



BE MECHATRONICS SESSION (2020 - 2024)

Instrumentation and Measuring System

Project Report

Group Members

Roll No

Muhammad Waleed Tariq

201111

Muhammad Faiq Nasir

201075

Muhammad Saud Mubashir

201147

Submitted to: Prof. Umer Farooq

Submission Date: 15th June, 2022

Project Report

Initial Feasibility	4
Comparison with other similar products:	4
Objective	4
Introduction	4
• Sensors	4
• Microcontroller	4
• Board and PCB	5
• Connections	5
• Software	5
Sensors Introduction and Schematics	5
1- Water Level Sensor	5
2- ACS712 Current Sensor	5
3- Pulse Sensor	6
4- MAX 30102 Oximeter	6
5- TTP223 Touch Sensor	7
6- SD Card Module	7
7- TCRT 5000 IR Sensor	8
8- Voltage Sensor	8
9- IR Optical Tachometer	9
10. MLX 90614 IR Temperature Sensor	9
Block Diagram	10
Project Working	11
IR Sensor	11
MAX30102 Oximeter	11
Current Sensor	11
Voltage Sensor	11
Touch Sensor	11
Optical Tachometer	12
Temperature Sensor	12

Heart rate Sensor	12
SD Card Module	12
Water Level Sensor	12
Proteus	12
• Schematic	12
• PCB	13
Hardware Implementation	13
Output	15
• Arduino One	15
• Arduino Two	15
• Output Saved in SD Card	15
Code (Arduino One)	16
Code (Arduino Two)	23
Conclusion	26

Project Report

Health Monitoring System

Initial Feasibility

It is basically based on the health monitoring system in which it can be used to determine the general health conditions which one might face in daily life. Conditions like change in heartbeat or measuring the oxygen level in blood and many more other conditions can be measured by this system.

Comparison with other similar products:

Other similar products like the fitness bands and temperature measuring gun are some products which perform one or two of the functions that our system performs. Like in a fitness band it measures your body temperature, change in heartbeat and oxygen in blood. So these are the products that are similar to our product

Objective

The objective of this project is to make a Data Acquisition System by using different sensors that revolve around the same application and collect their data and store it in an SD Card using DAC.

Introduction

We designed a health assistive system which we integrated with different sensors to be able to quickly diagnose immediate health issues that may be caused due to any reason.

- **Sensors**

1. MAX 30102 Oximeter
2. ACS712 Current Sensor
3. Water Level Sensor
4. Pulse Sensor
5. TTP223 Touch Sensor
6. HW-125 DAC
7. TCRT 5000 IR Sensor
8. Voltage Sensor
9. MLX 90614 Temperature Sensor
10. IR Optical Counter Tachometer

- **Microcontroller**

1. Arduino Uno
2. Laptop
3. Serial Cable

- **Board and PCB**

1. Vero Board
2. Copper Sheet
3. FeCl_3
4. Iron

- **Connections**

1. Copper Cables
2. Soldering Kit

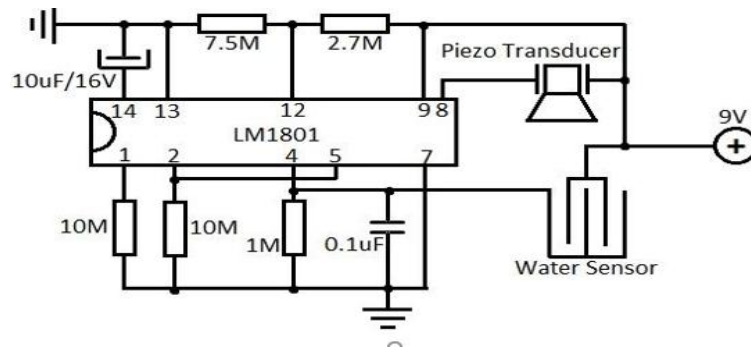
- **Software**

1. Proteus
2. Arduino IDE

Sensors Introduction and Schematics

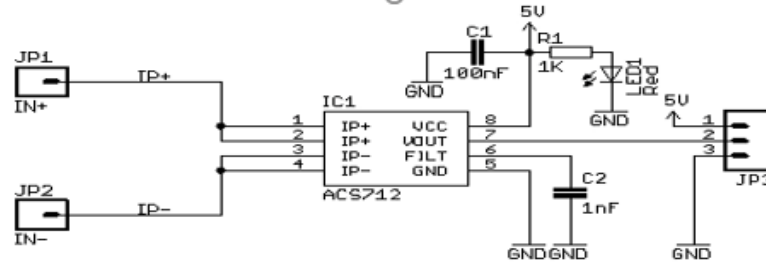
1- Water Level Sensor

The working principle of the water level sensor is that when it is put into a certain depth in the liquid to be measured, the pressure on the sensor's front surface is converted into the liquid level height. The calculation formula is $P = \rho \cdot g \cdot H + P_o$, in the formula P is the pressure on the liquid surface of the sensor, ρ is the density of the liquid to be measured, g is the local acceleration of gravity, P_o is the atmospheric pressure on the liquid surface, and H is the depth at which the sensor drops into the liquid.



2- ACS712 Current Sensor

Current flows through the onboard hall sensor circuit in its IC. The hall effect sensor detects the incoming current through its magnetic field generation. Once detected, the hall effect sensor generates a voltage proportional to its magnetic field that's then used to measure the amount of current.



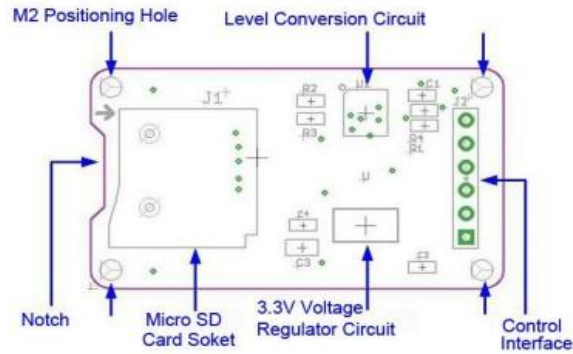
3- Pulse Sensor

The working principle of the pulse sensor is extremely simple. This sensor includes two faces where the first face is connected with an LED including an ambient light sensor whereas another face is connected with circuitry. This circuit aids in noise cancellation as well as amplification. On the front side, the LED is connected to a vein of a human body (ear tips or Fingertip), however, it should be located directly on top of a vein. The LED produces light that will drop directly on the vein. The veins in the body will have a flow of blood within them simply once the heart is pumping. So, if we check the blood flow, we can check the heartbeats also. If the blood flow is noticed then the sensor like ambient light will receive more light as they will be reflected through the blood, this small modification within received light can be analyzed on time to decide our heartbeats.



