INTERNET OF THINGS(IOT)

Academic year 2019-20

PROJECT REPORT

SMART MANAGEMENT SYSTEM FOR PHARMACEUTICAL REFRIGERATORS USING IOT

Submitted by:

ABHISHEK R(4NI16CS002)

CHETHAN K(4NI16EE008)

HARSHINI P(4NI16EC035)

J GUNASEKHAR(4NI16CS041)

MEGHANA G S(4NI16IS048)

SIDDHANTH ARORA(4NI16IS099)

Under the guidance

Mrs POORNIMA N **Mr. RAJIV N** **Mr ROHIT**

ABSTRACT

The objective of this project is to build a system which can monitor the proper storage and stock availability of medicines.

Nowadays most of the pharmaceutical stores have refrigerators to store some vaccines. When the stock inside the refrigerator gets exhausted the shopkeeper might not be aware of it which may lead to delay in order of the stock. Even there are chances where the shopkeeper may provide the wrong vaccine to the customer. Also, most of the elder people tend to forget to buy the vaccine before it gets over. There is no existing solution which overcomes the above-mentioned problems.

ACKNOWLEDGEMENT

We would like to thank **Mr. Rajiv N** and **Mr. Rohit** for accepting the idea of our project and provided with details for the scalability. They motivated us by providing timely guidelines and helped us in solving our queries.

Their help and guidance made us complete our project on time with all the functionalities promised. We learnt many new technologies and applied that in our project, which made our project efficient and ready to use.

We would also like to thank **Mrs. Poornima N** for helping out throughout the process of project making period for her guidance.

It was a cooperative work with all the group members contributing to the project and reaching to help out each other to bring the final product.

- Abhishek R
- Chethan K
- Harshini P
- J Gunasekhar
- Meghana G S
- Siddhanth arora

TABLE OF CONTENTS

LIST OF FIGURES

PROJECT IDEA

The proposed solution consists of an IOT system which is capable of:

- Monitoring the stock inside the refrigerator and sends an alert to the shopkeeper when any vaccine reaches below a number.
- Triggers an alarm when the vaccine is not placed in the appropriate compartment.
- Notifies the customer about reordering the vaccine based on their regular usage.

We are going to use a microcontroller board which continuously monitors the stock inside the refrigerator by integrating with several sensors and sends an alert to the shopkeeper using a Wi-Fi module. Using image processing technique an alarm will trigger if any vaccine is misplaced. Customer's purchase and usage will be stored in order to compute, when the stock needs to be reordered and notifies the same to the customer.

EXISTING MODEL

- Stock in the refrigerator get exhausted and shopkeeper might not be aware of it which may lead to delay in order of the stock.
- Shopkeeper may provide the wrong vaccine to the customer due to misplacement.
- Most of the elder people tend to forget to buy the vaccine before it gets over.

PROPOSED MODEL

Solution Proposal

The proposal takes care of various different terms inside the refrigerators. It maintains certain temperature desired for the refrigerator, also it stock management and reorders the vaccines after the stock falls below a certain threshold value.

- The proposed design aims to implement a Smart Pharmaceutical System, which is easy to use and economical for the user.

- It is capable of notifying its owner about the activities going on inside it via wireless system on the mobile phone.

- The whole system is governed by the STM32F103x8 cortex M3 ARM microcontroller and Wi-Fi transmits all the information to the android phone by using IoT.

- The items quantity is below the set threshold value to alert notification is sent to the user's mobile to reorder the stock before the get over.

- The proposed solution consists of an IOT system which is capable of monitoring the stock inside the refrigerator and sends an alert to the shopkeeper when any vaccine reaches below a number.

- It also triggers an alarm when the vaccine is not placed in the appropriate compartment. The temperature is taken care of by the temperature sensor and is always maintained.

