EMBEDDED SOLUTION FOR BEDRIDDEN PATIENTS

A CAPSTONE PROJECT REPORT

Submitted in partial fulfillment of the
requirement for the award of the
Degree of

BACHELOR OF TECHNOLOGY
IN
ELECTRONICS AND COMMUNICATION ENGINEERING

by

GOUTAM JEERALA (19BEC7047)
ADDAGALLA DHANYA SREE (19BEC7100)

Under the Guidance of

Prof. M. Kalyan Chakravarthi

SCHOOL OF ELECTRONICS ENGINEERING
VIT-AP UNIVERSITY
AMARAVATI- 522237

DECEMBER 2022

CERTIFICATE

This is to certify that the Capstone Project work titled "**EMBEDDED SOLUTION FOR BEDRIDDEN PATIENTS**" that is being submitted by **GOUTAM JEERALA (19BEC7047) and ADDAGALLA DHANYA SREE (19BEC7100)** is in partial fulfillment of the requirements for the award of Bachelor of Technology, is a record of bonafide work done under my guidance. The contents of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University for the award of any degree or diploma and the same is certified.

Prof. M. Kalyan Chakravarthi
Guide

The thesis is satisfactory/unsatisfactory

Internal Examiner **External Examiner**

Approved by

PROGRAM CHAIR
B. Tech. ECE Engineering

DEAN
School Of Electronics

ACKNOWLEDGEMENTS

We would like to express our special thanks of gratitude to our teacher Prof.M. Kalyan Chakravarthi who gave us the golden opportunity to do this wonderful project "EMBEDDED SOLUTION FOR BEDRIDDEN PATIENTS". We would like to thank our subject teacher again for guiding us through the reviews whom without this project would not have been completed and our knowledge of engineering application would not have improved. Lastly, we would like to thank our SENSE school respective DEAN for giving us an opportunity to carry out our studies in the University.

GOUTAM JEERALA

DHANASREE ADDAGALLA

ABSTRACT

In the 21st century, every day new inventions are being developed, launched, and marketed to human beings. Research and developments in technology and healthcare cause a significant increase in the quality of life and average life expectancy. As the statistical research and forecasts tell us, it is not possible to ignore the aging population, resulting in the need for laborious caregiving. Caregiving is highly costly also; there is a big injury rate and days lost in caregiving facilities due to the need for a highly physical workforce.

Measurement and control of pressure play an important role in different fields of Science and Technology. Also, it becomes essential to monitor the real-time weather condition of one place to another place. In this paper, we present the Cloud-Based monitoring and measurement of pressure using Arduino UNO. The Uno with Cable is a microcontroller board base on the ATmega328. It has 14 digital input/output pins (of which 6 can be used as PWM outputs); 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. A force-sensing resistor is used for measuring pressure.

Measured parameters are sent to the Cloud to ThingSpeak through NodeMCU. Pressure measurements made in real time are shown graphically. The software is developed in the Arduino integrated development environment (IDE).

TABLE OF CONTENTS

S.No.	Chapter	Title	Page Number
1.		Acknowledgment	3
2.		Abstract	4
3.		List of Figures and Table	6
4.	1 1.1 1.2 1.3	Introduction Objectives Background and Literature Survey Organization of the Report	7 7 8 8
5.	2 2.1 2.2 2.3 2.4 2.4.1 2.4.2	EMBEDDED SOLUTION FOR BEDRIDDEN PATIENTS Proposed System Working Methodology Standards System Details Software Hardware	9 9 11 11 12 12 13
6.	3 3.1	Cost Analysis List of components and their cost	19 19
7.	4	Results and Discussion	20
8.	5	Conclusion & Future Works	22
9.	6	Appendix	23
10.	7	References	28

List of Tables

Table	Title	Page No.
1	Cost Analysis	19

List of Figures

Figure No.	Title	Page No.
1	System Block diagram	10
2	Overview of Embedded System	10
3	Arduino IDE	12
4	ThingSpeak Analysis	12
5	Detailed graph information of a Force sensor	13
6	Arduino UNO	13
7	ATmega 328 Microcontroller	15
8	Node MCU	16
9	Node MCU (Arduino IDE)	17
10	Force Sensor	18
11	Hardware setup	20
12	Arduino IDE output and readings	21
13	Arduino IDE output and readings	21
14	Future Analysis	22

CHAPTER 1

INTRODUCTION

Being bedridden and bed rest are two concepts that appear to be similar, however, they are quite different. Bed rest refers to a limited period of rest as prescribed by a doctor or as deemed necessary because of an acute illness whereas being bedridden has a more negative connotation and is described as a final state that gradually leads to (social) death, according to Zegelin2. According to "Training Material – PT Protocol for Bedridden Patients" a bedridden patient, for various reasons, has to stay in bed for a long period.

Factors such as genetic disorders, work stress, and unhealthy diet can increase the probability of illness. In addition, as our population ages, the demand for health care in the elderly population increases. Subsequently, with increasing numbers of patients, physicians and hospitals come increasing demands. Additionally, increasing healthcare costs make treatment and health screening out of reach for many patients. Remote patient monitoring has been proposed to meet this increased demand. It has also become more popular as a means of simplifying the healthcare process. Remote patient monitoring can not only meet the increased demand but also increase overall efficiency by, for example, reducing travel and waiting time and decreasing the spread of infectious diseases in doctor's waiting rooms.

Any patient monitoring system consists of signal acquisition and a processing section to record the vital data from the patient's body. The patient monitoring system updates the physician about the patient's health status. In this context embedded systems play a vital role in IoT; these embedded systems use an admixture of digital and analog subsystems in gathering information from sensors.

1. Objectives

The following are the objectives of this project:

- To design an efficient system that can continuously monitor pressure changes.
- This Embedded system constantly checks the state of the patient after frequent intervals. It warns through the android application and suggests that action is now required if there is no value received or when it reached the maximum limit.

- When the patient is having an appointment with the doctor, previous details of his condition or a summary of his condition over the past few weeks/days can be shared which will be stored in the cloud.
- Sensors interfaced with Arduino constantly monitor the pressure changes of the patient.

0. Background and Literature Survey

Embedded solutions for bedridden patients are systems that are designed to improve the quality of life and care of those who are unable to leave the bed. These systems typically incorporate the use of technology to enable communication and interaction between the patient and those providing care. One type of embedded solution for bedridden patients is the telehealth system. This system enables remote monitoring of the patient's health and provides a two-way communication link between the patient and their care provider.

Telehealth systems can include features such as vital sign monitoring, remote diagnostics, appointment scheduling, health records management, and other features. Another type of embedded solution for bedridden patients is the robotic system. These systems are designed to assist with a variety of tasks including mobility, communication, and social interaction. Robotic systems are equipped with sensors and cameras to enable them to interact with their environment and the patient.

Additionally, they may be programmed to perform specific tasks such as turning on lights, providing medication reminders, and providing companionship. Finally, embedded solutions for bedridden patients may also include assistive technology such as voice-activated devices, sensor-based systems, and computer-based systems. These systems are designed to enable the patient to interact with their environment and to receive

Bedridden patients who are unable to move from their bed due to either an underlying medical condition or an injury. Bedridden patients have a greater risk of developing a range of
medical complications, such as pressure sores, deep vein thrombosis, and muscle atrophy. In order to provide better care for these patients, it is important that they are able to remain as comfortable and active as possible.

