

## Chapter1

# 1. Introduction

---

### 1.1 The Meaning of neonatal incubator:

Neonatal incubator is a medical device used to maintain an optimal environment for the care of newborn as shown in (Figure 1-1).



Figure1-1 Neonatal care or Specialized nurseries

### 1.2 The Usage Of neonatal incubator:

- The neonatal incubator used for premature babies that meaning who born pre-term or before 37 weeks.
- The neonatal incubator is used to monitor, treat and protect news babies and premature baby.
- The neonatal incubator provides warmth, humidity and oxygen all in a controlled environment as required by news born.

## 1.3 History of incubator:

- In 1891: first modern in an incubator invented by Dr.Alexander Lyon as shown in (Figure 1-3-1).

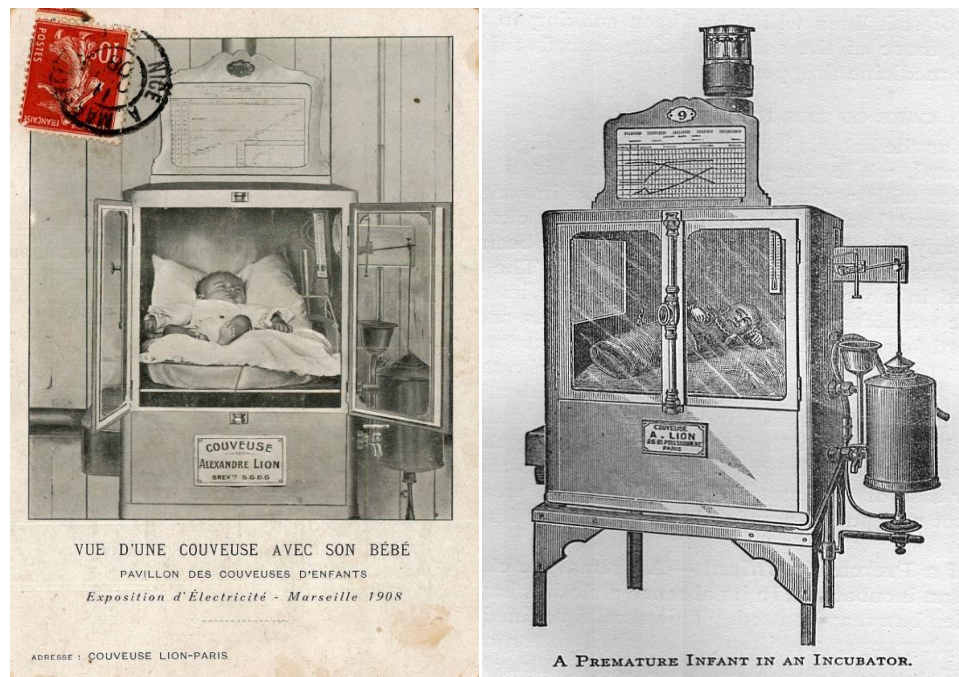


Figure1-3-1 First Incubator

- 1898: First American incubator hospital was set up at the Trans Mississippi exposition in Omaha Nebraska.
- 1907: Pierre constant Budin released the study of the influence of body temperature on infant mortality.
- 1932: Julius Hess in his patents for incubator proposed a mechanism for the addition of supplemental oxygen in the Incubator.

## 1.4 Types of incubator:

1. Portable and non-portable: portable incubation can be used to shift the patient to another area of the hospital as needed as shown in (Figure 1-4-1).



Figure 1-4-1 Portable Incubator

2. Open box type: it is also known as Armstrong, here neonate is kept on the Plexiglas bassinet to keep unstable babies or newly born Babies, as shown in (figure 1-4-2) a radiant warmer can be attached if The child needs.

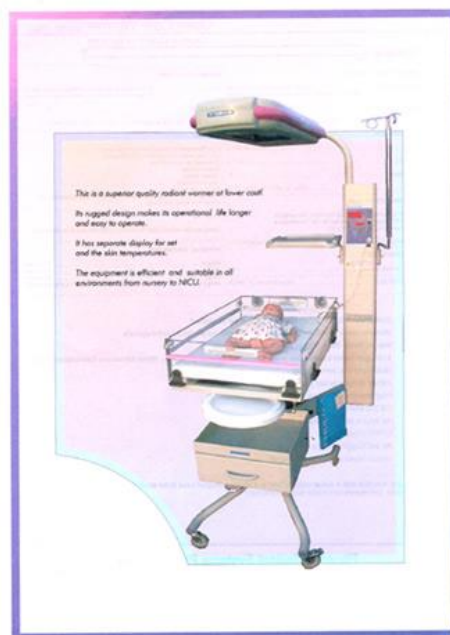


Figure1-4-2 Open box type

3. Close type: close type of incubator has a special function to concentrate fresh air after filtration. It prevents water loss from radiation as neonate remain inside the box the risk of infection is minimal as shown in a (figure1-4-3).



Figure 1-4-3 Close type

4. Double walled: the incubator has two walls as shown in (figure 1-4-4) as air not a good conductor of heat the incubator prevents heat and fluid loss.



Figure 1-4-4 Double walled

## 1.5 Parts of incubator:

1. Fan: The fan used for disinfection and fumigation.
2. Heater: The heater is adjustable and helps maintain an infant's core body temperature.
3. Air Distributors: Air distributors distribute air evenly to ensure equal temperature throughout the incubator. Which prevents spaces of cold or stale air.
4. Canopy: The canopy is clear, acrylic covering that protects the baby from the outside world and harmful germs that may infect the child. It also makes the perfect warm and oxygenated environment for baby that's similar to a mother's womb.
5. Filters: clean the air before it is pulled into the incubator, preventing harmful particles from entering the incubator and possibly infecting the infant's lungs.
6. Hygrometer: This part of the incubator measures the amount of humidity inside the incubator.  
It is also responsible for breathing warm and humidified air into the baby's lungs through endotracheal tubes that run from the baby's nostril into lungs.
7. Inlets: allow the administration of oxygen.
8. Portholes: portholes allow nurses and caretakers to handle the baby without contaminating the infant's environment. Portholes are holes sealed with rubber gloves that must insert hands into in order have limited and contamination-free access to the inside of the incubator.
9. Respiratory Tubing (Mechanical Respirator): as shown in (figure 1-5-1), these tubes are usually endotracheal (inserted through the nostril for access to lungs) and are a way to provide artificial oxygenation to an infant.

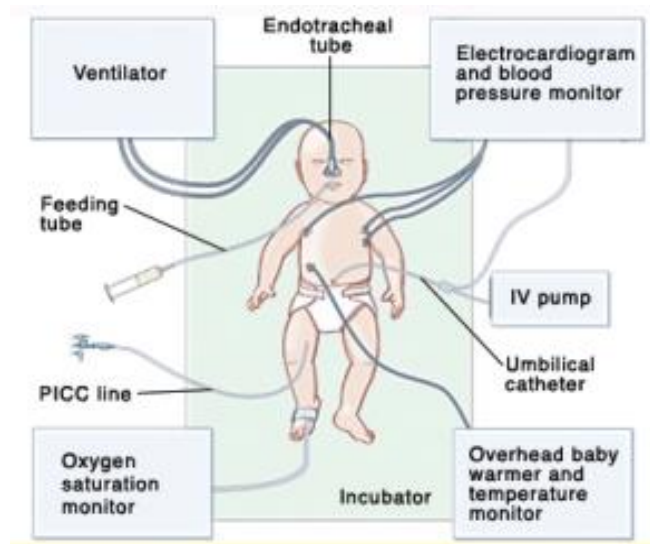


Figure 1-5-1 Baby inside the Incubator with all connection

## 1.6 Common conditions treated in the neonatal intensive care unit (NICA):

1. **Anaemia:** premature babies are often anemic because they do not have enough red blood cell. The fetus stores iron during the latter months of pregnancy and uses it after birth to make red blood cells. Infants born too may have had enough time to store iron.
2. **Breathing problems:** premature babies often have breathing problems because their lungs are not fully developed.
3. **Spina:** premature babies sometimes do not breathe regularly. A baby may take long breath, then a short one, then pause for five to 10 seconds before starting to breathe normally. This is called periodic breathing.
4. **Bronchopulmonary dysplasia (BPD):** The chronic lung disease is most common in premature babies who have related for respiratory distress syndrome.

5. Persistent pulmonary hypertension of newborn (PPHN): babies with PPNA cannot breathe properly because they have high blood pressure in their lungs. Babies with PPNA often need a ventilator to help them breath.
6. Heart valve abnormalities: some babies are born with heart valves that narrowed, closed or blocked and this condition prevents blood from flowing smoothly.
7. Body temperature and Weight: Babies who are born too small and soon often have trouble controlling their body temperature.
8. Jaundice: Babies with jaundice have a yellow wish color to their skin and eyes. Jaundice occurs when the liver is too immature or sick.

## **1.7 Temperature Body of Baby:**

### **1.7.1 Body Temperature divided into two Categories:**

1. Hypothermia: is low blood Sugar in that case the body temperature is low(less than 30 °c) it occurs when body temperature drops to 30 °c as shown in a (table 1-7-1).
2. Hyperthermia: is High blood sugar that case high body temperature (more than 37 °c), it occurs when care body more than 37.0 °c as shown in (table 1-7-2).

### 1.7.2 Temperature Range:

Age	1000to1200gm	1201to1500gm	1501to2500gms	2500gms and>36wee
0-12Hrs	35.0	34.0	33.3	32.8
12-24Hr	34.5	33.8	32.8	32.4
24-96Hr	34.5	33.5	32.3	32.0

**Table 3-7-1 Temperature Range**

Age	<1500gms	<1501to2.500gms	2.500gms >36weeks
5 – 14 days	33.5	32.1	32.0
2 – 3 week	33.1	31.7	30.0
3 – 4weeks	33.6	31.4	-
4 – 5 week	32.0	30.9	-
5 – 6 week	31.4	30.4	-

**Table 1-7-4 Temperature Range**



## Chapter 2

### Hardware and component

---

#### 1. Hardware

The hardware in information technology is physical aspect of computers, telecommunications, and other devices. The hardware for the infant incubator consists of (humidity and temperature) sensor to measure the humidity level and incubator temperature degree, body temperature to measure the temperature degree of the baby, and a pulse sensor to measure the heart rate of the baby. The measurement results which are the pulse rate, humidity level, incubator temperature degree and body temperature degree sent to the mobile phone via Arduino microcontroller.

#### 2. The component and their types

There are many types of sensors and microcontroller to use.

##### 2.1 Arduino

- Arduino is a prototype platform (open-source) based an easy-to-use hardware and software. It consists of a circuit board, which can be programming (referred to as a microcontroller) and a ready-made software called Arduino IDE (Integrated Development Environment), which is used to write and upload the computer code to the physical board. The key features are:
- Arduino boards can read analogue or digital input signals from different sensors and turn it into an output such as activating a motor, turning LED on/off, connect to the cloud and many other actions.
- Control the board functions by sending a set of instructions to the microcontroller on the board via Arduino IDE (referred to as uploading software).

- Unlike most previous programmable circuit boards, Arduino does not need an extra piece of hardware (called a programmer) to load a new code onto the board. The transfer operation between the hardware and software by contacting the pc and Arduino with USB cable.
- Additionally, the Arduino IDE as shown in (figure 2.1.1) uses a simplified version of Arduino c, making it easier to learn to program.
- Finally, Arduino provides a standard form factor that breaks the functions of the micro-controller into a more accessible package.