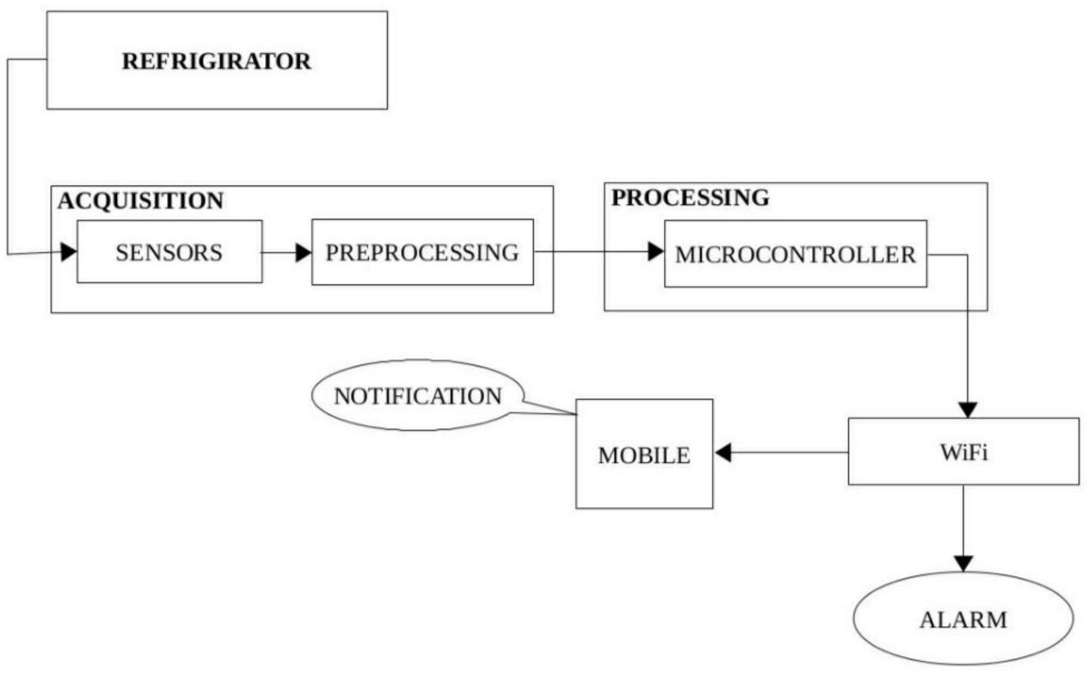

Figure 1

<u>Components</u>

- UV SENSOR for stock management.
- CAMERA SENSOR for stock misplace notification.
- TEMPERATURE SENSOR for temperature management and control.
- WiFi MODULE for automatic order placing and stock notification to owner.
- ALARM for signal of misplaced stock.

