xFF

# convert to grayscale, gaussian blur, and threshold
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
blur = cv2.GaussianBlur(gray,(5,5),0)
ret,thresh1 = cv2.threshold(blur,100,255,cv2.THRESH_BINARY_INV)



# Erode to eliminate noise, Dilate to restore eroded parts of image
mask = cv2.erode(thresh1, None, iterations=2)
mask = cv2.dilate(mask, None, iterations=2)

# Find all contours in frame
something, contours, hierarchy =
cv2.findContours(mask.copy(),1,cv2.CHAIN_APPROX_NONE)



if len(contours) > 0:
        # Find largest contour area and image moments
    c = max(contours, key = cv2.contourArea)
    M = cv2.moments(c)

    # Find x-axis centroid using image moments
    cx = int(M['m10']/M['m00'])
    cy=int(M['m01']/M['m00'])
```

```
if cx >= 150 and cy>40 and cy<90:
    GPIO.output(in1,10)
    GPIO.output(in2,GPIO.LOW)
    GPIO.output(in3,GPIO.LOW)
    GPIO.output(in4,4)
    time.sleep(.05)

if cx < 150 and cx >= 130 and cy>40 and cy<90:
    GPIO.output(in1,10)
    GPIO.output(in2,GPIO.LOW)
    GPIO.output(in3,GPIO.LOW)
    GPIO.output(in4,GPIO.LOW)
    time.sleep(.05)

if cx < 130 and cx > 70 and cy>40 and cy<90:
    GPIO.output(in1,GPIO.HIGH)
    GPIO.output(in2,GPIO.LOW)
    GPIO.output(in3,GPIO.HIGH)
    GPIO.output(in4,GPIO.LOW)
    time.sleep(.05)

if cx <= 70 and cx > 40 and cy>40 and cy<90:
    GPIO.output(in1,GPIO.LOW)
    GPIO.output(in2,GPIO.LOW)
    GPIO.output(in3,10)
    GPIO.output(in4,GPIO.LOW)
    time.sleep(.05)

if cx <= 40 and cy>40 and cy<90:
```

```
            GPIO.output(in1,GPIO.LOW)
            GPIO.output(in2,4)
            GPIO.output(in3,10)
            GPIO.output(in4,GPIO.LOW)
            time.sleep(.05)

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

    rawCapture.truncate(0)



GPIO.output(in1,GPIO.LOW)
GPIO.output(in2,GPIO.LOW)
GPIO.output(in3,GPIO.LOW)
GPIO.output(in4,GPIO.LOW)

GPIO.cleanup()
```

Shortest Path Using Image Processing:

```
import RPi.GPIO as GPIO
from time import sleep

##################camera
from picamera.array import PiRGBArray
import time
import cv2

import picamera

import numpy as np


#l####################eft motor

in1 = 24

in2 = 23

en = 25

temp1=1

#####################rightmotor

in3 = 17

in4 = 27

enB = 22


#####################value For Path

x=0

count=0

countx=0

county=0

flag=0




GPIO.setmode(GPIO.BCM)

GPIO.setwarnings(False)


#####################left motor

GPIO.setup(in1,GPIO.OUT)

GPIO.setup(in2,GPIO.OUT)

GPIO.setup(en,GPIO.OUT)
```

```
#GPIO.output(in1,GPIO.LOW)
#GPIO.output(in2,GPIO.LOW)
p=GPIO.PWM(en,1000)

#####################rightmotor
GPIO.setup(in3,GPIO.OUT)
GPIO.setup(in4,GPIO.OUT)
GPIO.setup(enB,GPIO.OUT)
#GPIO.output(in3,GPIO.LOW)
#GPIO.output(in4,GPIO.LOW)
p2=GPIO.PWM(enB,1000)

####################motor start
p.start(15)
p2.start(15)
print("\n")
print("working properly by camera.....")
print("\n")



########################camera
camera = picamera.PiCamera()
camera.resolution =(192,108)
camera.framerate = 20
rawCapture = PiRGBArray(camera,size=(192,108))
time.sleep(0.1)