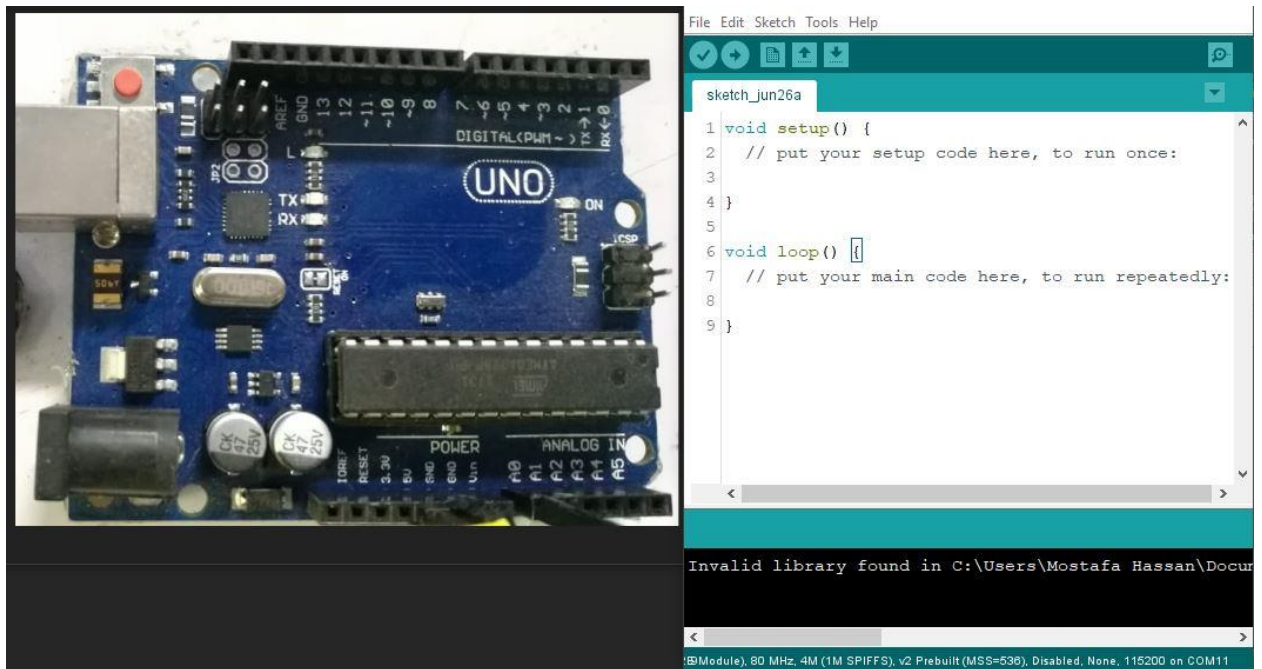


Figure 2-1-1 Arduino Uno and IDE software

### 2.1.1 Board Types:

Various kinds of Arduino boards are available shown as (tables 2.1.1) depending on different microcontrollers used. However, all Arduino boards have one thing in common: the software is Arduino IDE. The differences between the Arduino kits based on the number of inputs and outputs (the number of sensors, LEDs, and buttons to use on a single board), speed, operating voltage and form factor. Some boards are designed to be embedded and have no programming interface (hardware), which would need to buy separately. Some can run directly from a 3.7V battery; others need at least 5V.

Name	Processors	Operating voltage	CPU Speed	Analog in/out	Digital IO/PWM
UNO	ATmega328	5V/7-12V	16MHz	6	14
Mega 2560	ATmega2560	5V/7-12V	16MHz	16	54
Mini	ATmega328	5V/7-9V	16MHz	8	14
Nano	ATmega328 ATmega168	5V/7-12V	16MHz	8	14

Table 2-1-1 (Arduino boards microcontroller)

## 2.1.2 Arduino – Board Description

The Arduino UNO is the best board to get started with electronics and coding because it is the most popular board in the Arduino board family. Also, some boards look a bit different from the one given below shown in (figure 2-1-2), but most Arduinos have the majority of these components in common.

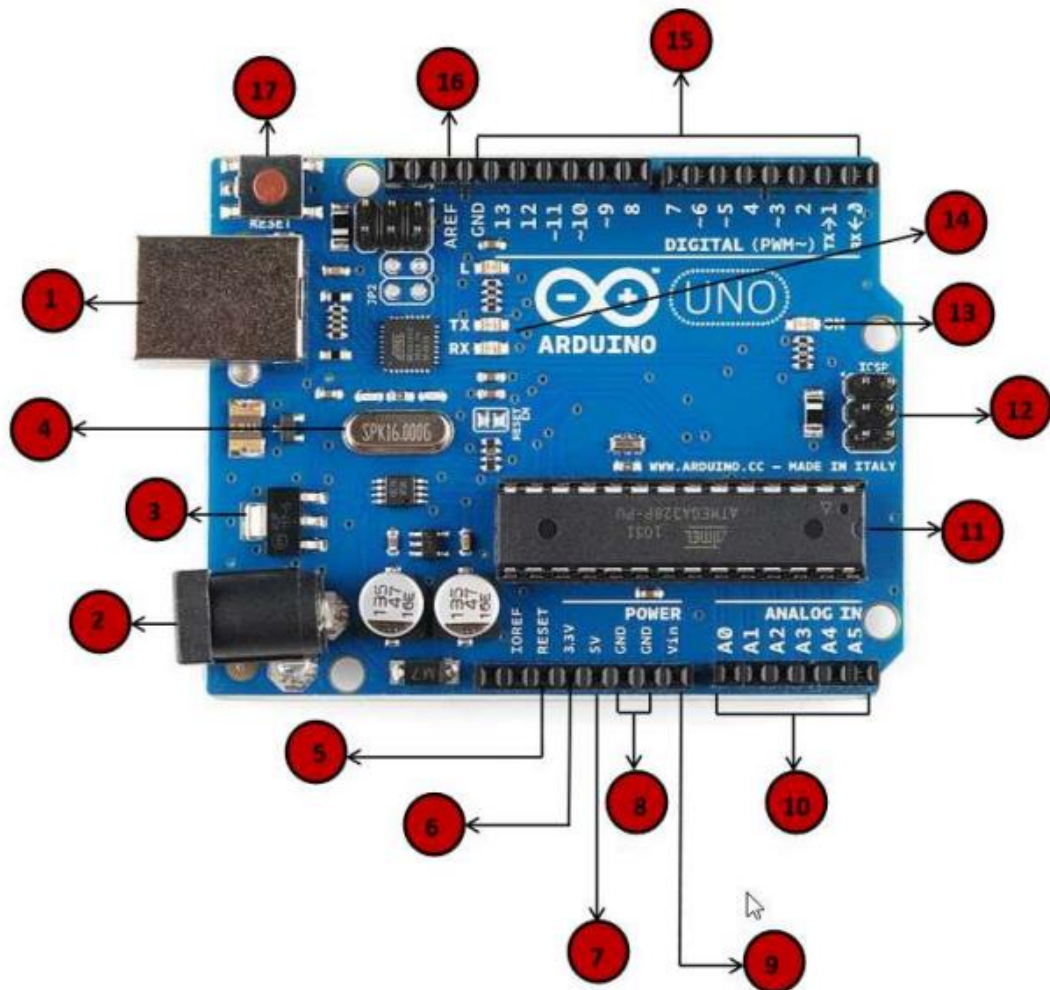


Figure 2-1-2 Arduino Uno component



### **Power USB**

Arduino board can plug by using the USB cable from the computer. Just 1 need to do is connect the USB cable to the USB connection (1).



### **Power (Barrel Jack)**

Arduino boards can plug directly from the AC mains power supply by connecting it to the Barrel Jack (2).



### **Voltage Regulator**

The function of the voltage regulator is to control the voltage given to the Arduino board and stabilize the DC voltages used by the processor and other elements.



### **Crystal Oscillator**

The crystal oscillator helps Arduino in dealing with time issues. How does Arduino calculate time? The answer is, by using the crystal oscillator. The number printed on top of the Arduino crystal is 16.000H9H. It tells us that the frequency is 16,000,000 Hertz or 16 MHz



### **Arduino Reset**

To reset the Arduino board, i.e., start the program from the beginning. Just reset the UNO board in two ways. First, by using the reset button (17) on the board. Second, connect an external reset button to the Arduino pin labelled RESET (5).



### **Pins (3.3, 5, GND, V<sub>in</sub>)**

- 3.3V (6): Supply 3.3 output volt
- 5V (7): Supply 5 output volt
- Most of the components used with Arduino board works fine with 3.3 volts and 5 volts.
- GND (8) (Ground): There are several GND pins on the Arduino, any of which can be used to ground the circuit.
- V<sub>in</sub> (9): This pin also can be used to power the Arduino board from an external power source, like AC mains power supply.



### **Analog pins**

The Arduino UNO board has five analogue input pins A0 through A5. These pins can read the signal from an analogue sensor like the humidity sensor or temperature sensor and convert it into a digital value that can be read by the microprocessor.



### **Main microcontroller**

Each Arduino board has its microcontroller (11). Assume it as the brain of the board. The main IC (integrated circuit) on the Arduino is slightly different from

board to board. The microcontrollers are usually of the ATMEL Company. Must know what IC the board has before loading up a new program from the Arduino IDE. This information is available on the top of the IC. For more details about the IC construction and functions, open the data sheet.



#### **ICSP pin**

Mostly, ICSP (12) is an AVR, a tiny programming header for the Arduino consisting of MOSI, MISO, SCK, RESET, VCC, and GND. It refer to as an SPI (Serial Peripheral Interface), which could be considered as an "expansion" of the output. Slaving the output device to the master of the SPI bus.



#### **Power LED indicator**

This LED should light up when plugging the Arduino into a power source to indicate that the board is powered up correctly. If this light does not turn on, then there is something wrong with the connection.



#### **TX and RX LEDs**

The two labels TX (transmit) and RX (receive) appears in two places on the Arduino UNO board:

First, at the digital pins 0 and 1, to indicate the pins responsible for serial communication. Second, the TX and RX led (13). The TX led flashes with different speed while sending the serial data. The speed of flashing depends on the baud rate used by the board. RX flashes during the receiving process.



## Digital I / O

The Arduino UNO board has 14 digital I/O pins (15) (of which 6 provide PWM (Pulse Width Modulation) output. These pins can be configured to work as digital input pins to read logic values (0 or 1) or as digital output pins to drive different modules like LEDs and relays. The pins labelled “~” can be used to generate PWM.



## AREF

AREF stands for Analog Reference. It is sometimes, used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the analogue input pins.

## 2.2 Node MCU

The ESP8266 is a fantastic chip for the Internet of Things projects. This chip has Wi-Fi connectivity, an onboard processor, and is compatible with the Arduino IDE as Shown in (figure 2-2-1).

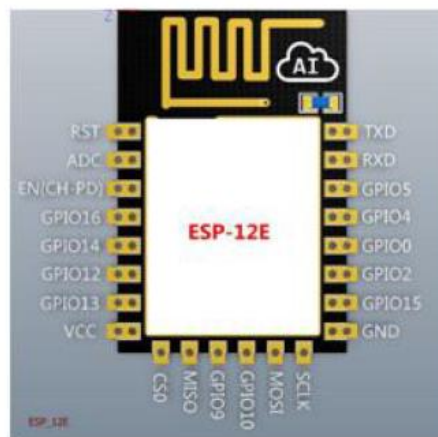


Figure 2-2-1 ESP 8266.



In the next table(2.2.1) and table (2.2.2)all pins name and functionality for it :

NO.	Pin Name	Function
1	RST	Reset the module
2	ADC	A/D Conversion result.Input voltage range 0-1v,scope:0-1024
3	EN	Chip enable pin.Active high
4	IO16	GPIO16; can be used to wake up the chipset from deep sleep mode.
5	IO14	GPIO14; HSPI_CLK
6	IO12	GPIO12; HSPI_MISO
7	IO13	GPIO13; HSPI_MOSI; UART0_CTS
8	VCC	3.3V power supply (VDD)
9	CS0	Chip selection
10	MISO	Salve output Main input

Table 2-2-1 (Node MCU pins numbers, names, and functions)

11	IO9	GPIO9
12	IO10	GPIO10
13	MOSI	Main output slave input
14	SCLK	Clock
15	GND	GND
16	IO15	GPIO15; MTDO; HSPICS; UART0_RTS
17	IO2	GPIO2; UART1_TXD
18	IO0	GPIO0
19	IO4	GPIO4
20	IO5	GPIO5
21	RXD	UART0_RXD; GPIO3
22	TXD	UART0_TXD; GPIO1

Table 2-2-1 (Node MCU pins numbers, names, and functions)

Node MCU is an open-source programmable board that provides the Internet stuff, which allows to connect things and to understand each other through the Internet the words mean things are all smart devices like:

Television, mobile phones, watches, glasses, alarms, surveillance, and other devices that can connect to each other over the Internet. This panel features a 12-ESP2866 chip. This painting is entirely new, especially in our Arab world. Where it first appeared in 2014.

## **Second Edition (V 1):**

### **Parties:**

The panel has ten terminals (D1-D10) that can be used as input or output and supports PWM. It also has a D0 terminal that does not support PWM and has a tip (A0) that can be used as analogue input. Analog

(3.3V) 3.3 volts and three (10v - 3.3v) VIN to the addition and four ends to the ground (GND).

### **Keys:**

The panel has two push button keys:-

- \* The first key (button flash):

It is used when installing the Node MCU system.

- \*The second key (button REST):

Used when uploading a program (code) new on Node MCU shown in (figure 2-2-2).



Figure 2-2-2 (NodeMCU 8266)

## 2.3 Temperature & humidity sensor

DHT11 Temperature & Humidity Sensor features a temperature & humidity sensor complex with a calibrated digital signal output it ensures high reliability and excellent long-term stability, this sensor includes a resistive-type humidity measurement component and an (NTC) temperature measurement component this sensor offering excellent quality, fast response, anti-interference ability and cost-effectiveness. Consists of 4 pins as Shown in (Figure2-3-1).

### 2.3.1 Technical Specifications:

- 3 to 5V power and I/O.
- 20-80% humidity readings with  $\pm 5\%$  accuracy.
- 0-50 ° c temperature readings with  $\pm 2^{\circ}\text{C}$ .
- 1 Hz sampling rate (one reading every second).