Architecture

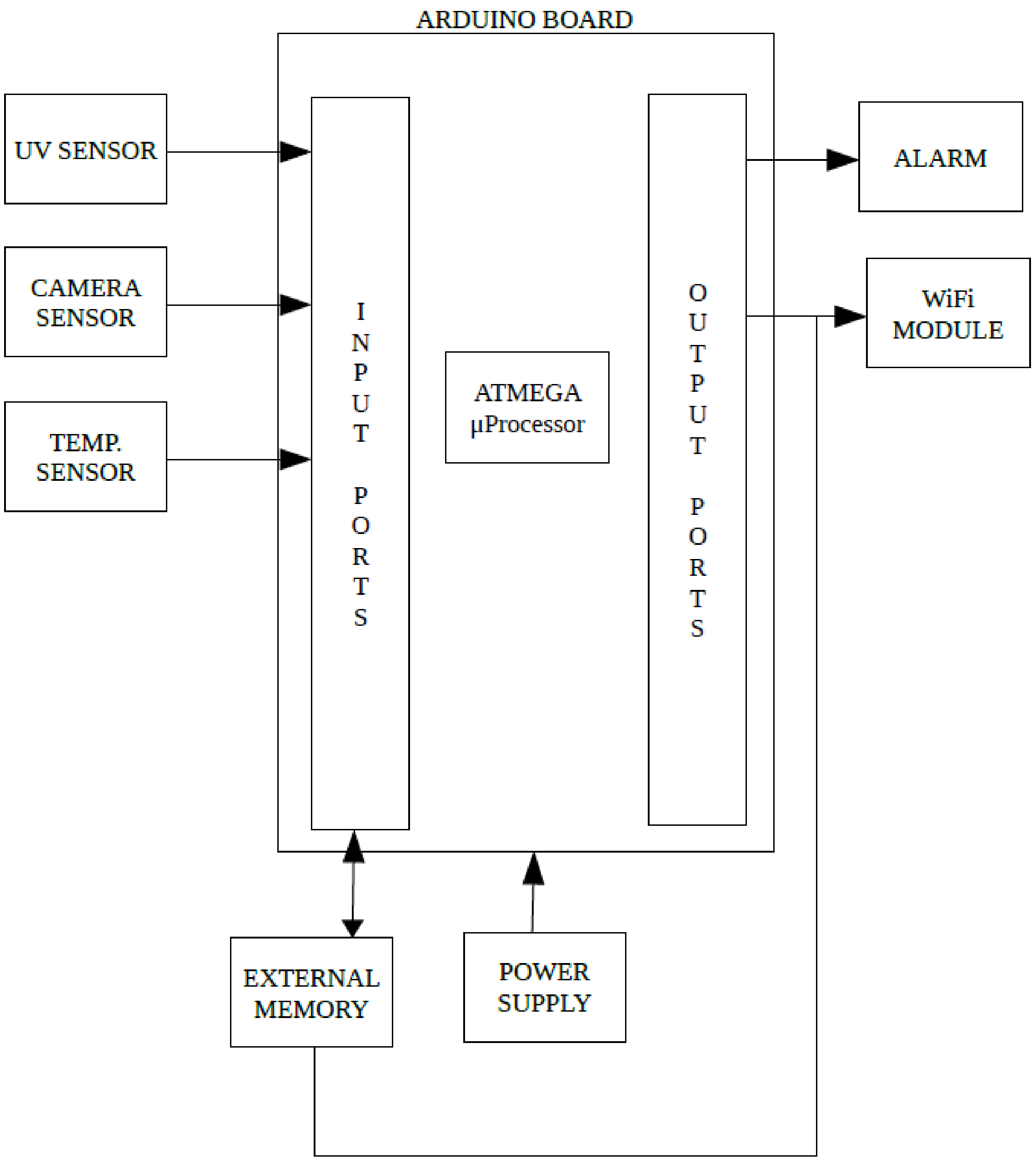

Figure 2

Implementation Flow Chart

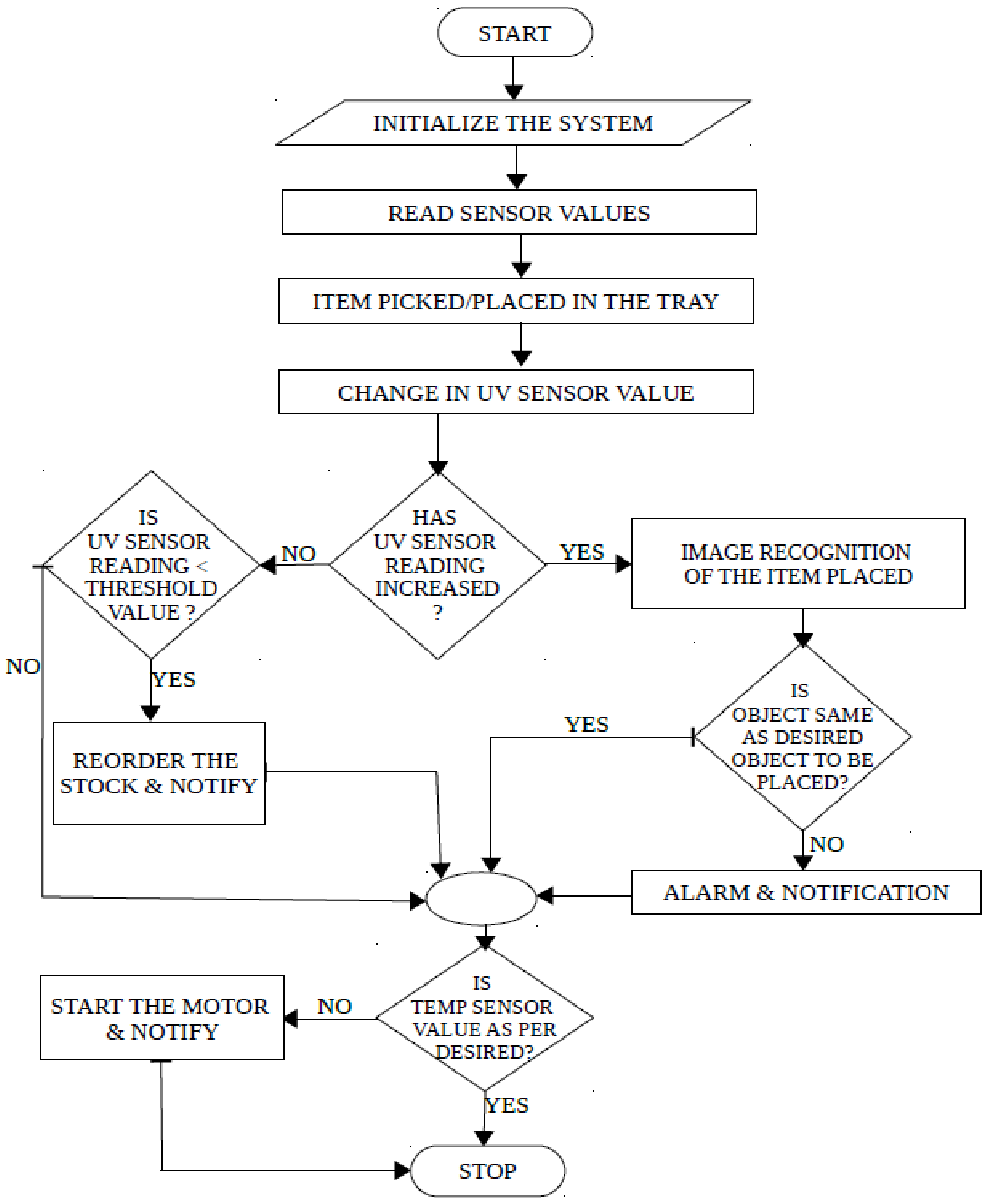
START
INITIALIZE THE SYSTEM
READ SENSOR VALUES
ITEM PICKED/PLACED IN THE TRAY
CHANGE IN UV SENSOR VALUE
HAS UV SENSOR READING INCREASED ?
YES
IMAGE RECOGNITION OF THE ITEM PLACED
IS OBJECT SAME AS DESIRED OBJECT TO BE PLACED?
YES
NO
ALARM & NOTIFICATION
NO
IS UV SENSOR READING < THRESHOLD VALUE ?
YES
REORDER THE STOCK & NOTIFY
NO
IS TEMP SENSOR VALUE AS PER DESIRED?
NO
START THE MOTOR & NOTIFY
YES
STOP

