Ministry of high education,
Culture and science city at Oct 6,
The High institute of computer Science & information systems

المعهد العالي لعلوم الحاسب ونظم المعلومات

Graduation Project:

Remote healthcare monitoring system using ZigBee and LabVIEW.

Prepared by

٢٠١٦٠٠٩٠- مصطفي محمد احمد
٢٠١٦٠٠٨٩- دنيا جمال فتحي
٢٠١٦٠٥٠٩- عبدالقادر محمد عزت
٢٠١٦٠٦٨٧- عبد الرحمن احمد علي
٢٠١٦٠٦٩٤- رامي عيد سعد فهمي
٢٠١٦٠٧٢٣- عبدالرحمن علاء محمد
٢٠١٦٠٧٣٧- دميانه سمير رياض
٢٠١٦٠٨٢١- منة حلمي حلمي
٢٠١٦١٠٨٧- عبدالرحمن حسن حسين

Assistant:

Eng. Marwa El-Enany

Supervised by

Dr: Noha Ramadan

Project No: 3 Academic Year: 2019/2020

Acknowledgement

We have taken efforts in this project. However, it would not have been possible without the great Culture and Science City where our beloved computer science faculty is hosted and with the kind support and help of its professors and on the behalf of my colleagues I would like to extend our sincere thanks to all of them.

First, we would like to thanks Dr. Noha Ramadan for her patience, guidance, for her supervision as well as for providing necessary information regarding the project and also for her support in completing the project, We could not have imagined having a better advisor and mentor for our graduation project.

We also would like to express our sincere gratitude to our advisor Eng. Marwa El Enany for the continuous support of our graduation project and study, for her patience, motivation, enthusiasm, and immense knowledge. Her guidance helped us in all the time of project and writing of this thesis.

Finally, we would also like to expand our deepest gratitude to our family and friends who have supported us all the way when they haven't had the slightest idea on what we're doing.

Abstract

Remote health care monitoring system (RHCMS) has drawn considerable attentions for the last decade. As the aging population is increasing at a rapid pace. While at the same time health care cost is skyrocketing .A need to monitor a patient from a remote location has arisen.

Adding to that, People from poor areas of the world aren't getting the healthcare they need and deserve. To solve these problems many research and commercial versions of RHCMS have been proposed and implemented till now. In these systems the performance was the main issue to accurately measure, record, and analyze patients' data. With the ascent of wireless network RHCMS can be widely deployed to monitor the health condition of a patient inside and outside of the hospitals. This project presents a wireless (Zigbee, Bluetooth,..etc) healthcare based monitoring system that can provide real time online information about the health condition of a patient. The proposed system can send alarming messages to the healthcare professional about the patient's condition. Also the proposed system can send reports to a patient monitoring system, which can be used by the healthcare professionals to give necessary medical advice from anywhere in the world at any given time.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

1.1 An overview

Over the recent years remote health care monitoring systems for the elderly people have drawn considerable attentions. According to UNFPA, the global population is no longer young for the first time in the history. Population ageing is affecting the entire world and is happening in all regions. But, it is progressing at a faster rate in the developing countries. Seven out of the fifteen countries in the developing world have more than 10 million old people. By the year 2050 another fifteen developing countries are expected to have 10 million old people. It is worthwhile to mention here that the average life expectancy in the United States was 47.3 years in 1900. But, it has increased to 68.2 years and 77.3 years in 1950 and 2002 respectively. People are living longer because of better nutrition, sanitation, medical advances, education, economic well-being, and health care. Population ageing poses challenges to individuals, families, and societies. By adopting proper policies societies should be prepared for an ageing world. Overall, the older people should not be considered as a burden for the society. Their wisdom, energy, and experience are added advantages for us to take care of the challenges of the 21st century. In order to keep the ageing population healthy we have to deal with some challenges. The major challenge for us is to keep them healthy with our limited resources. Although numerous groundbreaking achievements have been noticed in the health care sector for the recent years, the health care expense is still sky high and it has become an issue that even the developed countries are worry about. According to the data provided by the Kaiser Family Foundation the per capita expenditure of the health care is increasing at an exponential rate in some countries as shown in Figure 1. With such a high and continuously increasing healthcare expenses, medical care for the ageing people is becoming progressively challenging. One of the reasons for this high medical expense is hospitalization cost. The senior citizens are the most frequent visitors to the hospitals. They visit the hospitals for their medical treatment. Sometimes they have to stay there for a certain period of time for follow up of their medical treatment. Their staying in a hospital not only incurs expenses, but also incurs loss of patient's mobility. Remote Health Care Monitoring System (RHCMS) has been proposed as a solution to this problem. The main concept is to monitor a patient from a remote

location and to provide her/him with necessary medical advices. The RHCMS has numerous advantages compared to conventional healthcare systems [1].

Some of the advantages includes:

- monitoring a patient,
- responding to an emergency,
- assisting patient mobility,
- shortening hospital stay, and
- reducing medical expenses.

Using RHCMS the physical conditions of the patients can be monitored for twenty hours a day and seven days a week. The emergency services can be provided to the patients with a minimum delay. The patients can be served without going to a health care facility and admitting there. The healthcare professional can perform the follow up from a remote location and hence a patient needs to stay in a hospital for a short period of time. In a nutshell RHCMS reduces expenses related to the medical services.

Wireless networks and medical sensors have been combined in RHCMS. There have been many medical sensors available in the market. Most of the sensors can measure and display critical health monitoring data such as the pulse rate, blood pressure, temperature, and blood sugar of a patient. One of the limitations of the proposed RHCMS is its limited coverage area. The measured data cannot be transmitted beyond a certain distance. Thus, it is not possible for the healthcare professionals to monitor the medical conditions of a patient from a distant location. In a hospital either the nurse or the physician has to move from one patient to another patient for monitoring them. Hence, it may not be possible for them to monitor a patient's health conditions all the time. This situation can be even worse when they have to take care of a large number of patients in a hospital at a given time. In order to overcome the above mentioned limitations an on-line health monitoring system has been proposed in this work. The proposed system can monitor the temperature, pulse, muscle, and ECG data of a patient. The proposed system has been designed by using the Bluetooth technology. A major portion of this system has been implemented in LabView [2].

1. 2 Project tools

The project has been designed using the aid of some software and hardware tools.

1.2.1 Hardware tools:

1. Sensors
    - ECG
    - Heart Rate
    - Temperature sensor (LM35)
2. Microcontroller
    - Arduino UNO
    - PIC with PIC KIT 2
3. Communication medium
    - TTL (wired)
    - Bluetooth (wireless)

1.2.2 Software tools:

1. Display data
    - Labview
2. Simulation
    - Proteus

1. 3 Book structure

This book is organized as follows:

Chapter 1: Introduction

- Briefly describes the projects, its compotes and gives an overviews of the other chapters.

Chapter 2: System Analysis

- The system architecture is being discussed in this chapter. System analysis, required resources and components were presented in this chapter.

Chapter 3: Hardware

- detailed specifications of hardware components and comparisons between them were presented in this chapter. Also connection and final hardware design was proposed in this chapter.

Chapter 4: Software

- The software being used in this project is explained in this chapter. The software includes Labview which is a system-design platform and development environment for a visual programming language and The Proteus Design Suite which is a proprietary software tool suite used primarily for electronic design automation.

Chapter 5: Results

- The results that are being shown by Labview, Pic microcontroller, and Arduino microcontroller are all being shown in this chapter.

Conclusion and Future work

- A brief summary of the project and derived results were concluded beside the future proposal and suggestions for this project were done .

Chapter 2: System Analysis

2.1 Proposed System Architecture

- Patients use sensors to determine information about their health condition.
- Sensors send data extracted from Patients.
- Temperature sensor acquire temperature data.
- Heartbeat sensor determine the heart pulse in the form of graph.
- Heart rate sensor calculate the number of pulse in the form of digits.
- This data is sent to the Microcontroller using pins.
- Microcontroller is processing this data.
- This data is sent to LabVIEW using Bluetooth Module.
- Laptop Bluetooth receive this data.
- LabVIEW present result from this data.
- Mobile Application receive alarm from LabVIEW [3].

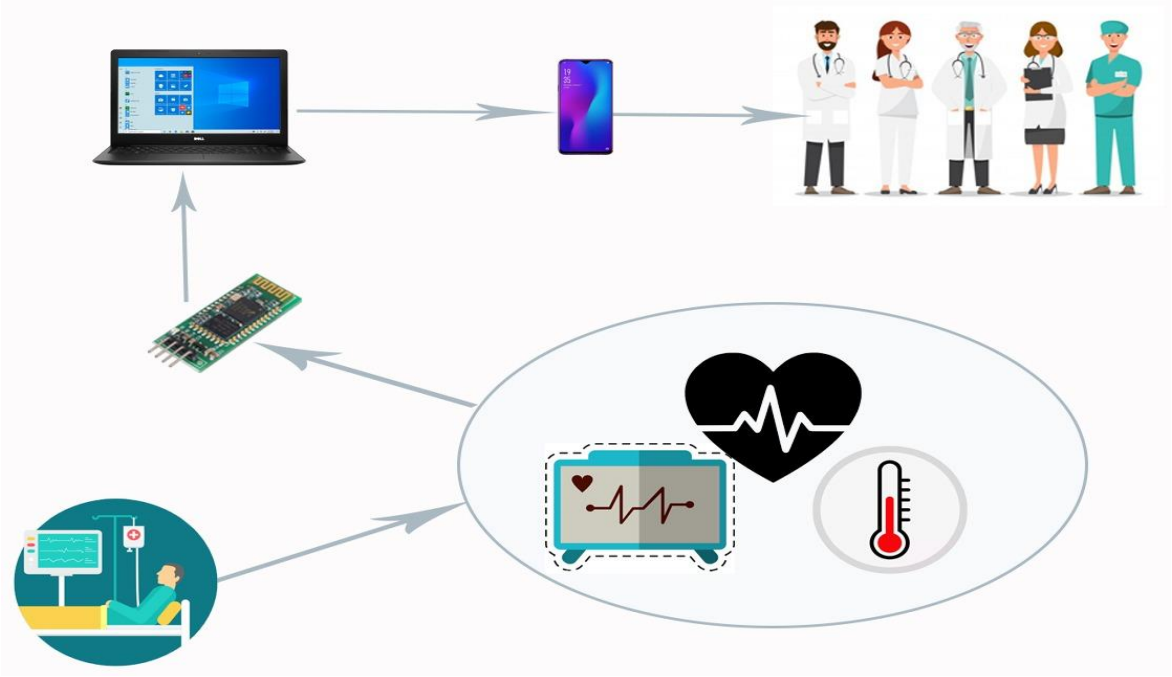

Figure 2.1 Proposed System Architecture Diagram with Microcontroller

2.2 System block diagram

Block diagram shows a system component and interaction between them, and the main components are (Sensors, Microcontroller, Bluetooth, Laptop's Bluetooth, LabVIEW, Mobile application, and User)

- All sensors send data to Microcontroller.
- Microcontroller takes this data to convert it from analog to digital then send it to Bluetooth Module.
- Bluetooth module is used to send this data to LabVIEW.
- Laptop Bluetooth will receive this data.
- LabVIEW displays the information of this data to the patient.
- User receive alarm on his Mobile Application.
- Mobile Application sends SMS as alarm Message to patient phone [4].

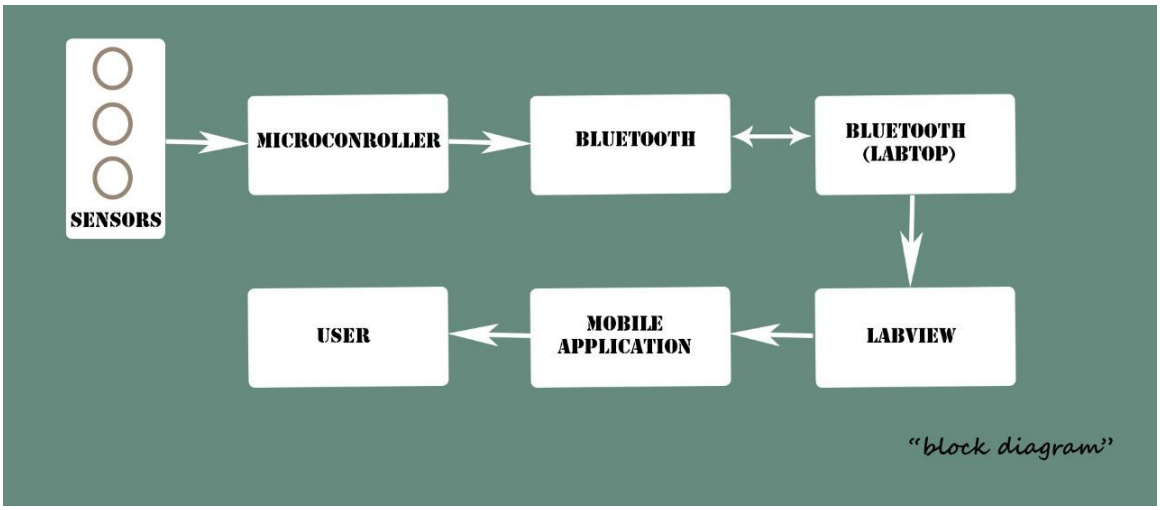

Figure 2.2 proposed system block diagram

2.3 System use case

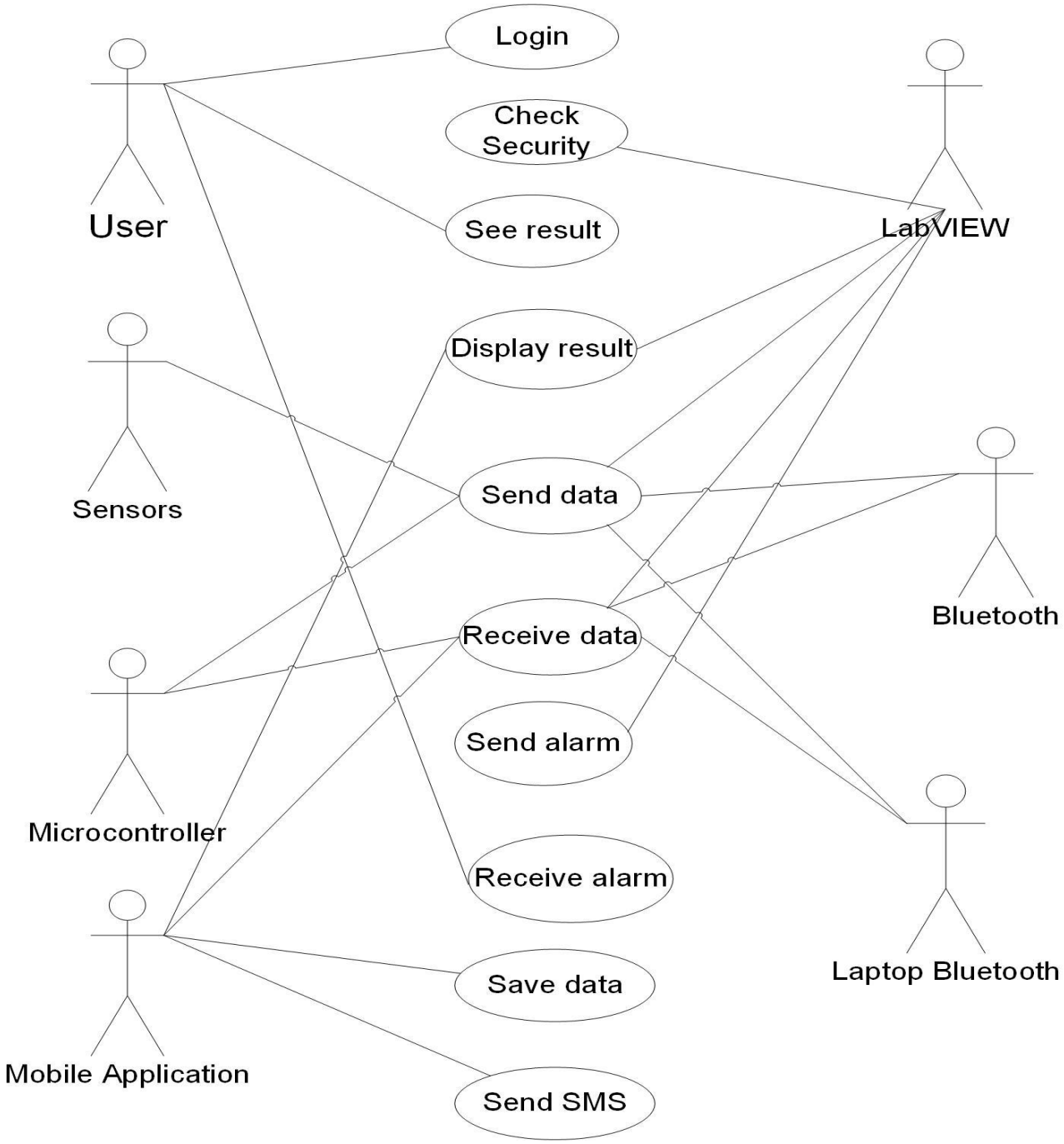

Figure 2.3 Use case diagram for the System

Actors	User, Sensors, Microcontroller, Bluetooth Module, Laptop Bluetooth, LabVIEW, and Mobile Application.
Description	User make the process of login, see result, and receive alarm, sensors make process of sending data to the Microcontroller, Microcontroller receive data, and send data, Bluetooth Module receive data, and send data to laptop Bluetooth, Laptop Bluetooth receive data, and send it to LabVIEW, LabVIEW receive data, display result, send data, check security, and send alarm to Mobile application, The Mobile application receive data, display result, save data, and send SMS.
Data	User and sensors are transferring data, Microcontroller receive & send sensors data through Bluetooth Module and Laptop Bluetooth, LabVIEW receive data and send it to Mobile application, Mobile application save data received.
Response	Fast display results, excellent represent of information quality, and high-performance Microcontroller.
Comment	Sensors types: - Temperature sensor (LM35):- measure the temperature of the human. Heartbeat sensor (ECG):- measure the heart condition in a form of graph. Heart rate sensor (Pbm):- measure the heart condition in the form of rates. Bluetooth Module: - transfer data from the Microcontroller to the laptop Bluetooth.

Table 2.1 description of healthcare system use case diagram [5]

2.4 First Scenario (using Pic as a Microcontroller)

In first scenario, it will determine some diagrams to show the interaction between components in the system with using a Pic as a Microcontroller:-

system diagram determines how the data flows from each sensor to the Pic Microcontroller, then it is transferred using Bluetooth module. After that data is received by laptop Bluetooth, and represented by LabVIEW as result on the screen to the user, and if result is worth an alarm is send to Mobile application [7].

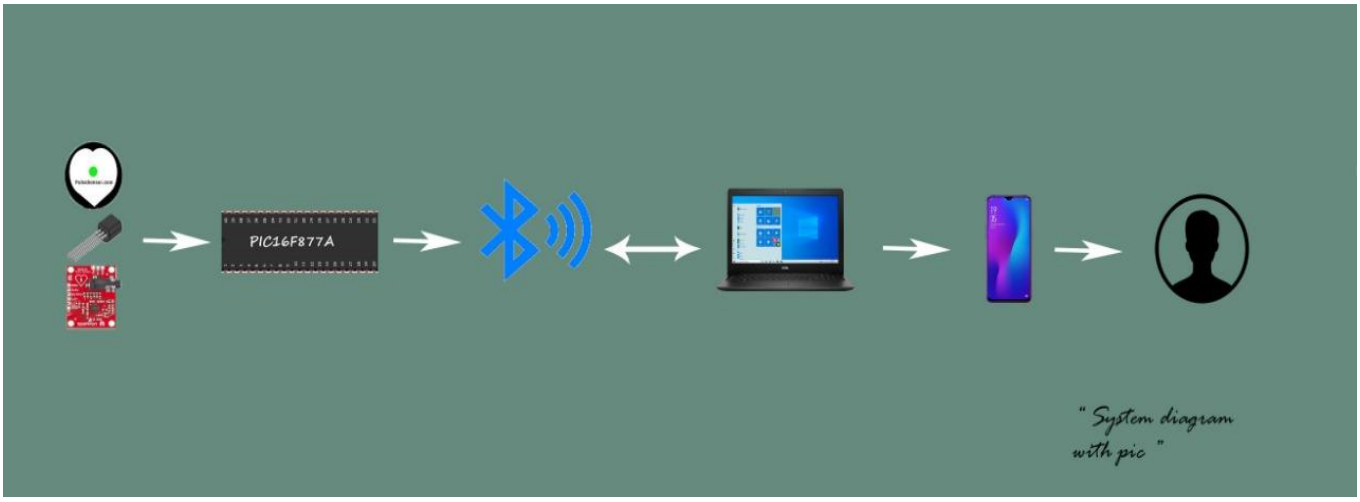

Figure 2.4 System diagram for the system with Pic Microcontroller

2.4.1 Deployment diagram with Pic Microcontroller

Deployment diagram shows how components connected to each other sensors as (Temperature, Heartbeat, and Heart rate) connected to the pic Microcontroller then Pic is connected to Bluetooth Module to send data [8].

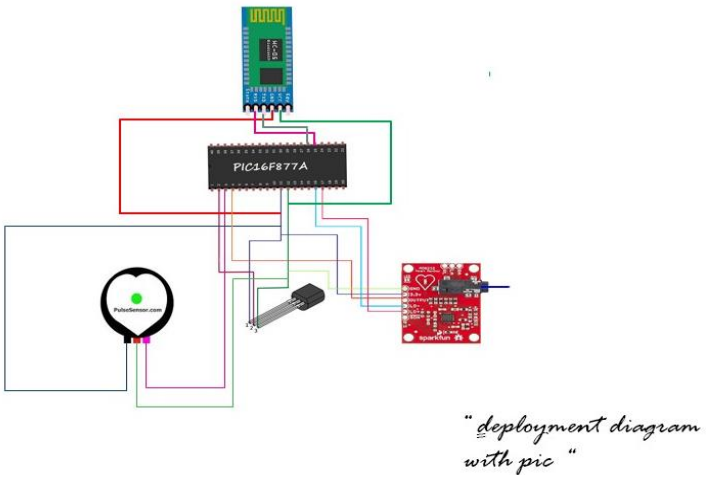

Figure 2.5 Deployment diagram for the system with Pic Microcontroller

2.5 Second Scenario (using Arduino as a Microcontroller)

In second scenario, we will determine some diagrams show to us the interactive of components in our system so we begin with the system with Arduino:-

system diagram determines how the data flows from each sensor to the Arduino, then it is transferred using Bluetooth module. After that data is received by laptop Bluetooth, and represented by LabVIEW as result on the screen to the user, and if result is worth an alarm is send to Mobile application.

