Near East University

Yakın Doğu Üniversitesi, Lefkoşa KKTC

FACULTY OF ENGINEERING

DEPARTMENT OF BIOMEDICAL ENGINEERING

2013/2014 ACADEMIC YEAR

BME 402
GRADUATION PROJECT – II

SUBMITTED TO: **EYLEM AKAY**

SUBMITTED BY: **CANBERK DEMIROLUK (20113250)**

CEMAY ULUĞ (20090391)

Nicosia 2013

Index

List of Figures

1. Introduction

The e-Health Sensor Platform allows to perform biometric and medical applications where body monitoring is needed by using 10 different sensors: pulse, oxygen in blood (SPO2), airflow (breathing), body temperature, electrocardiogram (ECG), glucometer, galvanic skin response (GSR - sweating), blood pressure (sphygmomanometer), patient position (accelerometer) and muscle/eletromyography sensor (EMG).

This information can be used to monitor in real time the state of a patient or to get sensitive data in order to be subsequently analysed for medical diagnosis. Biometric information gathered can be wirelessly sent using any of the 6 connectivity options available: Wi-Fi, 3G, GPRS, Bluetooth, 802.15.4 and ZigBee depending on the application.

If real time image diagnosis is needed a camera can be attached to the 3G module in order to send photos and videos of the patient to a medical diagnosis center.

Data can be sent to the Cloud in order to perform permanent storage or visualized in real time by sending the data directly to a laptop or Smartphone. iPhone and Android applications have been designed in order to easily see the patient's information.

Figure 1.1 Representation of Device

2. Features

The pack we are going to use in this project is the eHealth Sensor platform. The e-Health Sensor Shield is fully compatible with Arduino USB versions, Duemilanove and Mega.

- 8 non-invasive + 1 invasive medical sensors
- Storage and use of glucose measurements.
- Monitoring ECG signal.
- Monitoring EMG signals.
- Airflow control of patient.
- Body temperature data.
- Galvanic skin response measurements.
- Body position detection.
- Pulse and oxygen functions.
- Blood pressure control device.
- Multiple data visualization systems.
- Compatible with all UART device.

2.1 Electrical features

The e-Health shield can be powered by the PC or by an external power supply. Some of the USB ports on computers are not able to give all the current the module needs to work.

3. The shield

e-Health PCB

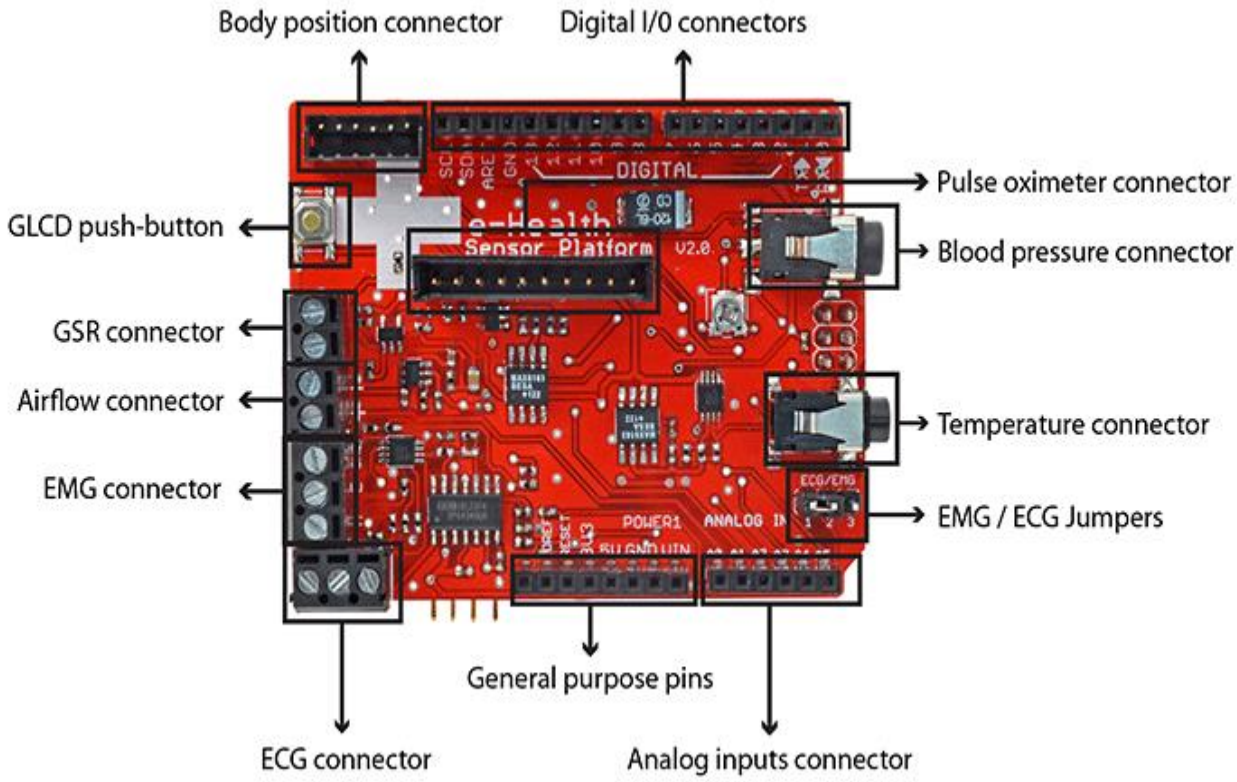

Figure 2.1 E-Health PCB Connections

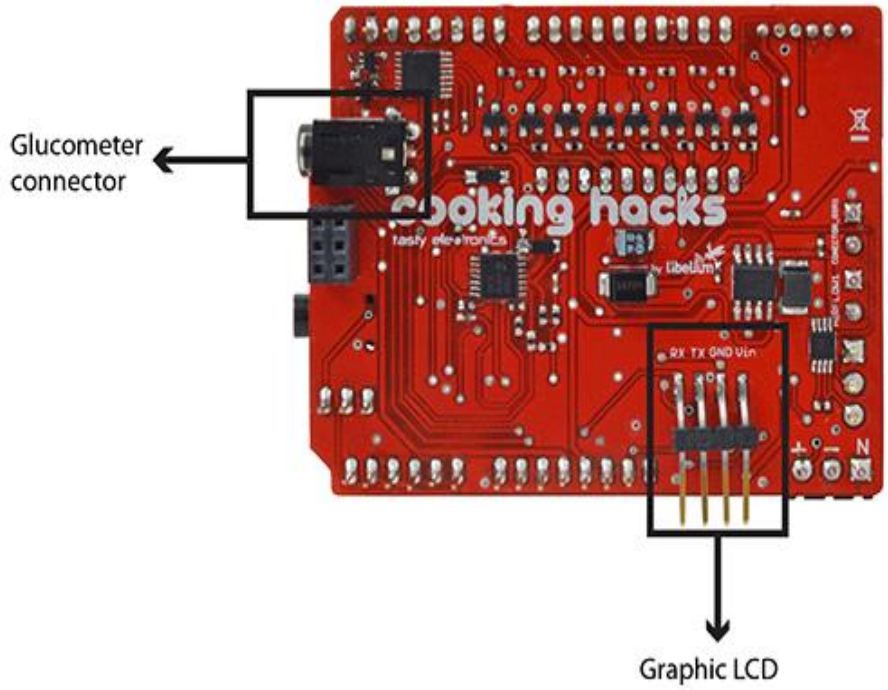

Figure 2.2 Glucometer and LCD Connections

3.1 Schematics

e-Health shield over Arduino

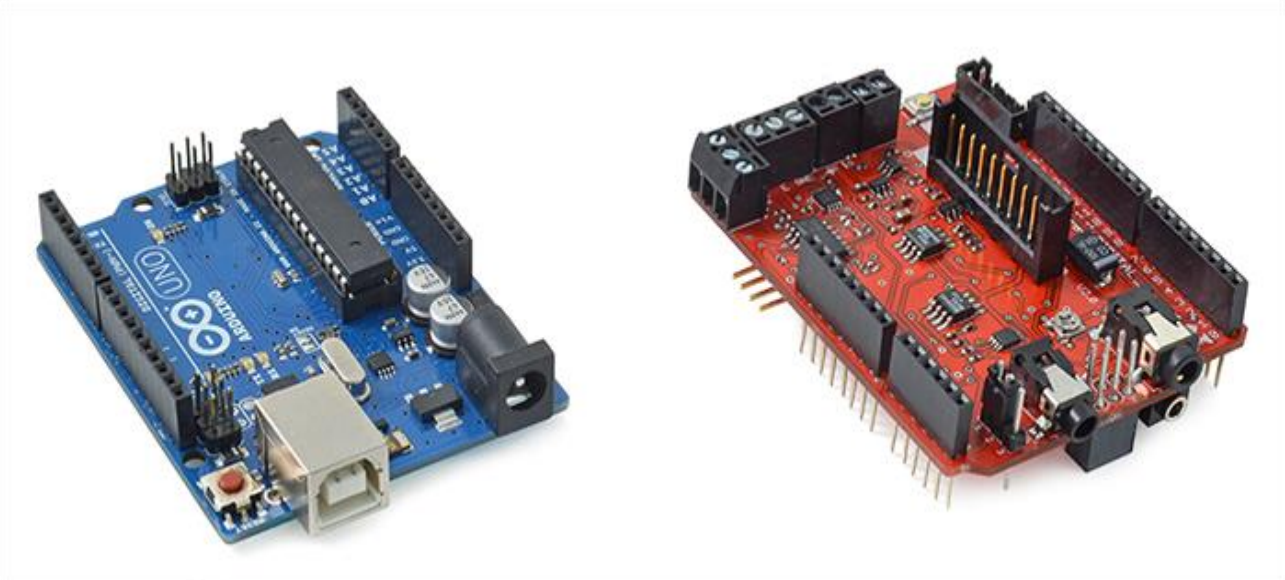

Figure 3.1 Arduino and E-Health PCB

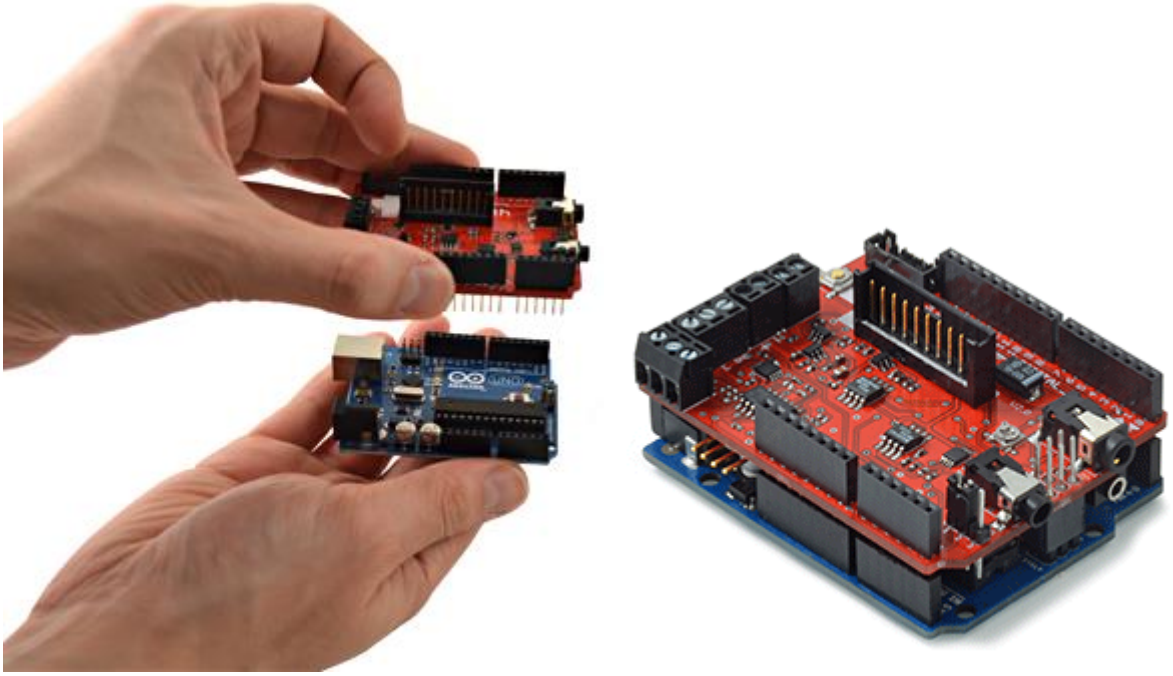

Figure 3.2 Connection of E-Health over Arduino

4. The library

The e-health Sensor Platform counts with a C++ library that lets you read easily all the sensors and send the information by using any of the available radio interfaces. This library offers an simple-to-use open source system.

We use the ArduPi libraries which allows developers to use the same code.

Library Codes

```
//Include eHealth library (it includes arduPi)
#include "eHealth.h"

int main (){
        setup();
        while(1){
                loop();
        }
        return (0);
}
```

Compilation of the program is done in two ways:

- g++ -c arduPi.cpp -o arduPi.o
- g++ -c eHealth.cpp -o eHealth.o
- g++ -lpthread -lrt user-e-health-app.cpp arduPi.o eHealth.o -o user-e-health-app
- Compiling everything in one step:

  g++ -lpthread -lrt user-e-health-app.cpp arduPi.cpp eHealth.cpp -o user-e-health-app

5. Sensor Platform

5.1 Pulse and Oxygen in Blood (SPO2)

SPO2 sensor features

Pulse oximetry a noninvasive method of indicating the arterial oxygen saturation of functional hemoglobin.
Oxygen saturation is defined as the measurement of the amount of oxygen dissolved in blood, based on the detection of Hemoglobin and Deoxyhemoglobin. Two different light wavelengths are used to measure the actual difference in the absorption spectra of HbO2 and Hb. The bloodstream is affected by the concentration of HbO2 and Hb, and their absorption coefficients are measured using two wavelengths 660 nm (red light spectra) and 940 nm (infrared light spectra). Deoxygenated and oxygenated hemoglobin absorb different wavelengths.

Figure 5.1 SPO2 Sensor

Deoxygenated hemoglobin (Hb) has a higher absorption at 660 nm and oxygenated hemoglobin (HbO2) has a higher absorption at 940 nm . Then a photo-detector perceives the non-absorbed light from the LEDs to calculate the arterial oxygen saturation.

A pulse oximeter sensor is useful in any setting where a patient's oxygenation is unstable, including intensive care, operating, recovery, emergency and hospital ward settings, pilots in unpressurized aircraft, for assessment of any patient's oxygenation, and determining the effectiveness of or need for supplemental oxygen.

Acceptable normal ranges for patients are from 95 to 99 percent, those with a hypoxic drive problem would expect values to be between 88 to 94 percent, values of 100 percent can indicate carbon monoxide poisoning.

The sensor needs to be connected to the Arduino, and don't use external/internal battery.