Figure 3

Use cases

- Monitoring the stock inside the refrigerator and sends an alert to the shopkeeper when any vaccines reaches below a certain threshold value.
- Triggers an alarm when the vaccine is not placed in the appropriate compartment.
- Notifies the owner and reorders the vaccines automatically without the intervention required based on the usage and left stock.
- Keep a note of the temperature maintained inside the compartment and if the temperature fluctuates it tries to maintain it.

Screenshots

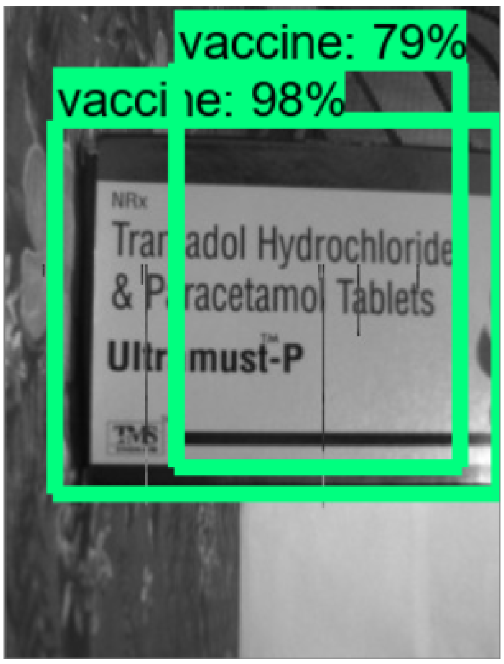

Figure 4

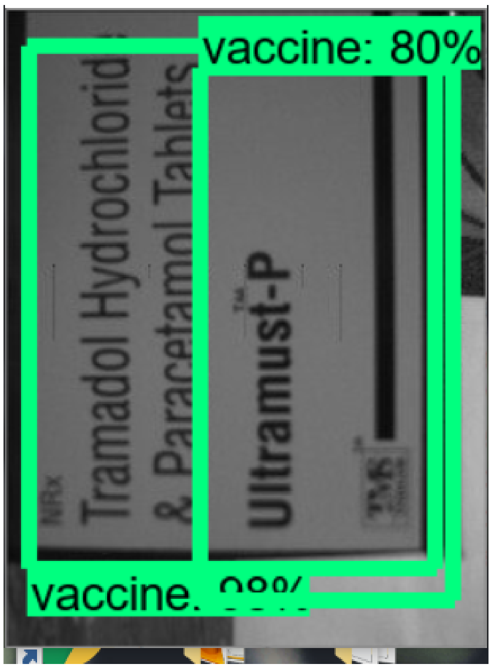

Figure 5

Training images

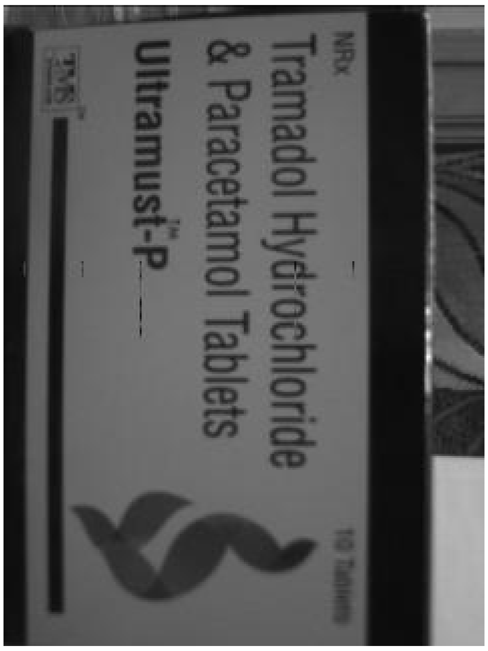

Figure 6

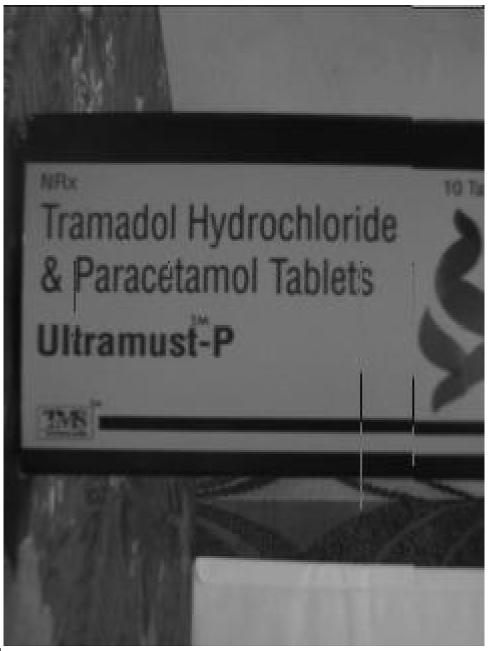

Figure 7

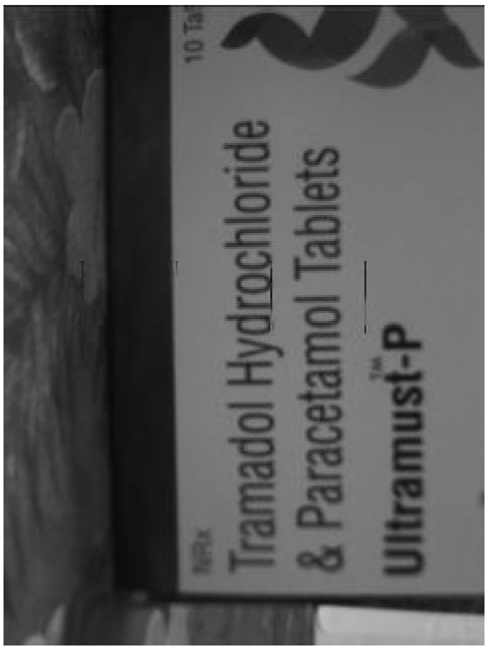

Figure 8

Source Code

Image object detection

```
import os
import cv2
import numpy as np
import tensorflow as tf
import sys

# This is needed since the notebook is stored in the object_detection folder.
sys.path.append("..")

# Import utilities
from utils import label_map_util
from utils import visualization_utils as vis_util

# Name of the directory containing the object detection module we're using
MODEL_NAME = 'inference_graph'
IMAGE_NAME = 'test3.jpg'

# Grab path to current working directory
CWD_PATH = os.getcwd()

# Path to frozen detection graph .pb file, which contains the model that is used
# for object detection.
PATH_TO_CKPT = os.path.join(CWD_PATH,MODEL_NAME,'frozen_inference_graph.pb')

# Path to label map file
PATH_TO_LABELS = os.path.join(CWD_PATH,'training','labelmap.pbtxt')

# Path to image
PATH_TO_IMAGE = os.path.join(CWD_PATH,IMAGE_NAME)

# Number of classes the object detector can identify
NUM_CLASSES = 1

# Load the label map.
# Label maps map indices to category names, so that when our convolution
# network predicts `5`, we know that this corresponds to `king`.
# Here we use internal utility functions, but anything that returns a
# dictionary mapping integers to appropriate string labels would be fine
label_map = label_map_util.load_labelmap(PATH_TO_LABELS)
categories = label_map_util.convert_label_map_to_categories(label_map,
max_num_classes=NUM_CLASSES, use_display_name=True)
category_index = label_map_util.create_category_index(categories)

# Load the Tensorflow model into memory.
detection_graph = tf.Graph()
with detection_graph.as_default():
        od_graph_def = tf.GraphDef()
        with tf.gfile.GFile(PATH_TO_CKPT, 'rb') as fid:
```

```
        serialized_graph = fid.read()
        od_graph_def.ParseFromString(serialized_graph)
        tf.import_graph_def(od_graph_def, name='')

        sess = tf.Session(graph=detection_graph)

# Define input and output tensors (i.e. data) for the object detection classifier
# Input tensor is the image
image_tensor = detection_graph.get_tensor_by_name('image_tensor:0')

# Output tensors are the detection boxes, scores, and classes
# Each box represents a part of the image where a particular object was detected
detection_boxes = detection_graph.get_tensor_by_name('detection_boxes:0')

# Each score represents level of confidence for each of the objects.
# The score is shown on the result image, together with the class label.
detection_scores = detection_graph.get_tensor_by_name('detection_scores:0')
