

VARICOSE VEINS DISEASE DETECTION AND AUTOMATED TREATMENT USING BAN (BODY AREA NETWORK

A PROJECT REPORT

Submitted by

**PRAVEEN KUMAR.K [211419104198]
SUNDARESWAR.N [211419104274]
VISHNU.M [211419104308]**

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING



PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

APRIL 2023

PANIMALAR ENGINEERING COLLEGE
(An Autonomous Institution, Affiliated to Anna University, Chennai)

BONAFIDE CERTIFICATE

Certified that this project report “**VARICOSE VEINS DISEASE DETECTION AND AUTOMATED TREATMENT USING BAN (BODY AREA NETWORK)**” Is the Bonafide work of “**PRAVEENKUMAR.K(211419104198), SUNDARESWAR.N(211419104274), VISHNU.M(211419104308)**” who carried out the project work under supervision.

SIGNATURE

Dr.L. JABASHEELA,M.E., Ph.D.,
PROFESSOR
HEAD OF THE DEPARTMENT

DEPARTMENT OF CSE,
PANIMALAR ENGINEERING COLLEGE,
NASARATHPETTAI,
POONAMALLEE,
CHENNAI-600 123.

SIGNATURE

Mr.M.MOHAN,B.E.,M.Tech.,(Ph.D.),
SUPERVISOR
ASSISTANT PROFESSOR GRADE I

DEPARTMENT OF CSE,
PANIMALAR ENGINEERING COLLEGE,
NASARATHPETTAI,
POONAMALLEE,
CHENNAI-600 123.

Certified that the above candidates were examined in the End Semester Project
Viva-Voce Examination held on 10.03.2023

INTERNAL EXAMINER

EXTERNAL EXAMINER

DECLARATION BY THE STUDENT

We **PRAVEENKUMAR.K (211419104198), SUNDARESWAR.N (211419104274), VISHNU.M (211419104308)** hereby declare that this project report titled **“VARICOSE VEINS DISEASE DETECTION AND AUTOMATED TREATMENT USING BAN (BODY AREA NETWORK)”** under the guidance of **Mr.M.MOHAN,B.E.,M.Tech.,(Ph.D.)** is the original work done by us and we have not plagiarized or submitted to any other degree in any university by us.

ACKNOWLEDGEMENT

We would like to express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D.** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We express our sincere thanks to our beloved Directors **Tmt.C.VIJAYARAJESWARI, Dr.C.SAKTHI KUMAR,M.E.,Ph.D** and **Dr.SARANYASREE SAKTHI KUMAR B.E.,M.B.A.,Ph.D.**, for providing us with the necessary facilities to undertake this project.

We also express our gratitude to our Principal **Dr.K.Mani, M.E., Ph.D.** who facilitated us in completing the project.

We thank the Head of the CSE Department, **Dr. L.JABASHEELA , M.E.,Ph.D.**, for the support extended throughout the project.

We would like to thank my **Mr.M.MOHAN,B.E.,M.Tech.,(Ph.D.)** and all the faculty members of the Department of CSE for their advice and encouragement for the successful completion of the project.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	V
	LIST OF TABLES	VI
	LIST OF FIGURES	VII
1	INTRODUCTION	1
	1.1 SCOPE OF THE PROJECT	3
	1.2 EXISTING SYSTEM	3
	1.2.1 EXISTING SYSTEM DISADVANTAGES	3
	1.3 PROPOSED SYSTEM	4
	1.3.1 PROPOSED SYSTEM ADVANTAGES	5
2	LITERATURE SURVEY	6
3	SYSTEM SPECIFICATIONS	7
	3.1 HARDWARE REQUIREMENTS	11
	3.1.1 ARDUINO	11
	3.1.2 INTERNET OF THINGS	13
	3.1.3 ESP-12E BASED NODEMCU	20
	3.1.4 LIQUID CRYSTAL DISPLAY (LCD)	30
	3.1.5 12C LCD ADAPTAR	36
	3.1.6 DS18B20 TEMPERATURE SENSOR	46

	3.1.7 BATTERY	52
	3.1.8 ZIGBEE	59
	3.1.9 FORCE SENSOR	60
	3.1.10 VIBRATION MOTORS	67
	3.2 SOFTWARE REQUIREMENTS	73
	3.2.1 EMBEDDED C	73
	3.2.2 ARDUINO SOFTWARE IDE	83
4	SYSTEM DESIGN	99
	4.1 BLOCK DIAGRAM	100
	4.2 ENTITY RELATIONSHIP DIAGRAM	101
	4.2.1 ER DIAGRAM FOR VARICOSE VEINS	102
	DETECTION USING BAN	
	4.3 DATA FLOW DIAGRAM	102
	4.3.1 LEVEL – 0 DFD	105
	4.3.2 LEVEL – 1 DFD	105
	4.4 UML DIAGRAMS	106
	4.4.1 USE CASE DIAGRAM	108
	4.4.4.1 USE CASE DIAGRAM FOR	109
	VARICOSE VEIN DETECTION USING BAN	
	4.4.2 CLASS DIAGRAM	110
	4.4.2.1 CLASS DIAGRAM FOR VARICOSE	111
	VEINS DETECTION USING BAN	

	4.4.3 ACTIVITY DIAGRAM	112
	4.4.3.1 ACTIVITY DIAGRAM FOR VARICOSE	113
	VEINS DETECTION USING BAN	
5	SYSTEM MODULES	114
	5.1 MODULE DESCRIPTION	115
	5.2 MODULE NAMES	115
	5.2.1 Temperature Monitor	115
	5.2.2 Rectify Varicose Veins	116
6	SYSTEM TESTING	117
	6.1 SYSTEM TESTING	118
	6.1.1 Unit Testing	119
	6.1.2 Integration Testing	119
	6.2 TEST CASES AND REPORTS	120
	6.2.1 Upper Body Module	120
	6.2.2 Lower Body Module	121
	6.2.3 Database Module	122
7	CONCLUSION AND FUTURE ENHANCEMENT	124
8	SCREENSHOTS	126
9	APPENDIX	129
10	BIBLIOGRAPHY	132

ABSTRACT

In existing system separate person need to check the patient and give treatment, whenever the patient needs treatment, they need a medical assistant help. So, to avoid this need we induced a new system automated treatment using BAN (Body area network). Varicose veins are twisted, enlarged veins. Any superficial vein may become varicose, but the veins most affected are those in your legs. That is because standing and walking upright increases the pressure in the veins of your lower body. For many people, varicose veins and spider veins a common, mild variation of varicose veins are simply a cosmetic concern. For other people, varicose veins can cause aching pain and discomfort. Sometime varicose veins lead to more-serious problems. Treatment may involve self-care measures or procedures by your doctor to close or remove veins. Approximately 23% of US adults have varicose veins. If spider telangiectasia and reticular veins are also considered, the prevalence increases to 80% of men and 85% of women. Generally, more common in women and older adults, varicose veins affect 22 million women and 11 million men between the ages of 40 to 80 years. In this paper we proposed monitoring the varicose patients and automatically given the treatment whenever the patient need.

Keywords: DB18B20 Temperature Sensor, Body Area Network, Zigbee Tx & Rx, Vibration Motor

LIST OF TABLES

TABLE NO.	TABLE NAME	PAGE NO.
Table 3.1	LCD Function	32
Table 3.2	Command Codes for LCD	34
Table 3.3	DS18B20 Sensor to Arduino Board	46
Table 3.4	PP (Power Pack) Battery Family	55
Table 3.5	Testing and Charging	58
Table 3.6	FSR Analog Voltage	64
Table 3.7	Specifications	66
Table 3.8	Writing Sketches	85
Table 4.1	Different Symbols Used in Dataflow Diagram	104
Table 6.1	Temperature Sensor 1	120
Table 6.2	NodeMCU 1	120
Table 6.3	Temperature Sensor 2	121
Table 6.4	NodeMCU 2	121
Table 6.5	Pressure Sensor	122
Table 6.6	PLX-DAQ	122

LIST OF FIGURES

FIGURE NO.	FIGURE NAME	PAGE NO.
Figure 3.1	ESP-12E Based Node MCU	20
Figure 3.2	Node MCU Pin Configuration	21
Figure 3.3	ESP-12E Architecture	22
Figure 3.4	ESP-12E Pin Configuration	23
Figure 3.5	Pin Description	24
Figure 3.6	Pin Mode	24
Figure 3.7	Receiver Sensitivity	25
Figure 3.8	Schematics Of ESP-12E Wi-Fi-Module	25
Figure 3.9	Interfaces	27
Figure 3.10	Node MCU GPIO For Lua	28
Figure 3.11	Single-Chip USB-To-UART	29
Figure 3.12	Liquid Crystal Display 16x2	30
Figure 3.13	LCD Pinout Diagram	31
Figure 3.14	Data Register	33
Figure 3.15	Displaying Custom Characters on 16x2 LCD	35
Figure 3.16	PCF8574	36
Figure 3.17	Backlight Drive Pin	36
Figure 3.18	Solder Pads	37
Figure 3.19	Texas Instruments	37
Figure 3.20	NXP Semiconductors	38
Figure 3.21	IC2 LCD Pinout	39
Figure 3.22	LCD Contrast	40
Figure 3.23	Liquidcrystal_I2C	40
Figure 3.24	Sketch_Nov23A	41
Figure 3.25	Library Management	42

Figure 3.26	Com6	43
Figure 3.28	DS18B20 Temperature Sensor	46
Figure 3.29	DS18B20 Waterproof Version	46
Figure 3.30	Normal Mode	47
Figure 3.31	Parasite Mode	48
Figure 3.32	Installing Libraries	49
Figure 3.33	Installing Dallas Temperature Library	50
Figure 3.34	Arduino IDE Serial Monitor	51
Figure 3.35	Temperature Displayed	51
Figure 3.36	Multiple DS18B20 Sensors	52
Figure 3.37	Schematic Diagram	52
Figure 3.38	Connectors	56
Figure 3.39	9-Volt Battery	56
Figure 3.40	Alkaline Battery	56
Figure 3.41	Three Different 9-Volt Primary Battery	56
Figure 3.42	Rechargeable Battery	56
Figure 3.43	Force Sensor	60
Figure 3.44	Static Resistor	62
Figure 3.45	Stand-Off Resistance	63
Figure 3.46	FSR Analog Voltage	64
Figure 3.47	Force Sensor Connecting with Arduino	65
Figure 3.48	Exploded Coin Motor	68
Figure 3.49	Coin Motor Commutation Circuitry	70
Figure 3.50	Coin Motor Equivalent Circuit	70
Figure 3.51	Self-Adhesive Vibration Motor Mounting	71
Figure 3.52	8mm Coin (Shaftless) Vibration Motor	71
Figure 3.53	10mm Coin Spring-Tab Vibration Motor	72
Figure 3.54	Block Diagram of Embedded C	73
Figure 3.55	Basic Declaration of Embedded	74
Figure 3.56	Steps of Embedded C Programming	75

Figure 3.57	RTOS Process	76
Figure 3.58	Basic Structure of Embedded System	77
Figure 3.59	Flow Chart of Developing Embedded System	78
Figure 3.60	Arduino Software IDE	83
Figure 3.61	Writing Sketches	84
Figure 3.62	File	85
Figure 3.63	Edit	87
Figure 3.64	Sketch	89
Figure 3.65	Port	91
Figure 3.66	Help	92
Figure 3.67	Sketchbook	93
Figure 3.68	Uploading	94
Figure 3.69	Libraries	95
Figure 3.70	Arduino 1.8.5 Board	96
Figure 3.71	Com5 Port	96
Figure 3.72	Language Support	97
Figure 3.73	C Language Source Code	98
Figure 4.1	Block Diagram for Varicose Veins Disease Detection	100
Figure 4.2	ER Diagram for Varicose Veins Disease Detection	102
Figure 4.3	Level – 0 DFD	105
Figure 4.4	Level – 1 DFD	106
Figure 4.5	Use Case Diagram for Varicose Veins Detection	109
Figure 4.6	Class Diagram for Varicose Veins Detection	111
Figure 4.7	Activity Diagram for Varicose Veins Detection	113
Figure: 5.1	Temperature Monitor	116
Figure: 5.2	Rectify Varicose Veins	116
Figure 8.1	Upper Body Module	127
Figure 8.2	Lower Body Module	127
Figure 8.3	Varicose vein disease detected	128
Figure 8.4	Database Module	128

CHAPTER 1

INTRODUCTION

CHAPTER 1

INTRODUCTION

Varicose veins are a common medical condition that affects millions of people worldwide. It is a condition that occurs when the veins in the legs become enlarged and twisted, resulting in discomfort, pain, and cosmetic concerns. Timely diagnosis and treatment of varicose veins are crucial to prevent complications and improve the quality of life for patients. In recent years, there has been a growing interest in the development of innovative technologies to improve the detection and treatment of varicose veins. One such technology is the use of Body Area Networks (BANs) for automated detection and treatment of varicose veins. BANs are a type of wireless network that involves wearable devices and sensors that can monitor the body's functions and transmit data to a centralized system for analysis and processing. The transfer of health monitoring data from multiple patients using wireless body-area networks requires the use of robust, and energy and bandwidth efficient multiple- access schemes. This paper considers the frequency-division multiple access for the wireless uplink to a fixed access point when using infrared signals to collect medical data from several patients inside an emergency waiting room. We consider a new approach transmitting only the real part of a complex-valued signal, where no such constraints imposed. Based on the proposed scheme, and taking into account the limited dynamic range of an infrared light-emitting diode, we study the performance of direct current biased and asymmetrically clipped schemes, and show their advantage in terms of energy efficiency and computational complexity, as compared with the conventional schemes. For instance, we show that by using asymmetric clipping, around 35 mW less transmit power is needed to achieve a bit error rate of 10^{-3} in the considered scenario. The technological developments of the project are varicose veins patients monitoring and automated treatment is to give a temporary solution for varicose veins patients to make blood circulation and normalization.

1.1 Scope of the project

The technological developments of the project is varicose veins patients monitoring and automated treatment is to give a temporary solution for varicose veins patients to made blood circulation and normalization.

1.2 Existing System

- Sclerotherapy: This is a common treatment option for varicose veins that involves injecting a solution into the veins to close them off.
- Endovenous laser therapy (EVLT): This is a minimally invasive procedure that involves using laser energy to close off varicose veins.
- Radiofrequency ablation (RFA): This is another minimally invasive procedure that uses radiofrequency energy to close off varicose veins.
- Compression stockings: These are special stockings that help to improve blood flow in the legs and reduce swelling associated with varicose veins.
- Exercise and weight management: Regular exercise and maintaining a healthy weight can help to prevent the development of varicose veins and improve symptoms in those who already have them.
- Surgical procedures: Surgical procedures such as vein stripping and ligation are invasive treatment options for varicose veins. While they are effective, they can be painful and require a longer recovery time.

