# 

Members:

Florence Kyarikunda

Hunter Achieng’

Yvonne Katama

Mercy Birungi

Jane Wanjiru

Kay Linda

**TABLE OF CONTENTS**

[**INTRODUCTION**](#_ilg9dloans3x) **3**

[Fig. Circuit connection block image](#_l5em14hwtmxm) 3

[Fig.Problem logic flowchart](#_uulo3t71ijag) 4

[HARDWARE REQUIRED](#_hfx2ximy24v7) **4**

[SOFTWARE REQUIRED](#_d6zl54kwa2ss) **4**

[Fig.Arduino board connection with the hardware](#_63u1j82gyolg) 5

[INFLUX DB](#_sn37r5i6dzvm) **5**

[Fig. An image of Grafana Dashboard](#_st59nyn1yiuw) 6

# INTRODUCTION

With tons of new healthcare technology start-ups, IoT is rapidly revolutionizing the healthcare industry. In this project, we have designed a smart health monitoring system. Health is the most important part of any human’s life. Most humans live busy lives in which they do not have time to wait in queues when they visit the hospital just to be checked. Every day many lives are affected because the patients are not timely and properly operated.

Also, the real-time parameter values are not efficiently measured in clinics as well as in hospitals. Sometimes it becomes difficult for hospitals to check all the patient’s conditions that visit the hospital on a specific day. To deal with these types of situations our system is beneficial because it will use smart healthcare.

A smart healthcare system collects data about patients and uses predictive analytics to staff appropriately or change appointments as necessary. Our system is designed to be used in hospitals for measuring and monitoring various parameters like weight, pressure, heart rate, blood pressure, and temperature. This will make it possible to reduce wait times.

Collection of data sets will also be eased and this will guide doctors on how to serve patients more effectively. The data from the different sensors will be recorded using Arduino which will eventually be stored using the influx database and finally visualized using grafana

#### Flow chart showing the connection of the smart health system



#### 

#### 

# HARDWARE REQUIRED

Heart rate pulse sensor

HX711 Weight Measuring sensor

Blood Pressure sensor

DS18B20 Temperature sensor

ESP8266 Wi-Fi Module

Arduino Uno

LCD

# SOFTWARE REQUIRED

Arduino IDE

# STEPS TO CONNECT

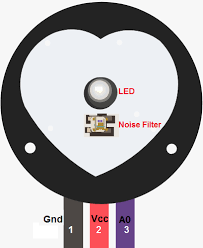
## Heart Rate Sensor

The heart rate sensor has three wires:

S: Signal color(purple) - connected to any analogue pin on the arduino.ie A0

+: Power color(red) - connect to either 3V or 5V.

-: Negative color(black) - connect to the ground.



#### Fig. Heart rate sensor front and back view

**Heart Rate codes**

int const PULSE\_SENSOR\_PIN = 0; // 'S' Signal pin connected to A0

int Signal; // Store incoming ADC data. Value can range from 0-1024

int Threshold = 550; // Determine which Signal to "count as a beat" and which to ignore.

void setup() {

pinMode(LED\_BUILTIN,OUTPUT); // Built-in LED will blink to your heartbeat

Serial.begin(9600); // Set comm speed for serial plotter window

}

void loop() {

Signal = analogRead(PULSE\_SENSOR\_PIN); // Read the sensor value

Serial.println(Signal); // Send the signal value to serial plotter

if(Signal > Threshold){ // If the signal is above threshold, turn on the LED

digitalWrite(LED\_BUILTIN,HIGH);

} else {

digitalWrite(LED\_BUILTIN,LOW); // Else turn off the LED

}

delay(10);

}

## Body Temperature Sensor

It has three pins:

Pin 1: connected to the Arduino’s 5v pin.

Pin2: connected to the analog power pins of the Arduino.

Pin 3: connected to ground.

## ESP8266 WIFI Module

The wifi module connection is as follows:

| **WIFI Module** | **Arduino** |
| --- | --- |
| Ground (GND) color: Black | Ground (GND) |
| VIN (Supplies module with power) color: Red | 3.3v |
| Enable (Switches on the module) color: Red | 3.3v |
| TX (Transmitter) color: Green or Yellow | RX (Receiver) |
| RX (Receiver) color: Purple or Green | TX (Transmitter) |

**Temperature Codes**

int pinTemp = A1; //This is where our Output data goes

void setup() {

Serial.begin(9600);

}

void loop() {

int temp = analogRead(pinTemp); //Read the analog pin

temp = temp \* 0.48828125; // convert output (mv) to readable celsius

Serial.print("Temperature: ");

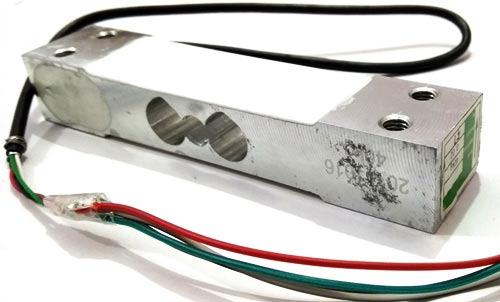
Serial.print(temp);