detection_classes = detection_graph.get_tensor_by_name('detection_classes:0')

# Number of objects detected
num_detections = detection_graph.get_tensor_by_name('num_detections:0')

# Load image using OpenCV and
# expand image dimensions to have shape: [1, None, None, 3]
# i.e. a single-column array, where each item in the column has the pixel RGB value
image = cv2.imread(PATH_TO_IMAGE)
image_expanded = np.expand_dims(image, axis=0)

# Perform the actual detection by running the model with the image as input
(boxes, scores, classes, num) = sess.run(
        [detection_boxes, detection_scores, detection_classes, num_detections],
        feed_dict={image_tensor: image_expanded})

# Draw the results of the detection (aka 'visualize the results')

vis_util.visualize_boxes_and_labels_on_image_array(
        image,
        np.squeeze(boxes),
        np.squeeze(classes).astype(np.int32),
        np.squeeze(scores),
        category_index,
        use_normalized_coordinates=True,
        line_thickness=8,
        min_score_thresh=0.60)
# All the results have been drawn on image. Now display the image.
cv2.imshow('Object detector', image)

# Press any key to close the image
cv2.waitKey(0)
# Clean up
cv2.destroyAllWindows()
```

Using ultrasonic distance sensor HC SR04 Buzzer LED

```
#include "dht.h"
#define dht_apin A0 // Analog Pin sensor is connected to

dht DHT;

// defines pins numbers
const int trigPin = 9;
const int echoPin = 10;
const int buzzer = 11;
const int ledPin = 13;
const int fan = 12;

// defines variables
long duration;
int distance;
int safetyDistance;

void setup() {
pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
pinMode(echoPin, INPUT); // Sets the echoPin as an Input
pinMode(buzzer, OUTPUT);
pinMode(ledPin, OUTPUT);
pinMode(fan,OUTPUT);
Serial.begin(9600); // Starts the serial communication
  delay(200);//Delay to let system boot
  Serial.println("DHT11 Humidity & temperature Sensor\n\n");
  delay(200);//Wait before accessing Sensor

}

void loop() {
// Clears the trigPin
digitalWrite(trigPin, LOW);
delayMicroseconds(2);

// Sets the trigPin on HIGH state for 10 micro seconds
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);

// Reads the echoPin, returns the sound wave travel time in microseconds
duration = pulseIn(echoPin, HIGH);




// Calculating the distance
distance= duration*0.034/2;
safetyDistance = distance;
if (safetyDistance > 15){
  digitalWrite(buzzer, HIGH);
  digitalWrite(ledPin, HIGH);
  digitalWrite(fan, HIGH);
}
else{
  digitalWrite(buzzer, LOW);
  digitalWrite(ledPin, LOW);
  digitalWrite(fan, LOW);
}

// Prints the distance on the Serial Monitor
Serial.print("Distance: ");
Serial.println(distance);

            DHT.read11(dht_apin);

            Serial.print("Current humidity = ");
            Serial.print(DHT.humidity);
            Serial.print("%  ");
            Serial.print("temperature = ");
            Serial.print(DHT.temperature);
            Serial.println("C  ");
   if (DHT.temperature >= 20.00)
   {
            digitalWrite(fan, HIGH);

   }
   else
   {
            digitalWrite(fan, LOW);
   }
            delay(500);//Wait 5 seconds before accessing sensor again.

}
```

