SOLDIER HEALTH MONITORING AND TRACKING SYSTEM USING IOT

A PROJECT PHASE - II REPORT

Submitted by

G. BALAJI	**142219205010**
BALASUBRAMANIAN KALYAN	**142219205012**
ELA BARATH	**142219205019**
V. GOKUL ARAVIND	**142219205022**

in partial fulfilment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

SRM VALLIAMMAI ENGINEERING COLLEGE

(AN AUTONOMOUS INSITUTION)

SRM NAGAR, KATTANKULATHUR, CHENGALPATTU

ANNA UNIVERSITY :: CHENNAI 600 025

APRIL 2023

ANNA UNIVERSITY : CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"SOLDIER HEALTH MONITORING AND TRACKING SYSTEM USING IOT"** is the bonafide work of **"G. BALAJI, BALASUBRAMANIAN KALYAN, ELA BARATH and V. GOKUL ARAVIND"** who carried out the project phase - II work under my supervision.

SIGNATURE

Dr. S. NARAYANAN

HEAD OF THE DEPARTMENT

Associate Professor
Department of Information Technology

SRM Valliammai Engineering College
(AN AUTONOMOUS INSITUTION)
SRM Nagar,
Kattankulathur,
Chengalpattu - 603 203.

SIGNATURE

Dr. S. RAVIKUMAR

SUPERVISOR

Associate Professor
Department of Artificial Intelligence & Data Science
SRM Valliammai Engineering College
(AN AUTONOMOUS INSITUTION)
SRM Nagar,
Kattankulathur,
Chengalpattu - 603 203.

Submitted for the viva voce held on ……...

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

First and foremost, we would like to extend our heartfelt respect and gratitude to our Director **Dr. B. Chidhambararajan** Principal **Dr. M. Murugan** who helped us in our endeavours.

We also extend our heartfelt respect to our beloved Head of the Department **Dr. S. Narayanan B.E., M.Tech., Ph.D., Associate Professor** for offering his sincere support throughout the course work progress.

We thank our Project Coordinator(s) **Dr. A. R.Revathi, B.E., M.Tech., M.B.A., Ph.D., Associate Professor** and **Mrs. U. Chindiyababy, M.E., (Ph.D)., Assistant Professor (O.G)** for their consistent guidance and encouragement throughout the progress of the project.

We thank our Project guide **Dr. S. Ravikumar, M.E., Ph.D., Associate Professor** for his valuable guidance, support and active interest for the successful implementation of the project.

We would also thank all **the Teaching and Non-Teaching staff** members of our department for their constant support and encouragement throughout the course of this project work.

Finally, the constant support from our lovable Parents and Friends is untold and immeasurable.

ABSTRACT

This work deals with keeping track of the soldier's parameters, such as temperature, breathing, and heart rate. In designing the soldier monitoring system, we use the Arduino Uno and Node MCU, which can be defined as systems used to monitor physiological information such as heartbeat, body temperature, gas-related parameters, and so on. We are monitoring the heartbeat rate and temperature of body parameters through the Arduino Uno. Connections are made on the Arduino's digital and analogue pins. We use A0 and A1 analogue pins, 7 and 8 digital pins, and pin number 6 for output. A0 is connected to a metal sensor denoted RV2, and A1 is connected to RV3, which is the sensor used to detect obstacles. Digital pins 7 and 8 are connected to R2 and R3. The sensors R2 and R3 are used to collect data for human detection and temperature detection. R2 has a switch (SW1) connected, which enables us to activate the sensor whenever needed. Switch SW2 is also connected to R3. Inputs from the sensor are calculated in Arduino, which can get the values, and these values are also sent through NodeMCU. GPS is used to detect the accurate location of the soldier, which will also be implemented in future work. The perfect direction and the medical-related information of the soldier can be sent to the base station in real time so that the base station can take the desired steps. The Internet of Things (IoT) with Global Positioning System (GPS) is used for tracking the soldier's location and monitoring their health, and this will have a great impact on our nation’s army.

Keywords: NodeMCU, Analog pins, Digital Pins, SW1, SW2, GPS.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSION
SW	Switch
IOT	Internet Of Things
GPS	Global Positioning System
LCD	Liquid Crystal Display
RF	Radio Frequency
IDE	Integrated Development Environment
LED	Light Emitting diode
IC	Integrated Circuit
USB	Universal Serial Bus
DC	Direct Current
GUI	Graphical User Interface

CHAPTER – I
INTRODUCTION

1.1 OVERVIEW

This work focuses on monitoring military factors including body temperature, respiration rate, and heart rate. The soldier monitoring system, which uses the Arduino Uno and Node MCU, may be described as a system used for monitoring physiological data, including parameters like heartbeat, body temperature, metrics connected to gases, etc. Based on a prototype model, the Arduino Uno is used to track the body's temperature and heartbeat rate. To track the soldier's location, the Global Positioning System (GPS) and the Internet of Things (IoT) are used. To maintain peace and ensure the safety of a country's people, the borders need to be kept under 24/7 monitoring.

Especially under current circumstances, when activities like terrorist infiltrations and the illegal movement of both living and non-living beings have become common, it becomes of the utmost importance to strictly protect the border areas against such activities. To curb such happenings in the border areas, the least that can be done is to provide a constant scenario. This monitoring takes place manually by the border security forces, which are responsible for continuously keeping an eye on the borders. It takes a lot of manpower and assets as the borders are stretched across hundreds of miles and have extreme terrain as well as climatic conditions. Hence, the need of the hour is to design an automated border surveillance system that can perform the surveillance task without requiring any human assistance.

1.2 OBJECTIVES

Nowadays, the security system of the nation depending upon the enemy's war and so the security of the soldiers is considered as an important role in it. Concerning the safety of the soldiers, there are numerous tools to observe the health condition of the soldiers. The proposed system uses GPS to track the direction of the soldier in the form of latitude and longitude values. So that direction can be found easily. The proposed system can be mounted on the soldier's body to track their health status and current location using GPS. This information will be transmitted to the control room through IoT. The proposed system comprises of tiny wearable physiological equipment's sensors, transmission modules. Hence, with the use of the proposed equipment, it is possible to implement a low-cost mechanism to protect the valuable human life on the battlefield.

1.3 CHALLENGES

- The framework for making a robot for surveillance purpose is existing.
- It only has limited range of surveillance.
- We can't control the robot with the help of laptop/mobile manually.

1.4 DESIGN ANALYSIS

The design was way more effective than we originally thought off at the start of our project. Initially our plan was to design a soldier unit that could be placed on wrist of the soldier but we couldn’t do this because of soldering performed by hand, large battery (16 batteries) and a LCD. But still we have created a reasonable unit which can be placed in a bag on back of soldier.

CHAPTER-II

LITERATURE SURVEY

2.1 LITERATURE SURVEY

1. **Novel Wearable Device for Health Monitoring and Tracking of Soldiers Based on LoRa Module – Yashash Jain, Bhupesh Soni, Ayush Goyal, Chetna Sharma, 2020, IEEE, CICT.**

   Internet has changed our life but the internet of things (IoT) is about to change this world again. This paper reports state-of-the-art wearable technology for soldiers useful in monitoring their health on battlegrounds including their exact location using IoT.

2. **IoT-based Health Monitoring via LoRaWAN – Afef Mdhaffar, Tarak Chaari, Kaouthar Larbi, Mohammed Jmaiel, Bernd Freisleben, 2017, IEEE, ICST.**

   In this paper, we present a new IoT-based health monitoring approach in which collected medical sensor data is sent to an analysis module via low-cost, low-power and secure communication links provided by a LoRaWAN network infrastructure.

3. **Design and Implementation of a Smart Soldier Uniform – Antoine Abi Zeid Daou, Christian Haddad, Roy Abi Zeid Daou, 2021, IEEE, IMCET.**

   Advancements are being made towards a cheap and effective means for health monitoring. A mobile monitoring system is proposed for monitoring using light weight, low power wireless sensors.

4. **Soldier Safety using GPS and GSM Modem – J. Swetha Priyanka, Aditi Deshpande, G. Raja Mourya, Anil Kumar, 2017, IEEE, ICIRCA.**

   The paper reports an Internet of Things (IoT) based health tracking system for soldiers. The proposed system can be mounted on the soldier's body to track their health status and current location using GPS.

5. **Implementation of Soldier Tracking and Health Monitoring System, Laxman Thakre, Nayan Patil, Prashant Kapse, Piyush Potbhare, 2022, IEEE, ICETET.**

   In modern times enemy warfare is a major factor in the sovernity of any country. National security lies in the hand of army (on the ground), the ships (at sea), the air force (in the air). An crucial and

indispensable role is played by the military. There are many concerns about the safety of the soldiers. This program will be useful for soldiers, participants in special trips or special missions. This system enables tracking of these soldiers via GPS (Global positioning systems). It could be MHealth. M-health can be explained as mobile computing, medical sensors and healthcare communications technology. In the enemy's territory soldiers are not only exposed to physical threats, but also, they feel exhausted and weary caused by prolonged work or poor sleep. Therefore, for security reasons we require a tool for distant soldiers to monitor his wellbeing. Therefore, in this project the tool is used using biosensors for the purpose of monitoring health. And a Global positioning system is used to track a soldier's location. Adding to it the GSM modem which make the system compatible wireless. In our module we have come about with the design of tracking a soldier and giving him a military position during the war.

6. **IoT based Soldier Health and Position Tracking System, J Lakshmi Prasanna, M. Ravi Kumar, Chella Santhosh, S V Aswin Kumar, P. Kasulu, 2022, IEEE, ICCMC.**

In the present scenario, the countries' security depends on the army. In this aspect, their respective more solid health and tracking are more crucial to defend themselves. In this paper, with the help of the Internet of Things (IoT) and GPS, tracking and

monitoring the health conditions of the soldiers can be done by using live track applications. The proposed device, which uses GPS for monitoring the health of soldiers and their current position, can be placed on the soldier's body. This data will be sent to the control room based on the live track program. And the proposed system is made up of small physiological devices, sensors, and transmitting modules that can be worn on the body. A low-cost, high-reliability simple lifeguard using these sensors are necessary for soldiers.