This can be accomplished through the use of embedded solutions that allow for remote monitoring and interaction with the patient. This literature review and analysis will focus on the use of embedded solutions for bedridden patients and their potential benefits. Literature Review Recent studies have shown that embedded solutions have the potential to improve the care of bedridden patients.

One study conducted by Wang et al. (2020) evaluated the use of a wireless telemonitoring system to monitor bedridden patients in a home care setting. The system was used to collect patient data (such as heart rate, respiration rate, and blood pressure) and transmit it wirelessly to a remote monitoring station. The results of the study showed that the system was able to provide accurate and timely data and that it was able to improve the quality of care for the patient.

1.3 Organization of the Report

The remaining chapters of the project report are described as follows:

- Chapter 2 contains the proposed system, methodology, hardware and software details.
- Chapter 3 gives the cost involved in the implementation of the project.

- Chapter 4 discusses the results obtained after the project was implemented.
- Chapter 5 concludes the report.
- Chapter 6 consists of codes.
- Chapter 7 gives references.

CHAPTER 2

EMBEDDED SOLUTION FOR BEDRIDDEN PATIENTS

This Chapter describes the proposed system, working methodology, software and hardware details.

2.1 Proposed System

In the proposed method we are using force to detect the presence of a patient. Here we are using Arduino as the main controller for it we are interfacing the force sensors. If the patient doesn't move or if there is no change in force sensor value, then in the serial monitor we get an alert as "Alert! Patient is not moving". If there is no force detected then it will display "No patient on bed". The sensor's data will be sent to the thingspeak server through NodeMCU.

The following block diagram (figure 2) shows the system architecture of this project.

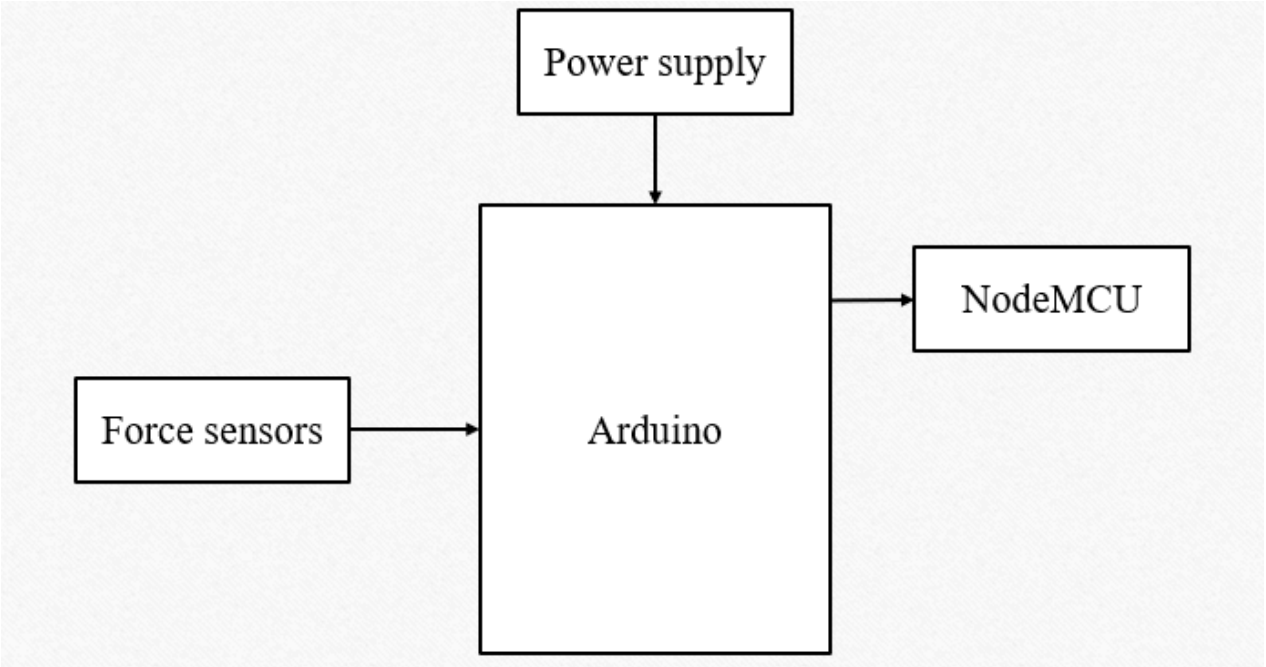


Figure 1 System Block Diagram

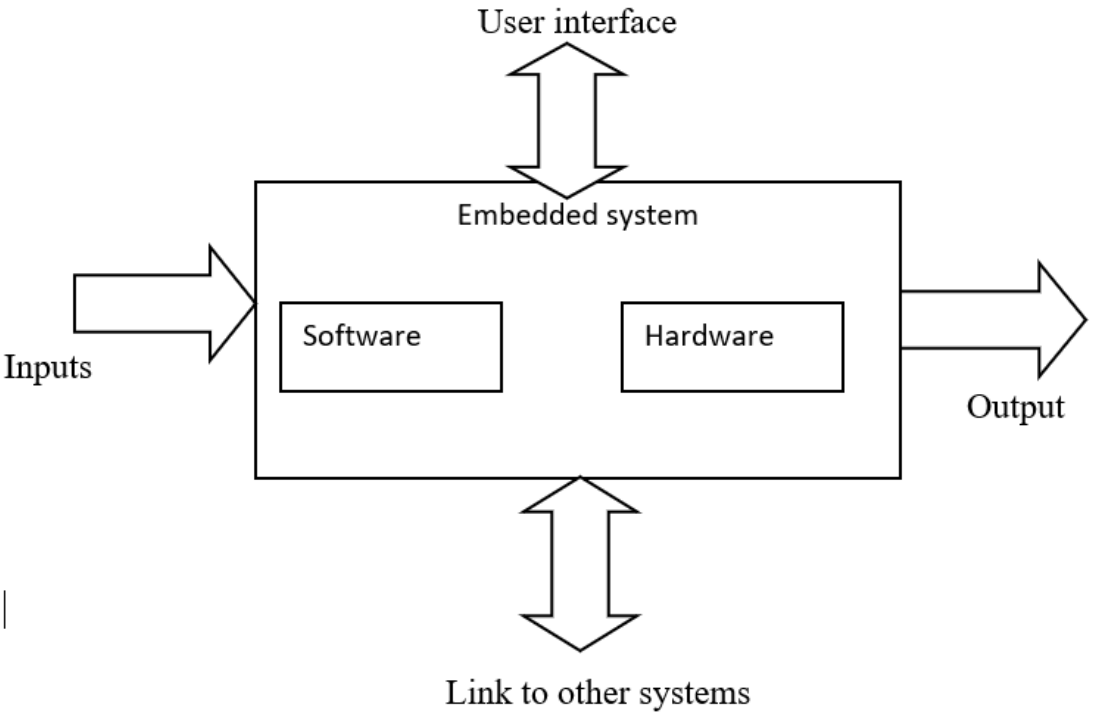


Figure 2 Overview of Embedded system

2.2 Working Methodology

The system has two sections, hardware, and software. Hardware consists of Ardunio, Node MCU, and force sensors. Arduino is connected to force sensors which constantly monitor the pressure of the patient. These force sensors are attached to the patient's body parts (1 for the head, 2 for the left and right shoulders, 1 for the hip, and 2 for the left and right legs). Now the pressure is calculated and analysed for every six iterations and alerts the condition of the patient.