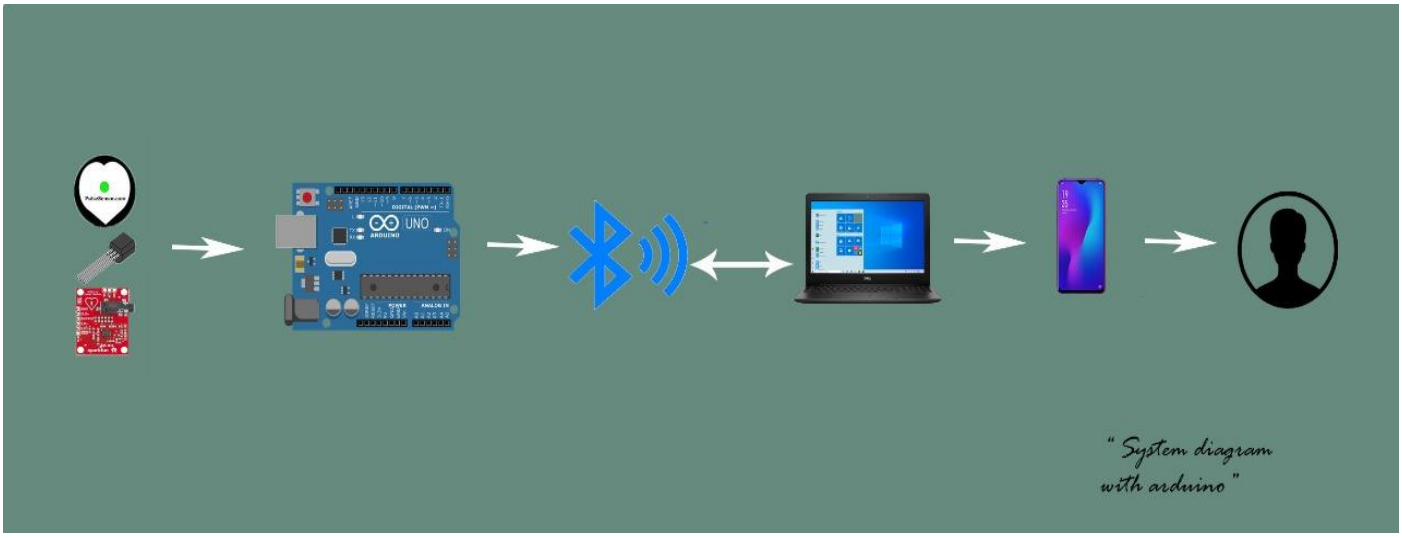

Figure 2.6 System diagram for the system with Arduino

2.5.1 Deployment diagram with Arduino

Deployment diagram shows how components connected to each other sensors as (Temperature, Heartbeat, and Heart rate) connected to the Arduino then Arduino is connected to Bluetooth Module to send data.

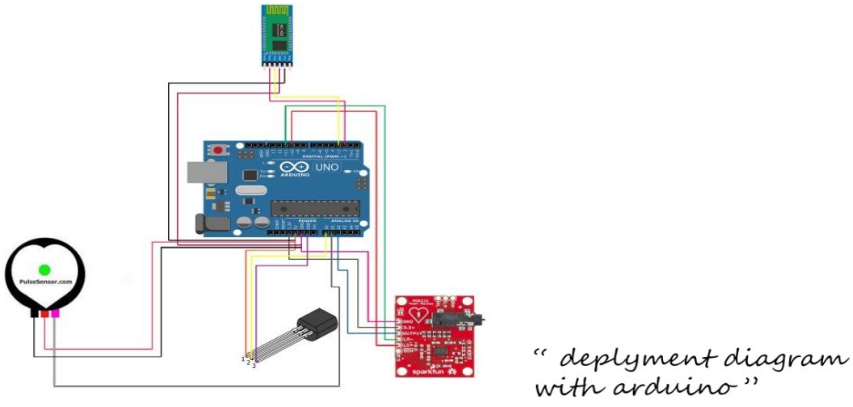

Figure 2.7 Deployment diagram for the system with Arduino

2.6 Sequence diagram

In second scenario, we will determine some diagrams show to us the interactive of components in our system so we begin with the system with Arduino [6]:-

1. User login to the Mobile Application using own account.
2. The account is checked if it is valid or not.
3. Data is acquired by sensors and sent to LabVIEW.
4. LabVIEW displays result data to the users.
5. When entered data is worth, LabVIEW sends alarm to the Mobile Application.

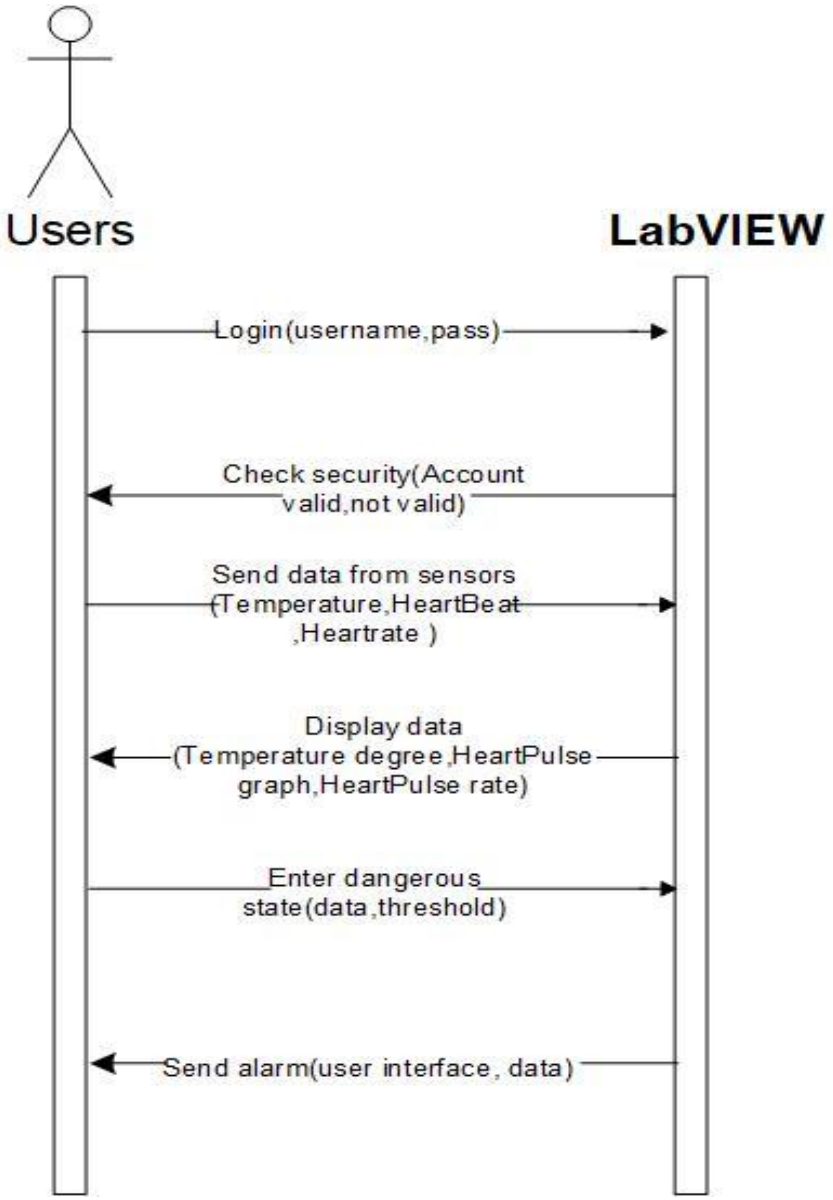

Figure 2.8 Sequence diagram

2.7 System Data flow diagram

In Figure 2.9, determines the data flow from the components of the system, user, and Mobile application. The data flows from the users to the sensors and sent from sensors to microcontroller to process it until it is represented by LabVIEW and if it worth LabVIEW will send alarm to mobile application to alarm user using SMS message [9].

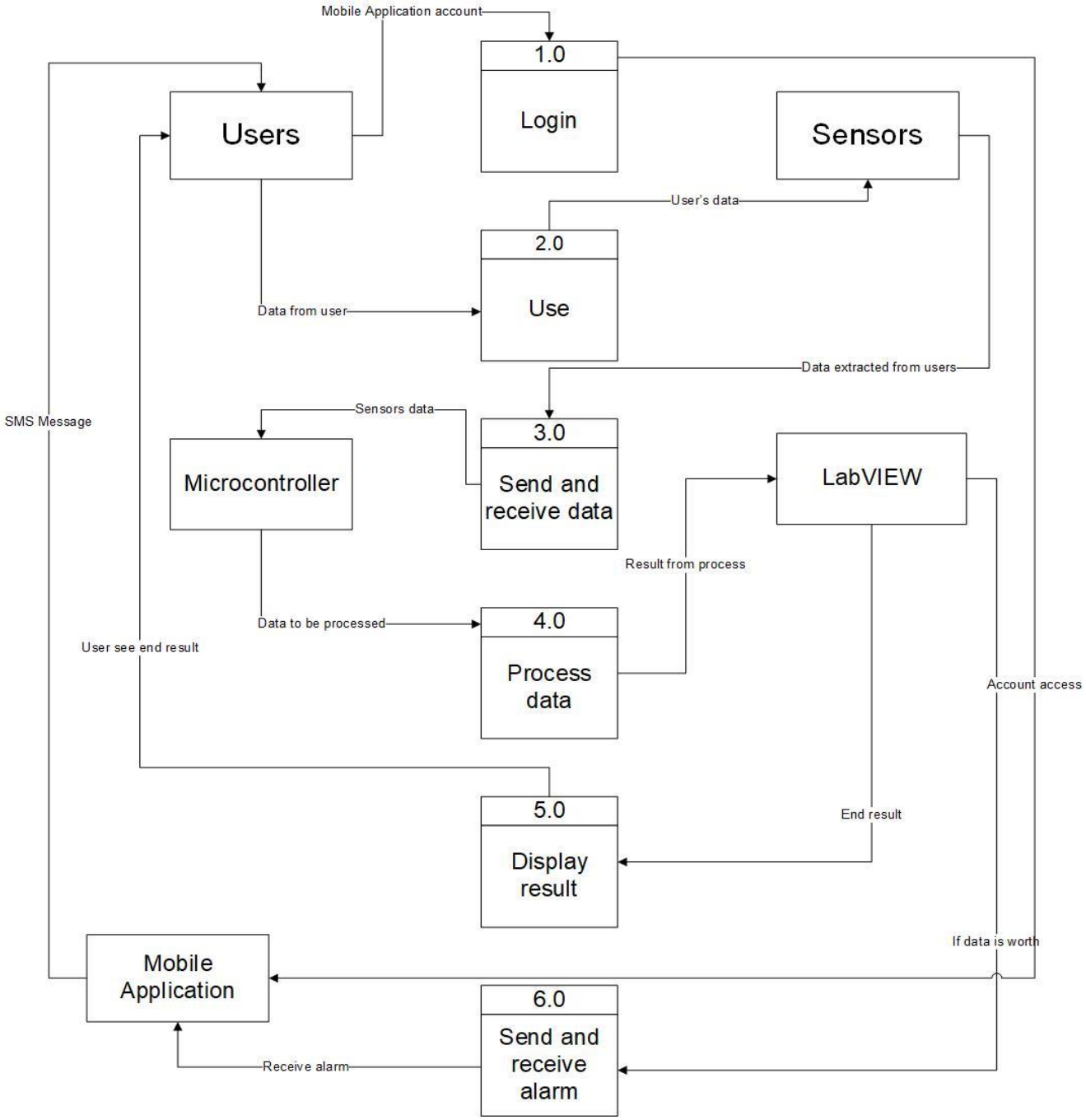

Figure 2.9 Data flow diagram for the system

Chapter 3: Hardware Integration

3.1 System block diagram

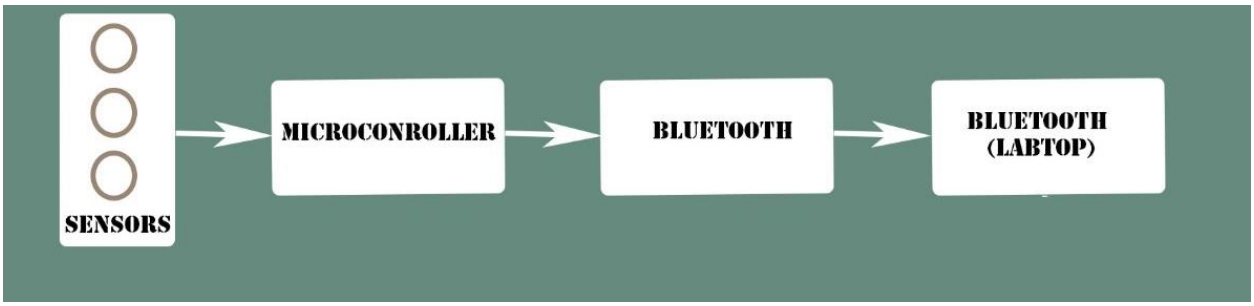

Figure 3.1 system block diagram with Arduino

- All sensors send data to Microcontroller.
- Microcontroller takes this data & sends it to the LabVIEW.
- Bluetooth module is used to send this data to LabVIEW.
- Laptop Bluetooth receives this data.
- LabVIEW displays the information of this data to the patient.

3.2 PIC16F877A

The PIC16F877A features 256 bytes of EEPROM data memory, self-programming, an ICD, 2 Comparators, 8 channels of 10-bit Analog-to-Digital (A/D) converter, 2 capture/compare/PWM functions, the synchronous serial port can be configured as either 3-wire Serial Peripheral Interface (SPI™) or the 2-wire Inter-Integrated Circuit (I²C™) bus and a Universal Asynchronous Receiver Transmitter (USART). All of these features make it ideal for more advanced level A/D applications in automotive, industrial, appliances and consumer applications. This powerful yet easy-to-program PIC® architecture into a 40- or 44-pin package and is upwards compatible with the PIC16C5X, PIC12CXXX and PIC16C7X devices [11].

3.2.1 PICKIT 2

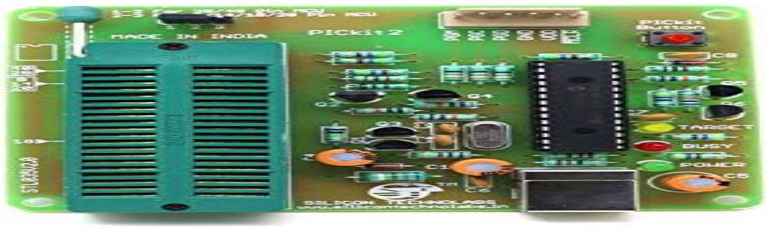

Figure 3.2 PICKIT2

The PIC kit™ 2 Development Programmer/Debugger (PG164120) is a low-cost development tool with an easy to use interface for programming and debugging Microchip's Flash families of microcontrollers. The full featured Windows programming interface supports baseline (PIC10F, PIC12F5xx, and PIC16F5xx), midrange (PIC12F6xx, PIC16F), PIC18F, PIC24, dsPIC30, dsPIC33, and PIC32 families of 8-bit, 16-bit, and 32-bit microcontrollers, and many Microchip Serial EEPROM products. With Microchip's powerful MPLAB Integrated Development Environment (IDE) the PIC kit 2 enables in-circuit debugging on most PIC® microcontrollers. In-Circuit-Debugging runs halts and single steps the program while the PIC microcontroller is embedded in the application. When halted at a breakpoint, the file registers can be examined and modified [10].

3.2.2 PIC16F877A specifications

CPU	8-bit PIC
Number of Pins	40
Operating Voltage (V)	2 to 5.5 V
Number of I/O pins	33
ADC Module	8ch, 10-bit
Timer Module	8-bit(2), 16-bit(1)
Comparators	2

DAC Module	Nil
Communication Peripherals	UART(1), SPI(1), I2C(1), MSSP(SPI/I2C)
External Oscillator	Up to 20Mhz
Internal Oscillator	Nil
Program Memory Type	Flash
Program Memory (KB)	14KB
CPU Speed (MIPS)	5 MIPS
RAM Bytes	368
Data EEPROM	256 bytes

Table 3.1 PIC16F877A Specifications

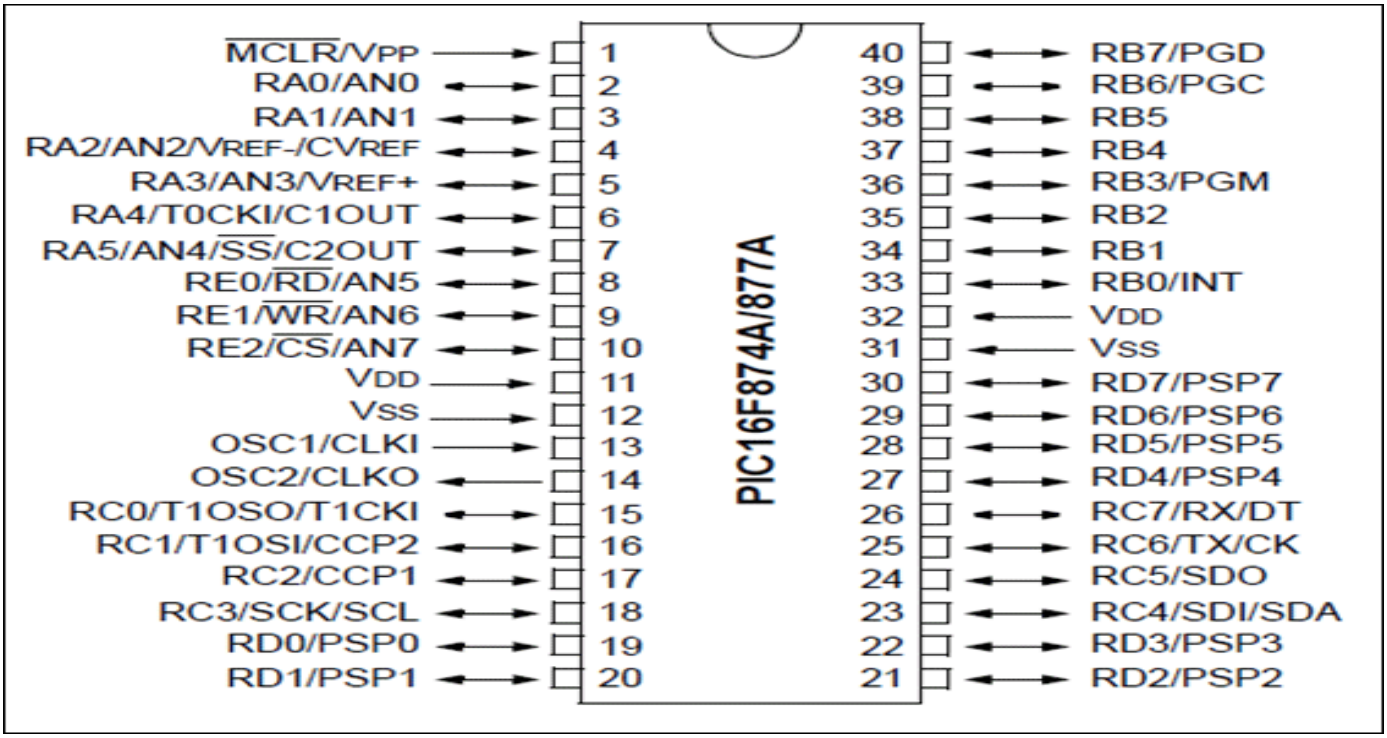

Figure 3.3 pic16F877A layout [12]

3.2.3 PIC16F877A breadboard wiring

PIC 16f877a works in a special environment and can't take voltage directly so to create such environment it needs:

- Breadboard

  A breadboard is a simple device designed to let you create circuits without the need for soldering. It comes in various sizes, and the design can vary, but as a general rule they look something like this the power rails run horizontally as two rows at the top and bottom. Meanwhile, the vertical columns run inwards as you move down the board [14].

- Voltage regulator (5v)

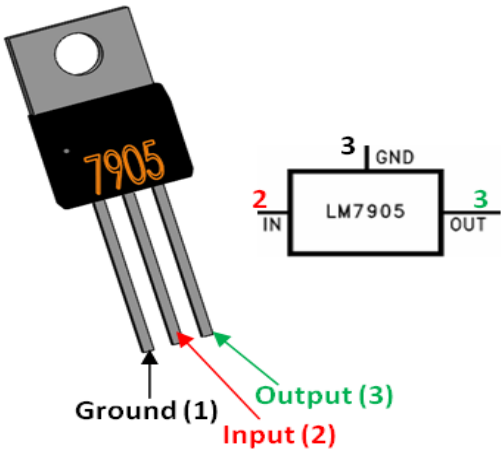

Figure 3.4 5v regulators Layout

Voltage regulators are very common in electronic circuits. They provide a constant output voltage for a varied input voltage. In our case the 7805 IC is an iconic regulator IC that finds its application in most of the projects. The name 7805 signifies two meaning, "78" means that it is a positive voltage regulator and "05" means that it provides 5V as output. So our 7805 will provide a +5V output voltage [15].

- Crystal 4.000MHz

An internal circuit is used, that is called oscillator for generating the device clock, which is required for executing the instruction and peripherals of the function [16].

- Capacitors 22PF

Its specific capacitor is normally used with crystals for loading purposes. Load capacitance is the amount of external circuit capacitance in parallel with the crystal itself [17].