TITLE & YEAR	Description	Pros	Cons
Autonomous Boat for Fisherman Border Security using Raspberry (2019)	This often led to danger for the life of employees, soldiers and common man working or living in border areas	Timing response is good	Database isn't used
Smart Border Surveillance System using Wireless Sensor Network and Computer Vision (2020)	The movement of a robot is also controlled automatically through obstacle detecting sensors to avoiding the collision. This surveillance system using spy robot can be customized for various fields like industries, banks and shopping malls.	Simple touch of NFC enabled mobile devices can benefit	Accuracy is reduced based on the exploitation Gabor filter
Android Based Autonamous Inteligent Pod for Border Security Using Raspberry Pi (2019)	Based on the decision, a buzzer and electric current through fence for further protection can be initiated. Sensors are be integrated through IoT for an efficient control of large border area and connectivity between sites.	Alerting is possible.	Takes more time

TITLE & YEAR	Description	Pros	Cons
Implementation of spy robot for a surveillance system using Internet protocol of Raspberry Pi (2020)	The spy robot system comprises the Raspberry Pi (small single-board computer), night vision pi camera and sensors	More access point is there	Can't able to assist in cancer detection at its earlier stage
Raspberry PI based Integrated Autonomous Vehicle using LabVIEW (2019)	This paper presents the exploitation of wireless IoT system for the tethered drone PMU system.	Simple to implement and maintenance	Increases the time of the radiologist in evaluation. Hard to implement in Neural network systems.
Intelligent Border Security Intrusion Detection (2020)	Tethered drones' benefits are ideally suited for military uses such as border security and surveillance system, where day-night surveillance capabilities are crucial to monitoring perimeters.	More reliability	Timing response is not good. Take more the analysis

TABLE 2.1 LITERATURE SURVEY

CHAPTER – III

SYSTEM AND DESIGN

3.1 EXISTING SYSTEM

- Already existing systems use robots that have limited range of communication as they are based on RF Technology, Zigbee and Bluetooth.
- Some existing projects use short range wireless camera.
- Some existing robots can only be controlled with a manual mode which needs human supervision throughout the whole surveillance process.

3.1.1 CHALLENGES

- The framework for making a robot for surveillance purpose is existing.
- It only has limited range surveillance.
- We can't control the robot with the help of laptop/mobile manually.

To overcome:

- Wireless technology is one of the most integral technologies in the electronic field. This technology is used to serve our project as a supreme part of surveillance act.
- Automatic monitoring can also be done.
- Can be accessed with the laptops or mobile.
- It overcomes the problem of limited range surveillance by using the concept of IoT.

3.2 PROPOSED SYSTEM

This reports an IoT based system for the health monitoring and tracking of the soldiers. Arduino board is used which is a low-cost solution for the possessing purpose. Biomedical sensors provide heartbeat, body temperature, and environmental parameters of every soldier to control room. This technology can be helpful to provide the accurate location of missing soldier in critical condition and overcome the drawback of soldiers missing in action. The addressing system is also helpful to improve the communication between soldier to soldier in emergency situation an provide proper navigation to control room. Thus, we can conclude that this system will act as a lifeguard to the army personnel of all over the globe. In future, a portable handheld sensor device with more sensing options may be developed to aid the soldiers. This project utilizes a PIR sensor for human detection, a metal detector to detect the presence of explosives and a wireless camera for monitoring the scenario continuously at the remote station.

3.3 ARCHITECTURE DIAGRAM

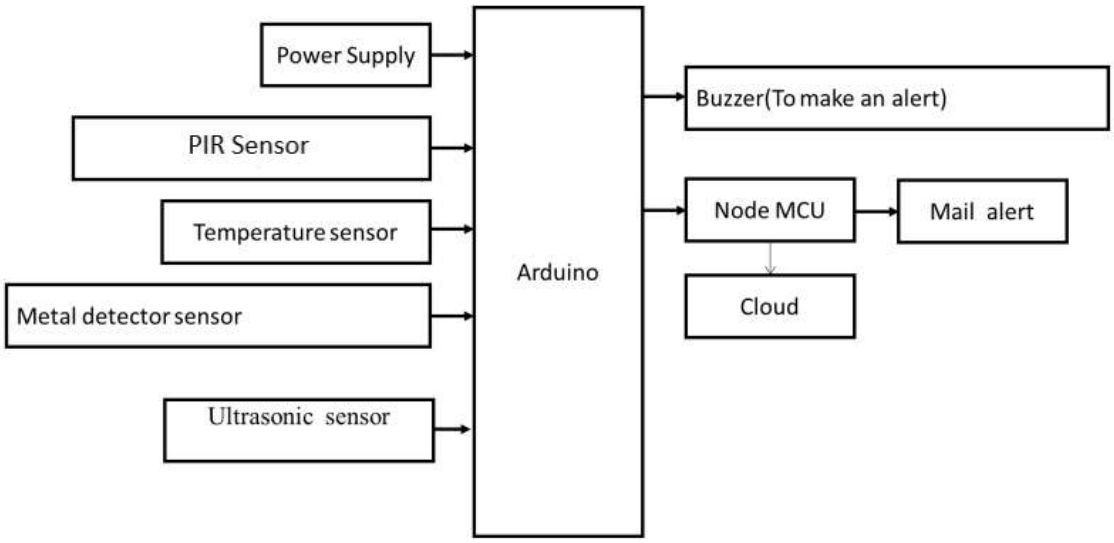

Fig 3.1: Architecture of Soldier Health Monitoring and Tracking System using IoT

3.4 HARDWARE REQUIREMENTS

Arduino Microcontroller

Arduino is an open-source platform used for building electronics projects. Arduino consists of both a physical programmable circuit board (often referred to as a microcontroller) and a piece of software, or IDE (Integrated Development Environment) that runs on your computer, used to write and upload computer code to the physical board.

The Arduino platform has become quite popular with people just starting out with electronics, and for good reason. Unlike most previous

programmable circuit boards, the Arduino does not need a separate piece of hardware (called a programmer) in order to load new code onto the board -- you can simply use a USB cable. Additionally, the Arduino IDE uses a simplified version of C++, making it easier to learn to program. Finally, Arduino provides a standard form factor that breaks out the functions of the micro-controller into a more accessible package. The Uno is one of the more popular boards in the Arduino family and a great choice for beginners.

The Arduino hardware and software was designed for artists, designers, hobbyists, hackers, newbies, and anyone interested in creating interactive objects or environments. Arduino can interact with buttons, LEDs, motors, speakers, GPS units, cameras, the internet, and even your smart-phone or your TV! This flexibility combined with the fact that the Arduino software is free, the hardware boards are pretty cheap, and both the software and hardware are easy to learn has led to a large community of users who have contributed code and released instructions for a huge variety of Arduino-based projects.

For everything from robots and a heating pad hand warming blanket to honest fortune telling machines the Arduino can be used as the brains behind almost any electronics project.

There are many varieties of Arduino boards that can be used for different purposes. Some boards look a bit different from the one below, but most Arduinos have the majority of these components in common:

Power (USB / Barrel Jack) Every Arduino board needs a way to be connected to a power source. The Arduino UNO can be powered from a USB cable coming from your computer or a wall power supply is terminated in a barrel jack.

Pins (5V, 3.3V, GND, Analog, Digital, PWM, AREF) The pins on your Arduino are the places where you connect wires to construct a circuit (probably in conjuction with a breadboard and some wire. They usually have black plastic 'headers' that allow you to just plug a wire right into the board. The Arduino has several different kinds of pins, each of which is labeled on the board and used for different functions. GND Short for 'Ground'. There are several GND pins on the Arduino, any of which can be used to ground your circuit. 5V & 3.3V As you might guess, the 5V pin supplies 5 volts of power, and the 3.3V pin supplies 3.3 volts of power. Most of the simple components used with the Arduino run happily off of 5 or 3.3 volts. Analog The area of pins under the 'Analog In' label (A0 through A5 on the UNO) are Analog In pins. These pins can read the signal from an analog sensor (like a temperature sensor) and convert it into a digital value that we can read. Digital Across from the analog pins are the digital pins (0 through 13 on the UNO). These pins can be used for both digital input (like telling if a button is pushed) and digital output (like powering an LED). PWM You may have noticed the tilde (~) next to some of the digital pins (3, 5, 6, 9, 10, and 11 on the UNO). These pins act as normal digital pins, but can also be used for something called Pulse-Width Modulation (PWM). AREF Stands for Analog Reference. Most of the time you can leave

this pin alone. It is sometimes used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the analog input pins.

Reset Button Just like the original Nintendo, the Arduino has a reset button. Pushing it will temporarily connect the reset pin to ground and restart any code that is loaded on the Arduino. This can be very useful if your code doesn't repeat, but you want to test it multiple times. Unlike the original Nintendo however, blowing on the Arduino doesn't usually fix any problems.

Power LED indicator Just beneath and to the right of the word "UNO" on your circuit board, there's a tiny LED next to the word 'ON'. This LED should light up whenever you plug your Arduino into a power source. If this light doesn't turn on, there's a good chance something is wrong. Time to re-check your circuit.

TX RX LEDs TX is short for transmit; RX is short for receive. These markings appear quite a bit in electronics to indicate the pins responsible for serial communication. In our case, there are two places on the Arduino UNO where TX and RX appear -- once by digital pins 0 and 1, and a second time next to the TX and RX indicator LEDs. These LEDs will give us some nice visual indications whenever our Arduino is receiving or transmitting data (like when we're loading a new program onto the board).

Main IC The black thing with all the metal legs is an IC, or Integrated Circuit. Think of it as the brains of our Arduino. The main IC on the Arduino is slightly different from board type to board type, but is

usually from the ATmega line of IC's from the ATMEL company. This can be important, as you may need to know the IC type (along with your board type) before loading up a new program from the Arduino software. This information can usually be found in writing on the top side of the IC. If you want to know more about the difference between various IC's, reading the datasheets is often a good idea.

Voltage Regulator The voltage regulator is not actually something you can (or should) interact with on the Arduino. But it is potentially useful to know that it is there and what it's for. The voltage regulator does exactly what it says -- it controls the amount of voltage that is let into the Arduino board. Think of it as a kind of gatekeeper; it will turn away an extra voltage that might harm the circuit. Of course, it has its limits, so don't hook up your Arduino to anything greater than 20 volts.

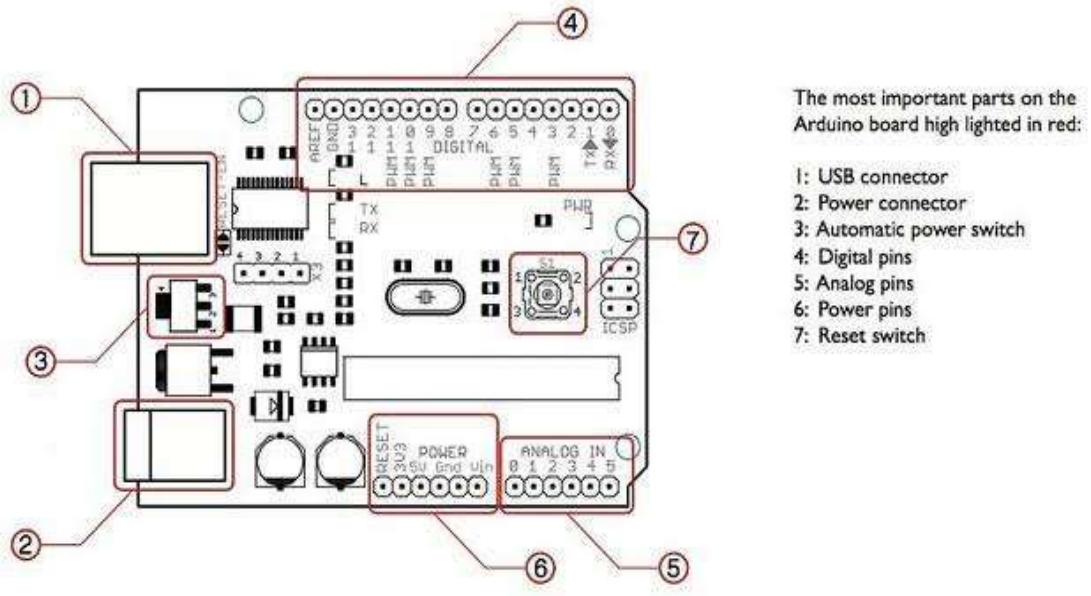

Fig 3.2: Arduino board

Node MCU

NodeMCU is an open-source LUA based firmware developed for the ESP8266 wifi chip. By exploring functionality with the ESP8266 chip, NodeMCU firmware comes with the ESP8266 Development board/kit i.e., NodeMCU Development board.