2.3 Standards

Various standards used in this project are

Secure and reliable data storage:

Thinkspeak is a cloud-based data storage system that provides secure and reliable storage. It uses an advanced encryption system to protect the data stored in its cloud, and also offers a range of additional features to ensure data security and reliability. These features include two-factor authentication, secure access control, and data integrity checks.

Easy-to-use interface:

The device we used has an intuitive and user-friendly interface to facilitate the monitoring of the bedridden patient.

Robust connectivity:

The ESP8266 is a highly integrated Wi-Fi-enabled microcontroller with robust connectivity. It has a built-in Wi-Fi module, which can be used to connect to any compatible Wi-Fi network or create an Access Point. The NodeMCU firmware also comes with a built-in TCP/IP stack, which allows it to receive and send data over the internet. It also has a USB port, which can be used to connect it to a computer or other peripheral devices.

Automated alerts:

This device is able to provide automated alerts to relevant personnel in the event that the patient is still in bed without movement, or No patient is on the bed for easy analysis.

2.4 System Details

This section describes the software and hardware details of the system:

2.4.1 Software Details

Arduino IDE

The Arduino IDE (Integrated Development Environment) is used to write the computer code and upload this code to the physical board. The Arduino IDE is very simple and this simplicity is probably one of the main reasons Arduino became so popular. We can certainly state that being compatible with the Arduino IDE is now one of the main requirements for a new microcontroller board.

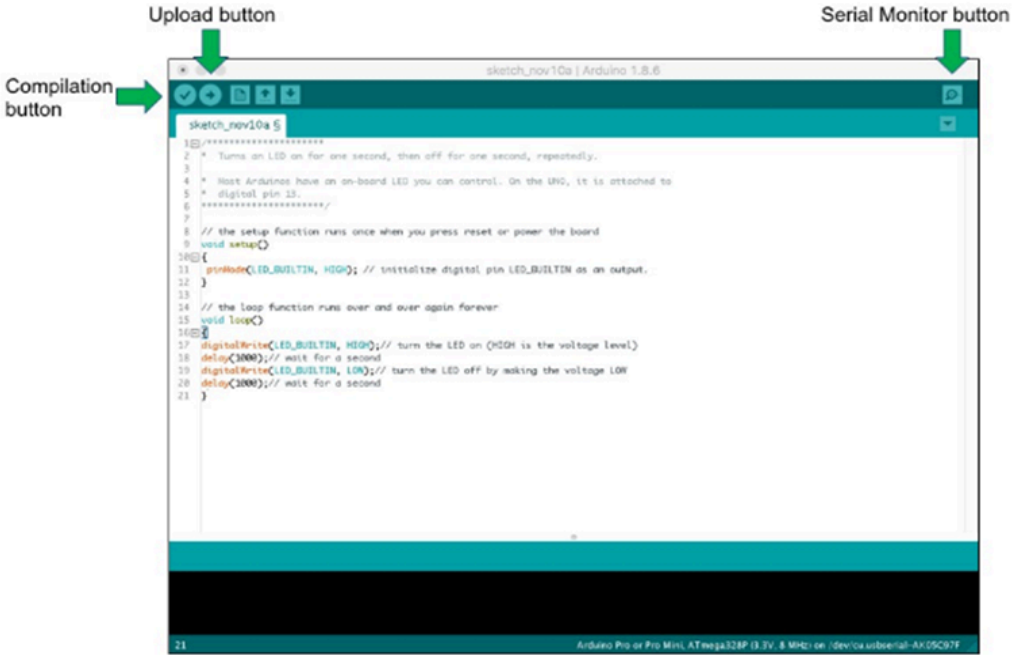


Figure 3 Arduino IDE

ThingSpeak

ThingSpeak is open-source software written in Ruby which allows users to communicate with internet-enabled devices. It facilitates data access, retrieval and logging of data by providing an API to both the devices and social network websites