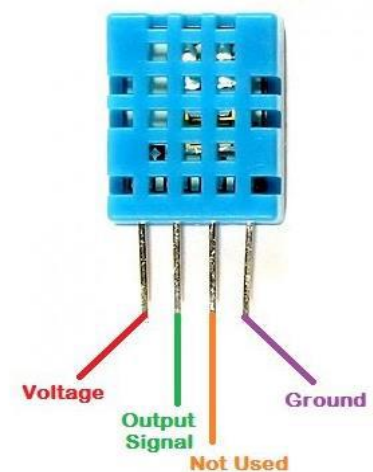


Figure 2-3-1DHT11 pins

### 2.3.2The types of DHT sensors:

The main specifications and differences between the two sensors as shown in Figure 2-3-2 DHT11 and DHT22 and table 2-3-1.



Figure 2-3-2 (DHT11 & DHT22)

Sensor Name	Temperature Range	Humidity Range	Sampling Rate	Body Size	Operating Voltage	MaxCurrent
DHT11	0-50°C / ±2°C	20–80%/±5%	1Hz (one reading every second)	15.5mm x 12mm x 5.5mm	3-5V	2.5mA
DHT22	-40 - 125°C ±0.5°C	0–100% / ±2-5%	0.5Hz (one reading every two second)	15.1mm x 25mm x 7.7mm	3-5V	2.5mA

Table 2-3-1(compare between DHT11 & DHT22)

### 2.3.3 NTC Thermistor meaning:

The component responsible for measuring the temperature.

- Thermistors are temperature-sensing elements made of semiconductor material in order to display significant changes in resistance in proportion to small changes in temperature. This resistance can be measured by using a small and measured direct current passed through the thermistor in order to measure the voltage drop produced.
- For measuring humidity, they use the humidity sensing component which has two electrodes so as the humidity changes, the conductivity of the resistance between these electrodes changes. This change in resistance is measured and processed by the IC which makes it ready to be read by a microcontroller  
Shown in (Figure 2-3-3).

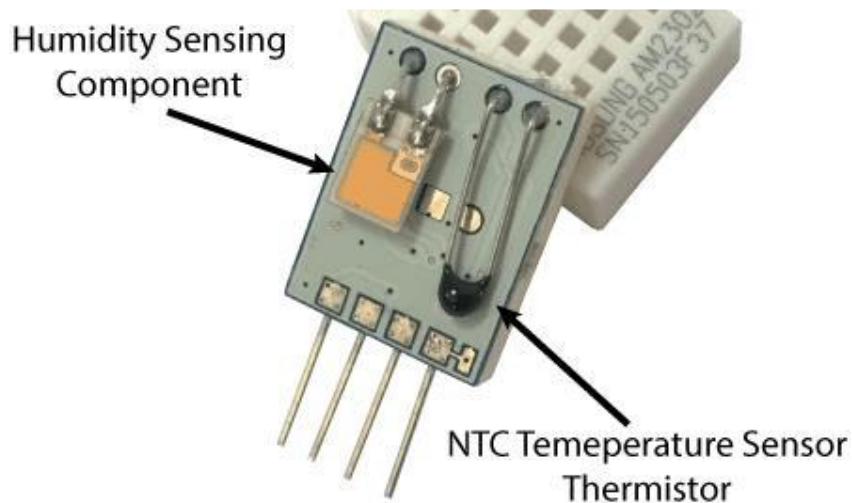


Figure 2-3-3 NTC Thermistor

## 2.4 Body temperature sensor:

Temperature control (thermoregulation) is part of a homeostatic mechanism that keeps the organism at optimum operating temperature, as the temperature affects the rate of chemical reactions. In humans, the average internal temperature is 37.0 °C (98.6 °F), though it varies among individuals. However, no person always has precisely the same temperature at every moment of the day. Temperatures cycle regularly up and down through the day, as controlled by the person's circadian rhythm. The lowest temperature occurs about two hours before the person wakes typically up. Additionally, temperatures change according to activities and external factors.

### 2.4.1 The types of body temperature sensors:

Various kinds of body temperature sensors available with deference local sensor accuracy, operating temperature range, supply voltage (Max & Min), supply current, sensor gain and output impedance as shown in table 2-4-1

LM35	TMP36	TMP35
Temperature Range: -55:150 °C	Temperature Range: -40:125°C	Temperature Range: 10:125°C
Accuracy : 0.5 °C	Accuracy : 1°C	Accuracy : 2°C
voltage level power: 4 : 30v.	voltage level power: 2.7 : 5.5 v	voltage level power: 2.7 : 5.5 v
Supply Voltage: 5	Supply Voltage: 5	Supply Voltage: 5

Table 2-4-1 (The different types of body temperature sensors)

## 2.5 Pulse Sensor (Heart Sensor):

Pulse Sensor is a well-designed plug-and-play heart-rate sensor for Arduino. It can be used by Students, artists, athletes, makers, and game & mobile developers who want to incorporate live heartrate Data into their projects easily. The sensor clips onto a fingertip or earlobe and plugs right into Arduino with some Jumper cables. It also includes graphs pulse in real time Pulse Sensor works with either a 3V or 5V Arduino as shown in (figure 2-5-1).

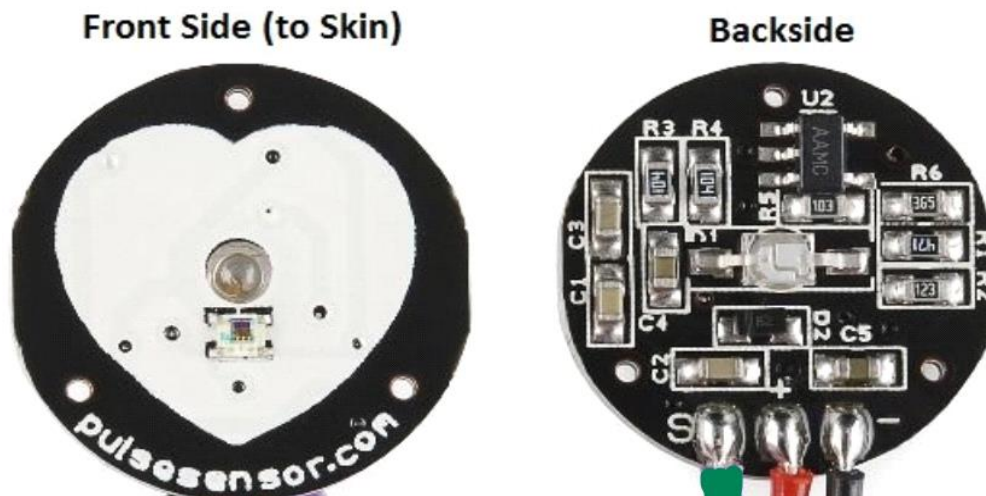


Figure2-5-1 PulseSensor

### 2.5.1 Specification of Pulse sensor:

Diameter = 0.625" (~16mm)

Overall thickness = 0.125" (~3mm)

Working Voltage = 3V to 5V

Working Current = ~4mA at 5V

## 2.5.2Pulse Sensor Kit Contents:

As shown in (figure 2-5-2) explaining the component of pulse sensor:

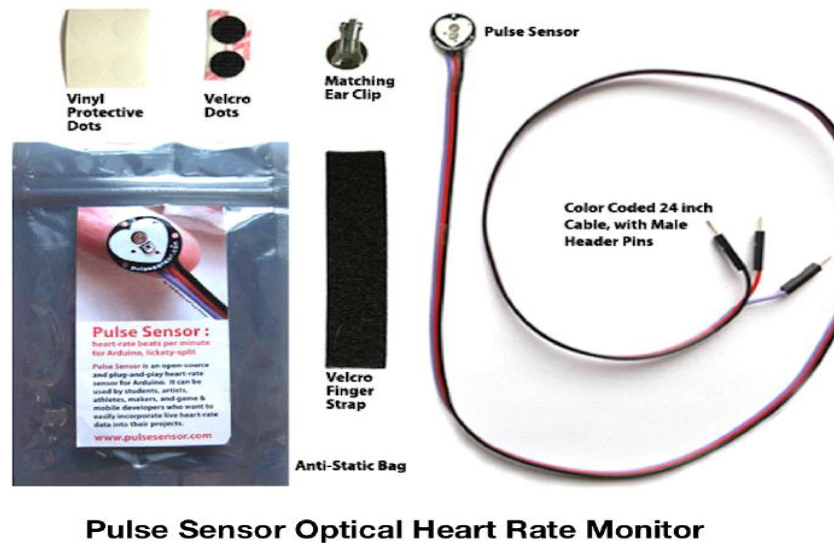


Figure 2-5-2 Pulse Sensor Kit Contents

1. A 24-inch Color-Coded Cable, with (male) header connectors which makes it easy to embed the Sensor into project, and connect to an Arduino. No soldering is required.
2. An Ear Clip perfectly sized to the sensor. We searched many places to find just the right clip. It can be hot glued to the back of the sensor and easily worn on the earlobe.
3. 2 Velcro Dots These are 'hook' side and are also perfectly sized to the sensor. Velcro dots Very useful if you want to make a Velcro (or fabric) strap to wrap around a fingertip.
4. Velcro strap to wrap the Pulse Sensor around your finger.



5. Transparent Stickers. These are used on the front of the Pulse Sensor to protect it from oily fingers and Sweaty earlobes.
6. The Pulse Sensor has three holes around the outside edge which make it easy to sew it into almost anything.

### 2.5.2pin in pulse sensor:

Absolute Maximum Ratings	Min	Typ	Max	Unit
Operating Temperature Range	-40		+85	°C
Input Voltage Range	3		5.5	V
Output Voltage Range	0.3	Vdd/2	Vdd	V
Supply Current	3		4	mA

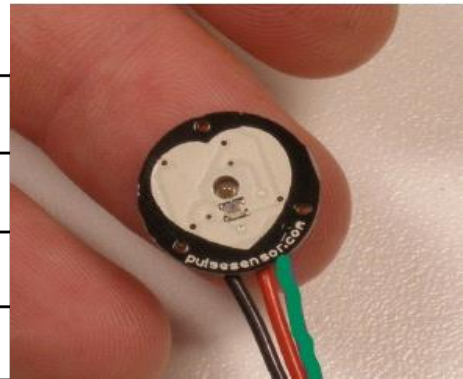


Figure 2-5-9 Wires of Pulse Sensor

1. RED wire = supply, ( 3V up to 5V )
2. BLACK wire = GND : ( - : ground )
3. GREEN wire = Signal: ( S: signal, connected to any of your microcontroller's digital pin.).

### 2.5.3 Features Of Pulse sensor:

- It includes Kit accessories for high-quality Sensor readings.
- Designed for Plug and Play.
- Small size and embeddable into wearables.
- It Works with 3Volts or 5Volts.
- Well-documented Arduino library.

## Project Prototype:-



## Chapter 3

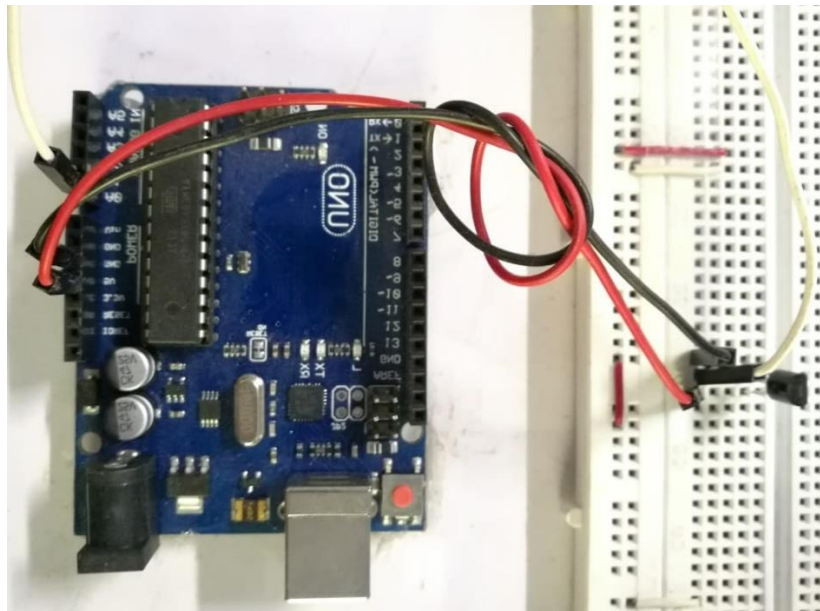
### Design and Implementations

---

#### 3.1 Design of sensors:

##### 3.1.1 Design of LM 35 (Body Temperature Sensor):-

The connection between sensor Lm35 and Arduino Uno: LM35 has three pins. The positive pin must connect to positive part of Arduino (5V) by using wire and negative part of the sensor must connect to the negative part of Arduino (GND (Ground)). The middle pin of the sensor that is responsible for transferring the data, which connects to the analogue part of Arduino (A0) as below (Figure 3.1.1).



**Figure 3.1.1** Connection between LM35 and Arduino

##### 3.1.2 DHT 11 (Humidity & Temperature Incubator Sensor):-

The connection between sensors DHT11 and Arduino Uno: DHT11 has four pins, the positive pin must connect to positive part of Arduino (5V) and negative part of the sensor must connect to the negative part of the Arduino (GND (Ground)). DHT11 has a pin that is responsible for transferring data

responsible for the, which connects to digital part of Arduino (2), as a below (Figure 3.1.2).