1.2.1 Existing System Disadvantages

- No Automatic system to give the treatment.
- Treatment cost is high.
- Invasiveness and time-consuming.
- Requires the assistance of a physician.
- Some treatments may not be suitable for all patients.

1.3 Proposed System

Varicose veins are a common medical condition that affects a significant number of people world-wide. While there are various treatment options available, most of them are invasive, time-consuming, and may not provide immediate relief to patients. In this proposed system, we aim to use a combination of a thermistor and a pressure sensor to detect the abnormality in varicose veins and provide temporary treatment to the patients. The system will consist of a wearable device that will be placed over the affected area. The device will contain a thermistor sensor that will measure the temperature of the skin over the varicose veins and a pressure sensor to detect the swelling. The sensors will be connected to a microcontroller unit that will process the data and analyze it for abnormality. If the temperature and pressure sensor readings indicate abnormality, the microcontroller will activate a massaging mechanism to provide temporary relief to the patient. The massaging mechanism will be in the form of a pair of socks containing a vibration motor that will be worn by the affected patient. The vibration effect will help to reduce the inflammation and pain associated with varicose veins, providing immediate relief to the patient. In addition to the temporary treatment, the proposed system will also collect data on the patient's physiological signals, such as temperature and pressure, which will be transmitted to a centralized system for analysis and diagnosis. The data collected will help healthcare providers to monitor the patient's condition and provide personalized treatment options based on the severity of the condition. The proposed system has several advantages over current treatment options for varicose veins. It is non-invasive, easy to use, and provides immediate relief to the patient. The system also collects real-time data that can be used for further analysis and diagnosis, enabling healthcare providers to provide more personalized and effective treatment options.

1.3.1 Proposed System Advantages

- Does not need the human help.
- Non-invasive: The proposed system is non-invasive, which means it does not require any surgery or incision. This makes it less risky and more comfortable for patients.
- Easy to use: The wearable device used in the proposed system is easy to use and can be worn by patients without any discomfort.
- Immediate relief: The proposed system provides immediate relief to patients suffering from varicose veins by providing temporary treatment through a massaging mechanism.
- Real-time data collection: The system collects real-time data on the patient's physiological signals, such as temperature and pressure, which can be used for further analysis and diagnosis.
- Personalized treatment options: The data collected by the proposed system can be used by healthcare providers to provide personalized treatment options based on the severity of the condition.
- Cost-effective: The proposed system is cost-effective compared to traditional treatment options, which can be expensive and time-consuming.

CHAPTER 2

LITERATURE SURVEY

CHAPTER 2

LITERATURE SURVEY

Md Jahid Hasan, Mohammad Ali Khalighi, Volker Jungnickel, Luis Nero Alves, Bastien Be´chadergue^[1] et al An Energy Efficient Optical Wireless OFDMA Scheme for Medical Body-Area Networks :IEEE Transactions on Green Communications and Networking.

Radiofrequency ablation of the superficial venous systems has become one of the mainstays of minimally invasive approaches to varicose veins and chronic venous insufficiency. These procedures have high rates of success with scarce complications. Radiofrequency obliteration of the subcutaneous veins can be a method of choice in patients with obesity.

O. Shumkov, M. Smagin, V. Nimaev, A. Sadovskii^[2] et al Radiofrequency ablation of varicose veins in obese patients :International Multiconference Bioinformatics of Genome Regulation and Structure\Systems Biology, 2018.

The doctor determines whether there are lesions in the human body through the diagnosis of medical images, and classifies and identifies the lesions. Therefore, the automatic classification and recognition of medical images has received extensive attention. Since the inflammatory phenomenon of vascular endothelial cells is closely related to the varicose veins of the lower extremities, in order to realize the automatic classification and recognition of varicose veins of the lower extremities, this paper proposes a varicose vein recognition algorithm based on vascular endothelial cells inflammation images and multi-scale deep learning, called MSDCNN.

O. Haddad, M. A. Khaleghi, S. Zvanovec, and M. Adel^[3] et al Channel characterization and modeling for optical wireless body-area networks :IEEE OpenJournal of the Communications Society, vol. 1, pp. 760–776, June 2020.

During the last decade less invasive endogenous methods of treatment of lower limb varicose veins (LLVV) have obtained widespread appreciation.

Nevertheless, the problem of improving their long-term results is still actual. The aim of this study is investigation of the bio thermomechanical response of the venous wall to the low frequency ultrasound exposure, which is an advanced method in the treatment of LLVV. The model designed to analyze frequency range of the ultrasound instrument and the different values of its pullback velocity. According to the simulation results, the frequency range, in which necrotic changes of the venous wall can be caused was determined. The dependencies of the venous wall collagen denaturation time on the temperature and pullback velocity of the ultrasound instrument were obtained. After the developed model is supplemented with temperature dependencies of the physical properties of the venous wall, these results can be used to form requirements for ultrasound treatment modes for lower limb varicose veins

S. Movassaghi, M. Abolhasan, J. Lipman, D. Smith, and A. Jamalipour^[5] et al
Wireless body area networks A survey :IEEE Communications Sur- veys & Tutorials, vol. 16, no. 3, pp. 1658– 1686, Mar. 2014.

The mechanical parameters of the human veins are often measured in different test conditions. There are no studies on the influence of different test conditions on these parameters to date. This complicates the comparison between experimental data in different studies and creates the need to establish a universal test method for mechanical parameters acquisition. The influence of the test conditions of the test results were studied by comparison between mechanical parameters of veins using uniaxial tension in air at a room temperature and in the sodium chloride solution at 37 °C. Due to the substantial nonlinearity of the stress strain data and the lack of suitable constitutive equation the proposed parameters for comparison were maximum stress, maximum strain, and Young's moduli at small and large strains.

B. Latre´, B. Braem, I. Moerman, C. Blondia, and P. Demeester^[7] et al
A survey on wireless body area networks :Wireless Networks, vol. 17, no. 1, pp. 1–18, Jan. 2011.

Capillary blood pressure (CBP) is the primary driving force for fluid exchange

across micro vessels. Subclinical systemic venous congestion prior to overt peripheral edema can directly result in elevated peripheral CBP. Therefore, CBP measurements can enable timely edema control in a variety of clinical cases including venous insufficiency, heart failure and so on. However, currently CBP measurements can be only done invasively and with a complicated experimental setup. In this work, we proposed an opto-mechanical system to achieve non-invasive and automatic CBP measurements through modifying the widely implemented oscillometer technique in home-use arterial blood pressure monitors. The proposed CBP system is featured with a blue light photoplethysmography sensor embedded in finger/toe cuffs to probe skin capillary pulsations. The experimental results demonstrated the proposed CBP system can track local CBP changes induced by different levels of venous congestion. Leveraging the decision tree technique, we demonstrate the use of a multi-site CBP measurement at fingertips and toes to classify four categories of subjects including patients with peripheral arterial disease, varicose veins, and heart failure. Our work demonstrates the promising non-invasive CBP measurement as well as its great potential in realizing point-of-care systems for the management of cardiovascular diseases. point-of-care systems for the management of cardiovascular diseases.

CHAPTER 3

SYSTEM SPECIFICATIONS

CHAPTER 3

SYSTEM SPECIFICATIONS

HARDWARE REQUIREMENTS

- POWER SUPPLY
- ESP8266
- DB18B20 TEMPERATURE SENSOR
- I2C LCD MODULE
- LCD
- ZIGBEE
- VIBRATION MOTOR
- FORCE SENSOR

SOFTWARE REQUIREMENTS

- ARDUINO IDE
- EMBEDDED C

3.1 HARDWARE DESCRIPTION

3.1.1 Arduino

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards can 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. 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 needs. The software, too, is open-source, and it is growing through the contributions of users worldwide.

Why Arduino?

Thanks to its simple and accessible user experience, Arduino has been used in thousands of different projects and applications. The Arduino software is easy-to use for beginners, yet flexible enough for advanced users. It runs on Mac, Windows, and Linux. Teachers and students use it to build low-cost scientific tools to prove chemistry and physics principles, or to get started with programming and robotics. Designers and architects build interactive prototypes, musicians and artists use it for installations and to experiment with new musical instruments. Makers, of course, use it to build many of the projects exhibited at the Maker Faire, for example. Arduino is a key tool to learn new things. Anyone - children, hobbyists, artists, programmers - can start tinkering just following the step- by-step instructions of a kit, or sharing ideas online with other members of the Arduino community. There are many other microcontrollers and microcontroller platforms available for physical computing. Parallax Basic Stamp, Netmedia's BX-24, Phidgets, MIT's Handyboard, and many others offer similar functionality.

- **Inexpensive** - Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand, and even the pre-assembled Arduino modules cost less than \$50.

- **Cross-platform** - The Arduino Software (IDE) runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.
- **Simple, clear programming environment** - The Arduino Software (IDE) is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it is conveniently based on the Processing programming environment, so students learning to program in that environment will be familiar with how the Arduino IDE works.
- **Open source and extensible software** - The Arduino software is published as open-source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people wanting to understand the technical details can make the leap from Arduino to the AVR C programming language on which it's based. Similarly, you can add AVR-C code directly into your Arduino programs if you want to.
- **Open source and extensible hardware** - The plans of the Arduino boards are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the breadboard version of the module in order to understand how it works and save money.

3.1.2 Internet Of Things

The **Internet of Things (IoT)** is the network of physical devices, vehicles, buildings, and other items embedded with electronics, software, sensors, actuators, and network connectivity that enable these objects to collect and exchange data. In 2013 the Global Standards Initiative on Internet of Things (IoT-GSI) defined the IoT as "the infrastructure of the information society. The IoT allows objects to be sensed and controlled remotely across existing network infrastructure, creating opportunities for more direct integration of the physical world into computer-based systems, and resulting in improved efficiency, accuracy, and economic benefit. When IoT is augmented with sensors and

actuators, the technology becomes an instance of the more general class of cyber-physical systems, which also encompasses technologies such as smart grids, smart homes, intelligent transportation, and smart cities. Each thing is uniquely identifiable through its embedded computing system but can interoperate within the existing Internet infrastructure. Experts estimate that the IoT will consist of almost 50 billion objects by 2020.

Infrastructure

The Internet of Things will become part of the fabric of everyday life. It will become part of our overall infrastructure just like water, electricity, telephone, TV and most recently the Internet. Whereas the current Internet typically connects full-scale computers, the Internet of Things (as part of the Future Internet) will connect everyday objects with a strong integration into the physical world.

1. Plug and Play Integration

If we look at IoT-related technology available today, there is a huge heterogeneity. It is typically deployed for very specific purposes and the configuration requires significant technical knowledge and may be cumbersome. To achieve a true Internet of Things we need to move away from such small-scale, vertical application silos, towards a horizontal infrastructure on which a variety of applications can run simultaneously.

2. Infrastructure Functionality

The infrastructure needs to support applications in finding the things required. An application may run anywhere, including on the things themselves. Finding things is not limited to the start-up time of an application. Automatic adaptation is needed whenever relevant new things become available, things become unavailable or the status of things changes. The infrastructure must support the monitoring of such changes and the adaptation that is required as a result of the changes.

3. Physical Location and Position

As the Internet of Things is strongly rooted in the physical world, the notion of physical location and position are very important, especially for finding things, but also for deriving knowledge. Therefore, the infrastructure must support finding things according to location. Taking mobility into account, localization technologies will play an important role for the Internet of Things and may become embedded into the infrastructure of the Internet of Things.

4. Security and Privacy

In addition, an infrastructure needs to provide support for security and privacy functions including identification, confidentiality, integrity, non-repudiation authentication and authorization. Here the heterogeneity and the need for interoperability among different ICT systems deployed in the infrastructure and the resource limitations of IoT devices (e.g., Nano sensors) must be taken into account.

Data Management

Data management is a crucial aspect in the Internet of Things. When considering a world of objects interconnected and constantly exchanging all types of information, the volume of the generated data and the processes involved in the handling of those data become critical. A long-term opportunity for wireless communications chip makers is the rise of Machine-to-Machine (M2M) computing, which one of the enabling technologies for Internet of Things. This technology spans a broad range of applications. While there is consensus that M2M is a promising pocket of growth, analyst estimates on the size of the opportunity diverge by a factor of four. Conservative estimates assume roughly 80 million to 90 million M2M units will be sold in 2014, whereas more optimistic projections forecast sales of 300 million units. Based on historical analyses of adoption curves for similar disruptive technologies, such as portable MP3 players and antilock braking systems for cars, it is believed that unit sales in M2M could rise by as much as a factor of

ten over the next five years. There are many technologies and factors involved in the “data management” within the IoT context. Some of the most relevant concepts which enable us to understand the challenges and opportunities of data management

- Data Collection and Analysis
- Big Data
- Semantic Sensor Networking
- Virtual Sensors
- Complex Event Processing.

Application Areas

In the last few years, the evolution of markets and applications, and therefore their economic potential and their impact in addressing societal trends and challenges for the next decades has changed dramatically. Societal trends are grouped as: health and wellness, transport and mobility, security and safety, energy and environment, communication, and e-society, as presented in Figure 2.15. These trends create significant opportunities in the markets of consumer electronics, automotive electronics, medical applications, communication, etc. The applications in these areas benefit directly by the More-Moore and More-than-Moore semiconductor technologies, communications, networks, and software developments.

A) Cities

Smart Parking: Monitoring of parking spaces availability in the city.

Structural health: Monitoring of vibrations and material conditions in buildings, bridges, and historical monuments.

Noise Urban Maps: Sound monitoring in bar areas and centric zones in real time.

Traffic Congestion: Monitoring of vehicles and pedestrian levels to optimize

Smart Lightning: Intelligent and weather adaptive lighting in street lights.

Waste Management: Detection of rubbish levels in containers to optimize the trash

collection routes.

Intelligent Transportation Systems: Smart Roads and Intelligent Highways

with warning messages and diversions according to climate conditions and unexpected events like accidents or traffic jams.

B) Environment

Forest Fire Detection: Monitoring of combustion gases and pre-emptive fire conditions to define alert zones.

Air Pollution: Control of CO₂ emissions of factories, pollution emitted by cars and toxic gases generated in farms.

Landslide and Avalanche Prevention: Monitoring of soil moisture, vibrations and earth density to detect dangerous patterns in land conditions.

Earthquake Early Detection: Distributed control in specific places of tremors.

C) Water

Water Quality: Study of water suitability in rivers and the sea for fauna and Eligibility for drinkable use.

Water Leakages: Detection of liquid presence outside tanks and pressure variations along pipes.

River Floods: Monitoring of water level variations in rivers, dams, and reservoirs.