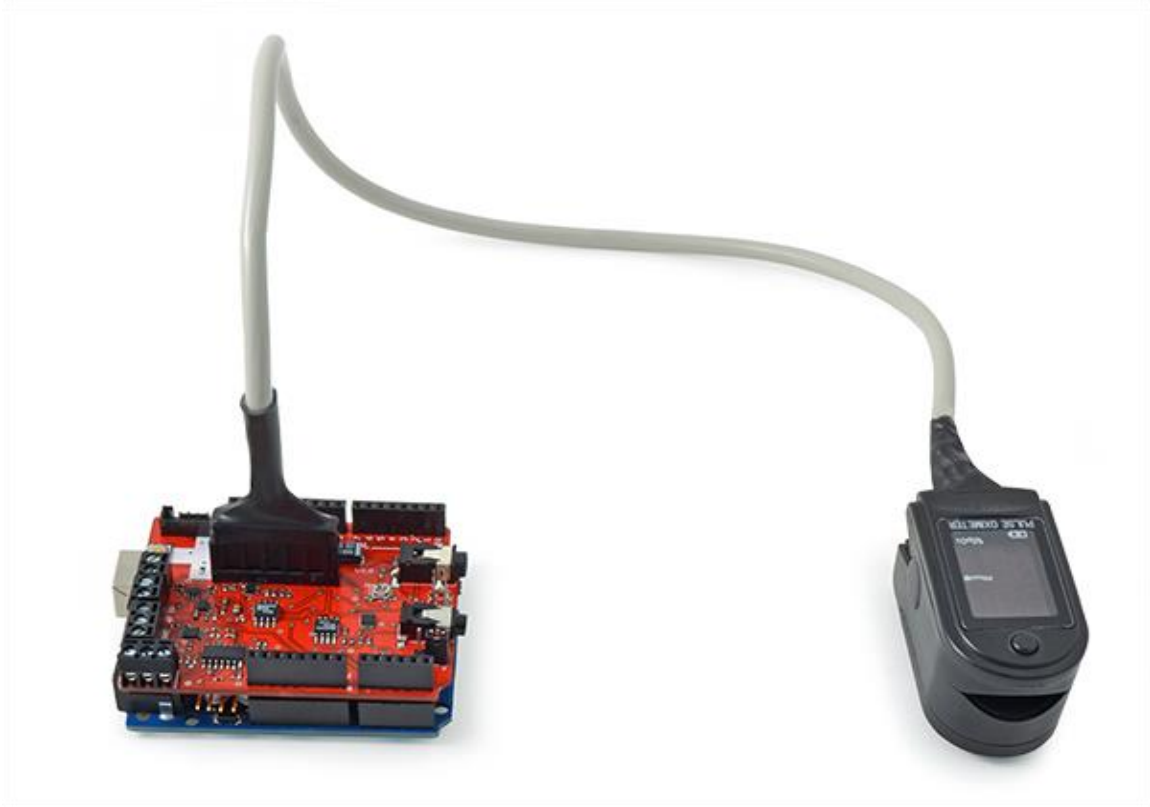

Figure 5.2 SPO2 Sensor Connection to E-Health

Figure 5.3 SPO2 Sensor Connected to a Patient

Initializing

This sensor use interruptions and it is necessary to include a special library when using it.

```
#include < PinChangeInt.h >
```

After this include, interruptions should be attached in the code to get data from th sensor. The sensor will interrupt the process to refresh the data stored in private variables.

```
PCintPort::attachInterrupt(6, readPulsioximeter, RISING);
```

The digital pin 6 of Arduino is the pin where sensor send the interruption and the function readpulsioximeter will be executed.

```
void readPulsioximeter(){
        cont ++;
        if (cont == 50) { //Get only one 50 measures to reduce the latency
                eHealth.readPulsioximeter();
                cont = 0;
        }
}
```

Before start using the SP02 sensor, it must be initialized. Use the next function in setup to configure some basic parameters and to start the communication between the Arduino/RaspberryPi and sensor.

Reading the sensor

For reading the current value of the sensor, we use the function:

```
{
        eHealth.readPulsioximeter();
}
```

This function will store the values of the sensor in private variables.

Getting data

To view data we can get the values of the sensor stored in private variable by using the function:

```
{
        int SPO2 = eHealth.getOxygenSaturation()
        int BPM = eHealth.getBPM()
}
```

Full Code

```
#include <PinChangeInt.h>
#include <eHealth.h>

int cont = 0;

void setup() {
  Serial.begin(115200);
  eHealth.initPulsioximeter();

  //Attach the inttruptions for using the pulsioximeter.
  PCintPort::attachInterrupt(6, readPulsioximeter, RISING);
}

void loop() {

  Serial.print("PRbpm : ");
  Serial.print(eHealth.getBPM());

  Serial.print("    %SPo2 : ");
  Serial.print(eHealth.getOxygenSaturation());

  Serial.print("\n");
  Serial.println("==============================");
  delay(500);
}


//Include always this code when using the pulsioximeter sensor
//=========================================================
================
void readPulsioximeter(){

  cont ++;

  if (cont == 50) { //Get only of one 50 measures to reduce the latency
    eHealth.readPulsioximeter();
    cont = 0;
  }
}
```

5.2 Electrocardiogram (ECG)

ECG sensor features

The electrocardiogram (ECG or EKG) is a diagnostic tool that is routinely used to assess the electrical and muscular functions of the hear.

The Electocardiogram Sensor(ECG) has grown to be one of the most commonly used medical tests in modern medicine. Its utility in the diagnosis of a myriad of cardiac pathologies ranging from myocardial ischemia and infarction to syncope and palpitations has been invaluable to clinicians for decades.

The accuracy of the ECG depends on the condition being tested. A heart problem may not always show up on the ECG. Some heart conditions never produce any specific ECG changes. ECG leads are attached to the body while the patient lies flat on a bed or table.

What is measured or can be detected on the ECG (EKG)?

The orientation of the heart (how it is placed) in the chest cavity.
Evidence of increased thickness (hypertrophy) of the heart muscle.
Evidence of damage to the various parts of the heart muscle.
Evidence of acutely impaired blood flow to the heart muscle.
Patterns of abnormal electric activity that may predispose the patient to abnormal cardiac rhythm disturbances.
The underlying rate and rhythm mechanism of the heart.

Schematic representation of normal ECG

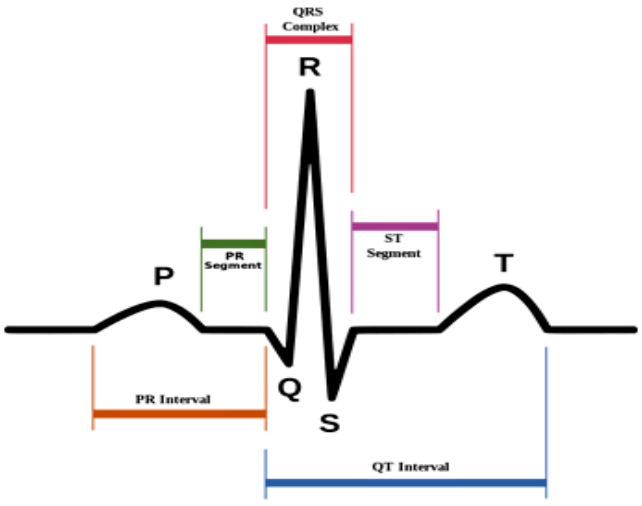

Figure 5.4 Schematic Representation of ECG

Connecting the sensor

Figure 5.5 ECG Sensor

Figure 5.6 ECG Sensor Connection to E-Health

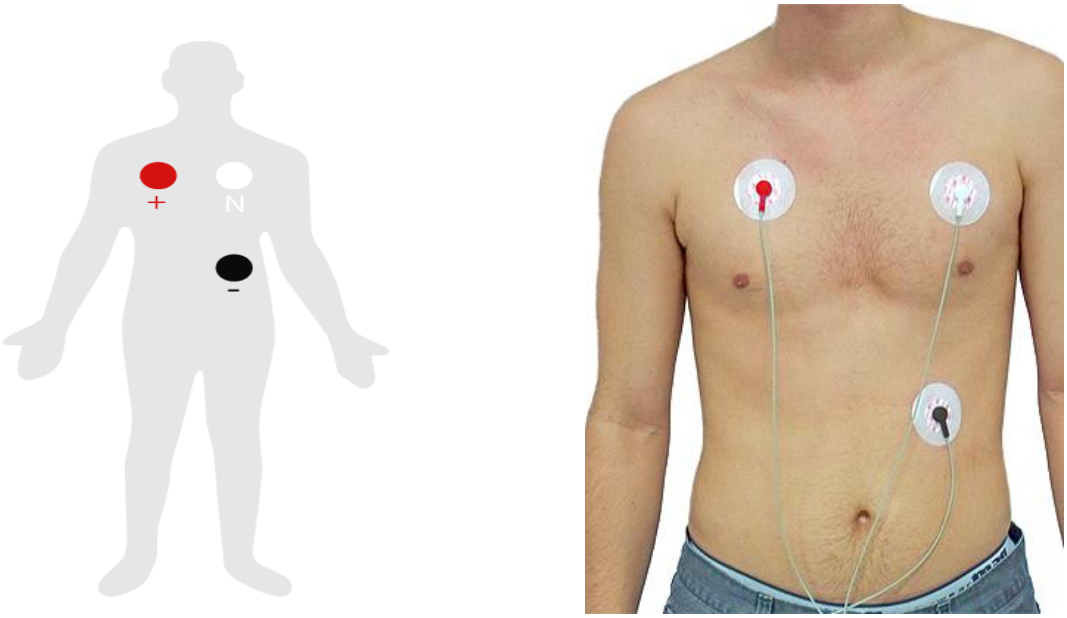

Figure 5.7 ECG Sensor Connected to a Patient

Getting data

This ECG returns an analogic value in volts (0 – 5) to represent the ECG wave form.

```
{
        float ECGvolt = eHealth.getECG();
}
```

Full Code

```
#include <eHealth.h>

// The setup routine runs once when you press reset:
void setup() {
  Serial.begin(115200);
}

// The loop routine runs over and over again forever:
void loop() {

  float ECG = eHealth.getECG();

  Serial.print("ECG value :  ");
  Serial.print(ECG, 2);
  Serial.print(" V");
  Serial.println("");

  delay(1);         // wait for a millisecond
}
```

5.3 Airflow: Breathing

Airflow sensor features

Anormal respiratory rates and changes in respiratory rate are a broad indicator of major physiological instability, and in many cases, respiratory rate is one of the earliest indicators of this instability. Therefore, it is critical to monitor respiratory rate as an indicator of patient status. AirFlow sensor can provide an early warning of hypoxemia and apnea.

The nasal/mouth airflow sensor is a device used to measure the breathing rate in a patient in need of respiratory help or person. This device consists of a flexible thread which fits behind the ears, and a set of two prongs which are placed in the nostrils. Breathing is measured by these prongs.

The specifically designed cannula/holder allows the thermocouple sensor to be placed in the optimal position to accurately sense the oral/nasal thermal airflow changes as well as the nasal temperature air. Comfortable adjustable and easy to install.

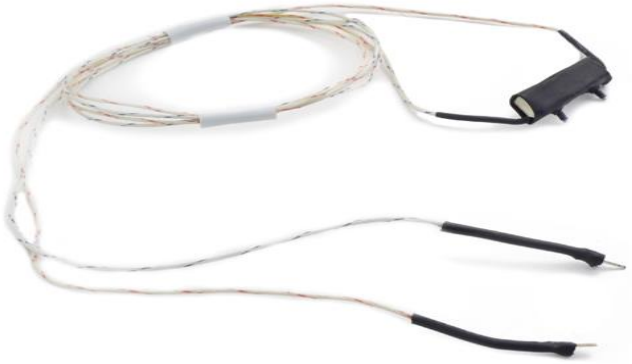

Figure 5.8 Airflow Sensor

Single channel oral or nasal reusable Airflow Sensor. Stay-put prongs position sensor precisely in airflow path.
A normal adult human has a respiratory rate of 15–30 breaths per minute.

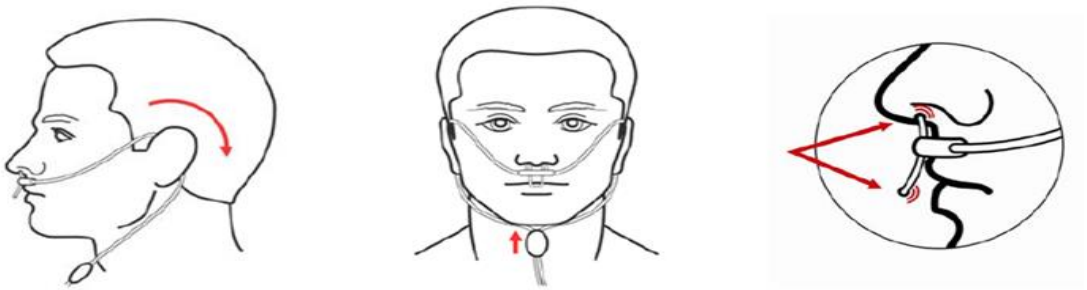

Figure 5.9 Airflow Sensor Connected to a Patient

Connecting the sensor

Figure 5.10 Airflow Sensor Connection Ends

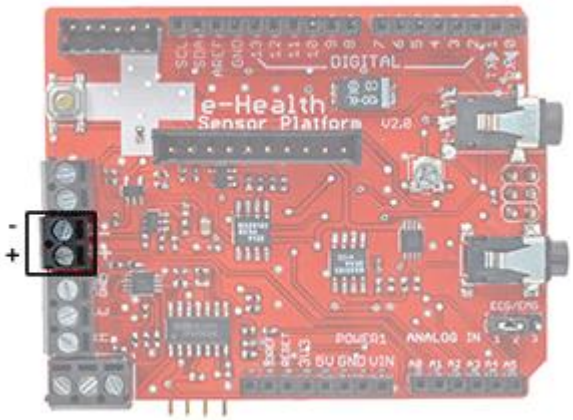

Figure 5.11 Airflow Sensor Connection Points

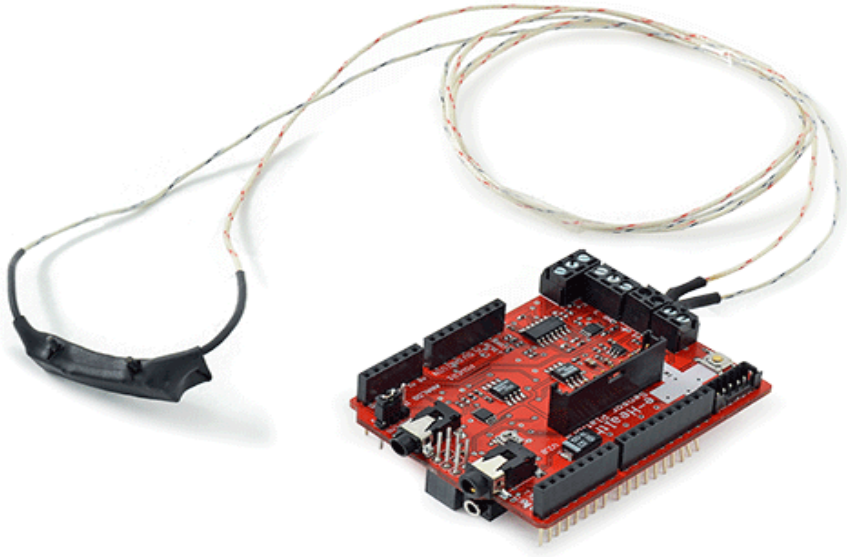

Figure 5.12 Airflow Sensor Connection Points

Getting data

The air flow sensor is connected to the Arduino/RasberryPi by an analog input and returns a value from 0 to 1024. With this function this value can be gathered directly and print a wave form in the serial monitor.

```
{
        int airFlow = eHealth.getAirFlow();
        eHealth.airFlowWave(air);
}
```

Full Code

```
#include <eHealth.h>

// The setup routine runs once when you press reset:
void setup() {
  Serial.begin(115200);
}

// the loop routine runs over and over again forever:
void loop() {

  int air = eHealth.getAirFlow();
  eHealth.airFlowWave(air);
}
```

5.4 Body Temperature

Temperature sensor features

Body temperature depends upon the place in the body at which the measurement is made, and the time of day and level of activity of the person. Different parts of the body have different temperatures.