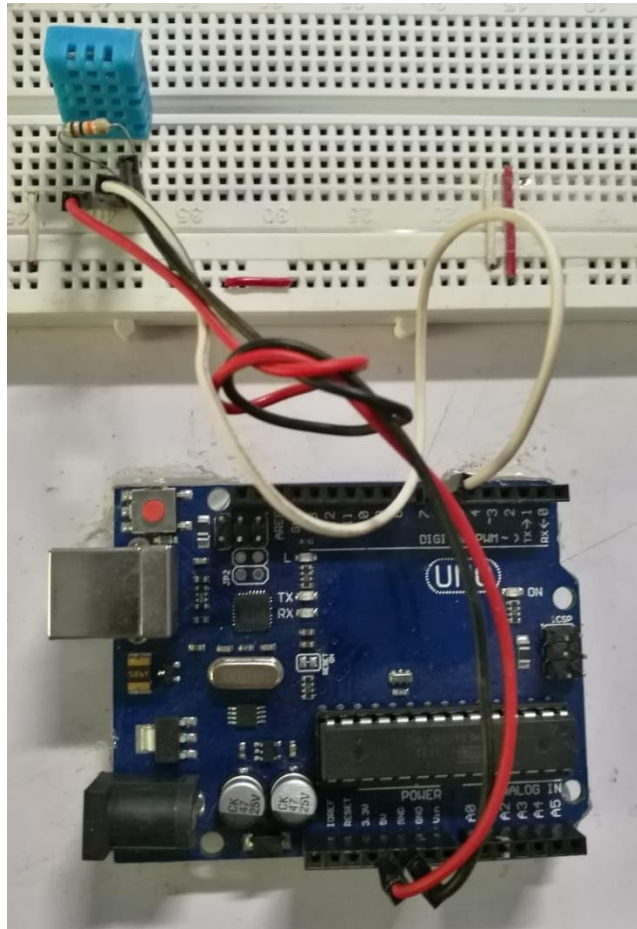


Figure 3.1.2 Connection between DHT11 and Arduino

### 3.1.3 Pulse sensor (Heart Sensor):-

The connection between sensor pulse sensor and Arduino Uno: Pulse sensor has three parts, The positive part must connect to positive part of Arduino (5v) ,The negative part of the pulse sensor must connect to the negative part of the Arduino device(GRD(Ground)). However the part of the sensor responsible for the transferring the data, which connected to the analogue part of Arduino (A0) as a below (Figure 3.1.3) this sensor will be better if it connects to analogue pin.

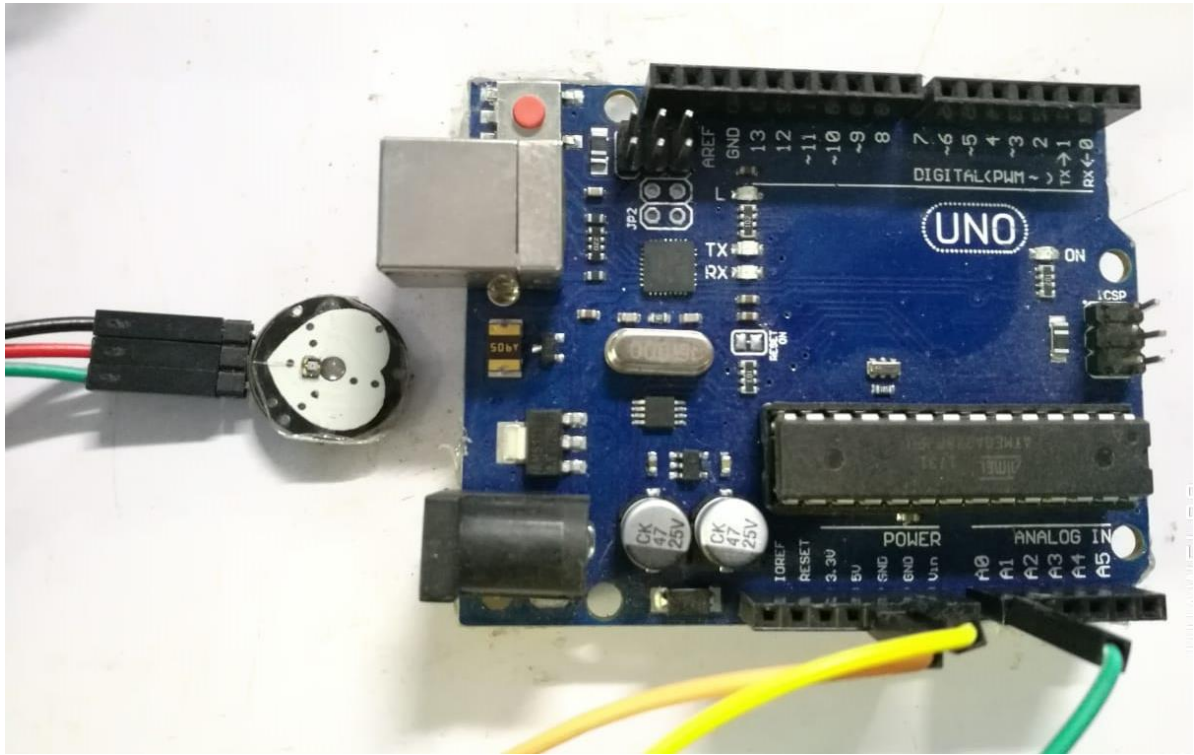


Figure 3.1.3 Connection between pulse sensor and Arduino

### 3.1.4 The Three Sensors together using the ESP and the Arduino:-

- LM35 and DHT11 Connect to ESP to avoid the problems with the pulse sensor because when it connects with the other sensor that makes some noise -So the pulse sensor connect with the Arduino Uno only.
- The Arduino must connect with the ESP to upload the data to the server.

The connection between LM35 and DHT11 with ESP:

The connection between sensor LM35 and ESP :As saying before that LM35 has three pins The positive pin must connect to positive part of ESP



but at this time connect to (3V)not (5V) Because ESP gives just 3 volts ,and negative part of the sensor must connect to the negative part of ESP(GRD(Ground)). The middle pin of the sensor that responsible for transferring the data, which connects to the analogue part of ESP (A0)

The connection between sensor DHT11 and ESP:As said before that DHT11 has four pins, The positive pin must connect to positive part of ESP but at this time connect to (3V) not (5V) because ESP gives just 3volts, and negative part of the sensor must connect to the negative part of ESP(GRD(Ground)). DHT11 has a pin that responsible for transferring data responsible for the, which connects to digital part of ESP (2) as a below (Figure 3.1.2).

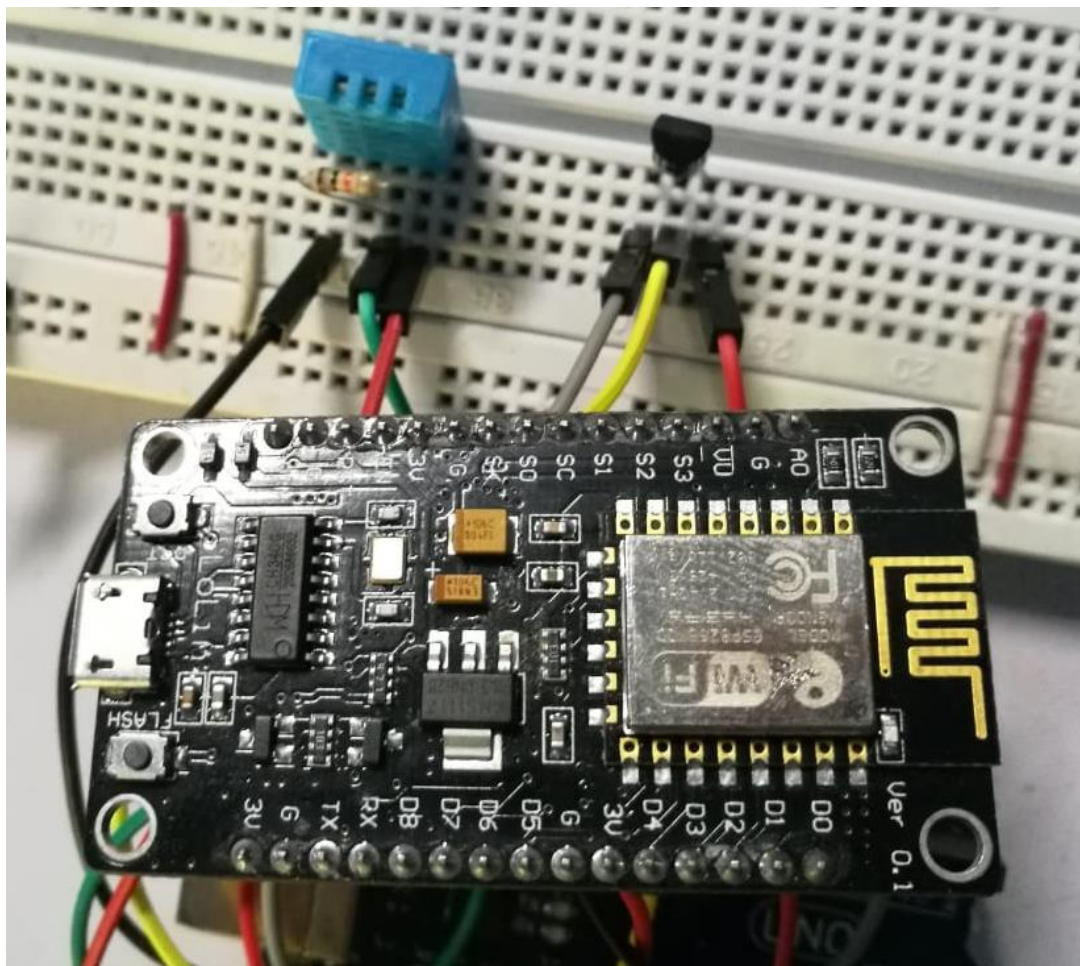


Figure 3.1.4DHT11and LM35 with ESP

## 3.2 Hardware Implementation

### 3.2.1 LM35 Implementation:

-In this sensor there are no any libraries to declare. But there is equation used to convert voltage result to Celsius result that will be the body temperature.

$$T = ((X * 5.0) / 1024) * 100, \text{ the equation to calculate Body Temperature.}$$

Equation 3-2-1 Body temperature

### 3.2.2 DHT11 Implementation:

-DHT11 have a library must declare at Arduino IDE, This libraries have two main function to get the Humidity & Temperature for the sensor

```
#include <dht.h>
```

To determine Humidity It should calculate  
The Td (dew point), RH (relative humidity), T (dry bulb).

### 3.2.3 Pulse Sensor (Heart Sensor) Implementation:

-Pulse sensor has a library must declare at Arduino IDE, This is a library is responsible for running pulse sensor.

```
#include <PulseSensorPlayground.h>
```

A condition has been added in the code writing stipulating that if the wave frequency reaches 500 Hz. This frequency will count as one pulse

### 3.2.4 LM35 and DHT11 with ESP Implementation:

After LM35 and DHT11 connect with (NODE MCU 1.0 ESP -12E MODULE).ESP has a library must declare at Arduino IDE:

```
#include <ESP8266WiFi.h>
#include <SoftwareSerial.h>
SoftwareSerial NodeSerial(D2,D3);
```

### **Explaining of libraries:**

- #include <ESP8266WiFi.h>: this library define ESP this is Wi-Fi.
- #include <SoftwareSerial.h>: this library allow Arduino send data to ESP by RX & TX, RX and TX are pins in Arduino connected to ESP.
- SoftwareSerial NodeSerial(D2,D3): D2=RX :Receiver AND D3=TX :sender (ESP) connected with D2=RX : Receiver AND D3=TX: sender (Arduino).



## **A Function that we used at program Arduino IDE & Explaining the function of the code of ESP and Sensors:**

```
pinMode(D2,INPUT);
// D2 is acts sender that send data from ESP and Arduino
pinMode(D3,OUTPUT);
// D3 is acting receiver that receive data from ESP and Arduino
X=analogRead(A0);
// creating a variable called X to read the data of sensor LM35 that allocated
at pin A0
T= ((X*2.5)/1024)*100;
// this equation1 that convert voltage to Celsius
float h = dht.readHumidity();
// this function to read the Humidity that DHT11 is measured it and
store it at a variable (h)
float t = dht.readTemperature();
// this function to read the Temperature of incubator that DHT11 is
measured it and store it at a variable (t)
NodeSerial.print(0xA0);
NodeSerial.print("\n");
// this is pass between Arduino IDE and ESP that is receive data
while(NodeSerial.available()>0){ }
// making a check if sensor read any data
float val = NodeSerial.parseFloat();
//(Parse Float) this function reads data from sensors (receive data and
in the same time send data) and save it into variable name is Val
```

## 3.3 Software Implementation

### 3.3.1 Blynk Implementation:

After connecting two sensors (LM35 and DHT11) are connected with ESP and Pulse sensor with Arduino, to show the measurements of this sensors needed to use the Blynk app to show measurements as a test.