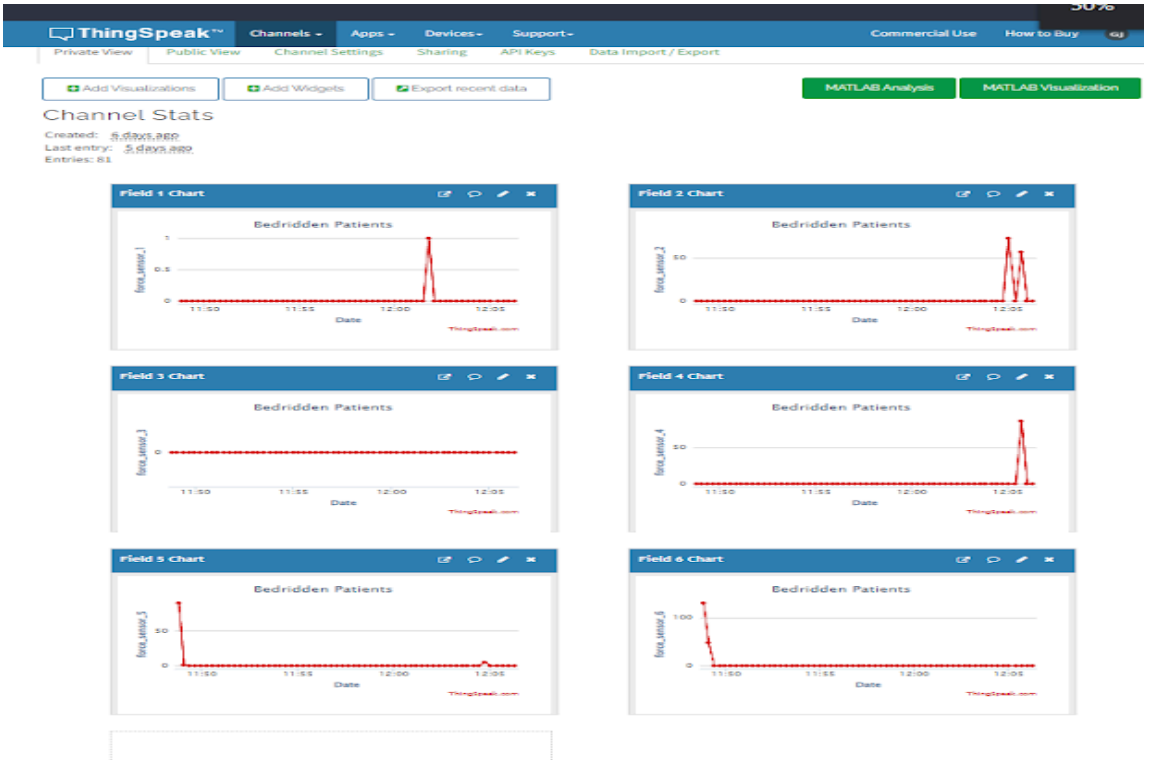


Figure 4 ThingSpeak Analysis

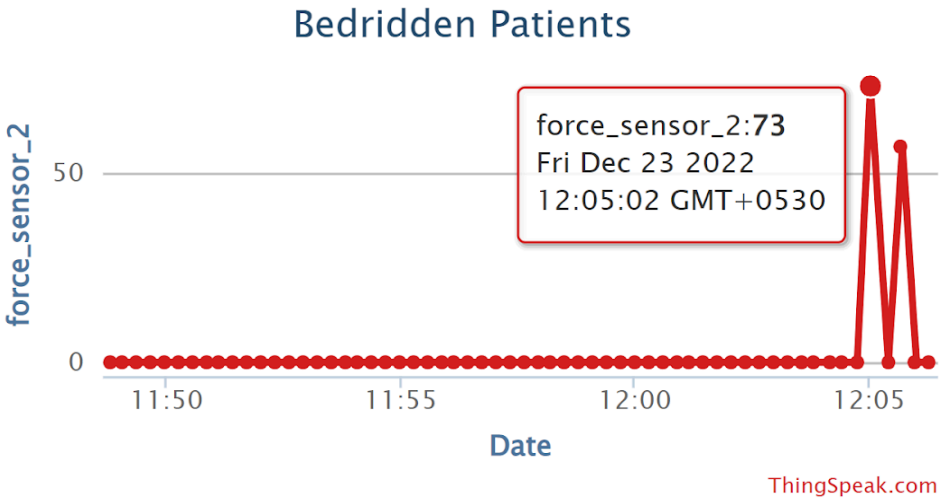
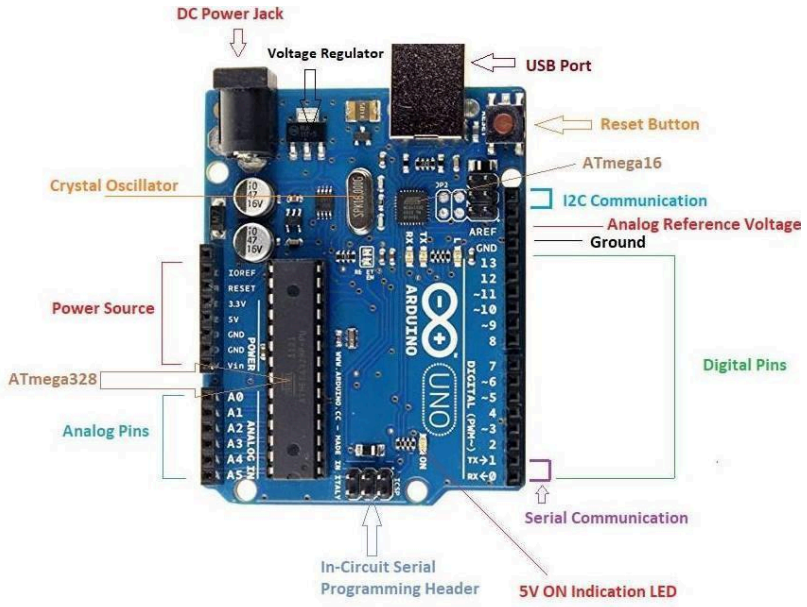


Figure 5 Detailed graph information of a force sensor

2.4.2 Hardware Details

As shown in Figure 1 we have various hardware components being used in this system. The details of each component are as follows

.Arduino UNO



Arduino UNO

Figure 6 Arduino UNO

Arduino Uno is a microcontroller board developed by Arduino. cc which is an open-source electronics platform mainly based on the AVR microcontroller Atmega328.

The first Arduino project was started at Interaction Design Institute Ivrea in 2003 by David Cuartillas and Massimo Bonzi with the intention of providing a cheap and flexible way to students and professionals for controlling a number of devices in the real world.

The current version of Arduino Uno comes with a USB interface, 6 analog input pins, 14 I/O digital ports that connect with external electronic circuits. Out of 14 I/O ports, 6 pins can be used for PWM output. It allows the designers to control and sense the external electronic devices in the real world

This board comes with all the features required to run the controller and can be directly connected to the computer through a USB cable that is used to transfer the code to the controller using IDE (Integrated Development Environment) software, mainly developed to program

Arduino. IDE is equally compatible with Windows, MAC or Linux Systems; however, Windows is preferable to use. Programming languages like C and C++ are used in IDE. Apart from USB, a battery or AC to DC adapter can also power the board.

Arduino Uno boards are quite like other boards in the Arduino family in terms of use and functionality, however, Uno boards don't come with FTDI USB to Serial driver chip. There are many versions of Uno boards available, however, Arduino Nano V3 and Arduino Uno are the most official versions that come with an Atmega328 8-bit AVR Atmel microcontroller where RAM memory is 32KB.

Features of Arduino

Arduino Uno comes with a USB interface i.e. A USB port is added on the board to develop serial communication with the computer. Atmega328 microcontroller is placed on the board that comes with a number of features like timers, counters, interrupts, PWM, CPU, and I/O pins and is based on a 16MHz clock that helps in producing more frequency and number oF instructions per cycle.

It is an open-source platform where anyone can modify and optimize the board based on the number of instructions and tasks they want to achieve.

There are 14 I/O digital and 6 analog pins incorporated in the board that allows the external connection with any circuit with the board. These pins provide flexibility and ease of use to the external devices that can be connected through these pins.

There is no hard and fast interface required to connect the devices to the board. Simply plug the external device into the pins of the board that are laid out on the board in the form of the header.

The 6 analog pins are marked as A0 to A5 and come with a resolution of 10 bits. These pins measure from 0 to 5V, however, they can be configured to the high range using the analogReference () function and AREF pin.

Only 5 V is required to turn the board on, which can be achieved directly using a USB port or external adapter, however, it can support external power sources up to 12 V which can be regulated and limited to 5 V or 3.3 V based on the requirement of the project. 13KB of flash memory is used to store the number of instructions in the form of code.

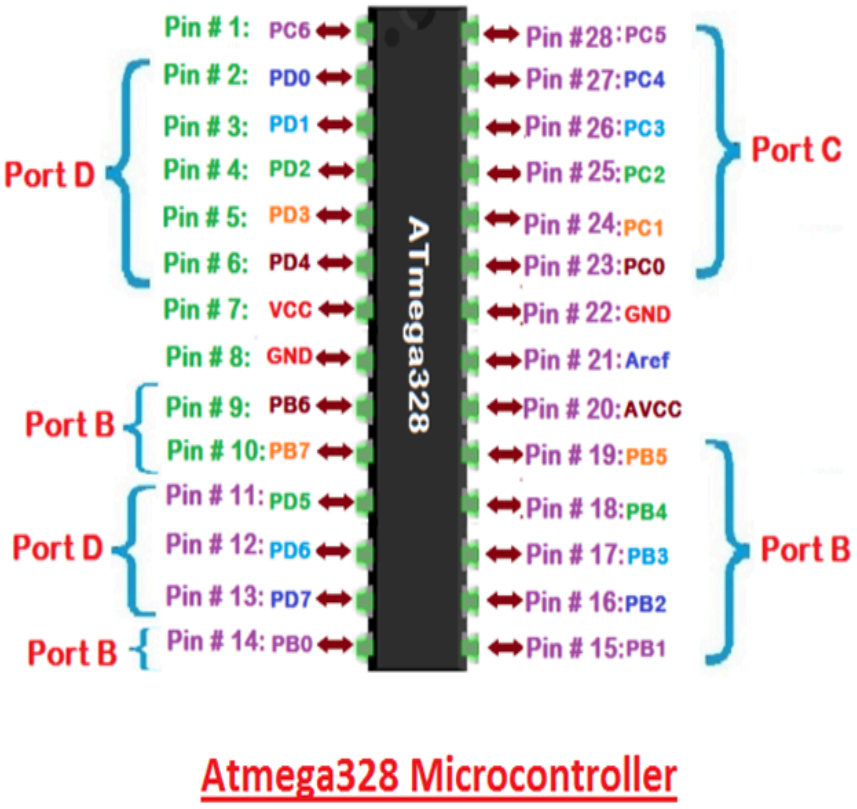


Figure 7 Atmega328 Microcontroller

ii) Node MCU

NodeMCU is an open-source firmware and development kit that plays a vital role in designing your own IoT product using a few Lua script lines.
Multiple GPIO pins on the board allow you to connect the board with other peripherals and are capable of generating PWM, I2C, SPI, and UART serial communications.