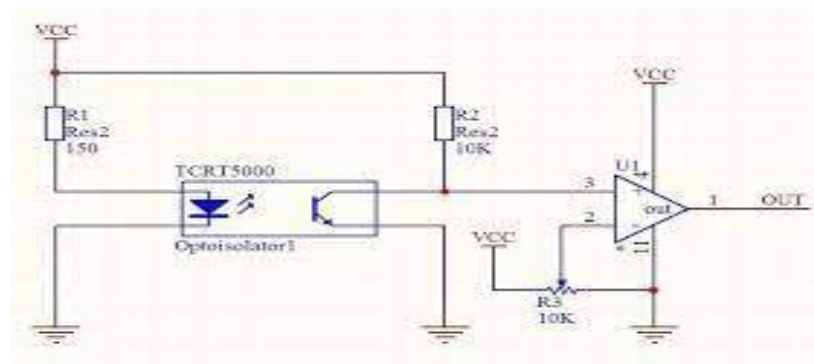
4- MAX 30102 Oximeter

The MAX30102 uses a method called photoplethysmography to measure the heart rate of someone. This method shines light on the skin and the perfusion of the blood is measured. One of the practical aspects of this approach is that it is possible to differentiate between the light reflected by the blood of an artery (produces an AC output) and other components of the body such as bones and tissues (produces a DC output). The photo-diode in the sensor then converts the light to current that we can use as comprehensible data. To counter difficulties such as skin tone differences LEDs with different wavelengths are used. In the MAX30102 there is an extra green LED for this purpose.



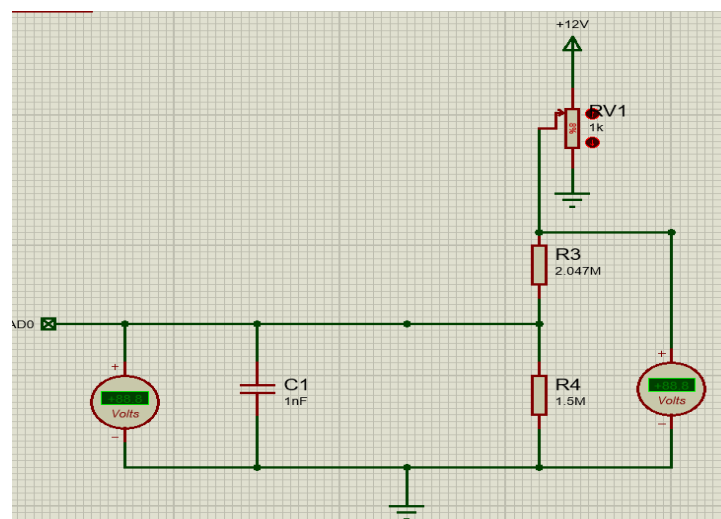
7- TCRT 5000 IR Sensor

The TCRT5000 itself works by transmitting infrared light from the LED and registering any reflected light on its phototransistor that alters the flow of current between its emitter and collector according to the level of light it receives.



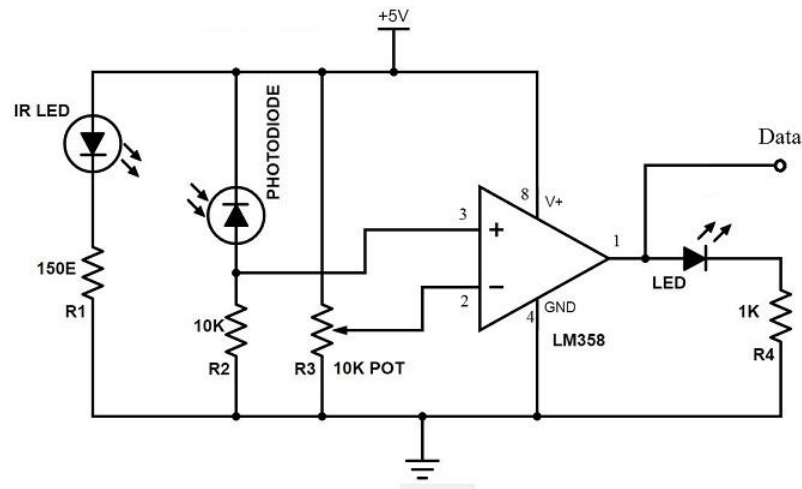
8- Voltage Sensor

This Sensor is designed by us. This sensor actually converts the 12V signal to an Arduino Readable 5V Signal using resistors and voltage divider rule then this signal is fed to Arduino through analogue port and using slope formula this 5V signal is again converted to its original value.



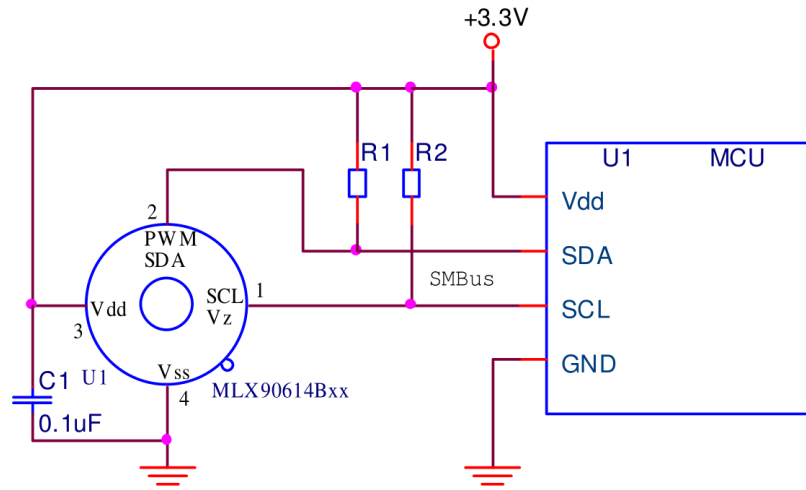
9- IR Optical Tachometer

Optical/Photo Tachometers operate by shining a light source, typically a LED light or Class 2 laser beam, against the rotating element. This light source creates a focused beam of light which will be reflected back off of a reflective object that is placed in its path. The optical sensor of the tachometer will be triggered as the light is reflected back towards the tachometer. By measuring the rate that the sensor receives the signal, a measurement of the rotational speed can be determined.