Serial.println("C"); //print the temperature status

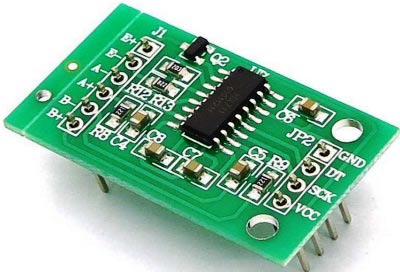
delay(1000);

}

## Weight Measurement HX711 Module



#### Fig. Weight measuring load cell



#### Fig.HX711 weight sensor module

A load cell of 200kg will be used to measure the force or weight.

The load cell converts the force to an electrical output but the signal is too low and needs to be amplified.

The HX711 weight sensor module is used to amplify the electrical output from the load cell and send the signal to the Arduino.

The load cell has four wires:

Red: Connected to the E+ pin of the HX711 module.

Black: Connected to the E- pin of the HX711 module.

White: Connected to the A-pin of the HX711 module.

Blue : Connected to the A+ pin of HX711 module.

The HX711 has extra pins: Ground, DT, SCK, VCC

GND: connected to Arduino’s ground

DT (Data IO connection): connected to A1 on Arduino

SCK: connected to A2 on Arduino

VCC: connected to LCD’s VDD

An LCD will be used to display the weight readings from the Arduino.

VEE : connects to ground on Arduino (-ve supply voltage)

VDD: connects to HX711’s VCC and Arduino’s 5v (+ve supply voltage)

+L: connects to the resistor and then the ground

-L, VSS(-ve voltage), and RW: connected to the resistor and attached to the ground

D7-14: connected to pin13

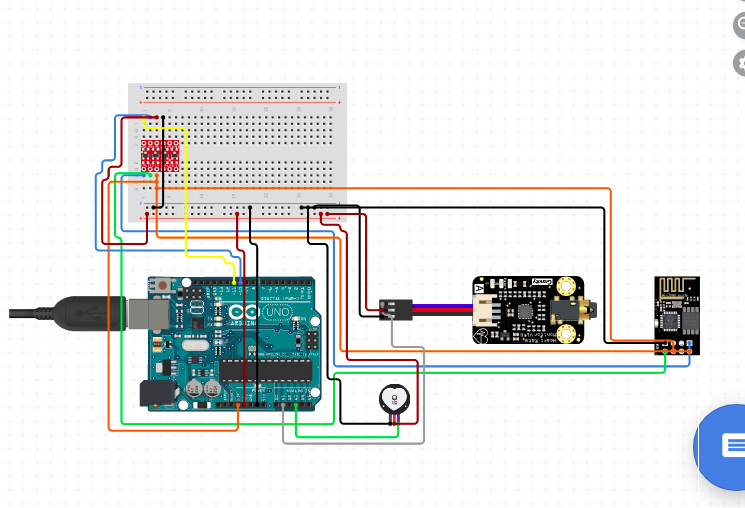
D6-13: connects to pin12

D5-12: connects to pin11

D4-11: connects to pin10

E-6: connects to pin9

RS-4: connects to pin8



#### Fig.Arduino board connection with the hardware

#### 

**Weight ‘s code**

#include <LiquidCrystal.h>

const int rs = 12, en = 11, d4 = 4, d5 = 5, d6 = 6, d7 = 7;

LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

#define DT A2

#define SCK A3

#define sw 3

long sample=0;

float val=0;

long count=0;

unsigned long readCount(void)

{

unsigned long Count;

unsigned char i;

pinMode(DT, OUTPUT);

digitalWrite(DT,HIGH);

digitalWrite(SCK,LOW);

Count=0;

pinMode(DT, INPUT);

while(digitalRead(DT));

for (i=0;i<24;i++)

{

digitalWrite(SCK,HIGH);

Count=Count<<1;

digitalWrite(SCK,LOW);

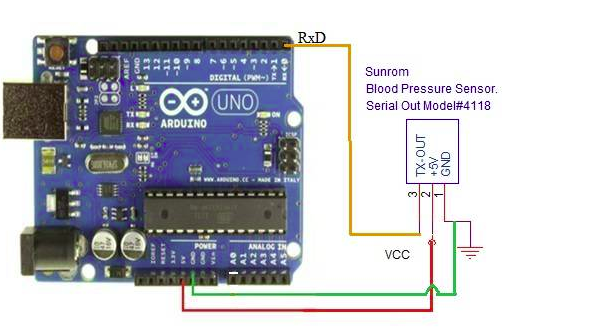
if(digitalRead(DT))

Count++;

