EL-203 Embedded Hardware Design
Project Title :- Defective object detection

Group Members:-

201401136 - Dhruv
201401137 - Mohit
201401138 - Dhaval
201401139 - Hardik
201401140 - Parth
201401141 - Nilay

Introduction:-

We have proposed that we will detect an object and get the results whether the given object is defective or not.

Problem Statement:-

An object detection and checking with sample object of same type.

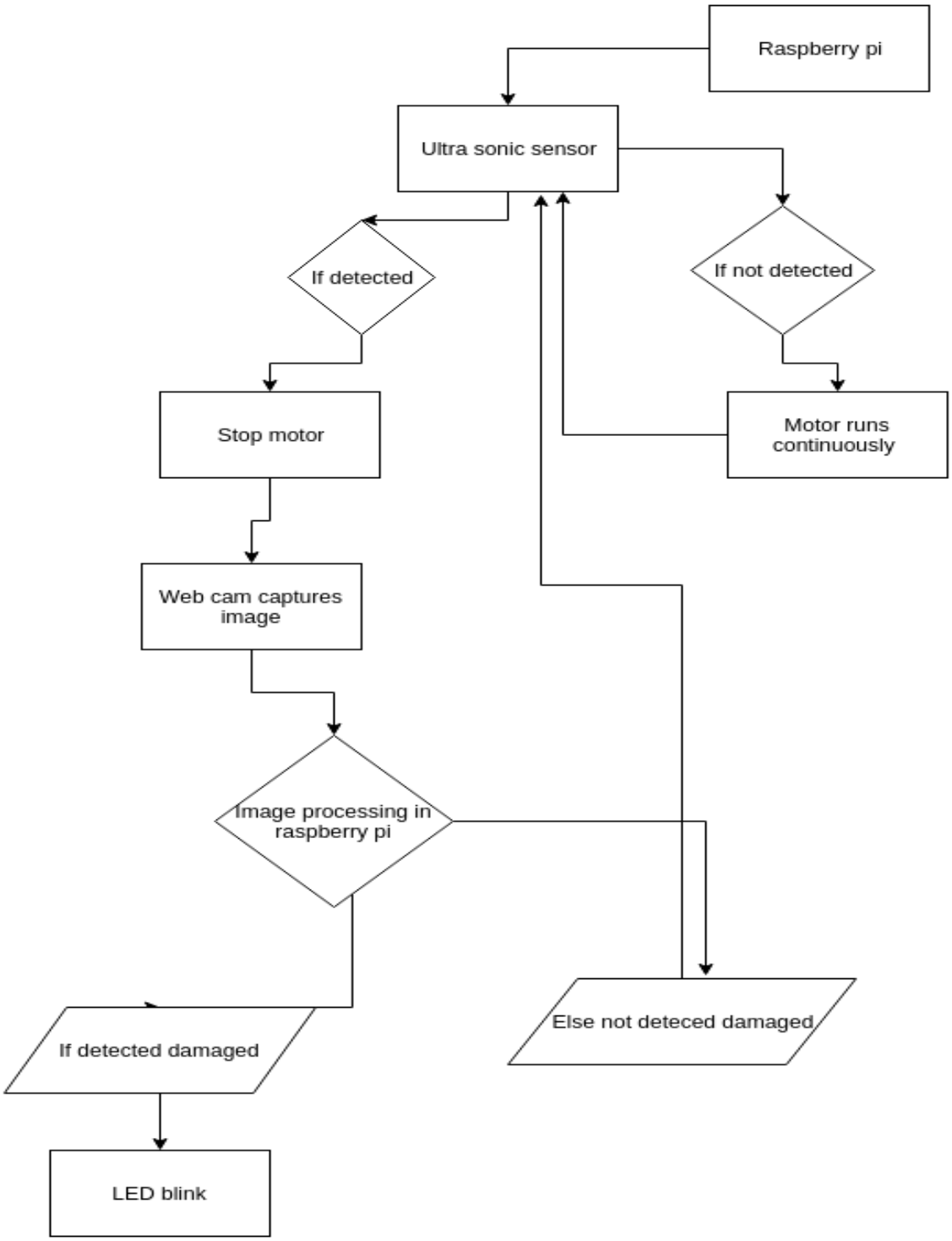
Proposal:-

Here a small general purpose battery will move through the conveyor belt, after detecting it an image will be captured by webcam. Image detection will be performed on raspberry pi and we will get the result of detection.

Instruments:-

Raspberry Pi
Conveyor belt
Web camera
Ultrasonic sensor
Bread-board
Relay circuit