- The interface of the module is mainly divided into two parts including both Firmware and Hardware where the former runs on the ESP8266 Wi-Fi SoC and later is based on the ESP-12 module.

The firmware is based on Lua – A scripting language that is easy to learn, giving a simple programming environment layered with a fast-scripting language that connects you with a well-known developer community.

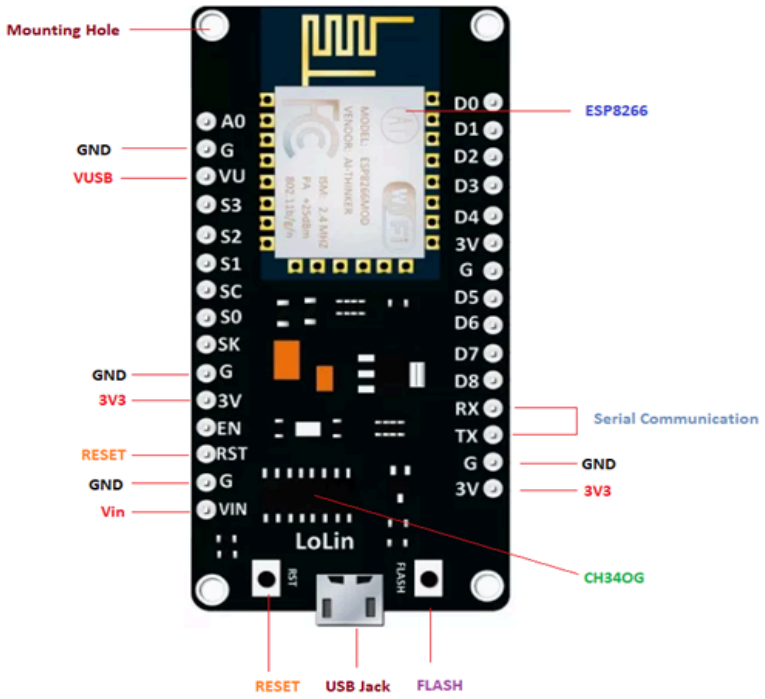


Figure 8 Node MCU

open-source firmware gives you the flexibility to edit, modify and rebuilt the existing module and keep changing the entire interface until you succeed in optimizing the module as per your requirements.

- USB to UART converter is added to the module that helps in converting USB data to UART data which mainly understands the language of serial communication.

Instead of the regular USB port, the MicroUSB port is included in the module that connects it to the computer for dual purposes: programming and powering up the board.

- The board incorporates a status LED that blinks and turns off immediately, giving you the current status of the module if it is running properly when connected to the computer.

The ability of the module to establish a flawless Wi-Fi connection between two channels makes it an ideal choice for incorporating it with other embedded devices like Raspberry Pi.

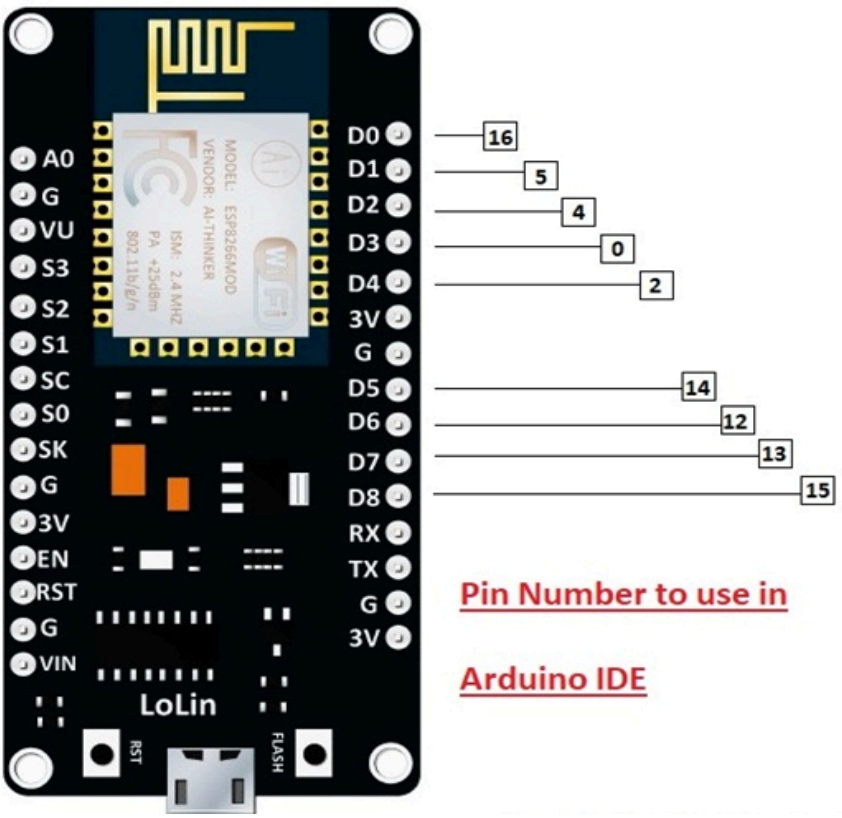


Figure 9 Node MCU(Arduino IDE)

iii) Force Sensor

A Force Sensing Resistor, also known as a Force Sensor, or simply an FSR, is a simple and inexpensive sensor designed to measure physical pressure, squeeze, and weight. It can be found in a variety of portable electronics, including electronic drums, handheld gaming devices, and mobile phones.

This sensor is excellent at measuring pressure, but not so accurate at estimating how much weight is on it. So, if you just want to know "whether the sensor has been squeezed or pressed and how hard," it could be a good choice for your next force-sensing project. The patent for the technology used in FSRs is owned by Interlink Electronics, which has been in business since 1985. The most common types of FSR that you will encounter are Interlink FSR-402 and FSR-406.



Figure 10 Force Sensor

An FSR is simply a variable resistor whose resistance varies in response to pressure applied to the sensing area. It is made up of several thin, flexible layers. When squeezed, more of the carbon elements that normally offer resistance are brought into contact with the conductive traces, thereby lowering the resistance. There is a wide selection of FSRs available, each with its own unique size, shape, and sensing range.

The majority of FSRs have circular or rectangular sensing areas. Rectangular FSRs are ideal for wide-area sensing, whereas small circular sensors can provide higher accuracy.

CHAPTER 3

COST ANALYSIS

3.1 List of components and their cost

The costs of the various components used in this project are given below in Table 3.1.

Table 3.1 List of components and their costs

COMPONENT	COST
Arduino Uno	₹ 1000
Node MCU - ESP8266	₹ 1270
Force Sensor - 6 Units	₹ 5400
Miscellaneous	₹ 1500
TOTAL	₹ 9170

CHAPTER 4

RESULTS AND DISCUSSIONS

a. Sensor Readings

The microcontroller unit was able to transmit the values collected from the sensors by the system depicted in figure 11 to the Arduino UNO, which is shown in Figures 12 and 13.