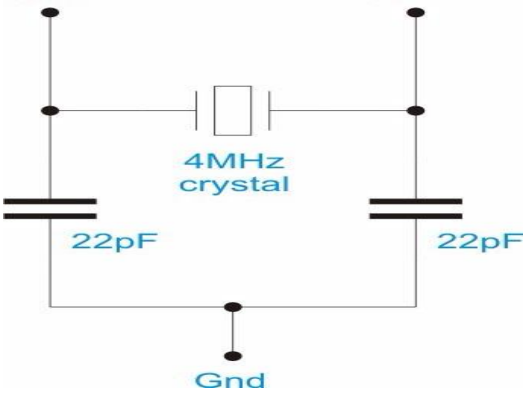

Figure 3.5 connecting crystal capacitors lay out

- Power source
- Jumper wires

3.2.4 PIC Wiring

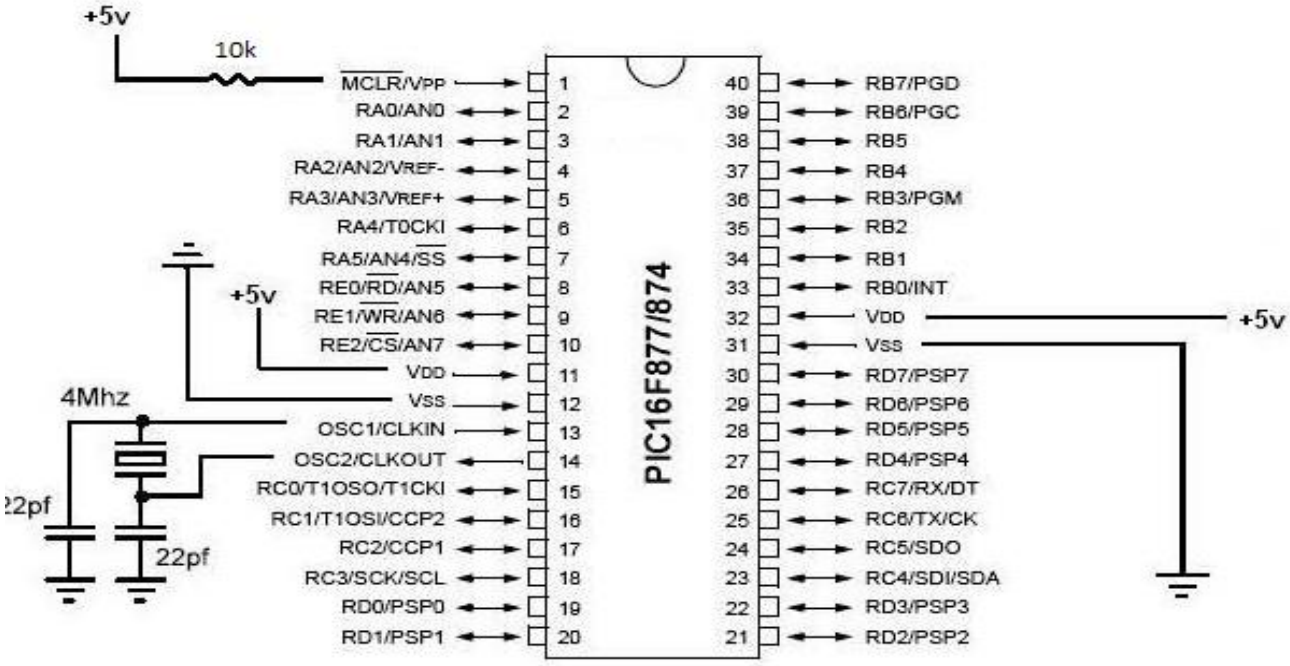

Figure 3.6 PIC wiring Layout

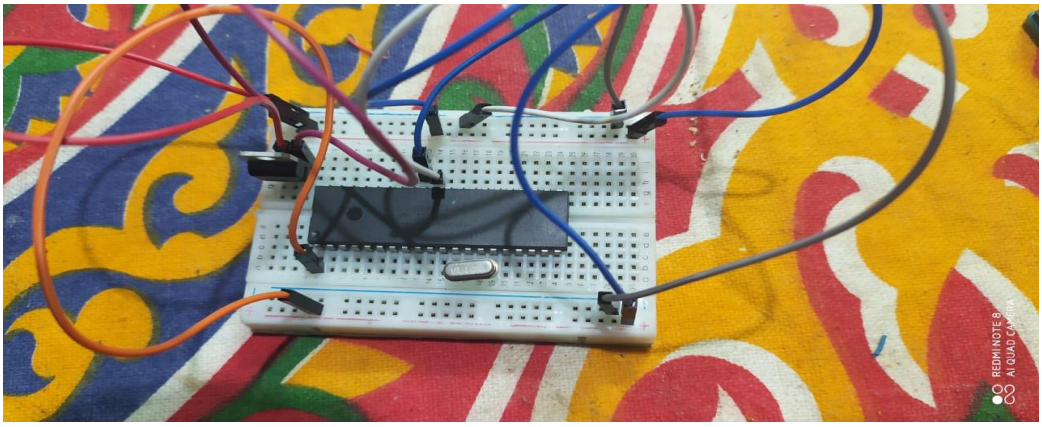

Figure 3.7 live pic wiring

- First connect 10k ohm resistor from +5v to the MCLR (Pin1).
- Connect 4 MHz crystal to Pins (13 and 14).
- Connect 2 capacitors 22pf from GND to the crystal.
- Place the voltage regulator in the breadboard and power it with 9v battery.
- Power up the pic from the VOUT of the regulator (+5v) and the GND to Pins (11, 12 or 32, 31).

3.3 Arduino Uno

Arduino Uno is a microcontroller board based on 8-bit ATmega328P microcontroller. Along with ATmega328P, it contains other components such as crystal oscillator, serial communication, voltage regulator, etc. to support the microcontroller. Arduino Uno has 14 digital input/output pins (out of which 6 can be used as PWM outputs), 6 analog input pins, a USB connection, A Power barrel jack, an ICSP header and a reset button [13].

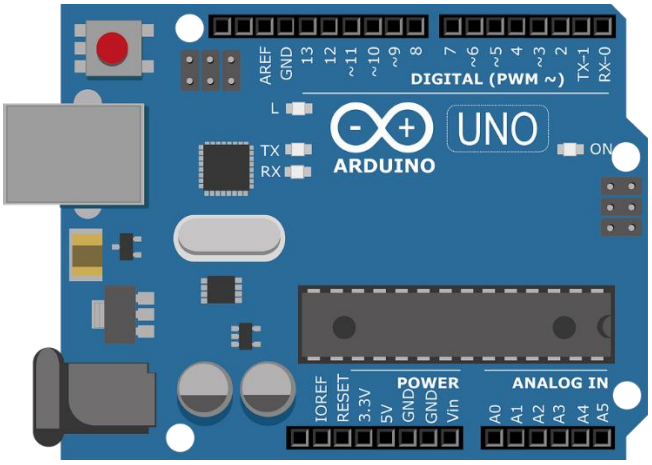

Figure 3.8 Arduino Uno

3.3.1 Arduino Uno Technical Specifications

Microcontroller	ATmega328P – 8 bit AVR family microcontroller
Operating Voltage	5V
Recommended Input Voltage	7-12V
Input Voltage Limits	6-20V
Analog Input Pins	6 (A0 – A5)
Digital I/O Pins	14 (Out of which 6 provide PWM output)
DC Current on I/O Pins	40 Ma
DC Current on 3.3V Pin	50 mA
Flash Memory	32 KB (0.5 KB is used for Bootloader)
SRAM	2 KB
EEPROM	1 KB
Frequency (Clock Speed)	16 MHz

Table 3.2 Arduino Specifications

3.4 Comparison between Arduino and PIC16F877A

Microcontroller	ATmega328P	PIC16F877A
Operating Voltage	5V	2V
Input Voltage	7-12V	2-5.5V
Digital I/O Pins	14 (6 provide PWM output)	33 (2 provide PWM output)
Flash Memory	32 KB	14 KB
SRAM	2 KB	368 B
EEPROM	1 KB	256 B
Clock Speed	16 MHz	5 MHz

Table 3.3 Comparison between Arduino and PIC16F877A based on characteristics [30]

- PIC16F877A is better for this project because it is has less power consumption and less wasted memory
- But we choose Arduino Uno for its hardware and programming simplicity, better power handling and less external noise than PIC16F877A

3.5 Sensors

3.5.1 Temperature sensor (lm35)

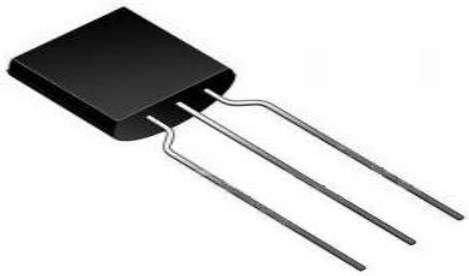

Figure 3.9 lm35 temperature sensor

LM35 is a temperature sensor that outputs an analog signal which is proportional to the instantaneous temperature. The output voltage can easily be interpreted to obtain a temperature reading in Celsius [18].

3.5.1.1 LM 35 Pinout

Figure 3.10 LM35 Pinout

The lm35 has 3 pins:

- **PIN 1:** VCC, it used as input at this pin we apply +5 V input voltage.
- **PIN 2:** At this pin, we get output voltage.
- **PIN 3:** This pin is used for ground [19].

3.5.1.2 LM35 Description

- LM35 is a temperature measuring device having an analog output voltage proportional to the temperature.
- It provides output voltage in Centigrade (Celsius). It does not require any external calibration circuitry.

- The sensitivity of LM35 is 10 mV/ degree Celsius. As temperature increases, output voltage also increases. E.g. 250 mV means 25°C.
- It is a 3-terminal sensor used to measure surrounding temperature ranging from -55 °C to 150 °C.
- LM35 gives temperature output which is more precise than thermistor output [20].

3.5.1.3 LM35 Features

- Calibrated Directly in Celsius (Centigrade)
- Linear + 10-mV/°C Scale Factor
- 0.5°C Ensured Accuracy (at 25°C)
- Rated for Full −55°C to 150°C Range
- Suitable for Remote Applications
- Low-Cost Due to Wafer-Level Trimming
- Operates From 4 V to 30 V
- Less Than 60-µA Current Drain
- Low Self-Heating, 0.08°C in Still Air
- Non-Linearity Only ±¼°C Typical
- Low-Impedance Output, 0.1 Ω for 1-mA Load [21]

3.5.1.4 LM35 Applications

- It's used for measuring the temperature of a particular environment.
- It provides thermal shut down for a circuit or component used in a specific project.
- It can be used for battery temperature measurement.
- It can be used in HVAC applications as a temperature measurement device.

3.5.1.5 Lm35 Interface with Arduino and Pic

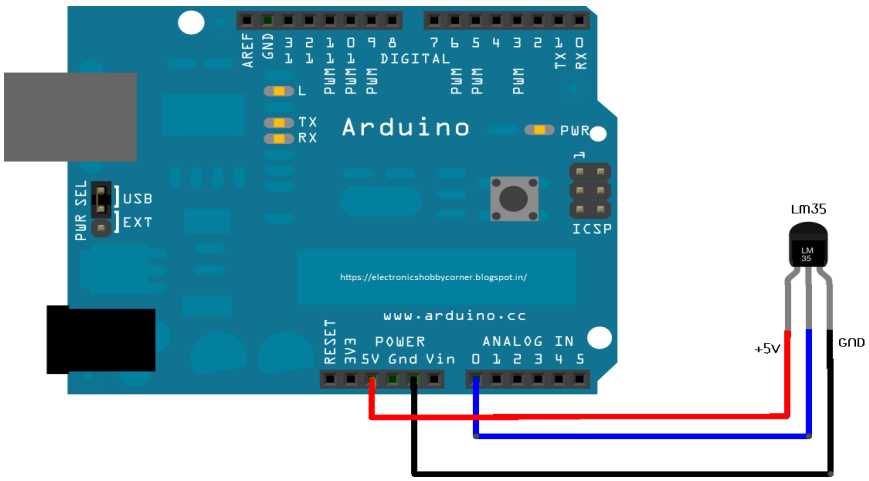

Figure 3.11 lm 35 with Arduino Pinout

Board Label	Pin Function	Arduino Connection
GND	Ground	GND
5v	5v PIN	5v
OUTPUT	Output Signal	A0

Table 3.4 LM35 with Arduino interface

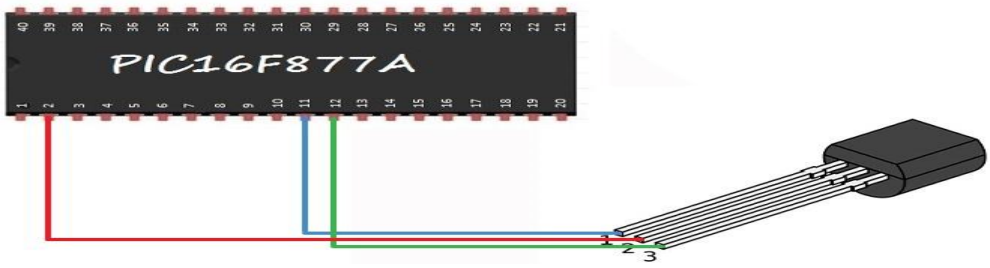

Figure 3.12 lm 35 with PIC Pinout

Board Label	Pin Function	PIC Connection
GND	Ground	PIN 12 GND
5v	5v PIN	PIN 11 5v
OUTPUT	Output Signal	PIN 2 A0

Table 3.5 LM35 with PIC interface

3.5.2 Heart rate sensor

Amped Pulse Sensor is a plug-and-play heart rate sensor for Arduino. It can use by people who want to easily incorporate live heart-rate data into their projects. It essentially combines a simple optical heart rate sensor with amplification and noise cancellation circuitry making it fast and easy to get reliable pulse reading. The visual method (known as optical imaging) measures your heart rate by sensing developments in blood flow through the index finger. The outputs are in the form of pulse waves [22].

Figure 3.13 Heart rate sensor

3.5.2.1 Heart rate Pinout

- **PIN 1**: This pin is used for ground.
- **PIN 2**: VCC, it used as input at this pin we apply +5 V input voltage.
- **PIN 3**: At this pin, we get output voltage.

3.5.2.2 Heart rate description

Coming to the hardware preparation, the hook up is extremely simple as you just need to wire Pulse Sensor's "+" to "5V", "-" to "GND", and "S" to "A0" of microcontroller Uno board. Pin 13 LED will blink with heartbeat. If you want to use pin 13 for something else, adjust the interrupt handler. Sensors can be put on the finger or earlobe, through interconnected line can be connected to the microcontroller [23].

- Pulse rate sensor is used to detect heartbeats.
- It can be wore on the finger or earlobe and connected via cables.
- It also carries an open-source program to display heart rate via diagrams in real time.
- Power supply: 3V~5V.
- It is an integrated optical amplifying circuit and noise eliminating circuit heart rate sensor.

3.5.2.3 Heart rate Features

- Biometric Pulse Rate or Heart Rate detecting sensor
- Plug and Play type sensor
- Operating Voltage: +5V or +3.3V
- Current Consumption: 4Ma
- Inbuilt Amplification and Noise cancellation circuit.
- Diameter: 0.625
- Thickness: 0.125 Thick [24]

3.5.2.4 Heart rate Applications

- Sleep Tracking
- Anxiety monitoring
- Remote patient monitoring/alarm system
- Health bands
- Advanced gaming consoles

3.5.2.5 Heart rate Interface with Arduino and pic

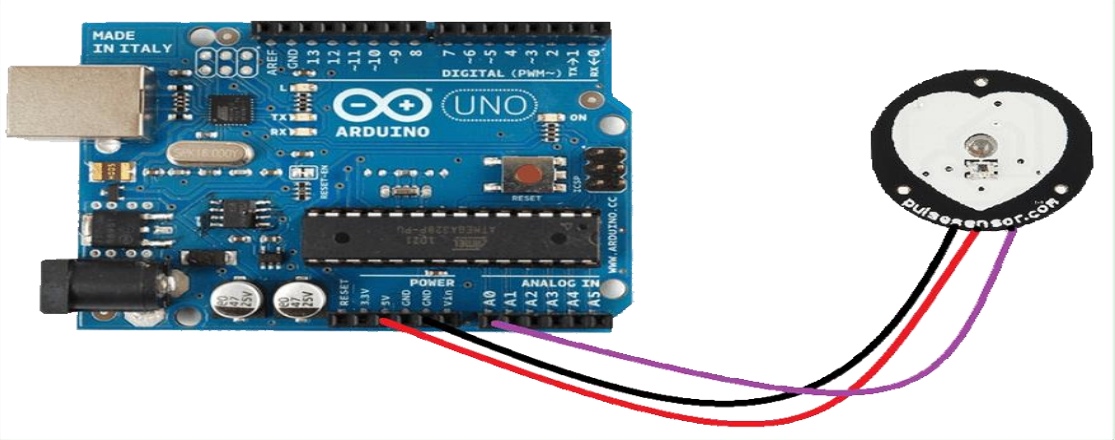

Figure 3.14 heart rate with Arduino Pinout

Board Label	Pin Function	Arduino Connection
GND	Ground	GND
5v	5v PIN	5v
OUTPUT	Output Signal	A0

Table 3.6 heart rate with Arduino interface

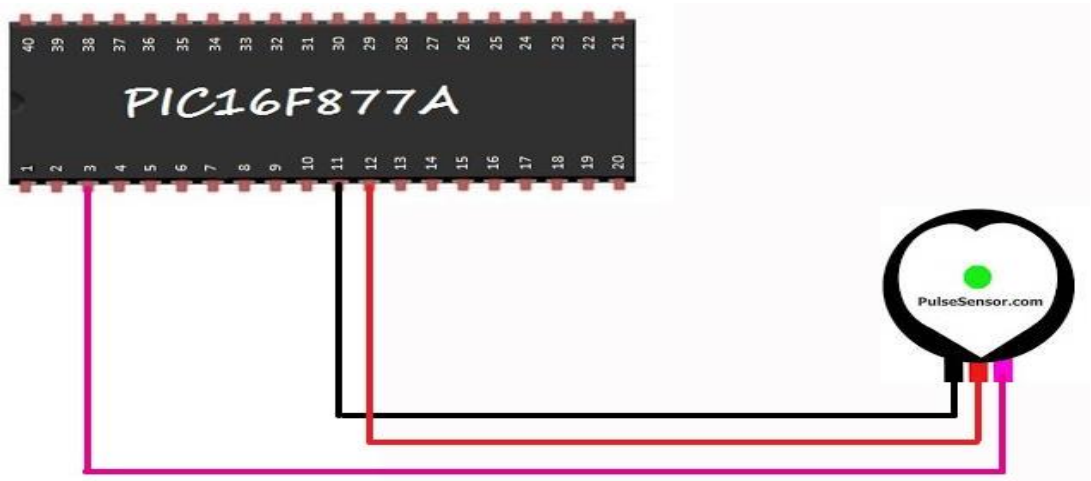

Figure 3.15 heart rate with pic Pinout

Board Label	**Pin Function**	**PIC Connection**
GND	Ground	PIN 12 GND
5v	5v PIN	PIN 11 5v
OUTPUT	Output Signal	PIN 3 A0

Table 3.7 heart rate with PIC interface

3.5.3 ECG AD8232 Heartbeat sensor

The AD8232 is a neat little chip used to measure the electrical activity of the heart. This electrical activity can be charted as an ECG or Electrocardiogram. Electrocardiography is used to help diagnose various heart conditions [26].

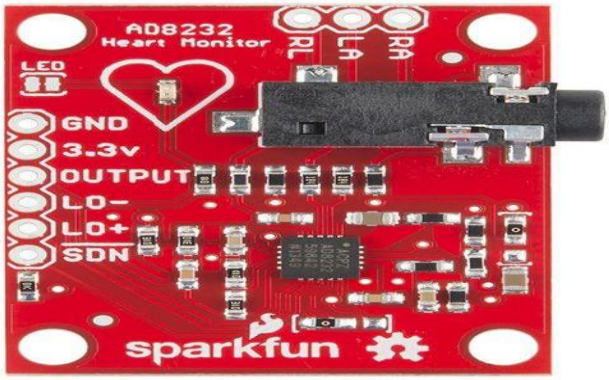

Figure 3.16 ECG sensor

3.5.3.1 ECG Pinout

- **PIN 1:** This pin is used for ground.
- **PIN 2:** VCC, it used as input at this pin we apply +3.3 V input voltage.
- **PIN 3:** At this pin, we get output voltage.
- **PIN 4:** Leads Off Comparator Output. In dc leads off detection mode, LO− is high when the electrode to −IN is disconnected, and it is low when connected
- **PIN 5:** Leads Off Comparator Output. In dc leads off detection mode, LOD+ is high when the +IN electrode is disconnected, and it is low when connected

3.5.3.2 ECG Description

- The AD8232 Heart Rate Monitor breaks out nine connections from the IC. We traditionally call these connections "pins" because they come from the pins on the IC, but they are actually holes that you can solder wires or header pins to.
- We'll connect five of the nine pins on the board to your Arduino. The five pins you need are labeled **GND**, **3.3v**, **OUTPUT**, **LO-**, and **LO+** [25].

3.5.3.3 ECG Features

- Fully integrated single-lead ECG front end
- Common-mode rejection ratio: 80 dB (dc to 60 Hz)
- Two or three-electrode configurations
- Qualified for automotive application
- Single-supply operation: 2.0 V to 3.5
- Fast restore feature improves filter settling
- Size: 3.5cm x 3cm

3.5.3.4 ECG Applications

- Fitness and activity heart rate monitors
- Portable ECG
- Remote health monitors
- Gaming peripherals
- Bio potential signal acquisition

3.5.3.5 ECG Electrodes placement

Now that the electronics are complete, let's look at sensor pad placement. It is recommended to snap the sensor pads on the leads before application to the body.

Figure 3.17 ECG pads

The closer to the heart the pads are, the better the measurement. The cables are color coded to help identify proper placement as shown in the table based on Einthoven's triangle. The sensors

can be placed on the forearms and leg as shown on the diagram on the left. Or they can be placed on the chest near the arms and above the right, lower abdomen (i.e. just above the right hip)

Cable color	Signal
RED	Right side of heart
GREEN	Lower right side
YELLOW	Left side of heart

Table 3.8 ECG Electrodes Placement

3.5.3.6 ECG Interface with Arduino and pic

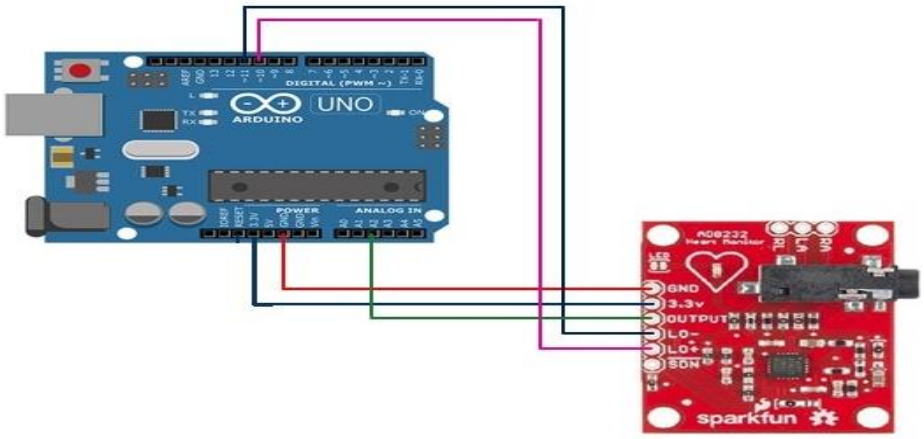

Figure 3.18 ECG with Arduino Pinout

Board Label	Pin Function	Arduino Connection
GND	Ground	GND
3.3v	3.3v PIN	3.3v
OUTPUT	Output Signal	A2
LO-	Leads-off Detect -	11
LO+	Leads-off Detect +	10
SDN	Shutdown	Not used

Table 3.9 ECG with Arduino interface

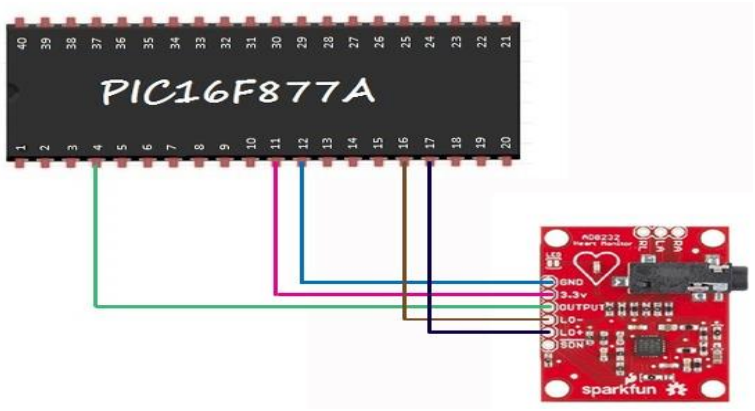

Figure 3.19 ECG with PIC Pinout

Board Label	Pin Function	PIC Connection
GND	Ground	PIN 12 GND
3.3v	5v PIN	PIN 11 5v no 3.3v in pic (problem)
OUTPUT	Output Signal	PIN 4
LO-	Leads-off Detect -	PIN16
LO+	Leads-off Detect +	PIN17
SDN	Shutdown	Not used

Table 3.10 ECG with PIC interface

3.6 Communication devices

The ZigBee module is better for this project and despite the presence of many technologies such as Bluetooth, ZigBee and Cloud.....etc. We have chosen the Bluetooth module Instead of ZigBee module because ZigBee module is not available in the country.