10. MLX 90614 IR Temperature Sensor

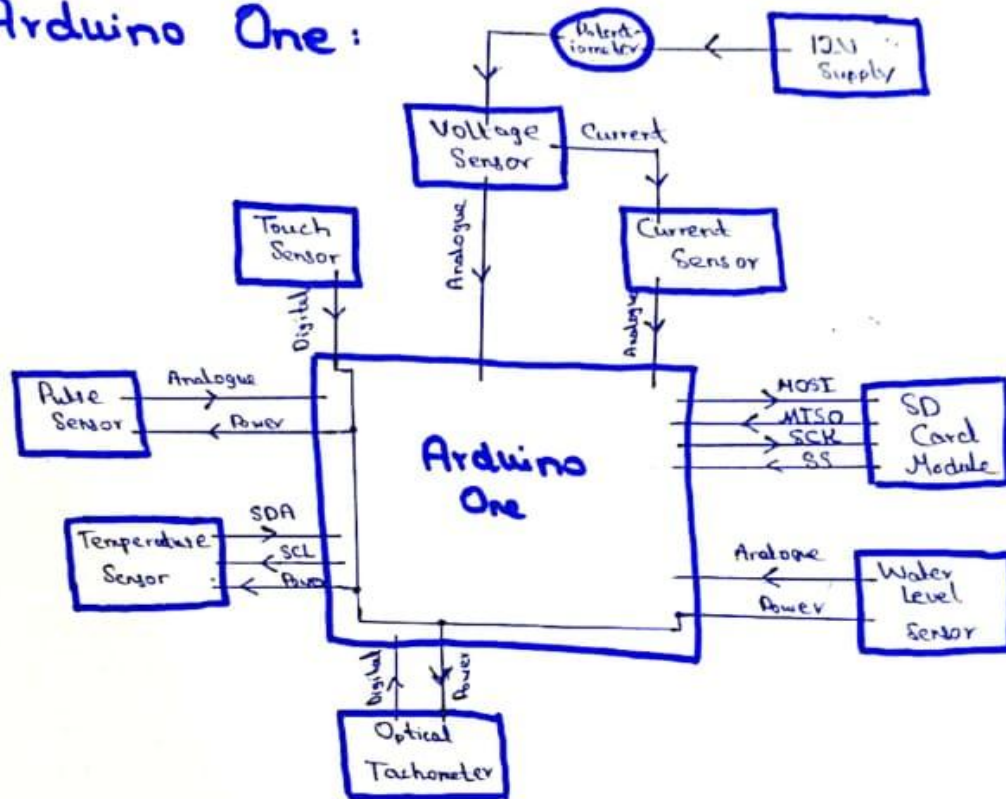
The working principle of infrared sensor MLX90614 is to transform the infrared radiation signal collected from objects and bodies into electrical signals, send the electrical signal into converter after noise amplification processing by amplifier, then the electrical signal is converted to digital signals and store the processed signals into the internal memory, finally send the signals into the SCM control system for further processing



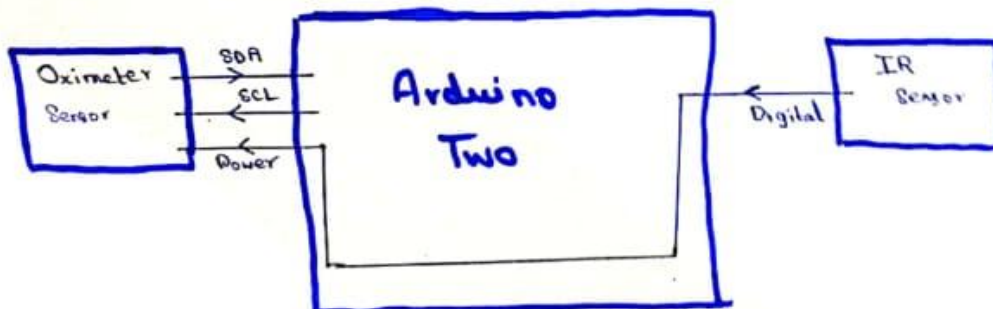
Block Diagram

Block Diagram

◎ Arduino One :



◎ Arduino Two :



Project Working

IR Sensor

IR sensor is used for smart switching of MAX30102 Oximeter whenever we pass out hand over the IR sensor module it sends a signal to the Arduino that turns on the VCC for the Oximeter, The Oximeter then measures the Blood Oxygen level of the person and displays it on the serial port

MAX30102 Oximeter

The Oximeter comes into play after the IR Sensor turns it on as explained above, the oximeter takes 100 samples of 16 bits but these samples were using 92% of the dynamic memory available on the Arduino which forced us to use a second Arduino for our project so that we can interface the remaining sensors. The oximeter takes these 100 samples using the IR LED on it, and uses these samples to calculate the oxygen level of the person who has placed their finger on the sensing area.

Current Sensor

The current sensor we opted for is a 5A current sensor module for the Arduino which has 2 input pins i.e., positive terminal pin and negative terminal pin. The sensor converts the 5A signal into a 5V signal that is read by the Arduino which converts these raw values into readable values to represent the current flowing through it.

Voltage Sensor

This sensor is fabricated on PCB by us ourselves, it converts 12 V DC to 5V signal to make it readable by Arduino's analog read command. We used a potentiometer to variate the supply voltage and, in the code, we used the mapping function to change this 5V signal back into a 12V reading. In theory we used the VDR to determine the ratio of 1:1.37 (Resistance) to convert 12V to 5V, but in practice we found the most suitable ratio to be 1:1.243 (Resistance).

Touch Sensor

This sensor was used in our project to turn on U encoder, Heart rate sensor and the Temperature sensor, when these sensors needed to be turned on the serial monitor instructed the user to touch the touch sensor to proceed with the readings.

Optical Tachometer

This sensor works for 10 seconds, multiplies the taken reading by 6 and converts to RPM to display it on the serial monitor

Temperature Sensor

The IR thermometer works by focusing light that is coming from the object in the form of IR rays and funneling that light into a detector. We have also calibrated this sensor to tell the ambient temperature along with the temperature of the target body.

Heart rate Sensor

A pulse sensor or any optical heart-rate sensor, for that matter, works by shining a green light ($\sim 550\text{nm}$) on the finger and measuring the amount of reflected light using a photosensor.

SD Card Module

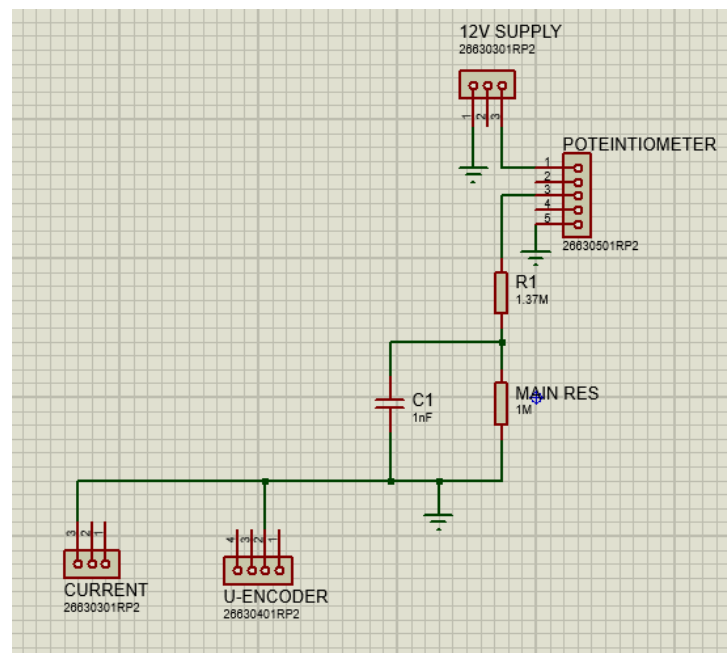
Using the MOSI and MISO pins it stores the acquainted data into a file in the SD card of the format .txt named as myFile.