D) Energy Smart Grid, Smart Metering

Smart Grid: Energy consumption monitoring and management.

Tank level: Monitoring of water, oil and gas levels in storage tanks and cisterns. **Photovoltaic Installations:** Monitoring and optimization of performance in solar energy plants.

Water Flow: Measurement of water pressure in water transportation systems.

Silos Stock Calculation: Measurement of emptiness level and weight of the goods.

E) Security & Emergencies

Perimeter Access Control: Access control to restricted areas and detection of people in non-authorized areas.

Liquid Presence: Liquid detection in data centers, warehouses and sensitive building grounds to prevent break downs and corrosion.

Radiation Levels: Distributed measurement of radiation levels in nuclear power stations surroundings to generate leakage alerts.

Explosive and Hazardous Gases: Detection of gas levels and leakages in industrial environments, surroundings of chemical factories and inside mines.

F) Industrial Control

M2M Applications: Machine auto-diagnosis and assets control.

Indoor Air Quality: Monitoring of toxic gas and oxygen levels inside chemical plants to ensure workers and goods safety.

Temperature Monitoring: Control of temperature inside industrial and medical fridges with sensitive merchandise.

Ozone Presence: Monitoring of ozone levels during the drying meat process in food factories.

Indoor Location: Asset indoor location by using active (ZigBee, UWB) and passive tags (RFID/NFC).

Vehicle Auto-diagnosis: Information collection from CAN Bus to send real time alarms to emergencies or provide advice to drivers.

G) Agriculture

Wine Quality Enhancing: Monitoring soil moisture and trunk diameter in vineyards to control the amount of sugar in grapes and grapevine health.

Green Houses: Control micro-climate conditions to maximize the production of fruits and vegetables and its quality.

Golf Courses: Selective irrigation in dry zones to reduce the water resources required

in the green.

Meteorological Station Network: Study of weather conditions in fields to forecast ice formation, rain, drought, snow or wind changes.

Compost: Control of humidity and temperature levels in alfalfa, hay, straw, etc. to prevent fungus and other microbial contaminants.

H) Domestic & Home Automation

Energy and Water Use: Energy and water supply consumption monitoring to obtain advice on how to save cost and resources.

Remote Control Appliances: Switching on and off remotely appliances to avoid accidents and save energy.

Intrusion Detection Systems: Detection of window and door openings and violations to prevent intruders.

Art and Goods Preservation: Monitoring of conditions inside museums and art warehouses.

I) Health

Fall Detection: Assistance for elderly or disabled people living independent. **Medical Fridges:** Control of conditions inside freezers storing vaccines, medicines and organic elements.

Sportsmen Care: Vital signs monitoring in high performance centers and fields.

Ultraviolet Radiation: Measurement of UV sun rays to warn people not to be exposed in certain hours.

3.1.3 ESP-12E Based NodeMCU

The ESP8266 is the name of a micro controller designed by Expressive Systems. The ESP8266 itself is a self-contained Wi-Fi networking solution offering as a bridge from existing micro controller to Wi-Fi and is also capable of running self- contained applications. This module comes with a built in USB connector and a rich assortment of pin-outs. With a micro-USB cable, you can connect NodeMCU devkit to your laptop and flash it without any trouble, just like Arduino. It is also immediately breadboard friendly.

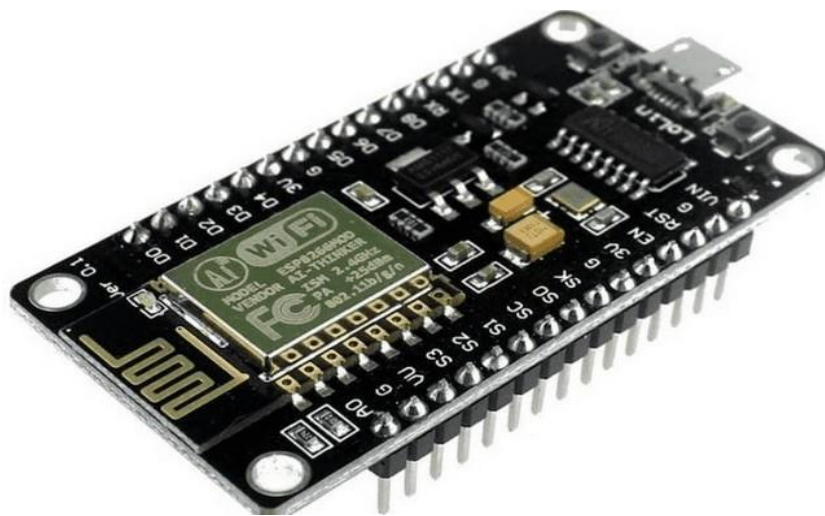


Figure 3.1 Esp-12e Based NodeMCU

ESP-12E Wi-Fi module is developed by Ai-thinker Team. core processor ESP8266 in smaller sizes of the module encapsulates Ten silica L106 integrates industry-leading ultra- low power 32-bit MCU micro, with the 16-bit short mode, Clock speed support 80 MHz, 160 MHz, supports the RTOS, integrated Wi-Fi MAC/BB/RF/PA/LNA, on-board antenna. The module supports standard IEEE802.11 b/g/n agreement, complete TCP/IP protocol stack. Users can use the add modules to an existing device networking, or building a separate network controller. ESP8266 is high integration wireless SOCs, designed for space and power constrained mobile platform designers.

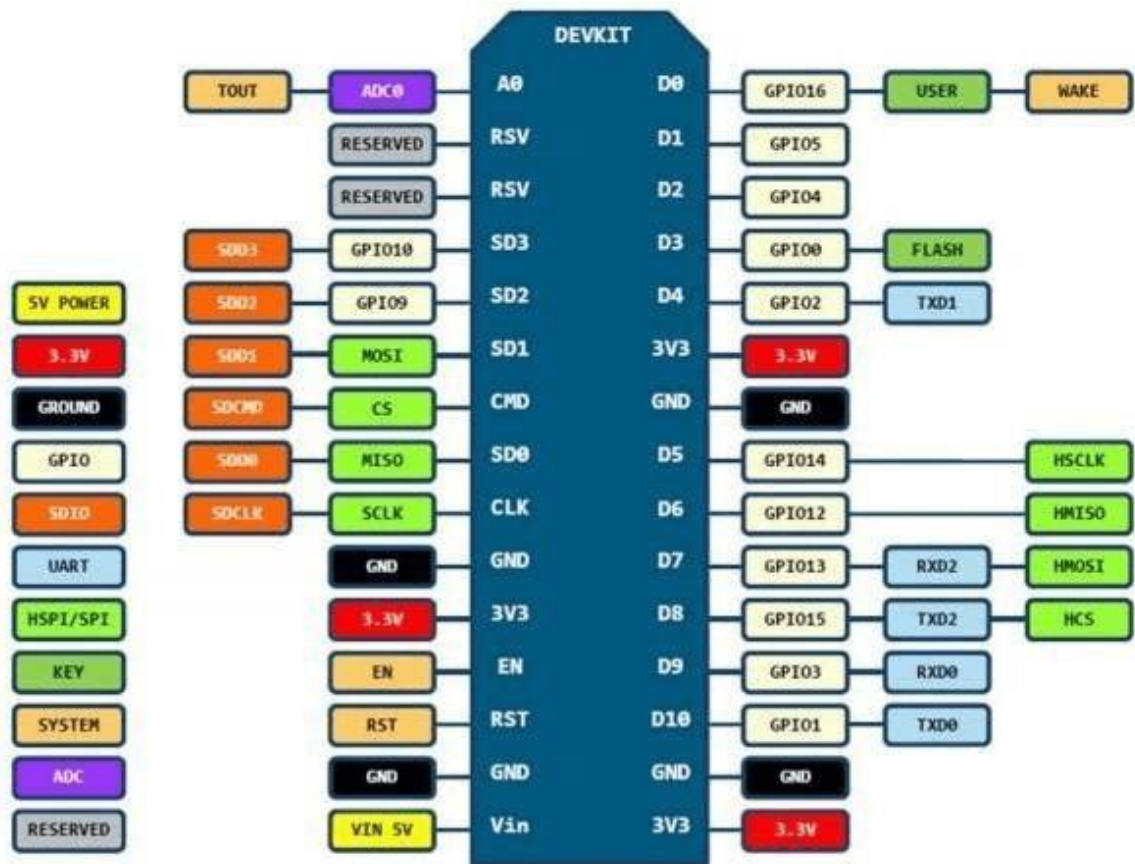


Figure 3.2 NodeMCU Pin Configuration

It provides unsurpassed ability to embed Wi-Fi capabilities within other systems, or to function as a standalone application, with the lowest cost, and minimal space requirement. ESP8266EX offers a complete and self-contained Wi-Fi networking solution; it can be used to host the application or to offload Wi-Fi networking functions from another application processor. When ESP8266EX hosts the application, it boots up directly from an external flash. It has integrated cache to improve the performance of the system in such applications. Alternately, serving as a Wi-Fi adapter, wireless internet access can be added to any micro controller-based design with simple connectivity. ESP8266EX is among the most integrated Wi-Fi chip in the industry it integrates the antenna switches, RF balun, power amplifier, low noise receive amplifier, filters, power management modules, it requires minimal external circuitry, and the entire solution, including front-end. ESP8266EX also

integrates an enhanced version of Ten silica's L106 Diamond series 32-bit processor, with on-chip SRAM, besides the Wi-Fi functionalities. ESP8266EX is often integrated with external sensors and other application specific devices through its GPIOs. For low-power operation, advance signal processing, and spur

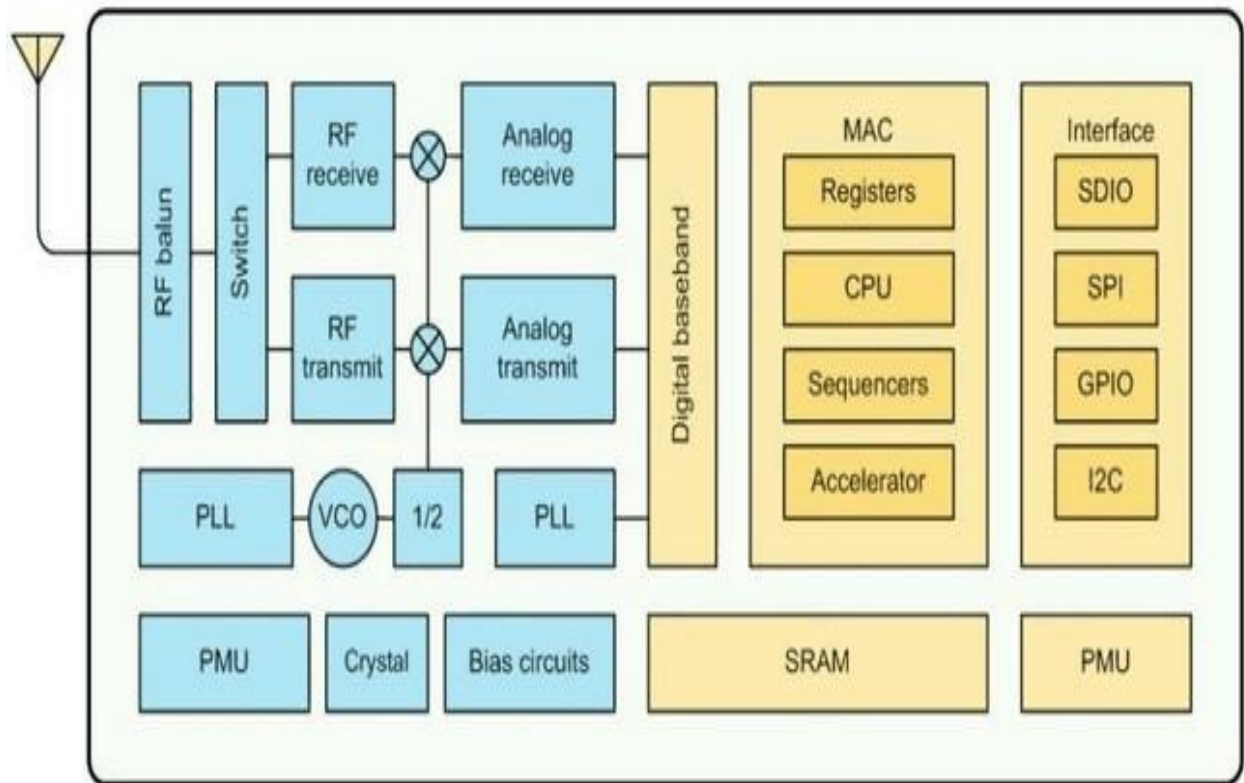


Figure 3.3 ESP-12E Architecture

cancellation and radio co-existence features for common cellular, Bluetooth, DDR, LVDS, LCD interference mitigation.

