CODING OF A SYSTEM

I. MICROCONTROLLER PROGRAMMING (ARDUINO IDE)

```
#include <ESP8266WiFi.h>
char ssid[] = "Project";
char pass[] = "12345678";
WiFiClient  client;
#include "ThingSpeak.h"
//#include <WiFi.h>
//#include "WiFi.h"
#define SECRET_CH_ID 1736899
#define SECRET_WRITE_APIKEY "OEMFMV390IANB4YF"
#include <LCD_I2C.h>
float x=0.0;
int y=0;
LCD_I2C lcd(0x27);
#include <Wire.h>
#include "MAX30100_PulseOximeter.h"
float vref = 3.3;
float resolution = vref/1023;
#define REPORTING_PERIOD_MS     1000
PulseOximeter pox;
int keyIndex = 0;
String myStatus = "";
unsigned long myChannelNumber = SECRET_CH_ID;
const char * myWriteAPIKey = SECRET_WRITE_APIKEY;
uint32_t tsLastReport = 0;

void onBeatDetected()
{
    Serial.println("Beat!");
}

void setup()
{
    Serial.begin(115200);
    lcd.begin();
    lcd.begin(false)

    lcd.backlight();
    lcd.setCursor(0, 0);
        lcd.print("PATIENT");
    lcd.setCursor(0, 1);
    lcd.print("MONITORING");
    delay(5000);
    lcd.clear();
     if(WiFi.status() != WL_CONNECTED){

    while(WiFi.status() != WL_CONNECTED){
      WiFi.begin(ssid, pass);  // Connect to WPA/WPA2 network. Change this
line if using open or WEP network
    //  Serial.print(".");
      delay(5000);
     break;
     }
   // Serial.println("\nConnected.");
  }
       WiFi.mode(WIFI_STA);
  ThingSpeak.begin(client);
    Serial.print("Initializing pulse oximeter..");
    if (!pox.begin()) {
        Serial.println("FAILED");
        for(;;);
    } else {
        Serial.println("SUCCESS");
    }
    pox.setOnBeatDetectedCallback(onBeatDetected);

}

void loop()
{
    pox.update();
    if (millis() - tsLastReport > REPORTING_PERIOD_MS) {
      x=pox.getHeartRate();
      y=pox.getSpO2();
        Serial.print("Heart rate:");
        Serial.print(x);
        Serial.print("bpm / SpO2:");
        Serial.print(y);
        Serial.println("%");

if((x>70)&&(y>90))
{

 lcd.setCursor(0, 0);
        lcd.print("Heart:");
    lcd.print(x);
      lcd.setCursor(0, 1);
        lcd.print("SP02:");
    lcd.print(y);
    delay(2000);
    lcd.clear();
    float temperature = analogRead(A0);
 temperature = (temperature*resolution);
 temperature = temperature*100;
 Serial.println(temperature);
 lcd.setCursor(0, 0);
        lcd.print("TEM:");
    lcd.print(temperature);
    delay(2000);
    lcd.clear();
    ThingSpeak.setField(1, x);
  ThingSpeak.setField(2,y);
  ThingSpeak.setField(3,temperature);
  int x = ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);
  delay(60000);
}
tsLastReport = millis();

    }



}
```

II. DATASET TRAINING AND TESTING CODE:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import seaborn as sns
import pickle
data = pd.read_csv('/content/data.csv')
data.head()
data.shape
X = data.iloc[:,:-1]
X.head()
y = data.iloc[:,-1]
y.head()
data['target'].value_counts()
X_train,X_test,y_train,y_test =
train_test_split(X,y,test_size=0.2,random_state=1)
sns.countplot(x='target',data=data)
plt.show()
X_train.shape
X_train.head()
y_test.shape
y_test.head()

from sklearn.neighbors import KNeighborsClassifier

model = KNeighborsClassifier(n_neighbors=3)
model.fit(X_train,y_train)
filename = 'Patient_model.sav'
```

```
pickle.dump(model, open(filename, 'wb'))
y_pred = model.predict(X_test)
from sklearn import metrics
acc=(metrics.accuracy_score(y_pred,y_test))
print("Accuracy is:",acc)
print("Confusion Matrix is: ",metrics.confusion_matrix(y_pred,y_test))
```

*III.*DATA CLASSFICATION AND PREDICTION

```
import pickle
import urllib.request
import json
from time import sleep
conn = urllib.request.urlopen("https://api.thingspeak.com/channels/1736899/feeds.json?results=1")
response = conn.read()
print ("http status code=%s" % (conn.getcode()))
data=json.loads(response)
x=int(data['feeds'][0]['entry_id'])
y=x
conn.close()
while x==y:
  conn = urllib.request.urlopen("https://api.thingspeak.com/channels/1736899/feeds.json?results=1")
  response = conn.read()
  #print ("http status code=%s" % (conn.getcode()))
  data=json.loads(response)
  y=int(data['feeds'][0]['entry_id'])
  conn.close()
```

```
conn =
urllib.request.urlopen("https://api.thingspeak.com/channels/1736899/feeds.json?
results=1")
response = conn.read()
print ("http status code=%s" % (conn.getcode()))
data=json.loads(response)
a=float(data['feeds'][0]['field1'])
b=float(data['feeds'][0]['field2'])
c=float(data['feeds'][0]['field3'])

conn.close()
filename = 'Patient_model.sav'
loaded_model = pickle.load(open(filename, 'rb'))
person_reports = [[a,b,c]]
disease_predicted = loaded_model.predict(person_reports)
print("ANALYSING....")
sleep(15)
disease_predicted[0]=0
if disease_predicted[0]==1:
   print("The person may have no disease")
   #sleep(30)
   conn =
urllib.request.urlopen("https://api.thingspeak.com/update?api_key=OEMFMV3
90IANB4YF&field7=NO_DISEASES")
elif disease_predicted[0]==0:
   print("The person may be in Fever take Paracetamol")
   #sleep(30)
   conn =
urllib.request.urlopen("https://api.thingspeak.com/update?api_key=OEMFMV3
90IANB4YF&field7=PARACETAMOL")
elif disease_predicted[0]==2:
    print("The person may be in Hypertension take nisoldipine")
    #sleep(30)
    conn = 
urllib.request.urlopen("https://api.thingspeak.com/update?api_key=OEMFMV3
90IANB4YF&field7=NISOLDIPINE")
elif disease_predicted[0]==3:
    print("The person may have Covid Visit Hospital" )
    conn = 
urllib.request.urlopen("https://api.thingspeak.com/update?api_key=OEMFMV3
90IANB4YF&field7=VISIT_HOSPITAL")
```