Water Level Sensor

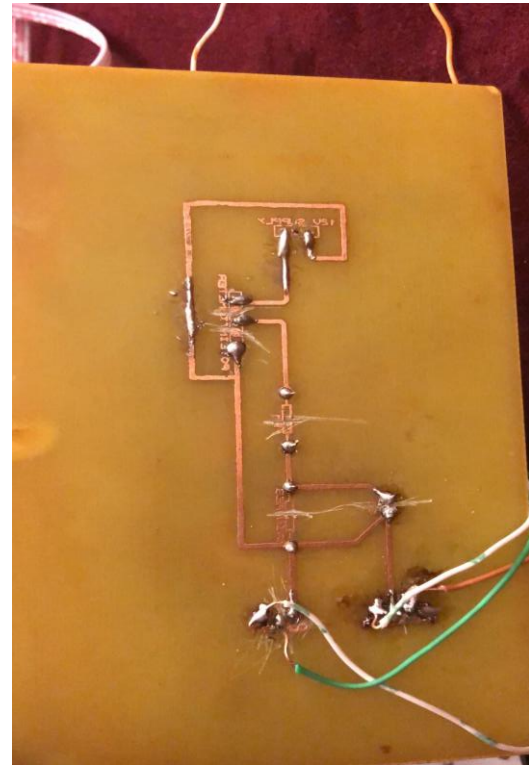
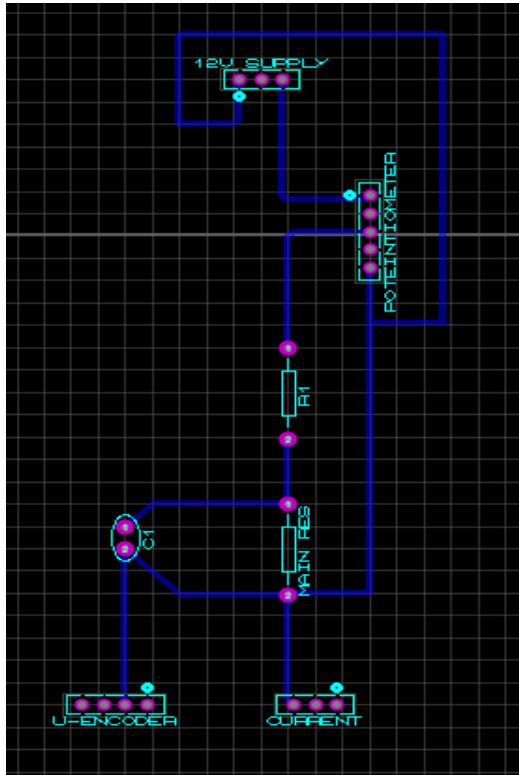
The working of the water level sensor is pretty straightforward. The series of exposed parallel conductors together acts as a variable resistor (just like a potentiometer) whose resistance varies according to the water level. The change in resistance corresponds to the distance from the top of the sensor to the surface of the water.