Features

- 802.11 b/g/n
- Integrated low power 32-bit MCU
- Integrated 10-bit ADC
- Integrated TCP/IP protocol stack
- Integrated TR switch, balun, LNA, power amplifier and matching network

- Integrated PLL, regulators, and power management units
- Supports antenna diversity
- Wi-Fi 2.4 GHz, support WPA/WPA2
- Support STA/AP/STA+AP operation modes
- Support Smart Link Function for both Android and iOS devices
- Support Smart Link Function for both Android and iOS devices
- SDIO 2.0, (H) SPI, UART, I2C, I2S, IRDA, PWM, GPIO
- STBC, 1x1 MIMO, 2x1 MIMO
- A-MPDU & A-MSDU aggregation and 0.4s guard interval
- Deep sleep power < 5uA
- Wake up and transmit packets in < 2ms
- Standby power consumption of < 1.0mW (DTIM3)
- +20dBm output power in 802.11b mode
- Operating temperature range -40C ~ 125C



Figure: 3.4 ESP-12E Pin Configuration

NO.	Pin Name	Function
1	RST	Reset the module
2	ADC	A/D Conversion result.Input voltage range 0-1v,scope:0-1024
3	EN	Chip enable pin.Active high
4	IO16	GPIO16; can be used to wake up the chipset from deep sleep mode.
5	IO14	GPIO14; HSPI_CLK
6	IO12	GPIO12; HSPI_MISO
7	IO13	GPIO13; HSPI_MOSI; UART0_CTS
8	VCC	3.3V power supply (VDD)
9	CS0	Chip selection
10	MISO	Salve output Main input
11	IO9	GPIO9
12	IO10	GPIO10
13	MOSI	Main output slave input
14	SCLK	Clock
15	GND	GND
16	IO15	GPIO15; MTDO; HSPICS; UART0_RTS
17	IO2	GPIO2; UART1_TXD
18	IO0	GPIO0
19	IO4	GPIO4
20	IO5	GPIO5
21	RXD	UART0_RXD; GPIO3
22	TXD	UART0_TXD; GPIO1

Figure: 3.5 Pin Description

Mode	GPIO15	GPIO0	GPIO2
UART	Low	Low	High
Flash Boot	Low	High	High

Figure: 3.6 Pin Mode

Receiver Sensitivity

Parameters	Min	Typical	Max	Unit
Input frequency	2412		2484	MHz
Input impedance		50		Ω
Input reflection			-10	dB
Output power of PA for 72.2Mbps	15.5	16.5	17.5	dBm
Output power of PA for 11b mode	19.5	20.5	21.5	dBm
Sensitivity				
DSSS, 1Mbps		-98		dBm
CCK, 11Mbps		-91		dBm
6Mbps (1/2 BPSK)		-93		dBm
54Mbps (3/4 64-QAM)		-75		dBm
HT20, MCS7 (65Mbps, 72.2Mbps)		-72		dBm
Adjacent Channel Rejection				
OFDM, 6Mbps		37		dB
OFDM, 54Mbps		21		dB
HT20, MCS0		37		dB
HT20, MCS7		20		dB

Figure: 3.7 Receiver Sensitivity

Schematic

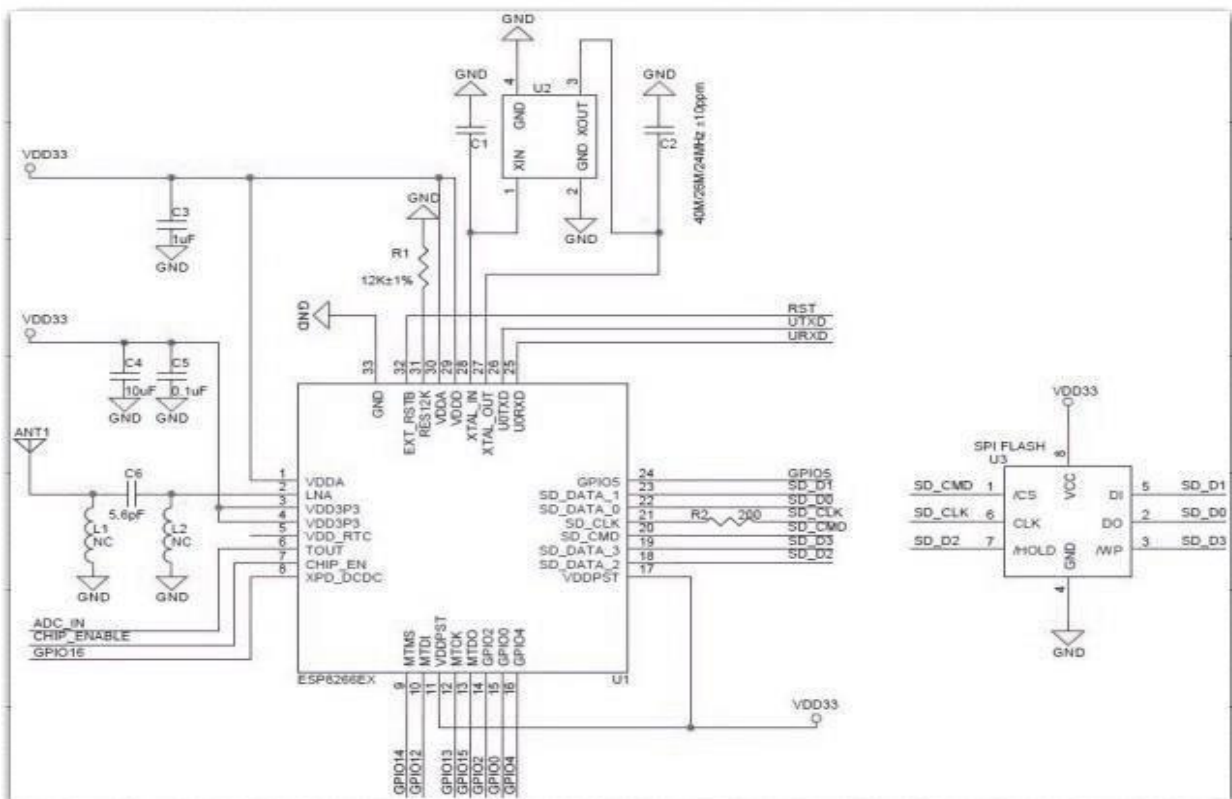


Figure: 3.8 Schematics of Esp-12e Wi-Fi Module

Functional Description

MCU

ESP8266EX is embedded with Ten silica L106 32-bit micro controller (MCU), which features extra low power consumption and 16-bit RSIC. The CPU clock speed is 80MHz. It can also reach a maximum value of 160MHz. ESP8266EX is often integrated with external sensors and other specific devices through its GPIOs; codes for such applications are provided in examples in the SDK.

Memory Organization

Internal SRAM and ROM

ESP8266EX Wi-Fi SoC is embedded with memory controller, including RAM and ROM. MCU can visit the memory units through iBus, dBus, and AHB interfaces. All memory units can be visited upon request, while a memory arbiter will decide the running sequence according to the time when these requests are received by the processor. According to our current version of SDK provided, SRAM space that is available to users is assigned as below:

- RAM size < 36kB when ESP8266EX is working under the station mode and is connected to the router, programmable space accessible to user in heap and data section is around 36kB.)
- There is no programmable ROM in the SoC, therefore, user program must be stored in an external SPI flash.

External SPI Flash

This module is mounted with a 4 MB external SPI flash to store user programs. If larger definable storage space is required, a SPI flash with larger memory size is preferred. Theoretically speaking, up to 16 MB memory capacity can be supported. Suggested SPI Flash memory capacity:

- OTA is disabled: the minimum flash memory that can be supported is 512 kB.
- OTA is enabled: the minimum flash memory that can be supported is 1 MB. Several SPI modes can be supported, including Standard SPI, Dual SPI, and Quad SPI.

Crystal

Currently, the frequency of crystal oscillators supported includes 40MHz, 26MHz and 24MHz. The accuracy of crystal oscillators applied should be $\pm 10\text{PPM}$, and the operating temperature range should be between -20°C and 85°C . When using the downloading tools, please remember to select the right crystal oscillator type. In circuit design, capacitors C1 and C2, which are connected to the earth, are added to the input and output terminals of the crystal oscillator respectively. The values of the two capacitors can be flexible, ranging from 6pF to 22pF, however, the specific capacitive values of C1 and C2 depend on further testing and adjustment on the overall performance of the whole circuit. Normally, the capacitive values of C1 and C2 are within 10pF if the crystal oscillator frequency is 26MHz, while the values of C1 and C2 are 10pF

Interfaces

Interface	Pin Name	Description
HSPI	IO12(MISO) IO13(MOSI) IO14(CLK) IO15(CS)	SPI Flash 2, display screen, and MCU can be connected using HSPI interface.
PWM	IO12(R) IO15(G) IO13(B)	Currently the PWM interface has four channels, but users can extend the channels according to their own needs. PWM interface can be used to control LED lights, buzzers, relays, electronic machines, and so on.
IR Remote Control	IO14(IR_T) IO5(IR_R)	The functionality of Infrared remote control interface can be implemented via software programming. NEC coding, modulation, and demodulation are used by this interface. The frequency of modulated carrier signal is 38KHz.
ADC	TOUT	ESP8266EX integrates a 10-bit analog ADC. It can be used to test the power-supply voltage of VDD3P3 (Pin3 and Pin4) and the input power voltage of TOUT (Pin 6). However, these two functions cannot be used simultaneously. This interface is typically used in sensor products.
I2C	IO14(SCL) IO2(SDA)	I2C interface can be used to connect external sensor products and display screens, etc.
Interface	Pin Name	Description
UART	UART0: TXD (U0TXD) RXD (U0RXD) IO15 (RTS) IO13 (CTS) UART1: IO2(TXD)	Devices with UART interfaces can be connected with the module. Downloading: U0TXD+U0RXD or GPIO2+U0RXD Communicating: UART0: U0TXD, U0RXD, MTDO (U0RTS), MTCK (U0CTS). Debugging: UART1_TXD (GPIO2) can be used to print debugging information. By default, UART0 will output some printed information when the device is powered on and is booting up. If this issue exerts influence on some specific applications, users can exchange the inner pins of UART when initializing, that is to say, exchange U0TXD, U0RXD with U0RTS, U0CTS.
I2S	I2S Input: IO12 (I2SI_DATA); IO13 (I2SI_BCK); IO14 (I2SI_WS); I2S Output: IO15 (I2SO_BCK); IO3 (I2SO_DATA); IO2 (I2SO_WS).	I2S interface is mainly used for collecting, processing, and transmission of audio data.

Figure: 3.9 Interfaces

NodeMCU GPIO for Lua

The GPIO(General Purpose Input/Output) allows us to access to pins of ESP8266 , all the pins of ESP8266 accessed using the command GPIO, all the access is based on the I/O index number on the NodeMCU dev kits, not the internal GPIO pin, for example, the pin ‘D7’ on the NodeMCU dev kit is mapped to the internal GPIO pin 13, if you want to turn ‘High’ or ‘Low’ that particular pin you need to called the pin number ‘7’, not the internal GPIO of the pin. When you are programming with generic ESP8266 this confusion will arise which pin needs to be called during programming, if you are using NodeMCU devkit, it has come prepared for working with Lua interpreter which can easily program by looking the pin names associated on the Lua board. If you are using generic ESP8266 device or any other vendor boards please refer to the table below to know which IO index is associated to the internal GPIO of ESP8266.

Nodemcu dev kit	ESP8266 Pin	Nodemcu dev kit	ESP8266 Pin
D0	GPIO16	D7	GPIO13
D1	GPIO5	D8	GPIO15
D2	GPIO4	D9	GPIO3
D3	GPIO0	D10	GPIO1
D4	GPIO2	D11	GPIO9
D5	GPIO14	D12	GPIO10
D6	GPIO12		

Figure: 3.10 NodeMCU Gpio for Lua

D0 or GPIO16 can be used only as a read and write pin, no other options like PWM/I2C are supported by this pin. In our example in chapter 5 on blinking the blue LED, the blue LED in connected to GPIO2, it is defined as Pin4 (D4) in Lua script.

Single-Chip Usb-To-Uart Bridge

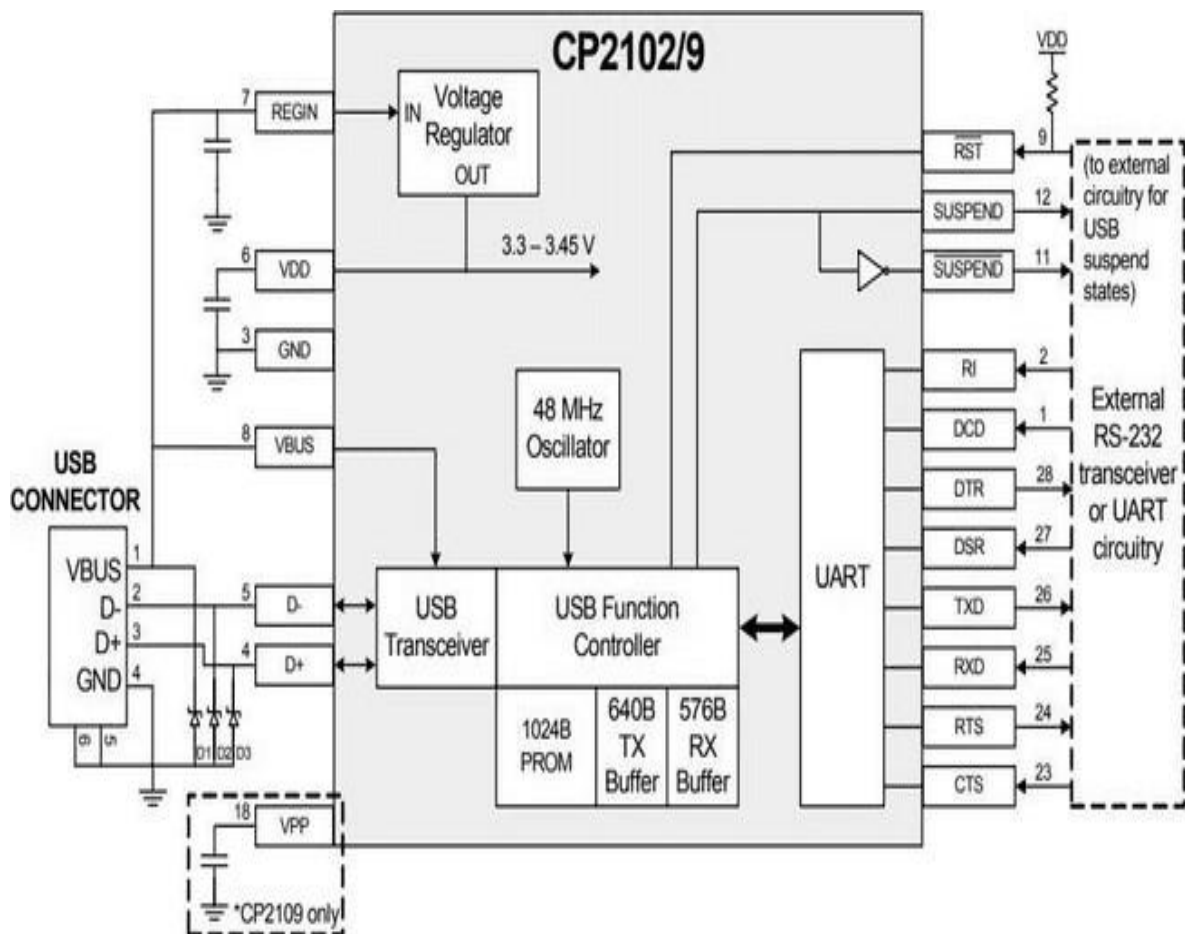


Figure: 3.11 Single-Chip Usb-To-Uart

The CP2102/9 is a highly-integrated USB-to-UART Bridge Controller providing a simple solution for updating RS232 designs to USB using a minimum of components and PCB space. The CP2102/9 includes a USB 2.0 full speed function controller, USB transceiver, oscillator, EEPROM or EPROM, and asynchronous serial data bus (UART) with full modem control signals in a compact 5 x 5 mm QFN-28 package. No other external USB components are required. The on-chip programmable ROM may be used to customize the USB Vendor ID, Product ID, Product Description String, Power Descriptor, Device, Release Number, and Device Serial Number as desired for OEM applications. The programmable ROM is programmed on-board via the USB, allowing the programming step to be easily integrated into the product manufacturing and testing process. Royalty-free

Virtual COM Port (VCP) device drivers provided by Silicon Laboratories allow a CP2102/9-based product to appear as a COM port to PC applications. The CP2102/9 UART interface implements all RS-232 signals, including control and handshaking signals, so existing system firmware does not need to be modified. In many existing RS-232 designs, all that is required to update the design from RS-232 to USB is to replace the RS232 level-translator with the CP2102/9. Direct access driver support is available through the Silicon Laboratories Subpress driver set.