Figure 5.13 Body Temperature Sensor

The commonly accepted average core body temperature (taken internally) is 37.0°C (98.6°F). In healthy adults, body temperature fluctuates about 0.5°C (0.9°F) throughout the day, with lower temperatures in the morning and higher temperatures in the late afternoon and evening, as the body's needs and activities change.

It is of great medical importance to measure body temperature. The reason is that a number of diseases are accompanied by characteristic changes in body temperature. Likewise, the course of certain diseases can be monitored by measuring body temperature, and the efficiency of a treatment initiated can be evaluated by the physician.

Hypothermia	<35.0 °C (95.0 °F)
Normal	36.5–37.5 °C (97.7–99.5 °F)
Fever or Hyperthermia	>37.5–38.3 °C (99.5–100.9 °F)
Hyperpyrexia	>40.0–41.5 °C (104–106.7 °F)

Sensor Calibration

The precision of the Body Temperature Sensor is enough in most applications. But you can improve this precision by a calibration process.

When using temperature sensor, you are actually measuring a voltage, and relating that to what the operating temperature of the sensor must be. If you can avoid errors in the voltage measurements, and represent the relationship between voltage and temperature more accurately, you can get better temperature readings.

Connecting the sensor

Figure 5.14 Body Temperature on a Patient

Geting data

The corporal temperature can be taken by a simple function. This function return a float with the last value of temperature measured by the Arduino.

```
{
        float temperature = eHealth.getTemperature();
}
```

Full Code

```
#include <eHealth.h>

// the setup routine runs once when you press reset:
void setup() {
  Serial.begin(115200);
}

// the loop routine runs over and over again forever:
void loop() {
  float temperature = eHealth.getTemperature();

  Serial.print("Temperature (ºC): ");
  Serial.print(temperature, 2);
  Serial.println("");

  delay(1000);    // wait for a second
}
```

5.5 Blood Pressure

Blood pressure sensor features

Blood pressure is the pressure of the blood in the arteries as it is pumped around the body by the heart. When your heart beats, it contracts and pushes blood through the arteries to the rest of your body. This force creates pressure on the arteries. Blood pressure is recorded as two numbers—the systolic pressure (as the heart beats) over the diastolic pressure (as the heart relaxes between beats).

Figure 5.15 Blood Pressure Sensor

Monitoring blood pressure at home is important for many people, especially if you have high blood pressure. Blood pressure does not stay the same all the time. It changes to meet your body's needs. It is affected by various factors including body position, breathing or emotional state, exercise and sleep. It is best to measure blood pressure when you are relaxed and sitting or lying down.

Classification of blood pressure for adults (18 years and older)

	Systolic (mm Hg)	**Diastolic (mm Hg)**
Hypotension	< 90	< 60
Desired	90–119	60–79
Prehypertension	120–139	80–89
Stage 1 Hypertension	140–159	90–99
Stage 2 Hypertension	160–179	100–109
Hypertensive Crisis	≥ 180	≥ 110

High blood pressure (hypertension) can lead to serious problems like heart attack, stroke or kidney disease. High blood pressure usually does not have any symptoms, so you need to have your blood pressure checked regularly.

Special Features:

- Automatic measurement of systolic, diastolic and pulse with time & date
- Large LCD screen with LED backlight
- TOUCH PAD KEY
- 80 measurement results with time & date stored in the device

Key Specifications:

- Measurement method: Oscillometric system
- Display type：Digital LCD 97mm [L] x 87mm [W]
- Measuring range: Pressure 0-300 mmHg
- Pulse 30~200 p/min
- Measuring accuracy: Pressure≤±3 mmHg
- Pulse≤5%
- Operating environment: Temperature 10-40℃
- Relative humidity≤80%
- Power source: 4 x AA batteries
- Dimension: 150mm [L] x 110mm [W] x 65mm [H]
- Weight: About 370g
- Cuff size: 520mm [L] x 135mm [W]
- Cuff range: 220mm [L] - 320mm [W]

Connecting the sensor

Before start using the sphygmomanometer we need one measure at least in the memory of the blood pressure sensor. After that we can get all the information contained in the device.

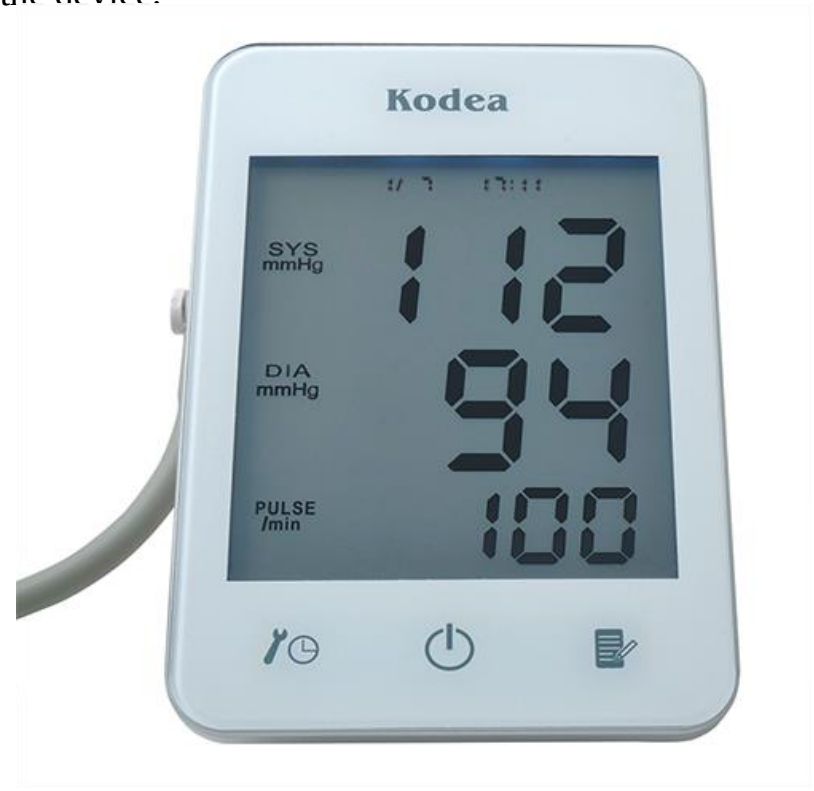

Figure 5.16 Blood Pressure Sensor Screen

Figure 5.17 Blood Pressure Sensor Connected to a Patient

Getting blood pressure data

Some parameters must be initialized to start using blood pressure sensor (sphygmomanometer). The next function initializes some variables and reads the sphygmomanometer data.

```
{
	eHealth.readBloodPressureSensor();
	Serial.begin(115200);
}
```

The amount of data read is accessible with another public function.

```
{
	uint8_t numberOfData = eHealth.getBloodPressureLength();
	Serial.print(F("Number of measures : "));
	Serial.println(numberOfData, DEC);
	delay(100);
}
```

The next functions return the values of systolic and diastolic pressure measured by the sphygmomanometer and stored in private variables of the e-Health class.

```
{
	int systolic = eHealth.getSystolicPressure(1);
	int diastolic =eHealth.getDiastolicPressure(1);
}
```

The number given to the function in the variable i is the number of the measurement we want to obtain.

Full Code

```
#include <eHealth.h>

void setup() {

  eHealth.readBloodPressureSensor();
  Serial.begin(115200);
  delay(100);
}

void loop() {

  uint8_t numberOfData = eHealth.getBloodPressureLength();
  Serial.print(F("Number of measures : "));
  Serial.println(numberOfData, DEC);
  delay(100);


  for (int i = 0; i<numberOfData; i++) {

Serial.println(F("========================================"));

    Serial.print(F("Measure number "));
    Serial.println(i + 1);

    Serial.print(F("Date -> "));
    Serial.print(eHealth.bloodPressureDataVector[i].day);
    Serial.print(F(" of "));

Serial.print(eHealth.numberToMonth(eHealth.bloodPressureDataVector[i].mont
h));
    Serial.print(F(" of "));
    Serial.print(2000 + eHealth.bloodPressureDataVector[i].year);
    Serial.print(F(" at "));

    if (eHealth.bloodPressureDataVector[i].hour < 10) {
      Serial.print(0); // Only for best representation.
    }

    Serial.print(eHealth.bloodPressureDataVector[i].hour);
    Serial.print(F(":"));
```

```
    if (eHealth.bloodPressureDataVector[i].minutes < 10) {
      Serial.print(0);// Only for best representation.
    }
    Serial.println(eHealth.bloodPressureDataVector[i].minutes);

    Serial.print(F("Systolic value : "));
    Serial.print(30+eHealth.bloodPressureDataVector[i].systolic);
    Serial.println(F(" mmHg"));

    Serial.print(F("Diastolic value : "));
    Serial.print(eHealth.bloodPressureDataVector[i].diastolic);
    Serial.println(F(" mmHg"));

    Serial.print(F("Pulse value : "));
    Serial.print(eHealth.bloodPressureDataVector[i].pulse);
    Serial.println(F(" bpm"));
  }

  delay(20000);
}
```

5.6 Patient Position and Falls

Position sensor features

The Patient Position Sensor(Accelerometer) monitors five different patient positions (standing/sitting, supine, prone, left and right.)

In many cases, it is necessary to monitor the body positions and movements made because of their relationships to particular diseases (i.e., sleep apnea and restless legs syndrome). Analyzing movements during sleep also helps in determining sleep quality and irregular sleeping patterns. The body position sensor could help also to detect fainting or falling of elderly people or persons with disabilities.

Figure 5.18 Patient Position and Falls Sensor

eHealth Body Position Sensor uses a triple axis accelerometer to obtain the patient's position.

Features:

- 1.95 V to 3.6 V supply voltage
- 1.6 V to 3.6 V interface voltage
- ±2g/±4g/±8g dynamically selectable full-scale

This accelerometer is packed with embedded functions with flexible user programmable options, configurable to two interrupt pins.The accelerometer has user selectable full scales of ±2g/±4g/±8g with high pass filtered data as well as non filtered data available real-time.

Body positions:

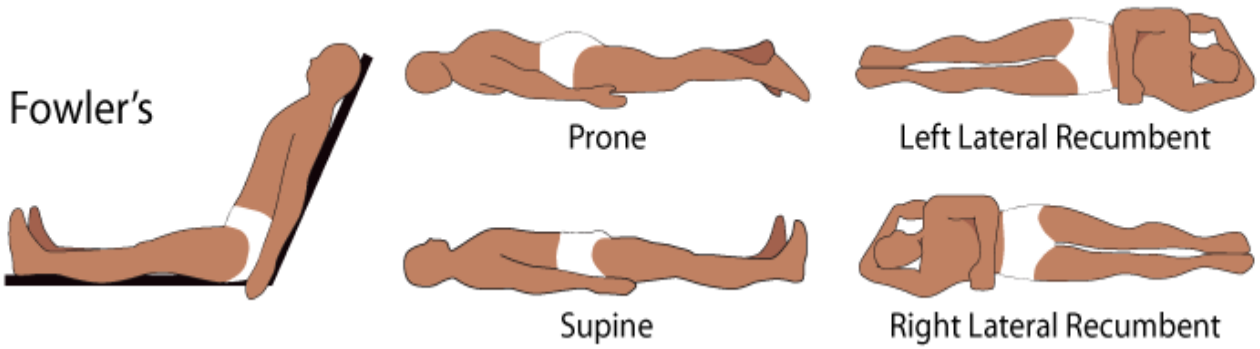

Figure 5.19 Patient Body Position

Connecting the sensor

Figure 5.20 Patient Position and Falls Connection to E-Health

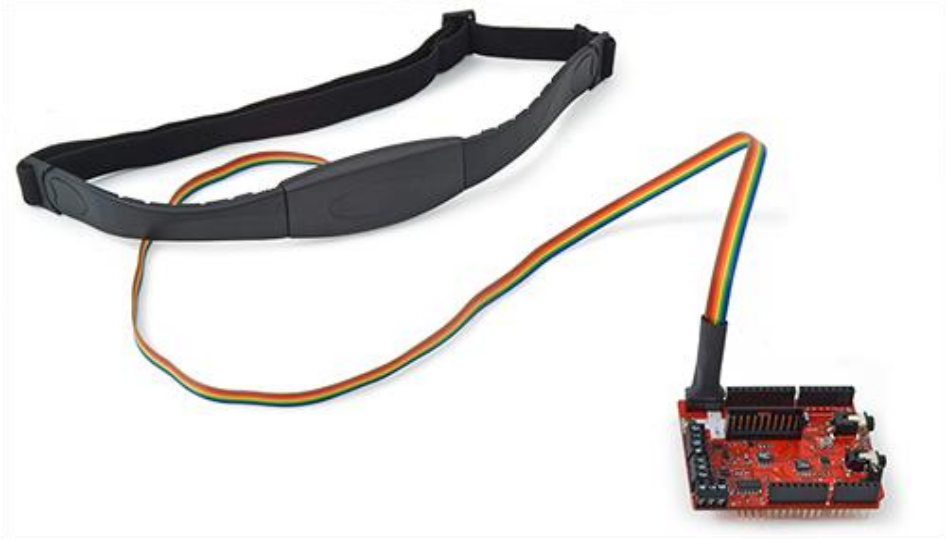

Figure 5.21 Patient Position and Falls Connection to E-Health

Initializing the sensor

Before start using the sensor, we have to initialize some parameters. We use the following function to configure the sensor.

```
{
        eHealth.initPositionSensor();
}
```

Getting data

The next functions return the a value that represent the body position stored in private variables of the e-Health class.

```
{
        uint8_t position = eHealth.getBodyPosition();
}
```

The position of the pacient

1 == Supine position.
2 == Left lateral decubitus.
3 == Rigth lateral decubitus.
4 == Prone position.
5 == Stand or sit position.

Printing Data

For representing the data in an easy way in the Arduino serial monitor, e-health library, includes a printing function.

```
{
        Serial.print("Current position : ");
        uint8_t position = eHealth.getBodyPosition();
        eHealth.printPosition(position);
}
```

Full Code

```
#include <eHealth.h>

void setup() {

  Serial.begin(115200);
  eHealth.initPositionSensor();
}

void loop() {

  Serial.print("Current position : ");
  uint8_t position = eHealth.getBodyPosition();
  eHealth.printPosition(position);

  Serial.print("\n");
  delay(1000); // wait for a second.
}
```

5.7 Galvanic Skin Response (GSR)

GSR sensor features

Skin conductance, also known as galvanic skin response (GSR) is a method of measuring the electrical conductance of the skin, which varies with its moisture level. This is of interest because the sweat glands are controlled by the sympathetic nervous system, so moments of strong emotion, change the electrical resistance of the skin. Skin conductance is used as an indication of psychological or physiological arousal, The Galvanic Skin Response Sensor (GSR – Sweating) measures the electrical conductance between 2 points, and is essentially a type of ohmmeter.