Proteus

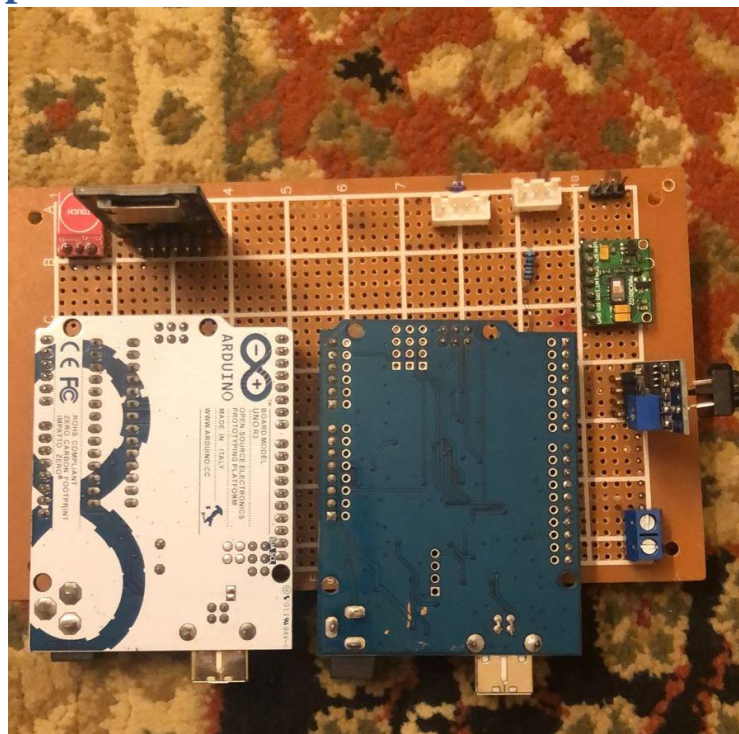
- Schematic

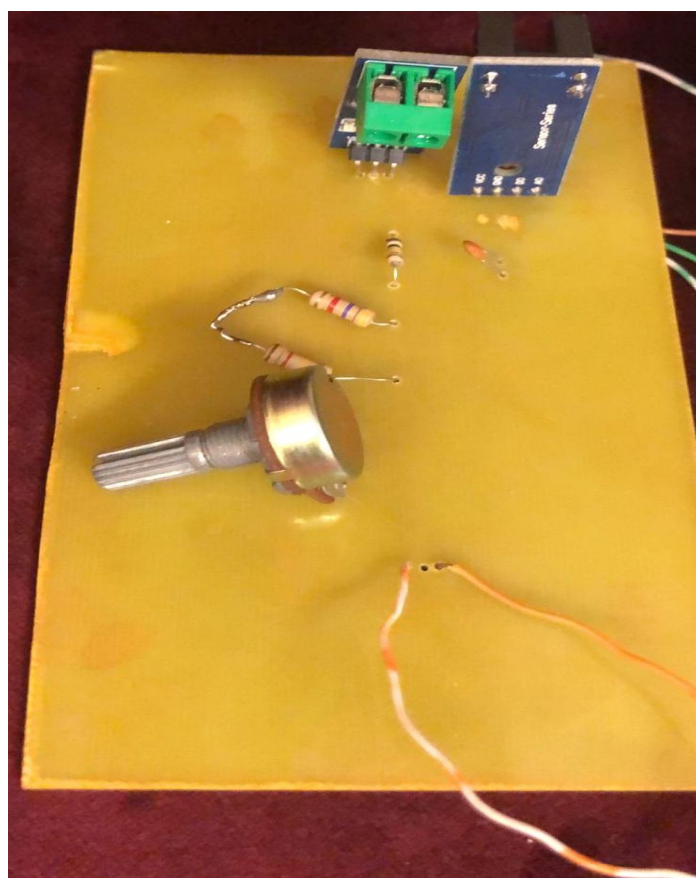
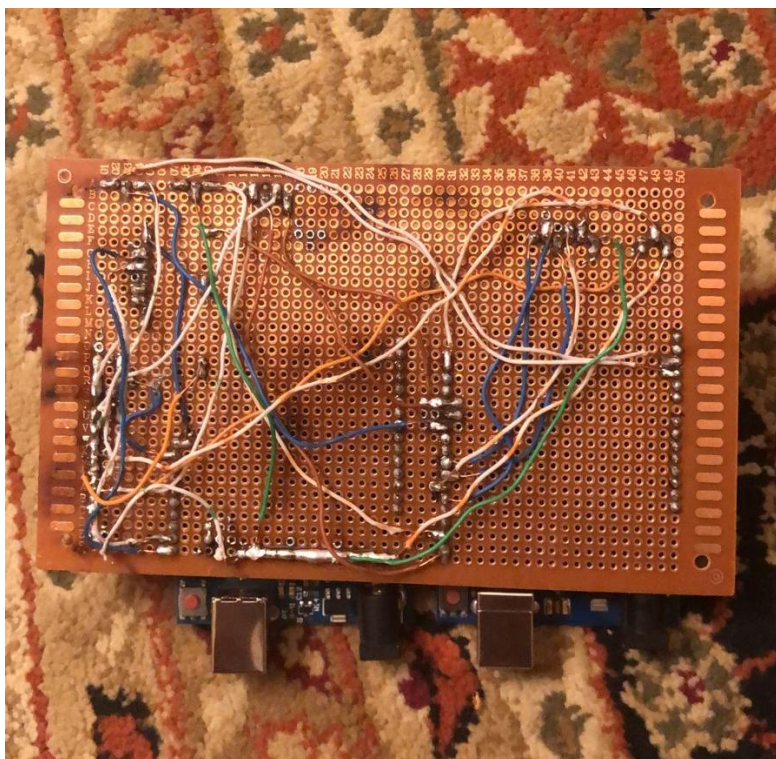


- PCB



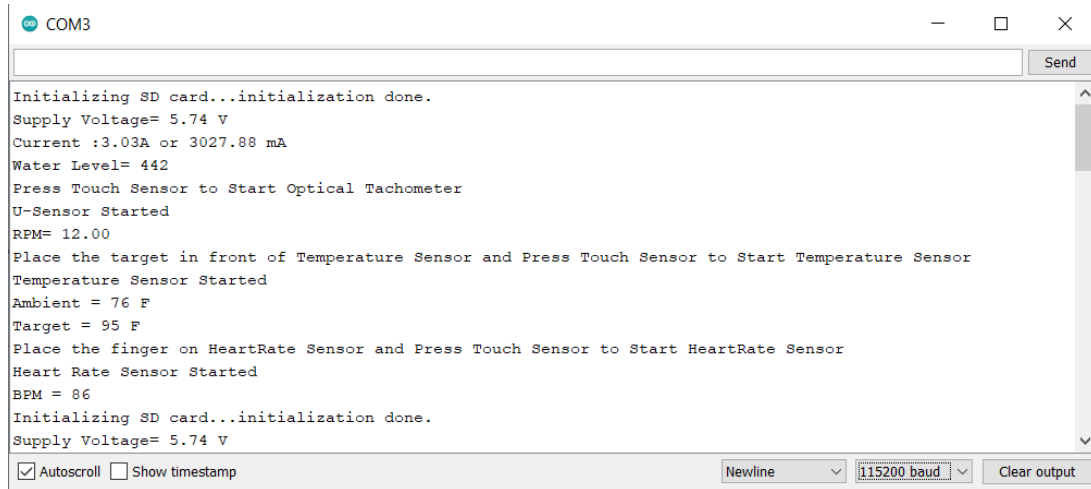
Hardware Implementation





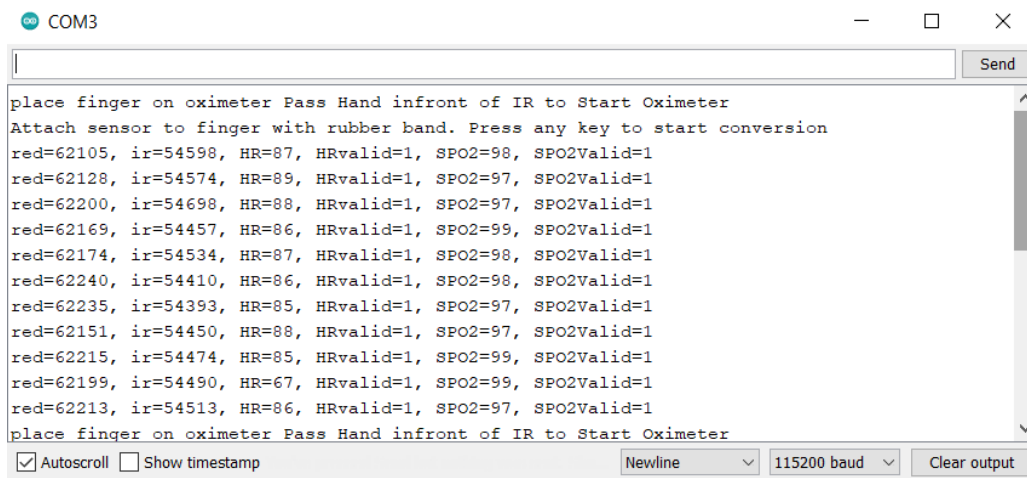
Output

- **Arduino One**



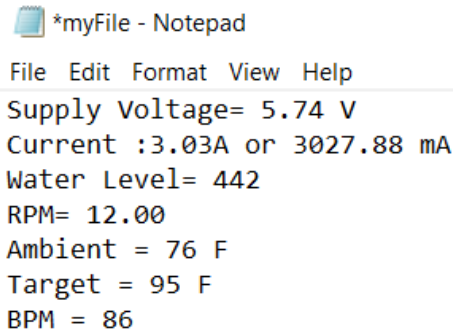
```
Initializing SD card...initialization done.
Supply Voltage= 5.74 V
Current :3.03A or 3027.88 mA
Water Level= 442
Press Touch Sensor to Start Optical Tachometer
U-Sensor Started
RPM= 12.00
Place the target in front of Temperature Sensor and Press Touch Sensor to Start Temperature Sensor
Temperature Sensor Started
Ambient = 76 F
Target = 95 F
Place the finger on HeartRate Sensor and Press Touch Sensor to Start HeartRate Sensor
Heart Rate Sensor Started
BPM = 86
Initializing SD card...initialization done.
Supply Voltage= 5.74 V
```