3.1.4 Liquid Crystal Display (LCD)

LCD screen is an electronic display module and find a wide range of applications. A 16x2 LCD display is very basic module and is very commonly used in various devices and circuits. These modules are preferred over seven segments and other multi segment LEDs. The reasons being: LCDs are economical; easily programmable; have no limitation of displaying special & even custom characters (unlike in seven segments), animations and so on. A 16x2 LCD means it can display 16 characters per line and there are 2 such lines. In this LCD each character is displayed in 5x7 pixel matrix. This LCD has two registers, namely, Command and Data. The command register stores the command instructions given to the LCD. A command is an instruction given to LCD to do a predefined task like initializing it, clearing its screen, setting the cursor position, controlling display etc. The data register stores the data to be displayed on the LCD. The data is the ASCII value of the character to be displayed on the LCD.

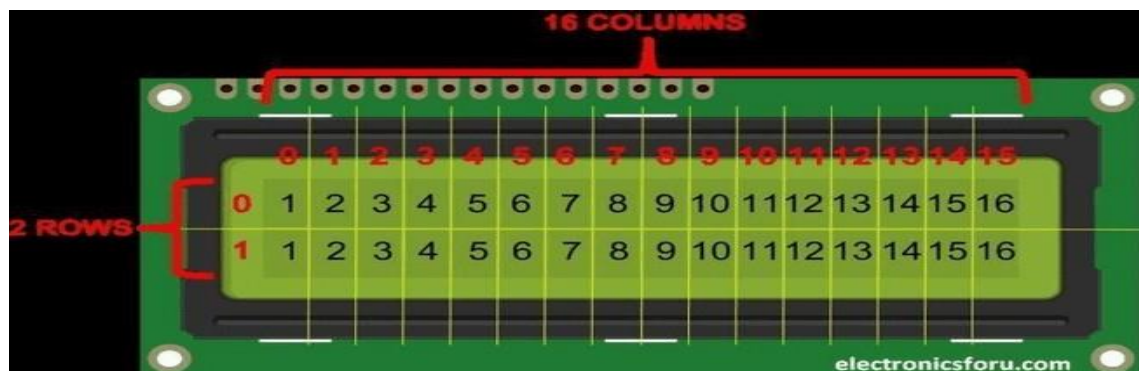


Figure: 3.12 Liquid Crystal Display

We come across LCD displays everywhere around us. Computers, calculators, television sets, mobile phones, digital watches use some kind of display to display the time. An LCD is an electronic display module which uses liquid crystal to produce a visible image. The 16×2 LCD display is a very basic module commonly used in projects. The 16×2 translates to a display 16 characters per line in 2 such lines. In this LCD each character is displayed in a 5×7-pixel matrix.

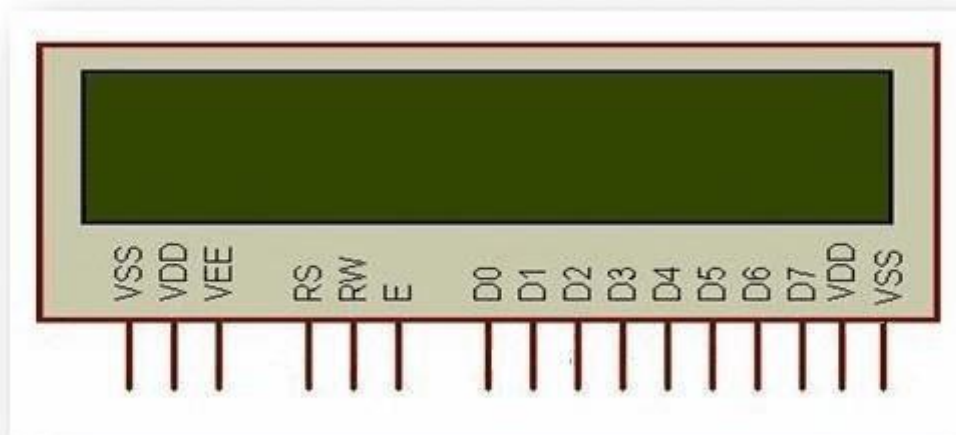


Figure: 3.13 16x2 Lcd Pinout Diagram

LCD function

PIN NO.	FUNCTION	NAME
1	Ground (0V)	Ground
2	Supply voltage; 5V (4.7V – 5.3V)	VCC
3	Contrast adjustment; the best way is to use a variable resistor such as a potentiometer. The output of the potentiometer is connected to this pin. Rotate the potentiometer knob forward and backwards to adjust the LCD contrast.	Vo / VEE

4	Selects command register when low, and data register when high	RS (Register Select)
5	Low to write to the register; High to read from the register	Read/write
6	Sends data to data pins when a high to low pulse is given; Extra voltage push is required to execute the instruction and EN (enable) signal is used for this purpose. Usually, we make it en=0 and when we want to execute the instruction, we make it high en=1 for some milliseconds. After this we again make it ground that is, en=0.	Enable
7	8-bit data pins	DB0
8		DB1
9		DB2
10		DB3
11		DB4
12		DB6
13		DB7
14	Backlight VCC (5V)	Led+
15	Backlight Ground (0V)	Led-

Table 3.1 Lcd Function

Rs (Register Select)

A 16X2 LCD has two registers, namely, command and data. The register select is used to switch from one register to other. RS=0 for command register, whereas RS=1 for data register.

Command Register

The command register stores the command instructions given to the LCD. A command is an instruction given to LCD to do a predefined task like initializing it, clearing its screen, setting the cursor position, controlling display etc. Processing for commands happens in the command register.

Data Register

The data register stores the data to be displayed on the LCD. The data is the ASCII value of the character to be displayed on the LCD. When we send data to LCD it goes to the data register and is processed there. When RS=1, data register is selected

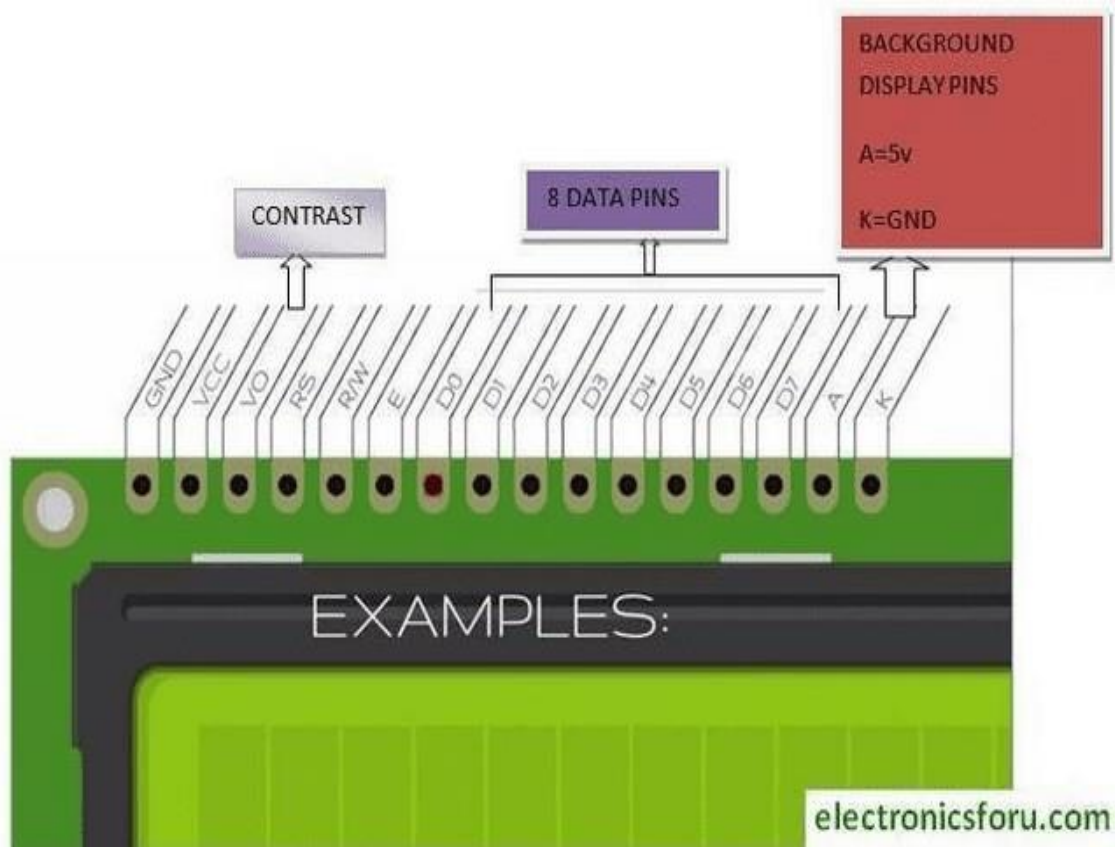


Figure: 3.14 Data Register

IMPORTANT COMMAND CODES FOR LCD

SR.NO.	HEX CODE	COMMAND TO LCD INSTRUCTION REGISTER
1	01	Clear display screen
2	02	Return home

3	04	Decrement cursor (shift cursor to left)
4	06	Increment cursor (shift cursor to right)
5	05	Shift display right
6	07	Shift display left
7	08	Display off, cursor off
8	0A	Display off, cursor on
9	0C	Display on, cursor off
10	0E	Display on, cursor blinking
11	0F	Display on, cursor blinking
12	10	Shift cursor position to left
13	14	Shift cursor position to right
14	18	Shift the entire display to the left
15	1C	Shift the entire display to the right
16	80	Force cursor to beginning (1st line)
17	C0	Force cursor to beginning (2nd line)
18	38	2 lines and 5×7 matrix

Table 3.2 Command Codes for Lcd

Displaying Custom Characters On 16x2 Lcd

Generating custom characters on LCD is not very hard. It requires the knowledge about custom generated random-access memory (CG-RAM) of LCD and the LCD chip controller. Most LCDs contain Hitachi HD4478 controller. CG-RAM is the main component in making custom characters. It stores the custom characters once declared in the code. CG-RAM size is 64 bytes providing the option of creating eight characters at a time. Each character is eight bytes in size. CG-RAM address starts from 0x40 (Hexadecimal) or 64 in decimal. We can generate custom characters at these addresses. Once we generate our characters at these addresses, now we can print them on the LCD at any time by just sending simple commands to the LCD. Character addresses and printing commands are below. In the table above you can see starting addresses for each character with their printing commands. The first character is generated at address 0x40 to 0x47 and is printed on LCD by just sending simple command 0 to the LCD.

CG-RAM Characters	CG-RAM Address (Hexadecimal)	Commands to display Generated Characters
1 st Character	0x40	0
2 nd Character	0x48	1
3 rd Character	0x56	2
4 th Character	0x64	3
5 th Character	0x72	4
6 th Character	0x80	5
7 th Character	0x88	6
8 th Character	0x96	7

electronicsforu.com

Figure: 3.15 Displaying Custom Characters On 16x2 Lcd

3.1.5 I2c LCD Adapter

At the heart of the adapter is an 8-bit I/O expander chip – PCF8574. This chip converts the I2C data from an Arduino into the parallel data required for an LCD display.



Figure: 3.16 Pcf8574

The board also comes with a small trim pot to make fine adjustments to the display's contrast.

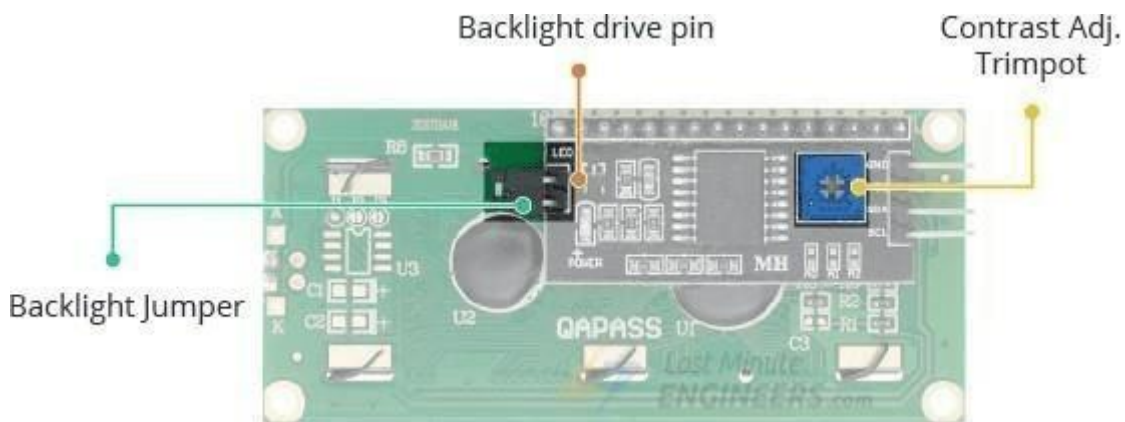


Figure: 3.17 Backlight Drive Pin

In addition, there is a jumper on the board that supplies power to the backlight. To control the intensity of the backlight, you can remove the jumper and apply external voltage to the header pin that is marked 'LED.'

I2C Address of LCD

If you are using multiple devices on the same I2C bus, you may need to set a different I2C address for the LCD adapter so that it does not conflict with another I2C device.

To do this, the adapter has three solder jumpers (A0, A1 and A2) or solder pads.



Figure: 3.18 Solder Pads

Each of these is used to hardcode the address. If a jumper is shorted with a blob of solder, it sets the address. An important point here is that several companies the same PCF8574 chip, Texas Instruments and NXP Semiconductors, to name a few. And the I2C address of your LCD depends on the chip manufacturer.

If your LCD has Texas Instruments' PCF8574 chip:

According to the [Texas Instruments' datasheet](#), the three address selection bits (A0, A1 and A2) are placed at the end of the 7-bit I2C address register.

