REMOTE HEART RATE AND HEALTH MONITORING USING IOT

A PROJECT REPORT

Submitted by

F.ASHFAQUE AHAMED(311819104006)

K.MOHAMED MUSHRAF (311819104025)

B.NITHISH (311819104032)

in partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

MOHAMED SATHAK A. J. COLLEGE OF ENGINEERING,

SIRUSERI CHENNAI – 603 103

ANNA UNIVERSITY: CHENNAI 600 025

MAY 2023

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"REMOTE HEART RATE AND HEALTH MONITORING USING IOT" is** the bonafide work of **F.ASHFAQUE AHAMED (311819104006), K.MOHAMED MUSHRAF (311819104025), B.NITHISH (311819104032)** who carried out the project work under my supervision.

Mr.S.VIMALATHITHAN

HEAD OF THE DEPARTMENT

Department of Computer Science and Engineering
Mohammed Sathak A J College of Engineering.,
Siruseri,Chennai-603103

Mrs Kanmani M

SUPERVISOR

Assistant Professor
Department of Computer Science an Engineering
Mohammed Sathak A J College of Engineering.,
Siruseri,Chennai-603103

Project Viva-Voice held on _____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

First and foremost, we thank the almighty for helping us in all situations for bringing out this project successfully. We thank **OUR PARENTS** for their support throughout the project and also the course. We express our sincere heartfelt gratitude to **ALHAJ JANAB S.M. YOUSUF,** Chairman, and **Mr. MOHAMED SATHAK**, Director, Mohammed Sathak A J College of Engineering, Chennai. We record our immense pleasure in expressing sincere gratitude to our Principal **Dr. K. S. SRINIVASAN,** Mohammed Sathak A J College of Engineering, for granting permission to undertake the project in our college. We express our sincere thanks to our **Head of the Department Mr. S VIMALATHITHAN** and our **Project Coordinator Mrs Kanmani M ,** Assistant Professor, Department of Computer Science and Engineering, Mohammed Sathak A J College of Engineering and Supervisor **_____,** Assistant Professor, Mohammed Sathak A J College of Engineering for their constant encouragement and Guidance for this project. We wish to express our thanks to all the **Faculty Members and staff** in the Department of Computer Science and Engineering for their valuable support throughout the course. We also thank our Friends for their support the project.

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

This project focuses on the development of a remote heart rate and temperature monitoring system using IoT (Internet of Things) technology. The system allows individuals to conveniently and remotely monitor their vital signs, providing real-time feedback and facilitating proactive healthcare management. Wearable devices equipped with sensors are used to capture heart rate and temperature data, which is then transmitted wirelessly to a centralized hub. The hub, connected to the internet, relays the data to a cloud-based platform for further analysis and visualization. The integration of IoT technology enables users and healthcare professionals to access and monitor the data from anywhere, promoting self-awareness and facilitating timely interventions. This system has the potential to enhance healthcare management, improve patient outcomes, and empower individuals to take an active role in maintaining their health.

Chapter 1

1.1 Introduction

The project focuses on the development of a remote heart rate and temperature monitoring system using IoT (Internet of Things) technology. This innovative system enables individuals to monitor their vital signs conveniently and remotely, providing real-time feedback and facilitating proactive healthcare management. By leveraging IoT, data collected from wearable sensors can be transmitted wirelessly to a centralised platform, allowing users and healthcare professionals to monitor and analyse the data from anywhere.

1.1.2 Heart Rate Monitoring: The heart rate monitoring aspect of the project involves the use of wearable devices equipped with sensors that can measure the user's heart rate accurately. These devices may include smartwatches, chest straps, or finger-mounted sensors. The sensors capture the heart rate data and transmit it to a central hub via wireless communication protocols such as Bluetooth or Wi-Fi. The hub, typically connected to the internet, relays the data to a cloud-based platform for further processing and analysis.

1.1.3 Temperature Monitoring: The temperature monitoring component focuses on measuring and tracking the user's body temperature. This can be achieved using wearable temperature sensors, such as skin patches or smart thermometers, which continuously monitor the user's body temperature. Similar to the heart rate monitoring, the temperature data is transmitted wirelessly to the central hub and then to the cloud-based platform for storage and analysis.

1.1.4 IoT Integration: The integration of IoT technology is crucial in this project. The central hub acts as a bridge between the wearable devices and the cloud platform. It collects the data from the sensors, processes it, and establishes a connection with the internet to transmit the information securely. The cloud-based platform receives the data and provides a user interface through which users can access and visualize their heart rate and temperature readings. Additionally, the platform can generate alerts and notifications based on predefined thresholds to alert users or healthcare providers of any anomalies or critical conditions.

Rural hospitals face limited healthcare facilities, leading to issues in health management and a lack of accessibility. The shortage of doctors in India, with a doctor-patient ratio of 1:1000, further exacerbates the situation. To address these challenges, affordable and user-friendly health monitoring systems have been developed to provide easy and assured care, reducing the need for expensive daily check-ups while ensuring safe handling of equipment.

1.2PURPOSE

The purpose of the remote heart rate and temperature monitor using IoT is to provide individuals with a convenient and effective means of monitoring their vital signs remotely. The system enables individuals to monitor their heart rate and temperature in real time, allowing them to stay informed about their health status continuously. By providing individuals with access to their heart rate and temperature data, the system promotes proactive healthcare management. The system also serves the purpose of remote patient monitoring, enabling healthcare professionals to remotely monitor their patients' heart rate and temperature readings. By leveraging IoT technology, the system offers convenience and flexibility in monitoring vital signs.

1.3PROBLEM STATEMENT

The existing methods of health monitoring lack real-time tracking and timely intervention, posing risks to individuals' well-being. This project aims to develop a Remote Heart Rate and Health Monitoring System using IoT to address this challenge. The system will track heart rate and body temperature in real time, transmit data to a web-based platform, and send alerts via a GSM module when the body temperature exceeds a predefined limit. By leveraging IoT technology, this project seeks to enable continuous monitoring, timely interventions, and improved healthcare accessibility for individuals.

2 ANALYSIS AND DESIGN

2.1 EXISTING SYSTEM

In traditional method, doctors play an important role in health check up. For this process requires a lot of time for registration, appointment and then check up. Also reports are generated later. Due to this lengthy process working people tend to ignore the check ups or postpone it. This modern approach reduces time consumption in the process.

DISADVANTAGE:

This process requires a lot of time for registration, appointment and then check up. Also reports are generated later. Due to this lengthy process working people tend to ignore the check-ups or postpone it.

2.2 PROPOSED SYSTEM

Health monitoring sensors collect data, which is wirelessly communicated to a server for processing and aggregation. IoT devices help store patient details and reports, display understandable health information on the web, and send medicine alerts. The proposed patient monitoring system, utilizing Arduino Uno, monitors health parameters and allows easy access to stored data on personal computers.

ADVANTAGE

1)Doctors to monitor patients remotely without risk of infection

2)Sensing and Data collection

3)Tracking an patient

3 MODULE DESCRIPTION

3.1 SOFTWARE REQUIREMENT SPECIFICATION

The Software Requirements Specification is produced at the culmination of the analysis task. The function and performance allocated to software as part of system engineering are refined by establishing a complete information description, a detailed functional and behavioural description, an indication of performance requirements and design constraints, appropriate validation criteria, and other data pertinent to requirements.

- Operating System: 64-bit Windows 10
- CPU: Intel i7 –6500U / AMD A8-6600k or Better
- HDD: 4GB (SSD Recommended) available Space
- Memory: 4GB RAM
- Connectivity: Internet Connection of at least 2mbps

HARDWARE REQUIREMENTS:

- Arduino UNO
- NODEMCU
- GSM
- Heart Beat Sensor
- LCD Display
- Buzzer
- Power Supply

SOFTWARE REQUIREMENTS:

- ARDUINO IDE

3.2 SEQUENCE DIAGRAM

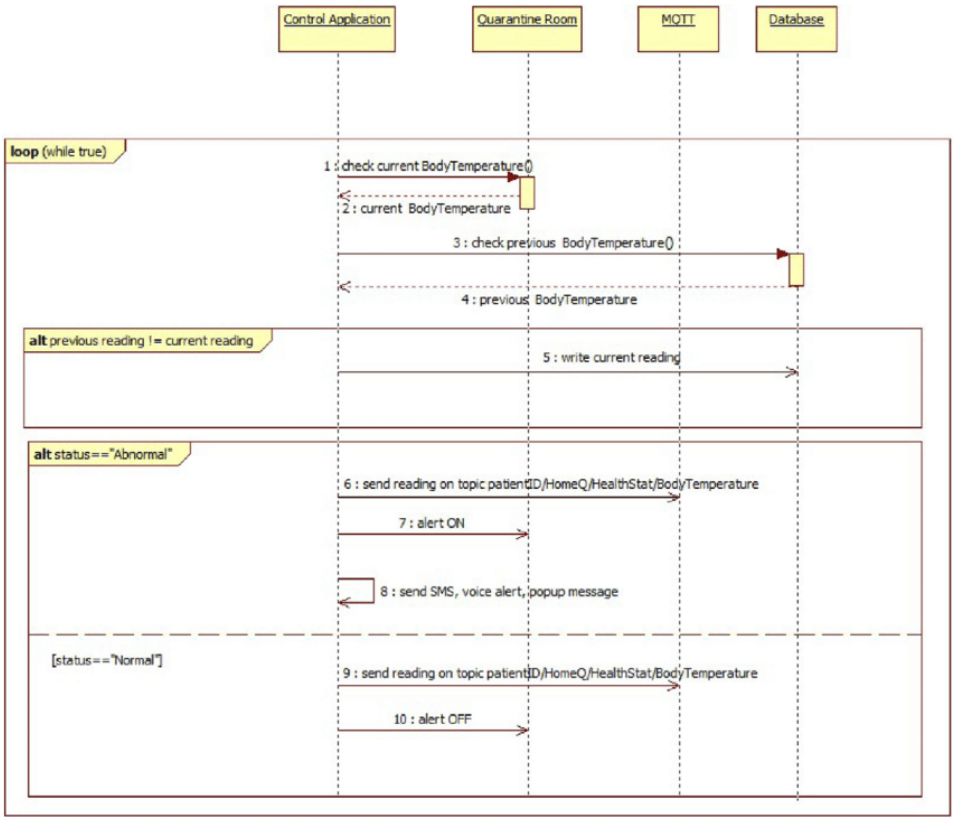


Fig 3.2 Sequence Diagram

3.3 DATA FLOW DIAGRAM

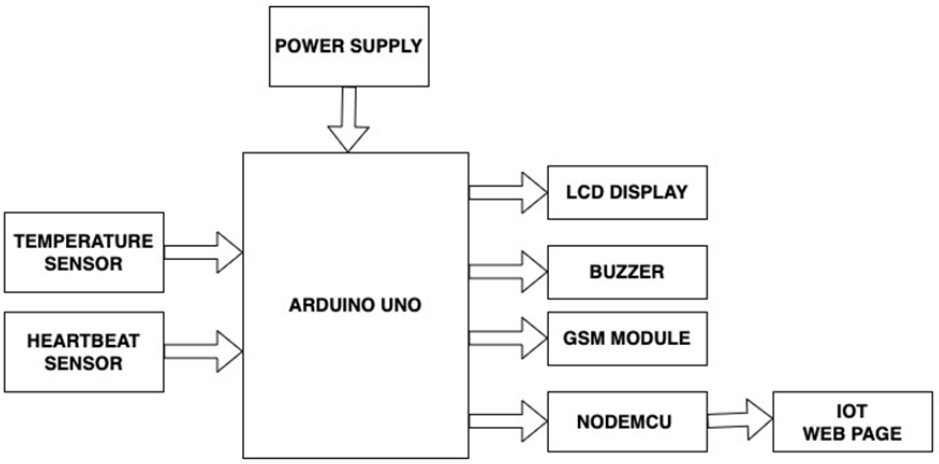


Fig 3.3 DataFlow Diagram

3.4 WORKING PRINCIPLE

MODULE 1:

The patient's data is collected via a temperature sensor and a heartbeat sensor, then sent to an Arduino microcontroller and values will be show on LCD Display.

MODULE 2:

When the sensor values exceed the threshold limit, the buzzer will give sound alert and GSM module send the SMS notification to authorized person.

MODULE 3:

Every bit of information is continuously uploaded to the IOT webpage.

4 HARDWARE

4.1 ARDUINO UNO:

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.

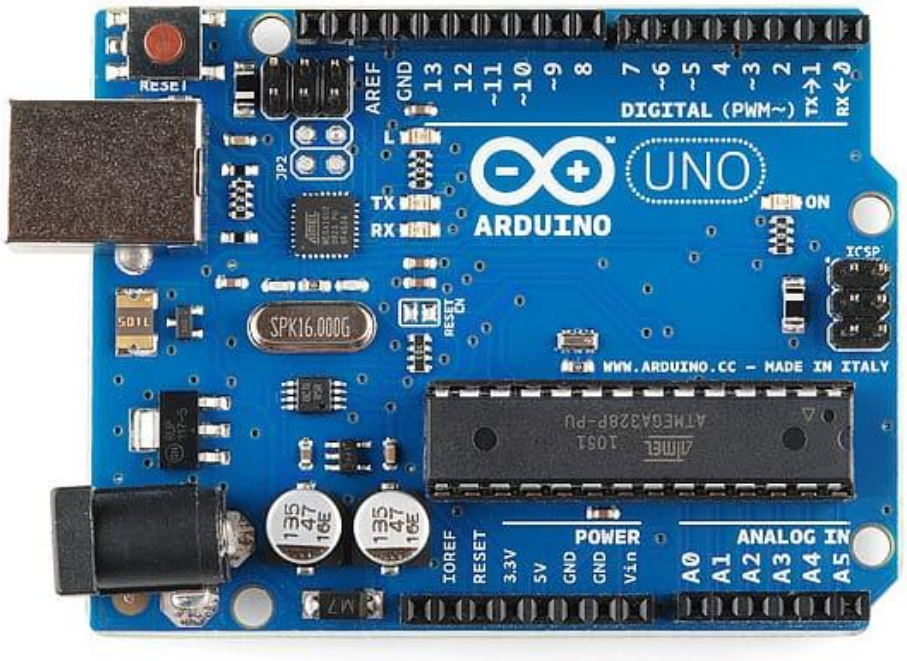


Fig 4.1 Arduino Board

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 (like this) that is terminated in a barrel jack. In the picture above the USB connection is labelled (1) and the barrel jack is labelled (2). The USB connection is also how you will load code onto your Arduino board. More on how to program with Arduino can be found in our Installing and Programming Arduino tutorial. NOTE: Do NOT use a power supply greater than 20 Volts as you will overpower (and thereby destroy) your Arduino. The recommended voltage for most Arduino models is between 6 and 12 Volts.

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 conjunction 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 labelled on the board and used for different functions.

GND (3): Short for 'Ground'. There are several GND pins on the Arduino, any of which can be used to ground your circuit.

5V (4) & 3.3V (5): 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 (6): 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 (7): 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 (8): 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). We have a tutorial on PWM, but for now, think of these pins as being able to simulate analog output (like fading an LED in and out).

AREF (9): 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 (10). 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' (11). 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 (12). 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 (13). 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 AT mega 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 (14) 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

ATMEGA328 Microcontroller

The Atmel AVR® core combines a rich instruction set with 32 general purpose working registers.

All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in a single instruction executed in one clock cycle.

The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

The ATmega328/P provides the following features:

32Kbytes of In-System Programmable Flash with Read-While-Write capabilities, 1Kbytes EEPROM, 2Kbytes SRAM, 23 general purpose I/O lines, 32 general

purpose working registers, Real Time Counter (RTC), three flexible Timer/Counters with compare modes and PWM, 1 serial programmable USARTs , 1 byte-oriented 2-wire Serial Interface (I2C), a 6- channel 10-bit ADC (8 channels in TQFP and QFN/MLF packages) , a programmable Watchdog Timer with internal Oscillator, anSPI serial port, and six software selectable power saving modes.

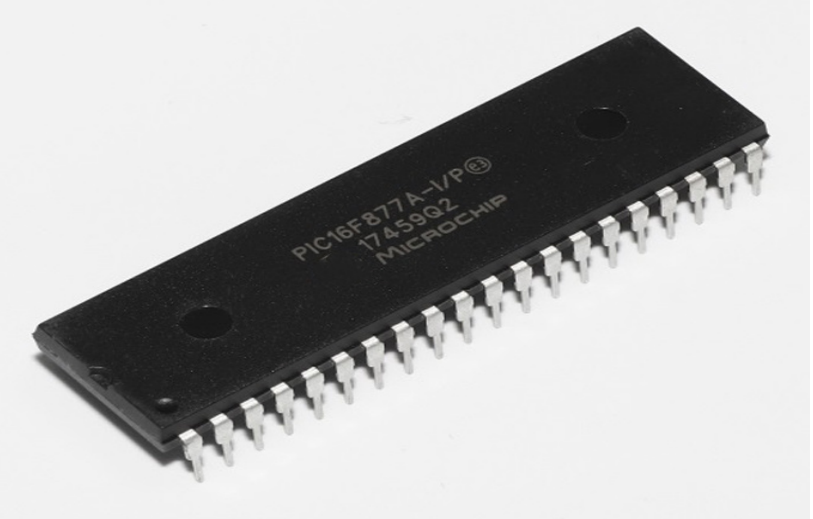


Fig 4.2 Arduino Pinout

The Boot program can use any interface to download the application program in the Application Flash memory.

Software in the Boot Flash section will continue to run while the Application Flash section is updated, providing true Read-While-Write operation.