- **Arduino Two**



```
place finger on oximeter Pass Hand infront of IR to Start Oximeter
Attach sensor to finger with rubber band. Press any key to start conversion
red=62105, ir=54598, HR=87, HRvalid=1, SPO2=98, SPO2Valid=1
red=62128, ir=54574, HR=89, HRvalid=1, SPO2=97, SPO2Valid=1
red=62200, ir=54698, HR=88, HRvalid=1, SPO2=97, SPO2Valid=1
red=62169, ir=54457, HR=86, HRvalid=1, SPO2=99, SPO2Valid=1
red=62174, ir=54534, HR=87, HRvalid=1, SPO2=98, SPO2Valid=1
red=62240, ir=54410, HR=86, HRvalid=1, SPO2=98, SPO2Valid=1
red=62235, ir=54393, HR=85, HRvalid=1, SPO2=97, SPO2Valid=1
red=62151, ir=54450, HR=88, HRvalid=1, SPO2=97, SPO2Valid=1
red=62215, ir=54474, HR=85, HRvalid=1, SPO2=99, SPO2Valid=1
red=62199, ir=54490, HR=67, HRvalid=1, SPO2=99, SPO2Valid=1
red=62213, ir=54513, HR=86, HRvalid=1, SPO2=97, SPO2Valid=1
place finger on oximeter Pass Hand infront of IR to Start Oximeter
```

- **Output Saved in SD Card**



```
*myFile - Notepad
File Edit Format View Help
Supply Voltage= 5.74 V
Current :3.03A or 3027.88 mA
Water Level= 442
RPM= 12.00
Ambient = 76 F
Target = 95 F
BPM = 86
```

Code (Arduino One)

```
#include <Wire.h>
#include <Adafruit_MLX90614.h>
#include "MAX30105.h"
#include "spo2_algorithm.h"
#include <SPI.h>
#include <SD.h>

//SD Card Variables
File myFile;
const int chipSelect = 10;

//Voltage Sensor Variables
float valVoltage = 0;
int rawVoltage = 0;
int VoltagePin=A0;

//Current Sensor Variables
int CurrentPin=A1;

//U-Sensor Variables
int USensorPin = 7;
int USensorPower = 3;
unsigned long start_time = 0;
unsigned long end_time = 0;
int steps=0;
float temp=0;
float rpm=0;

//Touch Sensor Variable
const int TouchSensorPin = 2;
const int TouchSensorPower = 9;

//Water Level Variable
int LevelPin=A2;

//Temperature Sensor Variables
Adafruit_MLX90614 mlx = Adafruit_MLX90614();
int TempSensorPower=5;

//HeartRate Variables
int pulsePin = A3;           // Pulse Sensor purple wire connected to analog pin A0
int blinkPin = 6;           // pin to blink led at each beat
int HeartRatePower=4;
// Volatile Variables, used in the interrupt service routine!
volatile int BPM;           // int that holds raw Analog in 0. updated every 2mS
```



```

volatile int Signal;           // holds the incoming raw data
volatile int IBI = 600;       // int that holds the time interval between beats! Must be seeded!
volatile boolean Pulse = false; // "True" when User's live heartbeat is detected. "False" when not
                                // a "live beat".
volatile boolean QS = false;   // becomes true when Arduino finds a beat.
static boolean serialVisual = true; // Set to 'false' by Default. Re-set to 'true' to see Arduino Serial
Monitor ASCII Visual Pulse
volatile int rate[10];         // array to hold last ten IBI values
volatile unsigned long sampleCounter = 0; // used to determine pulse timing
volatile unsigned long lastBeatTime = 0; // used to find IBI
volatile int P = 512;          // used to find peak in pulse wave, seeded
volatile int T = 512;          // used to find trough in pulse wave, seeded
volatile int thresh = 525;     // used to find instant moment of heart beat, seeded
volatile int amp = 100;        // used to hold amplitude of pulse waveform, seeded
volatile boolean firstBeat = true; // used to seed rate array so we startup with reasonable BPM
volatile boolean secondBeat = false; // used to seed rate array so we startup with reasonable
BPM

```

```

void setup() {
  Serial.begin(115200);
  Serial.print("Initializing SD card...");
  if (!SD.begin(10)) {
    Serial.println("initialization failed!");
    while (1);
  }
  Serial.println("initialization done.");
  // open the file. note that only one file can be open at a time,
  // so you have to close this one before opening another.
  myFile = SD.open("Data.txt", FILE_WRITE);
  // if the file opened okay, write to it:
  if (myFile) {
    Serial.println("Writing to Data.txt...");
    myFile.println("*****Data Aqquisition Started*****");
  }
  pinMode(USensorPower,OUTPUT);
  pinMode(USensorPin,INPUT_PULLUP);
  pinMode(TouchSensorPin, INPUT);
  pinMode(TempSensorPower, OUTPUT);
  pinMode(HeartRatePower, OUTPUT);
  pinMode(blinkPin,OUTPUT);
  pinMode(TouchSensorPower,HIGH);
}

```

```

void loop() {
  VoltageSensor();
  CurrentSensor();
}

```

```

    WaterLevel();
    Serial.println("Press Touch Sensor to Start Optical Tachometer");
    TouchSensor();
    USensor();
    Serial.println("Place the target in front of Temperature Sensor and Press Touch Sensor to Start
Temperature Sensor");
    TouchSensor();
    TemperatureSensor();
    Serial.println("Place the finger on HeartRate Sensor and Press Touch Sensor to Start HeartRate
Sensor");
    TouchSensor();
    HeartRate();
}

void VoltageSensor(){
    rawVoltage=analogRead(VoltagePin);
    valVoltage = mapfloat(rawVoltage, 0, 1023, 0, 12);
    Serial.print("Supply Voltage= ");
    Serial.print(valVoltage);
    Serial.println(" V");
    myFile.print("Supply Voltage= ");
    myFile.print(valVoltage);
    myFile.println(" V");
}

float mapfloat(float x, float in_min, float in_max, float out_min, float out_max){
    return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
}

void CurrentSensor(){
    float averageCurrent = 0;
    for(int i = 0; i < 1000; i++) {
        averageCurrent = averageCurrent + (.0264 * analogRead(CurrentPin) -13.51);//for the 5A mode,
        //averageCurrent = averageCurrent + (.049 * analogRead(CurrentPin) -25);// for 20A mode
        delay(1);
    }
    Serial.print("Current :");
    Serial.print(averageCurrent/1000);
    Serial.print("A or ");
    Serial.print(averageCurrent);
    Serial.println(" mA");
    myFile.print("Current :");
    myFile.print(averageCurrent/1000);
    myFile.print("A or");
    myFile.print(averageCurrent);
    myFile.println("mA");
}