Camera OV7670

```
struct regval_list{
  uint8_t reg_num;
  uint16_t value;
};

const struct regval_list qvga_ov7670[] PROGMEM = {
  { REG_COM14, 0x19 },
  { 0x72, 0x11 },
  { 0x73, 0xf1 },
  { REG_HSTART, 0x16 },
  { REG_HSTOP, 0x04 },
  { REG_HREF, 0xa4 },
  { REG_VSTART, 0x02 },
  { REG_VSTOP, 0x7a },
  { REG_VREF, 0x0a },

  { 0xff, 0xff }, /* END MARKER */
};

const struct regval_list yuv422_ov7670[] PROGMEM = {
  { REG_COM7, 0x0 },  /* Selects YUV mode */
  { REG_RGB444, 0 },  /* No RGB444 please */
  { REG_COM1, 0 },
  { REG_COM15, COM15_R00FF },
  { REG_COM9, 0x6A }, /* 128x gain ceiling; 0x8 is reserved bit */
  { 0x4f, 0x80 },   /* "matrix coefficient 1" */
  { 0x50, 0x80 },   /* "matrix coefficient 2" */
  { 0x51, 0 },      /* vb */
  { 0x52, 0x22 },   /* "matrix coefficient 4" */
  { 0x53, 0x5e },   /* "matrix coefficient 5" */
  { 0x54, 0x80 },   /* "matrix coefficient 6" */
  { REG_COM13, COM13_UVSAT },
  { 0xff, 0xff },   /* END MARKER */
};

const struct regval_list ov7670_default_regs[] PROGMEM = {//from the linux driver
  { REG_COM7, COM7_RESET },
  { REG_TSLB, 0x04 }, /* OV */
  { REG_COM7, 0 },  /* VGA */
  /*
  * Set the hardware window.  These values from OV don't entirely
  * make sense - hstop is less than hstart.  But they work...
  */
  { REG_HSTART, 0x13 }, { REG_HSTOP, 0x01 },
  { REG_HREF, 0xb6 }, { REG_VSTART, 0x02 },
  { REG_VSTOP, 0x7a }, { REG_VREF, 0x0a },

  { REG_COM3, 0 }, { REG_COM14, 0 },

 /* Mystery scaling numbers */
  { 0x70, 0x3a }, { 0x71, 0x35 },
  { 0x72, 0x11 }, { 0x73, 0xf0 },
  { 0xa2,/* 0x02 changed to 1*/1 }, { REG_COM10, 0x0 },
  /* Gamma curve values */

  /* AGC and AEC parameters.  Note we start by disabling those features,
  then turn them only after tweaking the values. */

  /* Almost all of these are magic "reserved" values.  */
  /* More reserved magic, some of which tweaks white balance */

  /* Matrix coefficients */

  /* Extra-weird stuff.  Some sort of multiplexor register */


void error_led(void){
  DDRB |= 32;//make sure led is output
  while (1){//wait for reset
	PORTB ^= 32;// toggle led
	_delay_ms(100);
  }
}

void twiStart(void){
  TWCR = _BV(TWINT) | _BV(TWSTA) | _BV(TWEN);//send start
  while (!(TWCR & (1 << TWINT)));//wait for start to be transmitted
  if ((TWSR & 0xF8) != TW_START)
	error_led();
}

void twiWriteByte(uint8_t DATA, uint8_t type){
  TWDR = DATA;
  TWCR = _BV(TWINT) | _BV(TWEN);
  while (!(TWCR & (1 << TWINT))) {}
  if ((TWSR & 0xF8) != type)
	error_led();
}

void twiAddr(uint8_t addr, uint8_t typeTWI){
  TWDR = addr;//send address
  TWCR = _BV(TWINT) | _BV(TWEN);    /* clear interrupt to start transmission */
  while ((TWCR & _BV(TWINT)) == 0); /* wait for transmission */
  if ((TWSR & 0xF8) != typeTWI)
	error_led();
}

void writeReg(uint8_t reg, uint8_t dat){
  //send start condition
  twiStart();
  twiAddr(camAddr_WR, TW_MT_SLA_ACK);
  twiWriteByte(reg, TW_MT_DATA_ACK);
  twiWriteByte(dat, TW_MT_DATA_ACK);
  TWCR = (1 << TWINT) | (1 << TWEN) | (1 << TWSTO);//send stop
  _delay_ms(1);
}

static uint8_t twiRd(uint8_t nack){
  if (nack){
	TWCR = _BV(TWINT) | _BV(TWEN);
```

```
            while ((TWCR & _BV(TWINT)) == 0); /* wait for transmission */
            if ((TWSR & 0xF8) != TW_MR_DATA_NACK)
            error_led();
            return TWDR;
  }
  else{
            TWCR = _BV(TWINT) | _BV(TWEN) | _BV(TWEA);
            while ((TWCR & _BV(TWINT)) == 0); /* wait for transmission */
            if ((TWSR & 0xF8) != TW_MR_DATA_ACK)
            error_led();