3.6.1 ZigBee module

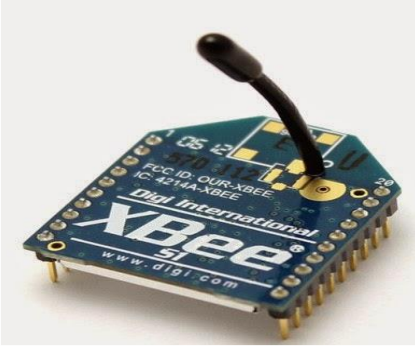

Figure 3.20 ZigBee module

ZigBee is a technology of data transfer in wireless network and a specification for a suite of high level communication protocols using small low-power digital radio based on the IEEE 802.15.4-2003 standard for wireless personal area networks (WPANs) where technological standard created for control and sensor network. ZigBee is designed for wireless automation and other lower data tasks such as smart home automation and remote monitoring and created by ZigBee Alliance [28].

3.6.1.1 ZigBee features

- Transmission Frequency: 2.4GHz to 2.5GHz
- Number of Channels: 16 Direct Sequence Channels
- Featured with UART (250 Kb/s maximum) and SPI (5 Mb/s maximum) interface
- Featured with software adjustable transmitting power
- Indoor/Urban Range: 200ft
- Outdoor RF line-of-sight Range: up to 4000ft
- Transmit Power Output: 6.3mW (8dBm) in Boost mode, 2mW (3dBm) in Normal mode
- RF Data Rate: 250,000 bps
- Receiver Sensitivity: -102dBm in Boost mode, -100dBm in Normal mode
- Supply Voltage Range: +2.1V to +3.6V
- Operating Current: 33mA (at3.3V, for Normal mode), 45mA (at 3.3V, for Boost mode)
- Idle Current: 9mA
- Maximum output current on all pins together: 40mA
- Power-down current: <1uA @25C
- ESD protection: 3000V

- Operating Temperature: -40ºC to 85° C
- ZigBee uses the IEEE 802.15.4 PHY and MAC to allow networks to handle with any number of devices

3.6.1.2 ZigBee Applications

- Home automation
- Commercial building automation
- Industrial application control
- Smart energy
- Medium range wireless communication

3.6.1.3 ZigBee interface with Arduino and Pic

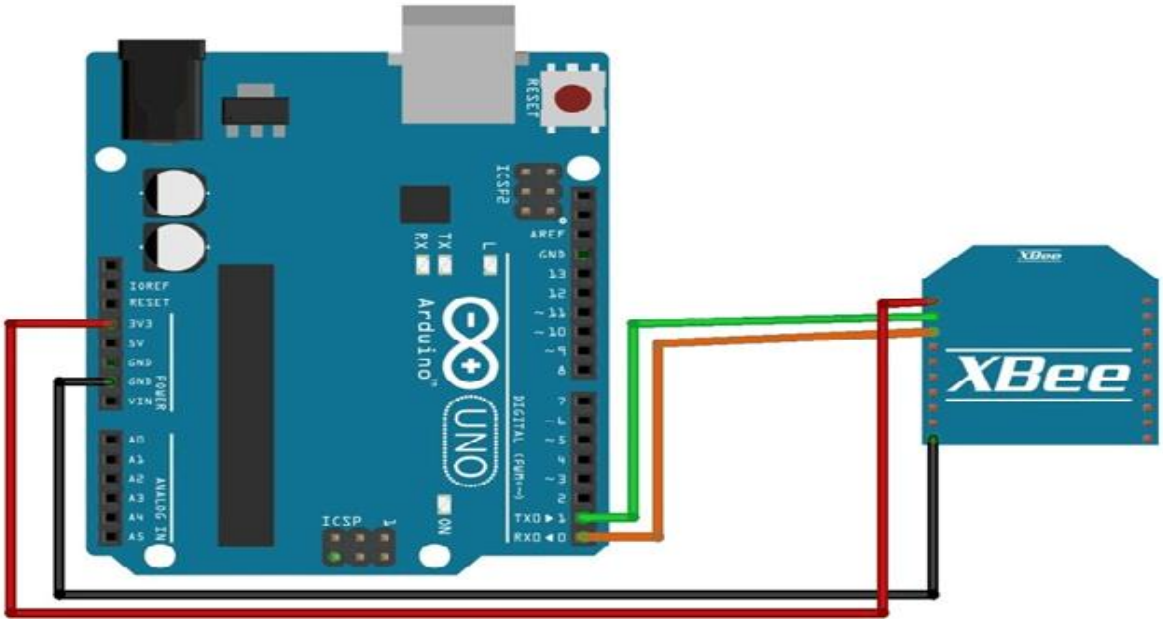

Figure 3.21 ZigBee with Arduino Pinout

Board Label	Pin Function	Arduino Connection
GND	Ground	GND
3.3v	3.3v pin	3.3v
TX	Transmit	RX
RX	Receive	TX

Table 3.11 ZigBee with Arduino interface

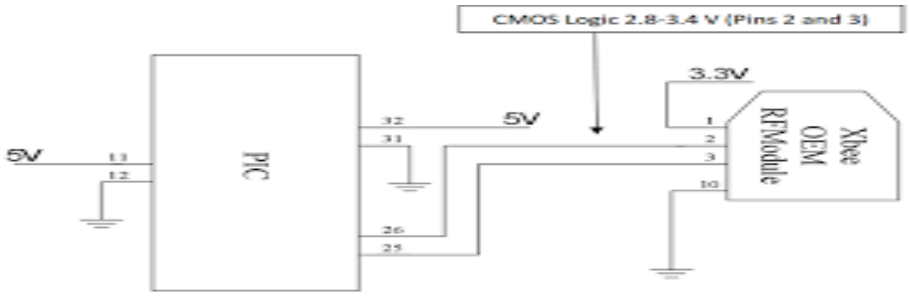

Figure 3.22 ZigBee with PIC Pinout

Board Label	Pin Function	PIC Connection
GND	Ground	Pin 12 GND
3.3v	3.3v pin	Pin11 5v no 3.3v in pic (problem)
TX	Transmit	RX Pin 26
RX	Receive	TX Pin 25

Table 3.12 ZigBee with PIC interface

3.6.2 Bluetooth Module (HC-05):

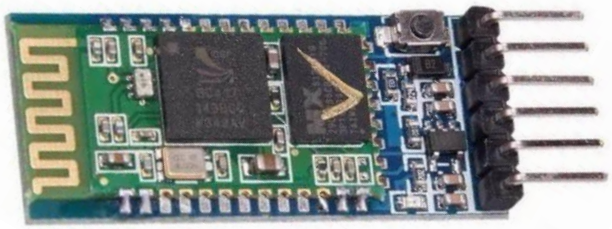

Figure 3.23 HC-05

Bluetooth is a type of wireless communication protocol used to send and receive date between two devices. These small modules run on 3.3V power with 3.3V serial signal levels. The communication range is approximately 9 Meters. It operates on 2.4 GHz frequency. We will be using the HC-05 Bluetooth module which communicates with microcontrollers over the serial USART bus and The Bluetooth module HC-05 is MASTER/SLAVE module. The Role of the module (Master or Slave) can be configured only by AT COMMANDS [29].

3.6.2.1 HC-05 features

- Serial Bluetooth module for Arduino and other microcontrollers
- Operating Voltage: 4V to 6V (Typically +5V)
- Operating Current: 30mA
- Range: <100m
- Works with Serial communication (USART) and TTL compatible
- Follows IEEE 802.15.1 standardized protocol
- Uses Frequency-Hopping Spread spectrum (FHSS)
- Can operate in Master, Slave or Master/Slave mode
- Can be easily interfaced with Laptop or Mobile phones with Bluetooth
- Supported baud rate: 9600,19200,38400,57600,115200,230400,460800.

3.6.2.2 HC-05 applications

- Wireless communication between two microcontrollers
- Communicate with Laptop, Desktops and mobile phones
- Data Logging application
- Consumer applications
- Wireless Robots
- Home Automation
- Short range wireless communication

3.6.2.3 HC-05 Interface with Arduino and PIC16F877A

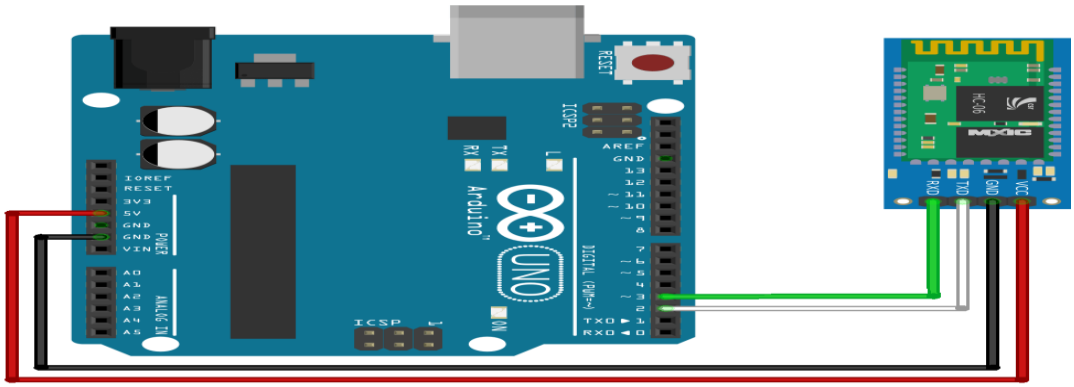

Figure 3.24 HC-05 with Arduino Pinout

Board Label	Pin Function	Arduino Connection
GND	Ground	GND
3.3v	3.3v pin	3.3v
TX	Transmit	RX
RX	Receive	TX

Table 3.13 HC-05 with Arduino interface

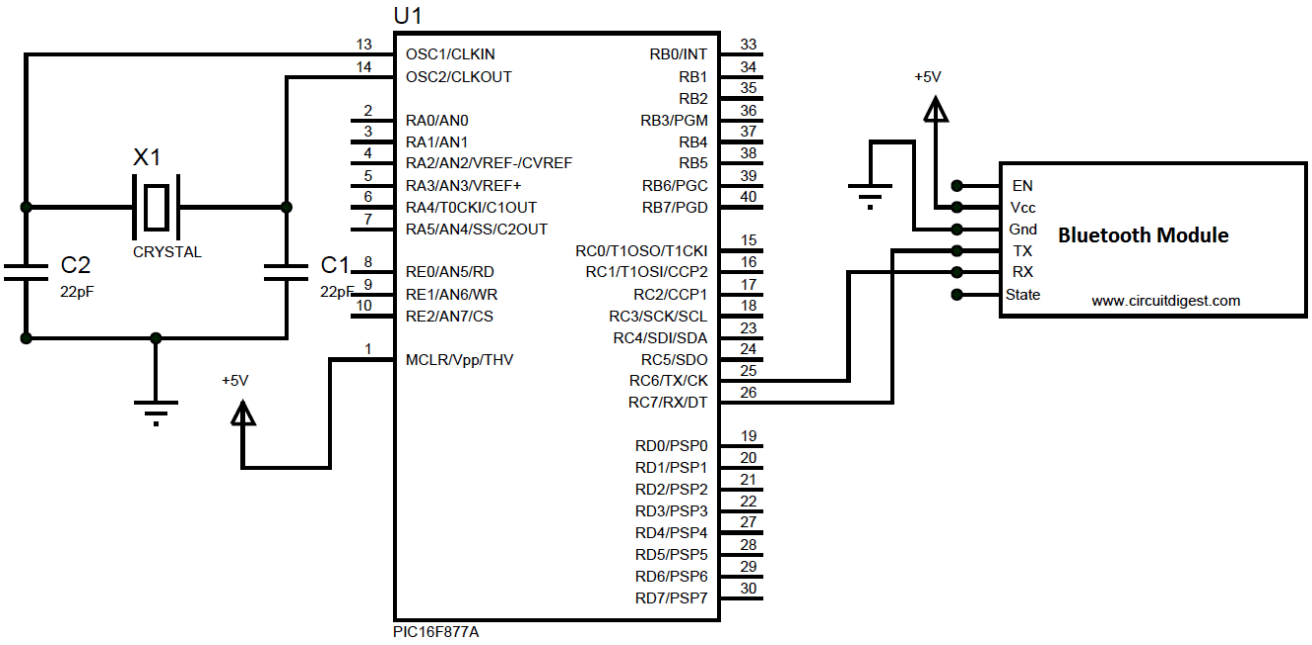

Figure 3.25 HC-05 with PIC Pinout

Board Label	Pin Function	PIC Connection
GND	Ground	Pin 12 GND
3.3v	3.3v pin	Pin11 5v no 3.3v in pic (problem)
TX	Transmit	RX Pin 26
RX	Receive	TX Pin 25

Table 3.14 HC-05 with PIC Interface

3.7 Comparison between ZigBee module and Bluetooth module

Characteristics	ZigBee	Bluetooth
Range	10-100 m	2-10 m
Data Rate	20-250 kbps	I Mbps
Simplicity	High	Medioum
Time needed for a node to join the network	30 ms	20 sec
Transition of a node from sleep to active mode	l5ms	3 sec
Time anode uses the channel	15 ms	2 ms
Battery life	Years	Days
Security	AES (128 bits)	4LFSR (64, 128 bits)
Network topology	Star, Peer to Peer, Mesh	Piconet
Maximum number of devices in a network	65,000	8
ISM frequency	868 MHZ, 915 MHz, 2.4, GHz	2.4 GHz
Cost	Low	Medium

Table 3.15 Comparison of Bluetooth module and ZigBee module based on characteristics [27]

Chapter 4: Software

4.1 LabVIEW

LabVIEW is a system-design platform and development environment for a visual programming language from National Instruments. The graphical language is named "G".

Originally released for the Apple Macintosh in 1986, LabVIEW is commonly used for data acquisition, instrument control, and industrial automation on a variety of operating systems (OSs), including Microsoft Windows, various versions of Unix, Linux, and macOS. The latest versions of LabVIEW are LabVIEW 2019 and LabVIEW NXG 3.1, released in May 2019.

With regard to show a reading of sensors, we used Arduino and LabVIEW to show the data and waves of the sensors readings. Arduino is responsible for data conversion and data transfer. LabVIEW is data acquisition software to realize communication and show the readings

Arduino: is one of the cost-effective boards you can purchase and start programming to interact physical elements with your logic and create results the way you want. by using LabVIEW, you can display the various process parameters like Motor Speed, LED ON/OFF Feedback, Temperature Signal and many others digital or analogue processes. Moreover, you can create unlimited Buttons, Switches and Message Box to control the Arduino [31].

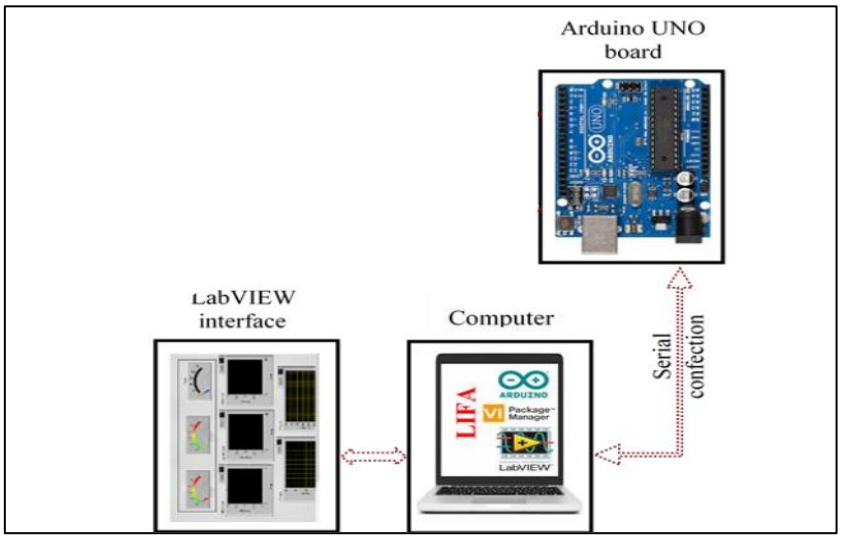

Figure 4.1 interface between LabVIEW and Arduino

To make interface between LabVIEW and Arduino Follow the following steps:

- First, download The NI-VISA driver.
- install the driver.
- the NISA driver will automatically create a tool in with the name of COM port as shown in VI below.
- this tool will automatically start appearing in the LabVIEW components window.
- simply run the LabVIEW and following window will appear

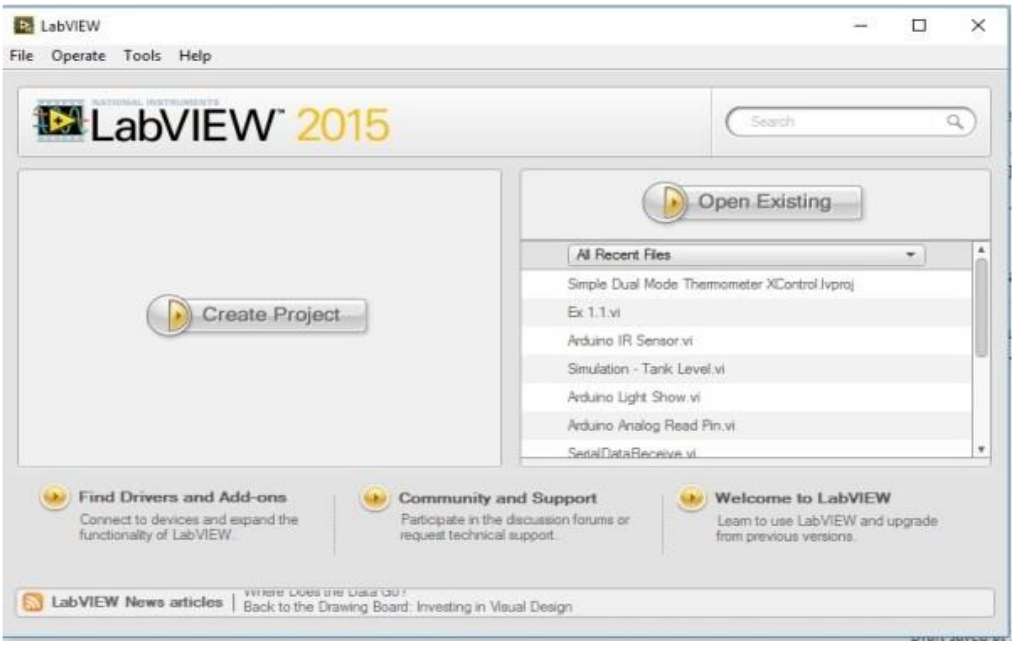

Figure 4.2 first window in LabVIEW

To make VI click on create the project after that following window will appear

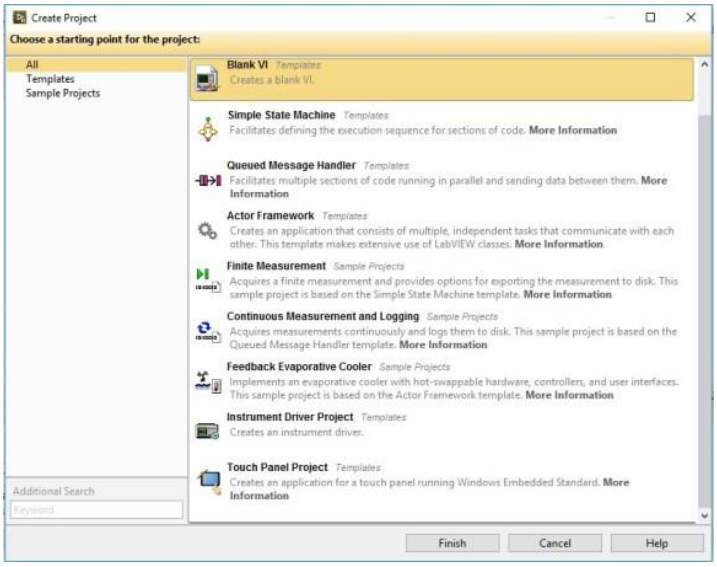

Figure 4.3 create new project in LabVIEW

Now select blank VI and click on Finish. When clicking on finish, a blank VI will open, which we used to create our project.

LabVIEW has two types of consoles. One is called block diagram where placing all components and the other is called the front window. The front window is used as a graphical user interface and it is where users get their data and make changes in data. Block diagram and the front panel are shown below:

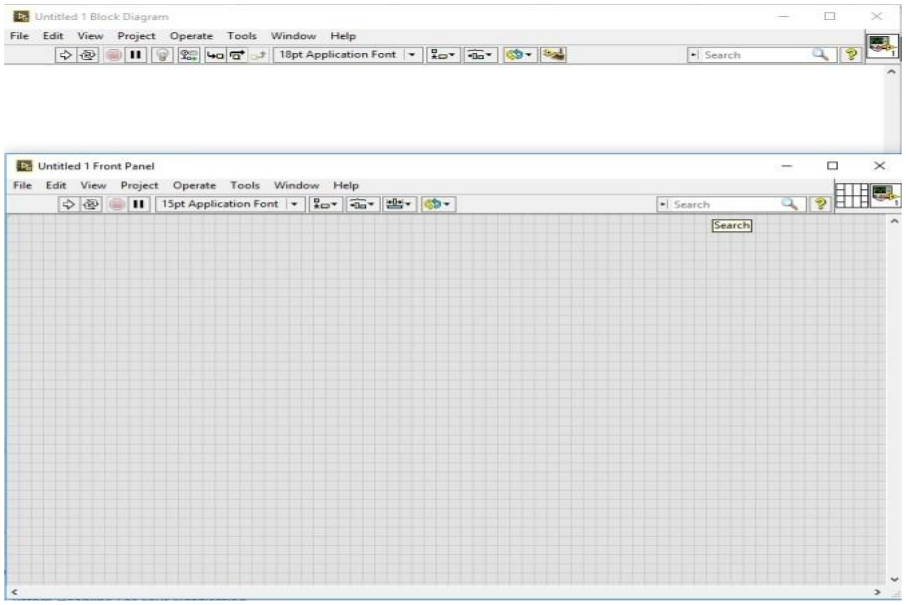

Figure 4.4 design page in LabVIEW “Front panel” & “Block diagram”

In this VI use VISA resources to get data serially. I have designed a VI which is shown below. In this VI getting data from a serial port. Arduino will be connected to the computer serial port and the date received from Arduino will be displayed in the response window.