```

```

void USensor(){
  digitalWrite(USensorPower, HIGH);
  Serial.println("U-Sensor Started");
  delay (1000);
  steps=0;
  start_time = millis();
  end_time = start_time+10000;
  while(millis()<end_time)
  {
    if(digitalRead(USensorPin))
    {
      steps=steps+1;
    }
  }
  temp=steps*6;
  rpm=(temp/2);
  Serial.print("RPM= ");
  Serial.println(rpm);
  myFile.print("RPM= ");
  myFile.println(rpm);
  digitalWrite(USensorPower,LOW);
}

void TouchSensor(){
  digitalWrite(TouchSensorPower, HIGH);
  delay(1000);
  while(1){
    if(digitalRead(TouchSensorPin) == HIGH){
      break;
    }
  }
  digitalWrite(TouchSensorPower, LOW);
}

void WaterLevel(){
  int Level=analogRead(LevelPin);
  Serial.print("Water Level= ");
  Serial.println(Level);
  int x= map(Level, 0, 1023, 0, 10);
  myFile.print("Water Level= ");
  myFile.println(x);
}

void TemperatureSensor(){
  digitalWrite(TempSensorPower, HIGH);

```

```

Serial.println("Temperature Sensor Started");
mlx.begin();
delay(1000);
Serial.print("Ambient= ");
Serial.print(mlx.readAmbientTempF());
Serial.println(" F");
Serial.print("Target ");
Serial.print(mlx.readObjectTempF());
Serial.println(" F ");
myFile.print("Ambient= ");
myFile.print(mlx.readAmbientTempF());
myFile.println(" F");
myFile.print("Target ");
myFile.print(mlx.readObjectTempF());
myFile.println(" F ");
digitalWrite(TempSensorPower, LOW);
}

```

```

void HeartRate(){
  digitalWrite(HeartRatePower, HIGH);
  Serial.println("Heart Rate Sensor Started");
  delay(1000);
  interruptSetup();          // sets up to read Pulse Sensor signal every 2mS
  serialOutput();
  if (QS == true) // A Heartbeat Was Found
  {
    // BPM and IBI have been Determined
    // Quantified Self "QS" true when arduino finds a heartbeat
    serialOutputWhenBeatHappens(); // A Beat Happened, Output that to serial.
    QS = false; // reset the Quantified Self flag for next time
  }
  delay(20); // take a break
  digitalWrite(HeartRatePower, HIGH);
}

```

```

void interruptSetup(){
  // Initializes Timer2 to throw an interrupt every 2mS.
  TCCR2A = 0x02; // DISABLE PWM ON DIGITAL PINS 3 AND 11, AND GO INTO CTC
MODE
  TCCR2B = 0x06; // DON'T FORCE COMPARE, 256 PRESCALER
  OCR2A = 0x7C; // SET THE TOP OF THE COUNT TO 124 FOR 500Hz SAMPLE RATE
  TIMSK2 = 0x02; // ENABLE INTERRUPT ON MATCH BETWEEN TIMER2 AND OCR2A
  sei(); // MAKE SURE GLOBAL INTERRUPTS ARE ENABLED
}
void serialOutput(){ // Decide How To Output Serial.
  if (serialVisual == true)
  {

```

```

    arduinoSerialMonitorVisual('-', Signal);    // goes to function that makes Serial Monitor
Visualizer
}
else
{
    sendDataToSerial('S', Signal);    // goes to sendDataToSerial function
}
}
void serialOutputWhenBeatHappens(){
if (serialVisual == true) // Code to Make the Serial Monitor Visualizer Work
{
    Serial.print(" Heart-Beat Found "); //ASCII Art Madness
    Serial.print("BPM: ");
    Serial.println(BPM);
    myFile.print(" Heart-Beat Found "); //ASCII Art Madness
    myFile.print("BPM: ");
    myFile.println(BPM);
}
else
{
    sendDataToSerial('B',BPM); // send heart rate with a 'B' prefix
    sendDataToSerial('Q',IBI); // send time between beats with a 'Q' prefix
}
}
void arduinoSerialMonitorVisual(char symbol, int data ){
    const int sensorMin = 0;    // sensor minimum, discovered through experiment
    const int sensorMax = 1024;    // sensor maximum, discovered through experiment
    int sensorReading = data; // map the sensor range to a range of 12 options:
    int range = map(sensorReading, sensorMin, sensorMax, 0, 11);
    // do something different depending on the
    // range value:
}
void sendDataToSerial(char symbol, int data ){
    Serial.print(symbol);
    Serial.println(data);
}
ISR(TIMER2_COMPA_vect){ //triggered when Timer2 counts to 124
    cli();                // disable interrupts while we do this
    Signal = analogRead(pulsePin);    // read the Pulse Sensor
    sampleCounter += 2;    // keep track of the time in mS with this variable
    int N = sampleCounter - lastBeatTime;    // monitor the time since the last beat to avoid noise
    // find the peak and trough of the pulse wave
    if(Signal < thresh && N > (IBI/5)*3) // avoid dichrotic noise by waiting 3/5 of last IBI
    {
        if (Signal < T) // T is the trough
        {

```

```

    T = Signal; // keep track of lowest point in pulse wave
}
}
if(Signal > thresh && Signal > P)
{
    // thresh condition helps avoid noise
    P = Signal; // P is the peak
} // keep track of highest point in pulse wave
// NOW IT'S TIME TO LOOK FOR THE HEART BEAT
// signal surges up in value every time there is a pulse
if (N > 250)
{
    // avoid high frequency noise
    if ( (Signal > thresh) && (Pulse == false) && (N > (IBI/5)*3) )
    {
        Pulse = true; // set the Pulse flag when we think there is a pulse
        digitalWrite(blinkPin,HIGH); // turn on pin 13 LED
        IBI = sampleCounter - lastBeatTime; // measure time between beats in mS
        lastBeatTime = sampleCounter; // keep track of time for next pulse
        if(secondBeat)
        {
            // if this is the second beat, if secondBeat == TRUE
            secondBeat = false; // clear secondBeat flag
            for(int i=0; i<=9; i++) // seed the running total to get a realistic BPM at startup
            {
                rate[i] = IBI;
            }
        }
        if(firstBeat) // if it's the first time we found a beat, if firstBeat == TRUE
        {
            firstBeat = false; // clear firstBeat flag
            secondBeat = true; // set the second beat flag
            sei(); // enable interrupts again
            return; // IBI value is unreliable so discard it
        }
        // keep a running total of the last 10 IBI values
        word runningTotal = 0; // clear the runningTotal variable
        for(int i=0; i<=8; i++)
        {
            // shift data in the rate array
            rate[i] = rate[i+1]; // and drop the oldest IBI value
            runningTotal += rate[i]; // add up the 9 oldest IBI values
        }
        rate[9] = IBI; // add the latest IBI to the rate array
        runningTotal += rate[9]; // add the latest IBI to runningTotal
        runningTotal /= 10; // average the last 10 IBI values
        BPM = 60000/runningTotal; // how many beats can fit into a minute? that's BPM!
        QS = true; // set Quantified Self flag
        // QS FLAG IS NOT CLEARED INSIDE THIS ISR
    }
}