Since NodeMCU is an open-source platform, its hardware design is open for edit/modify/build. NodeMCU Dev Kit/board consist of ESP8266 wi-fi enabled chip. The ESP8266 is a low-cost Wi-Fi chip developed by Espress if Systems with TCP/IP protocol.

Operating Voltage: 2.5 to 3.3V, Operating current: 800 mA 3.3V 600mA on-board voltage regulation

ESP8266 comes up with 2 switches one is reset and another one is flash button, reset button is used to reset NodeMCU and flash button is used to download and is used while upgrading the firmware. The board has built in LED indicator which is connected to D0 pin.

The NodeMCU board also contains a CP2102 USB to UART module to convert the data from USB to serial so that it can be controlled and programmed via computer.

The esp8266 has 4 power pins: One VIN pin for input power supply and three 3.3V pins for output power supply. Even if 5V regulated supply is given through VIN, the voltage regulator will decrease it to 3.3v during output.

The esp8266 has 3 GND pins which indicate ground supply. Generally, the negative terminals are connected to these pins.

Esp8266 board also has I2C pins which can be used both as I2C master and I2C Slave. These pins are used to connect various I2C sensors and peripherals in your project. I2C interface functionality can be controlled via programming, and the clock frequency is 100 kHz at a maximum.

Esp8266 NodeMCU has 17 GPIO pins which can be assigned to various functions such as UART, PWM, I2C, IR and Button via programming. When configured as an input pin, the GPIO pins can also be set to edge-trigger or level-trigger to generate CPU interrupts.

ESP8266 NodeMCU has 2 UART interfaces, i.e. UART0 and UART1, which offer asynchronous communication, and may communicate at up to 4.5 Mbps. TXD0, RXD0, RST0 & CTS0 pins can be used for communication. It supports fluid control. However, TXD1 pin features only data transmit signal so, it's usually used for printing log. ESP8266 has two SPI in slave and master modes. These SPIs also support the following general features: 4 timing modes of the SPI format transfer. Up to 64-byte FIFO buffer. Esp8266 has a secure digital I/O interface which is used directly control the SD cards.

Esp8266 has 4 channels of Pulse width modulation (PWM). The output can be controlled via programming and is frequently used for driving motors and LEDs. The frequency ranges from 100Hz to 1KHz.

There are three control pins on the esp8266: The enable pin (EN), the reset pin (RST) and the wake pin. The esp8266 chip works when the enable pin is high. When the enable pin is low, the chip works on minimum power. The reset pin is used to reset the esp8266 chip. The wake pin is used to wake up the chip from deep sleep mode.

PIR Sensor

A Passive Infrared Sensor is an electronic sensor that measures infrared light radiating from objects. PIR sensors mostly used in PIR-based motion detectors. Also, it used in security alarms and automatic lighting applications. The below image shows a typical pin configuration of the PIR sensor, which is quite simple to understand the pinouts. The PIR sensor consists of 3 pins,

Pin 1 corresponds to the drain terminal of the device, which connected to the positive supply 5V DC.

PIN 2 corresponds to the source terminal of the device, which connects to the ground terminal via a 100K or 47K resistor. The Pin2 is the output pin of the sensor. The pin 2 of the sensor carries the detected IR signal to an amplifier.

PIN 3 of the sensor connected to the ground.

Generally, PIR Sensor can detect animal/human movement in a requirement range. PIR is made of a pyroelectric sensor, which is able

to detect different levels of infrared radiation. The detector itself does not emit any energy but passively receives it.

It detects infrared radiation from the environment. Once there is infrared radiation from the human body particle with temperature, focusing on the optical system causes the pyroelectric device to generate a sudden electrical signal.

Simply, when a human body or any animal passes by, then it intercepts the first slot of the PIR sensor. This causes a positive differential change between the two bisects. When a human body leaves the sensing area, the sensor generates a negative differential change between the two bisects.

The passive infrared sensor does not radiate energy to space. It receives the infrared radiation from the human body to make an alarm. Any object with temperature is constantly radiating infrared rays to the outside world. The surface temperature of the human body is between 36° C - 27 ° C and most of its radiant energy concentrated in the wavelength range of 8 um-12 um.

Passive infrared alarms classified into infrared detectors (infrared probes) and alarm control sections. The most widely used infrared detector is a pyroelectric detector. It uses as a sensor for converting human infrared radiation into electricity. If the human infrared radiation is directly irradiated on the detector, it will, of course, cause a temperature change to output a signal. But in doing all this, the detection distance will not be more. In order to lengthen the detection

distance of the detector, an optical system must be added to collect the infrared radiation. Usually, plastic optical reflection system or plastic Fresnel lens used as a focusing system for infrared radiation.

In the detection area, the lens of the detector receives the infrared radiation energy of the human body through the clothing and focused on the pyroelectric sensor. When the human body moves in this surveillance mode, it enters a certain field of view in sequence and then walks out of the field of view. The pyroelectric sensor sees the moving human body for a while and then does not see it, so the infrared radiation of human body constantly changes the temperature of the pyroelectric material. So that it outputs a corresponding signal, which is the alarm signal. indoor passive infrared Detection distances range from 25 cm to 20 m. Indoor curtain type the detection distance ranges from 25 cm to 20 m. Outdoor passive infrared The detection distance ranges from 10 meters to 150 meters. Outdoor passive infrared curtain detector distance from 10 meters to 150 meters.

Temperature Sensor

Temperature sensors are devices that detect and measure coolness and hotness and convert it into an electrical signal. Temperature sensors are utilized in our daily lives, be it in the form of domestic water heaters, thermometers, refrigerators, or microwaves. There is a wide range of applications of temperature sensors, including the geotechnical monitoring field and others. A temperature sensor can also be defined

as a simple instrument that measures the degree of coldness or hotness and then converts it into a readable unit. There are specialized temperature sensors used to measure the temperature of the boreholes, soil, huge concrete dams, or buildings.

Temperature sensors are devices designed for measuring the degree of coolness and hotness in an object. The voltage across the diode determines the working of a temperature meter. The change of temperature varies directly proportional to the diode's resistance. The cooler the temperature, the lesser the resistance will be and vice-versa. A measurement of the resistance across the diode is done, and the measurement is converted into units of temperature that are readable and displayed in numeric form over readout units. In the field of geotechnical monitoring, these temperature sensors are utilized in the measuring of internal temperatures of structures such as dams, bridges, power plants.

There are many different types of temperature sensors, but the most common way that is used in their categorization is based on the mode of connection that includes contact and non-contact temperature sensors. Examples of contact sensors include thermistors and thermocouples because their contact with the objects they measure is direct, whereas in non-contact type temperature sensors measure the heat source's radiation. Such temperature meters are mostly used in hazardous environments such as thermal power plants or nuclear power plants. Temperature sensors are used to measure the hydration heat in mass concrete structures, in the field of geotechnical monitoring. They

can also be utilized to monitor the migration of seepage or groundwater. One of the areas that they are most commonly used is while curing the concrete since it has to be relatively warm, in order to properly set and cure. The variations of seasons cause the expansion of structures or contraction thereby, bringing an overall change to its volume.

The working principle of a temperature sensor is the voltage across the terminals of the diode. If there is an increase in the voltage, the temperature also increases. This is followed by a drop in the voltage between the terminals of the transistor of base and emitter in a diode. There are also temperature sensors that work on the principle of stress change caused by changes in temperature. In a vibrating wire temperature meter, dissimilar metals have different linear coefficients of expansion. It mainly consists of a magnetic stretched wire of high tensile strength with two ends fixed to any dissimilar metal so that any temperature change will directly affect the tension in the wire and its natural vibration frequency. The dissimilar metal can be made from aluminium since it has a larger linear expansion coefficient than steel. When the conversion of the temperature signal into frequency occurs, the very same read-out unit that is used for other vibrating wire sensors can also be utilized in the monitoring of temperature also. The specially built vibrating wire sensor is the one that senses the temperature change and then the temperature change is converted into an electrical signal which is then transmitted to the read out the unit as a frequency.

Metal Detector

Metal detector sensor is referring to a special sensor or tools used in metal detectors that contains special designed circuits for detecting of metallic objects underground. Metal detector sensor maybe a search coil as in electromagnetic metal detectors, or refer to a special circuit contained in search probe in more complex metal detectors devices like 3D imaging devices. In metal detectors plus a principle of metal detectors and its uses in treasure hunting and other applications. Metal detectors is an electronic device specially designed for detecting of buried underground metal objects such as golden treasures, archaeological treasures of ancient civilizations and various kinds of precious and non-precious metals. Metal detectors can be used by prospectors and treasure hunters. They help detect any metal object under the ground for varying depths depending on the device type and technology.

While explaining how metal detectors work can be complicated, the principle behind their function is fairly simple metal detectors transmit an electromagnetic field toward the ground and then analyse a reflected magnetic field – resulting from hitting a metal object like a coin as it is returned from the area that the signal was originally transmitted into (the ground). In electromagnetic detectors for example there are two copper coils in the search head [search coil] of the metal detector. One coil act as a transmitter, and the other coil acts as a receiver. The first one transmits a magnetic field that is generated by electricity that moves through the coil. That magnetic field that is being transmitted

will cause electricity to flow into metal objects that it comes into contact with. The second coil, the receiver, identifies the difference in the magnetic field that is created as the buried metal absorbs it and the electricity begins to flow through it. When the change is detected, the second coil sends the alert to the control box through the attached cable, and you hear the signal from the speakers or your headphones. The weaker the returning magnetic field the weaker the alert. In other words, one coil sends, the other receives, detects changes and lets the control box know and process the difference in signals and convert it to a digital signal that can be shown as a number (Target ID) or as an audio tone.

Metal detectors are very diverse. There are many types of metal detectors which are classified according to different classifications. Metal detectors can be classified according to their uses or according to the search system and the technology in the device or according to other factors. But the most commonly used classification is according to the search technology of the device.

Arduino is an open-source hardware and software company, project and user community that designs and manufactures single-board microcontrollers and microcontroller kits for building digital devices. Arduino board designs use a variety of microprocessors and controllers. The boards are equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (‘shields’) or breadboards (for prototyping) and other circuits. The boards feature serial communications interfaces, including

Universal Serial Bus (USB) on some models, which are also used for loading programs from personal computers.