Block Diagram:-



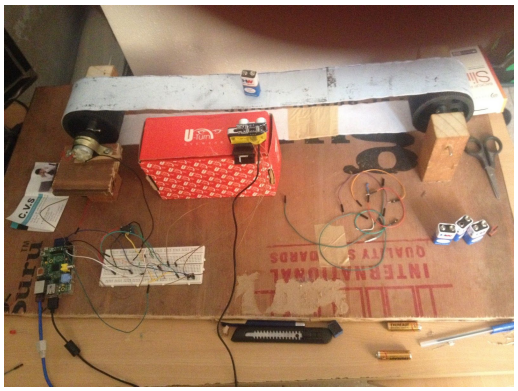
Description :-

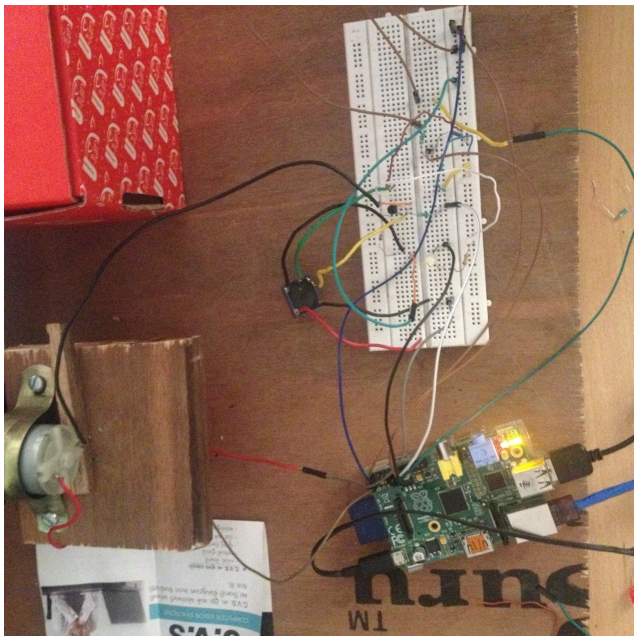
Our model contains a conveyor belt attached with 100 rpm gear motor which is connected to the Raspberry pi through relay circuit. There is an ultrasonic sensor set at the front side of the belt, which will detect the moving object and send an output signal to the raspberry pi. Raspberry pi will take it as an input signal and stop the gear motor. When the motor is stopped an image will be captured by usb webcam ,which is being controlled by Raspberry pi.Raspberry pi will use manhattan norm algorithm for finding absolute difference between captured image and sample image.After that Pi will generate the result that whether the given object has any damage or not, if it has been damaged LED will blink.

Project Scope :-

This is the main logic that is being used in industry to identify defected objects from the bulk of their products. It is useful to maintain the product accuracy in the world of stealing designs and products. Using this mechanism company can definitely identify the wrong objects and duplicates of same product.

Circut :-





Code :-

---> im.py

```
import sys
from scipy.misc import imread
from scipy.linalg import norm
from scipy import sum, average

def main():
    file1, file2 = sys.argv[1:1+2]
    img1 = to_grayscale(imread(file1).astype(float)) #grayscale image to reduce noise
    img2 = to_grayscale(imread(file2).astype(float))
    n_m, n_0 = compare_images(img1, img2)
    print n_m/img1.size

def compare_images(img1, img2):
    img1 = normalize(img1) #normalize image
    img2 = normalize(img2)
    diff = img1 - img2  #difference of two images
    m_norm = sum(abs(diff))  #main logic of Manhattan Norm
    z_norm = norm(diff.ravel(), 0)
    return (m_norm, z_norm)

def to_grayscale(arr):
    "If arr is a color image (3D array), convert it to grayscale (2D array)."
    if len(arr.shape) == 3:
        return average(arr, -1)
    else:
        return arr

def normalize(arr):
    rng = arr.max()-arr.min()
    amin = arr.min()
    return (arr-amin)*255/rng

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

---> script.py

```
import time
import RPi.GPIO as GPIO
import subprocess
import os
```

```
TRIG = 21                              # Pin for ultrasonic sensor
ECHO = 22                              # Pin for ultrasonic sensor
LED = 18
GPIO.setmode(GPIO.BCM)
GPIO.setup(LED,GPIO.OUT)                       #setup
GPIO.setup(23,GPIO.OUT)                #setup
GPIO.setup(TRIG,GPIO.OUT)                      #setup
GPIO.setup(ECHO,GPIO.IN)               #setup
def main():                            # function
      start_motor()                    # motor will start
      img="click.jpg"                  # captured image through webcam
      temp_img="template.jpg"                  # sample template image

      while True:
            x=read_from_sensor()               #continuously sence by sensor
            if x!=-1:                          #if detected
                  time.sleep(0.4)
                  stop_motor()           #then motor stop
                  os.system('fswebcam '+str(img))            # command for capturing image
                        os.system('python im.py '+str(img)+' '+str(temp_img)+'>diff.txt')    #
compare 2 images
                  output=open('diff.txt').read().strip()
                  print output
                  x=float(output)
                  print x
                  print "output is" +str(x)
                  if x>45:                                   # threshold
                        blink_led()
                  start_motor()
                  time.sleep(1.5)


def blink_led():                                   #code for LED blink
      GPIO.output(LED,True)
      time.sleep(3)
      GPIO.output(LED,False)
      time.sleep(3)

def start_motor():                     # code for motor starting
      GPIO.output(23,True)

def read_from_sensor():                # read sensor input data
      print "Waiting For Sensor To Settle"
```

```
        pulse_start = 0
        GPIO.output(TRIG,True)
        time.sleep(0.00001)
        GPIO.output(TRIG,False)
        while GPIO.input(ECHO)==0:
                pulse_start = time.time()

        while GPIO.input(ECHO)==1:
                pulse_end = time.time()
        pulse_duration = pulse_end - pulse_start

        distance = pulse_duration * 17150

        distance = round(distance,2)
        if distance < 13 :
                return 1
        else:
                return -1

def stop_motor():
        GPIO.output(23,False)

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