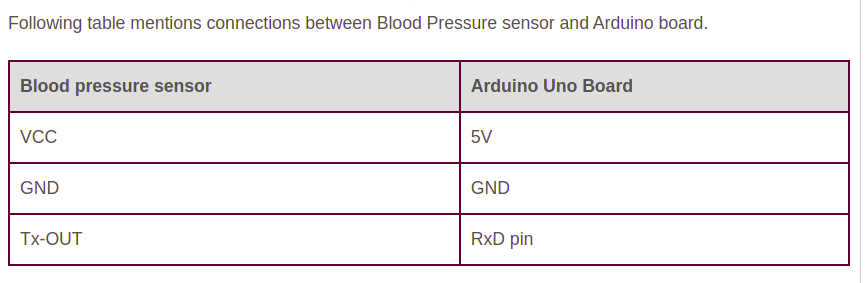
}

Blood Pressure Sensor

For the blood pressure sensor to be connected to the Arduino, the blood pressure monitor has to be dismantled in order to access the I2C EEPROM found in the blood pressure monitor. The I2C EEPROM is then interfaced with the Arduino so that it can send information to it . The blood pressure sensor provides output in 8 bit ASCII format which ranges from 000 to 255 . The Arduino is then set to communicate at 9600 baud/ beats per second.



#### Fig.Arduino board connection with the hardware



**Blood pressure code**

#include <SoftwareSerial.h>

int sensorValue; // variable to store the output from the sensor

SoftwareSerial serial(9,10);

char sbuffer[14], ch;

unsigned char pos;

unsigned char read1, read2, read3;

boolean newData=false;

void setup() {

Serial.begin(9600);

Serial.begin(9600);

Serial.println("<Arduino is ready>");

}

void loop() {

recvChar();

showNewData();

}

void recvChar() {

if (Serial.available() > 0)

{

while(Serial.available()>0)

{

ch = Serial.read(); //loop till character received

if(ch==0x0A) // if received character is <LF>, 0x0A, 10 then process buffer

{

pos = 0; // buffer position reset for next reading

newData=true;

// extract data from serial buffer to 8 bit integer value

// convert data from ASCII to decimal

read1 = ((sbuffer[1]-'0')\*100) + ((sbuffer[2]-'0')\*10) +(sbuffer[3]-'0');

read2 = ((sbuffer[6]-'0')\*100) + ((sbuffer[7]-'0')\*10) +(sbuffer[8]-'0');

read3 = ((sbuffer[11]-'0')\*100) + ((sbuffer[12]-'0')\*10) +(sbuffer[13]-'0');

}

else

{ //store serial data to buffer

sbuffer[pos] = ch;

pos++;

}

}

}

}

void showNewData() {

if (newData == true)

{

Serial.println("Calculating Results ... ");

Serial.println(read1);

Serial.println(read2);

Serial.println(read3);

newData = false;

}

}

# INFLUX DB

For our data storage, the Influx database will organize our data collected from sensors and other devices that comprise the internet of things into immutable runs of values for a single column of a series.

We will send our data to influx DB via the native UDP(User Datagram Protocol) plugin since it requires little processing and networking.InfluxDB has a compact, text-baseline protocol for writing data over both HTTP API as well as the UDP interface. We will use device-specified tags to help us specify details about where or how the data was taken or measured. For instance, for temperature, we will use a tag like a temperature, device =arduino(port number) value=(the value we will have).In our User Datagram protocol, we will enable it since it is normally disabled by default. To receive packets via UDP interface we will create a specified database using the command

CREATE DATABASE Arduino. After the data has been written to InfluxDB we can now issue a query to retrieve it. We will then visualize our data using grafana.



#### Fig. An image of Grafana Dashboard

# **BILL OF MATERIALS**

| NO. | COMPONENT | DESCRIPTION | QTY | IMAGE | SOURCE |
| --- | --- | --- | --- | --- | --- |
| 1. | Arduino Board | Arduino Uno | 1 |  | https://bit.ly/3eGL8UB |
| 2. | Bread Board | - | 2 |  | https://bit.ly/3eGL8UB |
| 3. | Pulse Sensor | ECG Heart Rate Monitor Sensor | 1 |  | https://bit.ly/3rnN8WR |
| 4. | Connecting Wires | Jumper Cables | 20 |  | https://bit.ly/3eGL8UB |
| 5. | Blood Pressure Sensor | Medical blood pressure sensor | 1 |  | https://bit.ly/3iBugzu |
| 6. | Weight Measuring cell | Load cell | 5@20kg |  | https://bit.ly/2TuswQe |
| 7. | Weight sensor | HX711 Weight Sensor | 1 |  | https://bit.ly/3kHWtar |
| 8. | Screen | LCD | 1 |  | https://bit.ly/36OqSvI |
| 9. | WIFI Module | ESP8266 WIFI Module | 1 |  | https://bit.ly/3kKf3Pk |
| 10. | Body temperature sensor | DS18B20 Temperature sensor | 1 |  | https://bit.ly/2UYK33i |