VI which I made in LabVIEW is shown below.

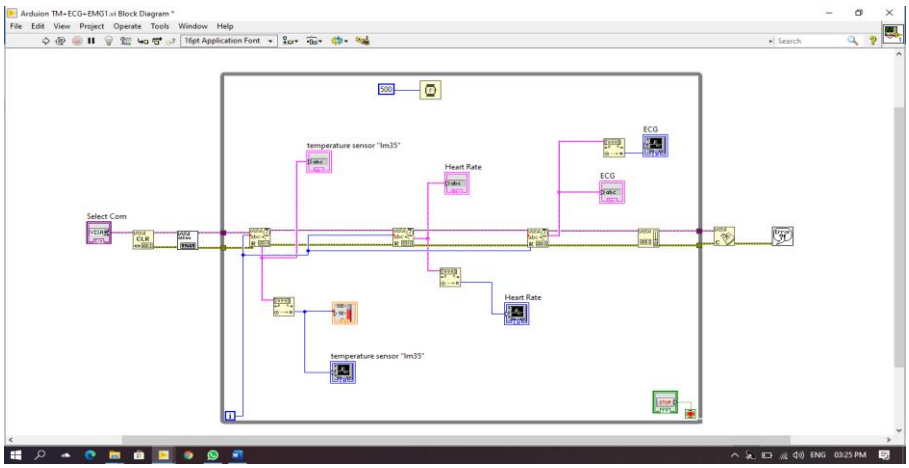

Figure 4.5 code three sensors in block diagram

Connect visa resource with visa clear to clear any previous input or output data after that connect it with visa open to open the port and show which port is open and visa read to receive data from Arduino or pic then LabVIEW read the data that received as a string , so should convert it to number to display as a wave.

You need another visa read for the second sensor and also repeat the previous connector to convert the data to the number to display as a wave.

This VI will using Loop to get data continuously from Arduino. The response component is used to display data.

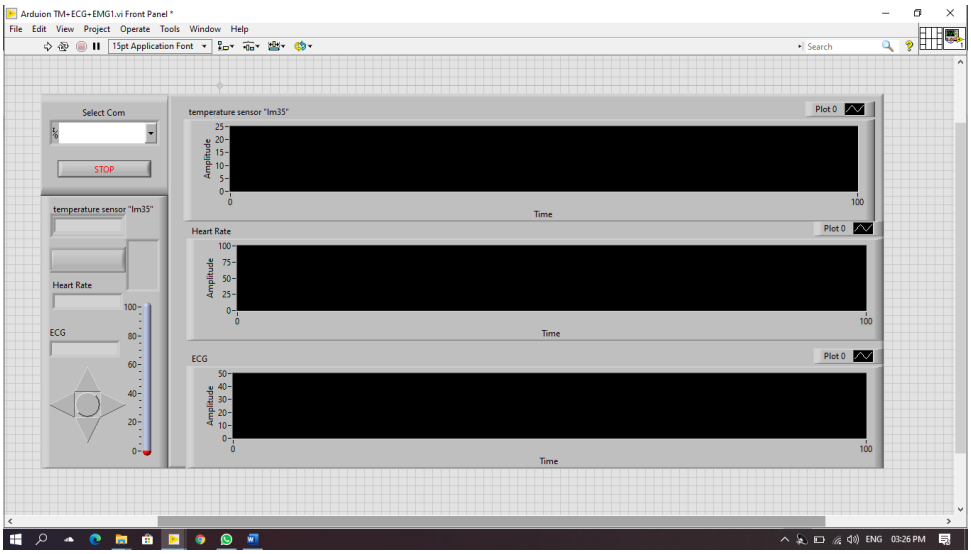

Figure 4.6 code three sensors in front panel

Select COM port to which Arduino is connected. In my case, it is connected with COM4 port. So, selected the COM4 port.

After that click on start reading and you will be getting a date in response window.

To stop simulation just click on stop simulation and you will stop getting data on response window.

4.2 Proteus

The Proteus Design Suite: is a proprietary software tool suite used primarily for electronic design automation. The software is used mainly by electronic design engineers and technicians to create schematics and electronic prints for manufacturing printed circuit boards [32].

4.2.1 Micro-controller simulation in Proteus

The micro-controller simulation in Proteus works by applying either a hex file or a debug file to the microcontroller part on the schematic. It is then co-simulated along with any analog and digital electronics connected to it. This enables its use in a broad spectrum of project prototyping in areas such as motor control, temperature control and user interface design. It also finds use in the general hobbyist community and, since no hardware is required, is convenient to use as a training or teaching tool. Support is available for co-simulation of:

- Microchip Technologies PIC10, PIC12, PIC16, PIC18, PIC24, dsPIC33 Microcontrollers.
- Atmel AVR (and Arduino), 8051 and ARM Cortex-M3 Microcontrollers
- NXP 8051, ARM7, ARM Cortex-M0 and ARM Cortex-M3 Microcontrollers.
- Texas Instruments MSP430, PICCOLO DSP and ARM Cortex-M3 Microcontrollers.
- Parallax Basic Stamp, Freescale HC11, 8086 Microcontrollers.

4.2.2 Using proteus to simulate temperature sensor “lm35”

1. Open Proteus
2. Click on file ----> new project

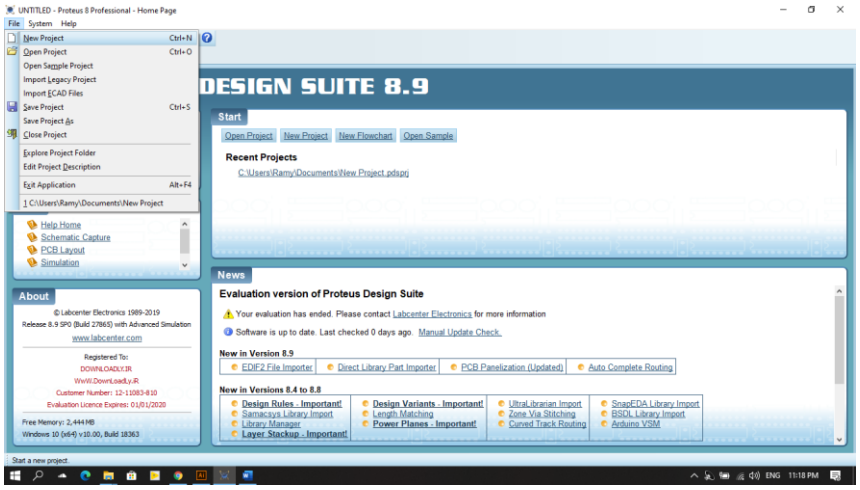

Figure 4.7 create new project in proteus

3. Write the name for your project and choose the location you will save the project Click next --> next --> next --> next --> next --> finish

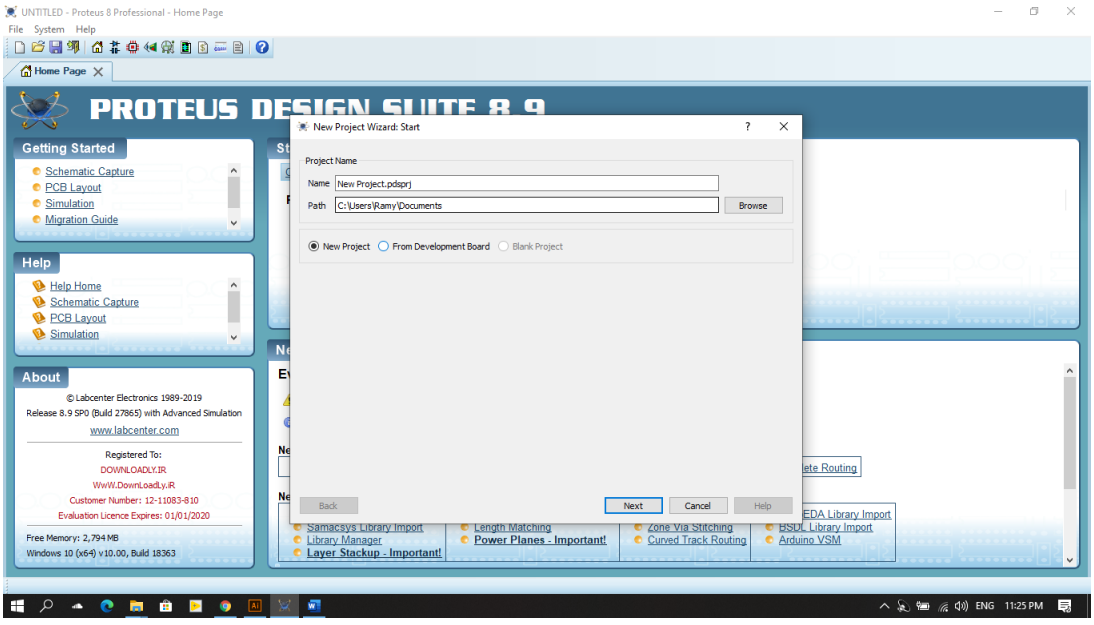

Figure 4.8 write name of project and choose the location you will save the project

4. To add the device you will use

   Click (P) “pick devices”

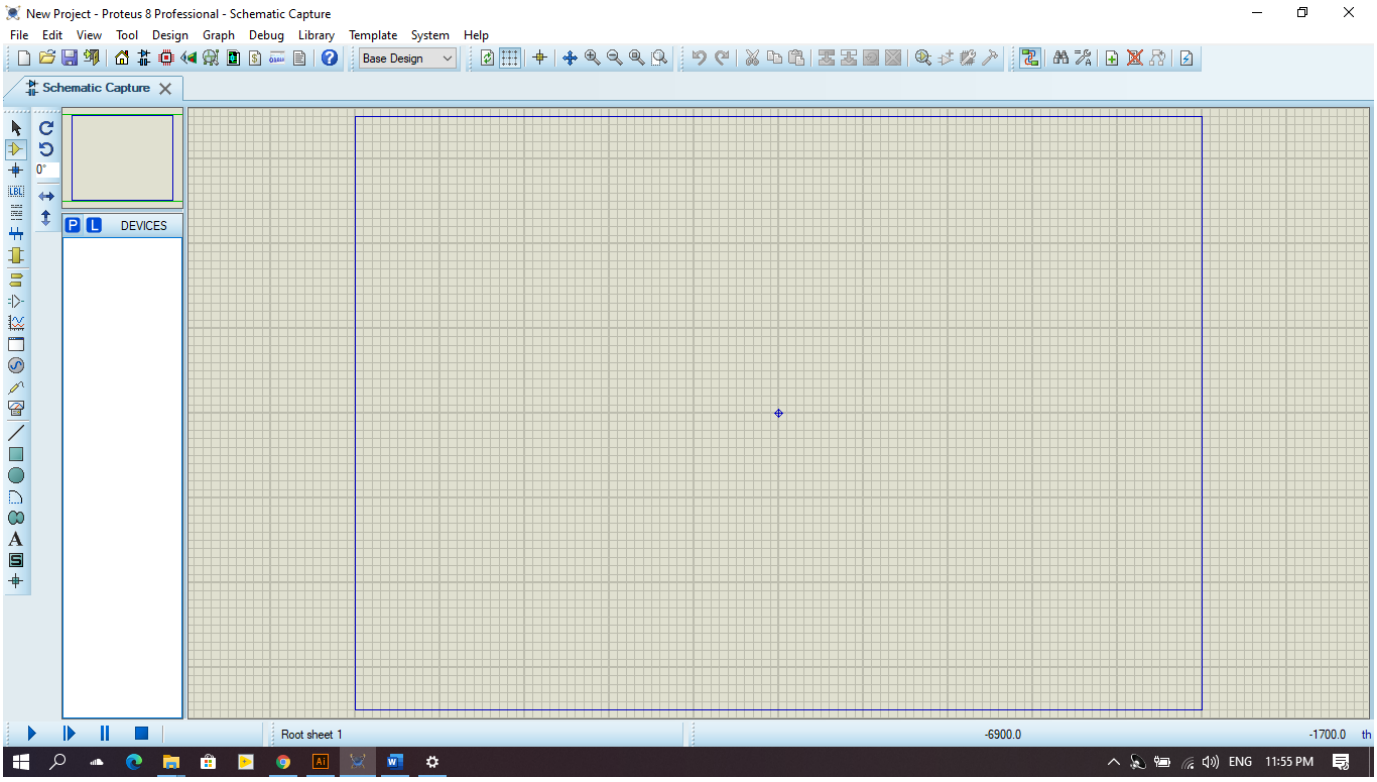

Figure 4.9 adding the device

1. Write the name of the device you will use and enter OK (EX: - pic 16F877A)

   And again, to choose a sensor (EX: - LM35 Temperature Sensor)