Ultrasonic Sensor

An ultrasonic sensor is an instrument that measures the distance to an object using ultrasonic sound waves. An ultrasonic sensor uses a transducer to send and receive ultrasonic pulses that relay back information about an object's proximity. High-frequency sound waves reflect from boundaries to produce distinct echo patterns. Ultrasonic sensors work by sending out a sound wave at a frequency above the range of human hearing. The transducer of the sensor acts as a microphone to receive and send the ultrasonic sound. Our ultrasonic sensors, like many others, use a single transducer to send a pulse and to receive the echo. The sensor determines the distance to a target by measuring time lapses between the sending and receiving of the ultrasonic pulse. The working principle of this module is simple. It sends an ultrasonic pulse out at 40kHz which travels through the air and if there is an obstacle or object, it will bounce back to the sensor. By calculating the travel time and the speed of sound, the distance can be calculated.

Ultrasonic sensors are a great solution for the detection of clear objects. For liquid level measurement, applications that use infrared sensors, for instance, struggle with this particular use case because of target translucence. For presence detection, ultrasonic sensors detect

objects regardless of the color, surface, or material (unless the material is very soft like wool, as it would absorb sound.) To detect transparent and other items where optical technologies may fail, ultrasonic sensors are a reliable choice.

Ultrasonic distance, level, and proximity sensors are commonly used with microcontroller platforms like Raspberry Pi, ARM, PIC, Arduino, Beagle Board, and more. Ultrasonic sensors transmit sound waves toward a target and will determine its distance by measuring the time it took for the reflected waves to return to the receiver. This sensor is an electronic device that will measure the distance of a target by transmitting ultrasonic sound waves, and then will convert the reflected sound into an electrical signal. Sensors are often used as proximity sensors. Ultrasonic sensors are also used in obstacle avoidance systems, as well as in manufacturing.

Short Range sensors offer the opportunity for closer range detection where you may need a sensor that ranges objects as close to 2cm. These are also built with very low power requirements in mind, as well as environments where noise rejection is necessary. Ultrasound is reliable in any lighting environment and can be used inside or outside. Ultrasonic sensors can handle collision avoidance for a robot, and being moved often, as long as it isn't too fast. Ultrasonics are so widely used.

3.5 SOFTWARE REQUIREMENTS

Android Application

Android applications are organized as a collection of components. There are four types of components, and applications can be composed of one or more of each type. A dynamic instance of a component corresponds to an application subset that can be executed independently of the others. So, in many ways, an Android application can be thought of as a collection of interacting components. Android application components come in four flavours. Activities. User-facing components that implement display and input capture. Services. Background components that operate independent of any user-visible activity. Broadcast receivers. A component that listens for and responds to system-wide broadcast announcements. Content providers. Components that make application data accessible to external applications and system components.

Activities. An activity component implements interactions with the user. Activities are typically designed to manage a single type of user action, and multiple activities are used together to provide a complete user interaction. For example, a mapping application may consist of two activities: one that presents to the user a list of locations to map, and one to display a map graphic that includes the chosen location. An activity includes a default window for drawing visual elements. An activity will use one or more view objects, which are organized hierarchically, to draw or capture user input. Views can be thought of

as widgets, or user-interface objects, such as check boxes, images, and lists that are common to all types of GUI-based development environments. The Android SDK includes a number of views for developer use.

Services Long-running or background components that do not directly interact with the user are expressed as service components. For example, I/O operations that are initiated by an activity may not complete before the user-facing activity disappears. In this instance, a service component can be used to carry out the I/O task, independent of the lifetime of the UI elements that initiated it. Services define and expose their own interfaces, which other components bind to in order to make use of the service. As is common with UI elements in GUI environments, services typically launch their own threads in order to allow the main application process thread to make progress and schedule threads associated with other components.

Broadcast receivers. As previously discussed, system-wide broadcast events can be generated by the system software or by applications. Components that listen to these broadcasts on behalf of applications are broadcast receivers. An application can include multiple broadcast receivers listening for announcements. In response, a broadcast receiver can initiate another component, such as an activity, to interact with the user or use the system-wide notification manager.

Content providers. Components that provide access to an application's data are content providers. Base classes are provided in the Android SDK for both the content provider (that is, the content provider

component must extend the base class) and the component seeking access. The content provider is free to store the data in whatever back-end representation it chooses, be it the file system, the SQLite service, or some application-specific representation (including those implemented via remote web services). Android applications consist of combinations of these component type instances. The invocation of components is managed through a system-wide broadcast mechanism based on intents.

Embedded C

In every embedded system-based project, Embedded C programming plays a key role to make the microcontroller run & perform the preferred actions. At present, we normally utilize several electronic devices like mobile phones, washing machines, security systems, refrigerators, digital cameras, etc. The controlling of these embedded devices can be done with the help of an embedded C program. For example, in a digital camera, if we press a camera button to capture a photo then the microcontroller will execute the required function to click the image as well as to store it.

Embedded C programming builds with a set of functions where every function is a set of statements that are utilized to execute some particular tasks. Both the embedded C and C languages are the same and implemented through some fundamental elements like a variable, character set, keywords, data types, declaration of variables,

expressions, statements. All these elements play a key role while writing an embedded C program.

The embedded system designers must know about the hardware architecture to write programs. These programs play a prominent role in monitoring and controlling external devices. They also directly operate and use the internal architecture of the microcontroller, such as interrupt handling, timers, serial communication, and other available features.

As we discussed earlier, the designing of an embedded system can be done using Hardware & Software. For instance, in a simple embedded system, the processor is the main module that works like the heart of the system. Here a processor is nothing but a microprocessor, DSP, microcontroller, CPLD & FPGA. All these processors are programmable so that it defines the working of the device.

An Embedded system program allows the hardware to check the inputs & control outputs accordingly. In this procedure, the embedded program may have to control the internal architecture of the processor directly like Timers, Interrupt Handling, I/O Ports, serial communications interface, etc.

So embedded system programming is very important to the processor. There are different programming languages are available for embedded systems such as C, C++, assembly language, JAVA, JAVA script, visual basic, etc. So, this programming language plays a key role while

making an embedded system but choosing the language is very essential.

Embedded C language is used to develop microcontroller-based applications. Embedded C is an extension to the C programming language including different features such as addressing I/O, fixed-point arithmetic, multiple-memory addressing, etc.

It processes cross development in nature. It depends on the hardware architecture of the microcontroller & other devices. Embedded C compilers are OS independent. In embedded C language, specific compilers are used. The popular compilers used in this language are Keil, BiPOM Electronics & green hill

Its format mainly depends on the kind of microprocessor used. Optimization of this language is a high level. It is not easy to modify & read. Bug fixing of this language is complicated.

CHAPTER – IV

IMPLEMENTATION

4.1 WORK PROCEDURE

The temperature sensor is precisely integrated circuit temperature sensor whose output voltage is linearly proportional to 0 C. Temperature sensor is connected to pin in Arduino and VIN provides continuous temperature in analog form. When low to high pulse is applied to the Arduino which has inbuilt ADC then it converts the analog value into digital form. When it successfully converts the value, it sends the value to the LCD and also to the base where the soldier is monitored. 1°C rise in temperature increases voltage by 10mV. The sensor thus has an advantage their linear temperature sensor calibrated in Kelvin, as the user is not required to subtract a large constant voltage from its output to obtain convenient centigrade scaling low cost is assured by trimming calibration at water level. The temperature sensors low Output impedance, linear output precise inherent calibration make interfacing to readout. In the same way other sensors are also used to monitor and calculate the data and sent to LCD and also the base.

The GPS unit contains a GPS module along with a GPS receiver antenna. The module functions according to its built and the antenna receives the information from the GPS satellite in NMEA (National Marine Electronics Association) format. This data is then sent to the micro controller wherein it is decoded to the required format and sent further. The GPS module continuously transmits serial data (RS232 protocol) in the form of sentences according to NMEA standards. The

latitude, longitude, time, date and speed values of the receiver are contained in the GPRMC sentence as given in the following example (also refer NMEA format for other sentences). In this project, these values are extracted from the GPRMC sentence and are displayed on LCD. The general NMEA (National Marine Electronics Association) format consists of an ASCII string commencing with a character.

4.2 Arduino Uno Process:

The Arduino Uno is a microcontroller-based development board that can be used to build a wide range of electronic projects. It has a number of digital and analog input/output (I/O) pins that allow it to interface with a variety of sensors and other electronic devices.

To collect information from connected sensors, you'll typically connect each sensor to a different input pin on the Arduino board. There are a few different types of input pins that you can use, depending on the type of sensor you're working with.

Analog sensors, such as temperature and light sensors, provide a continuous range of values that represent the physical quantity being measured. These sensors are connected to the analog input pins of the Arduino Uno board. The analog input pins convert the voltage values produced by the sensors into digital values that the Arduino can process. This process is done using an analog-to-digital converter (ADC) that is built into the microcontroller on the board.

Digital sensors, such as switches and motion detectors, provide binary signals that represent the presence or absence of a particular condition. These sensors are connected to the digital input pins of the Arduino Uno board. The digital input pins read the voltage level of the signal, and the Arduino interprets this as a high or low signal.

Once you've connected your sensors to the appropriate input pins on the Arduino, you can start collecting data. This is typically done using software code that runs on the microcontroller. The code will read the state or voltage level of each input pin, process the data as needed, and send the results to an output device such as an LED display, a motor, or a computer interface.

The Arduino Uno also has digital and analog output pins that allow it to control other electronic devices, such as LEDs, motors, and other actuators. These output pins can be used to provide feedback based on the sensor data that the Arduino has collected. For example, you could use an LED to indicate the temperature in a room, or a motor to adjust the position of a robotic arm based on sensor readings.

Once the Arduino has received data from the sensors, it can process and store this data using variables or arrays in its memory. The program running on the Arduino can then perform calculations, make decisions, and control output devices based on the sensor data it has received.

There are many different programming languages and development environments that can be used to write code for the Arduino, but the most common is the Arduino Integrated Development Environment

(IDE). The IDE includes a number of pre-built libraries and functions that make it easy to interface with sensors and other electronic devices.

With a little bit of programming knowledge and some basic electronics skills, you can use the Arduino to build a wide range of projects that collect and process data from connected sensors. Whether you're building a weather station, a home automation system, or a robot, the Arduino is a versatile and powerful tool that can help you bring your ideas to life.

In summary, the Arduino Uno collects information from connected sensors through its analog and digital input/output (I/O) pins. Analog sensors provide a continuous range of values that are converted into digital values using the ADC, while digital sensors provide binary signals that are interpreted as high or low signals. The Arduino can then process and store this data using variables or arrays in its memory, and use it to control output devices based on the sensor data it has received. The Arduino is a powerful and versatile tool that can be used to build a wide range of electronic projects, and with the right skills and knowledge, the possibilities are endless.

Sensors used in the project are Ultrasonic HC-SR04 Sensor, Humidity DHT11 Sensor, Metal detector Tube Type Inductive Proximity Sensor Detection Switch NPN DC6-36V 4mm Normally Open switch LJ12A3-4-Z/BX, Infrared Obstacle Avoidance IR Sensor Module, Heart Beat Sensors LM358.