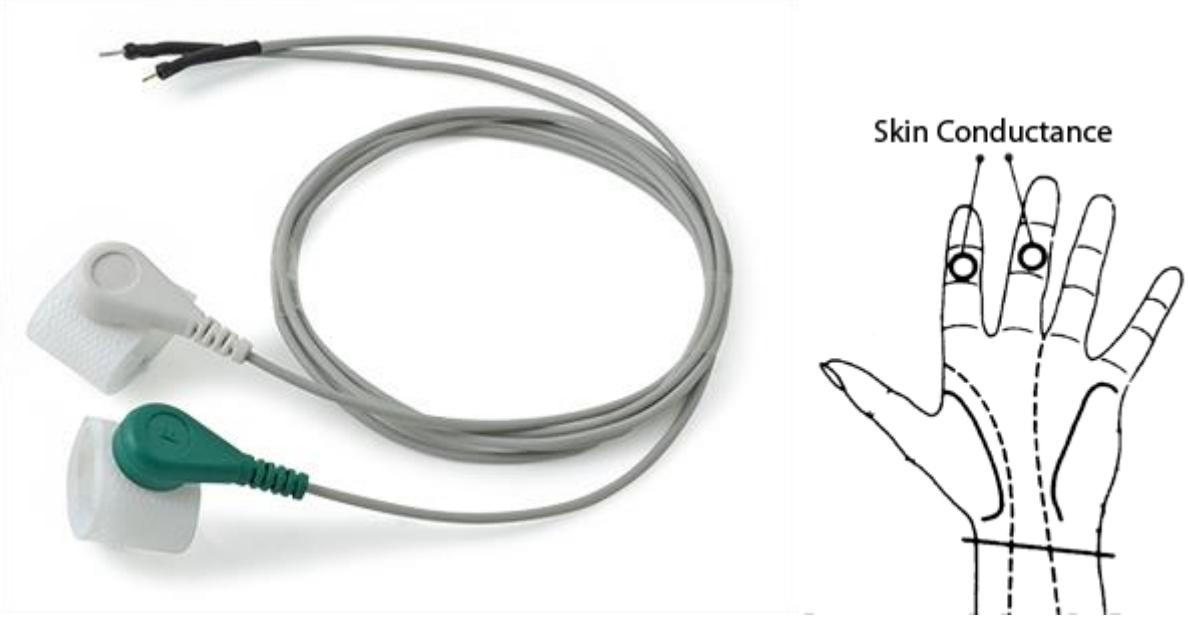

Figure 5.22 Galvanic Skin Response Sensor

In skin conductance response method, conductivity of skin is measured at fingers of the palm. The principle or theory behind functioning of galvanic response sensor is to measure electrical skin resistance based on sweat produced by the body. When high level of sweating takes place, the electrical skin resistance drops down. A dryer skin records much higher resistance. The skin conductance response sensor measures the psycho galvanic reflex of the body. Emotions such as excitement, stress, shock, etc. can result in the fluctuation of skin conductivity. Skin conductance measurement is one component of polygraph devices and is used in scientific research of emotional or physiological arousal.

Connecting the sensor

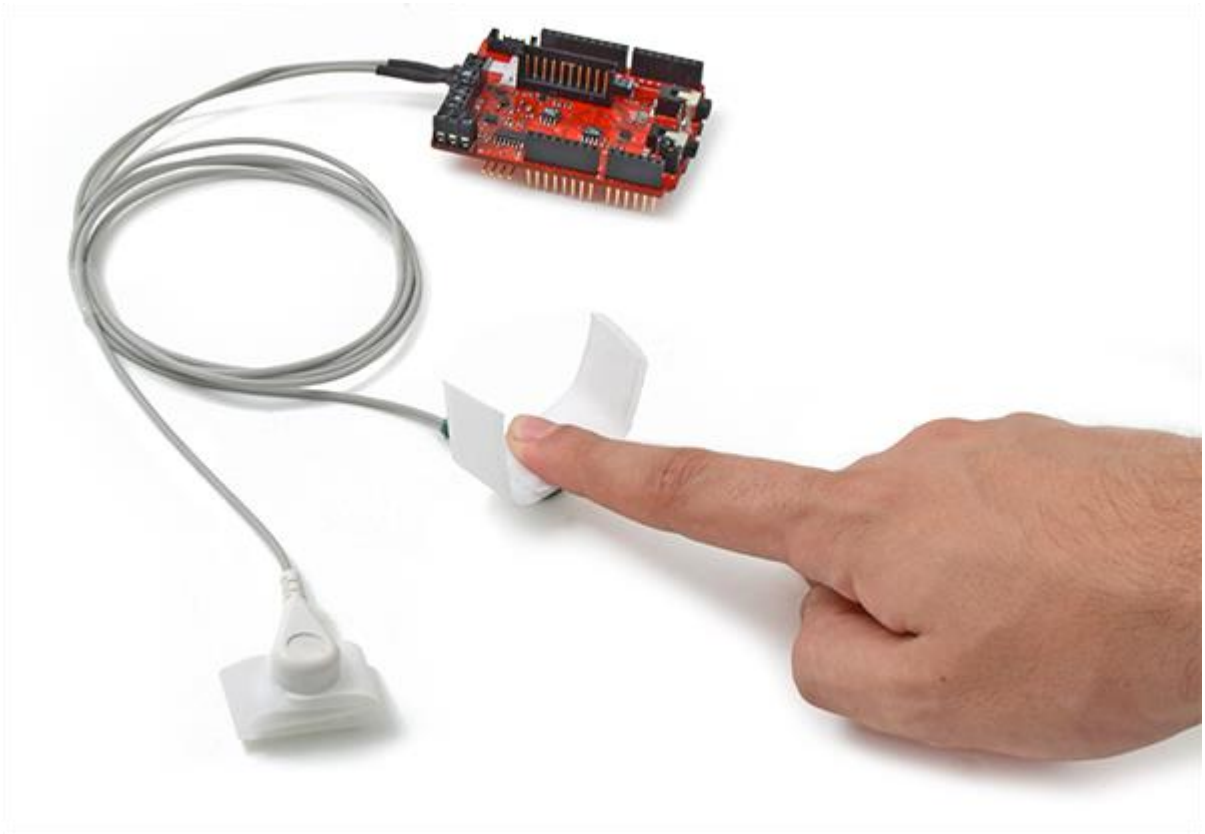

Figure 5.23 Galvanic Skin Response Sensor Connected to a Patient

The galvanic skin sensor has two contacts and it works like a ohmmeter measuring the resistance of the materials.

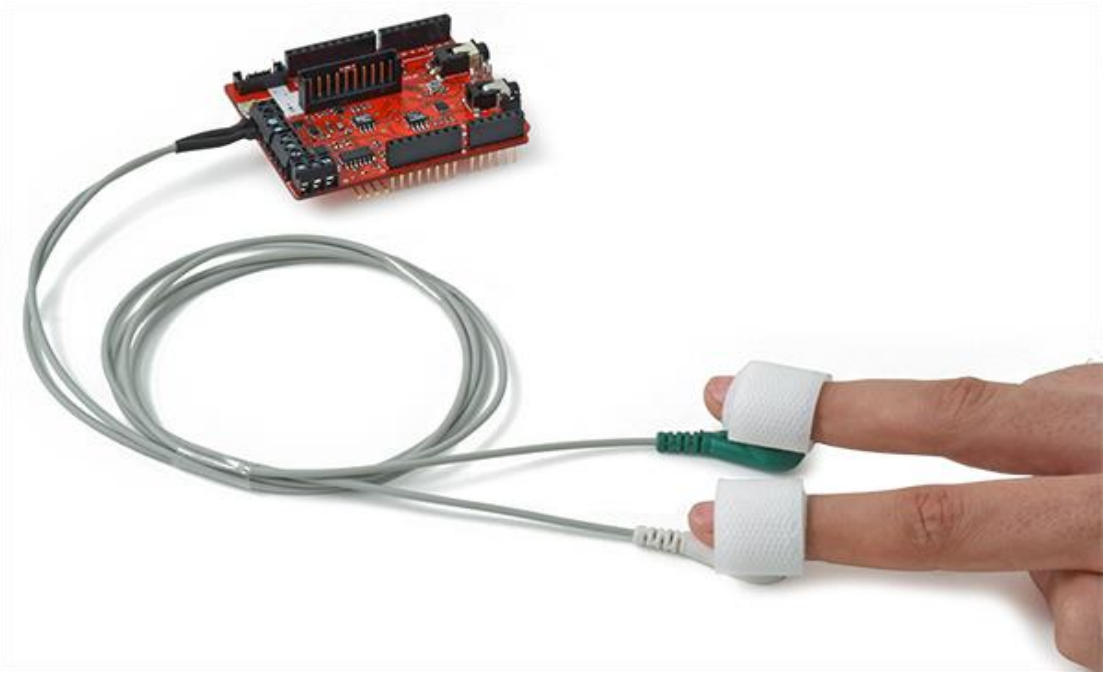

Figure 5.24 Galvanic Skin Response Sensor Connected to a Patient

Getting data

With this simple functions we can read the value of the sensor. The library returns the value of the skin resistance and the skin conductance.

```
{
        float conductance = eHealth.getSkinConductance();
        float resistance = eHealth.getSkinResistance();
        float conductanceVol = eHealth.getSkinConductanceVoltage();
}
```

Full Code

```
#include <eHealth.h>

// the setup routine runs once when you press reset:
void setup() {
  Serial.begin(115200);
}

// the loop routine runs over and over again forever:
void loop() {

  float conductance = eHealth.getSkinConductance();
  float resistance = eHealth.getSkinResistance();
  float conductanceVol = eHealth.getSkinConductanceVoltage();

  Serial.print("Conductance : ");
  Serial.print(conductance, 2);
  Serial.println("");

  Serial.print("Resistance : ");
  Serial.print(resistance, 2);
  Serial.println("");

  Serial.print("Conductance Voltage : ");
  Serial.print(conductanceVol, 4);
  Serial.println("");

  Serial.print("\n");

  // wait for a second
  delay(1000);
}
```

5.8 Glucometer

Glucometer sensor features

Glucometer is a medical device for determining the approximate concentration of glucose in the blood. A small drop of blood, obtained by pricking the skin with a lancet, is placed on a disposable test strip that the meter reads and uses to calculate the blood glucose level. The meter then displays the level in mg/dl or mmol/l.

Figure 5.25 Glucometer Sensor

Despite widely variable intervals between meals or the occasional consumption of meals with a substantial carbohydrate load, human blood glucose levels tend to remain within the normal range. However, shortly after eating, the blood glucose level may rise, in non-diabetics, temporarily up to 7.8 mmol/L (140 mg/dL) or a bit more.

Glucometer configuration

For configuring the date time, open the batery compartment and press the black button. It's simple to set the parameters of our glucometer.

Did you see a small button near the battery (backside)? You can set the time and mg/dl by it. When glucometer is off, plz press once short time of the backside button to open it, then you can see the screen of year month date and time, short press the front button is to increase the data (long pressing will increase the data fast), short pree the backside botton is to switch the year to month/date or time or glucose units. To save the settings, plz short press the backside button several

times until you see the big OFF on screen, then plz do nothing just wait it auto off.

To change the glucose units, when you see the time screen, just keeping short presses for several times untill you see only mmol/l is flashing, then press the front big button in long time (about 3 secs), you will see the mmol/l becomes to mg/dl. Now plz press the back small button twice short times to save it. You can see big OFF on screen, then plz do nothing just wait it auto off. Now you finished the settings of mg/dl. Setting other parameters is similar.

Connecting the sensor

Figure 5.26 Glucometer Sensor

Figure 5.27 Glucometer Sensor Application on a Patient

Figure 5.28 Glucometer Sensor Application on a Patient

Figure 5.29 Glucometer Sensor Connection to E-Health

Getting data

With a simple function we can read all the measures stored in the glucometer and show them in the terminal. The function must be used before the intilizazion of the serial monitor.

```
{
        eHealth.readGlucometer();
        Serial.begin(115200);
}
```

The amount of data read is accessible with another public function.

```
{
        uint8_t numberOfData eHealthClass.getGlucometerLength()
}
```

The maximum number of measures is 32. The vector where data is a public variable of the e-Health class.

```
{
  Serial.print(F("Glucose value : "));
  Serial.print(eHealth.glucoseDataVector[i].glucose);
  Serial.println(F(" mg/dL"));
}
```

Full Code

```
#include <eHealth.h>

void setup() {

  eHealth.readGlucometer();
  Serial.begin(115200);
  delay(100);
}

void loop() {

  uint8_t numberOfData = eHealth.getGlucometerLength();
  Serial.print(F("Number of measures : "));
  Serial.println(numberOfData, DEC);
  delay(100);


  for (int i = 0; i<numberOfData; i++) {
    // The protocol sends data in this order

Serial.println(F("========================================"));

    Serial.print(F("Measure number "));
    Serial.println(i + 1);

    Serial.print(F("Date -> "));
    Serial.print(eHealth.glucoseDataVector[i].day);
    Serial.print(F(" of "));
    Serial.print(eHealth.numberToMonth(eHealth.glucoseDataVector[i].month));
    Serial.print(F(" of "));
    Serial.print(2000 + eHealth.glucoseDataVector[i].year);
    Serial.print(F(" at "));

    if (eHealth.glucoseDataVector[i].hour < 10) {
      Serial.print(0); // Only for best representation.
    }

    Serial.print(eHealth.glucoseDataVector[i].hour);
    Serial.print(F(":"));

    if (eHealth.glucoseDataVector[i].minutes < 10) {
      Serial.print(0);// Only for best representation.
```

```
    }
    Serial.print(eHealth.glucoseDataVector[i].minutes);

    if (eHealth.glucoseDataVector[i].meridian == 0xBB)
      Serial.println(F(" pm"));
    else if (eHealth.glucoseDataVector[i].meridian == 0xAA)
      Serial.println(F(" am"));

    Serial.print(F("Glucose value : "));
    Serial.print(eHealth.glucoseDataVector[i].glucose);
    Serial.println(F(" mg/dL"));
  }

  delay(20000);
}
```

5.9 Electromyogram (EMG)

EMG sensor features

An electromyography (EMG) measures the electrical activity of muscles at rest and during contraction.

Electromyography (EMG) is a technique for evaluating and recording the electrical activity produced by skeletal muscles. EMG is performed using an instrument called an electromyograph, to produce a record called an electromyogram. An electromyograph detects the electrical potential generated by muscle cells when these cells are electrically or neurologically activated. The signals can be analyzed to detect medical abnormalities, activation level, recruitment order or to analyze the biomechanics of human or animal movement.

EMG signals are used in many clinical and biomedical applications. EMG is used as a diagnostics tool for identifying neuromuscular diseases, assessing low-back pain, kinesiology, and disorders of motor control. EMG signals are also used as a control signal for prosthetic devices such as prosthetic hands, arms, and lower limbs.

This sensor will measure the filtered and rectified electrical activity of a muscle, depending the amount of activity in the selected muscle.

Features:

- Adjustable gain
- Small Form Factor
- Full integrated

Figure 5.30 Electromyogram Sensor

Uses muscles to control any type of actuator (motors, servos, lights ...). Interacts with the environment with muscles.