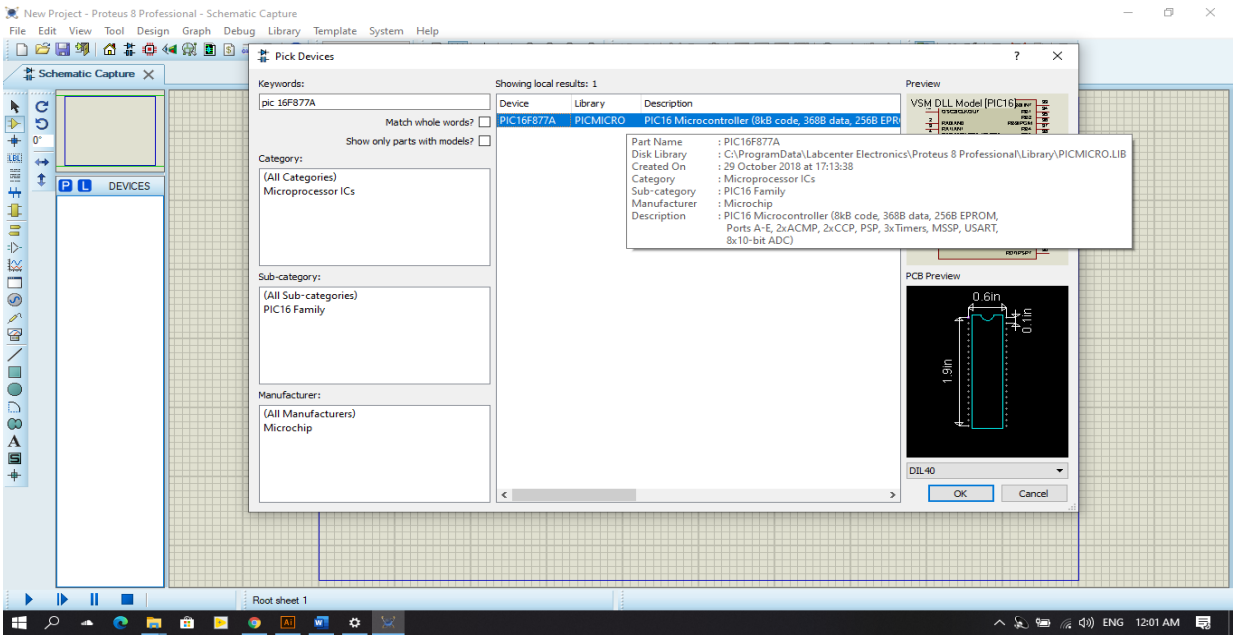

Figure 4.10 searching for device

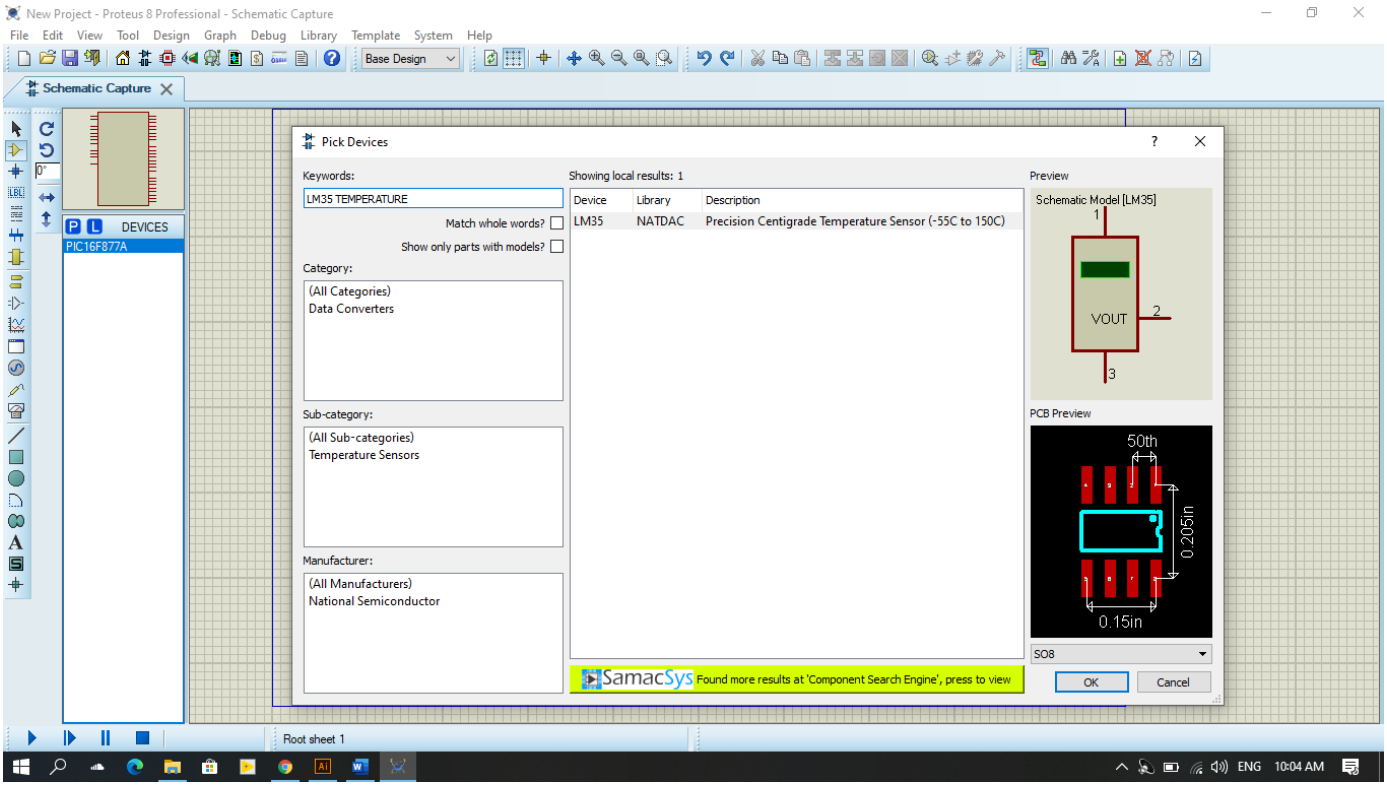

Figure 4.11 searching for device

2. Click on your device (pic 16F877A) and double click on free space to put a device

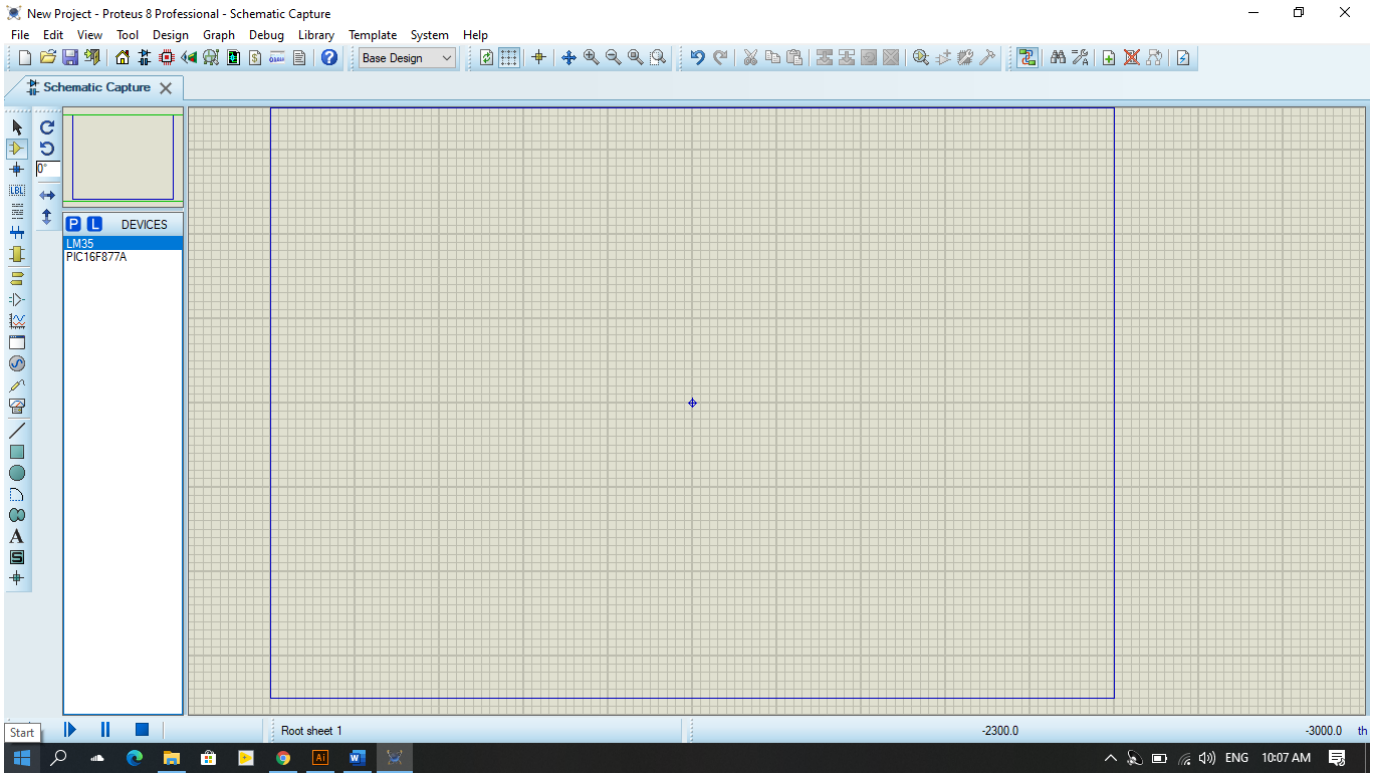

Figure 4.12 select the device to put in free space

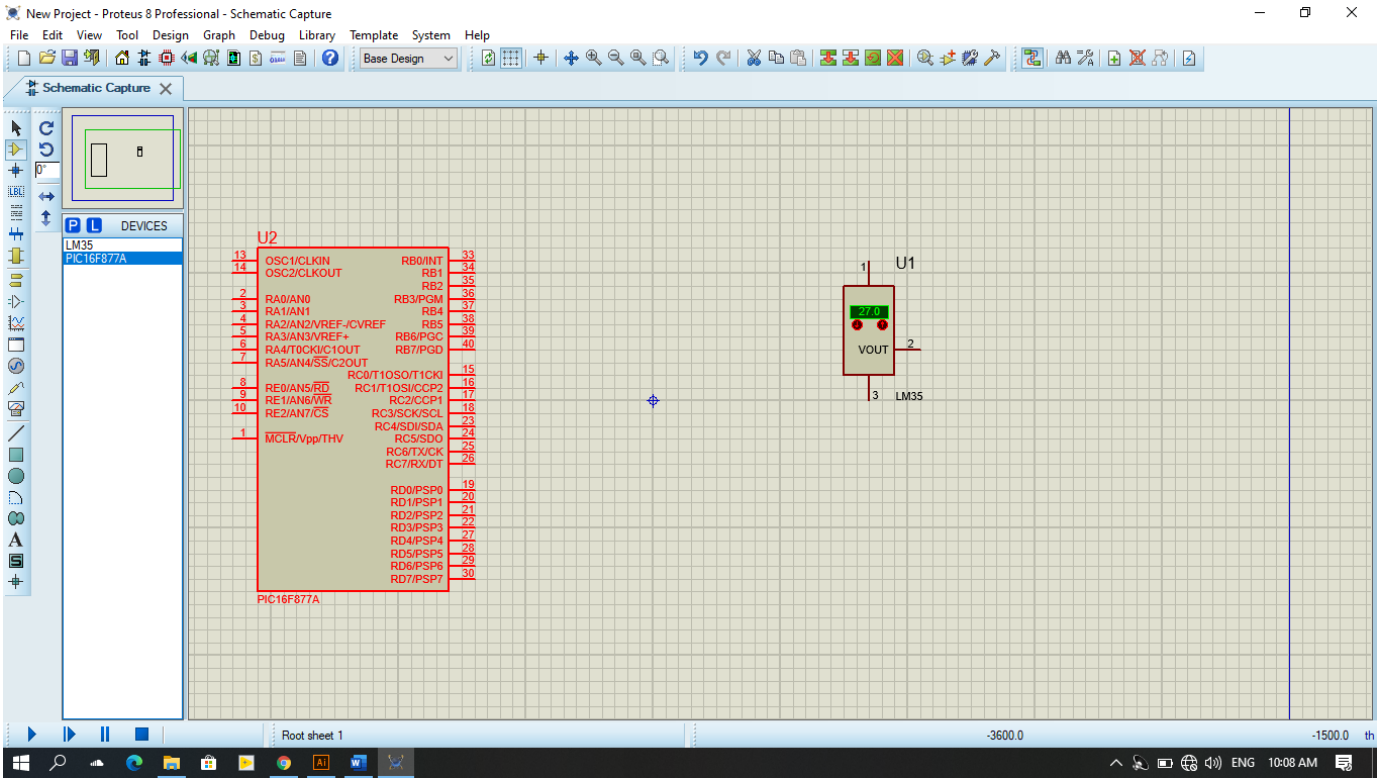

Figure 4.13 put the device in free space

3. To add “Ground” click on terminals mode and choose “Ground” and double click in free space

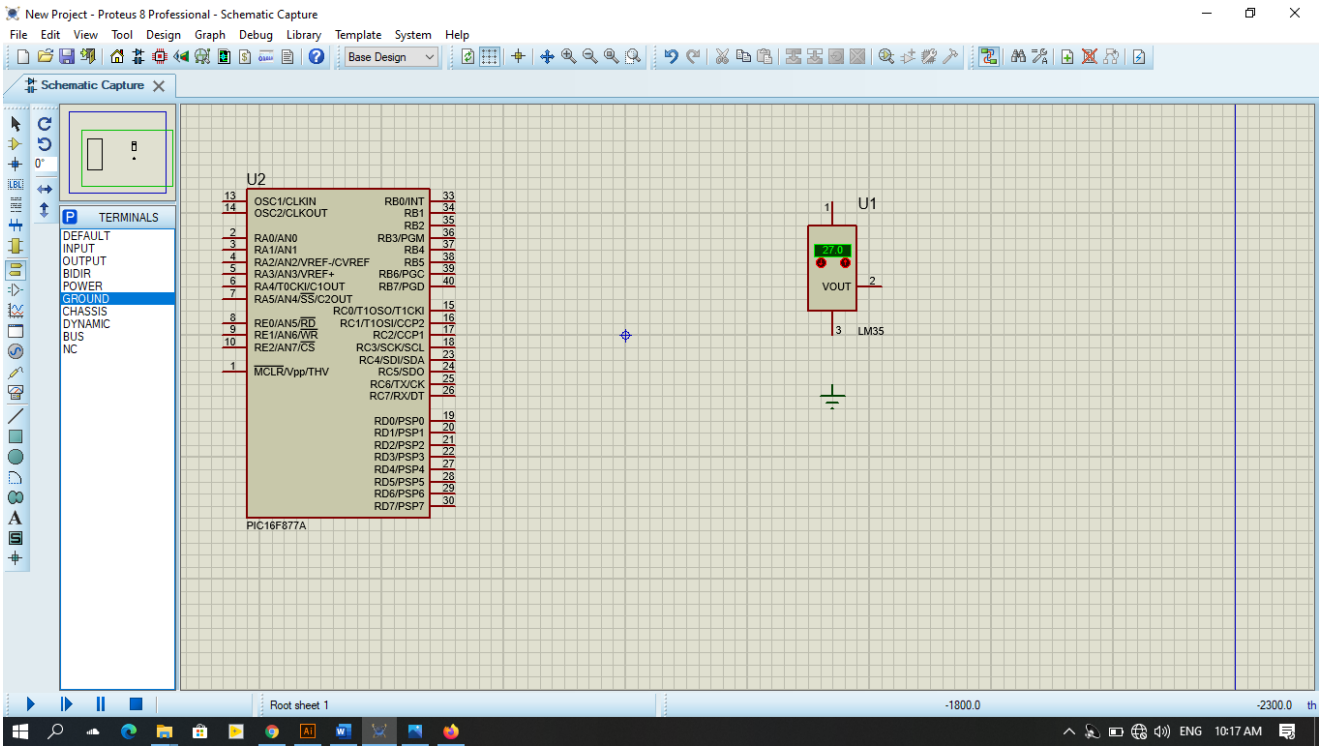

Figure 4.14 add “GROUND”

4. To add “DC” click on Generator mode and choose “DC” and double click in free space

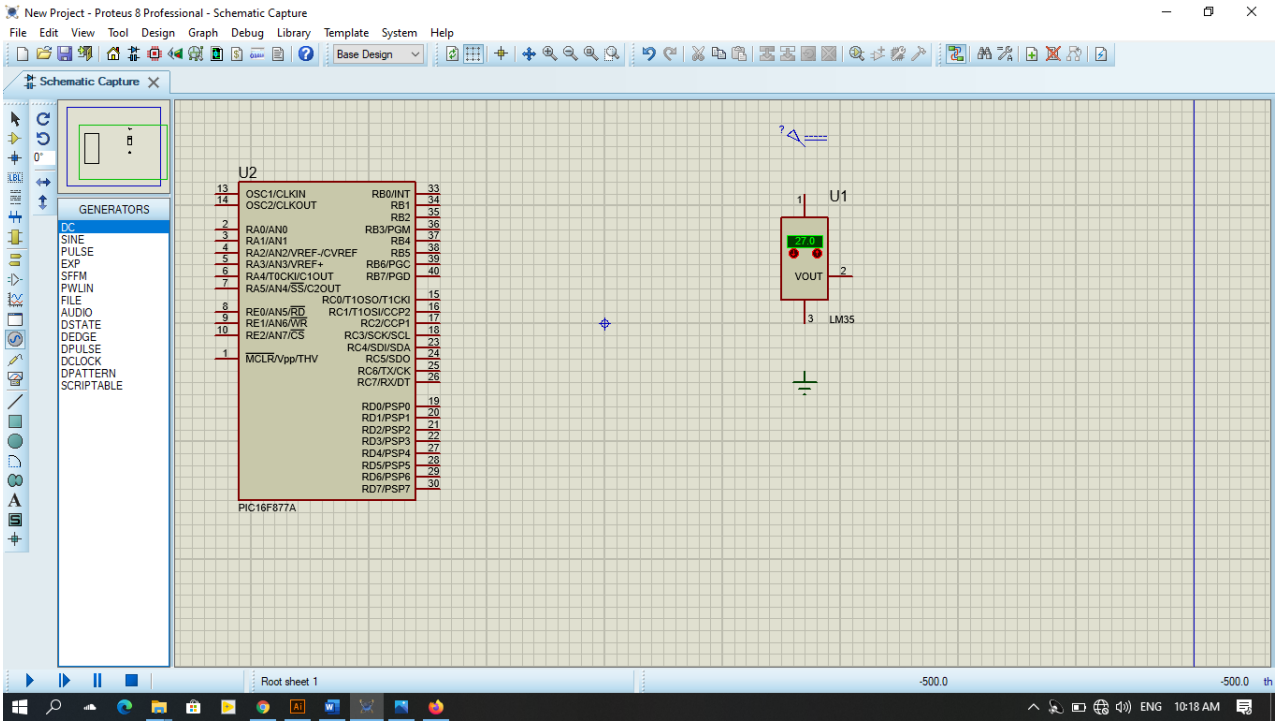

Figure 4.15 add “DC”

5. Click on “DC” and write name of DC “Ex: - voltage” and choose the number of voltages you are needed in your device

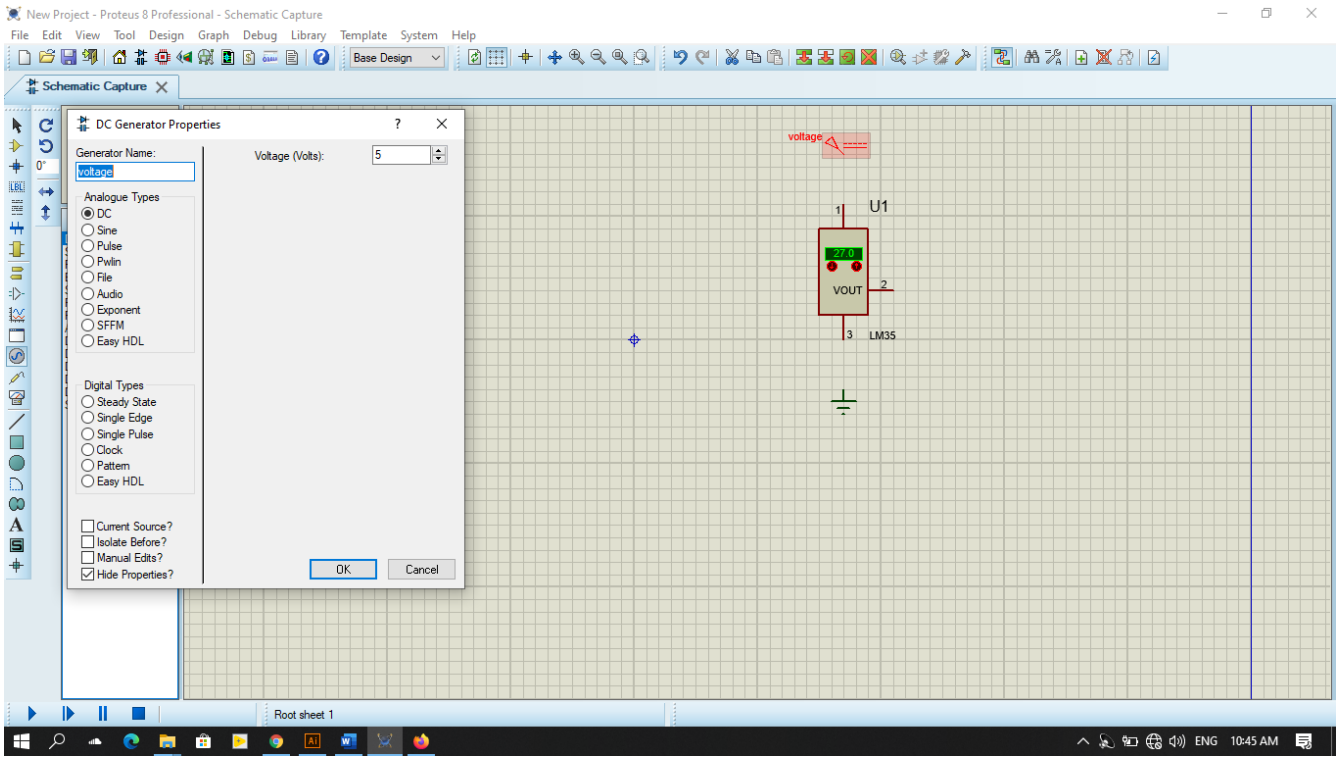

Figure 4.16 select the num of voltage “DC” And rename the “DC”

6. To add a screen to show result and choose instruments and choose virtual terminal and double click on free space

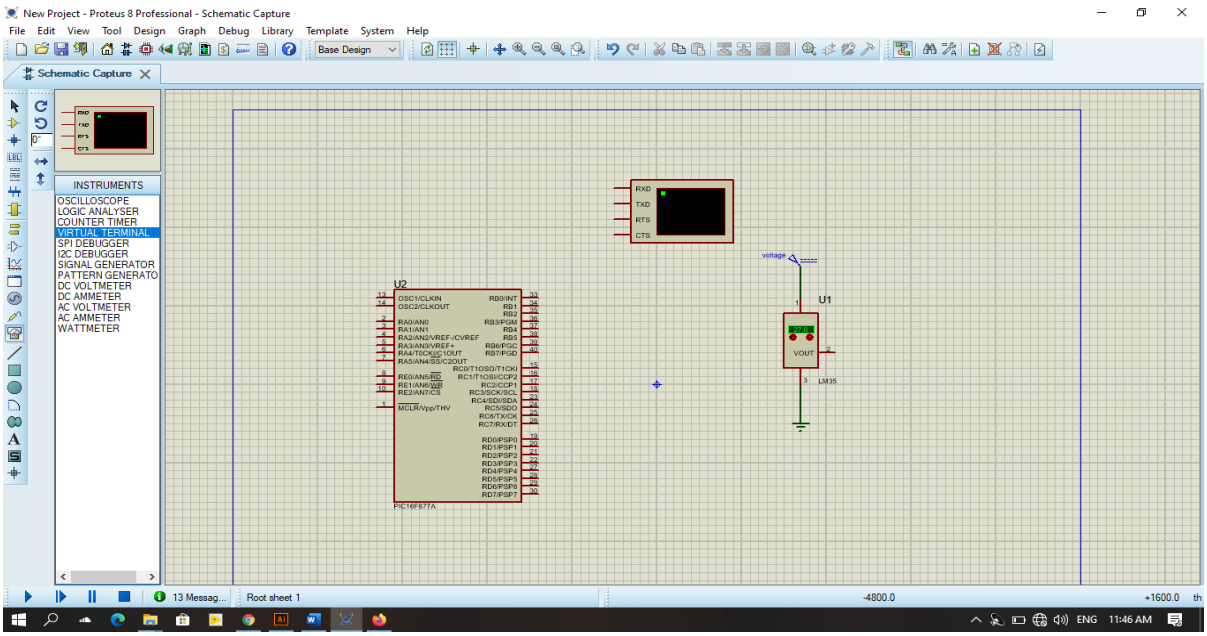

Figure 4.17 add screen to show the result

7. To connect a device to other (pic with Lm32 with screen) click on the edge of a device to connect with other edges of the device

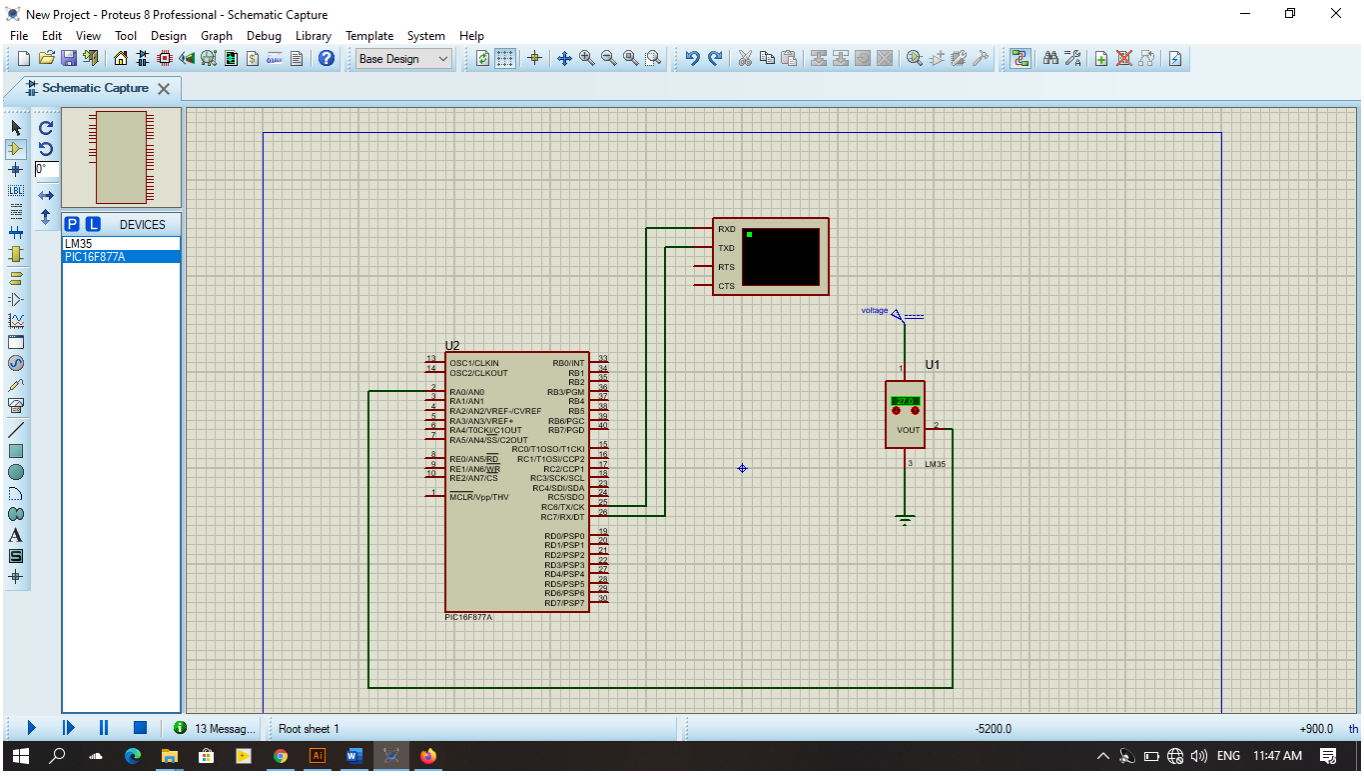

Figure 4.18 connect the devices to other

8. Now click on pic to select “program file” and choose “Processor Clock Frequency” and click ok

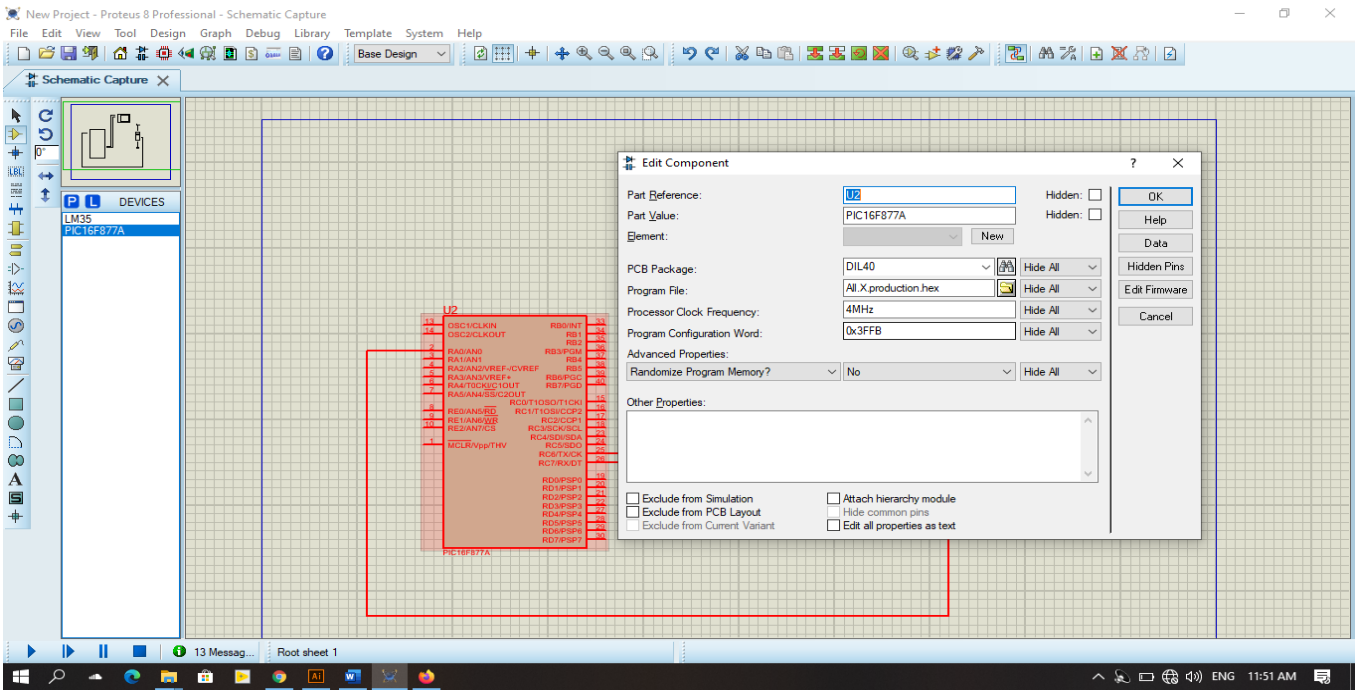

Figure 4.19 select “program file” and choose “Processor Clock Frequency

9. now click F12 or debug --> run simulation “to run the program”