- **Blynk APP:** is a Platform with iOS and Android apps to control Arduino, over the Internet. Blynk is simple to use and set up all necessary environment (Server, client and Board). Blynk allows the user to Interact with Pins (Digital and analogue pin), Send and receive data from hardware.
- **Blynk connected by:**
  - Ethernet
  - Wi-Fi
  - Bluetooth
  - Serial USB
- **There are three major components in the platform of Blynk :**
  1. Blynk app: allows for creating amazing interfaces for projects using various widgets we provide.
  2. Blynk Server: responsible for all the communications between the Smartphone and hardware. There are a Blynk Cloud and a private\_Blynk server locally. It is an open-source, could easily handle thousands of devices and can even be launched on a Raspberry.
  3. Blynk Libraries: for all the popular hardware platforms - enable communication with the server and process all the incoming and out coming commands.

- **Description of Blynk working :**

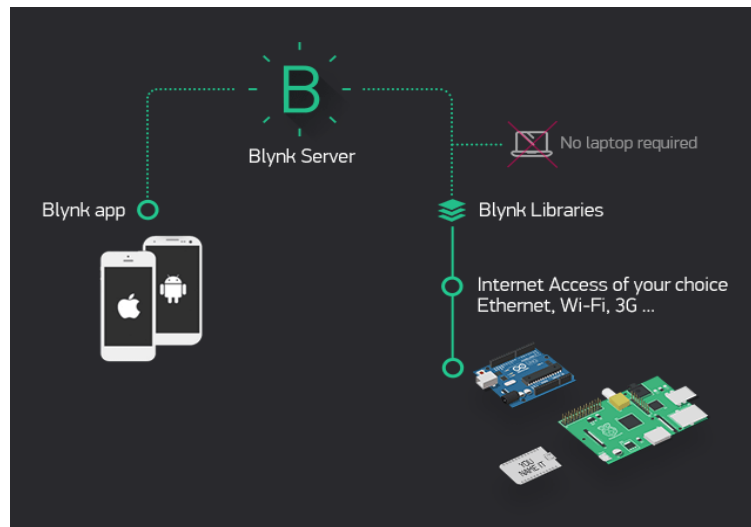


Figure 3-2-5 Blynk Working

- **Pros of Blynk:**

- Easy and simple.
- Many features.
- No additional hardware.
- Active community.
- Cool GUI.

Blynk have Libraries must declare at Arduino IDE, this libraries are

```
#define BLYNK_PRINT Serial

#include <SPI.h>

#include <BlynkSimpleEsp8266.h>
```

## Explaining of libraries of Blynk:

- `#define BLYNK_PRINT Serial`: Defines the object that use for printing.
- `#include <SPI.h>`: used by microcontrollers for communicating with one or more peripheral devices.
- `#include <BlynkSimpleEsp8266.h>`: define Blynk app to ESP (define them each one).

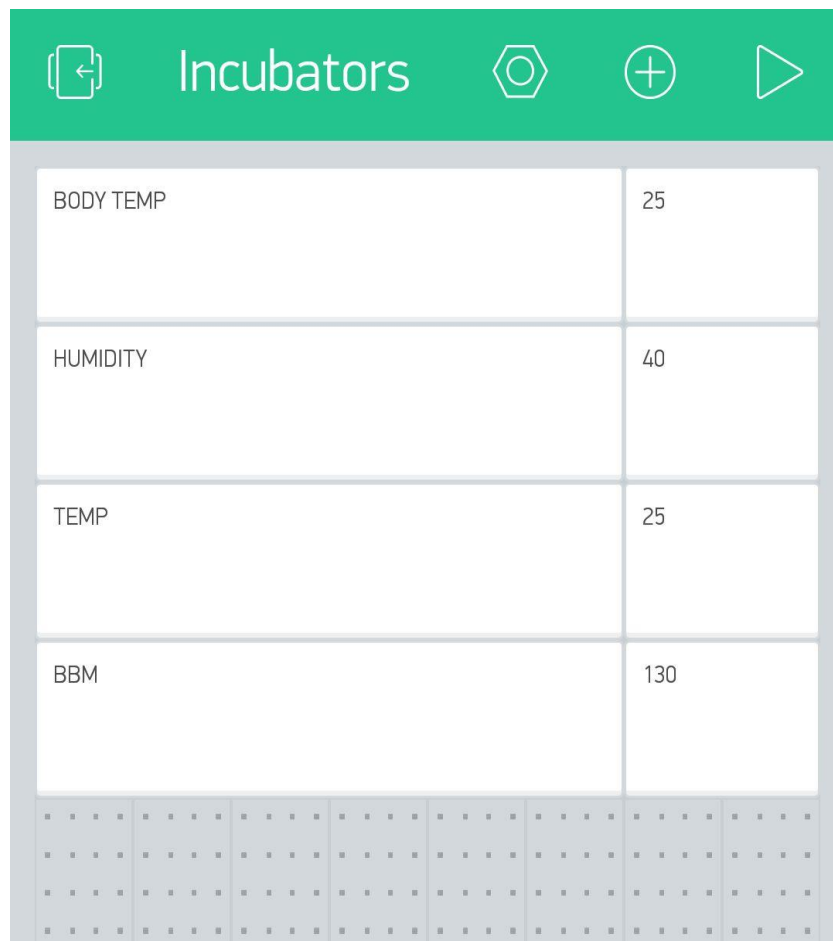
## A Function that use at program Arduino IDE:

```
Blynk.virtualWrite(V4, h);  
  
Blynk.virtualWrite(V6, t);  
  
Blynk.begin(auth, ssid, pass);  
  
Blynk.run();  
  
Blynk.virtualWrite(V5,T);  
  
Blynk.virtualWrite(V3,val);
```

## Explaining the function of code that is used at Arduino IDE:

- `Blynk.begin(auth, ssid, pass)`: `auth`(authentication code that the application will send to communicate with the ESP), `ssid`(The name of the network),`pass`(the password of the network).
- `Blynk.run()`:all blynk Function happens here.
- `Blynk.virtualWrite(V3,val)`: display Val (pulse sensor)in area name is V3that is created at Blynk app.
- `Blynk.virtualWrite(V4, h)`: display h (Humidity)in area name is V4 that create at Blynk app.

- Blynk.virtualWrite(V5,T): T(Body Temperature)in area name is V5 that create at Blynk app.
- Blynk.virtualWrite(V6, t): (Temperature)in area name is V6 that create at Blynk app.



Incubators	
BODY TEMP	25
HUMIDITY	40
TEMP	25
BBM	130

**Figure 3-2-6 Blynk app measurement**

### 3-3-2 uploading measurements on Firebase database:

After displaying measurements at Blynk app, the measurements upload at firebase database.

- **Firebase database:** Store and sync data with our NoSQL cloud database. Data is synced across all clients in real-time, and remains available when the app goes offline.

Firebase database have libraries must declare at Arduino IDE:

```
#include <FirebaseArduino.h>

// This Library to connect firebase with Arduino

#define FIREBASE_HOST

// this library responsible for the name of Host at firebase

#define FIREBASE_AUTH

//: this library responsible for the Authentication at firebase
```

- **Functions That used at the program:**

```
Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
Firebase.setFloat("Temperature", t);
```

### Explaining the function of code:-

- `Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);`: `Firebase.Begin` has two parameters `Host` and `Auth`, `Host` the firebase database host, usually `X.firebaseio.com`, `Auth` Optional credentials for the database, a secret or token.
- `Firebase.setFloat("Temperature", t);`: Writes the float value to the node located at path, The path inside of database to the node to update.

After Uploading measurements on Firebase database, finally making an android app and IOS app belongs project to add and follow the data of Premature Babies.

### 3-3-3 Implementation phases on android and IOS app:

#### 1. Use Case Diagram for app:

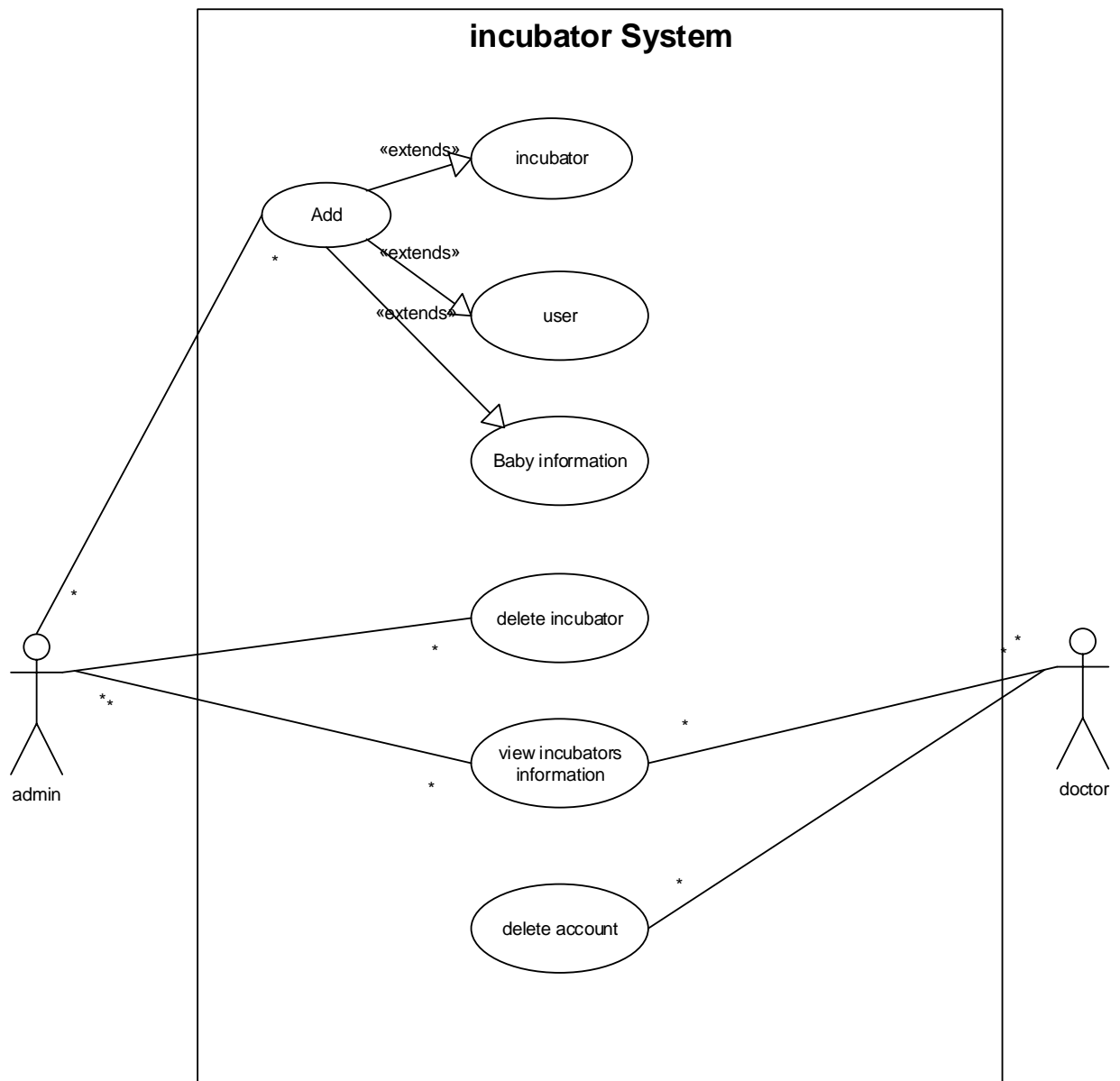


Figure 3-3-1 UML Use case for app

## 2. Use case narratives

Use Case Name	Incubator system
Use case ID	100
Primary Actors	Doctor as (admin), doctors
Description	First check the user The number and information of the incubators is shown It gives him a warning when a certain defect occurs in the temperature and heartbeat
Precondition	The doctor must be authorized
Basic flow of events	The doctor login in the system The system checks and gives the user the authority to choose a incubator and presents his information
Alternative flows	System doesn't identify an account