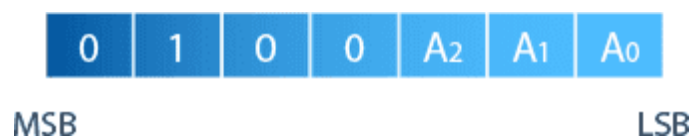
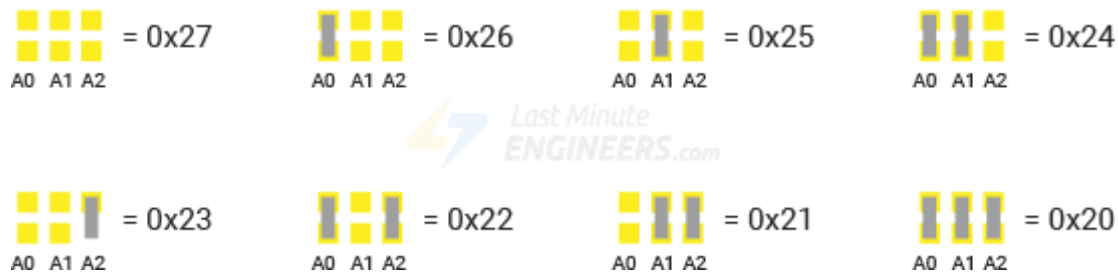


Figure: 3.19 Texas Instruments

Since there are 3 address inputs, which can take 2 states, either HIGH/LOW, we can therefore create 8 (2^3) different combinations (addresses). By default, all 3 address inputs are pulled HIGH using onboard pullups, giving the PCF8574 a default I2C address of 0100111_{Binary} or $0x27_{\text{Hex}}$. By shorting the solder jumpers, the address inputs are pulled LOW. If you were to short all three jumpers, the address would be $0x20$. The range of all possible addresses spans from $0x20$ to $0x27$. Please see the illustration below.



If your LCD has NXP's PCF8574 chip:

According to the [NXP Semiconductors datasheet](#), the three address selection bits (A0, A1 and A2) are also placed at the end of the 7-bit I2C address register. But the other bits in the address register are different.

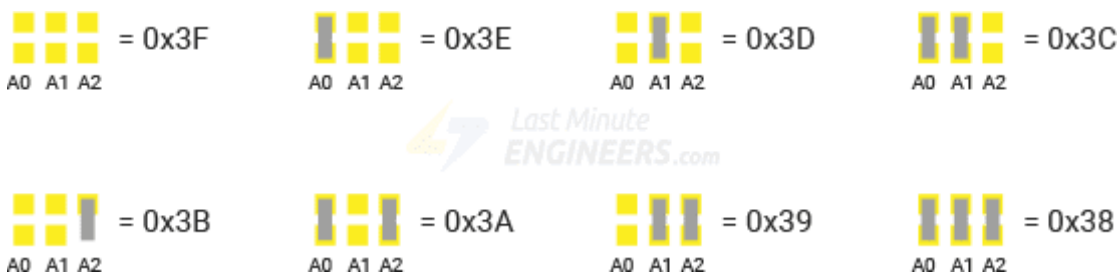


Figure: 3.20 NXP Semiconductors

Since there are 3 address inputs, which can take 2 states, either HIGH/LOW, we can therefore create 8 (2^3) different combinations (addresses).

By default, all 3 address inputs are pulled HIGH using onboard pullups, giving the PCF8574 a default I2C address of 0111111_{Binary} or 0x3F_{Hex}.

By shorting the solder jumpers, the address inputs are pulled LOW. If you were to short all three jumpers, the address would be 0x38. The range of all possible addresses spans from 0x38 to 0x3F. Please see the illustration below.



I2C LCD display Pinout

An I2C LCD has only 4 pins that connect it to the outside world.

The connections are as follows:

GND is a ground pin. Connect it to the ground of the Arduino.

VCC supplies power to the module and LCD. Connect it to the Arduino's 5V output or an external 5V power supply.

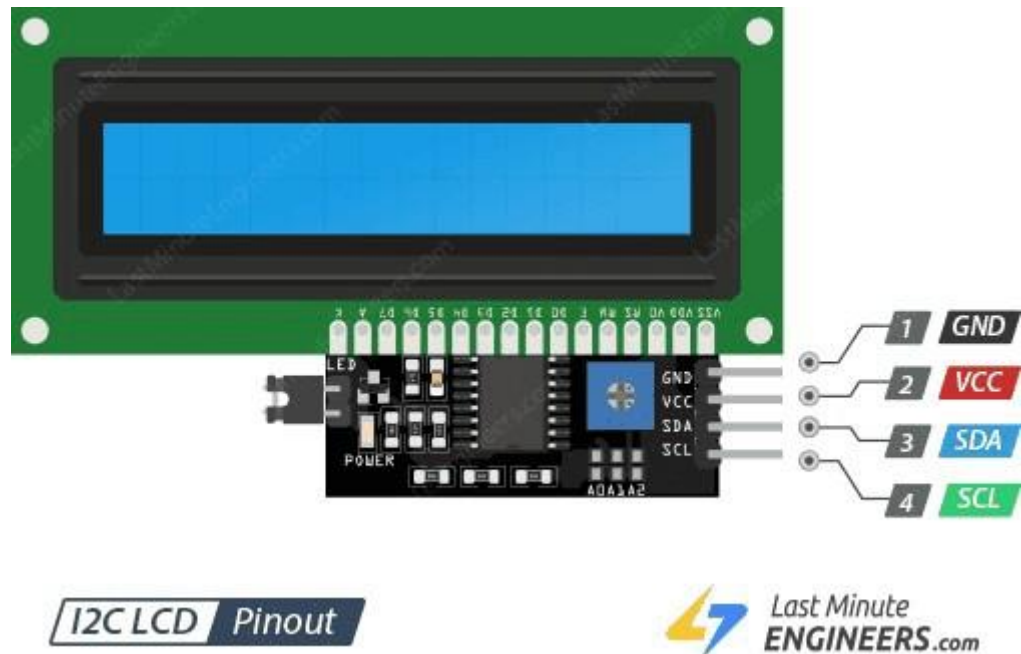


Figure: 3.21 IC2 LCD PINOUT

SDA is the I2C data pin. Connect it to the Arduino's I2C data pin.

SCL is the I2C clock pin. Connect it to the Arduino's I2C clock pin.

Hooking up an Arduino Uno to an I2C LCD display. Start by connecting the VCC pin to the 5V output on the Arduino and GND to ground. On Arduino boards with the R3 layout, the SDA (data line) and SCL (clock line) are on the pin headers close to the AREF pin. They are also known as A5 (SCL) and A4 (SDA).

If you are using a different Arduino board, please refer to the table below.

	SCL	SDA
Arduino Uno	A5	A4
Arduino Nano	A5	A4
Arduino Mega	21	20
Leonardo/Micro	3	2

The following diagram shows you how to wire everything up.

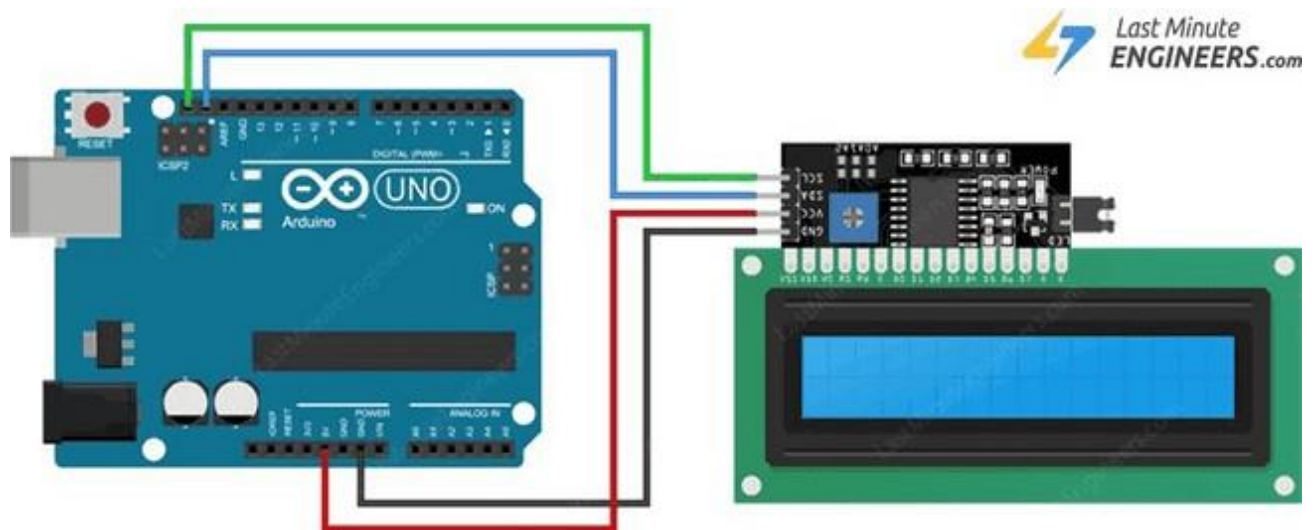


Figure: 3.22 Lcd Contrast

After wiring up the LCD you will need to adjust the contrast of the display. On the I2C module you will find a potentiometer that you can rotate with a small screwdriver. Plug in the Arduino's USB connector to power the LCD. You will see the backlight lit up. Now as you turn the knob on the potentiometer, you will start to see the first row of rectangles.

If that happens, Congratulations! Your LCD is working fine.

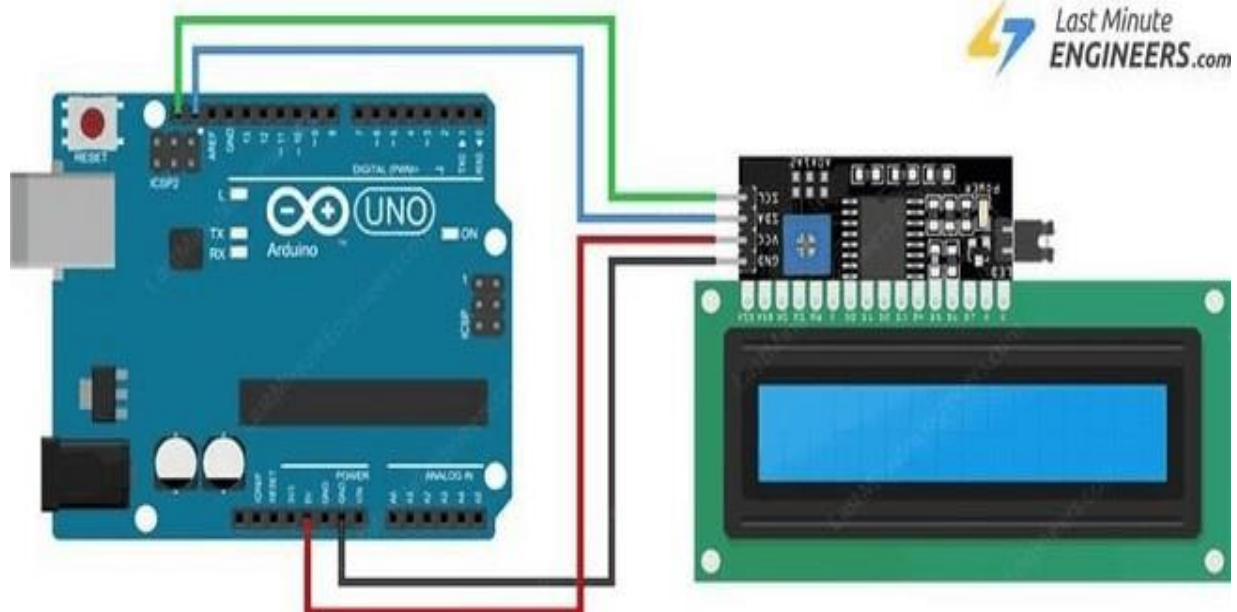


Figure: 3.23 Liquidcrystal_I2c

Once this is done, we can start programming the LCD. Library Installation
To drive an I2C LCD you must first install a library called [LiquidCrystal_I2C](#). This library is an enhanced version of the Liquid Crystal library that comes with your Arduino IDE. To install the library, navigate to Sketch > Include Libraries > Manage Libraries... Wait for Library Manager to download the library index and update the list of installed libraries.

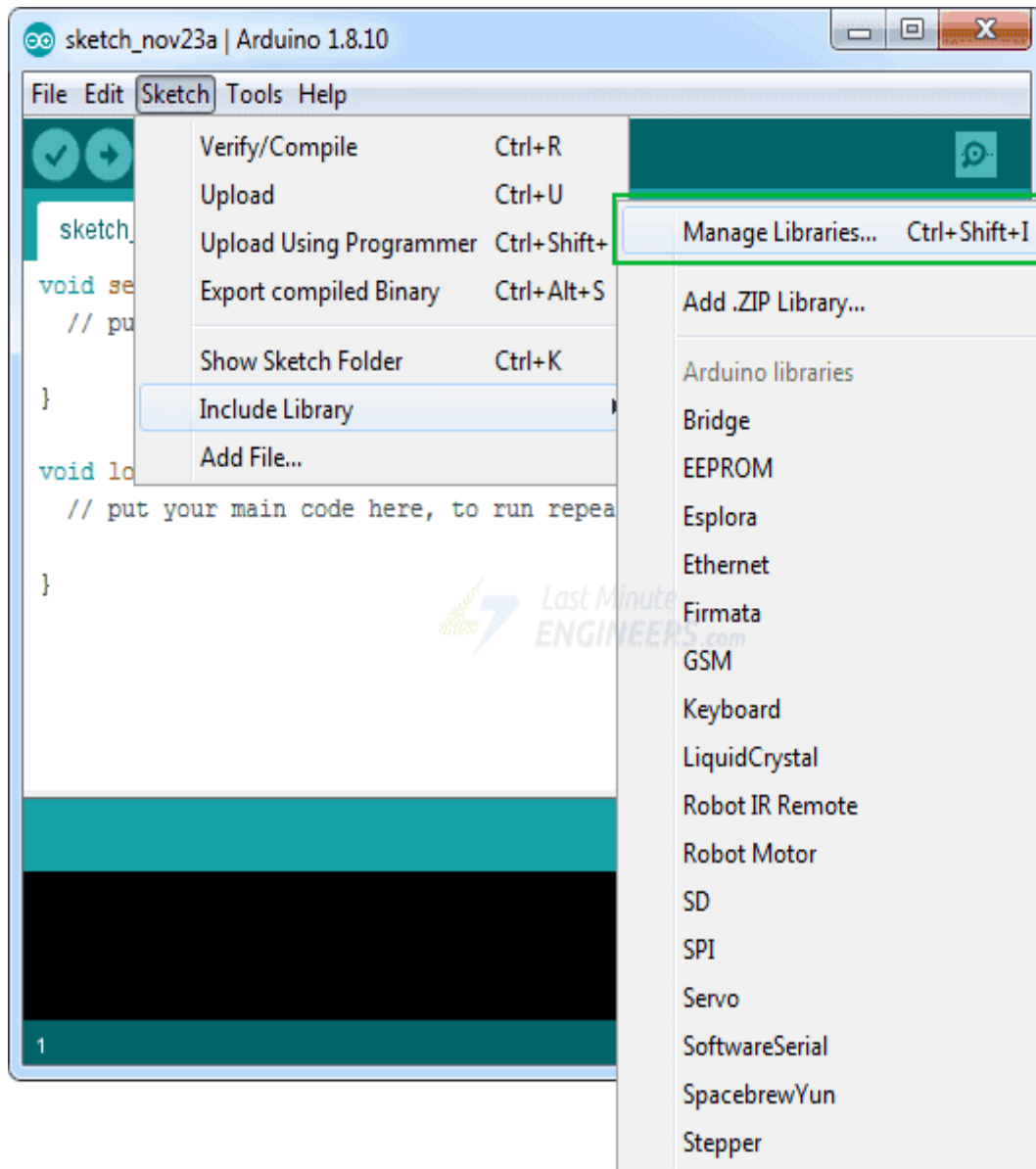


Figure: 3.24 Sketch_Nov23a

Filter your search by typing 'liquid crystal '. There should be some entries. Look for the Liquid Crystal I2C library by Frank de Bra bander. Click on that entry, and then select Install.