Ultrasonic HC SR04 Sensor:

The ultrasonic HC SR04 sensor is a popular sensor that measures distance using sound waves. It sends out ultrasonic waves and measures the time it takes for the waves to bounce back after hitting an object. The distance can then be calculated using the speed of sound.

To connect the HC-SR04 sensor to the Arduino Uno, you need to connect the Vcc pin of the sensor to the 5V pin on the Arduino Uno. Then, connect the GND pin of the sensor to the GND pin on the Arduino Uno. Next, connect the Trig pin of the sensor to a digital output pin on the Arduino Uno. Finally, connect the Echo pin of the sensor to a digital input pin on the Arduino Uno.

To use the HC-SR04 sensor in your Arduino Uno project, you need to write a program that sends a trigger signal to the Trig pin, waits for the Echo pin to go high, and then measures the duration of the high signal. The duration can be converted to distance using the formula: distance = duration * speed_of_sound / 2, where speed_of_sound is the speed of sound in air (approximately 343 m/s at room temperature).

The HC-SR04 ultrasonic sensor is commonly used to measure distance. It works by sending out an ultrasonic pulse and measuring the time it takes for the pulse to bounce back. Here's how to connect and use the HC-SR04 sensor with the Arduino Uno:

- Connect the Vcc pin of the HC-SR04 to the 5V pin on the Arduino Uno.

- Connect the GND pin of the HC-SR04 to the GND pin on the Arduino Uno.
- Connect the Trigger pin of the HC-SR04 to a digital output pin on the Arduino Uno.
- Connect the Echo pin of the HC-SR04 to a digital input pin on the Arduino Uno.

After you've connected the sensor, you can write a program in the Arduino IDE to read the distance data from the sensor. Our code reads the distance data from the HC-SR04 sensor and prints it to the Serial Monitor. The distance value is calculated using the formula: distance = duration * 0.034 / 2. The value 0.034 represents the speed of sound in cm/microsecond, and the division by 2 is necessary because the ultrasonic pulse travels to the target and back to the sensor.

Humidity DHT11 Sensor:

The DHT11 sensor is a low-cost sensor that measures temperature and humidity. It has a single-wire digital interface and can be used with the Arduino Uno to monitor the environment.

To connect the DHT11 sensor to the Arduino Uno, you need to connect the Vcc pin of the sensor to the 5V pin on the Arduino Uno. Then, connect the GND pin of the sensor to the GND pin on the Arduino Uno. Finally, connect the Data pin of the sensor to a digital input/output pin on the Arduino Uno.

To use the DHT11 sensor in your Arduino Uno project, you need to write a program that reads the data from the sensor using the digital input/output pin. The data is transmitted by the sensor in a specific format, which includes a 40-bit data stream that contains the temperature and humidity readings.

The DHT11 humidity sensor is a commonly used sensor that measures temperature and humidity. Here's how to connect and use the DHT11 sensor with the Arduino Uno:

- Connect the Vcc pin of the DHT11 to the 5V pin on the Arduino Uno.
- Connect the GND pin of the DHT11 to the GND pin on the Arduino Uno.
- Connect the data pin of the DHT11 to a digital input/output pin on the Arduino Uno.

After you've connected the sensor, We have to write a program in the Arduino IDE to read the temperature and humidity data from the sensor.

Metal Detection Sensor Inductive Type:

The tube type inductive proximity sensor is a type of sensor that detects the presence of metal objects. It works by generating an electromagnetic field and detecting changes in the field when a metal object is present.

To connect the inductive proximity sensor to the Arduino Uno, you need to connect the Brown wire of the sensor to the 5V pin on the Arduino Uno. Then, connect the blue wire of the sensor to the GND pin on the Arduino Uno. Finally, connect the Black wire of the sensor to a digital input pin on the Arduino Uno.

To use the inductive proximity sensor in your Arduino Uno project, you need to write a program that reads the data from the digital input pin connected to the sensor. When a metal object is present, the sensor will detect a change in the electromagnetic field and output a high signal.

Tube Type Inductive Proximity Sensor Detection Switch NPN DC6-36V 4mm Normally Open switch LJ12A3-4-Z/BX:

The inductive proximity sensor is used to detect metal objects without physical contact. Here's how to connect and use the inductive proximity sensor with the Arduino Uno:

- Connect the Brown wire of the sensor to the 5V pin on the Arduino Uno.
- Connect the Blue wire of the sensor to the GND pin on the Arduino Uno.
- Connect the Black wire of the sensor to a digital input pin on the Arduino Uno.

After you've connected the sensor, We have to write a program in the Arduino IDE to detect the presence of a metal object. Our code reads

the digital input from the sensor and prints a message to the Serial Monitor depending on whether a metal object is detected or not.

Infrared Obstacle Avoidance IR Sensor Module:

The infrared obstacle avoidance sensor is a type of sensor that detects the presence of objects using infrared light. It sends out an infrared signal and measures the amount of reflected light to determine the presence of an object.

To connect the infrared obstacle avoidance sensor to the Arduino Uno, you need to connect the Vcc pin of the sensor to the 5V pin on the Arduino Uno. Then, connect the GND pin of the sensor to the GND pin on the Arduino Uno. Finally, connect the Out pin of the sensor to a digital input pin on the Arduino Uno.

To use the infrared obstacle avoidance sensor in your Arduino Uno project, you need to write a program that reads the data from the digital input pin connected to the sensor. When an object is detected, the sensor will output a high signal.

The IR sensor module is commonly used for obstacle avoidance in robotics projects. It works by emitting an infrared beam and measuring the reflection of the beam from an object. Here's how to connect and use the IR sensor module with the Arduino Uno:

- Connect the Vcc pin of the IR sensor module to the 5V pin on the Arduino Uno.

- Connect the GND pin of the IR sensor module to the GND pin on the Arduino Uno.
- Connect the OUT pin of the IR sensor module to a digital input pin on the Arduino Uno.

After you've connected the sensor, we have to write a program in the Arduino IDE to detect the presence of an obstacle. Our code reads the digital input from the sensor and prints a message to the Serial Monitor depending on whether an obstacle is detected or not.

Heartbeat Sensor LM358:

The LM358 heartbeat sensor is a type of sensor that measures the heart rate by detecting changes in blood volume. It works by shining a light through the skin and detecting changes in the amount of light that is reflected back.

To connect the heartbeat sensor to the Arduino Uno, you need to connect the Vcc pin of the sensor to the 5V pin on the Arduino Uno. Then, connect the GND pin of the sensor to the GND pin on the Arduino Uno. Finally, connect the Out pin of the sensor to an analog input pin on the Arduino Uno.

To use the heartbeat sensor in your Arduino Uno project, you need to write a program that reads the analog data from the analog input pin connected to the sensor. The data can be processed to calculate the heart rate.

The LM358 heart rate sensor module is commonly used to measure heart rate. It works by detecting the change in blood flow through the finger or earlobe. Here's how to connect and use the LM358 heart rate sensor module with the Arduino Uno:

- Connect the Vcc pin of the LM358 heart rate sensor module to the 5V pin on the Arduino Uno.
- Connect the GND pin of the LM358 heart rate sensor module to the GND pin on the Arduino Uno.
- Connect the S pin of the LM358 heart rate sensor module to an analog input pin on the Arduino Uno.

After you've connected the sensor, we have to write a program in the Arduino IDE to read the heart rate data from the sensor. Our code reads the analog input from the sensor and prints the heart rate value to the Serial Monitor

Overall, to use these sensors with Arduino Uno, you need to connect them properly to the board, write the code to read data from them, and then process and use that data in your project. It's important to follow the specifications and guidelines provided by the sensor manufacturer and also consult the Arduino documentation for proper usage.

In summary, these are the steps to connect and use the sensors with the Arduino Uno:

Ultrasonic HC SR04 Sensor:

- Connect the Vcc pin of the HC-SR04 sensor to the 5V pin on the Arduino Uno.
- Connect the GND pin of the HC-SR04 sensor to the GND pin on the Arduino Uno.
- Connect the Trig pin of the HC-SR04 sensor to a digital output pin on the Arduino Uno.
- Connect the Echo pin of the HC-SR04 sensor to a digital input pin on the Arduino Uno.

Humidity DHT11 Sensor:

- Connect the Vcc pin of the DHT11 sensor to the 5V pin on the Arduino Uno.
- Connect the GND pin of the DHT11 sensor to the GND pin on the Arduino Uno.
- Connect the Data pin of the DHT11 sensor to a digital input/output pin on the Arduino Uno.

Tube Type Inductive Proximity Sensor Detection Switch NPN DC6-36V 4mm Normally Open Switch LJ12A3-4-Z/BX (Metal Detection Sensor):

- Connect the Brown wire of the sensor to the 5V pin on the Arduino Uno.

- Connect the Blue wire of the sensor to the GND pin on the Arduino Uno.
- Connect the Black wire of the sensor to a digital input pin on the Arduino Uno.

Infrared Obstacle Avoidance IR Sensor Module:

- Connect the Vcc pin of the IR sensor module to the 5V pin on the Arduino Uno.
- Connect the GND pin of the IR sensor module to the GND pin on the Arduino Uno.
- Connect the OUT pin of the IR sensor module to a digital input pin on the Arduino Uno.

Heart Beat Sensors LM358:

- Connect the Vcc pin of the LM358 heart rate sensor module to the 5V pin on the Arduino Uno.
- Connect the GND pin of the LM358 heart rate sensor module to the GND pin on the Arduino Uno.
- Connect the S pin of the LM358 heart rate sensor module to an analog input pin on the Arduino Uno.

Once you've connected the sensors to the Arduino Uno, you can write a program in the Arduino IDE to read data from the sensors and perform actions based on that data. The exact code you write will depend on the specific sensor and the task you want to perform with it, but the examples provided in this answer should give you a good starting point.

4.2.1 Arduino Work Procedure:

Our code is written in Python and uses the pandas library to read data from a CSV file and perform some data analysis.First, the pandas library is imported and assigned the alias pd. This library is commonly used for data manipulation and analysis in Python.

Next, the read_csv function from pandas is used to read data from a CSV file named data.csv and assign it to a variable called data. The head function is then used to display the first five rows of the data.

The describe function is then called on the data variable to generate summary statistics for the numerical columns in the data. This includes the count, mean, standard deviation, minimum value, and maximum value for each column.

The groupby function is used to group the data by the sex column, and the mean function is called to calculate the mean value for each column within each group. The resulting data is assigned to a variable called grouped_data. the to_csv function is used to write the grouped_data variable to a new CSV file named grouped_data.csv.

This code is written in Python programming language and is intended to read a CSV (comma-separated values) file, perform some operations on the data, and then write the output to a new CSV file.