Connecting the sensor

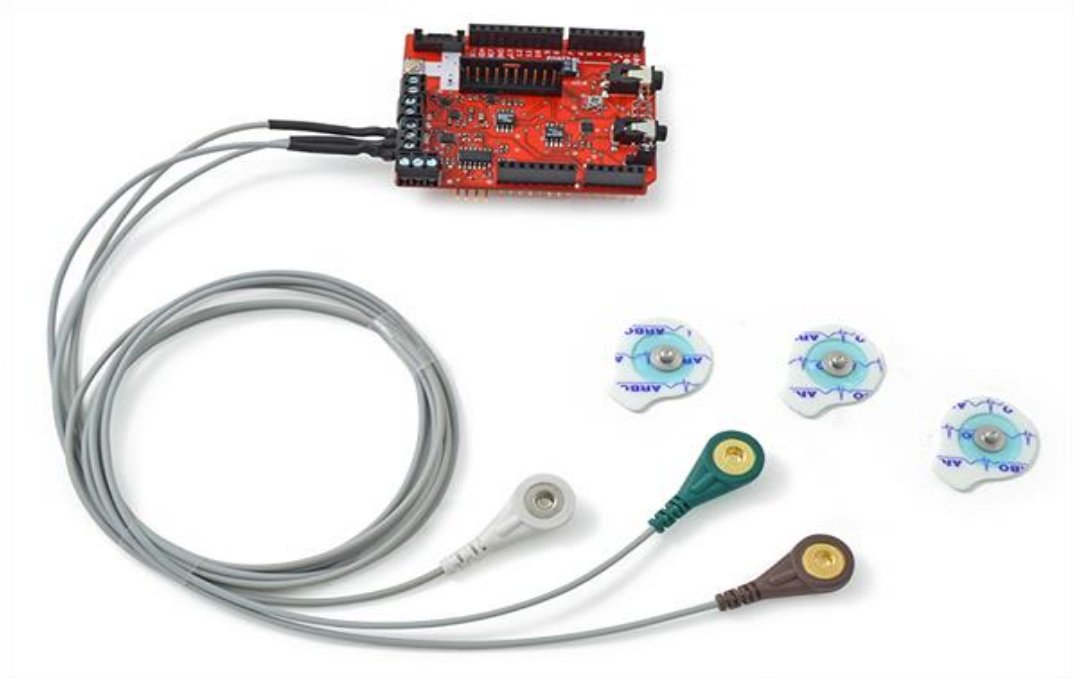

Figure 5.31 EMG Sensor Connection to E-Health

Figure 5.32 EMG Sensor Connected to a Patient

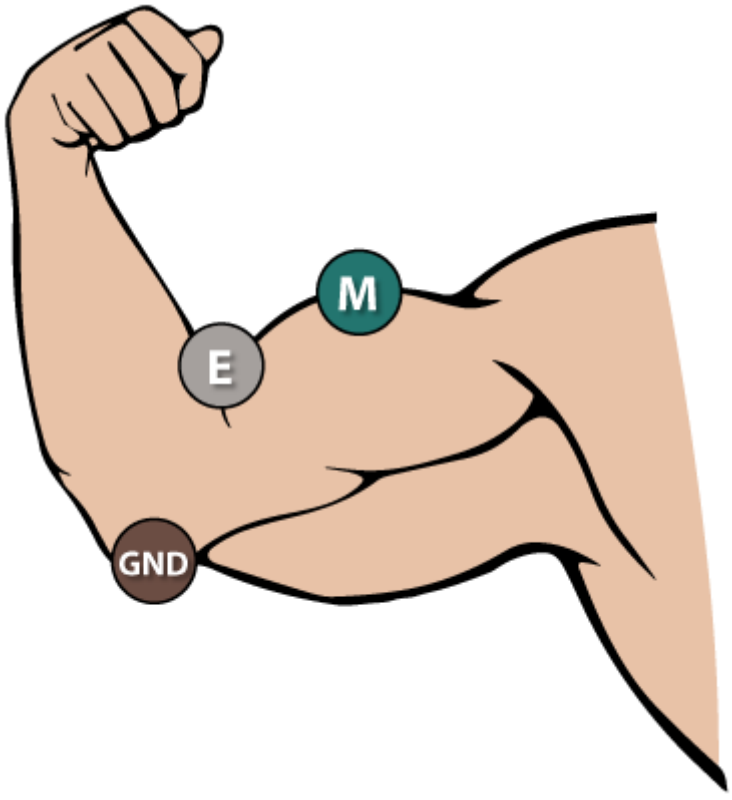

Figure 5.33 Representation of EMG Sensor Connection

Getting data

This EMG returns an analogic value in volts (read 0 – 1023 by the ADC) to represent the EMG wave form.

```
{
        float EMG = eHealth.getEMG();
}
```

Full Code

```
#include <eHealth.h>

// The setup routine runs once when you press reset:
void setup() {
  Serial.begin(115200);
}

// The loop routine runs over and over again forever:
void loop() {

  int EMG = eHealth.getEMG();

  Serial.print("EMG value :  ");
  Serial.print(EMG);
  Serial.println("");

  delay(100);      // wait for a millisecond
}
```

6. Visualizing the data

6.1 GLCD

The e-Health library includes functions to manage a graphic LCD for visualizing data. The Serial Graphic LCD backpack is soldered to the 128x64 pixel Graphic LCD and provides the user a simple serial interface.

Features

Connecting the GLCD

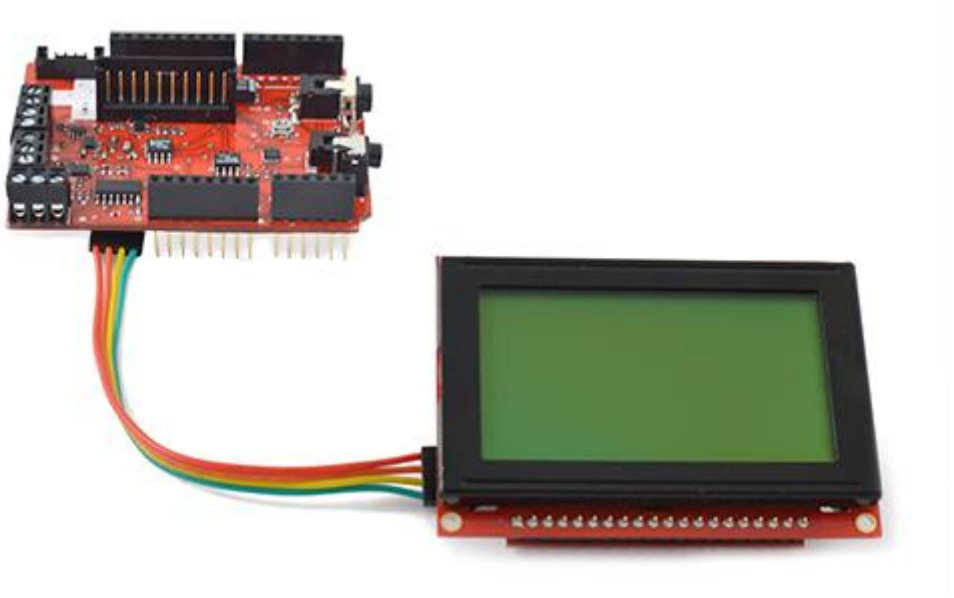

Figure 6.1 GLCD Connection to E-Health

The sensor measures are distributed in three screens that change by pressing the button.

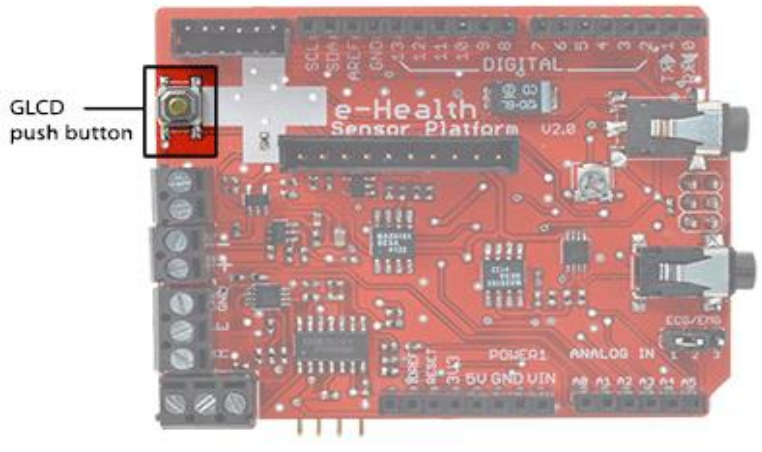

Figure 6.2 GLCD Push Button

In the first screen some values like temperature, pulse or oxygen can be seen.

Figure 6.3 Temperature Reading on GLCD

In the second screen the air flow wave is presented.

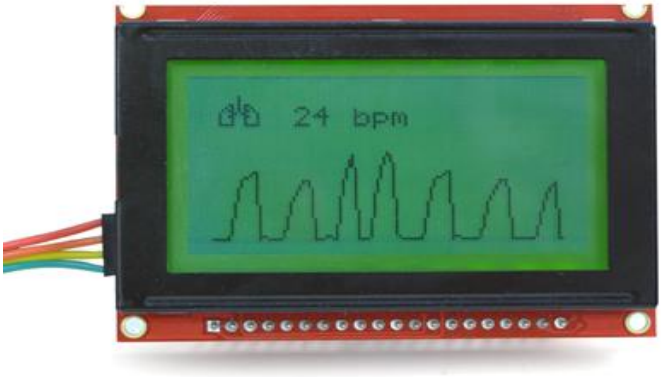

Figure 6.4 Airflow Reading on GLCD

When no respiration is detected, the screen will advise of a risk of apnea.

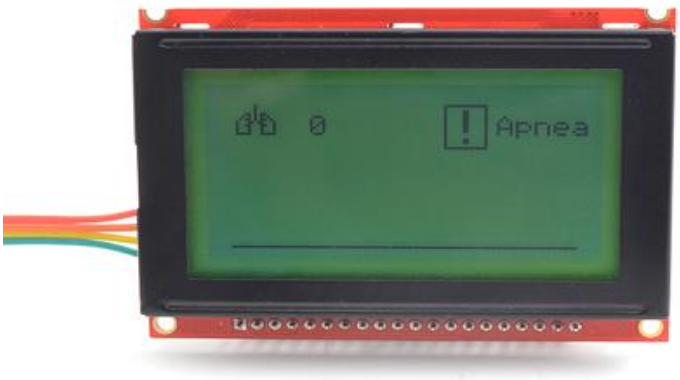

Figure 6.5 Apnea Detection on GLCD

The last screen draws the ECG wave.

Figure 6.6 ECG Reading on GLCD

After the third screen, pressing the button again returns to the first screen.

The push button

The eHealth board includes an integrated push button to change between the screen or it can be used like a general purpouse button for any application. The push button is routed to the digital pin 4.

GLCD Library

The eHealth library includes all the necessary functions to manage the LCD and show in real time the data sensor measures. In order to use this functions, before all, you should include the corresponding library.

```
#include < eHealthDisplay.h >
```

Library functions:

Initializing the LCD

Use the init function in the setup before start using the LCD.

```
{
        eHealthDisplay.init();
}
```

Values Screen

In this screen you can see some numerical parameters like pulse, oxygen , temperature, amd a representation of the current body position.

Initializing

This screen must be initialized with the next function:

```
{
        eHealthDisplay.initValuesScreen();
}
```

Getting data

To refresh the values in the LCD execute the next function.

```
{
        eHealthDisplay.printValuesScreen();
}
```

AirFlow screen

In this screen we could see the air flow (breathing) wave and the number of breathings per minute. This screen includes an apnea adviser when no breathing is detected.

Initializing

This screen must be initialized with the next function:

```
{
        eHealthDisplay.initAirFlowScreen();
}
```

Getting data

To refresh the values in the LCD execute the next function.

```
{
        eHealthDisplay.printAirFlowScreen();
}
```

ECGScreen

This screen draws the electrocardiogram wave and measures the heart beat rate.

Initializing

This screen must be initialized with the next function:

```
{
        eHealthDisplay.initECGScreen();
}
```

Getting data

To refresh the values in the LCD execute the next function.

```
{
        eHealthDisplay.printECGScreen();
}
```

Full Code

```
#include <PinChangeInt.h>
#include <eHealthDisplay.h>
#include <eHealth.h>

#define pushButton 4

uint8_t screenNumber = 1;
uint8_t buttonState;
uint8_t cont = 0;

void setup() {
  Serial.begin(115200);
  delay(100);

  //Configure and initializes the LCD.
  eHealthDisplay.init();
  delay(100);
}

void loop() {

  eHealthDisplay.initValuesScreen();
  delay(100);

  //Attach the inttruptions for using the pulsioximeter

  PCintPort::attachInterrupt(6, readPulsioximeter, RISING);

  while(screenNumber == 1) {
    //It prints data sensor measures in the LCD.
    buttonState = digitalRead(pushButton);
    delay(10);

    if(buttonState == 1) {
      screenNumber++;
    }

    eHealthDisplay.printValuesScreen();
  }

  PCintPort::detachInterrupt(6);
```

```
  eHealthDisplay.initAirFlowScreen();

  while( screenNumber == 2) {
    buttonState = digitalRead(pushButton);
    delay(10);

    if(buttonState==1){
      screenNumber++;
    }

    eHealthDisplay.printAirFlowScreen();
  }

  eHealthDisplay.initECGScreen();

  while( screenNumber == 3) {
    buttonState = digitalRead(pushButton);
    delay(10);

    if(buttonState==1){
      screenNumber++;
    }

    eHealthDisplay.printECGScreen();
  }

  screenNumber = 1;
  delay(10);
}

void readPulsioximeter() {

  cont ++;

  if (cont == 50) {
    //Get only one of 50 measures to reduce the latency
    eHealth.readPulsioximeter();
    cont = 0;
  }
}
```

6.2 KST: Real-time data viewing and plotting

KST is the fastest real-time large-dataset viewing and plotting tool available (you may be interested in some benchmarks) and has built-in data analysis functionality. It is very user-friendly and contains many powerful built-in features and is expandable with plugins and extensions.

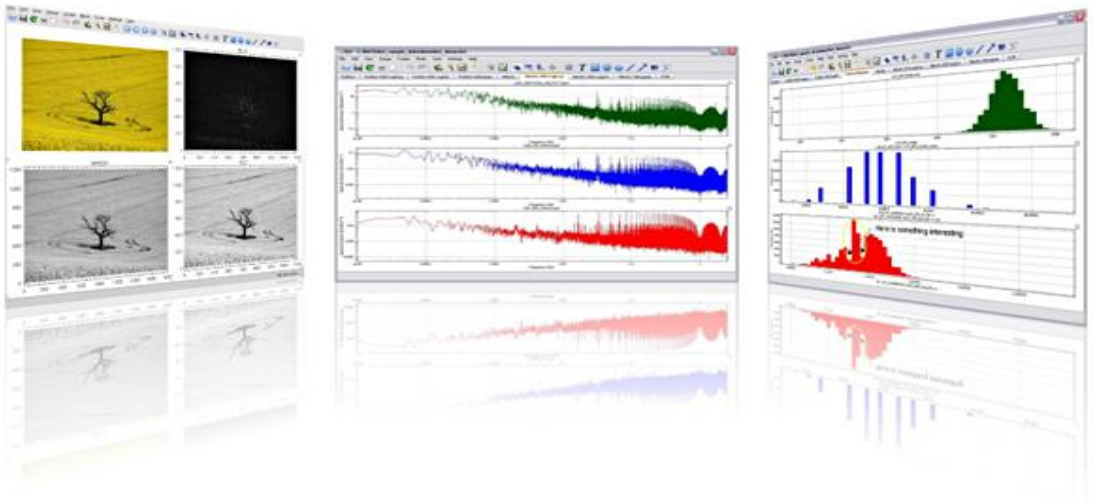

Figure 6.7 KST Viewing and Plotting

We are going to use KST for representing the ECG wave, airFlow and galvanic skin response.