By combining an 8-bit RISC CPU with In-System Self-Programmable Flash on a monolithic chip, the Atmel ATmega328/P is a powerful microcontroller that provides a highly flexible and cost-effective solution to many embedded control application

4.2 NODEMCU

When prototyping new IoT equipment, development boards such as Arduino and Raspberry Pi are popular options.

These development boards are fundamentally mini-computers that a normal PC or Mac can connect to and program.

- The design boards can then connect and regulate sensors in the field after it has been programmed.
- The design boards need a way to connect to the internet because the "I" in IoT stands for internet.
- In the field, using wireless networks is the best way to connect to the Internet. However, wireless networks are not supported by Arduino and Raspberry Pi. In order to access the wireless module, developers will need to add a wifi or cellular module to the board and write code.
- I will present an open source IoT development board called Nodemcu in this paper. One of its most distinctive characteristics is that it has built-in wifi connectivity assistance, making the development of IoT applications much easier

What is NodeMCU?

- The NodeMCU is an open-source development environment built around the ESP8266 chip, a low-cost and feature-rich system-on-a-chip (SoC) for IoT projects. It simplifies ESP8266 development by providing an open-source firmware and a DEVKIT board. The firmware, based on eLua scripting language, offers an easy programming environment, while the DEVKIT board provides a convenient circuit board with the ESP8266 chip. The NodeMCU is a valuable component in our project as it enables easier access and utilization of the ESP8266 chip for real-time health monitoring and data transmission in the IoT system.

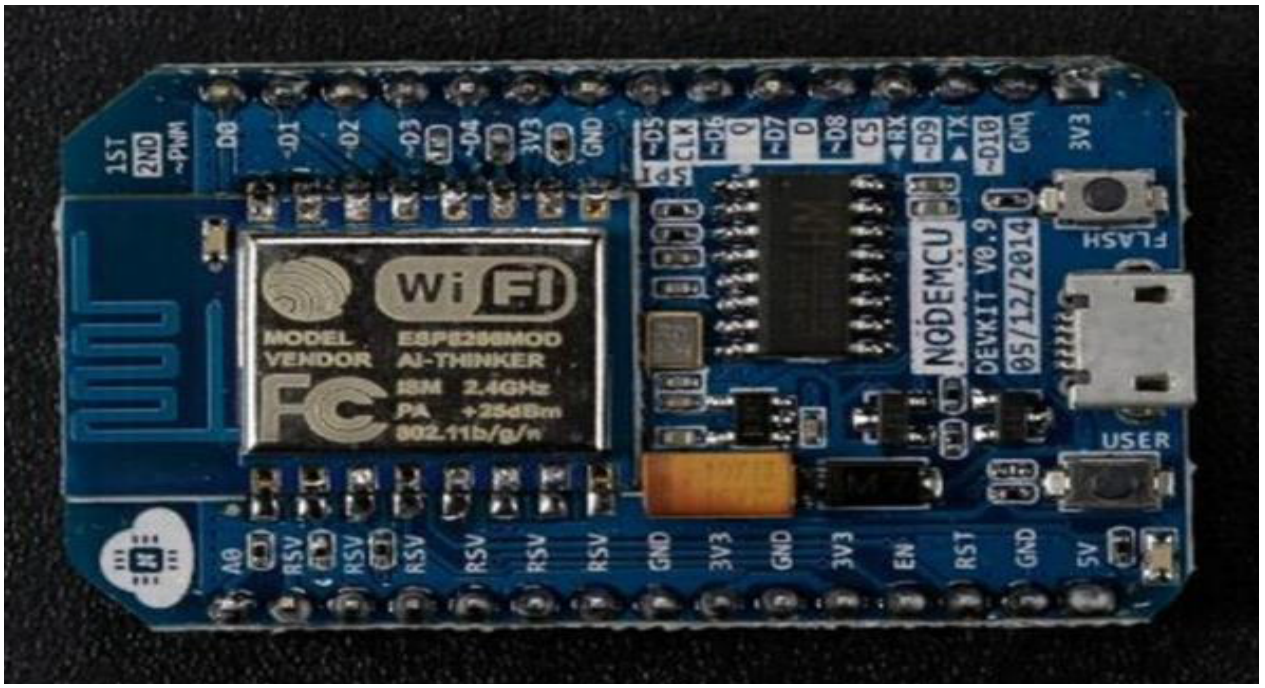


Fig 4.3 Node MCU

- For a flexible IoT controller, the Arduino project produces an open source hardware architecture and SDK software.
- The Arduino hardware, similar to Nodemcu, is a microcontroller board with a ready-to-use USB connector, LED lights, and normal information pins. It also describes normal interfaces for sensors or other boards to communicate with. But unlike Nodemcu, with memory chips and a range of programming settings, the Arduino board can have distinct kinds of CPU chips (usually an ARM or Intel x86 chip).
- In reality, the ESP8266 chip also has an Arduino reference design. Arduino flexibility also implies important differences across various suppliers, however. For instance, most boards in Arduino do

4.2.1 The ESP8266 as a microcontroller - Hardware

While the ESP8266 is often used as a 'dumb' Serial-to-Wifi bridge, it's a very powerful microcontroller on its own. In this chapter, we'll look at the non-Wi-Fi specific functions of the ESP8266.

4.2.2 Digital I/O

Just like a normal Arduino, the ESP8266 has digital input/output pins (I/O or GPIO, General Purpose Input/output pins).

As the name implies, they can be used as digital inputs to read a digital voltage, or as digital outputs to output either 0V (sink current) or 3.3V (source current).

4.2.3 Voltage and current restrictions

The ESP8266 is a 3.3V microcontroller, so its I/O operates at 3.3V as well. The pins are not 5V tolerant applying more than 3.6V on any pin will kill the chip.

The maximum current that can be drawn from a single GPIO pin is 12mA.

Usable pins

The ESP8266 has 17 GPIO pins (0-16), however, you can only use 11 of them, because 6 pins (GPIO 6 - 11) are used to connect the flash memory chip. This is the small 8-legged chip right next to the ESP8266. If you try to use one of these pins, you might crash your program.

GPIO 1 and 3 are used as TX and RX of the hardware Serial port (UART), so in most cases, you can't use them as normal I/O while sending/receiving serial data.

4.2.4 Boot modes

As mentioned in the previous chapter, some I/O pins have a special function during boot: They select 1 of 3 boot modes:

Table :

GPIO15	GPIO0	GPIO1	MODE
0V	0V	3.3V	UartBootloader
0V	3.3V	3.3V	Boot sketch(SPI flash)

GPIO15	GPIO0	GPIO2	Mode
3.3V	x	x	SDIO mode (not used for Arduino)

Note: you don't have to add an external pull-up resistor to GPIO2, the internal one is enabled at boot.

- We made sure that these conditions are met by adding external resistors in the previous chapter, or the board manufacturer of your board added them for you. This has some implications, however:
- GPIO15 is always pulled low, so you can't use the internal pull-up resistor. You have to keep this in mind when using GPIO15 as an input to read a switch or connect it to a device with an open-collector (or open-drain) output, like I²C.
- GPIO0 is pulled high during normal operation, so you can't use it as a Hi-Z input.
- GPIO2 can't be low at boot, so you can't connect a switch to it.

Internal pull-up/-down resistors

GPIO 0-15 all have a built-in pull-up resistor, just like in an Arduino. GPIO16 has a built-in pull-down resistor.

PWM

Unlike most Atmel chips (Arduino), the ESP8266 doesn't support hardware PWM, however, software PWM is supported on all digital pins. The default PWM range is 10-bits @ 1kHz, but this can be changed (up to >14-bit@1kHz).

Analog input

The ESP8266 has a single analog input, with an input range of 0 - 1.0V. If you supply 3.3V, for example, you will damage the chip. Some boards like the NodeMCU have an on-board resistive voltage divider, to get an easier 0 - 3.3V range.

You could also just use a trimpot as a voltage divider.

The ADC (analog to digital converter) has a resolution of 10 bits.

Communication

Serial

The ESP8266 has two hardware UARTS (Serial ports): UART0 on pins 1 and 3 (TX0 and RX0 resp.), and UART1 on pins 2 and 8 (TX1 and RX1 resp.), however, GPIO8 is used to connect the flash chip. This means that UART1 can only transmit data.

UART0 also has hardware flow control on pins 15 and 13 (RTS0 and CTS0 resp.). These two pins can also be used as alternative TX0 and RX0 pins.

I²C

The ESP doesn't have a hardware TWI (Two Wire Interface), but it is implemented in software. This means that you can use pretty much any two digital pins.

By default, the I²C library uses pin 4 as SDA and pin 5 as SCL. (The data sheet specifies GPIO2 as SDA and GPIO14 as SCL.) The maximum speed is approximately 450kHz.

SPI

The ESP8266 has one SPI connection available to the user, referred to as HSPI. It uses GPIO14 as CLK, 12 as MISO, 13 as MOSI and 15 as Slave Select (SS). It can be used in both Slave and Master mode (in software).