```

```

}
if (Signal < thresh && Pulse == true)
{
    // when the values are going down, the beat is over
    digitalWrite(blinkPin,LOW);          // turn off pin 13 LED
    Pulse = false;                        // reset the Pulse flag so we can do it again
    amp = P - T;                          // get amplitude of the pulse wave
    thresh = amp/2 + T;                   // set thresh at 50% of the amplitude
    P = thresh;                           // reset these for next time
    T = thresh;
}
if (N > 2500)
{
    // if 2.5 seconds go by without a beat
    thresh = 512;                         // set thresh default
    P = 512;                             // set P default
    T = 512;                             // set T default
    lastBeatTime = sampleCounter;         // bring the lastBeatTime up to date
    firstBeat = true;                     // set these to avoid noise
    secondBeat = false;                   // when we get the heartbeat back
}
sei();                                  // enable interrupts when youre done!
}

```

Code (Arduino Two)

```

#include <Wire.h>
#include "MAX30105.h"
#include "spo2_algorithm.h"

```

```

MAX30105 particleSensor;

```

```

#define MAX_BRIGHTNESS 255
#ifdef __AVR_ATmega328P__ || defined(__AVR_ATmega168__)
//Arduino Uno doesn't have enough SRAM to store 100 samples of IR led data and red led data in
32-bit format
//To solve this problem, 16-bit MSB of the sampled data will be truncated. Samples become 16-
bit data.
uint16_t irBuffer[100]; //infrared LED sensor data
uint16_t redBuffer[100]; //red LED sensor data
#else
uint32_t irBuffer[100]; //infrared LED sensor data
uint32_t redBuffer[100]; //red LED sensor data
#endif

int32_t bufferLength; //data length
int32_t spo2; //SPO2 value
int8_t validSPO2; //indicator to show if the SPO2 calculation is valid
int32_t heartRate; //heart rate value

```

```

int8_t validHeartRate; //indicator to show if the heart rate calculation is valid

byte pulseLED = 11; //Must be on PWM pin
byte readLED = 13; //Blinks with each data read
int power= 8;
int pinIR=9;
bool x;

void setup()
{
  Serial.begin(115200); // initialize serial communication at 115200 bits per second:

  pinMode(pulseLED, OUTPUT);
  pinMode(readLED, OUTPUT);
  pinMode(power, OUTPUT);
  pinMode(pinIR, INPUT);
}

void loop()
{
  Serial.println("place finger on oximeter Pass Hand infront of IR to Start Oximeter");
  x= digitalRead(pinIR);
  if (x=1){
    Oximeter();
  }
  delay(1000);
}

void Oximeter(){
  digitalWrite(power, HIGH);
  // Initialize sensor
  if (!particleSensor.begin(Wire, I2C_SPEED_FAST)) //Use default I2C port, 400kHz speed
  {
    Serial.println(F("MAX30105 was not found. Please check wiring/power."));
    while (1);
  }

  Serial.println(F("Attach sensor to finger with rubber band. Press any key to start conversion"));
  while (Serial.available() == 0) ; //wait until user presses a key
  Serial.read();

  byte ledBrightness = 60; //Options: 0=Off to 255=50mA
  byte sampleAverage = 4; //Options: 1, 2, 4, 8, 16, 32
  byte ledMode = 2; //Options: 1 = Red only, 2 = Red + IR, 3 = Red + IR + Green
  byte sampleRate = 100; //Options: 50, 100, 200, 400, 800, 1000, 1600, 3200

```



```

int pulseWidth = 411; //Options: 69, 118, 215, 411
int adcRange = 4096; //Options: 2048, 4096, 8192, 16384

particleSensor.setup(ledBrightness, sampleAverage, ledMode, sampleRate, pulseWidth,
adcRange); //Configure sensor with these settings
bufferLength = 100; //buffer length of 100 stores 4 seconds of samples running at 25sps

//read the first 100 samples, and determine the signal range
for (byte i = 0 ; i < bufferLength ; i++)
{
    while (particleSensor.available() == false) //do we have new data?
        particleSensor.check(); //Check the sensor for new data

    redBuffer[i] = particleSensor.getRed();
    irBuffer[i] = particleSensor.getIR();
    particleSensor.nextSample(); //We're finished with this sample so move to next sample

    Serial.print(F("red="));
    Serial.print(redBuffer[i], DEC);
    Serial.print(F(", ir="));
    Serial.println(irBuffer[i], DEC);
}

//calculate heart rate and SpO2 after first 100 samples (first 4 seconds of samples)
maxim_heart_rate_and_oxygen_saturation(irBuffer, bufferLength, redBuffer, &spo2,
&validSPO2, &heartRate, &validHeartRate);

//Continuously taking samples from MAX30102. Heart rate and SpO2 are calculated every 1
second
while (1)
{
    //dumping the first 25 sets of samples in the memory and shift the last 75 sets of samples to the
top
    for (byte i = 25; i < 100; i++)
    {
        redBuffer[i - 25] = redBuffer[i];
        irBuffer[i - 25] = irBuffer[i];
    }

    //take 25 sets of samples before calculating the heart rate.
    for (byte i = 75; i < 100; i++)
    {
        while (particleSensor.available() == false) //do we have new data?
            particleSensor.check(); //Check the sensor for new data

        digitalWrite(readLED, !digitalRead(readLED)); //Blink onboard LED with every data read
    }
}

```

```

redBuffer[i] = particleSensor.getRed();
irBuffer[i] = particleSensor.getIR();
particleSensor.nextSample(); //We're finished with this sample so move to next sample

//send samples and calculation result to terminal program through UART
Serial.print(F("red="));
Serial.print(redBuffer[i], DEC);
Serial.print(F(", ir="));
Serial.print(irBuffer[i], DEC);

Serial.print(F(", HR="));
Serial.print(heartRate, DEC);

Serial.print(F(", HRvalid="));
Serial.print(validHeartRate, DEC);

Serial.print(F(", SPO2="));
Serial.print(spo2, DEC);

Serial.print(F(", SPO2Valid="));
Serial.println(validSPO2, DEC);
}

//After gathering 25 new samples recalculate HR and SP02
maxim_heart_rate_and_oxygen_saturation(irBuffer,    bufferLength,    redBuffer,    &spo2,
&validSPO2, &heartRate, &validHeartRate);
}
}

```

Conclusion

This system is equipped with sensors surrounding health diagnosis, it can measure your BPM, Temperature and Blood oxygen level. The system is designed so that the sensors work only when needed by touching the touch sensor or by swiping your hand over the IR sensor. The added sanitizer bottle is also equipped with a water level sensor that tell you how much sanitizer is left in the bottle so that you are always aware of when to fill it up.