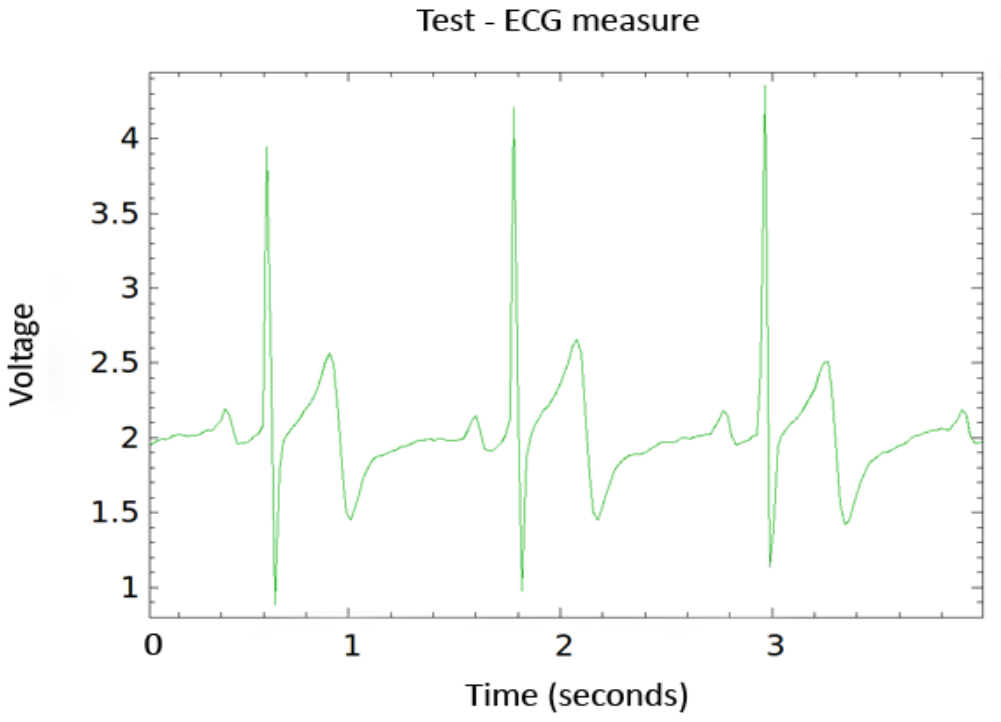

Figure 6.8 ECG Measurement in KST

EMG in KST

KST is configured to show a defined period of x time (x=300).

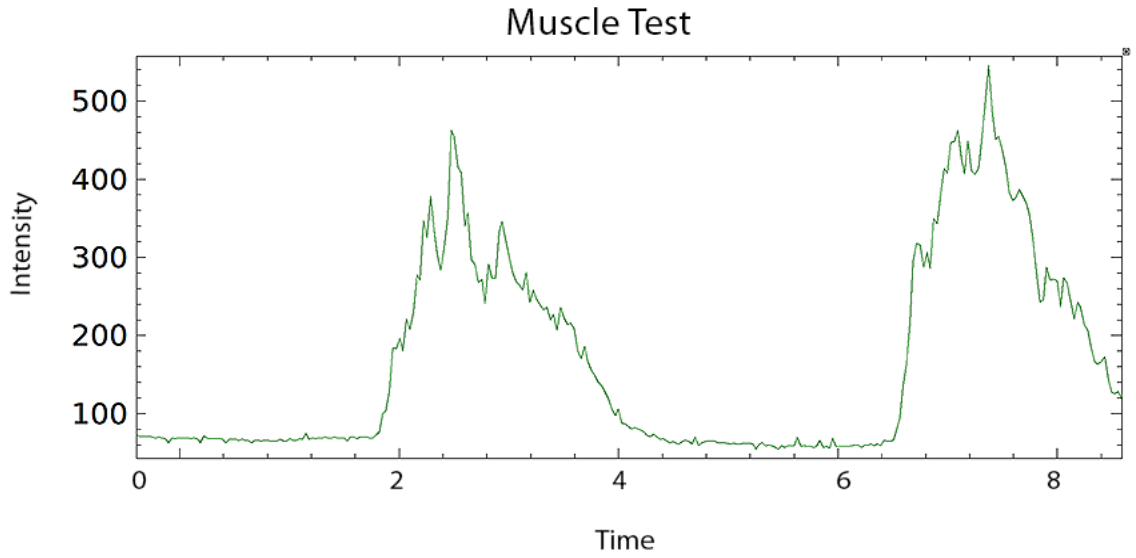

Figure 6.9 EMG Measurement in KST

Airflow in KST

We can configure the program according to the data we want to sample.

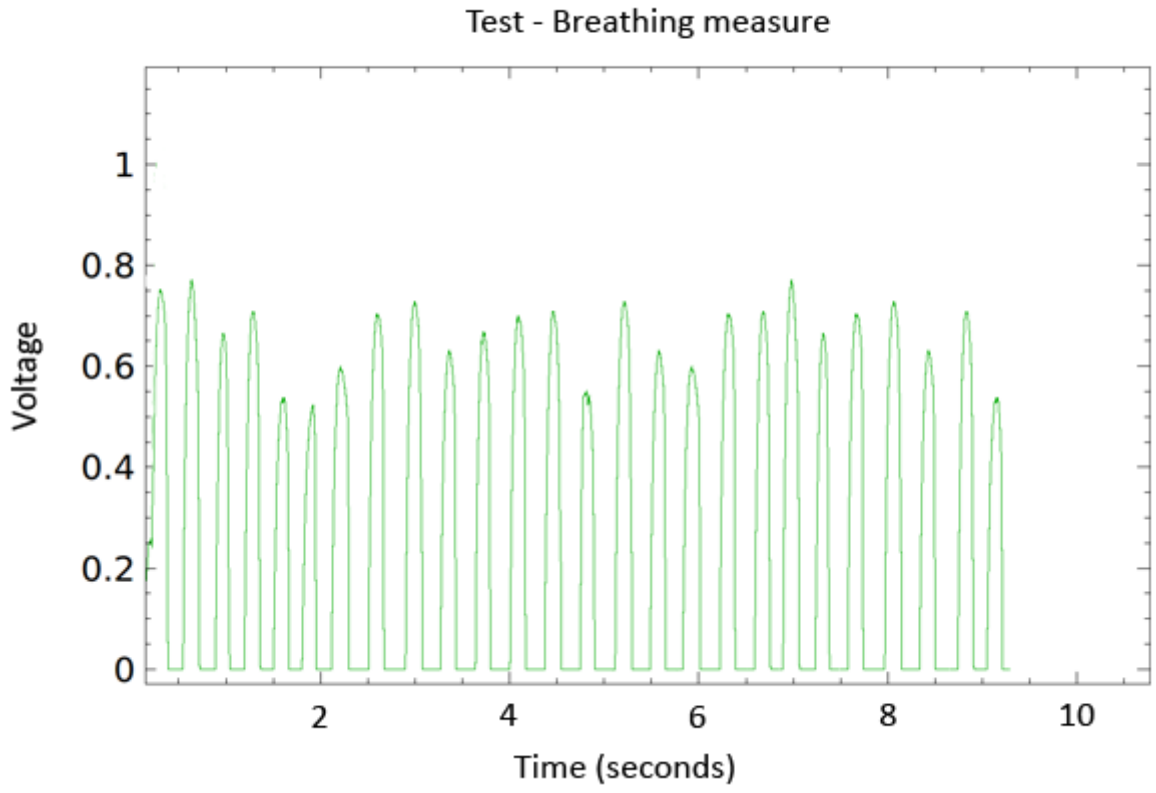

Figure 6.10 Airflow Measurement in KST

Full Code

```
#include <eHealth.h>
#include <PinChangeInt.h>

extern volatile unsigned long timer0_overflow_count;
float fanalog0;
int analog0;
unsigned long time;
byte serialByte;
char recv[128];
uint8_t cont = 0;

void setup() {
  Serial.begin(9600);
  Serial.println("Starting...");
  eHealth.initPulsioximeter();
  eHealth.initPositionSensor();
  PCintPort::attachInterrupt(6, readPulsioximeter, RISING);
}

void loop() {
    float temperature = eHealth.getTemperature();
    int BPM = eHealth.getBPM();
    int SPO2 = eHealth.getOxygenSaturation();
    float ECG = eHealth.getECG();
    fanalog0=eHealth.getAirFlow();
    time=(timer0_overflow_count << 8) + TCNT0;
    time=time*4;

    Serial.print(time);
    Serial.print(";");
    Serial.print(fanalog0,5);
    Serial.print(";");
    Serial.print(temperature);
    Serial.print(";");
    Serial.print(BPM);
    Serial.print(";");
    Serial.println(SPO2);
    Serial.print(";");
    Serial.println(ECG);
}
```

```
void readPulsioximeter(){

  cont ++;

  if (cont == 50) { //Get only one of 50 measures to reduce the latency
    eHealth.readPulsioximeter();
    cont = 0;
  }
}
```

In the complete code, we aim to get data from 5 different sensors at a time. We do this by outputting the data obtained from sensors against time in an endless loop. The data is then captured by a serial monitoring software and saved to a file. Inputting the file to KST with a custom delimiter of ";" results in the ability to read the data from sensors seperately and at once. Then this data is plotted against time in KST with separate plots.

The resultant data can then be saved as raw data file or as a screenshot to a high quality image file or even a vector file. This file can be analyzed by a physician or can be used for building a health history file for a patient.

6.3 Serial Console

All data can be visualized in the serial monitor of Arduino by using the next serial console program.

In the main menu we can find all e-Health sensors. By sending a command to the serial port to select one of them.

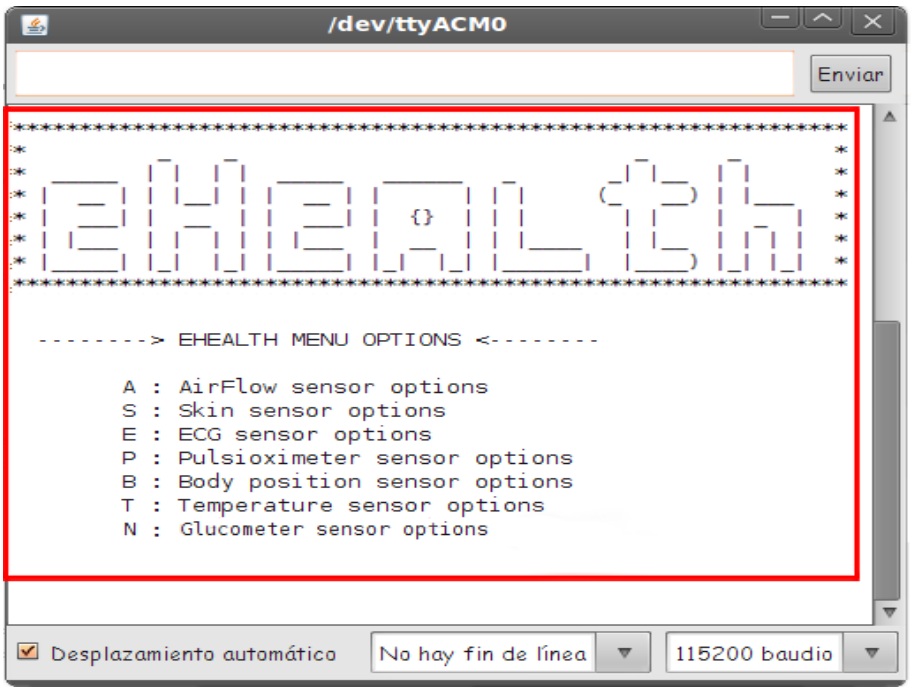

Figure 6.11 Serial Console Screen

A particular program can also be coded for serial console with various selective functions to represent the data incoming from sensors. Each option results in a loop for that sensor and runs until the data flow is cancelled by key stroke “F”. The program is also interactive and returns to main menu when a cancel signal is inputted.

Using a program in this manner, we have the ability to test and calibrate the sensors without having to direct them to a computer or another device. This is considered as the simplest form of data retrieval from our device.

6.4 SmartPhone Application

The wifi module may perform direct communications with iPhone and Android devices without the need of an intermediate router by creating an Adhoc network between them.