Figure 3-3-2 Use case narratives for app



### 3. Activity Admin Diagram:-

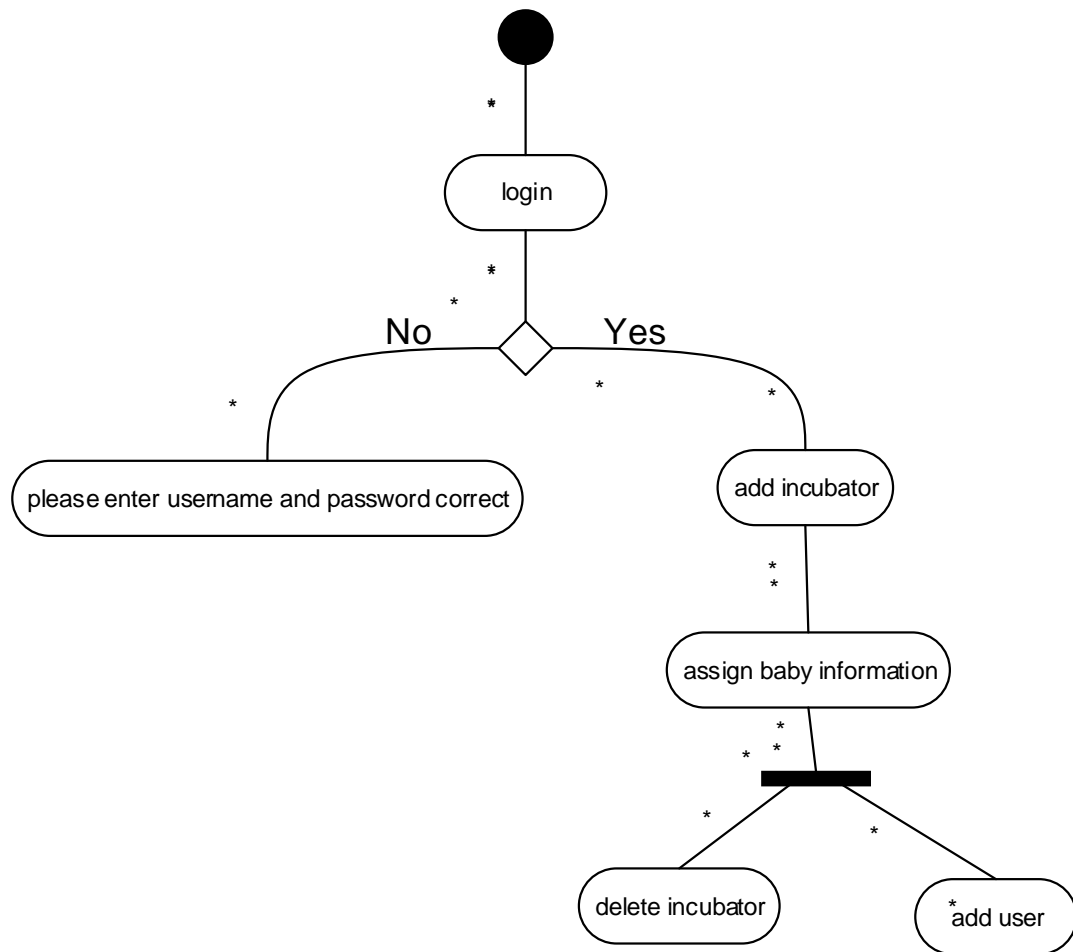


Figure 3-3-3 Activity Admin Diagram for app

#### 4. Activity User Diagram:-

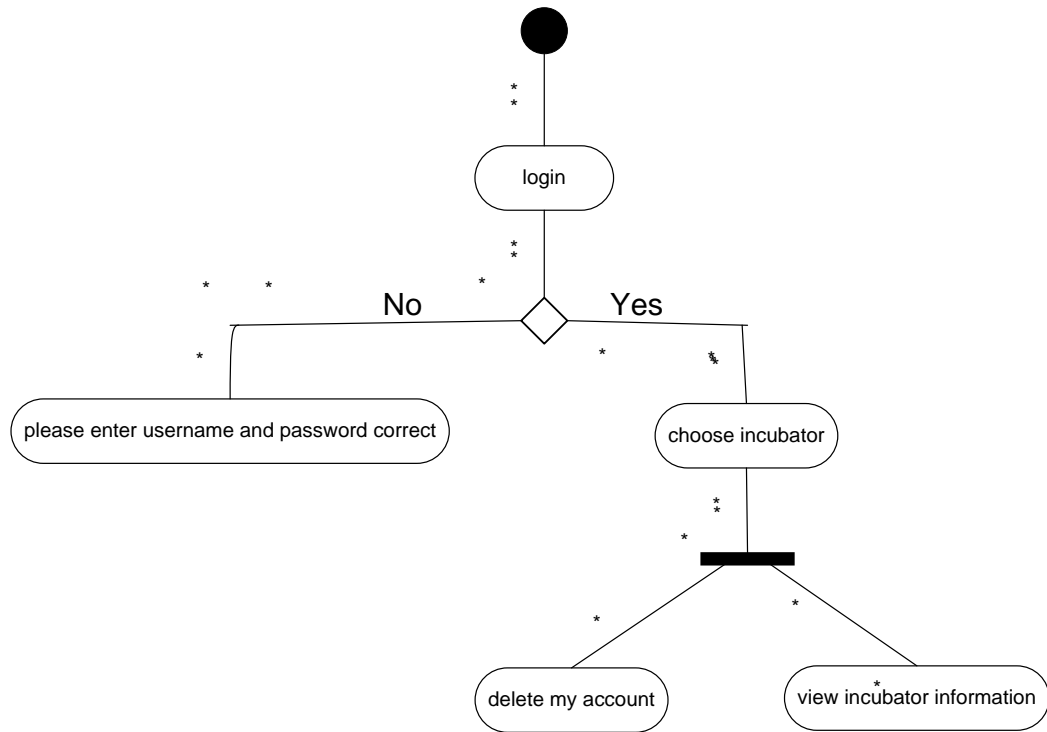


Figure 3-3-4 Activity User Diagram for app

#### 5. Sequence Diagram:-

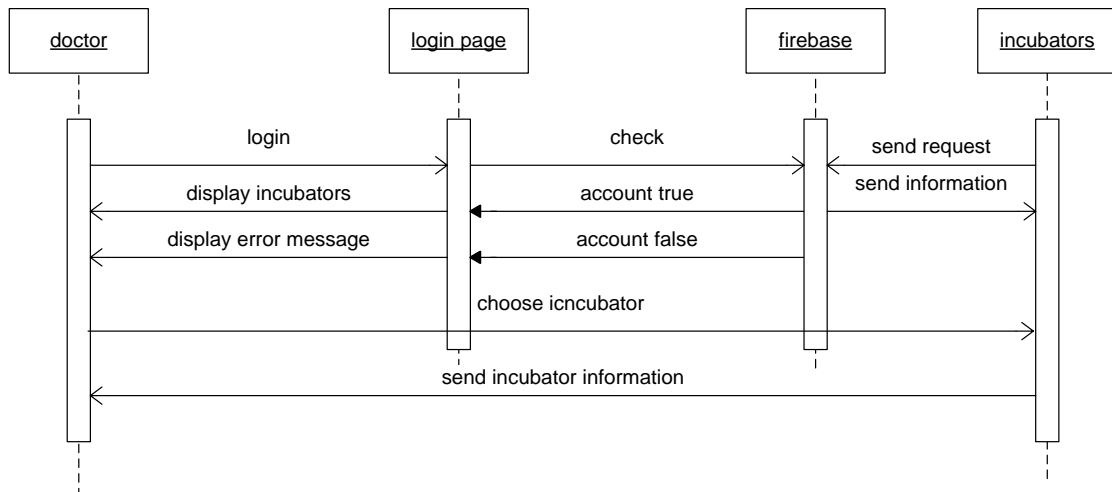


Figure 3-3-3 sequence diagram

## **6. User requirement:**

- **The incubator system should measure the child's temperature**
  - A. temperature incubator
  - B. heartbeat of the child
  - C. Admin should control the system
    - I. add a new incubator
    - II. delete incubator and add a user

## **7. System requirement:**

- A. System should check in firebase username and password
- B. After the system check if the doctor is present open the home page
- C. The system will be shown to the doctor incubators existing  
System will send alerts to the doctor In case of increased temperatures and heartbeat

## **Android and IOS app Implementation:**

Incubators monitoring app give admin the permission to add and delete incubators and the data that belongs to Babies.

Incubators app includes views as an Admin (Login activity, Add baby, Add incubator, Add user, Show incubators, show data, Remove incubator, and Remove user).

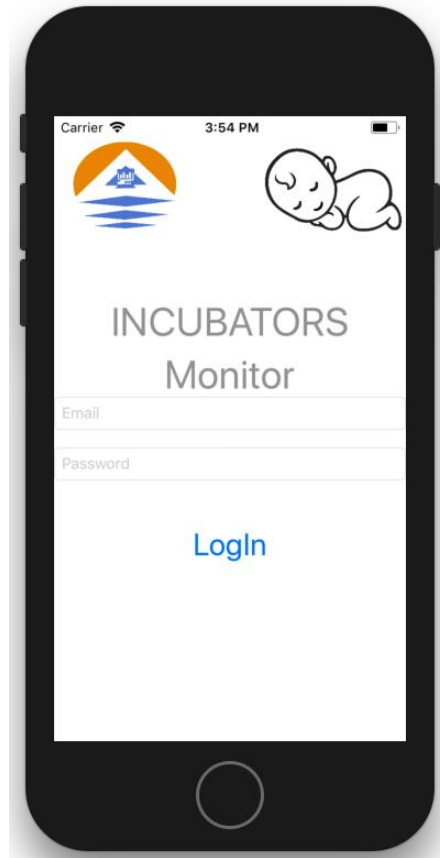


Figure 3-3-4 Login activity

- The process, in the Login activity takes Mail and Password that inserted then firebase check if Mail and Password found and registered on the firebase, then the firebase verified it as shown (figure 3-3-4 Login activity).
- The Post Condition, after verifying the Incubators activity that includes incubators that used as shown (figure 3-3-5 Incubators activity).

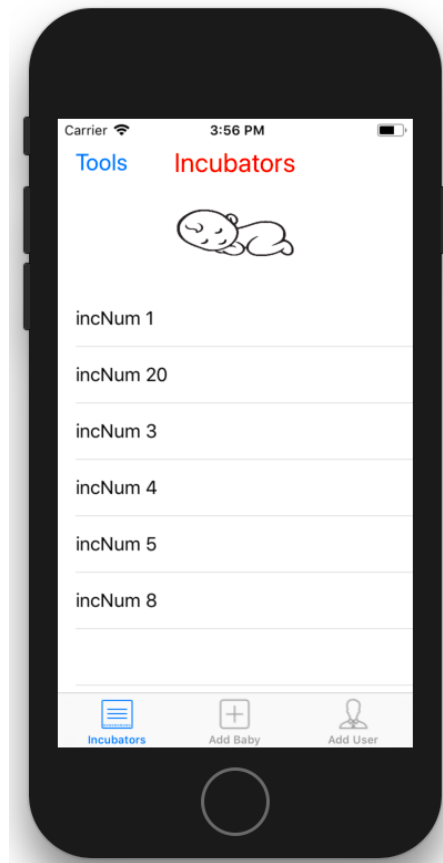


Figure 3-3-5 Incubators

- The pre-condition, after the user successes to login as shown in (Figure 3-3-4 Login activity).
- The process, in the Incubators activity that appearing Tab Bar activity that includes incubators that used as shown as shown in (Figure 3-3-5 Incubators).
- The Post Condition, the Incubators activity that includes incubators that used as shown (Figure 3-3-5 Add-Baby activity).

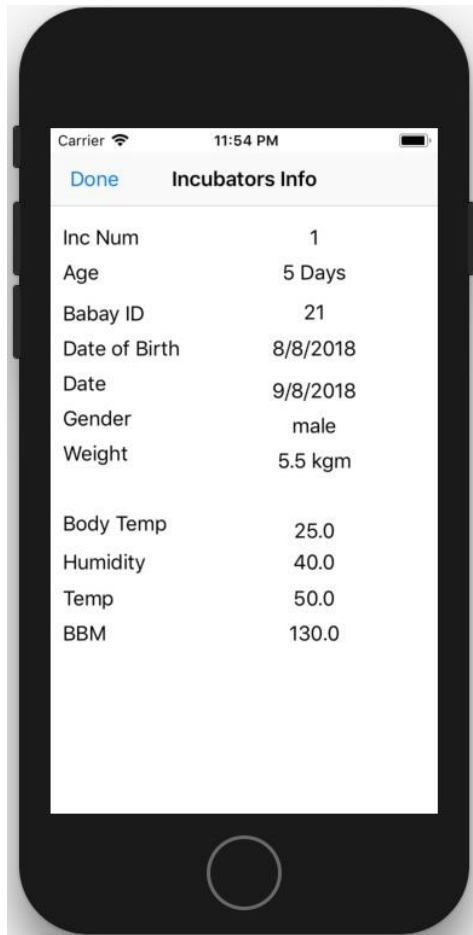


Figure 3-3-6 DataOfIncubators

- The pre-condition, the user was in the Incubators Activity then click on any Incubator as shown in (Figure 3-3-5 Incubators).
- The process, in the DataOfIncubators activity that appearing when click on any Incubator that includes incubators data as shown as shown in (Figure 3-3-6 DataOfIncubators).
- The Post Condition, the Add-Baby activity that includes (incubator number, child name, ID, date, Birthday of baby, weight and Gender of baby). as shown (Figure 3-3-7 Add-Baby activity).