The code first imports the csv module, which provides functionality to read and write CSV files. Next, it defines a function named

process_file(), which takes two arguments: input_file and output_file. This function is responsible for reading the input CSV file, performing some operations on the data, and writing the output to the output CSV file. The first line of the process_file() function opens the input file in "read" mode using the with statement. The csv.reader function is used to create a reader object, which can be used to iterate over the rows of the input file.

The second line of the function defines a list named header, which will contain the header row of the input file.

The third line of the function uses the next() function to skip the header row of the input file and move the reader object to the next row. The fourth line of the function defines an empty list named output_data, which will eventually contain the transformed data. The fifth line of the function uses a for loop to iterate over the remaining rows of the input file. For each row, it extracts the values of the "date" and "value" columns and converts them to the appropriate data types (datetime and float, respectively).

The sixth line of the function performs some operation on the data. In this case, it calculates the average value for each month and adds it to a dictionary object named monthly_averages. This dictionary object stores the average value for each month as a key-value pair, where the key is a string representation of the month and the value is the

calculated average.The seventh line of the function appends the transformed data to the output_data list.

After the for loop finishes iterating over all the rows in the input file, the function opens the output file in "write" mode and uses the csv.writer function to create a writer object. The writerow() function is used to write the header row to the output file.

The for loop is then used to iterate over the items in the monthly_averages dictionary object. For each key-value pair, the function creates a list containing the month and the average value, and then writes that list to the output file using the writerow() function.

Finally, the function closes both the input and output files using the with statement, ensuring that they are properly closed and that any changes made to the files are saved.

Overall, this code is a great example of how Python can be used to efficiently read, process, and write data from CSV files. It uses a combination of built-in Python functions and the csv module to accomplish this task.

4.3 NodeMCU Process:

The Arduino Uno is a microcontroller board that is used to collect data from sensors and send it to other devices, such as the NodeMCU. The Uno can communicate with the sensors using various communication

protocols such as I2C, SPI, and UART. It has analog and digital input/output pins that can be used to interface with a wide range of sensors.

The NodeMCU is a development board that uses the ESP8266 chip to provide Wi-Fi connectivity. It can receive data from the Uno using a serial communication protocol such as UART. The two devices are connected using a serial cable, which is typically a USB cable.

The process of collecting and sending data from the sensors to the NodeMCU begins with the Arduino Uno. The Uno has pins that are connected to the sensors, and it reads the data from these pins using the appropriate communication protocol. The Uno then processes the data and sends it to the NodeMCU using the serial communication protocol.

The NodeMCU receives the data from the Uno using its UART receiver. It then processes the data and sends it to the desired location, which could be a web server, a cloud service, or another device. The NodeMCU can also perform additional processing on the data if required.

In order to make the process of collecting and sending data more efficient, the Uno and NodeMCU can be programmed to work together. For example, the Uno can be programmed to only send data when it is required, and the NodeMCU can be programmed to receive the data and send it to the desired location automatically. This can reduce the amount of data that is sent over the serial cable, making the system more efficient and reducing the load on the communication channels.

The Arduino Uno collects data from sensors using various communication protocols and sends it to the NodeMCU using a serial communication protocol. The NodeMCU receives the data and processes it, and then sends it to the desired location. The two devices can be programmed to work together in order to make the process more efficient.

The process of transferring data from sensors connected to an Arduino Uno to a NodeMCU involves several steps, and understanding these steps is crucial for effectively implementing such a system.

The first step in this process is to connect the sensors to the Arduino Uno. Different sensors may require different connections, and it is important to refer to their data sheets to ensure the proper connections are made. For example, the ultrasonic HC-SR04 sensor requires connections to the VCC, GND, Trig, and Echo pins, while the DHT11 humidity sensor requires connections to the VCC, GND, and data pins. Similarly, the inductive proximity sensor, the infrared obstacle avoidance sensor, and the heart beat sensor each have their own specific connection requirements. Once all the sensors are connected to the Arduino Uno, the next step is to program it to read data from the sensors.

To read data from the sensors, we need to write code that defines the pins connected to the sensors and the protocols used to communicate with them. For example, to read data from the HC-SR04 sensor, we need to define the trigger pin as an output and the echo pin as an input,

and then send a trigger pulse to the trigger pin to start the measurement. The duration of the pulse on the echo pin corresponds to the distance of the object from the sensor. Similarly, to read data from the DHT11 humidity sensor, we need to define the data pin as an input and use the DHT library to read the temperature and humidity values from the sensor.

Once the Arduino Uno is programmed to read data from the sensors, the next step is to transmit the data to the NodeMCU. To achieve this, we can use a serial communication protocol such as UART, SPI, or I2C. In this case, we will use UART because it is the simplest and most widely used protocol. We need to connect the RX and TX pins of the Arduino Uno to the TX and RX pins of the NodeMCU, respectively. We can then write code on the Arduino Uno to send the data to the NodeMCU over UART.

On the NodeMCU side, we need to write code to receive the data sent by the Arduino Uno over UART. We can use the SoftwareSerial library to define a software serial port on the NodeMCU that will receive the data. We then need to parse the data received from the Arduino Uno to extract the relevant sensor readings. This can be done by sending the data in a predefined format, such as comma-separated values (CSV), where each sensor reading is separated by a comma. Once the data is parsed, we can then use it to perform various tasks such as sending it to a remote server or displaying it on a web page.

Overall, the process of transferring data from sensors connected to an Arduino Uno to a NodeMCU involves connecting the sensors to the Arduino Uno, programming it to read data from the sensors, transmitting the data to the NodeMCU over UART, and parsing the data on the NodeMCU to extract the relevant sensor readings. This requires a good understanding of both the hardware and software involved and may require some experimentation to get everything working smoothly. However, once everything is set up correctly, it can provide a powerful platform for collecting and analyzing data from multiple sensors in real-time. Details of how these sensors are connected to Arduino Uno and how the data is transferred to NodeMCU are explained below.

Ultrasonic HC-SR04 Sensor:

The HC-SR04 sensor is an ultrasonic sensor that uses sound waves to detect the distance of an object. It has two main components - a transmitter and a receiver. The transmitter sends out a sound wave and the receiver detects the sound wave when it bounces back from an object. The time it takes for the sound wave to bounce back is used to calculate the distance of the object.

To connect the HC-SR04 sensor to Arduino Uno, you need to connect the VCC pin to 5V, the GND pin to GND, the Trig pin to any digital pin, and the Echo pin to another digital pin. The Trig pin is used to send

a trigger signal to the sensor, and the Echo pin is used to receive the echo signal.

To read the distance measurement, you need to send a trigger signal by setting the Trig pin to HIGH for at least 10 microseconds. Then, you need to wait for the echo signal to be received by checking the value of the Echo pin. The time it takes for the signal to be received is proportional to the distance of the object. This time is measured using the pulseIn() function and converted into distance using the speed of sound.

To transfer the data from the HC-SR04 sensor to NodeMCU, the distance measurement is sent through a serial connection using the Serial.print() function. The NodeMCU can then receive the data using a serial connection and process it as needed.

Humidity DHT11 Sensor:

The DHT11 sensor is a digital humidity and temperature sensor that uses a capacitive humidity sensor and a thermistor to measure the surrounding air. It provides a digital signal output that can be easily read by Arduino Uno.

To connect the DHT11 sensor to Arduino Uno, you need to connect the VCC pin to 5V, the GND pin to GND, and the data pin to any digital pin. The data pin is used to communicate with the sensor using a digital signal.

To read the humidity and temperature measurements, you need to send a start signal to the sensor by setting the data pin to LOW for at least 18 milliseconds, then set it to HIGH for at least 20 microseconds. The sensor will then send a response signal and start transmitting the data. The data consists of 40 bits of information, including the humidity and temperature measurements.

To transfer the data from the DHT11 sensor to NodeMCU, the humidity and temperature measurements are sent through a serial connection using the Serial.print() function. The NodeMCU can then receive the data using a serial connection and process it as needed.

Metal Detection Sensor:

The LJ12A3-4-Z/BX sensor is a type of inductive proximity sensor that detects the presence of metal objects. It generates a magnetic field and detects changes in the field caused by the presence of metal objects.

To connect the LJ12A3-4-Z/BX sensor to Arduino Uno, you need to connect the VCC pin to 5V, the GND pin to GND, and the output pin to any digital pin. The output pin is used to communicate with the sensor using a digital signal.

The Tube Type Inductive Proximity Sensor Detection Switch NPN DC6-36V 4mm Normally Open Switch LJ12A3-4-Z/BX, commonly known as a Metal Detection Sensor, is a device that can detect the presence of metal objects without physical contact. It works on the

principle of electromagnetic induction, where a magnetic field is generated by the sensor coil, and any metal object in the vicinity of the sensor disturbs this magnetic field, producing an electrical signal that indicates the presence of the object.

To connect this sensor to the Arduino Uno, first, we need to connect the black wire of the sensor to GND pin, the brown wire to the +V pin, and the blue wire to the digital pin 2 of the Arduino. We also need to add a pull-up resistor to the digital pin 2 to ensure stable operation.

To use this sensor, we need to set the digital pin 2 as an input pin in the Arduino code. Then, we can use the digitalRead() function to read the state of the pin. If a metal object is present near the sensor, the pin will be pulled low, indicating the presence of the object. We can also use the attachInterrupt() function to trigger an interrupt when the pin state changes, allowing us to perform specific actions in response to the presence of a metal object.

To transfer the collected information from the Metal Detection Sensor to the NodeMCU, we can use the serial communication protocol. We can connect the TX pin of the Arduino to the RX pin of the NodeMCU using a jumper wire. Then, we can use the Serial.print() function to send the sensor data as a string over the serial connection. On the NodeMCU side, we can use the Serial.read() function to receive the data and process it as needed.

Infrared Obstacle Avoidance IR Sensor Module:

The Infrared Obstacle Avoidance IR Sensor Module is used to detect obstacles in front of the robot. It emits infrared radiation and detects the reflection of the radiation from any obstacles in front of it. The module consists of an infrared emitter and a receiver, and it has a detection range of about 2-40 cm. The output of the module is either high or low depending on whether an obstacle is detected or not.

To use the IR Sensor Module, it needs to be connected to the Arduino board. The module has three pins: VCC, GND, and OUT. The VCC pin should be connected to the 5V pin on the Arduino board, the GND pin should be connected to the GND pin on the board, and the OUT pin should be connected to any of the digital pins on the board.

Heartbeat Sensors LM358:

The heartbeat sensor LM358 is used to measure the heart rate of an individual. It works on the principle of detecting the change in blood volume in the fingertip with each heartbeat. The module consists of an infrared LED and a photodiode. The LED emits infrared radiation, which is reflected by the blood in the fingertip. The photodiode detects the amount of reflected radiation and generates an electrical signal, which is proportional to the blood volume in the fingertip.

To use the Heartbeat Sensor LM358, it needs to be connected to the Arduino board. The module has three pins: VCC, GND, and OUT. The

VCC pin should be connected to the 5V pin on the Arduino board, the GND pin should be connected to the GND pin on the board, and the OUT pin should be connected to any of the analog pins on the board.