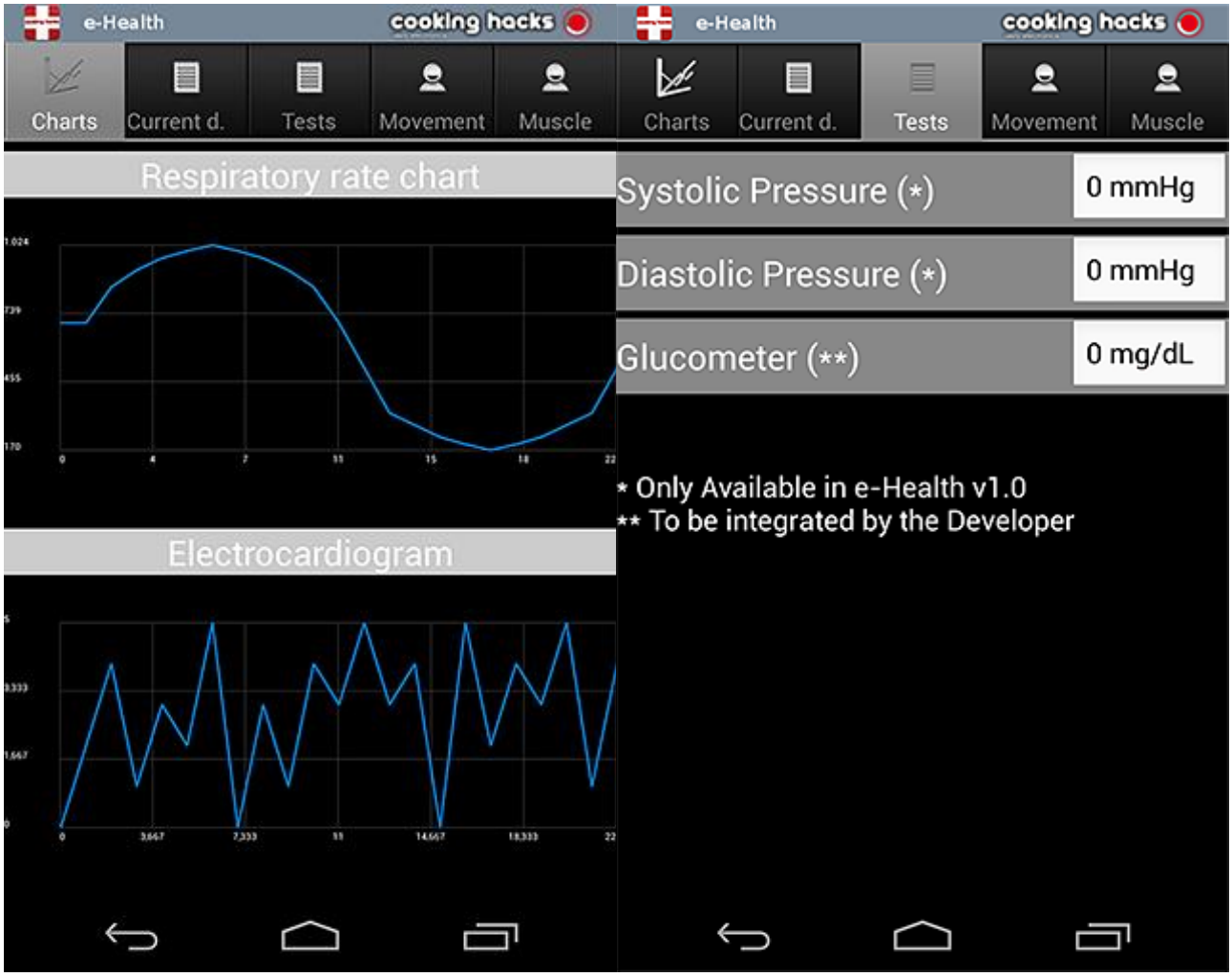

Figure 6.12 Smartphone Application

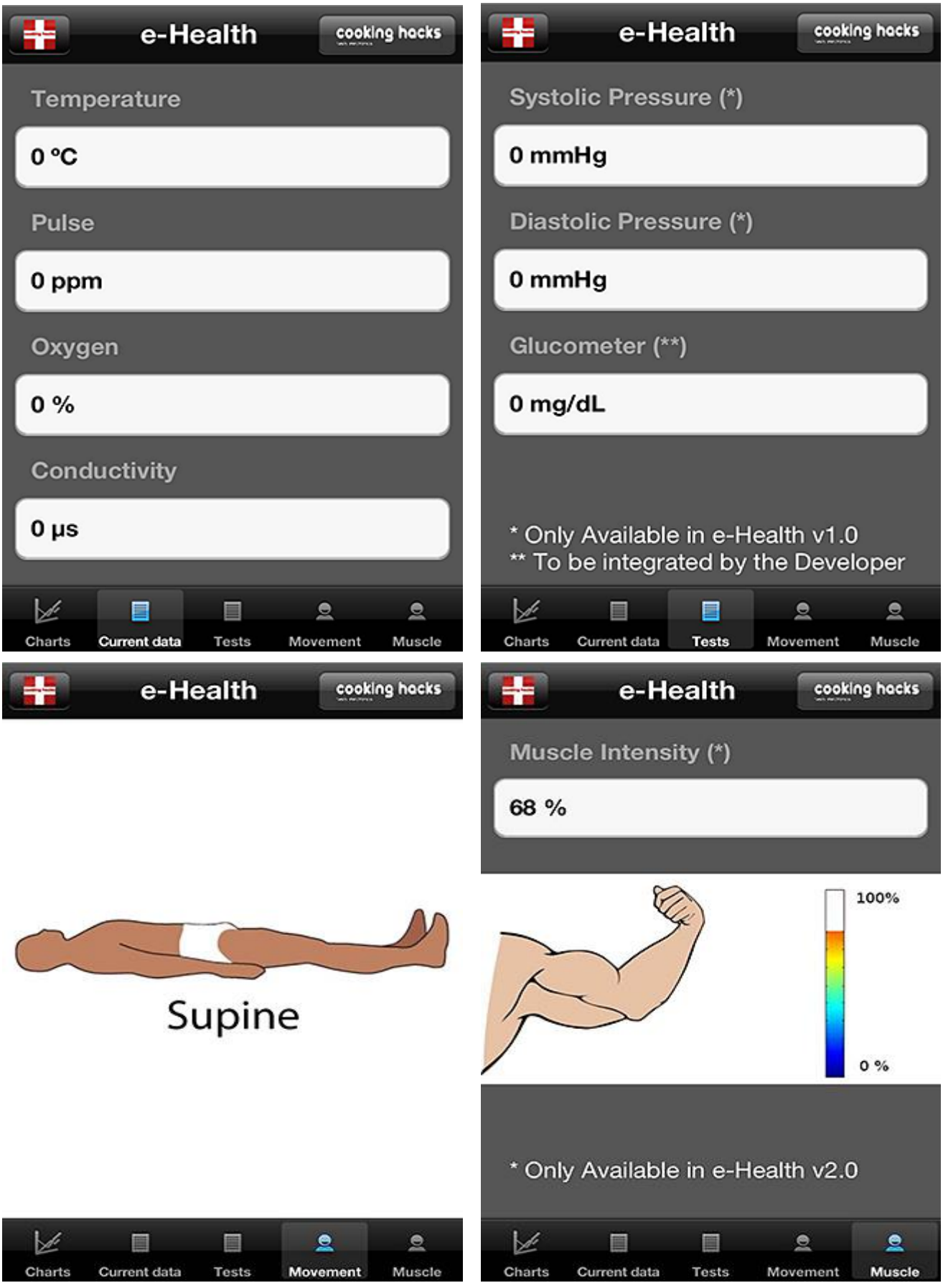

Figure 6.13 Various Measurements on Smartphone Application

Full Code

```
#include <PinChangeInt.h>
#include <eHealth.h>

char recv[128];
uint8_t cont = 0;

void setup()
{
  Serial.begin(9600);
  eHealth.initPulsioximeter();
  eHealth.initPositionSensor();
  //Attach the inttruptions for using the pulsioximeter.
  PCintPort::attachInterrupt(6, readPulsioximeter, RISING);
  delay(1000);
  delay(10000);
}
void loop()
{
  while (Serial.available()>0) {}
    Serial.print("$$$"); check();
    Serial.print("set ip dhcp 1\r"); check();
    Serial.print("set ip protocol 1\r"); check();
    Serial.print("set wlan join 0\r"); check();
    Serial.print("join ANDROID\r"); check();

    Serial.print("set i h 255.255.255.255\r"); delay(1000);

    Serial.print("set i r 12345\r"); check();
    Serial.print("set i l 2000\r"); check();
    Serial.print("exit\r"); check();

    while(1){

      //1. Read from eHealth.
       int airFlow = eHealth.getAirFlow();
       float temperature = eHealth.getTemperature();
       float conductance = eHealth.getSkinConductance();
       float resistance = eHealth.getSkinResistance();
       float conductanceVol = eHealth.getSkinConductanceVoltage();
       int BPM = eHealth.getBPM();
       int SPO2 = eHealth.getOxygenSaturation();
       uint8_t pos = eHealth.getBodyPosition();
```

```
      float ECG = eHealth.getECG();

     Serial.print(int(airFlow));     Serial.print("#");
     Serial.print(int(ECG));         Serial.print("#");
     Serial.print(int(temperature)); Serial.print("#");
     Serial.print(int(BPM));         Serial.print("#");
     Serial.print(int(SPO2));        Serial.print("#");
     Serial.print(int(conductance)); Serial.print("#");
     Serial.print(int(resistance));  Serial.print("#");
     Serial.print(int(airFlow));     Serial.print("#");
     Serial.print(int(pos));         Serial.print("#");
     Serial.print("\n");

     // Delay for data rate
     delay(250);
  }
}
void check(){
  cont=0; delay(500);
  while (Serial.available()>0)
  {
     recv[cont]=Serial.read(); delay(10);
     cont++;
  }
  recv[cont]='\0';
  Serial.println(recv);
  Serial.flush(); delay(100);
}
void readPulsioximeter(){

  cont ++;

  if (cont == 50) {
    eHealth.readPulsioximeter();
    cont = 0;
  }
}
```

7. Telemedicine

Telemedicine is the use of telecommunication and information technologies in order to provide clinical health care at a distance. It helps eliminate distance barriers and can improve access to medical services that would often not be consistently available in distant rural communities. It is also used to save lives in critical care and emergency situations.

Although there were distant precursors to telemedicine, it is essentially a product of 20th century telecommunication and information technologies. These technologies permit communications between patient and medical staff with both convenience and fidelity, as well as the transmission of medical, imaging and health informatics data from one site to another.

Telemedicine can be beneficial to patients living in isolated communities and remote regions, who can receive care from doctors or specialists far away without the patient having to travel to visit them. Recent developments in mobile collaboration technology can allow healthcare professionals in multiple locations to share information and discuss patient issues as if they were in the same place.

Remote patient monitoring through mobile technology can reduce the need for outpatient visits and enable remote prescription verification and drug administration oversight, potentially significantly reducing the overall cost of medical care. Telemedicine can also facilitate medical education by allowing workers to observe experts in their fields and share best practices more easily.

Telemedicine also can eliminate the possible transmission of infectious diseases or parasites between patients and medical staff. This is particularly an issue where MRSA is a concern. Additionally, some patients who feel uncomfortable in a doctors office may do better remotely. For example, white coat syndrome may be avoided. Patients who are home-bound and would otherwise require an ambulance to move them to a clinic are also a consideration.

eHealth Sensor platform allows to share medical data with the cloud, and perform real-time diagnosis. Thanks to many communications modules can send data over several transmission protocols.

7.1 Databases

A database is an organized collection of data. The data are typically organized to model relevant aspects of reality in a way that supports processes requiring this information as modelling the availability of rooms in hotels in a way that supports finding a hotel with vacancies.

Database management systems (DBMSs) are specially designed software applications that interact with the user, other applications, and the database itself to capture and analyze data. A general-purpose DBMS is a software system designed to allow the definition, creation, querying, update, and administration of databases. Well-known DBMSs include MySQL, MariaDB, PostgreSQL, SQLite, Microsoft SQL Server, Microsoft Access, Oracle, SAP HANA, dBASE, FoxPro, IBM DB2, LibreOffice Base, FileMaker Pro and InterSystems Caché.

Formally, database refers to the data themselves and supporting data structures. Databases are created to operate large quantities of information by inputting, storing, retrieving and managing that information. Databases are set up so that one set of software programs provides all users with access to all the data.

A database management system is a suite of computer software providing the interface between users and a database or databases. Because they are so closely related, the term "database" when used casually often refers to both a DBMS and the data it manipulates.

The interactions catered for by most existing DBMSs fall into four main groups:

- Data definition – Defining new data structures for a database, removing data structures from the database, modifying the structure of existing data.
- Update – Inserting, modifying, and deleting data.
- Retrieval – Obtaining information either for end-user queries and reports or for processing by applications.
- Administration – Registering and monitoring users, enforcing data security, monitoring performance, maintaining data integrity, dealing with concurrency control, and recovering information if the system fails.

A DBMS is responsible for maintaining the integrity and security of stored data, and for recovering information if the system fails.

7.1.1 MySQL

MySQL officially, but also called "My Sequel" is the world's second most widely used open-source relational database management system (RDBMS). It is named after co-founder Michael Widenius's daughter, My. The SQL phrase stands for Structured Query Language.

The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL was owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now owned by Oracle Corporation.

MySQL is a popular choice of database for use in web applications, and is a central component of the widely used LAMP open source web application software stack. LAMP is an acronym for "Linux, Apache, MySQL, Perl/PHP/Python." Free-software-open source projects that require a full-featured database management system often use MySQL.

For commercial use, several paid editions are available, and offer additional functionality. Applications which use MySQL databases include: TYPO3, MODx, Joomla, WordPress, phpBB, MyBB, Drupal and other software. MySQL is also used in many high-profile, large-scale websites.

MySQL ships with many command line tools, from which the main interface is 'mysql' client. Third-parties have also developed tools to manage MySQL servers.

MySQL Utilities – a set of utilities designed to perform common maintenance and administrative tasks. Originally included as part of the MySQL Workbench, the utilities are now a stand-alone download available from Oracle.

MySQL works on many system platforms, including AIX, BSDi, FreeBSD, HP-UX, eComStation, i5/OS, IRIX, Linux, OS X, Microsoft Windows, NetBSD, Novell NetWare, OpenBSD, OpenSolaris, OS/2 Warp, QNX, Oracle Solaris, Symbian, SunOS, SCO OpenServer, SCO UnixWare, Sanos and Tru64. A port of MySQL to OpenVMS also exists.

MySQL is written in C and C++. Its SQL parser is written in yacc, but it uses a home-brewed lexical analyzer. Many programming languages with language-specific APIs include libraries for accessing MySQL databases.

7.2 HTML

HTML or HyperText Markup Language is the standard markup language used to create web pages.

HTML is written in the form of HTML elements consisting of tags enclosed in angle brackets (like <html>). HTML tags most commonly come in pairs like <h1> and </h1>, although some tags represent empty elements and so are unpaired, for example <img>. The first tag in a pair is the start tag, and the second tag is the end tag.

The purpose of a web browser is to read HTML documents and compose them into visible or audible web pages. The browser does not display the HTML tags, but uses the tags to interpret the content of the page. HTML describes the structure of a website semantically along with cues for presentation, making it a markup language rather than a programming language.

HTML elements form the building blocks of all websites. HTML allows images and objects to be embedded and can be used to create interactive forms. It provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. It can embed scripts written in languages such as JavaScript which affect the behavior of HTML web pages.

Web browsers can also refer to Cascading Style Sheets (CSS) to define the look and layout of text and other material. The W3C, maintainer of both the HTML and the CSS standards, encourages the use of CSS over explicit presentational HTML.

HTML markup consists of several key components, including tags (and their attributes), character-based data types, character references and entity references. Another important component is the document type declaration, which triggers standards mode rendering.

HTML documents also imply a structure of nested HTML elements. These are indicated in the document by HTML tags, enclosed in angle brackets as: <p>

HTML Code - Homepage

```
<html>
<head>
<title>
Login page
</title>
</head>
<body>
<div style="text-align: center;"><IMG SRC="ehealth.jpg"
ALT="image"></div>
<div style="text-align: center;"><IMG SRC="ehealth2.jpg"
ALT="image"></div>
</h1>
<p>
<center><form name="login" >
Username<input type="text" name="userid"/>
Password<input type="password" name="pswrd"/>
<input type="submit" onclick="check(this.form)" value="Login"/>
<input type="reset" value="Cancel"/>
</form></center>
<script language="javascript">
function check(form)/*function to check userid & password*/
{
 /*checks whether the entered userid and password are matching*/
 if(form.userid.value == "doctor" && form.pswrd.value == "password")
  {
    window.open('list.php')/*opens the target page while Id & password
matches*/
  }
 else if(form.userid.value == "doctor2" && form.pswrd.value == "password")
  {
    window.open('list2.php')/*opens the target page while Id & password
matches*/
  }
 else
 {
   alert("Error Password or Username")/*displays error message*/
  }
}
</script>
</body>
</html>
```

HTML Code – Manual Entry

```
<!DOCTYPE html>
<html>
<body>
<div style="text-align: center;"><IMG SRC="ehealth.jpg"
ALT="image"></div>
<div id="container">
   <div id="content">
   <center><form name="input" action="add.php" method="get">
   <label>Name:</label> <input type="text" name="Name"><br>
   <label>Age:</label> <input type="text" name="Age"><br>
   <label>Temperature:</label></label> <input type="text"
name="Temp"><br>
   <label>Pulse:</label> <input type="text" name="Pulse"><br>
   <label>Oxygenation:</label> <input type="text" name="Oxygenation"><br>
   <p>
   <input type="submit" value="Submit">
   </form></center>
   </div>
</div>
</body>
</html>
<style>
          #container {
          width: 100%;
          clear: both;
       }
       #content {
          margin: auto;
          position:relative;
          width:300px;
       }
       .what {
          font-size:35px;
       }
       label {
          width: 100px;
          float: left;
       }
}
</style>
```

7.3 PHP

PHP is a server-side scripting language designed for web development but also used as a general-purpose programming language. As of January 2013, PHP was installed on more than 240 million websites (39% of those sampled) and 2.1 million web servers. Originally created by Rasmus Lerdorf in 1994, the reference implementation of PHP is now produced by The PHP Group. While PHP originally stood for Personal Home Page, it now stands for PHP: Hypertext Preprocessor, a recursive backronym.

PHP code can be simply mixed with HTML code, or it can be used in combination with various templating engines and web frameworks. PHP code is usually processed by a PHP interpreter, which is usually implemented as a web server's native module or a Common Gateway Interface (CGI) executable. After the PHP code is interpreted and executed, the web server sends resulting output to its client, usually in form of a part of the generated web page – for example, PHP code can generate a web page's HTML code, an image, or some other data. PHP has also evolved to include a command-line interface (CLI) capability and can be used in standalone graphical applications.

PHP is free software released under the PHP License. PHP has been widely ported and can be deployed on most web servers on almost every operating system and platform, free of charge.

The PHP interpreter only executes PHP code within its delimiters. Anything outside its delimiters is not processed by PHP (although non-PHP text is still subject to control structures described in PHP code). The most common delimiters are <?php to open and ?> to close PHP sections. <script language="php"> and </script> delimiters are also available, as are the shortened forms <? or <?= (which is used to echo back a string or variable) and ?> as well as ASP-style short forms <% or <%= and %>. Short delimiters make script files less portable, since support for them can be disabled in the local PHP configuration, and they are therefore discouraged. The purpose of all these delimiters is to separate PHP code from non-PHP code, including HTML.

PHP stores whole numbers in a platform-dependent range, either a 64-bit or 32-bit signed integer equivalent to the C-language long type. Unsigned integers are converted to signed values in certain situations; this behavior is different from other programming languages. Integer variables can be assigned using decimal (positive and negative), octal, hexadecimal, and binary notations.