###################Motor Direction
def forward():
    GPIO.output(in1,GPIO.HIGH)
    GPIO.output(in2,GPIO.LOW)
    GPIO.output(in3,GPIO.HIGH)
    GPIO.output(in4,GPIO.LOW)
    time.sleep(.05)

def stop():
    GPIO.output(in1,GPIO.LOW)
    GPIO.output(in2,GPIO.LOW)
    GPIO.output(in3,GPIO.LOW)
    GPIO.output(in4,GPIO.LOW)

def right():
    GPIO.output(in1,10)
    GPIO.output(in2,GPIO.LOW)
    GPIO.output(in3,GPIO.LOW)
    GPIO.output(in4,GPIO.LOW)
    time.sleep(.05)

def left():
    GPIO.output(in1,GPIO.LOW)
    GPIO.output(in2,GPIO.LOW)
    GPIO.output(in3,10)
    GPIO.output(in4,GPIO.LOW)
    time.sleep(.05)

def rightright():
    GPIO.output(in1,10)
    GPIO.output(in2,GPIO.LOW)
    GPIO.output(in3,GPIO.LOW)
    GPIO.output(in4,GPIO.LOW)
    time.sleep(.05)

def leftleft():
    GPIO.output(in1,GPIO.LOW)
    GPIO.output(in2,GPIO.LOW)
    GPIO.output(in3,10)
    GPIO.output(in4,GPIO.LOW)
    time.sleep(.05)

####################Video Processing
def line():
    for frame in camera.capture_continuous(rawCapture, format="bgr",
use_video_port=True):
    count+=1
    print(count)
    image = frame.array
    cv2.imshow('img',image)

    # Create key to break for loop
    key = cv2.waitKey(1) & 0xFF

    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    blur = cv2.GaussianBlur(gray,(5,5),0)
    ret,thresh1 = cv2.threshold(blur,100,255,cv2.THRESH_BINARY_INV)



    # Erode to eliminate noise, Dilate to restore eroded parts of image
    mask = cv2.erode(thresh1, None, iterations=2)
```

```
  mask = cv2.dilate(mask, None, iterations=2)


  # Find all contours in frame
  something, contours, hierarchy =
cv2.findContours(mask.copy(),1,cv2.CHAIN_APPROX_NONE)


  if len(contours) > 0:
      # Find largest contour area and image moments
    c = max(contours, key = cv2.contourArea)
    M = cv2.moments(c)


    # Find x-axis centroid using image moments
    cx = int(M['m10']/M['m00'])
    cy=int(M['m01']/M['m00'])


    if cx >= 150 and cy>40 and cy<90:
      rightright()


    elif cx < 150 and cx >= 130 and cy>40 and cy<90:
      right()


    elif cx < 130 and cx > 70 and cy>40 and cy<90:
      forward()


    elif cx <= 70 and cx > 40 and cy>40 and cy<90:
      left()


    elif cx <= 40 and cy>40 and cy<90:
      leftleft()
```

```
        elif cx<50 and cx>140 and cy>90 and cy<108:
            break
        elif cx>0 and cx<192 and cy>0 and cy<108:
            flag=1
            break
    #####################Use q for Stop Camera
    if key == ord("q"):
            break

    rawCapture.truncate(0)
###################turn Point
def stoppoint():
    forward()
    rightright()
    time.sleep(.8)
    stop()
    flag=0

line()
##########decision Point
if x==0:
    leftleft()
    forward()
    time.sleep(.5)
    line()
    if flag==1:
        stoppoint()
    countx=count
    x=1
```

```
if x==1:
    rightright()
    forward()
    time.sleep(.5)
    line()
    if flag==1:
        stoppoint()
    county=count-countx
    x=2

if x==2 and countx<county:
    leftleft()
    forward()
    time.sleep(.5)
    line()
    forward()
    time.sleep(.2)
    stop()
elif x==2 and countx>county:
    rightright()
    forward()
    time.sleep(.5)
    line()
    forward()
    time.sleep(.2)
    stop()

    stop()
GPIO.cleanup()
```