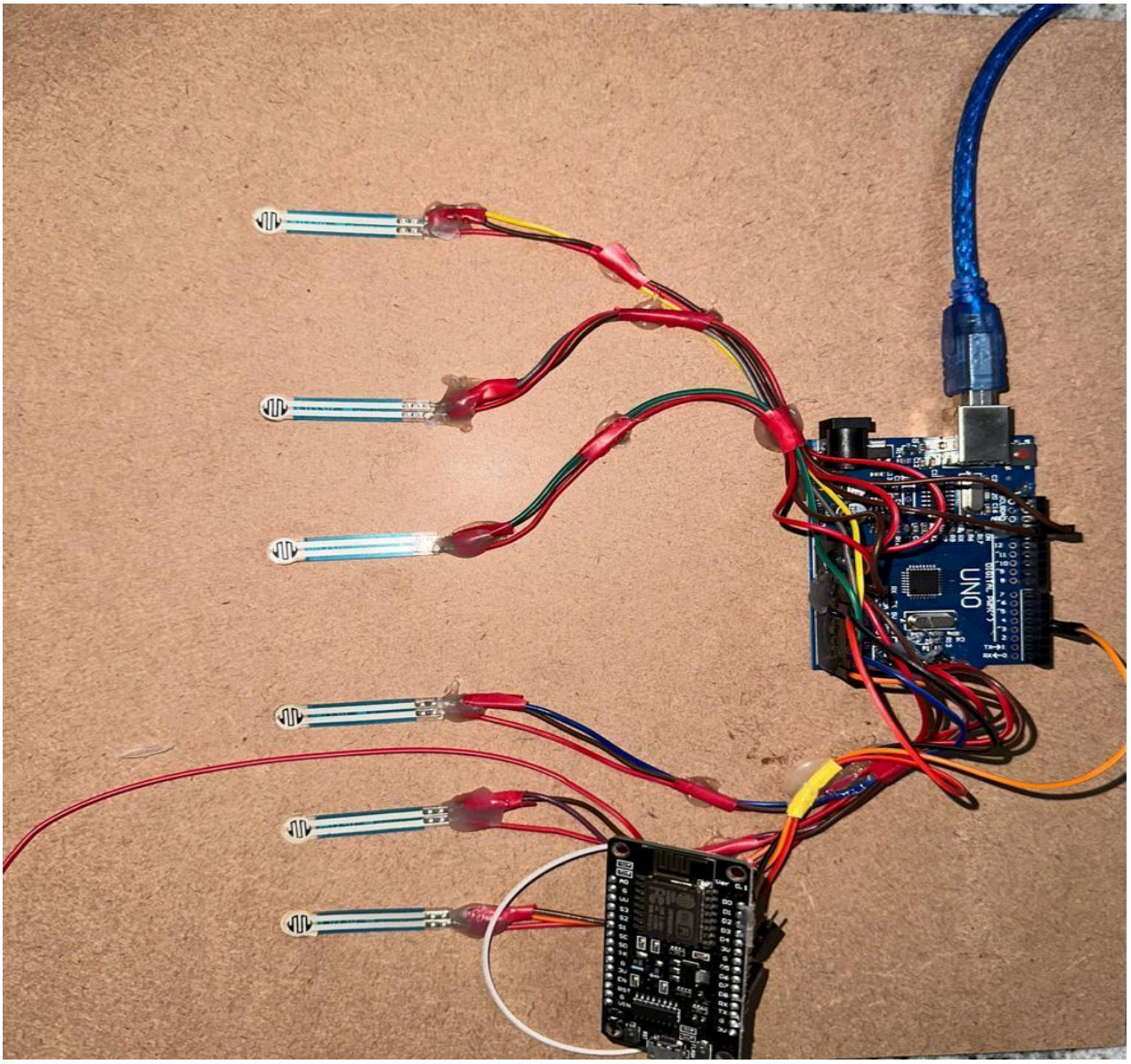


Figure 11 Hardware Setup

b. Integration Hardware and Software

Fig 12 and fig 13 depict the values of force and alerts about the status of the patient on the bed