Connecting Sensors to NodeMCU:

After connecting all the sensors to the Arduino board, the data from the sensors is sent to the NodeMCU board using the serial communication protocol. The NodeMCU board has a built-in Wi-Fi module that can be used to connect to the internet or a local network.

To connect the NodeMCU board to the Arduino board, follow these steps:

- Connect the VCC and GND pins of the NodeMCU board to the 5V and GND pins of the Arduino board, respectively.
- Connect the TX pin of the NodeMCU board to the RX pin of the Arduino board.
- Connect the RX pin of the NodeMCU board to the TX pin of the Arduino board.

Sending Sensor Data from Arduino to NodeMCU:

To send the sensor data from the Arduino board to the NodeMCU board, the Arduino board needs to be programmed to read the data from the sensors and send it to the NodeMCU board using the serial communication protocol. The Arduino code for reading data from the sensors and sending it to the NodeMCU board is completed

Once all the sensors are connected to the Arduino, the Arduino will collect the data from all the sensors and send it to the NodeMCU. This can be done using serial communication between the Arduino and the NodeMCU. The Arduino will use the Serial.print() function to send the sensor data over the serial port, and the NodeMCU will use the Serial.read() function to read the data from the serial port.

To establish serial communication between the Arduino and the NodeMCU, connect the TX pin of the Arduino to the RX pin of the NodeMCU and the RX pin of the Arduino to the TX pin of the NodeMCU. Make sure that the baud rate is set to the same value on both the Arduino and the NodeMCU, usually 9600.

In conclusion, connecting sensors to an Arduino and sending the data to a NodeMCU involves identifying the pins of the sensors and connecting them to the appropriate pins on the Arduino. Once the data is collected from all the sensors, it can be sent to the NodeMCU using serial communication. This allows for the collection and analysis of data from multiple sensors in a single system.

4.3.1 NodeMCU Work Procedure:

This code is designed to read data from multiple sensors connected to an Arduino Uno and send it to the Ubidots cloud platform using a NodeMCU board via Wi-Fi. The code uses a SoftwareSerial library to establish serial communication between the NodeMCU and the Arduino Uno. The Ubidots library is used to create a connection with the Ubidots platform and send sensor data. The code initializes the

required variables and sets up the connection with Ubidots by connecting to the Wi-Fi network.

The loop function reads the sensor values sent by the Arduino Uno through the SoftwareSerial connection. It then parses the values by splitting the received string using comma separator and stores each value in a separate variable. The parsed values are then added to the Ubidots data buffer using the corresponding variable names that are defined in the code.

The Ubidots send() function is then used to send the data to the Ubidots platform. If the data is sent successfully, a message is printed to the serial monitor indicating that the data has been sent. The loop function then waits for a certain delay before repeating the process. code demonstrates how to send sensor data from an Arduino Uno to the Ubidots cloud platform using a NodeMCU board. It provides a simple and effective solution for IoT projects that require real-time data monitoring and analysis.

This code is used to send sensor data to Ubidots IoT cloud platform using a NodeMCU board. The data is collected from an Arduino board which is connected to various sensors. The communication between the Arduino board and the NodeMCU board is established using serial communication. The NodeMCU board reads the data received from the Arduino board and sends it to the Ubidots platform using the Ubidots library.

The first few lines of the code include the required libraries and the initialization of the serial communication between the NodeMCU board and the Arduino board. The SoftwareSerial library is used to establish serial communication on the pins 14 and 12 of the NodeMCU board. The Ubidots library is also included, which is used to send data to the Ubidots cloud platform. The Ubidots token, WiFi SSID, and WiFi password are defined as constants.

The Ubidots object is then created with the token and the communication protocol is defined as HTTP. In the setup function, the serial communication is initiated with a baud rate of 115200, and the WiFi connection is established with the provided SSID and password. A delay of 500ms is introduced to ensure the WiFi connection is established before sending any data.

In the loop function, the code checks if any data is available on the serial communication line. If data is available, the data is read as a string. The string is then parsed to extract the values of each sensor data. The values of each sensor are then stored in separate variables. The Ubidots library is then used to add the data of each sensor as a variable with the respective name, and the value is passed as a parameter. Once all the data is added, the send function of the Ubidots library is called to send the data to the Ubidots platform. If the data is successfully sent, the message "Values sent by the device" is printed on the serial monitor and the code collects sensor data from an Arduino board and sends it to the Ubidots IoT cloud platform using a NodeMCU board. The data is transmitted using serial communication between the

two boards, and the Ubidots library is used to send the data to the Ubidots platform.

The code starts with including the SoftwareSerial library for creating a software serial port and Ubidots library for sending the sensor data to the cloud platform. The code defines a constant character pointer 'UBIDOTS_TOKEN' which stores the Ubidots account token for authentication and two character pointers 'WIFI_SSID' and 'WIFI_PASS' which store the Wi-Fi credentials.

The code then initializes an instance of the 'Ubidots' class with the token and the communication protocol 'UBI_HTTP'. The 'setup()' function initializes the serial communication with a baud rate of 115200 and the software serial port with RX and TX pins at 14 and 12 respectively. The code then establishes a Wi-Fi connection by calling the 'wifiConnect()' function of the 'Ubidots' class with the SSID and password as arguments. A delay of 500 milliseconds is added for the connection to be established. The 'loop()' function is where the sensor values are read and sent to the Ubidots cloud platform. The 'if' statement checks if any data is available on the software serial port, which indicates that sensor data is being received from the Arduino. If data is available, the received data is stored as a string in variable 'a' and printed to the serial monitor.

The 'for' loop is used to extract the sensor values from the received data string. The loop iterates through the string and checks for the presence of commas (',') which separate the sensor values. When a comma is

found, the code extracts the substring between the previous and current comma and stores it in the corresponding integer variable, converting the substring to an integer using the 'toInt()' function.

After extracting all the sensor values, the code adds them to the 'Ubidots' instance using the 'add()' function. The 'add()' function takes two arguments, the label for the sensor value and the actual value. The code then sends the data to the Ubidots cloud platform using the 'send()' function of the 'Ubidots' class, which sends all the added data to the Ubidots cloud platform for storage and analysis.

The program is written in the Arduino programming language. It uses several libraries including SoftwareSerial and Ubidots. The program reads data from a serial port and sends the values to Ubidots, an IoT platform for data analysis.

The program starts by declaring the necessary variables and libraries. It defines the Ubidots token, Wi-Fi SSID, and Wi-Fi password. It also defines an instance of the Ubidots class with the given token and HTTP as the communication protocol.

In the setup() function, it starts the Serial communication and initializes the SoftwareSerial object for communication on pins 14 and 12. It then connects to the Wi-Fi network using the provided SSID and password and waits for 500 milliseconds.

The loop() function is the main loop of the program that runs continuously. It checks if there is any data available on the serial port using the mySerial.available() function. If there is data, it reads it as a

string using the mySerial.readString() function and stores it in a variable called a. It then prints the received string to the Serial Monitor for debugging purposes and waits for 50 milliseconds.

Next, it loops through each character of the received string a using a for loop. It checks if the character is a comma (',') and increments a variable j that counts the number of commas encountered. It then extracts the values between the commas using the a.substring() function and converts them to integers using the toInt() function.

It stores each integer value in a separate variable based on the number of commas encountered. It also prints each value to the Serial Monitor for debugging purposes. Finally, it adds each value to the corresponding variable in the Ubidots class using the ubidots.add() function. If the program detects the fifth value, it sends all the added values to Ubidots using the ubidots.send() function. If the data is sent successfully, it prints a message to the Serial Monitor.

Overall, this program reads data from a serial port and sends it to Ubidots for data analysis. The data is parsed based on the number of commas encountered, and each value is added to a corresponding variable in Ubidots using the Ubidots library. The code prints a message to the serial monitor indicating if the data was sent successfully. The 'loop()' function then repeats, waiting for new sensor data to be received from the Arduino. This process continues until the program is stopped.

4.4 Sensor I/O:

Ultrasonic HC-SR04 Sensor:

Inputs:

Trigger signal from Arduino (5V)

Echo signal received from the object being measured

Outputs:

Distance (in cm) between the sensor and the object being measured

Humidity DHT11 Sensor:

Inputs:

Vcc (5V)

Ground (GND)

Data pin (connected to Arduino digital pin)

Outputs:

Temperature (in Celsius)

Relative humidity (in percentage)

Tube Type Inductive Proximity Sensor:

Inputs:

Vcc (6-36V)

Ground (GND)

Detection switch (connected to Arduino digital pin)

Outputs:

High or Low signal depending on whether a metal object is detected or not

Infrared Obstacle Avoidance IR Sensor Module:

Inputs:

Vcc (5V)

Ground (GND)

Data pin (connected to Arduino digital pin)

Outputs:

High or Low signal depending on whether an obstacle is detected or not

Heart Beat Sensors LM358:

Inputs:

Vcc (5V)

Ground (GND)

Analog signal from heart rate sensor (connected to Arduino analog pin)

Outputs:

Analog signal proportional to the heart rate (in beats per minute)

4.5 Cloud Storage:

Ubidots is an IoT platform that allows users to store, visualize, and analyze data from connected devices. It provides cloud-based storage for data collected from IoT devices, making it easy to manage and access data from anywhere with an internet connection. Ubidots offers various features that can be useful for data storage and analysis, including:

Data visualization: Ubidots provides customizable dashboards and widgets that allow users to visualize their data in real-time. This can help users to identify trends, patterns, and anomalies in their data.

Data analysis: Ubidots allows users to perform data analysis using tools like machine learning algorithms and statistical analysis. This can help users to gain insights into their data and make data-driven decisions.

Data integration: Ubidots can integrate with a wide range of devices, sensors, and platforms, making it easy to collect and store data from various sources.

In our code, the Ubidots library is used to store data from the five parameters - temperature, heart rate, motion, metal, and distance. Each parameter is represented as a variable in the Ubidots class and is updated with the latest value received from the Arduino board. The data can be monitored and analyzed in real-time using the Ubidots dashboard.

To view the data in Ubidots, the user needs to create a device label that matches the device ID used in the Arduino code. The five parameters can be added as variables in the device label. Once the data is sent from the Arduino board, it gets updated in the Ubidots dashboard. The user can view the latest data as well as the previous data stored in the Ubidots cloud.

Ubidots provides various options for data visualization, including line charts, bar charts, and gauges. The user can choose the appropriate visualization based on the type of data and the analysis required. The Ubidots dashboard also provides options to set thresholds and alerts based on the data received, which can be useful for real-time monitoring and alerts.

In summary, Ubidots provides a cloud-based platform for storing, visualizing, and analyzing IoT data. The Ubidots library in the Arduino code allows users to send data from the Arduino board to the Ubidots cloud, where it can be monitored and analyzed in real-time. The Ubidots dashboard provides various options for data visualization and analysis, making it a useful tool for IoT data management.