Table : GPIO overview

GPIO	FUNCTION	STATE	RESTRICTION
0	Boot mode select	3.3V	No Hi-Z
1	TX0	-	Not usable using Serial transmission
2	Boot mode select TX1	3.3V (boot only)	Don't connect to ground at boot time Sends debug data at boot time
3	RX0	-	Not usable during Serial transmission
4	SDA (I²C)	-	-
5	SCL (I²C)	-	-
6-11	Flash connection	x	Not usable, and not broken out
12	MISO (SPI)	-	-

13	MOSI (SPI)	-	-
14	SCK (SPI)	-	-
15	SS (SPI)	0V	Pull-up resistor not usable
16	Wake up from sleep	-	No pull-up resistor, but pull-down instead Should be connected to RST to wake up

The ESP8266 as a microcontroller - Software

Most of the microcontroller functionality of the ESP uses exactly the same syntax as a normal Arduino, making it really easy to get started.

Digital I/O

- Just like with a regular Arduino, you can set the function of a pin using pinMode (pin, mode); where pin is the GPIO number*, and mode can be either INPUT, which is the default, OUTPUT, or INPUT_PULLUP to enable the built-in pull-up resistors for GPIO 0-15. To enable the pull-down resistor for GPIO16, you have to use INPUT_PULLDOWN_16.
- To address a NodeMCU pin, e.g. pin 5, use D5: for instance: pinMode(D5, OUTPUT);
- To set an output pin high (3.3V) or low (0V), use digitalWrite (pin, value); where pin is the digital pin, and value either 1 or 0 (or HIGH and LOW).
- To read an input, use digitalRead(pin);
- To enable PWM on a certain pin, use analogWrite (pin, value); where pin is the digital pin, and value a number between 0 and 1023.
- You can change the range (bit depth) of the PWM output by using analogWriteRange(new_range);

- The frequency can be changed by using analogWriteFreq (new frequency).New frequency should be between 100 and 1000Hz.

4.2.5 Analog input

- Just like on an Arduino, you can use analogRead (A0) to get the analog voltage on the analog input. (0 = 0V, 1023 = 1.0V).
- The ESP can also use the ADC to measure the supply voltage (VCC). To do this, include ADC_MODE (ADC_VCC); at the top of your sketch, and use ESP.getVcc () to actually get the voltage. If you use it to read the supply voltage, you can't connect anything else to the analog pin.

4.2.6 Communication

Serial communication

- To use UART0 (TX = GPIO1, RX = GPIO3), you can use the Serial object, just like on an Arduino: Serial. Begin(baud).
- To use the alternative pins (TX = GPIO15, RX = GPIO13), use Serial. Swap() after Serial. Begin.
- To use UART1 (TX = GPIO2), use the Serial1 object.

All Arduino Stream functions, like read, write, print, println, ... are supported as well.

I²C and SPI

You can just use the default Arduino library syntax, like you normally would.

4.2.7 Sharing CPU time with the RF part

- One thing to keep in mind while writing programs for the ESP8266 is that your sketch has to share resources (CPU time and memory) with the Wi-Fi-

and TCP-stacks (the software that runs in the background and handles all Wi-Fi and IP connections). If your code takes too long to execute, and don't let the TCP stacks do their thing, it might crash, or you could lose data. It's best to keep the execution time of you loop under a couple of hundreds of milliseconds.

- Every time the main loop is repeated, your sketch yields to the Wi-Fi and TCP to handle all Wi-Fi and TCP requests.
- If your loop takes longer than this, you will have to explicitly give CPU time to the Wi-Fi/TCP stacks, by using including delay(0); or yield();. If you don't, network communication won't work as expected, and if it's longer than 3 seconds, the soft WDT (Watch Dog Timer) will reset the ESP. If the soft WDT is disabled, after a little over 8 seconds, the hardware WDT will reset the chip.
- From a microcontroller's perspective however, 3 seconds is a very long time (240 million clockcycles), so unless you do some extremely heavy number crunching, or sending extremely long strings over Serial, you won't be affected by this. Just keep in mind that you add the yield(); inside your for or while loops that could take longer than, say 100ms.

4.3 Liquid Crystal Display

LCD stands for Liquid Crystal Display. LCD is finding wide spread use replacing LEDs (seven segment LEDs or other multi segment LEDs) because of the following reasons:

- The declining prices of LCDs.
- The ability to display numbers, characters and graphics. This is in contrast to LEDs, which are limited to numbers and a few characters.

- Incorporation of a refreshing controller into the LCD, thereby relieving the CPU of the task of refreshing the LCD. In contrast, the LED must be refreshed by the CPU to keep displaying the data.
- Ease of programming for characters and graphics.

These components are “specialized” for being used with the microcontrollers, which means that they cannot be activated by standard IC circuits. They are used for writing different messages on a miniature LCD.

Fig 4.4 LCD

- A model described here is for its low price and great possibilities most frequently used in practice.
- It is based on the HD44780 microcontroller (Hitachi) and can display messages in two lines with 16 characters each. It displays all the alphabets, Greek letters, punctuation marks, mathematical symbols etc. In addition, it is possible to display symbols that user makes up on its own.
- Automatic shifting message on display (shift left and right), appearance of the pointer, backlight etc. are considered as useful characteristics.
- Pins Functions
- There are pins along one side of the small printed board used for connection to the microcontroller.
- There are total of 14 pins marked with numbers (16 in case the background light is built in).

4.3.1 LCD screen:

- LCD screen consists of two lines with 16 characters each. Each character consists of 5x7 dot matrix. Contrast on display depends on the power supply voltage and whether messages are displayed in one or two lines.
- For that reason, variable voltage 0-Vdd is applied on pin marked as Vee. Trimmer potentiometer is usually used for that purpose. Some versions of displays have built in backlight (blue or green diodes). When used during operating, a resistor for current limitation should be used (like with any LE diode).

4.3.2 LCD Basic Commands:

All data transferred to LCD through outputs D0-D7 will be interpreted as commands or as data, which depends on logic state on pin RS:

- RS = 1 - Bits D0 - D7 are addresses of characters that should be displayed. Built in processor addresses built in “map of characters” and displays corresponding symbols.
- Displaying position is determined by DDRAM address. This address is either previously defined or the address of previously transferred character is automatically incremented.

RS = 0 - Bits D0 - D7 are commands which determine display mode. List of commands which LCD recognizes are given in the table below:

LCD COMMANDS:

No	HEX Value	COMMAND TO LCD
1	0x01	Clear Display Screen
2	0x30	Function Set: 8-bit, 1 Line, 5x7 Dots
3	0x38	Function Set: 8-bit, 2 Line, 5x7 Dots
4	0x20	Function Set: 4-bit, 1 Line, 5x7 Dots
5	0x28	Function Set: 4-bit, 2 Line, 5x7 Dots
6	0x06	Entry Mode
7	0x08	Display off, Cursor off
8	0x0E	Display on, Cursor on
9	0x0C	Display on, Cursor off
10	0x0F	Display on, Cursor blinking
11	0x18	Shift entire display left
12	0x1C	Shift entire display right
13	0x10	Move cursor left by one character
14	0x14	Move cursor right by one character
15	0x80	Force cursor to beginning of 1st row
16	0xC0	Force cursor to beginning of 2nd row

Fig 4.5 LCD Commands

Advantages of an LCD's:

- LCD's consumes less amount of power compared to CRT and LED
- LCD's are consist of some microwatts for display in comparison to some mill watts for LED's
- LCDs are of low cost Provides excellent contrast
- LCD's are thinner and lighter when compared to cathode ray tube and LED

Disadvantages of an LCD's:

- Require additional light sources
- Range of temperature is limited for operation
- Low reliability
- Speed is very low
- LCD's need an AC drive

4.4POWER SUPPLY CIRCUIT:

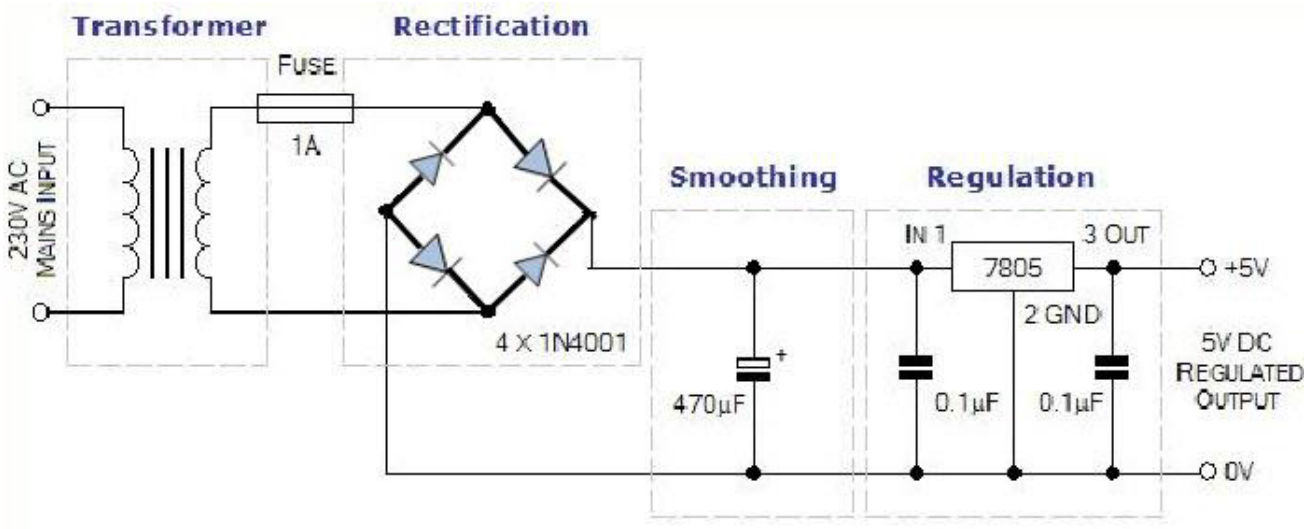


Fig 4.6 POWER SUPPLY CIRCUIT

A bridge rectifier circuit is a common part of the electronic power supplies. Many electronic circuits require rectified DC power supply for powering the various electronic basic components from available AC mains supply. We can find this rectifier in a wide variety of electronic AC power devices like home appliances, motor controllers, modulation process, welding applications, etc.

What is a Bridge Rectifier?

A Bridge rectifier is an Alternating Current (AC) to Direct Current (DC) converter that rectifies mains AC input to DC output. Bridge Rectifiers are widely used in power supplies that provide necessary DC voltage for the electronic components or devices. They can be constructed with four or more diodes or any other controlled solid state switches. Depending on the load current requirements, a proper bridge rectifier is selected. Components' ratings and specifications, breakdown voltage, temperature ranges, transient current rating, forward current rating, mounting requirements and other considerations are taken into account while selecting a rectifier power supply for an appropriate electronic circuit's application.

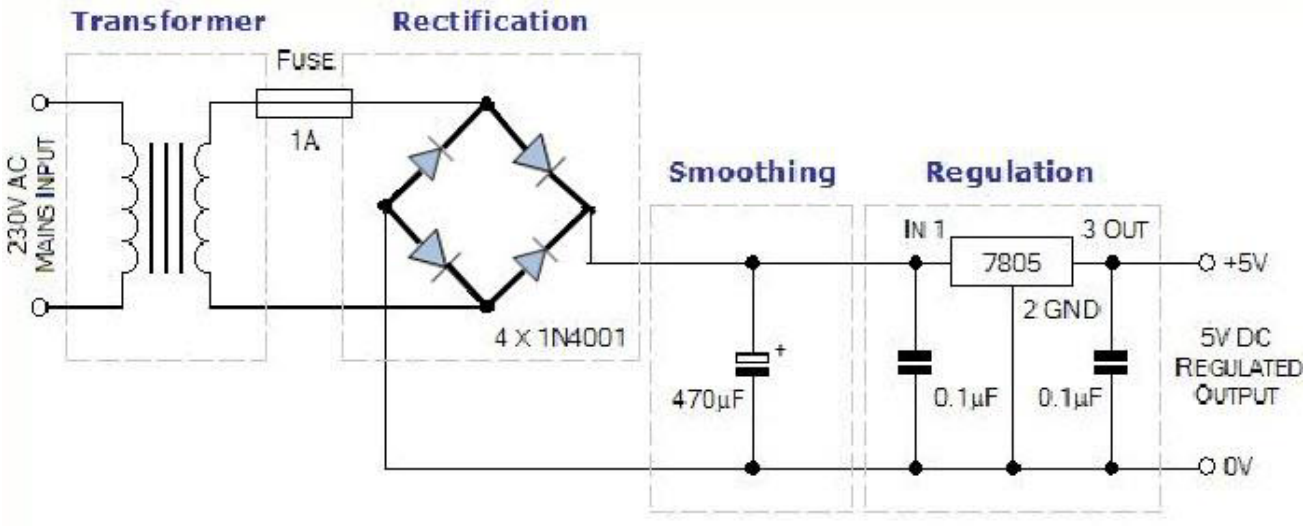
4.4.1 Bridge Rectifier Circuit Diagram

The main advantage of bridge rectifier is that it produces almost double the output voltage as with the case of a full wave rectifier using center-tapped

transformer. But this circuit doesn't need center tapped transformer so it resembles low-cost rectifier.

The bridge rectifier circuit diagram consists of various stages of devices like transformer, Diode Bridge, filtering and regulators. Generally all these blocks combination is called as regulated DC power supplythat powers various electronic appliances.

The first stage of the circuit is a transformer which is a step-down type that changes the amplitude of the input voltage. Most of the electronic projects uses 230/12V transformer to step-down the AC mains 230V to 12V AC supply.



Next stage is a diode-bridge rectifier which uses four or more diodes depending on the type of bridge rectifier. Choosing a particular diode or any other switching device for a corresponding rectifier needs some considerations of the device like

Peak Inverse Voltage (PIV), forward current If, voltage ratings, etc. It is responsible for producing unidirectional or DC current at the load by conducting a set of diodes for every half cycle of the input signal.

Since the output after the diode bridge rectifiers is of pulsating nature, and for producing it as a pure DC, filtering is necessary. Filtering is normally performed

with one or more capacitors attached acrossthe load, as you can observe in the below figure wherein smoothing of wave is performed. This capacitor rating also depends on the output voltage.

The last stage of this regulated DC supply is a voltage regulator that maintains the output voltage to a constant level. Suppose the microcontroller works at 5V DC, but the output after the bridge rectifier is around 16V, so to reduce this voltage, and to maintain a constant level – no matter voltage changes in input side – a voltage regulator is necessary.

4.4.2 Bridge Rectifier Operation

As we discussed above, a single-phase bridge rectifier consists of four diodes and this configuration is connected across the load. For understanding the bridge rectifier's working principle, we have to consider the below circuit for demonstration purpose.

During the Positive half cycle of the input AC waveform diodes D1 and D2 are forward biased and D3 and D4 are reverse biased. When the voltage, more than the threshold level of the diodesD1 and D2, starts conducting – the load current starts flowing through it, as shown as red lines path in the diagram below. During the negative half cycle of the input AC waveform, the diodes D3 and D4 are forward biased, and D1 and D2 are reverse biased. Load current starts flowing through the D3 and D4 diodes when these diodes starts conducting as shown in the figure.We

can observe that in both the cases, the load current direction is same, i.e., up to down as shown in the figure – so unidirectional,

which means DC current. Thus, by the usage of a bridge rectifier, the input AC current is converted into a DC current. The output at the load with this bridge wave rectifier is pulsating in nature, but for producing a pure DC requires additional filter like capacitor. The same operation is applicable for different bridge rectifiers, but in case of controlled rectifiers thyristors triggeringis necessary to drive the current to load.

4.4.3 VOLTAGE REGULATOR:

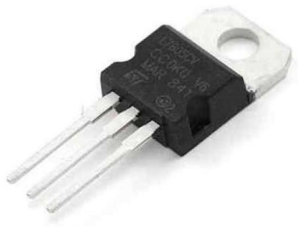


Fig 4.7 VOLTAGE REGULATOR

The name says it all: voltage regulator. The battery in your car that gets charged from the alternator, the outlet in your home that provides all the electricity you desire, the cell phone you likely keep on-hand every minute of the day they all require a specific voltage in order to function. Fluctuating outputs that jump from ±2V can cause inefficient operation and possibly even damage to your charging devices. There's a variety of reasons why a voltage fluctuation may occur: condition of the power grid, other appliances turning off and on, time of day, environmental factors, etc. Due to the need for a steady, constant voltage, enter the voltage regulator.

A voltage regulator is an integrated circuit (IC) that provides a constant fixed output voltage regardless of a change in the load or input voltage. It can do this many ways depending on the topology of the circuit within, but for the purpose of keeping this

project basic, we will mainly focus on the linear regulator. A linear voltage regulator works by automatically adjusting the resistance via a feedback loop, accounting for changes in both load and input, all while keeping the output voltage constant.

FILTER

In power supplies, capacitors are used to smooth (filter) the pulsating DC output after rectification so that a nearly constant DC voltage is supplied to the load. The pulsating output of the rectifiers has an average DC value and an AC portion that is called ripple voltage. Filter capacitors reduce the amount of ripple voltage to a level that is acceptable. It should be noted that resistors and inductors can be combined with the capacitors to form filter networks. Here we will concentrate on capacitive filters only. In a filter circuit the capacitor is charged to the peak of the rectified input voltage during the positive portion of the input. When the input goes negative, the capacitor begins to discharge into the load. The rate of discharge is determined by the RC time constant formed by the capacitor and the load's resistance.Filters is used to removes ripples and noise.