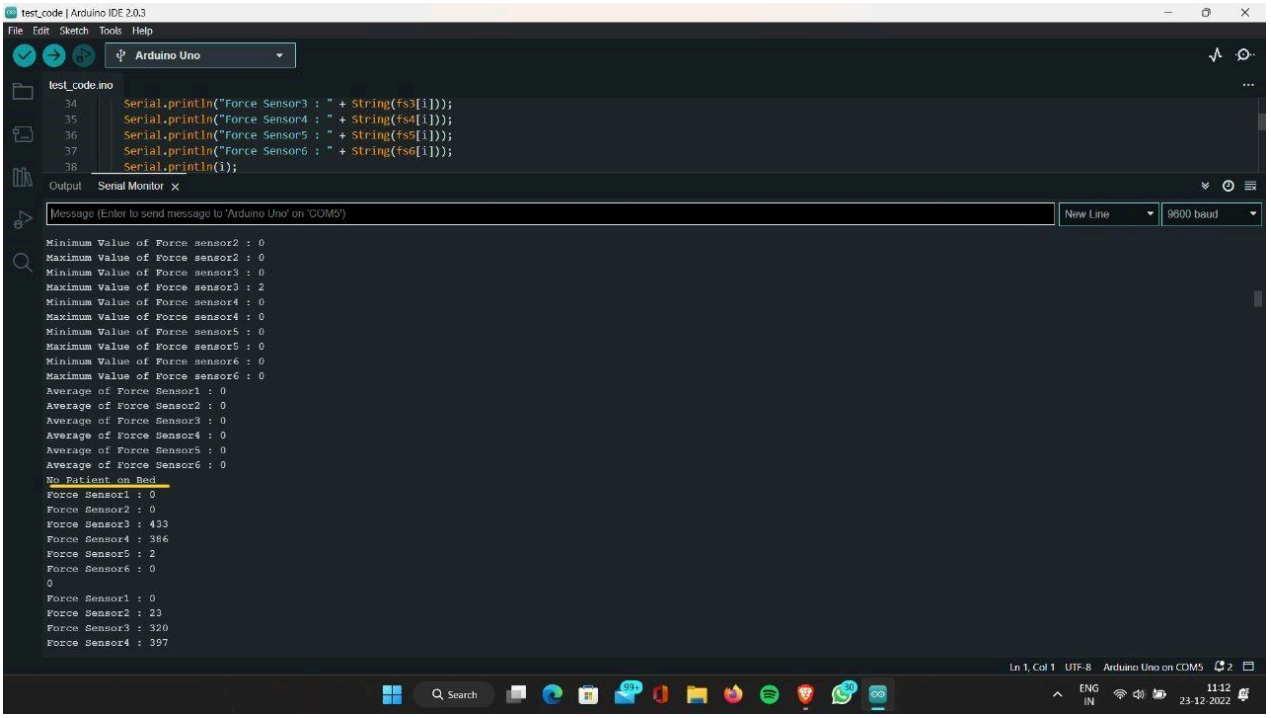


Figure 12 Arduino IDE output and readings

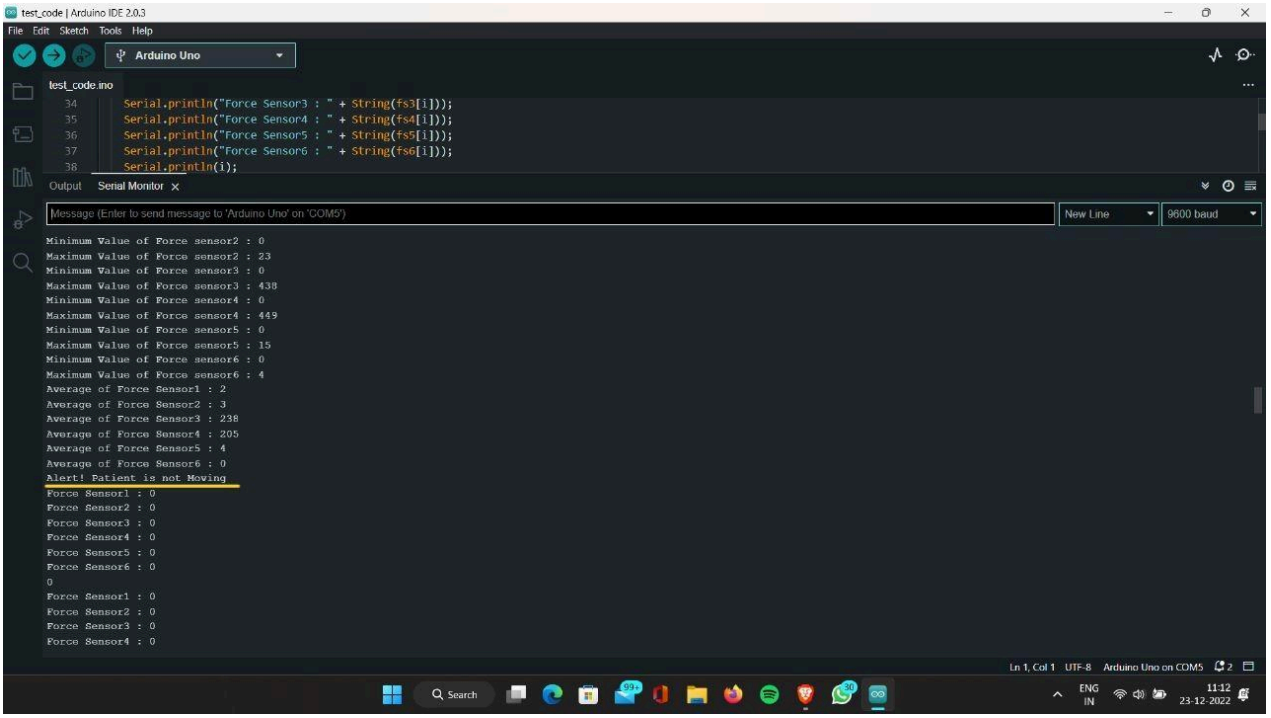


Figure 13 Arduino IDE readings and output

CHAPTER 5

CONCLUSION AND FUTURE WORK

The embedded solution for bedridden patients has the potential to revolutionize the way in which such patients are taken care of. Providing a comprehensive range of medical assistance. As this system can help to provide a better quality of life to those unable to move. Not only does it offer convenience, but it also helps to reduce the burden of care significantly on medical professionals, and caregivers and it provides cloud information to monitor the status of bedridden patients Ultimately, it is an innovative solution that can help to improve the lives of those who are bedridden and in need of assistance.

A lot can be done in this area. There is a large scope that could be ventured, and new designs or systems could be made to improve the conditions and efficiency of the solution that we created by using force sensors. We can figure out if in the near future any of the components might need attention such as motors to help the patients to move without the physical acquaintance of the doctors.