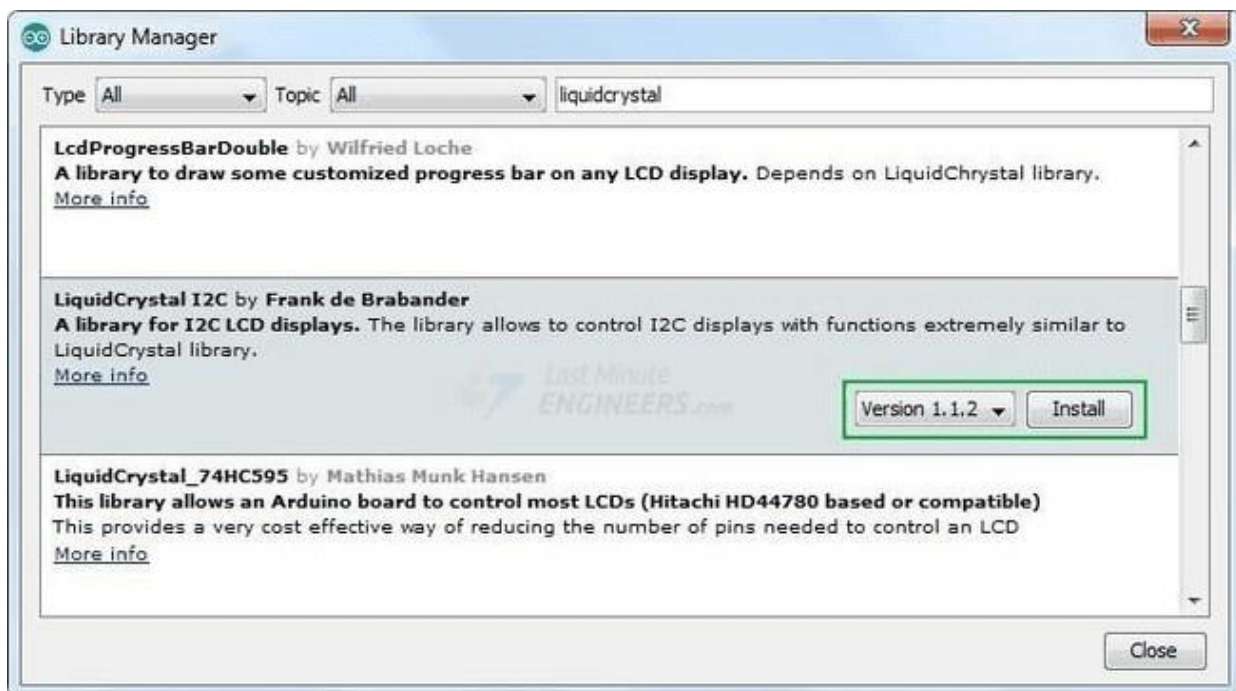


Figure: 3.25 Library Manager

Determining the I2C Address

The I2C address of your LCD depends on the manufacturer, as mentioned earlier. If your LCD has a Texas Instruments' PCF8574 chip, its default I2C address is 0x27_{Hex}. If your LCD has NXP Semiconductors' PCF8574 chip, its default I2C address is 0x3F_{Hex}. So, your LCD probably has I2C address 0x27_{Hex} or 0x3F_{Hex}. However, it is recommended that you find out the actual I2C address of the LCD before using it. Luckily there is an easy way to do this, thanks to the [Nick Gammon](#). Nick wrote a simple I2C scanner sketch that scans your I2C bus and returns the address of each I2C device it finds. Load this sketch into your Arduino and then open your Serial Monitor. You will see the I2C address of your I2C LCD display. The following test sketch will print 'Hello World!' on the first line of the LCD and 'LCD Tutorial' on the second line. But, before you proceed to upload the sketch, you need to make a small change to make it work for you. You must pass the I2C address of your LCD and the dimensions of the display to the constructor of the LiquidCrystal_I2C class.

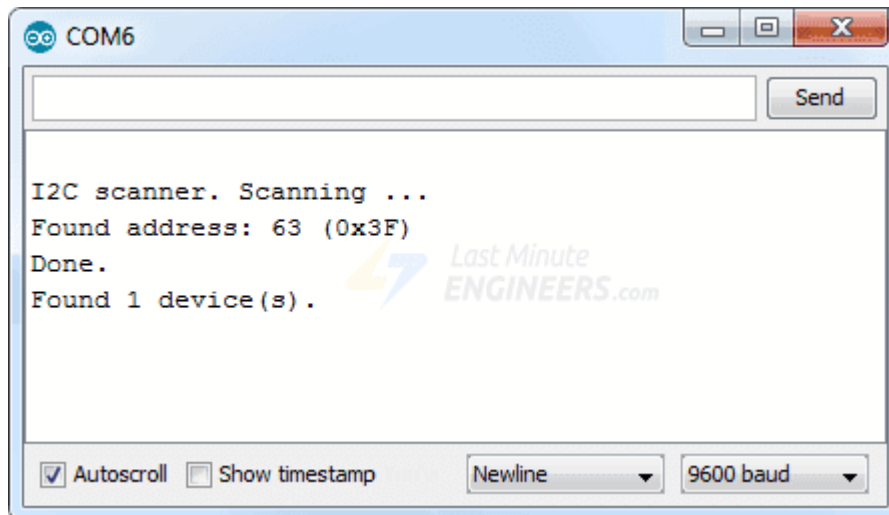


Figure: 3.26 COM6

Please note this address. You will need it in later examples. Basic Arduino Sketch – Hello World. If you are using a 16×2-character LCD, pass the 16 and 2. If you are using a 20×4 LCD, pass 20 and 4. You got the point! If all goes well, you should see something like this on the display.



Figure: 3.27 LCD TUTORIAL

Code Explanation:

The sketch begins by including the LiquidCrystal_I2C library.

First, an object of `LiquidCrystal_I2C` class is created. This object takes three parameters `LiquidCrystal_I2C (address, columns, rows)`. This is where you need to enter the address you found earlier, and the dimensions of the display.

```
LiquidCrystal_I2C lcd(0x3F,16,2);
```

Once you have declared a LiquidCrystal_I2C object, you can access the object methods specific to the LCD.

In 'setup' we call three functions. The first function is `init ()`. It initializes the LCD object. The second function is `clear ()`. This clears the LCD screen and moves the cursor to the top left corner.

```
LCD. init ();  
LCD. clear ();  
LCD. backlight ();
```

After that we set the cursor position to the third column of the first row by calling the function `lcd.setCursor (2, 0)`. The cursor position specifies the location where you want the new text to be displayed on the LCD. The upper left corner is assumed to be col=0, row=0.

```
lcd.setCursor (2,0);
```

Next, the string 'Hello World!' is printed by calling the `print ()` function.

```
LCD. Print ("Hello world!");
```

Similarly, the next two lines of code set the cursor position to the third column of the second row, and print 'LCD Tutorial' on the LCD.

```
lcd.setCursor (2,1);
```

```
LCD. Print ("LCD Tutorial");
```

`LCD.Home ()` function is used to position the cursor in the upper-left of the LCD without clearing the display.

`LCD.Blink ()` function displays a blinking block of 5×8 pixels at the position at which the next character is to be written.

`LCD.Cursor ()` displays an underscore (line) at the position at which the next character is to be written.

- `lcd.noBlink()` function turns off the blinking LCD cursor.
- `lcd.noCursor()` hides the LCD cursor.

`lcd.scrollDisplayRight()` function scrolls the contents of the display one space to the right. If you want the text to scroll continuously, you must use this function inside a for loop.

`lcd.scrollDisplayLeft()` function scrolls the contents of the display one space to the left. Like above function, use this inside a for loop for continuous scrolling. Create and Display Custom Characters. If you find the characters on the display dull and boring, you can create your own custom characters (glyphs) and symbols for your LCD. They are extremely useful when you want to display a character that is not part of the [standard ASCII character set](#). As discussed earlier in this tutorial a character is made up of a 5×8-pixel matrix, so you need to define your custom character within that matrix. You can use the `createChar` function to define a character. To use you first set up an array of 8 bytes.

3.1.6 DS18B20 Temperature Sensor

The DS18B20 temperature sensor is a one-wire digital temperature sensor. This means that it just requires one data line (and GND) to communicate with the Arduino. It can be powered by an external power supply or it can derive power from the data line (called “parasite mode”), which eliminates the need for an external power supply.

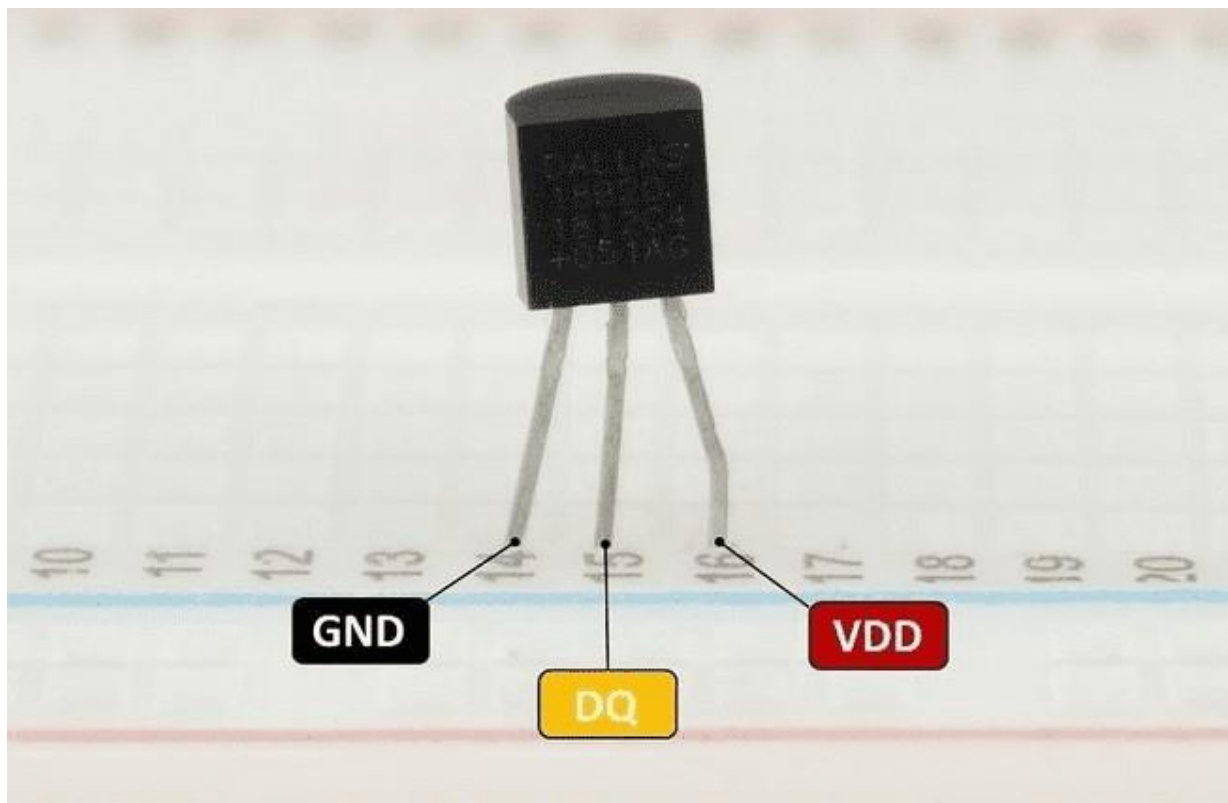


Figure: 3.28 DS18B20 Temperature Sensor

The following table shows the DS18B20 sensor to your Arduino board:

	Arduino
GND	GND
DQ	Any digital pin (with 4.7k Ohm pull-up resistor)
VDD	5V (normal mode) or GND (parasite mode)

Table 3.3 DS18B20 Sensor to Arduino Board

Each DS18B20 temperature sensor has a unique 64-bit serial code. This allows you to wire multiple sensors to the same data wire. So, you can get temperature from multiple sensors using just one Arduino digital pin.

The DS18B20 temperature sensor is also available in [waterproof version](#).



Figure: 3.29 DS18B20 Waterproof Version

Here is a summary of the most relevant specs of the DS18B20 temperature sensor:

- Communicates over one-wire bus communication
- Power supply range: 3.0V to 5.5V
- Operating temperature range: -55°C to +125°C
- Accuracy +/-0.5 °C (between the range -10°C to 85°C) For more information

consult the [DS18B20 datasheet](#).

Parts Required

To show you how the sensor works, we will build a simple example that reads the temperature from the DS18B20 sensor with the Arduino and displays the values on the Arduino Serial Monitor.