4.5 Pulse Sensor

The Pulse Sensor is a well-designed low-power plug-and-play heart-rate sensor for the Arduino. It can be used by students, artists, athletes, manufacturers, and game & mobile developers who want to incorporate live heart-rate data into their projects. And the best part is that this sensor plugs right into Arduino and easily clips onto a fingertip or earlobe. It is also super small (button-shaped) with holes, so it can be sewn into fabric. Hardware Overview The front of the sensor is the side with the heart logo. This is where you place your finger. On the front side you will see a small round hole, from where the Kingbright’s reverse mounted green LED shines.

Just below the LED is a small ambient light photo sensor – APDS-9008 from Avago, similar to that used in cell phones, tablets and laptops, to adjust the screen brightness in different light conditions. On the back of the module you will find the rest of the components including a microchip's MCP6001 Op-Amp and a bunch of resistors and capacitors that make up the R/C filter network. There is also a reverse protection diode to prevent damage if the power leads are accidentally reversed.

How Pulse Sensor Works?

Optical heart-rate sensors are very easy to understand in theory. If you've ever shined a flashlight through your finger tips and seen your heartbeat pulse, you have a good handle on the theory of optical heart-rate pulse sensors.

A pulse sensor or any optical heart-rate sensor, for that matter, works by shining a

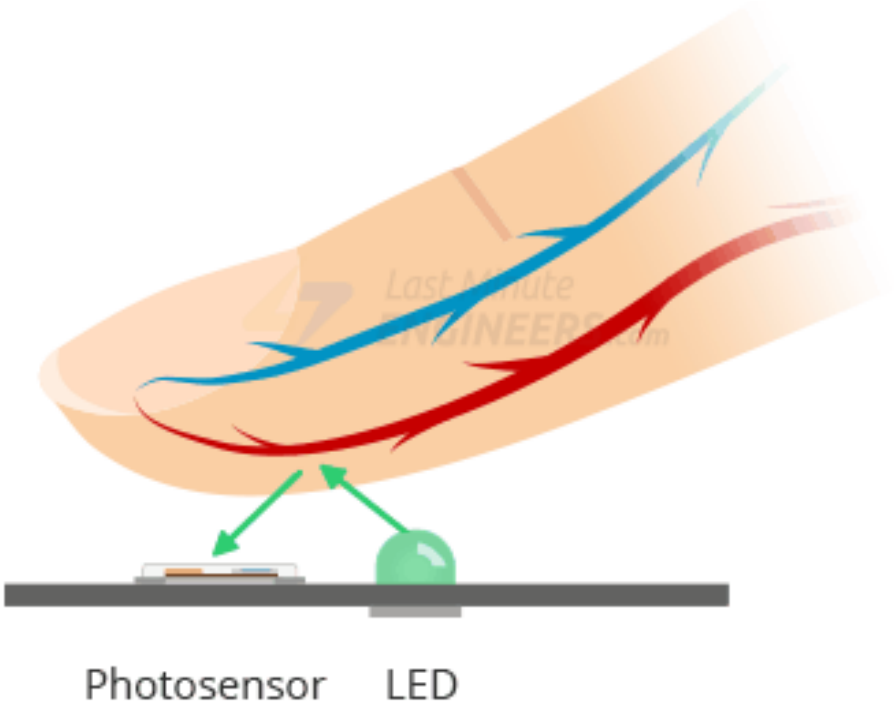


Fig 4.8 PULSE SENSOR

green light (~ 550nm) on the finger and measuring the amount of reflected light using a photosensor.

This method of pulse detection through light is called Photoplethysmogram.

Wiring Pulse Sensor with Arduino

Hooking up the Pulse Sensor to an Arduino is super simple. You only need to connect three wires: two for power and one for reading the sensor value.

The module can be powered from 3.3 or 5V. The positive voltage connects to '+' and ground connects to '-'. The 3rd 'S' wire is the analog signal output from the sensor and this will connect to the A0 analog input of an Ar

4.6 GSM

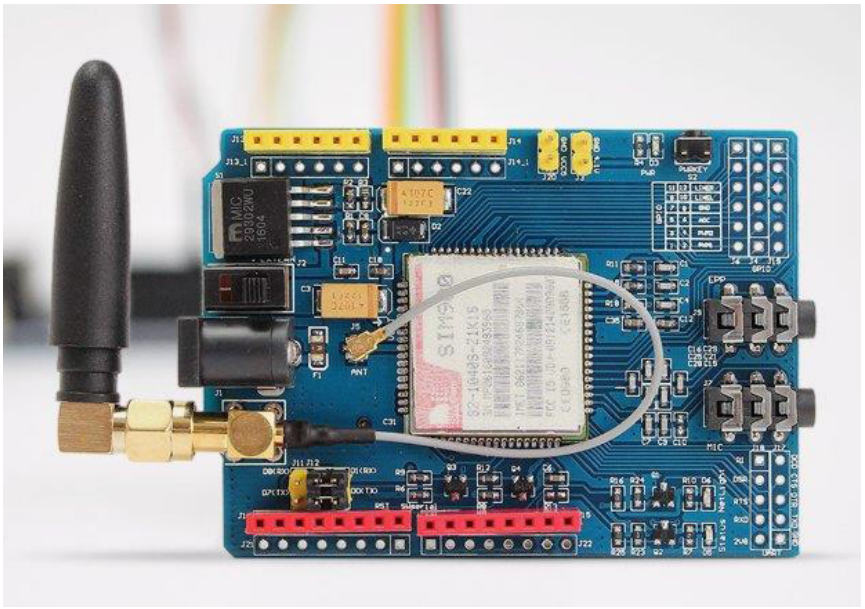


FIG 4.9GSM CIRCUIT BOARD

Whether you want to listen to what happens in your house that's miles away from you or activate sprinkler system in your garden just with a silent call; Then SIM900 GSM/GPRS shield serves as a solid launching point for you to get you started with IoT!

SIM900 GSM/GPRS shield is a GSM modem, which can be integrated into a great number of IoT projects. You can use this shield to accomplish almost anything a normal cell phone can; SMS text messages, Make or receive phone calls, connecting to internet through GPRS, TCP/IP, and more! To top it off, the shield supports quad-band GSM/GPRS network, meaning it works pretty much anywhere in the world.

4.6.1 Hardware Overview of SIM900 GSM/GPRS Shield

The SIM900 GSM/GPRS shield is designed to surround the SIM900 chip with everything necessary to interface with Arduino, plus a few extra goodies to take advantage of the chip's unique features.

Let's familiarize ourselves with these features and abilities of the shield. Here's a quick overview:

4.6.2 The SIM900 shield packs a surprising amount of features into its little frame. Some of them are listed below:

- Supports Quad-band: GSM850, EGSM900, DCS1800 and PCS1900
- Connect onto any global GSM network with any 2G SIM
- Make and receive voice calls using an external earphone & electret microphone Send and receive SMS messages
- Send and receive GPRS data (TCP/IP, HTTP, etc.)
- Scan and receive FM radio broadcasts

- Transmit Power:
- Class 4 (2W) for GSM850
- Class 1 (1W) for DCS1800
- Serial-based AT Command Set
- U.FL and SMA connectors for cell antenna
- Accepts Full-size SIM Card

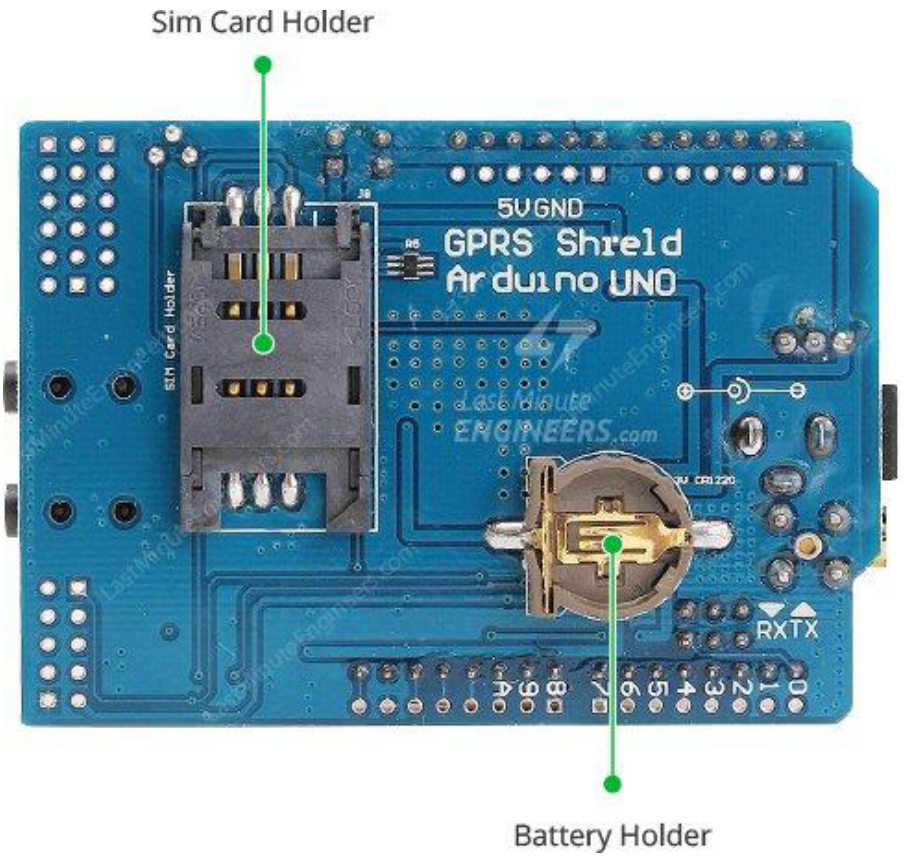


FIG 4.10 SIM900

5 Description of Arduino

5.1 Introduction:

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.