            return TWDR;
  }
}

uint8_t rdReg(uint8_t reg){
  uint8_t dat;
  twiStart();
  twiAddr(camAddr_WR, TW_MT_SLA_ACK);
  twiWriteByte(reg, TW_MT_DATA_ACK);
  TWCR = (1 << TWINT) | (1 << TWEN) | (1 << TWSTO);//send stop
  _delay_ms(1);
  twiStart();
  twiAddr(camAddr_RD, TW_MR_SLA_ACK);
  dat = twiRd(1);
  TWCR = (1 << TWINT) | (1 << TWEN) | (1 << TWSTO);//send stop
  _delay_ms(1);
  return dat;
}

void wrSensorRegs8_8(const struct regval_list reglist[]){
  uint8_t reg_addr, reg_val;
  const struct regval_list *next = reglist;
  while ((reg_addr != 0xff) | (reg_val != 0xff)){
            reg_addr = pgm_read_byte(&next->reg_num);
            reg_val = pgm_read_byte(&next->value);
   writeReg(reg_addr, reg_val);
            next++;
  }
}

void setColor(void){
  wrSensorRegs8_8(yuv422_ov7670);
 // wrSensorRegs8_8(qvga_ov7670);
}

void setResolution(void){
 writeReg(REG_COM3, 4); // REG_COM3 enable scaling
  wrSensorRegs8_8(qvga_ov7670);
}

void camInit(void){
 writeReg(0x12, 0x80);
  _delay_ms(100);
  wrSensorRegs8_8(ov7670_default_regs);
 writeReg(REG_COM10, 32);//PCLK does not toggle on HBLANK.
}

void arduinoUnoInut(void) {
  cli();//disable interrupts

        /* Setup the 8mhz PWM clock
  * This will be on pin 11*/
  DDRB |= (1 << 3);//pin 11
  ASSR &= ~(_BV(EXCLK) | _BV(AS2));
  TCCR2A = (1 << COM2A0) | (1 << WGM21) | (1 << WGM20);
  TCCR2B = (1 << WGM22) | (1 << CS20);
  OCR2A = 0;//(F_CPU)/(2*(X+1))
  DDRC &= ~15;//low d0-d3 camera
  DDRD &= ~252;//d7-d4 and interrupt pins
  _delay_ms(3000);

        //set up twi for 100khz
  TWSR &= ~3;//disable prescaler for TWI
  TWBR = 72;//set to 100khz

        //enable serial
  UBRR0H = 0;
  UBRR0L = 1;//0 = 2M baud rate. 1 = 1M baud. 3 = 0.5M. 7 = 250k 207 is 9600 baud rate.
  UCSR0A |= 2;//double speed aysnc
  UCSR0B = (1 << RXEN0) | (1 << TXEN0);//Enable receiver and transmitter
  UCSR0C = 6;//async 1 stop bit 8bit char no parity bits
}

void StringPgm(const char * str){
  do{
        while (!(UCSR0A & (1 << UDRE0)));//wait for byte to transmit
        UDR0 = pgm_read_byte_near(str);
        while (!(UCSR0A & (1 << UDRE0)));//wait for byte to transmit
  } while (pgm_read_byte_near(++str));
}

static void captureImg(uint16_t wg, uint16_t hg){
  uint16_t y, x;

  StringPgm(PSTR("*RDY*"));

  while (!(PIND & 8));//wait for high
  while ((PIND & 8));//wait for low

        y = hg;
  while (y--){
        x = wg;
```

```
            //while (!(PIND & 256));//wait for high
            while (x--){
            while ((PIND & 4));//wait for low
            UDR0 = (PINC & 15) | (PIND & 240);
            while (!(UCSR0A & (1 << UDRE0)));//wait for byte to transmit
            while (!(PIND & 4));//wait for high
            while ((PIND & 4));//wait for low
            while (!(PIND & 4));//wait for high
            }
            //  while ((PIND & 256));//wait for low
  }
            _delay_ms(100);
}

void setup(){
  arduinoUnoInut();
  camInit();
  setResolution();
  setColor();
 writeReg(0x11, 10); //Earlier it had the value:writeReg(0x11, 12); New version works better for me :)
!!!!
}

void loop(){
  captureImg(320, 240);
}
```

Future Enhancements

- Training the model to more accurate level.
- Improvising on customer interaction and customer notification.
- Getting the full working model.
- Update the customer database where each customer data will be stored for future use.

REFERENCES

https://www.wikipedia.org

https://www.instructables.com/

https://www.draw.io/

https://www.researchgate.net/