CHAPTER – V

CONCLUSION AND FUTURE WORK

5.1 CONCLUSION

The subjective of this paper is to present the information about the soldier monitoring system is successfully implemented and executed which can be capable of collect and process the physiological parameters from the human body. In future we can include the solar harvesting system to recharge the DC power source automatically when user is exposed to sun and we can also interface the camera which will helpful to the doctors / concerned persons to view the soldier activities remotely. In the future, we can include the solar harvesting system to recharge the DC power source automatically when the user is exposed to the sun and we can also interface the camera which will help the doctors/concerned persons to view the patient activities remotely.

The design was way more effective than we originally thought off at the start of our project. We tried following ethics in designing and implementation of the project. We won't claim that our circuit had 100% efficiency, as it did show some variance that we minimized to some extent. The good thing, we noted that there is a lot of possibility to make enhancements in this project. Our system is for one soldier. The communication between soldiers to soldier can be established. This system gives strength to the defence system of our country. So, we can accomplish that these types of strategies are very supportive for certifying security of the soldiers.

5.2 FUTURE WORK

The soldier health monitoring and location tracking system has great potential for future enhancements. One area for improvement is the ability to support multiple soldiers. While the current system is designed to monitor the health and location of a single soldier, extending this functionality to support multiple soldiers could be valuable. For instance, communication between soldiers could be established, enabling them to share information and coordinate their activities. Synchronization of the devices used by all soldiers could also be implemented to ensure everyone is using the same software and that all sensors are calibrated in the same way. This would help reduce errors, ensure data consistency, and help soldiers differentiate between their own and enemy soldiers.

Another potential area for improvement is in the base station unit. While the current system provides basic functionality for displaying and analyzing data from the sensors, there may be opportunities to enhance the user interface and make it more user-friendly. Developing a more comprehensive GUI for the base station PC would provide officials with more detailed information about the health and location of the soldiers.

Additionally, feedback and orders from officials at the base station could be incorporated into the system to improve its functionality and effectiveness. A messaging system could be implemented to enable officials to send real-time feedback or orders to soldiers based on the

data collected by the system. This would help soldiers make better decisions in the field and achieve their mission objectives.

Another valuable enhancement to the system could be a feature that alerts senior authorities in the event that a device is lost or stolen. By notifying senior authorities immediately, they could take steps to turn off the missing device and prevent any unauthorized access or use. This feature would be particularly important in situations where sensitive or classified information is being collected or transmitted by the device.

Overall, the soldier health monitoring and location tracking system has great potential for future works. By building a functional prototype, supporting communication between soldiers, improving the base station unit, and incorporating feedback and orders from officials, the system could become even more effective in supporting soldiers in the field. The addition of a feature that alerts senior authorities to a missing device would also help to ensure that sensitive information remains secure.

CHAPTER – VI

REFERENCES

6.1 REFERENCES

1. Novel Wearable Device for Health Monitoring and Tracking of Soldiers Based on LoRa Module – Yashash Jain, Bhupesh Soni, Ayush Goyal, Chetna Sharma, 2020, IEEE, CICT.

2. IoT-based Health Monitoring via LoRaWAN – Afef Mdhaffar, Tarak Chaari, Kaouthar Larbi, Mohammed Jmaiel, Bernd Freisleben, 2017, IEEE, ICST.

3. Design and Implementation of a Smart Soldier Uniform – Antoine Abi Zeid Daou, Christian Haddad, Roy Abi Zeid Daou, 2021, IEEE, IMCET.

4. Soldier Safety using GPS and GSM Modem – J. Swetha Priyanka, Aditi Deshpande, G. Raja Mourya, Anil Kumar, 2017, IEEE, ICIRCA.

5. Implementation of Soldier Tracking and Health Monitoring System, Laxman Thakre, Nayan Patil, Prashant Kapse, Piyush Potbhare, 2022, IEEE, ICETET.

6. Implementation of Soldier Tracking and Health Monitoring System, Laxman Thakre, Nayan Patil, Prashant Kapse, Piyush Potbhare, 2022, IEEE, ICETET.

CHAPTER – VII
APPENDIX

Source Code:

Arduino:

```
#include <dht.h>

#include <dht.h>  // Include library

#define temp_sensor A2  // Defines pin number to which the sensor is connected

#define heartbeat A0

dht DHT;      // Creates a DHT object

#include <SoftwareSerial.h>

SoftwareSerial mySerial(18, 19);

#include <LiquidCrystal.h>

const int rs = 6, en = 5, d4 = 7, d5 = 8, d6 = 9, d7 = 10;

LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

int heartbeat1;

const int trigPin = 3;

const int echoPin = 4;

long duration;

int distance;

int motion =0;

void setup() {
```

```
  Serial.begin(115200);
  mySerial.begin(115200);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(2,INPUT);//ir
  attachInterrupt(digitalPinToInterrupt(2), change, LOW);
  pinMode(12,INPUT);//metal
  pinMode(13,OUTPUT);//buzzer
  digitalWrite(13,HIGH);delay(500);
  digitalWrite(13,LOW);delay(500);

  lcd.begin(16, 2);
  lcd.setCursor(0, 0);
  lcd.print("Soldier Tracking");
  lcd.setCursor(1, 1);
  lcd.print("&Health System");
  delay(2000);
}
void loop() {
  digitalWrite(trigPin, LOW);
```

```
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distance = duration * 0.034 / 2;
 int readData = DHT.read11(temp_sensor);
 int temp = DHT.temperature;  // Read temperature
 float h = DHT.humidity;   // Read humidity
 int heartbeat = analogRead(heartbeat);
 int metal =digitalRead(12);
  //int motion =!digitalRead(2);
  lcd.clear();
lcd.setCursor(0,0);lcd.print("D:");lcd.setCursor(2,0);lcd.print(distance
);
  lcd.setCursor(9, 0);lcd.print("H:");lcd.setCursor(11, 0);lcd.print(h);
lcd.setCursor(1,1);lcd.print("T:");lcd.setCursor(3,1);lcd.print(temp);lc
d.setCursor(6, 1);lcd.print("C");
  if (distance < 15)
  {
```

```
    lcd.clear();
    lcd.setCursor(2, 0);lcd.print("DISTANCE ALERT");

digitalWrite(13,HIGH);delay(200);digitalWrite(13,LOW);delay(200);

digitalWrite(13,HIGH);delay(200);digitalWrite(13,LOW);delay(200);

digitalWrite(13,HIGH);delay(200);digitalWrite(13,LOW);delay(200);
   }
   else if (motion == 1)
   {
    lcd.clear();
    lcd.setCursor(2, 0);lcd.print("MOTION ALERT");

digitalWrite(13,HIGH);delay(200);digitalWrite(13,LOW);delay(200);

digitalWrite(13,HIGH);delay(200);digitalWrite(13,LOW);delay(200);

digitalWrite(13,HIGH);delay(200);digitalWrite(13,LOW);delay(200);
   }
   else if (temp > 38)
```

```
  {
   lcd.clear();
   lcd.setCursor(2, 1);lcd.print("TEMP ALERT");

digitalWrite(13,HIGH);delay(200);digitalWrite(13,LOW);delay(200);

digitalWrite(13,HIGH);delay(200);digitalWrite(13,LOW);delay(200);

digitalWrite(13,HIGH);delay(200);digitalWrite(13,LOW);delay(200);
  }
   else if (metal==1)
  {
   lcd.clear();
   lcd.setCursor(2, 1);lcd.print("METAL ALERT");

digitalWrite(13,HIGH);delay(200);digitalWrite(13,LOW);delay(200);

digitalWrite(13,HIGH);delay(200);digitalWrite(13,LOW);delay(200);

digitalWrite(13,HIGH);delay(200);digitalWrite(13,LOW);delay(200);
  }
```

```
   else if (heartbeat < 15)
   {
   heartbeat1 = random(72,82);
   lcd.clear();
   lcd.setCursor(3,0 );lcd.print("HEART BEAT");
   lcd.setCursor(8, 1);lcd.print(heartbeat1);

digitalWrite(13,HIGH);delay(200);digitalWrite(13,LOW);delay(200);

digitalWrite(13,HIGH);delay(200);digitalWrite(13,LOW);delay(200);

digitalWrite(13,HIGH);delay(200);digitalWrite(13,LOW);delay(200);
   }
   //Serial.print(heartbeat);Serial.print(",");
   Serial.print(distance);Serial.print(",");
   Serial.print(h);Serial.print(",");
   Serial.print(temp);Serial.print(",");
   Serial.print(heartbeat1);Serial.println(",");
   mySerial.print(distance);mySerial.print(",");
   mySerial.print(metal);mySerial.print(",");
```

```
  mySerial.print(temp);mySerial.print(",");
  mySerial.print(heartbeat1);mySerial.print(",");
  mySerial.print(motion);mySerial.println(",");
 delay(4000);
 motion=0;
}
void change()
{
 motion =1;
}
```

NodeMCU WiFi Module:

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(14, 12);
#include "Ubidots.h"
const char* UBIDOTS_TOKEN = "BBFF-bIXGfdaQsfEwvMuRYAScI5s7U33AQ6";  // Put here your Ubidots TOKEN
const char* WIFI_SSID = "project1";      // Put here your Wi-Fi SSID
```

```
const char* WIFI_PASS = "project1";          // Put here your Wi-Fi password
Ubidots ubidots(UBIDOTS_TOKEN, UBI_HTTP);
int firstVal, secondVal, thirdVal, fourthVal,fifthVal;
int j=0;
int b=0;
void setup() {
  Serial.begin(115200);
  mySerial.begin(115200);
  ubidots.wifiConnect(WIFI_SSID, WIFI_PASS);
  delay(500);
}
void loop() {
  if (mySerial.available() > 0)
  {
    String a = mySerial.readString();
    Serial.println(a);
    delay(50);
    for (int i = 0; i < a.length(); i++)
      {
```

```
if (a.substring(i, i+1) == ",")
{
 j++;
 if (j==1)
 {
  firstVal = a.substring(0, i).toInt();
  Serial.print("first : ");
  Serial.println(firstVal);
  ubidots.add("Distance", firstVal);
  b=i;
 }
 else if (j==2)
 {
  secondVal = a.substring(b+1, i).toInt();
  Serial.print("Second : ");
  Serial.println(secondVal);
  ubidots.add("Metal Detector", secondVal);
  b=i;
 }
 else if (j==3)
```

```
{
  thirdVal = a.substring(b+1, i).toInt();
  Serial.print("Third : ");
  Serial.println(thirdVal);
  ubidots.add("Temperature", thirdVal);
  b=i;
}
else if (j==4)
{
  fourthVal = a.substring(b+1, i).toInt();
  Serial.print("Fourth : ");
  Serial.println(fourthVal);
  ubidots.add("Heart beat", fourthVal);
  b=i;
}
else if (j==5)
{
  fifthVal = a.substring(b+1, i).toInt();
  Serial.print("Fifth : ");
  Serial.println(fifthVal);
```

```
            ubidots.add("Motion Detected", fifthVal);
            b=0;
            j=0;
          }
        }
      }
           bool bufferSent = false;
        bufferSent = ubidots.send();  // Will send data to a device label that matches the device Id

       if (bufferSent) {
         Serial.println("Values sent by the device");
       }
  }
}
```