Over the years Arduino has been the brain of thousands of projects, from everyday objects to complex scientific instruments. A worldwide community of makers - students, hobbyists, artists, programmers, and professionals - has gathered around this open-source platform, their contributions have added up to an incredible amount of accessible knowledge that can be of great help to novices and experts alike.

Arduino was born at the Ivrea Interaction Design Institute as an easy tool for fast prototyping, aimed at students without a background in electronics and programming. As soon as it reached a wider community, the Arduino board started changing to adapt to new needs and challenges, differentiating its offer from simple 8-bit boards to products for IoT applications, wearable, 3D printing, and embedded environments. All Arduino boards are completely open-source, empowering users to

build them independently and eventually adapt them to their particular needs. The software, too, is open-source, and it is growing through the contributions of users worldwide.

5.2 Input and Output:

Arduino - ArduinoBoardUno Each of the 14 digital pins on the Uno can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms.

Table : Specification

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

Frequency (Clock Speed)	16 MHz

5.2.1 Power:

The Arduino Uno can be powered via USB or an external power supply, automatically selected. External power can come from an AC-to-DC adapter or battery. The board operates on a supply of 6 to 20 volts, with a recommended range of 7 to 12 volts to ensure stability and prevent damage.

5.2.2 Memory:

SRAM 2 KB EEPROM 1 KB Frequency (Clock Speed) 16 MHz

The ATmega328 has 32 KB (with 0.5 KB used for the bootloader). It also has 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library).

5.2.3 Communication:

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on

Windows, a .inf file is required. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1)

5.3 Arduino Pinpoint:

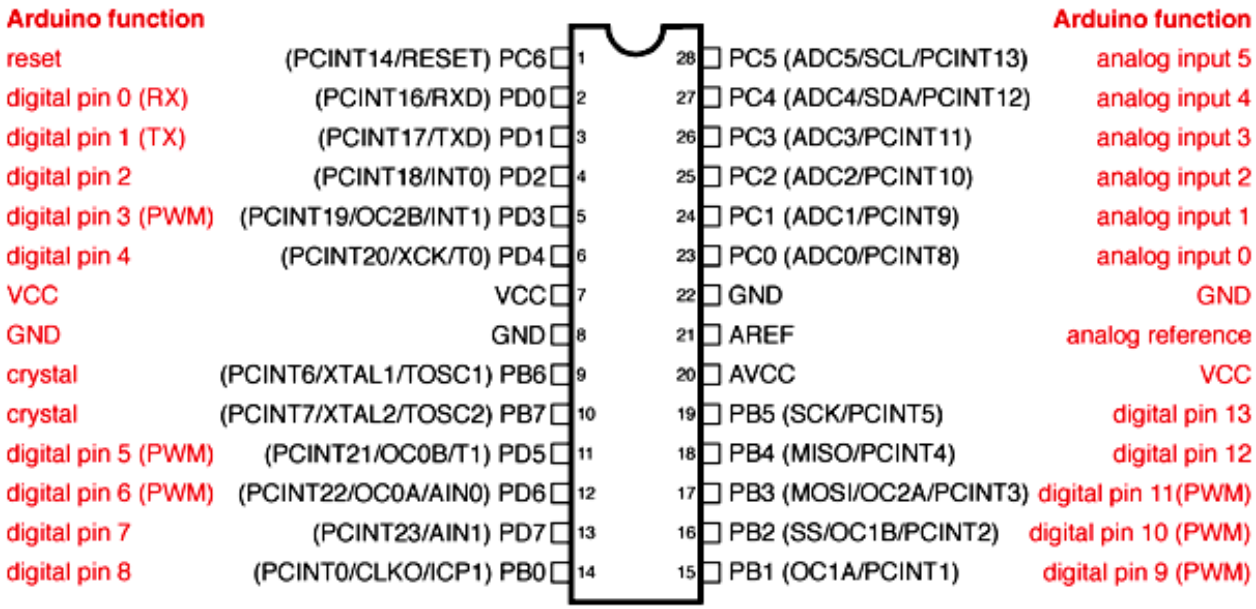


Fig 5.3:ARDUINO PINPOINT

5.4 DHT11 Temperature and Humidity Sensor

- DHT11 is a part of DHTXX series of Humidity sensors. The other sensor in this series is DHT22.

- Both these sensors are Relative Humidity (RH) Sensor. As a result, they will measure both the humidity and temperature. Although DHT11 Humidity Sensors are cheap and slow, they are very popular among hobbyists and beginners.

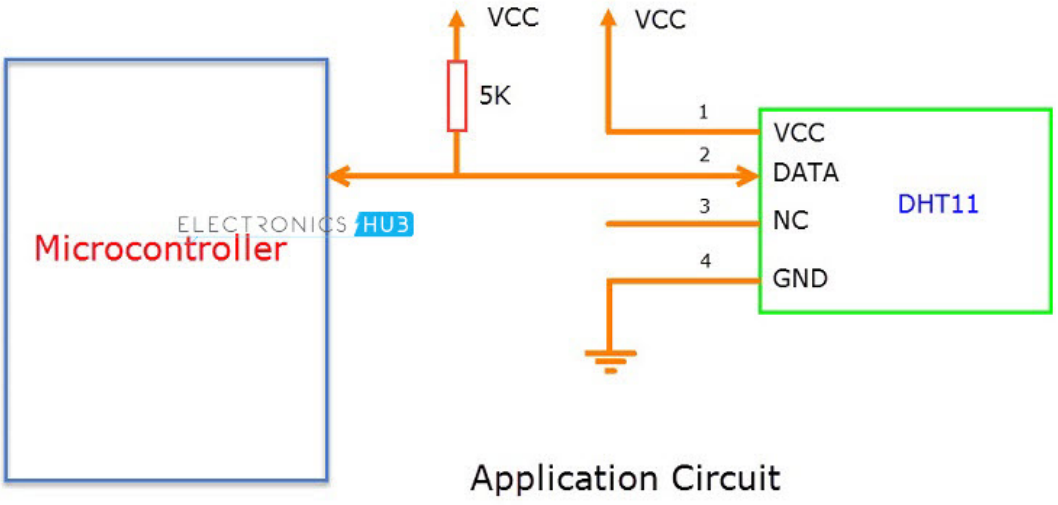


Fig 5.2 Application Circuit

All the DHT11 Sensors are accurately calibrated in the laboratory and the results are stored in the memory.

- A single wire communication can be established between any microcontroller like Arduino and the DHT11 Sensor.
- Also, the length of the cable can be as long as 20 meters. The data from the sensor consists of integral and decimal parts for both Relative Humidity (RH) and temperature.
- The data from the DHT11 sensor consists of 40 – bits and the format is as follows:

- 8 – Bit data for integral RH value, 8 – Bit data for decimal RH value, 8 – Bit data for integral Temperature value, 8 – Bit data for integral Temperature value and 8 – Bit data for checksum.

6 REFERENCE

1. Ebrahim Al Alkeem1, Dina Shehada1, Chan Yeob Yeun1,M. Jamal Zemerly ,Jiankun Hu

“New secure healthcare system using cloud of things”, Springer Science+Business Media

New York 2017

2. Yena Kim, SeungSeob Lee and SuKyoung Lee “Coexistence of ZigBee-based WBAN and

WiFi for Health Telemonitoring Systems” , DOI 10.1109/JBHI.2014.2387867, IEEE

Journal of Biomedical and Health Informatics

3. Mirza Mansoor Baig & Hamid Gholamhosseini “Smart Health Monitoring Systems: An

Overview of Design and Modeling”, Springer Science+Business Media New York 2013

4. Vandana Sharma, Ravi TiwariR, “A review paper on “IOT” & It’s Smart Applications", International Journal of Science, Engineering and Technology Research (IJSETR), Volume 5, Issue 2, pp 475-478, February 2016.