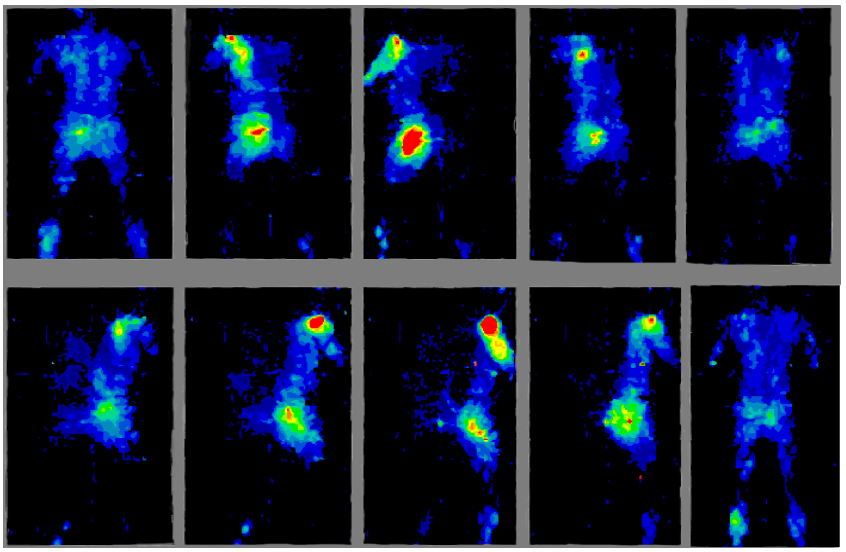


Figure 14 Future Analysis

CHAPTER 6

APPENDIX

Arduino Code

```
#include<SoftwareSerial.h>

SoftwareSerial nod(3,4);

int fs1[6], fs2[6], fs3[6], fs4[6], fs5[6], fs6[6]; //Variable declaration
int s1 = 0, s2 = 0, s3 = 0, s4 = 0, s5 = 0, s6 = 0;

void setup() {
  Serial.begin(9600);            //Baud rate setting
  nod.begin(9600);
}

void loop() {
  //force sensor reading
  for (int i = 0; i < 6; i++) {
    fs1[i] = analogRead(A0);
    fs2[i] = analogRead(A1);
    fs3[i] = analogRead(A2);
    fs4[i] = analogRead(A3);
    fs5[i] = analogRead(A4);
    fs6[i] = analogRead(A5);

    //Average Calculation
    s1 = s1 + fs1[i];
    s2 = s2 + fs2[i];
    s3 = s3 + fs3[i];
    s4 = s4 + fs4[i];
    s5 = s5 + fs5[i];
    s6 = s6 + fs6[i];

    //Monitor Printing of force sensor values
    Serial.println("Force Sensor1 : " + String(fs1[i]));
    Serial.println("Force Sensor2 : " + String(fs2[i]));
    Serial.println("Force Sensor3 : " + String(fs3[i]));
    Serial.println("Force Sensor4 : " + String(fs4[i]));
    Serial.println("Force Sensor5 : " + String(fs5[i]));
    Serial.println("Force Sensor6 : " + String(fs6[i]));
    Serial.println(i);
    delay(1000);
```

```
}

//Calculation of minimum and maximum values of the arrays
int Max1 = fs1[0];
int Min1 = fs1[0];
int Max2 = fs2[0];
int Min2 = fs2[0];
int Max3 = fs3[0];
int Min3 = fs3[0];
int Max4 = fs4[0];
int Min4 = fs4[0];
int Max5 = fs5[0];
int Min5 = fs5[0];
int Max6 = fs6[0];
int Min6 = fs6[0];
for (int i = 0; i < (sizeof(fs1) / sizeof(fs1[0])); i++)
{
  Max1 = max(fs1[i], Max1);
  Min1 = min(fs1[i], Min1);
}
for (int i = 0; i < (sizeof(fs2) / sizeof(fs2[0])); i++)
{
  Max2 = max(fs2[i], Max2);
  Min2 = min(fs2[i], Min2);
}
for (int i = 0; i < (sizeof(fs3) / sizeof(fs3[0])); i++)
{
  Max3 = max(fs3[i], Max3);
  Min3 = min(fs3[i], Min3);
}
for (int i = 0; i < (sizeof(fs4) / sizeof(fs4[0])); i++)
{
  Max4 = max(fs4[i], Max4);
  Min4 = min(fs4[i], Min4);
}
for (int i = 0; i < (sizeof(fs5) / sizeof(fs5[0])); i++)
{
  Max5 = max(fs5[i], Max5);
  Min5 = min(fs5[i], Min5);
}
for (int i = 0; i < (sizeof(fs6) / sizeof(fs6[0])); i++)
{
  Max6 = max(fs6[i], Max6);
  Min6 = min(fs6[i], Min6);
}
```

```
  //Monitor Printing of Minimum and Maximum Values
  Serial.println("Minimum Value of Force sensor1 : "+String(Min1));
  Serial.println("Maximum Value of Force sensor1 : "+String(Max1));
  Serial.println("Minimum Value of Force sensor2 : "+String(Min2));
  Serial.println("Maximum Value of Force sensor2 : "+String(Max2));
  Serial.println("Minimum Value of Force sensor3 : "+String(Min3));
  Serial.println("Maximum Value of Force sensor3 : "+String(Max3));
  Serial.println("Minimum Value of Force sensor4 : "+String(Min4));
  Serial.println("Maximum Value of Force sensor4 : "+String(Max4));
  Serial.println("Minimum Value of Force sensor5 : "+String(Min5));
  Serial.println("Maximum Value of Force sensor5 : "+String(Max5));
  Serial.println("Minimum Value of Force sensor6 : "+String(Min6));
  Serial.println("Maximum Value of Force sensor6 : "+String(Max6));

  delay(1000);

  //Monitor Printing of sensor average values
  s1 = s1 / 6; s2 = s2 / 6; s3 = s3 / 6; s4 = s4 / 6; s5 = s5 / 6; s6 = s6 / 6;
  Serial.println("Average of Force Sensor1 : " + String(s1));
  Serial.println("Average of Force Sensor2 : " + String(s2));
  Serial.println("Average of Force Sensor3 : " + String(s3));
  Serial.println("Average of Force Sensor4 : " + String(s4));
  Serial.println("Average of Force Sensor5 : " + String(s5));
  Serial.println("Average of Force Sensor6 : " + String(s6));

  //Condition Checking
  if((Min1 <= s1 <=Max1) && (Min2 <= s2 <=Max2) && (Min3 <= s3 <=Max3) && (Min4
<= s4 <=Max4) && (Min5 <= s5 <=Max5) && (Min6 <= s6 <=Max6) && (s1 > 10 || s2 >
10 || s3 > 10 || s4 > 10 || s5 > 10 || s6 > 10)){
     Serial.println("Alert! Patient is not Moving");
  }
  if(s1 < 10 && s2 < 10 && s3 < 10 && s4 < 10 && s5 < 10 && s6 < 10){
    Serial.println("No Patient on Bed");
  }
  nod.print("!"+String(s1)+"@"+String(s2)+"#"+String(s3)+"$"+String(s4)+"%"+String(s5)+
"^"+String(s6));

  //Resetting Values
  s1 = 0; s2 = 0; s3 = 0; s4 = 0; s5 = 0; s6 = 0;
  delay(1000);

}
```

Nod Thingspeak Code

```
#include <ESP8266WiFi.h>
#include "secrets.h"
#include "ThingSpeak.h"
//
String k, v;
char ssid[] = SECRET_SSID;   // your network SSID (name)
char pass[] = SECRET_PASS;   // your network password
int keyIndex = 0;            // your network key Index number (needed only for WEP)
WiFiClient  client;

unsigned long myChannelNumber = SECRET_CH_ID;
const char * myWriteAPIKey = SECRET_WRITE_APIKEY;

void setup() {
  Serial.begin(9600);
  Serial.println("Working");
  //!1@23#12$12%22^23
  if (WiFi.status() != WL_CONNECTED) {
    Serial.print("Attempting to connect to SSID: ");
    Serial.println(SECRET_SSID);
    while (WiFi.status() != WL_CONNECTED) {
      WiFi.begin(ssid, pass);  // Connect to WPA/WPA2 network. Change this line if using
open or WEP network
      Serial.print(".");
      delay(5000);
    }
    Serial.println("\nConnected.");
  }
  while (!Serial) {
    ; // wait for serial port to connect. Needed for Leonardo native USB port only
  }
  // WiFi.mode(WIFI_STA);
  ThingSpeak.begin(client);  // Initialize ThingSpeak
}

void loop() {
  if (Serial.available()) {
    k = Serial.readString();
    Serial.println(k);
    //      r = k.c_str();
    int m = k.length();
    Serial.println(m);
    if (k[0] == '!') {
```

```
    int p = k.indexOf('@');
    int q = k.indexOf('#');
    int r = k.indexOf('$');
    int s = k.indexOf('%');
    int t = k.indexOf('^');

    String a = k.substring(1, p);
    String b = k.substring(p + 1, q);
    String c = k.substring(q + 1, r);
    String d = k.substring(r + 1, s);
    String e = k.substring(s + 1, t);
    String f = k.substring(t + 1, m);

    Serial.println(a);
    Serial.println(b);
    Serial.println(c);
    Serial.println(d);
    Serial.println(e);
    Serial.println(f);
    ThingSpeak.setField(1, a);
    ThingSpeak.setField(2, b);
    ThingSpeak.setField(3, c);
    ThingSpeak.setField(4, d);
    ThingSpeak.setField(5, e);
    ThingSpeak.setField(6, f);
    int x = ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);

    if (x == 200) {
      Serial.println("Channel update successful.");
    }
    else {
      Serial.println("Problem updating channel. HTTP error code " + String(x));
    }
   }
  }
}
```

REFERENCES

[1] Arif. M. Sreevas. S. Nafseer. K. and Rahul. R. (2012), 'Automated online Blood bank database', India Conference (INDICON), Annual IEEE, Print ISBN: 978-1-4673-2270-6, pp. 012 - 017.

[2] Bing-Nan Li, Taipa Ming-Chui Dong, and Vai, M.1. (2006), From Codabar to ISBT 128: Implementing Barcode Technology in Blood Bank Automation System', 27th Annual International Conference of the Engineering in Medicine and Biology Society, IEEE-EMBS, pp. 542-545.

[3] Ibrahim. M and M. Youssef (2012), 'CeIlSense: An Accurate Energy-Efficient GSM Positioning System Vehicular Technology, IEEE Transactions on Volume: 61, Issue: 1, ISSN: 0018-9545, pp. 286 - 296.

[4]Dr.A.Sabanayagam, G.Anish Girija," DESIGN AND MODELING OF MOBILE HEALTH MONITORING SYSTEM", International Journal of Innovations in Scientific and Engineering Research (IJISER),vol4,no 2,pp.63- 65,2017.

[5] S. M. Mahalle, P. V. Ingole, "Design and Implementation of Wireless Body Area Sensor Network Based Health Monitoring System", International Journal of Engineering Research & Technology, Vol. 2 Issue 6, pp. 105- 113, June 2013

[6] Sonam Khedkar, Swapnil Thube, "Real Time Databases for Applications", International Research Journal of Engineering and Technology (IRJET) Real Time Databases for Applications, Volume: 04 Issue: 06 | June -2017

[7] R. Kumar; M. Pallikonda Rajasekaran, An IoT based patient monitoring system using raspberry Pi, 2016 International Conference on Computing Technologies and Intelligent Data Engineering (ICCTIDE'16).

[8] Sarfraz Fayaz Khan, Health care monitoring system in the Internet of Things (IoT) by using RFID, 2017 6th International Conference on Industrial Technology and Management (ICITM)