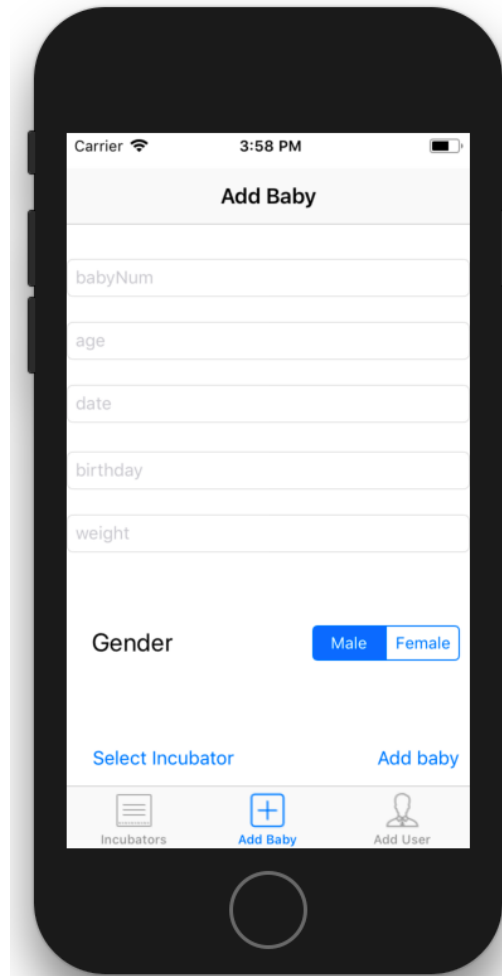


Figure 3-3-7 Add-Baby

- The pre-condition, the user was in the DataOfIncubators Activity then when he swap to the next Activity that appearing on the Tab Bar as shown in (Figure 3-3-6 DataOfIncubators activity).
- The process, in the Add-Baby activity on Tab Bar activity that includes (incubator number, child name, ID, date, Birthday of baby, weight and Gender of baby), when the admin insert all data and click button (add baby) data

uploaded to firebase and send it to the incubator that want to insert data in it as shown in (Figure 3-3-7 Add-Baby).

- The Post Condition, the Incubators activity that includes incubators that used as shown in (Figure 3-3-8 Incubators activity).



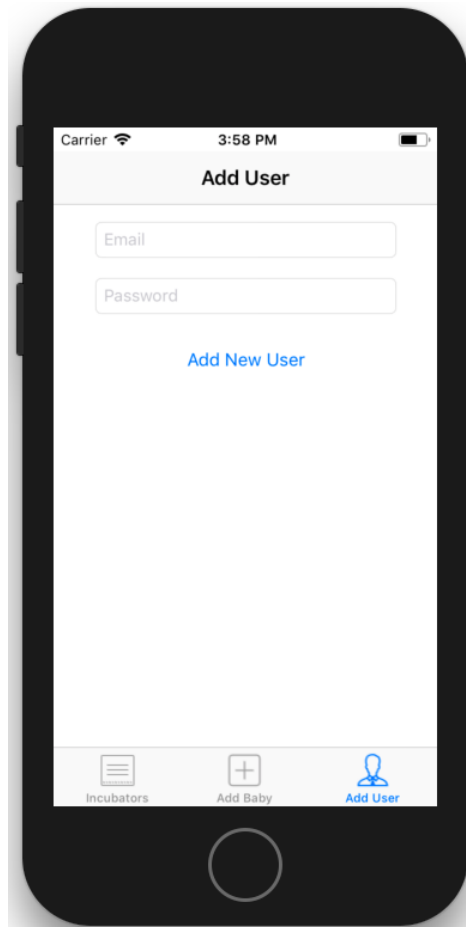


Figure 3-3-8 Add-User

- The pre-condition, the user was in the Add-Baby Activity then when he swap to the next Activity that appearing on the Tab Bar as shown in (Figure 3-3-7 Add-Baby activity).
- The process, in the Add-Baby activity on Tab Bar activity that includes (Mail of user, Password of user) When the admin insert the data and click button (add new user) data uploaded to as shown in (Figure 3-3-8 Add\_User).
- The Post Condition, the Incubators activity that includes incubators that used as shown in (Figure 3-3-5 Incubators activity).

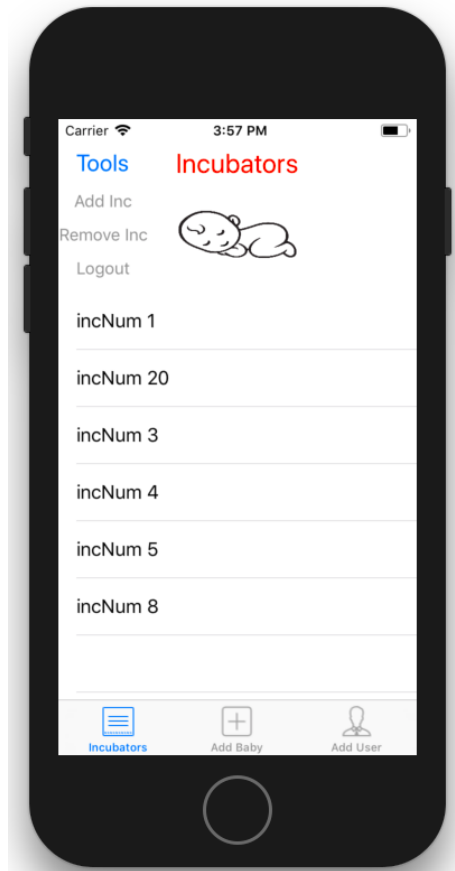


Figure 3-3-9 Tools

- The pre-condition, the user was in the Add-User Activity then when he back to the Incubators as shown in (Figure 3-3-8 Add-User activity).
- The process, in the Tools drop down menu that includes (Add incubator, remove incubator and logout) as shown in (Figure 3-3-9 Tools).
- The Post Condition, the Incubators activity that includes Add\_incubators that used as shown in (Figure 3-3-10 Add\_Incubator activity).

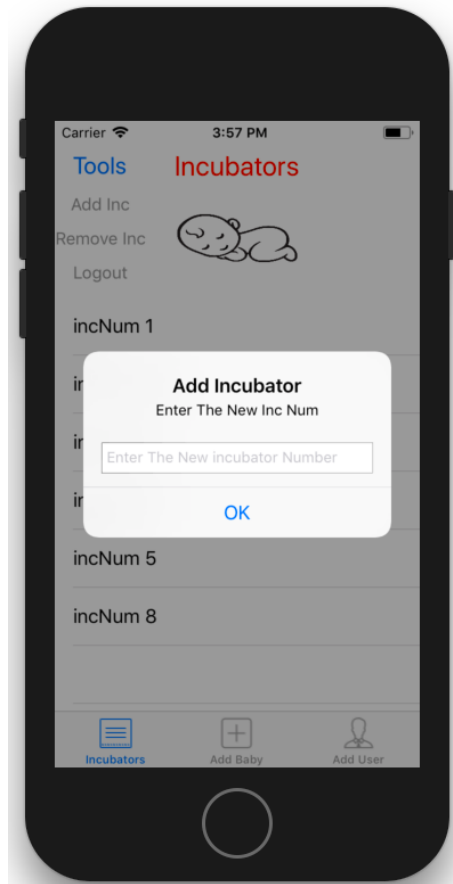


Figure 3-3-10 Add\_Incubator

- The pre-condition, the user was clicked on Tools as shown in (Figure 3-3-9 Tools\_Menu).
- The process, in the Add\_Incubator Dialog diagram that includes (Add incubator) when the user insert a new number of incubator and click ok button then check in firebase if this incubator founded or not if yes give a notification to say that the incubator used but if no the data send to firebase and store as shown in (Figure 3-3-10 Add\_Incubator).
- The Post Condition, the Remove\_Incubator Dialog diagram that includes incubators that used as shown in (Figure 3-3-11 Incubators activity).

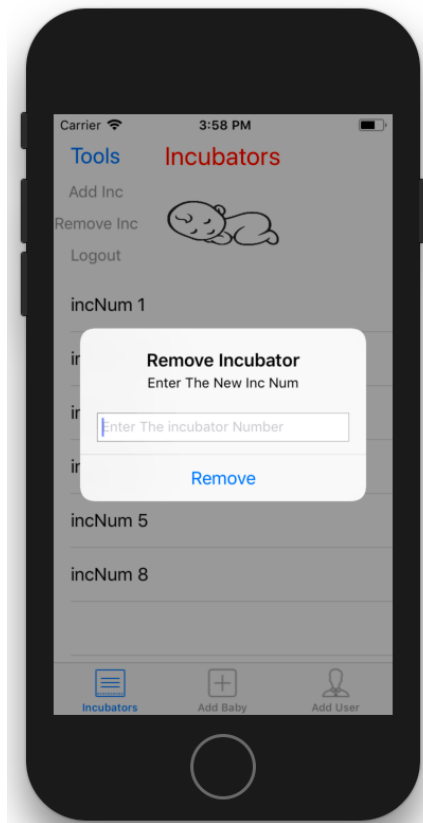


Figure 3-3-11 Remove\_Incubator

- The pre-condition, the user was clicked on Add\_Incubator as shown in (Figure 3-3-10 Add\_Incubator).
- The process, in the Add\_Incubator Dialog diagram that includes (Add incubator) when the user insert a the number of incubator that need to delete and click ok button then check in firebase if this incubator founded or not if yes show a Dialog diagram to confirm that if sure the user need to delete this incubator or no then if clicked yes it deleted from the firebase but if no give a

notification to say that this incubator not found as shown in (Figure 3-3-11 Remove \_Incubator).

- The Post Condition, the Remove\_Incubator Dialog diagram that includes incubators that used as shown in (Figure 3-3-12 Incubators activity).

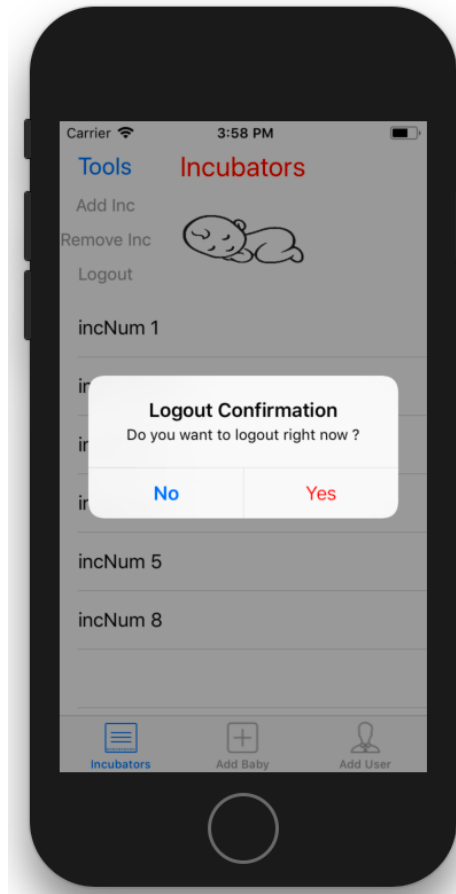


Figure 3-3-12 Logout

- The pre-condition, the user was clicked on [1] Remove\_Incubator as shown in (Figure 3-3-11 Remove \_Incubator).
- The process, in the Logout Dialog diagram that includes (Yes, No) when the user when click on yes he logout from the app and back to the Login Activity but when click no back to the last screen as shown in (Figure 3-3-12 Logout).
- The Post Condition, the Remove\_Incubator Dialog diagram that includes incubators that used as shown in (Figure 3-3-13 Incubators activity).

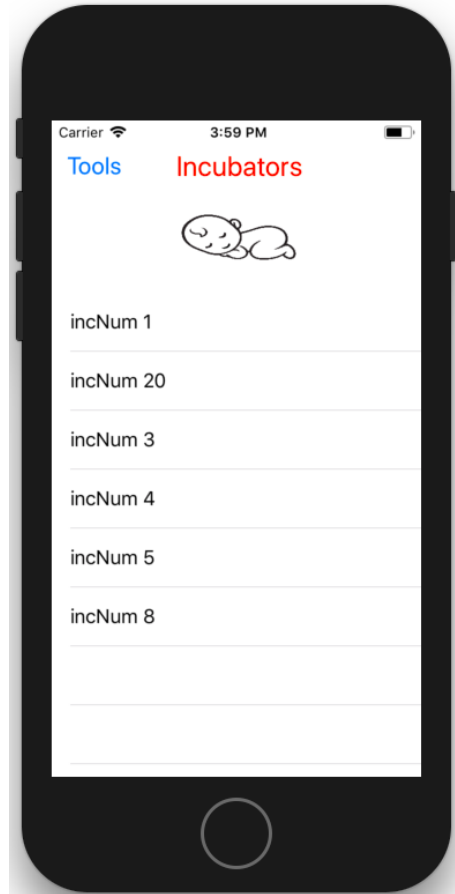


Figure 3-3-13 UserView

- The pre-condition, after the user successes to login as a normal User as shown in (Figure 3-3-4 Login activity).
- The process, in the Incubators activity that appearing that includes incubators that used as shown as shown in (Figure 3-3-13 Incubators).
- The Post Condition, the user was in the Incubators Activity then click on any Incubator as shown (Figure 3-3-6 DataOfIncubators).

## Chapter4

### 1. Conclusion

All in all this experience was very rewarding for us.

We all had to step into different roles, creative and technical, and it is a tribute to the flexibility of the team members that our resulting content is of a high standard.