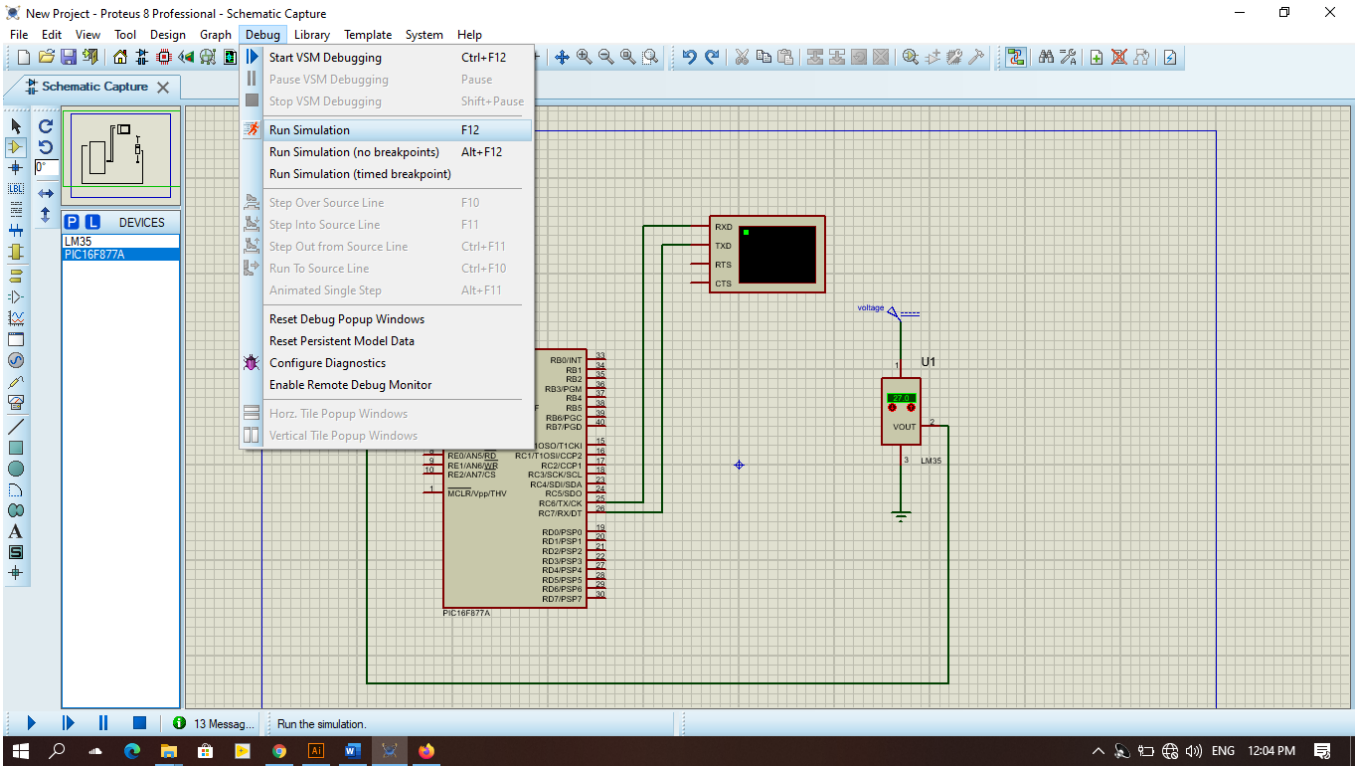

Figure 4.20 run program

4.3 Difficulties in connecting interface between LabVIEW and Arduino

Error 1 “interface between Arduino and LabVIEW”

The first problem when uploading the LIFA_BASE to the Arduino Uno (which I am using). a message appears and my LabVIEW can’t connect to Arduino.

Figure 4.21 error 1

- Solution Error 1

  restart pc and uploaded the LIFA_BASE to the Arduino again.

  Error 2 “Appear this error message”

Figure 4.22 error 2

Cause this problem that there is a library in Arduino cause in appears this error message.

- Solution Error 2

  There are two ways to solve this problem

  1- install older version Arduino like 1.0.5 version.

  2- Remove the library that causes this problem if we don't need it or move it to another location if we need it.

  Step 1

  Open Local Disk(C:) and go to program files

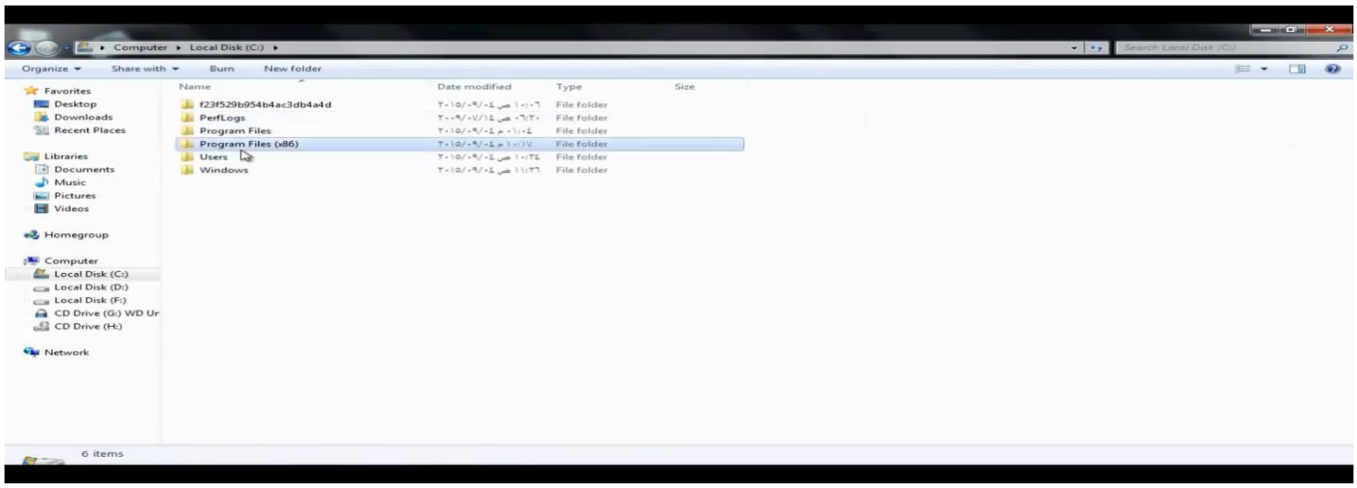

Figure 4.23 open local disk --> program files

Step 2

Click to Arduino

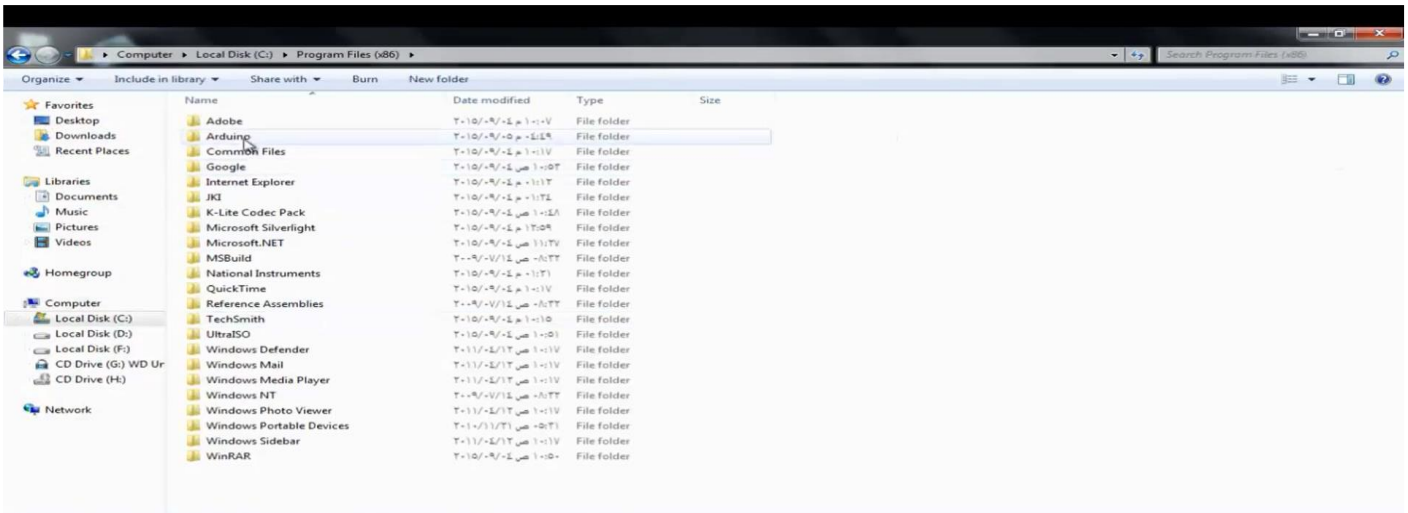

Figure 4.24 Arduino

Step 3

Click to libraries

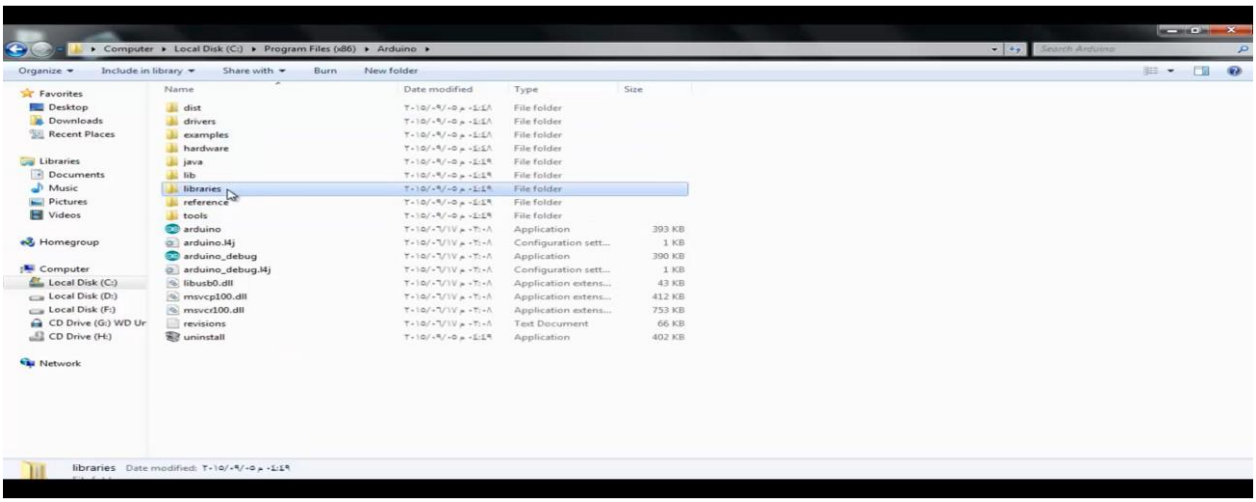

Figure 4.25 libraries

Step 4

Remove or move the RobotIRremote library

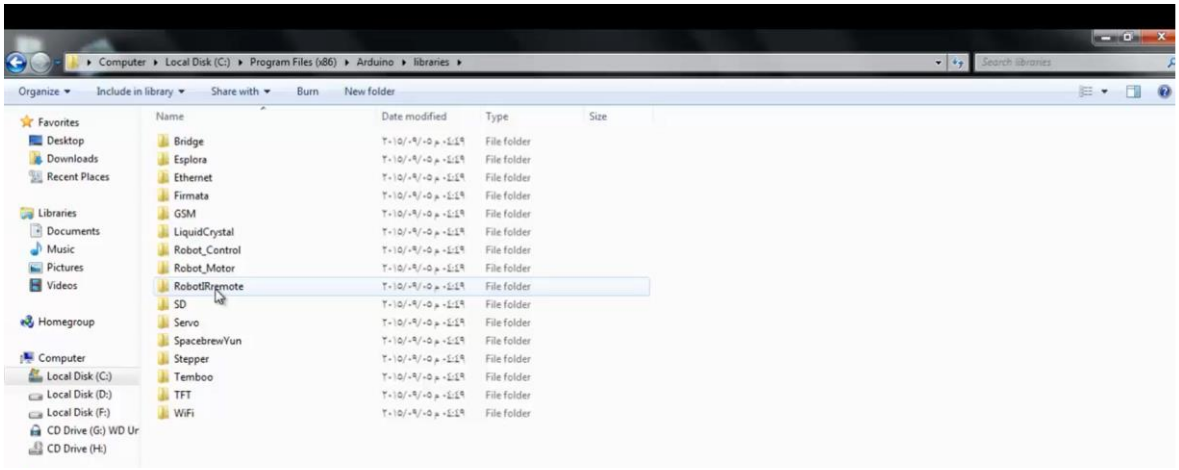

Figure 4.26 RobotIRremote

Step 5

Try to upload the LIFA_BASE to Arduino again and the Uploading will done without any problems

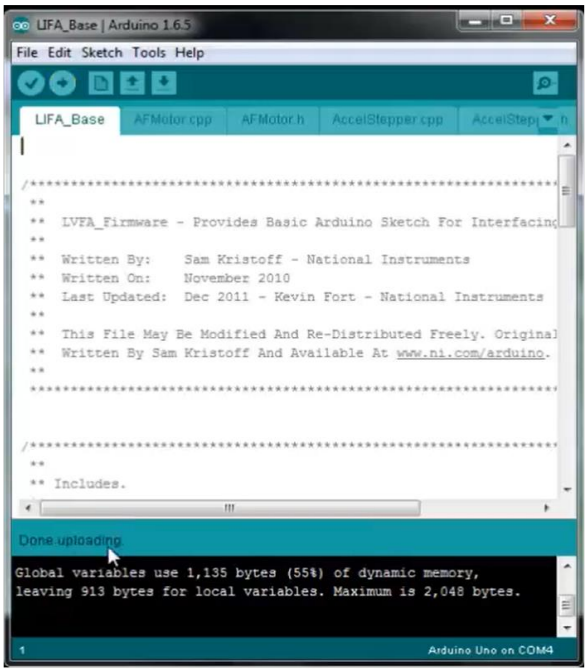

Figure 4.27 uploading LIFA_BASE

PIC cannot be connected directly via pickit2 because the pickit2 contains a chip that makes the LabVIEW or the device read that the pic will only program, but in the LabView, we need to take data from the pic and therefore the pickit2 is not suitable for interfering with the LabView.

Solution of this problem:

By buying Transmitter USB and there were two types

1- RS232 “this is a very old type and we have not found it”
2- TTL “we use it”

Chapter 5: Results

5.1 Operation of monitoring system

This project has been developed with PIC16F877A and Arduino microcontroller connected with sensors which are attached to the patient. All the sensors and location data sent from microcontroller to LabVIEW. A doctor or guardian can read monitor patient's data at any point in time. In case of emergencies, like temperature spike or heartbeat spike . an SMS and email alert sent to doctor and guardian's mobile respectively. And at any point of time either a doctor or guardian can read patient unique credentials and can track patient's location which would help medical services to send appropriate help in case of emergencies.

5.2 LabVIEW

We will now see the graph of the remote healthcare monitoring system. First, we will connect the Bluetooth in laptop to the Bluetooth Module in breadboard. Next, we will select the com from LabVIEW. Since we run the code and show readings.

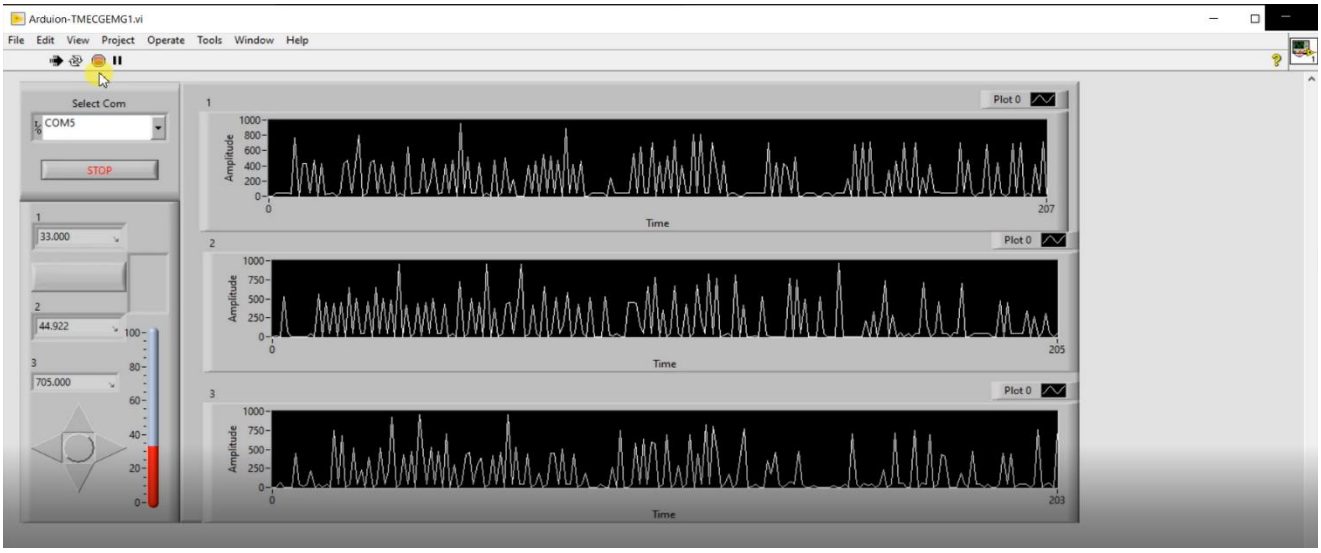

Figure 5.1 LabVIEW Readings4

5.3 Pic microcontroller

First scenario, get the result with pic after making the necessary connections, we have to switch on the power supply to the circuit. Now, we need to pair the Phone‟s Bluetooth to the HC – 05 Bluetooth Module. Before that, we have to install the serial monitor App in the phone. The home screen of the app looks something like this.

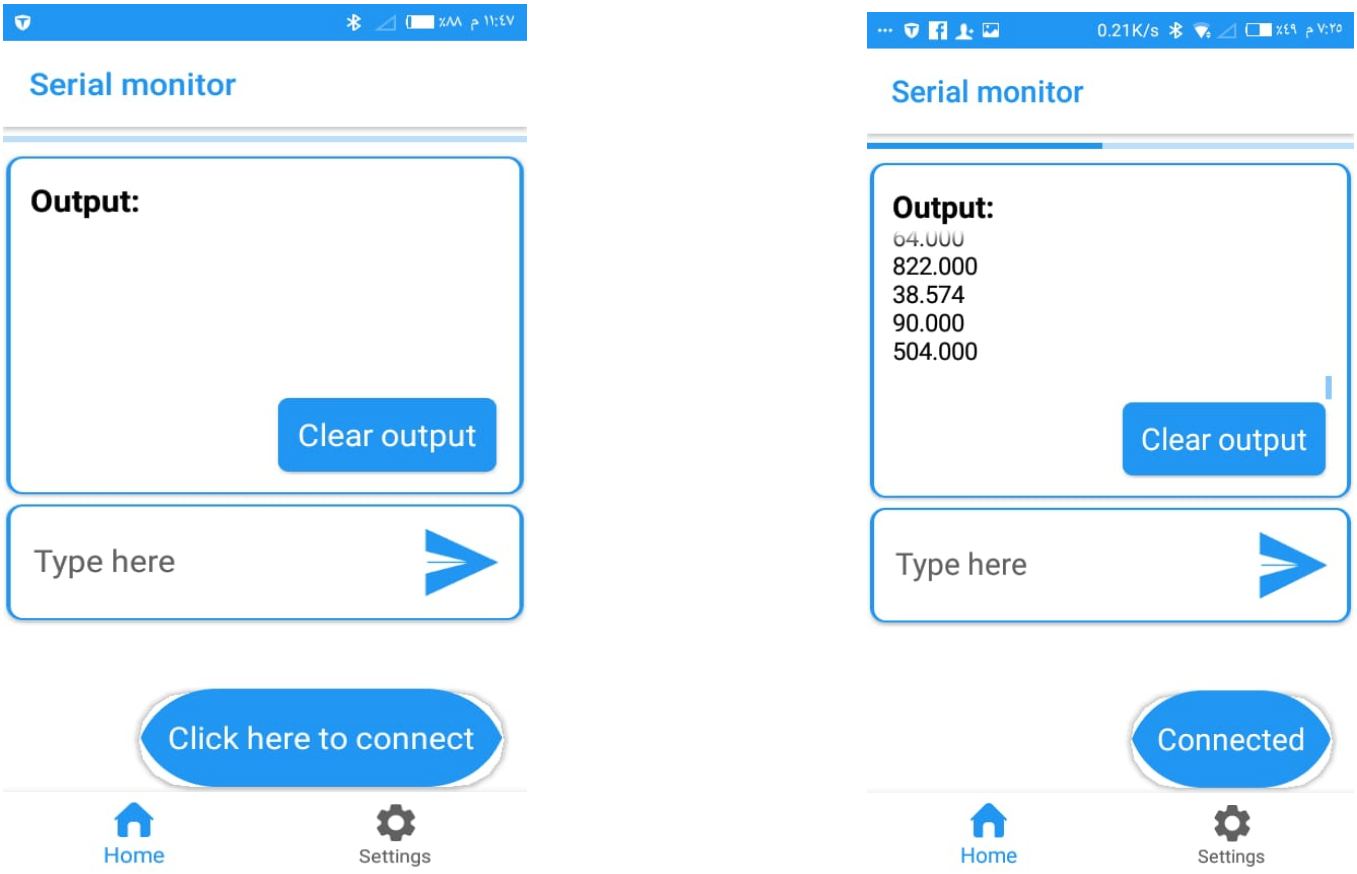

Figure 5.2 pic Readings4

After successful connection, the devices are ready to transmit data. For that, press the click here to connect icon on the app and start giving the output readings.