PHP Code – Add Entry

```
<?php
date_default_timezone_set('Europe/Istanbul');
$script_tz = date_default_timezone_get();
$fulldate = date("Y-m-d h:i");
  include("conec.php");
  $link=Conection();
$Sql="insert into Patients values ('".$_GET["Name"]."', '".$_GET["Age"]."',
'".$_GET["Temp"]."', '".$_GET["Pulse"]."', '".$_GET["Oxygenation"]."',
'".$fulldate."')";
  mysql_query($Sql,$link);
?>
```

PHP Code – Connect to Database

```
<?php
function Conection(){
  if (!($link=mysql_connect("localhost","wejgomi1","password")))  {
    exit();
  }
  if (!mysql_select_db("Patients",$link)){
    exit();
  }
  return $link;
}
?>
```

PHP Code – List Patient Entries

```
<html>
<body>
<style type="text/css">
th,td{
border-width:0px 1px 1px 0px;
}
</style>
<?php
$connection = mysql_connect('localhost', 'wejgomi1', 'password');
mysql_select_db('Patients');

$query = "SELECT * FROM Patients";
$result = mysql_query($query);
```

```
echo'<table border="1" align="center"><th
>Name</th><th>Age</th><th>Temperature</th><th>Pulse</th><th>Oxygenat
ion</th><th>Date</th>'; // start a table tag in the HTML

while($row = mysql_fetch_array($result)){   //Creates a loop to loop through
results
  echo "<tr><td>" . $row['Name'] . "</td><td>" . $row['Age'] . "</td><td>" .
$row['Temperature'] ." </td><td>" . $row['Pulse'] ." </td><td>" .
$row['Oxygenation'] ." </td><td>" . $row['Date'] . "</td></tr>";  //$row['index']
the index here is a field name
}

echo "</table>"; //Close the table in HTML

mysql_close(); //Close out the database connection
?>
<p>
<center><button
onclick="window.location.href='manual.html'">Manual</button></center>
</body>
</html>
```

Full Code

```
#include <PinChangeInt.h>
#include <eHealth.h>
#include <AltSoftSerial.h>
#include <WiFlyHQ.h>
AltSoftSerial wifiSerial(8,9);

int cont = 0;
char buf[32];

WiFly wifly;
const char mySSID[] = "RouterSSID";
const char myPassword[] = "RouterPass";
const char site[] = "23.229.134.71";

void setup() {
  Serial.begin(115200);
  eHealth.initPulsioximeter();.
  PCintPort::attachInterrupt(6, readPulsioximeter, RISING);
}

void loop() {
  delay(1000);
  Serial.print("PRbpm : ");
  Serial.print(eHealth.getBPM());

  Serial.print("    %SPo2 : ");
  Serial.print(eHealth.getOxygenSaturation());

  Serial.print("    Temperature : ");
  float tempval = eHealth.getTemperature();
  Serial.print(eHealth.getTemperature());

  Serial.print("\n");

Serial.println("=========================================");
  //pulsed = eHealth.getBPM();
  int oxygen = eHealth.getOxygenSaturation();
  float d = eHealth.getBPM();
  String namede = "Name"; // Input name
  int agede = 24; // Input age
  if (eHealth.getBPM()) {
    delay(10000);
```

```
Serial.println("Starting");
Serial.print("Free memory: ");
Serial.println(wifly.getFreeMemory(),DEC);

wifiSerial.begin(9600);
if (!wifly.begin(&wifiSerial, NULL)) {
    Serial.println("Failed to start wifly");
    terminal();
}

if (!wifly.isAssociated()) {
    Serial.println("Joining network");
    wifly.setSSID(mySSID);
    wifly.setPassphrase(myPassword);
    wifly.enableDHCP();

    if (wifly.join()) {
        Serial.println("Joined wifi network");
    } else {
        Serial.println("Failed to join wifi network");
        terminal();
    }
 } else {
     Serial.println("Already joined network");
 }
 Serial.print("MAC: ");
 Serial.println(wifly.getMAC(buf, sizeof(buf)));
 Serial.print("IP: ");
 Serial.println(wifly.getIP(buf, sizeof(buf)));
 Serial.print("Netmask: ");
 Serial.println(wifly.getNetmask(buf, sizeof(buf)));
 Serial.print("Gateway: ");
 Serial.println(wifly.getGateway(buf, sizeof(buf)));

 wifly.setDeviceID("Wifly-WebClient");
 Serial.print("DeviceID: ");
 Serial.println(wifly.getDeviceID(buf, sizeof(buf)));

 if (wifly.isConnected()) {
     Serial.println("Old connection active. Closing");
     wifly.close();
 }
```

```
    if (wifly.open(site, 80)) {
        Serial.print("Connected to ");
        Serial.println(site);
            /* Send the request */
        wifly.print("GET http://www.epatientrecords.info/add.php?");
        wifly.print("Name=");
        wifly.print( namede );
        wifly.print("&");
        wifly.print("Age=");
        wifly.print( agede );
        wifly.print("&");
        wifly.print("Temp=");
        wifly.print( tempval );
        wifly.print("&");
        wifly.print("Pulse=");
        wifly.print( d );
        wifly.print("&");
        wifly.print("Oxygenation=");
        wifly.print( oxygen );
        wifly.println(" HTTP/1.1");
        wifly.println("Host: http://www.epatientrecords.info");
        wifly.println( "Content-Type: application/x-www-form-urlencoded" );
        wifly.println( "Connection: close" );
        wifly.println();
        Serial.println("Data sent");
        delay(5000);
        eHealth.initPulsioximeter();
        PCintPort::attachInterrupt(6, readPulsioximeter, RISING);
    }
    else {
        Serial.println("Failed to connect");
        delay(5000);
        eHealth.initPulsioximeter();
        PCintPort::attachInterrupt(6, readPulsioximeter, RISING);
    }
  }
  delay(500);
}
void terminal()
{
   while (1) {
          if (wifly.available() > 0) {
             Serial.write(wifly.read());
```

```
        }


        if (Serial.available() > 0) {
            wifly.write(Serial.read());
        }
    }
}
void readPulsioximeter(){
  cont ++;
  if (cont == 50) { //Get only of one 50 measures to reduce the latency
    eHealth.readPulsioximeter();
    cont = 0;
  }
}
```

In this code, we aim to integrate the health data to a website. This is done by gathering the health data from sensors on the platfrom. The data is then digitilized by the code and saved as integers and floating points. The program consists of a loop that awaits a living-activity from patient as temperature, pulse and oxygenation ratio retrieval and change. Then the loop is hold, Arduino connects to an SSID broadcasted by a router/access point and connects to a specified IP address(in this case, our website) to input the data.

The data is sent by using PHP coding on the server side. After the data is sent to server, it is processed by our PHP code(program) which connects to a MySQL server(also on our website), sorts, names and inputs the data as columns. Every data is saved as a new data with Name, Age, Date and Time tags.

This data saved in MySQL database then can be interacted from our website(www.epatientrecords.info). The website consists of a login site which allows the entrance from different physicians with name/password information. The physician can view the info listed only for him/her own patients and take measures comparing it to the normal limits of such parameters.

There is also a manual input entrance available for physicians' own comparing to the data obtained from our platform with the data measured in physicians room/hospital. The data categorized in the table can also be saved to physicians' own computer or smartphone/tablet for future viewing.

7.4 Wifi

We use the wifi module Roving RN-171. This module fits in the XBee socket of our Communication Shield and allows to connect the Arduino shield to a WiFi network.

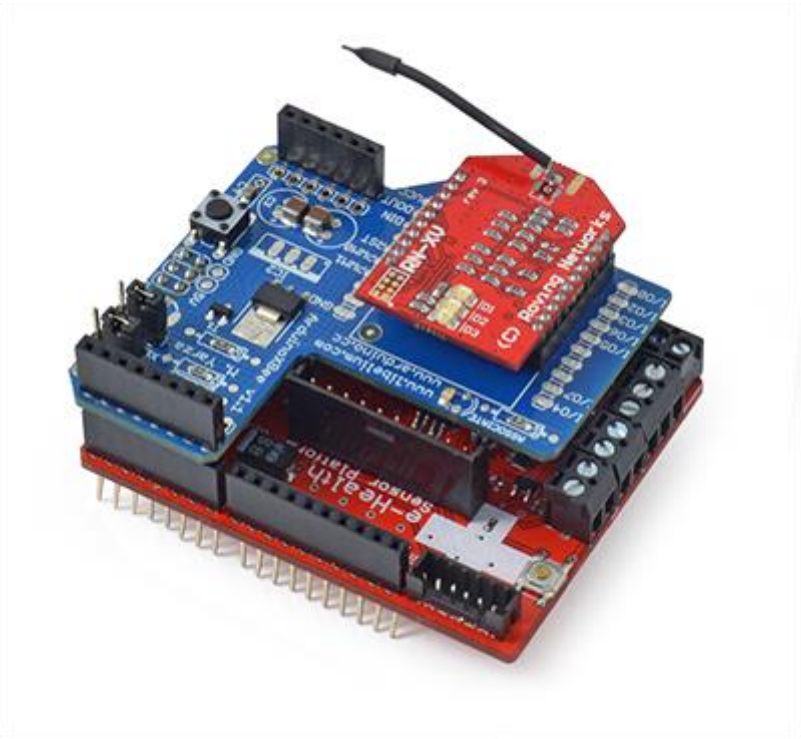

Figure 7.1 Wi-Fi Module

The RN-XV module is based upon Roving Networks' robust RN-171 Wi-Fi module and incorporates 802.11 b/g radio, 32 bit processor, TCP/IP stack, real-time clock, crypto accelerator, power management unit and analog sensor interface.

7.5 Bluetooth

Bluetooth Module for Arduino are plugged into the XBee Shield and get a serial communication between the computer and an Arduino board through Bluetooth protocol.

Figure 7.2 Bluetooth Module

Bluetooth module PRO for Arduino supports Serial Port Profile (SPP) to exchange data with other devices. This profile allows to create connections to another device using the same profile (p2p connection). It sends data to the specified device. This device is the one which the connection has been created to.

7.6 Zigbee / 802.15.4

The Arduino Xbee Shield allows Arduino board to communicate wirelessly using Zigbee.

Figure 7.3 Zigbee/802.15.4 Module

7.7 GPRS

GPRS Quadband Module for Arduino(SIM900) offers GPRS connection to Arduino board. Data can be sent by SMS or do missed calls from Arduino to mobile devices... or to another Arduino connected to this module.

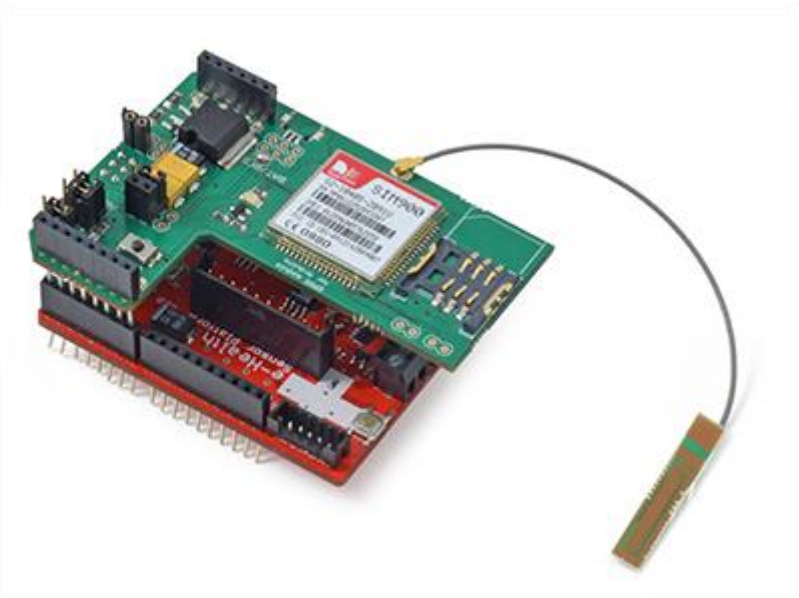

Figure 7.4 GPRS Module

7.8 3G

The 3G shield for Arduino enables the connectivity to high speed WCDMA and HSPA cellular networks in order to make possible the creation of the next level of worldwide interactivity projects inside the new era.

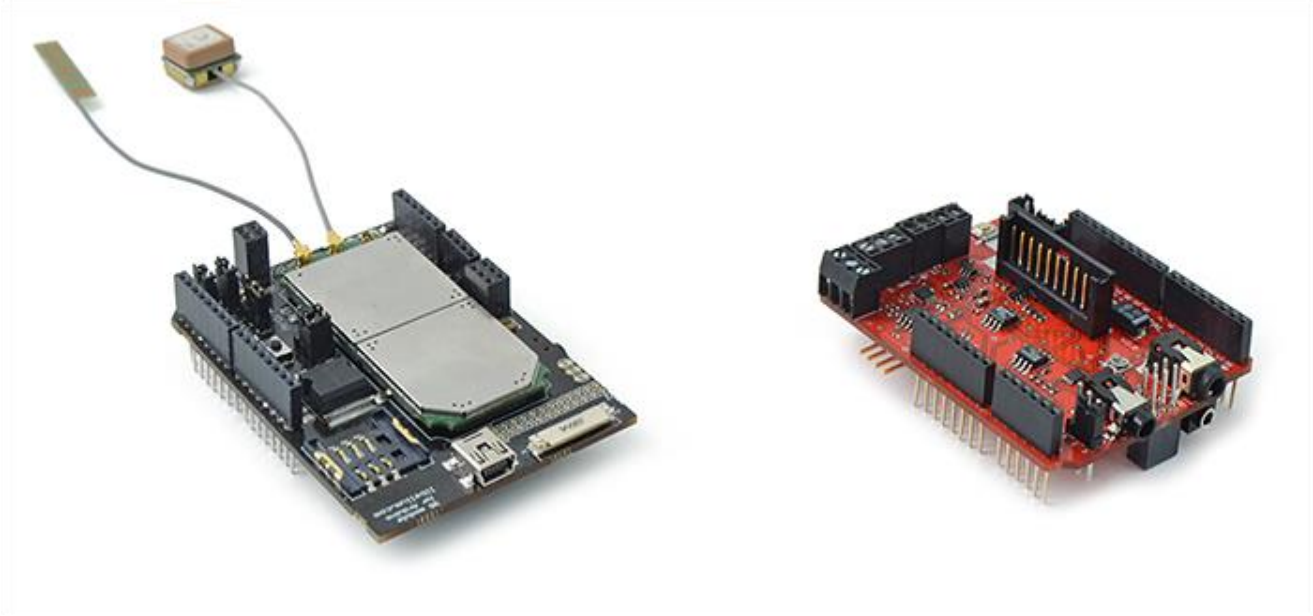

Figure 7.5 3G Module

7.9 Camera for Photo Diagnosis

The module allows to connect a camera for video recording and taking photos. Once saved the video or image file can be sent to an FTP or FTPS.

Camera for the 3G shield

Figure 7.6 Camera Module

8. References

1. http://www.cooking-hacks.com
2. http://www.wikipedia.org
3. http://www.arduino.cc
4. http://www.wolframalpha.com
5. http://nlhbi.nih.gov
6. http://www.ecglibrary.com
7. http://www.webmd.com
8. http://www.medicinenet.com
9. http://www.medterms.com
10. http://www.w3schools.com/
11. http://www.sparkfun.com
12. http://www.stackoverflow.com
13. http://www.synology.com
14. http://planet.mysql.com
15. http://www.dfrobot.com
16. http://www.launchpad.net