5.Mybotic, & Instructables. (2017, December 27). Blink LED Using ESP8266 NodeMCU Lua WiFi Tutorial. Retrieved from https://www.instructables.com/HOW-TO-BLINK-LED-USING-ESP8266/

CHAPTER 7

7.1 CONCLUSION

The results obtained from different sensor devices will be compared and analyzed in detail. The values are recorded using sensors and processed using microcontroller. For emergency send the alert signal to doctor. This system is low cost, self-monitoring device and used in remote areas efficiently.

7.2 FUTURE SCOPE OF EXPANSION

The project on Remote Heart Rate and Health Monitoring System using IoT has promising future prospects. Key areas for future development include advanced analytics, integration with wearable devices, expansion of monitored health parameters, telemedicine integration, user-friendly mobile application, and scalability for wider deployment. These enhancements can enable personalized health insights, seamless user experience, comprehensive health assessment, remote consultations, improved accessibility, and wider adoption of the system.

Chapter 8

APPENDIX 1

SAMPLE CODE

```
include <LiquidCrystal.h>

#include <DHT.h>

#include <SoftwareSerial.h>

#include <ArduinoJson.h>

#define DHTPIN 3

DHT dht(DHTPIN, DHT11);

LiquidCrystal lcd(13,12,11,10,9,8);

int hbval;

SoftwareSerial gsmSerial(7, 6);

SoftwareSerial nodemcu(5, 4);

String sms = "HIS/HER BODY CONDITION IS ABNORMAL, PLEASE HAVE A LOOK AT HIM/HER";

int HB;

const int beatsensor=A0;

void gsm()

{

  {

    if (gsmSerial.available() > 0)

      Serial.write(gsmSerial.read());
```

```
  }
  gsmSerial.println("AT+CMGF=1");    //Sets the GSM Module in Text Mode
  delay(500);  // Delay of 1000 milli seconds or 1 second
  gsmSerial.println("AT+CMGS=\"+919941414565\"\r"); // Replace x with mobile
number
  delay(500);
  gsmSerial.print(String(sms));
  delay(500);
  gsmSerial.println((char)26);// ASCII code of CTRL+Z
  delay(500);
}
void heartbeat()
{
   hbval= analogRead(A0);
  if(hbval>200)
  {
    HB=((hbval/1023)*9.516)*10);
  }
  else
  {
   HB=0;
  }
```

```
 if(HB>120)
{
  digitalWrite(2,HIGH);
  gsm();
}
else
{
  digitalWrite(2,LOW);
}
}
void setup()
{
      Serial.begin(115200);
      gsmSerial.begin(9600);
      nodemcu.begin(115200);
      lcd.begin(16, 2);
      lcd.print(" PATIENT HEALTH");
      lcd.setCursor(0,1);
       lcd.print("   MONITORING");
      delay(1500);
      lcd.clear();
      dht.begin();
```

```
      pinMode(2,OUTPUT);
      }
void loop()
{
 StaticJsonBuffer<500> jsonBuffer;
 JsonObject& root = jsonBuffer.createObject();
  float t = dht.readTemperature();
   heartbeat();
    if(t>34.3)
    {
     digitalWrite(2,HIGH);
     gsm();
    }
    else
    {
     digitalWrite(2,LOW);
    }
     Serial.println("Temperature: ");
     Serial.println(t);
     lcd.print("TEMPERATURE:");
     lcd.print(t);
     lcd.setCursor(0,1);
```

```
        lcd.print("HEARTBEAT :");
        lcd.print(HB);
        delay(1000);
        lcd.clear();
        root["a1"] = t;
  root["a2"] = HB;
  root.printTo(nodemcu);
  jsonBuffer.clear();
  delay(100);
}
```

APPENDIX 2

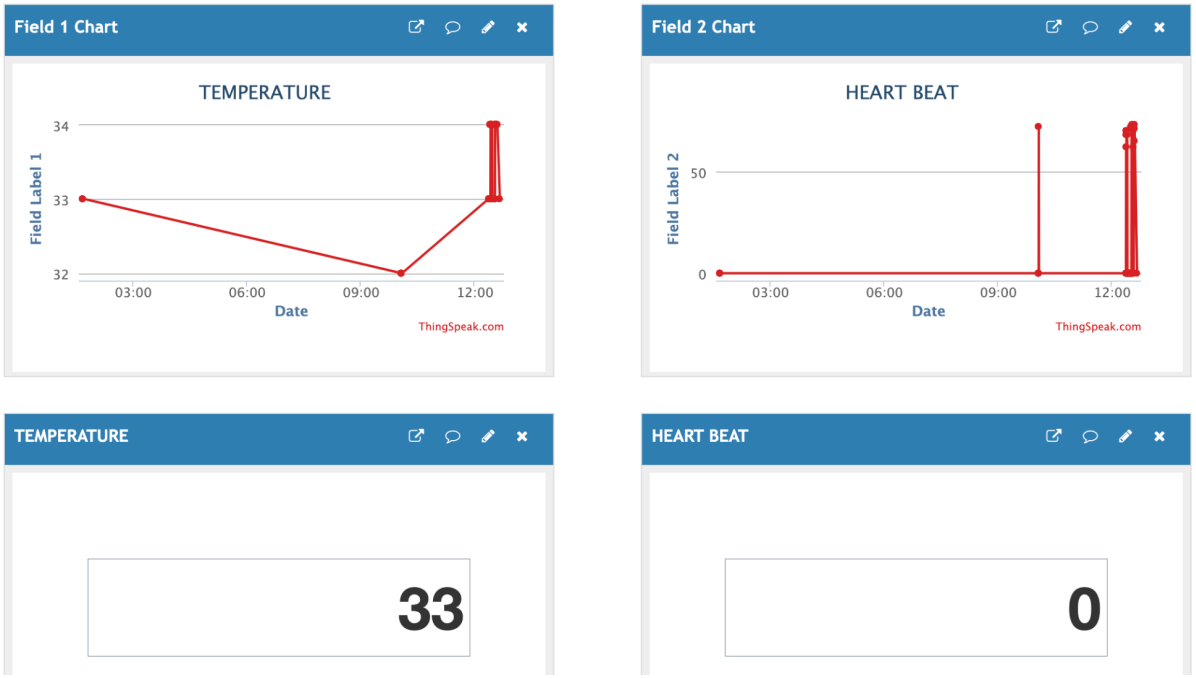


Fig 9.1 Output Screenshot

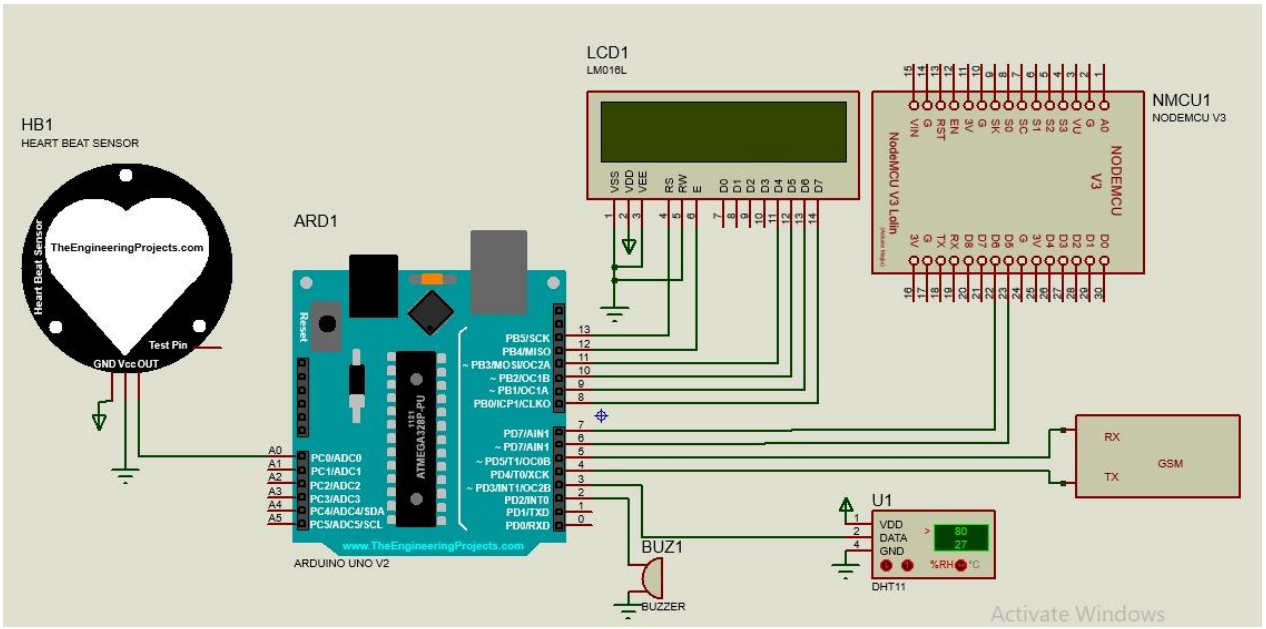


Fig 9.2 Final Setup