#### 1.1 challenges

- Working with Hardware equipment and connect it to the mobile app was a new experience that required vast knowledge about its properties. Typically working with different languages as c, java and swift.
- To facilitate the learning process it used video tutorials, text tutorials, learning, materials and searching into each software documentation. It is a matter of time patience and hard work.

#### 1.2 The achievements

- During the process of brainstorming we managed to develop creative thinking and imagination capability.
- Now we know much more about Hardware and mobile development process.
- Co-operation between group members.
- Coming to the point that managing to know more about Hardware. How does it work? It is properties and objects.
- At the start of the project, aiming to add more idea for the project to Rescue Premature babies by making incubator most efficiency and help doctors, nurses to follow the condition of babies.



# Appendix

## 1. Appendix A –Hardware coding

```
#define BLYNK_PRINT Serial // Comment this out to disable prints and save space
#include <SPI.h> //to make esp main compant and any other component pranches
#include <ESP8266WiFi.h> // to define esp this is wifi
#include <BlynkSimpleEsp8266.h> //to define app Blynk to esp (define them each one )
#include <SoftwareSerial.h> //this lib allow arduino send data to esp by RX & TX
SoftwareSerial NodeSerial(D2,D3); //D2=RX:Receiver AND D3=TX:sender (ESP) pluge with D2=RX:Receiver AND D3=TX:sender (Arduino)
//#include <SimpleTimer.h>
#include <DHT.h> // define lib sensor DHT
float X,T; // Variable to equatation about sensor LM35
// You should get Auth Token in the Blynk App.
// Go to the Project Settings (nut icon).
char auth[] = "28cb9dcae6584ec18c9b6b7cbfa323bf";

// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[] = " "; //wifi user name
char pass[] = " "; // wifi pass

#define DHTPIN D1 // What digital pin we're connected to

// Uncomment whatever type you're using!
#define DHTTYPE DHT11 // DHT 11
//#define DHTTYPE DHT22 // DHT 22, AM2302, AM2321
//#define DHTTYPE DHT21 // DHT 21, AM2301

DHT dht(DHTPIN, DHTTYPE); // object from lib to define what is pin you are connected and what type sensor you use
```

### appendix a-1 three sensor coding

```

void sendSensor()
{
  float h = dht.readHumidity();
  float t = dht.readTemperature(); // or dht.readTemperature(true) for Fahrenheit
  Serial.println(h);                //Serial. ____ to show as a serial not LCD
  Serial.println(t);
  if (isnan(h) || isnan(t)) {       //This condetion say if arduino didn't receive data from sensor show this message
    Serial.println("Failed to read from DHT sensor!");
    return;
  }
  // You can send any value at any time.
  // Please don't send more that 10 values per second.
  Blynk.virtualWrite(V4, h);        //V4 mean the humididty show in this area ... this area you decided in app blynk and also you named .
  Blynk.virtualWrite(V6, t);
}

```

## appendix a-2 three sensor coding

```

void setup()
{
  Serial.begin(9600); // See the connection status in Serial Monitor
  NodeSerial.begin(4800); //this to begin function NodeSerial At the top of the code
  Blynk.begin(auth, ssid, pass); //this to begin blynk and take the data from At the top of the code (auth, ssid, pass)
  // define which the receiver and sender
  pinMode(D2, INPUT);
  pinMode(D3, OUTPUT);

  dht.begin();

  // Setup a function to be called every second
  timer.setInterval(1000L, sendSensor);
}

void loop()
{
  Blynk.run(); // Initiates Blynk
  timer.run(); // Initiates SimpleTimer
  X=analogRead(A0); //this data cable pluge in pin A0
  T=((X*2.5)/1024)*100; //this is equation about body temperature
  Serial.println(T);
  Blynk.virtualWrite(V5,T); //display T in area name is V5 ..... V5 CREAT BY BLYNK APP

  NodeSerial.print(0xA0); //this is bass between Arduino and ESP in this case this is bass wil be receiver
  NodeSerial.print("\n");
}

```

## appendix a-3 three sensor coding

## 2. Appendix B–IOS coding:

```
2 func onLogin(){
3     if emailField.text == "" && passwordField.text == "" {
4         UIAlertController.showAlert(self, title: "Error", message: "all Fields Required")
5     }
6     else{
7         if emailField.text != nil && passwordField.text != nil {
8             Auth.auth().signIn(withEmail: emailField.text!, password: passwordField.text!) { (user, error) in
9                 if user != nil{
10                     if let uid = user?.uid
11                     {
12                         if uid == "WfJdq91POMTlvvKkATv15NfsGrf2"{
13                             self.performSegue(withIdentifier: "adminSegue", sender: nil)
14                         }
15                         else
16                         {if user!.isEmailVerified == false{
17                             UIAlertController.showAlert(self, title: "Verification", message: "Plz Check your ")
18                             user!.sendEmailVerification(completion: { (_error) in
19                                 if _error != nil{
20                                     UIAlertController.showAlert(self, title: "Verification error", message: _error!.localizedDescription)
21                                 })
22                             }
23                             else{
24                                 self.performSegue(withIdentifier: "userSegue", sender: nil)
25                             }
26                         }
27                     }
28                 }
29                 else{
30                     UIAlertController.showAlert(self, title: "error", message: error!.localizedDescription)
31                 }
32             }
33         }
34     }
35 }
36 }
```

### Appendix b-1 Login and authentication code

```
func getData() {

    ref = Database.database().reference().child("incNum")
    ref?.observe(.value, with: { (snapshot) in
        self.items.removeAll()
        var gData :[dataItems] = []
        for snap in snapshot.children
        {
            let lisitem = dataItems(snapshot: snap as! DataSnapshot)
            gData.append(lisitem)

        }
        self.items = gData
        self.tableView.reloadData()

    })

}
```

### Appendix b-2 get data from the firebase

```

func adduser(){
    if emailField.text == "" && passwordField.text == ""{
        UIAlertController.showAlert(self, title: "error", message: "all Fields Required ")
    }
    else{
        if emailField.text != nil && passwordField.text != nil {
            Auth.auth().createUser(withEmail: emailField.text!, password: passwordField.text!) { (user,
            error) in
                if user != nil {
                    UIAlertController.showAlert(self, title: "welcome", message: "You Have Successfull signed up ")
                    user!.sendEmailVerification(completion: { (_error) in
                        if _error != nil{
                            UIAlertController.showAlert(self, title: "Verification error", message: _error!.localizedDescription)
                        }
                    })
                }
            }
        }
        else{
            UIAlertController.showAlert(self, title: "Error", message: error!.localizedDescription)
        }
    }
}
}
}
}

```

## Appendix b-3 add user function

```

@IBAction func `add`(_ sender: Any) {
    let alert = UIAlertController(title: "Add Incubator", message: "Enter The New Inc Num",
    preferredStyle: .alert)
    alert.addTextField { (textField) in
        textField.placeholder = "Enter The New incubator Number"
        textField.keyboardType = .phonePad
    }
    alert.addAction(UIAlertAction(title: "OK", style: .default, handler: { [weak alert] (_) in
        let textField = alert?.textFields?.first
        if textField!.text == ""
        {
            return
        }
        else{
            self.ref = Database.database().reference().child("incNum")
            let newIncRef = self.ref!.child("incNum \(textField!.text ?? "nil)").child("incNum") //New
            inc reference
            newIncRef.setValue("\(textField!.text ?? "nil")") //New Inc IncNum Value
        }
    })))
    self.present(alert, animated: true, completion: nil)
}
}
}
}

```

## Appendix b-4 add incubator function

### 3. Appendix B—android coding:

```
private void startPosting() {  
  
    final String incNum_val = incNum.getText().toString().trim();  
  
    final DatabaseReference newPostRef = databaseReference.child( incNum_val );  
  
    databaseReference.addValueEventListener( new ValueEventListener() {  
        @Override  
        public void onDataChange(DataSnapshot dataSnapshot) {  
  
            newPostRef.child( "IncNum" ).setValue( incNum_val );  
            newPostRef.child( "IdNum" ).setValue( " " );  
            Intent singleBlogIntent = new Intent( packageContext: AddIncubator.this, TabBar.class );  
            startActivity( singleBlogIntent );  
  
            finish();  
        }  
  
        @Override  
        public void onCancelled(DatabaseError databaseError) {  
        }  
    } );  
}
```

#### Appendix c-1 login and authentication code

```
private void startPosting() {  
  
    final String incNum_val = incNum.getText().toString().trim();  
  
    final DatabaseReference newPostRef = databaseReference.child( incNum_val );  
  
    databaseReference.addValueEventListener( new ValueEventListener() {  
        @Override  
        public void onDataChange(DataSnapshot dataSnapshot) {  
  
            newPostRef.child( "IncNum" ).setValue( incNum_val );  
            newPostRef.child( "IdNum" ).setValue( " " );  
            Intent singleBlogIntent = new Intent( packageContext: AddIncubator.this, TabBar.class );  
            startActivity( singleBlogIntent );  
  
            finish();  
        }  
  
        @Override  
        public void onCancelled(DatabaseError databaseError) {  
        }  
    } );  
}
```

#### Appendix c-2 add incubator

```

private void startPosting() {
    progress.setMessage("Uploading Data ...");

    final String UserName_val = UserName.getText().toString().trim();
    final String UserPhone_val = UserPhone.getText().toString().trim();

    progress.show();
    final DatabaseReference newPostRef = databaseReference.child(UserPhone_val);

    databaseReference.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot dataSnapshot) {

            newPostRef.child("UserName").setValue(UserName_val);
            newPostRef.child("UserPhone").setValue(UserPhone_val);

            Intent singleBlogIntent = new Intent(getActivity().getApplication(), TabBar.class);
            startActivity(singleBlogIntent);
        }

        @Override
        public void onCancelled(DatabaseError databaseError) {
        }
    });
}

```

## Appendix c-3 add user

```

void checkInc() {

    final String incNum_val = incNum.getText().toString().trim();

    final Query firebaseSearchQuery = databaseReference.orderByChild("IncNum").startAt(incNum_val).endAt(incNum_val + "\uf8ff");

    firebaseSearchQuery.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot dataSnapshot) {
            if (dataSnapshot.exists()) {
                databaseReference.child(incNum_val).removeValue();
                finish();
            }
            else {
                incNum.setError("This Inc Not Found");
                return;
            }
        }

        @Override
        public void onCancelled(DatabaseError databaseError) {
        }
    });
}

```

## Appendix c-3 remove incubator

## References

- [1] "Android course," UdaCity, 1 2 2018. [Online]. Available: <https://eg.udacity.com>. [Accessed 9 1 2018].
- [2] "blynk doc," blynk , 9 5 2010. [Online]. Available: <http://docs.blynk.cc/>. [Accessed 9 4 2018].
- [3] Eric C. Eichenwald, Anne R.Hansen,Ann R.Strak, "Eric C Eichenwald, Ann R Star, Anne R Hansen, Camilia R Martin-Cloherty and Stark's Manual of Neonatal Care-Wolters Kluwer (2017) table 15.1," in *Neonatal Care*, Austin, 2007, pp. 188-189-190.
- [4] "firebase doc," 9 8 2010. [Online]. Available: <https://firebase.google.com/docs/>. [Accessed 24 6 2018].
- [5] "ios11 and swift4," IOS, 1 6 2018. [Online]. Available: <https://www.udemy.com/ios-11-swift-4-from-beginner-to-paid-professional/>. [Accessed 10 6 2018].
- [6] Arduino References, Simply Arduino, Cairo: Abdalla ali, 2017.

## ملخص المشروع باللغة العربية

- يهدف هذا المشروع الي متابعة و مراقبة الاطفال في الحاضنة وتسجيل البيانات الخاصة بهم في قاعدة البيانات ومن ثم يتم عرضها فى تطبيقات الهاتف المحمول.

و تحتوي هذه الحاضنة علي جهاز استشعار الرطوبة وجهاز استشعار نبضات القلب و جهاز استشعار درجة الحرارة وهذه هي الأجهزة المطلوبة لمتابعة حالة الطفل, حيث أن جهاز استشعار نبضات القلب يتم وضعه علي اصبع الطفل داخل الحاضنة و جهاز استشعار الرطوبة يكون داخل الحاضنة لقياس درجه حرارة و رطوبة الحاضنة, ويتم توصيل جهاز استشعار الحرارة و الرطوبة بشريحة المتحكم فى الاتصال اللاسلكي وجهاز استشعار القلب بمفرده بالشريحة الالكترونية ( الاردوينو اونو ) وذلك منعا لحدوث اى مشكله عند توصيلة مع جهاز استشعار الحرارة و الرطوبة.

- يتم رفع هذه القياسات التى تعطيها هذه الحساسات علي الفايربيز لتسجيلها في قاعدة بيانات اولا ثم يتم عرض هذه القياسات فى تطبيقات الهاتف.

واخيرا تم عمل تطبيق ( بنظام الاندرويد والايفون) وهذا التطبيق سيكون خاص بالدكتور يستطيع من خلاله متابعه حالته الطفل لحظة بلحظة وذلك بدون ان يكون متواجد داخل غرفة الحاضنة.