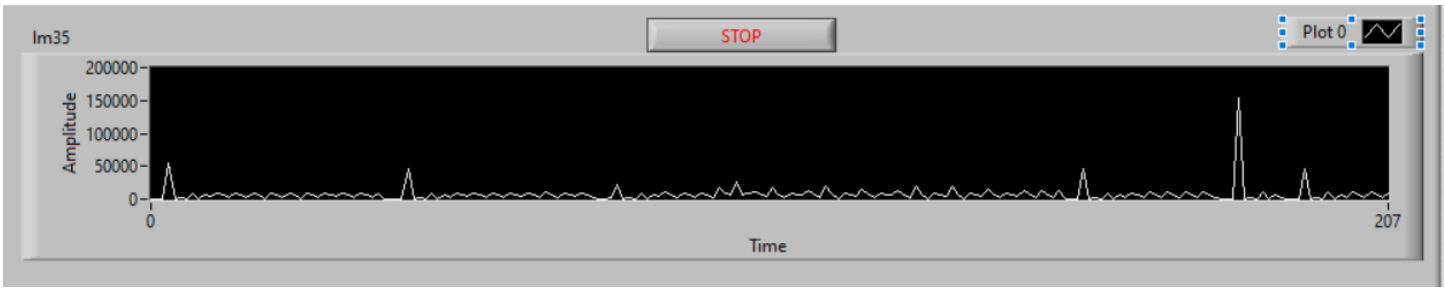

Figure 5.3 temperature graph in LabVIEW

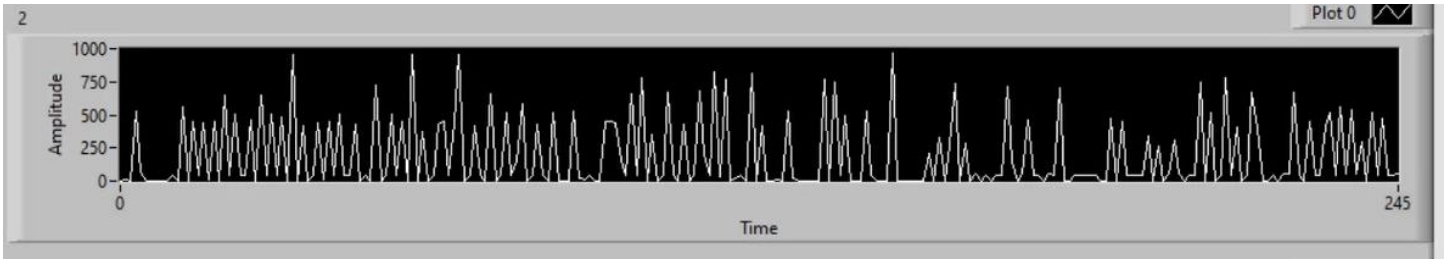

Figure 5.4 pulse sensor graph in LabVIEW

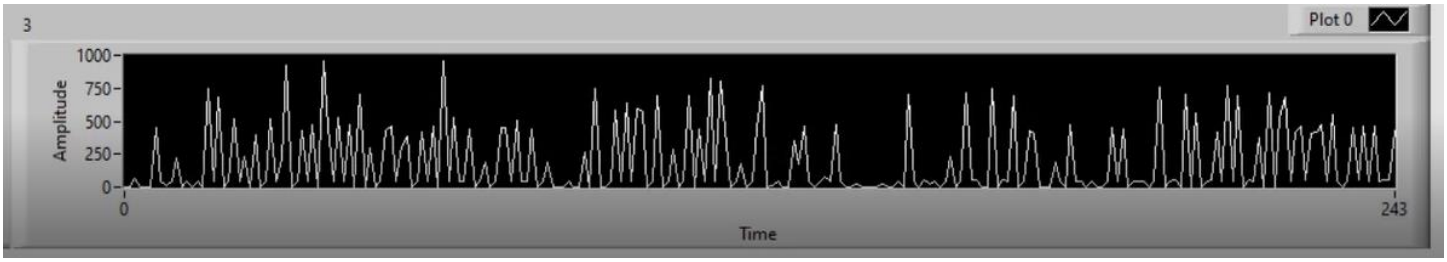

Figure 5.5 Ecg sensor graph in LabVIEW

5.4 Arduino microcontroller

Second scenario reading data from pulsesensor

After having the circuit ready, go to File > Examples > PulseSensor Playground > Getting_BPM_to_Monitor and upload the code. This code will be available in Arduino (after installing the PulseSensor Playground library).

Then, open the serial monitor. You should see something like the figure below:

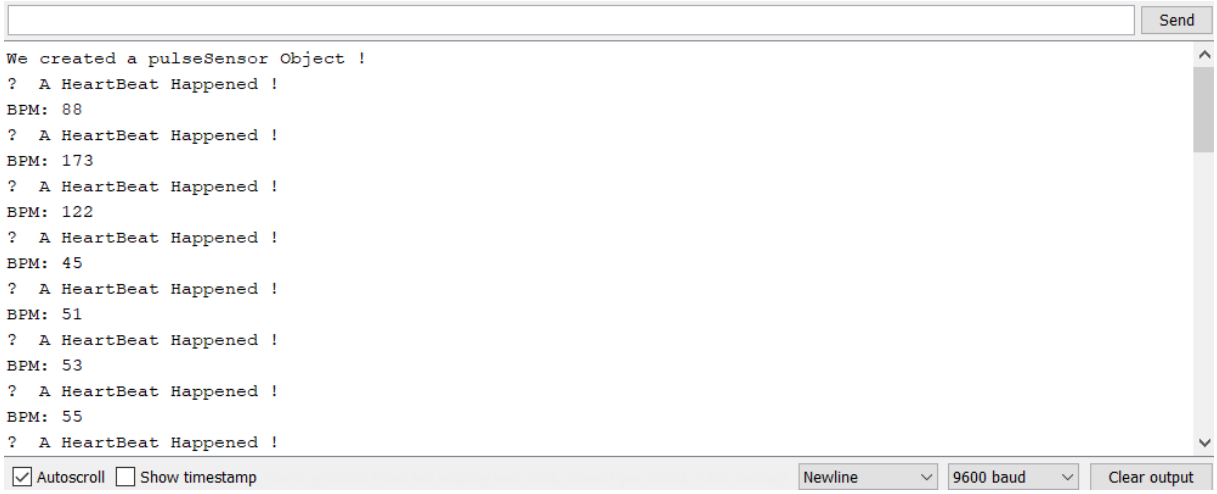

Figure 5.6 PulseSensor Readings with Arduino

5.4.1 Reading data from Ecg sensor

Once the code is uploaded you can go to the serial plotter and you can just see the graph begging plotted like this you can also see the similar wave from on CRO or ascilloscope.

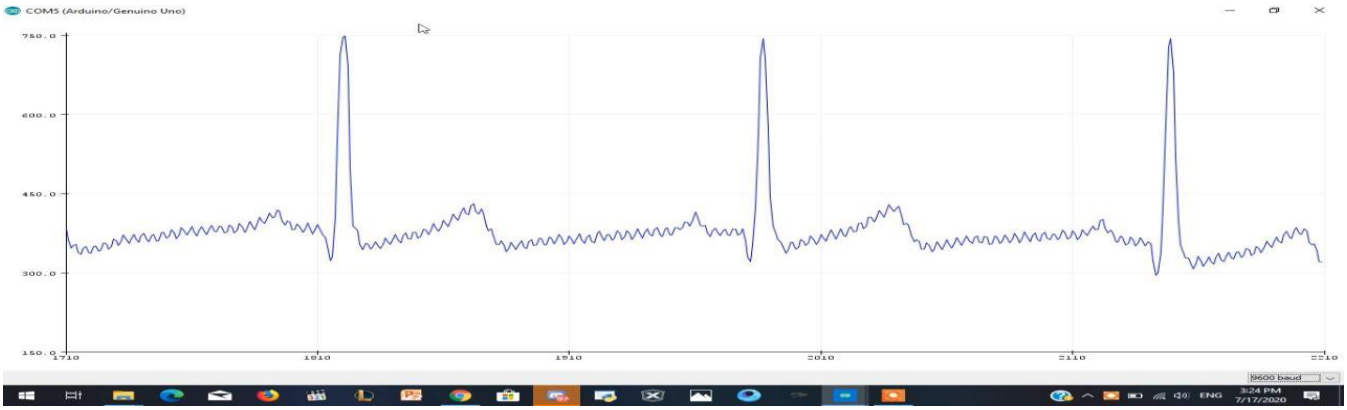

Figure 5.7 Ecg Sensor Readings with Arduino

5.4.2 Reading data from temperature sensor

Once the circuit is ready and the code is uploaded go to the serial plotter and you can just see the readings begging plotted like this

Figure 5.8 Temperature Sensor Readings with Arduino

5.5 Difficulties and integrating problems

- The source code of the ECG that we found in the datasheet was an around code not PIC, and we couldn't find code for the sensor that wording with PIC.

Solved by: Build the C code from scratch.

- After building the code on Micro-C compiler we didn't know that this compiler doesn't support PicKit2 that we are using.
  Solved by: Re-building the code on MATLAB.
- Tried to simulate the sensor using Protaso but the sensor isn't available on this program. [Not Solved]
- Our unfamiliarity with the Microcontroller ,because of our lack of experience we had some issues with the microcontroller since it is the first time we are dealing with this microcontroller such as buying a microcontroller version not compatible with the programming kit we got.

Solved by: The moment we knew about the microcontroller not being compatible we made some more researches so we found the most common version of the microcontroller which solved the problem of incompatibility.

- Lack of components Some components are hard to find due to some problems shipping them from aboard, so it causes wasting a lot of time.
  Not solved until now
- The microcontroller didn't print results by using a TTL USB module we tried to print the results of the sensors which were connected to the pic in the programmer but it didn't give any response.
- Solved: Because there isn't any other power source other than the serial adapter in the programmer which is responsible only for uploading codes to the pic so when we tried to connect the TTL it didn't send any feedback because there are two communication ways now in the programmer (serial cable, TTL module) so by making a wiring to the pic in an external breadboard with an external non communicational power source the TTL worked properly and gave a proper feedback.

Chapter 6: Conclusion and Future work

6.1 Conclusion

Remote health care monitoring system (RHCMS) is a technology to enable monitoring of patients outside of conventional clinical settings, such as in the home or in a remote area, which may increase access to care and decrease healthcare delivery costs at the same time.

In this version of the Remote health care monitoring system (RHCMS), the main concern of the project is the patients that have to be monitored with Intensive care devices like ECG, Temperature monitor, and muscle strength monitor, these patients need constant care.

By using sensors such as (ECG, EMG, Temperature, and Heartbeat) it can depict the patients' health state, predict any future problems that may occur, and prevent it from happening.

The System works remotely to give patients more comfort and the ability to move without the need to be connected to lots of devices that would obstruct his/her movement. They'll also be monitored remotely by doctors and medical professionals.

The software needed to integrate between microcontrollers, sensors, and LabVIEW in order to be this project successful.

It transmits the patients' health state data from the sensors using a wireless module to the LabVIEW

As a result of that, it will be able to come up with a new way of monitoring these patients without the need to be in the Intensive care unit and it also provides 24/7 medical support by only using sensors and microcontrollers.

6.2 Future work

In this project, it faced some limitations and problems that it will deal with in the future.

One of these limitations was to get ZigBee wireless module as it was not provided at the time of the implementation of the project, In the future, this module will be available and implemented to the system.

Another limitation was because of the availability of the ZigBee module that could not be used the EMG sensor because of its not compatible with the Bluetooth module, and this problem will be solved In the future when the ZigBee will be available.

Also, In the future, the system will be integrated with a mobile application that receives real-time statues about the patients.

Finally, the project will be available to be used by the patients.

References

[1]. Fariborz H., Moghawemi, M., and Mehrkanoon, S, "The design of an intelligent wireless sensor network for ubiquitous healthcare", In the Proceedings of the International Conference on Intelligent and Advanced System, November 25-28, 2007, Kualalumpur, Malaysia, pp. 414-417

[2]. AlSharqi, Khalifa, et al. "Zigbee based wearable remote healthcare monitoring system for elderly patients." International Journal of Wireless & Mobile Networks 6.3 (2014): 53.

[3]. Crawley, Edward, Bruce Cameron, and Daniel Selva. "*System architecture: strategy and product development for complex systems*. " Prentice Hall Press, 2015.

[4]. Lethbridge, Timothy Christian, and Robert Laganiere. "*Object-oriented software engineering*." New York: McGraw-Hill, 2005.

[5]. Jacobson, Ivar. "*Object-oriented software engineering: a use case driven approach*." Pearson Education India, 1993.

[6]. Dennis, Alan, Barbara Haley Wixom, and David Tegarden. "*Systems analysis and design: An object-oriented approach with UML*." John wiley & sons, 2015.

[7]. Fowler, Martin. "*UML distilled: a brief guide to the standard object modeling language*. " Addison-Wesley Professional, 2004.

[8]. Selic, Bran, and Sébastien Gérard. "*Modeling and analysis of real-time and embedded systems with UML and MARTE: Developing cyber-physical systems*." Elsevier, 2013.

[9]. Louis Henri Sullivan "The Tall Office Building Artistically Considered," Lippincott's Magazine, March 1896.

[10].https://www.microchip.com/DevelopmentTools/ProductDetails/PartNO/PG164120#:~:text=The%20PICkit%E2%84%A2%202%20Development,Microchip's%20Flash%20families%20of%20microcontrollers.&text=In%2DCircuit%2DDebugging%20runs%2C,is%20embedded%20in%20the%20application.

[11].https://www.microchip.com/wwwproducts/en/PIC16F877A

[12].https://components101.com/pic16f877a-pin-diagram-description-features-datasheet

[13].https://components101.com/microcontrollers/arduino-uno

[14].https://www.makeuseof.com/tag/what-is-breadboard/

[15].https://components101.com/7805-voltage-regulator-ic-pinout-datasheet

[16].https://microcontrollerslab.com/oscillator-types-microcontrollers/#:~:text=In%20a%20pic%20microtroller%20MID,and%20peripherals%20of%20the%20function.&text=The%20oscillator%20has%20following%20modes%20of%20operations.

[17].https://www.bc-robotics.com/shop/22pf-ceramic-capacitor/#:~:text=Description,a%20solderless%20breadboard%20or%20protoboard.

[18].https://simple-circuit.com/pic16f877a-lm35-temperature-sensor-ccs-c/

[19].https://www.engineersgarage.com/knowledge_share/lm35-description-and-working-principal/

[20].https://www.theengineeringprojects.com/2019/01/introduction-to-lm35.html

[21].https://www.electronicwings.com/components/lm35-temperature-sensor

[22].https://www.electroschematics.com/heart-rate-sensor/

[23].https://store.fut-electronics.com/products/heart-rate-pulse-sensor

[24].https://shop.openbci.com/products/pulse-sensor?variant=22543672899

[25].https://components101.com/modules/ad8232-ecg-module

[26].https://www.sparkfun.com/products/12650#:~:text=The%20AD8232%20is%20an%20integrated,motion%20or%20remote%20electrode%20placement

[27].https://www.researchgate.net/figure/The-comparison-chart-of-Bluetooth-Low-Energy-and-ZigBee_tbl1_271327094

[28]. https://components101.com/wireless/xbee-s2c-module-pinout-datasheet

[29]. https://components101.com/wireless/hc-05-bluetooth-module

[30].https://www.semanticscholar.org/paper/Car-alarm-detection-device-Faridah-Rahman/0178a4c78052d4990462ba4b12f6c2d2b3f207ef/figure/1

[31]. https://www.ni.com/en-lb/shop/labview.html

[32]. https://www.labcenter.com/

This project was uploaded in GitHub:

https://github.com/HealthCareProject/Remote-healthcare-monitoring-system-using-ZigBee-and-LabVIEW

Appendix A

(Arduino code with the Sensors)

```
#define USE_ARDUINO_INTERRUPTS true
#include <PulseSensorPlayground.h>
//  Variables
const int PulseWire = 0;
const int LED13 = 13;
int Threshold = 550;
const int lm35_pin = A1;
PulseSensorPlayground pulseSensor;
void setup() {
    Serial.begin(9600);
  pulseSensor.analogInput(PulseWire);
  pulseSensor.blinkOnPulse(LED13);
  pulseSensor.setThreshold(Threshold);
  pinMode(10, INPUT);
  pinMode(11, INPUT);
}
void loop() {
  HR();
  TEMP();
  ECG();
}
void HR() {
   int myBPM = pulseSensor.getBeatsPerMinute();

if (pulseSensor.sawStartOfBeat()) {
 Serial.print("BPM: ");
 Serial.println(myBPM);
  delay(20);
}
}
void TEMP() {
  int temp_adc_val;
  float temp_val;
```

```
  temp_adc_val = analogRead(lm35_pin);
  temp_val = (temp_adc_val * 4.88);
  temp_val = (temp_val/10);
  Serial.print("Temperature = ");
  Serial.print(temp_val);
  Serial.print(" Degree Celsius\n");
}
void ECG(){
if((digitalRead(10) == 1)||(digitalRead(11) == 1)){
Serial.println('!');
}
else{
Serial.println(analogRead(A2));
}
delay(1);
}
```

Appendix B

(PIC code with the Sensors)

```
#include <xc.h>
#include "config.h"
#include <stdint.h>
#include <stdio.h>
#include <pic16f877a.h>
//================================
//--------[ Definitiotns ]---------
#define _XTAL_FREQ 4000000
#define Baud 9600
//================================
//-----[ Globals ]-----
uint16_t AN0RES = 0, AN0RES2 = 1, AN0RES3 = 2;
float Temperature, temp_Voltage, Heart_Voltage, Heartbeat_Voltage;
char* TempSTR[10], HeartbeatSTR[10], HeartSTR[10];
//================================
//-----[ Functions Prototypes ]----
void ADC_Init();
uint16_t ADC_Read(uint8_t);
void UART_TX_Init(void);
void UART_Write(uint8_t);
void UART_Write_String(char *);
//================================
//---------[ Main Routine]---------
void main(void) {

  UART_TX_Init();
  ADC_Init();
  TRISCbits.TRISC1 = 1;
  TRISCbits.TRISC2 = 1;
  while(1)
  {
    // Tempreature Sensor
    AN0RES = ADC_Read(0);
    temp_Voltage = AN0RES * 0.0048828;
```

```
    Temperature = temp_Voltage / 0.01;
    sprintf(TempSTR, "%.3f \r\n", Temperature);
    UART_Write_String(TempSTR);

    // Heartbeat Sensor
    AN0RES2 = ADC_Read(1);
    Heartbeat_Voltage = AN0RES2 * 0.09;
    sprintf(HeartbeatSTR, "%.3f \r\n", Heartbeat_Voltage);
    UART_Write_String(HeartbeatSTR);

    // ECG Sensor
    if(PORTCbits.RC1 == 1 ||  PORTCbits.RC2 == 1){
        sprintf(HeartSTR, "!");
        UART_Write_String('HeartSTR');
    }
    else{
        AN0RES3 = ADC_Read(2);
        Heart_Voltage = AN0RES3;
        sprintf(HeartSTR, "%.3f \r\n", Heart_Voltage);
        UART_Write_String(HeartSTR);
    }
    __delay_ms(500);
  }
  return;
}
//================================
//--------[ ADC Routines ]---------
void ADC_Init()
{
  ADCON0 = 0x41; // Turn ADC ON, Select AN0 Channel, ADC Clock = Fosc/8
  ADCON1 = 0x80; // All 8 Channels Are Analog, Result is "Right-Justified"
  // ADC Clock = Fosc/8
}
uint16_t ADC_Read(uint8_t ANC)
{
  if(ANC<0 || ANC>7) // Check Channel Number Validity
  { return 0;}
```

```
  ADCON0 &= 0x11000101; // Clear The Channel Selection Bits
  ADCON0 |= ANC<<3; // Select The Required Channel (ANC)
  // Wait The Aquisition Time
  __delay_us(30); // The Minimum Tacq = 20us, So That should be enough
  GO_DONE = 1; // Start A/D Conversion
  while(ADCON0bits.GO_DONE); // Polling GO_DONE Bit
  // Provides Delay Until Conversion Is Complete
  return ((ADRESH << 8) + ADRESL); // Return The Right-Justified 10-Bit Result
}
//================================
//--------[ UART Routines ]--------
void UART_TX_Init(void)
{
  // Set Baud = 9600
  BRGH = 1;
  SPBRG = 25; //Writing SPBRG Register
  //--[ Enable The Ascynchronous Serial Port ]--
  SYNC = 0;
  SPEN = 1;
  //--[ Set The RX-TX Pins to be in UART mode (not io) ]--
  TRISC6 = 1; // As stated in the datasheet
  TRISC7 = 1; // As stated in the datasheet
  TXEN = 1; // Enable UART Transmission
}
void UART_Write(uint8_t data)
{
  while(!TRMT);
  TXREG = data;
}
void UART_Write_String(char *text)
{
  uint16_t i;
  for(i=0;text[i]!='\0';i++)
    UART_Write(text[i]);
}
```

Appendix C

(PIC Configrations)

```
#pragma config FOSC = HS        // Oscillator Selection bits (HS oscillator)

#pragma config WDTE = OFF       // Watchdog Timer Enable bit (WDT disabled)

#pragma config PWRTE = OFF      // Power-up Timer Enable bit (PWRT disabled)

#pragma config BOREN = ON       // Brown-out Reset Enable bit (BOR enabled)

#pragma config LVP = OFF         // Low-Voltage (Single-Supply) In-Circuit Serial Programming Enable bit (RB3 is digital I/O, HV on MCLR must be used for programming)

#pragma config CPD = OFF        // Data EEPROM Memory Code Protection bit (Data EEPROM code protection off)

#pragma config WRT = OFF       // Flash Program Memory Write Enable bits (Write protection off; all program memory may be written to by EECON control)

#pragma config CP = OFF         // Flash Program Memory Code Protection bit (Code protection off)


#include <xc.h>
```

نظام مراقبة الرعاية الصحية عن بعد باستخدام الزيجبي و اللاب فيو

يهدف المشروع إلي متابعة المرضي خاصة كبار السن من علي بعد خاصة بسبب زيادة حالات الشيخوخه وكبر السن وصعوبة الحركه زيادة علي ذلك حاجه الطبيب لمتابعة المريض بحاله متواصله ومتابعة الجرعات والحاله الجسديه وما الي ذلك. بالإضافة إلى ذلك ، لا يحصل الأشخاص من المناطق الفقيرة في العالم على الرعاية الصحية التي يحتاجونها ويستحقونها. لحل هذه المشاكل تم اقتراح العديد من الإصدارات البحثية والتجارية وتنفيذها حتى الآن. في هذه الأنظمة ، كان الأداء هو المشكلة الرئيسية من أجل قياس بيانات المرضى وتسجيلها وتحليلها بدقة. مع صعود الشبكة اللاسلكية يمكن نشرها على نطاق واسع لرصد الحالة الصحية للمريض داخل وخارج المستشفيات. يقدم هذا المشروع نظام مراقبة قائم على الرعاية الصحية لاسلكيًا (زيجبي ، وبلوتوث ، وما إلى ذلك) يمكن أن يوفر معلومات فورية عبر الإنترنت حول الحالة الصحية للمريض. النظام المقترح قادر على إرسال رسائل إلى أخصائي الرعاية الصحية حول حالة المريض. بالإضافة إلى ذلك ، يمكن للنظام المقترح إرسال تقارير إلى نظام مراقبة المرضى ، والذي يمكن استخدامه من قبل المتخصصين في الرعاية الصحية لتقديم المشورة الطبية اللازمة من أي مكان في العالم في أي وقت معين.