Here is a list of parts you need to complete this tutorial

- [Arduino UNO](#) – read [Best Arduino Starter Kits](#)
- [DS18B20 temperature sensor](#) (one or multiple sensors) – [waterproof version](#)
- [4.7k Ohm resistor](#)

- [Breadboard](#)

You can use the preceding links or go directly to [MakerAdvisor.com/tools](https://makeradvisor.com/tools) to find all the parts for your projects at the best price

Schematic

The sensor can operate in two modes:

- **Normal mode:** 3-wire connection is needed. You provide power to the VDD pin. Here is the schematic you need to follow:

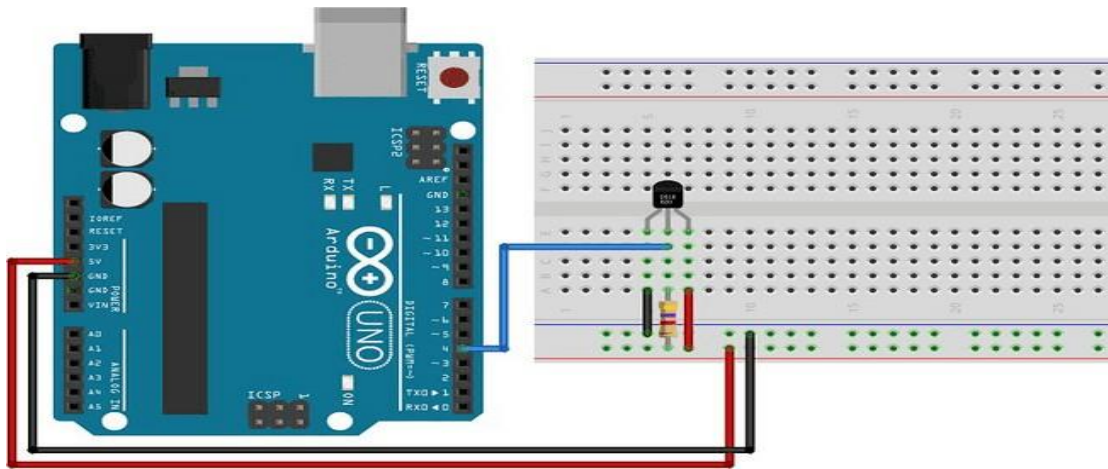


Figure: 3.30 Normal Mode

- **Parasite mode:** You only need data and GND. The sensor derives its power from the data line. Here is the schematic you need to follow:

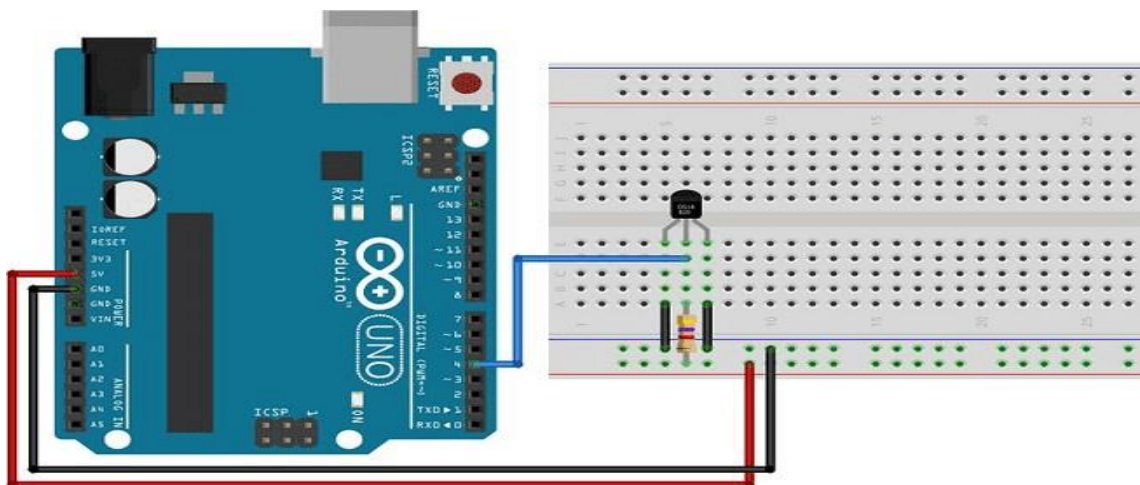


Figure: 3.31 Parasite Mode

You can read the temperature of more than one sensor at the same time using just one Arduino digital pin. For that, you just need to wire together all the sensors' data pins to an Arduino digital pin.

Upload Code – Single DS18B20

To interface with the DS18B20 temperature sensor, you need to install the One Wire [library by Paul Stoffregen](#) and the [Dallas Temperature library](#). Follow the next steps to install those libraries.

Installing Libraries

1. Open your Arduino IDE and go to **Sketch > Include Library > Manage Libraries**. The Library Manager should open.
2. Type “**One Wire**” in the search box and install the One Wire library.

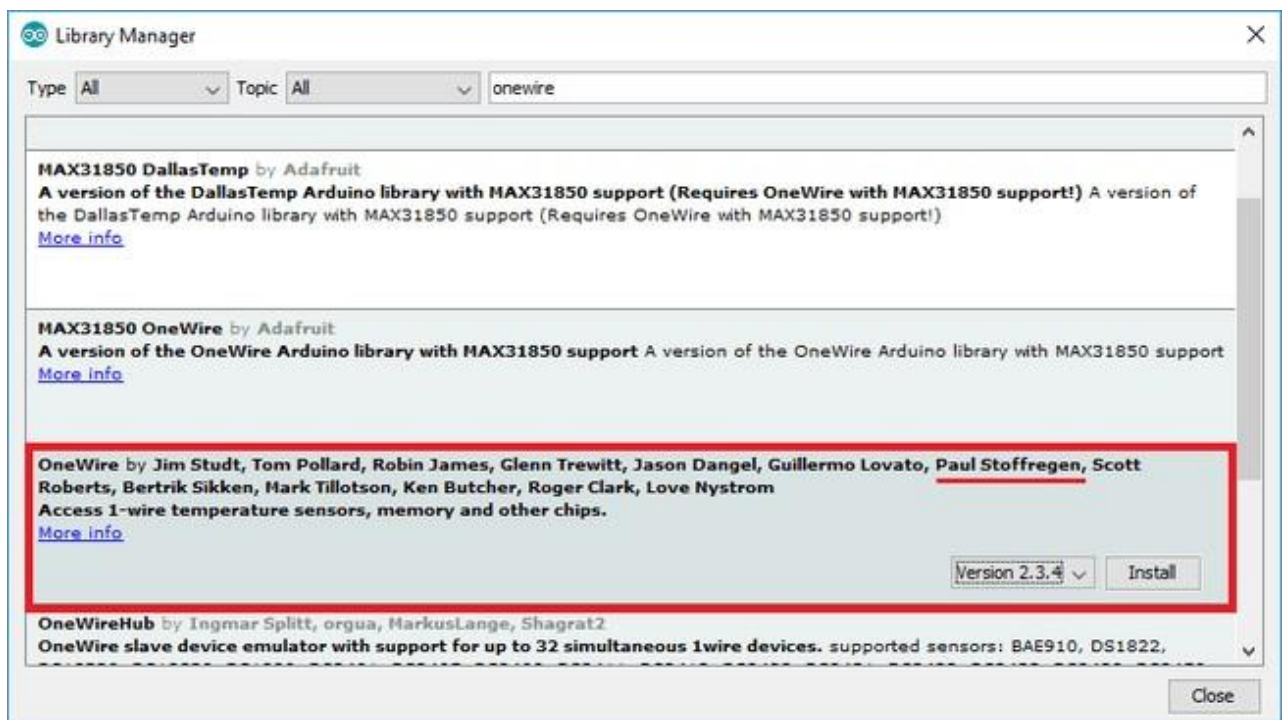


Figure: 3.32 Installing Libraries

3. Then, search for “**Dallas**” and install the Dallas Temperature library by Miles Burton.

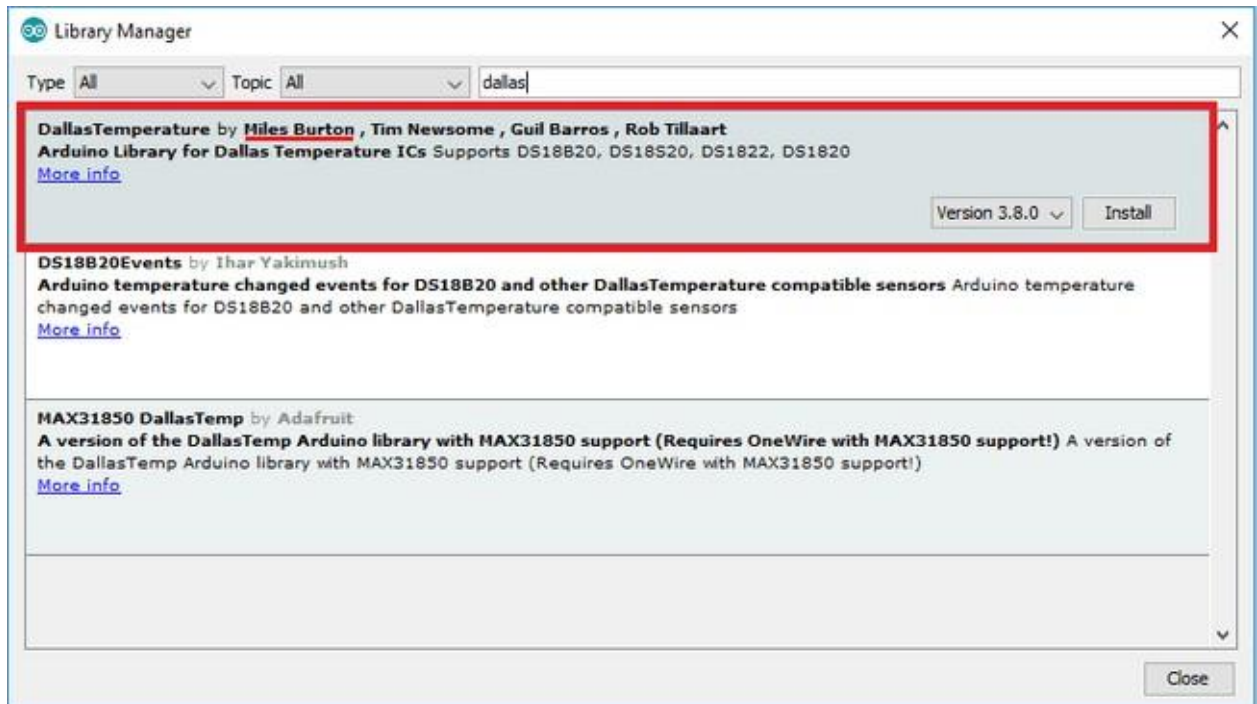


Figure: 3.33 Installing Dallas Temperature Library

After installing the needed libraries, upload the following code to your Arduino board. This sketch is based on an example from the Dallas Temperature library. There are many ways to get the temperature from DS18B20 temperature sensors. If you are using just one single sensor, this is one of the easiest and simplest ways.

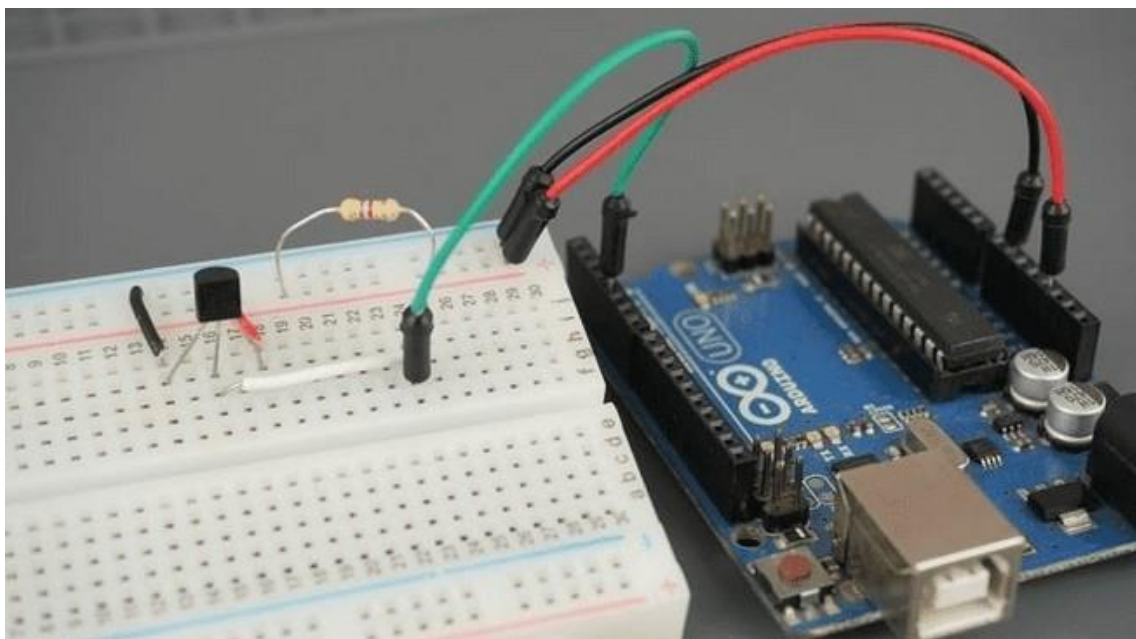


Figure: 3.34 Arduino Ide Serial Monitor

Demonstration

After uploading the code, open the Arduino IDE Serial Monitor at a 9600 baud rate. You should get the temperature displayed in both Celsius and Fahrenheit:

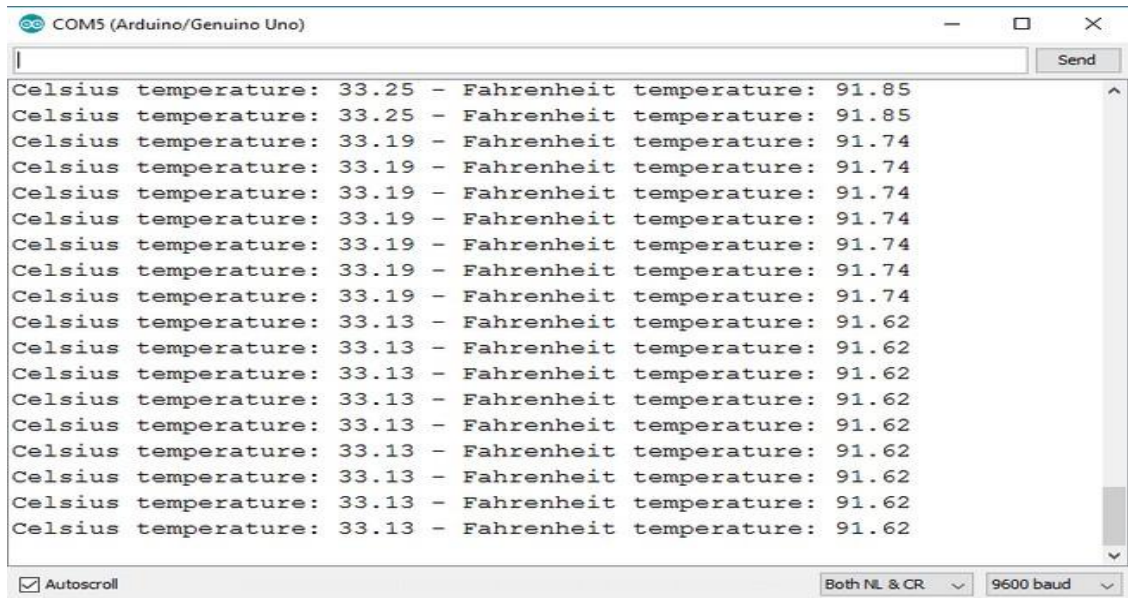


Figure: 3.35 Temperature Displayed

Getting Temperature from Multiple DS18B20 Sensors

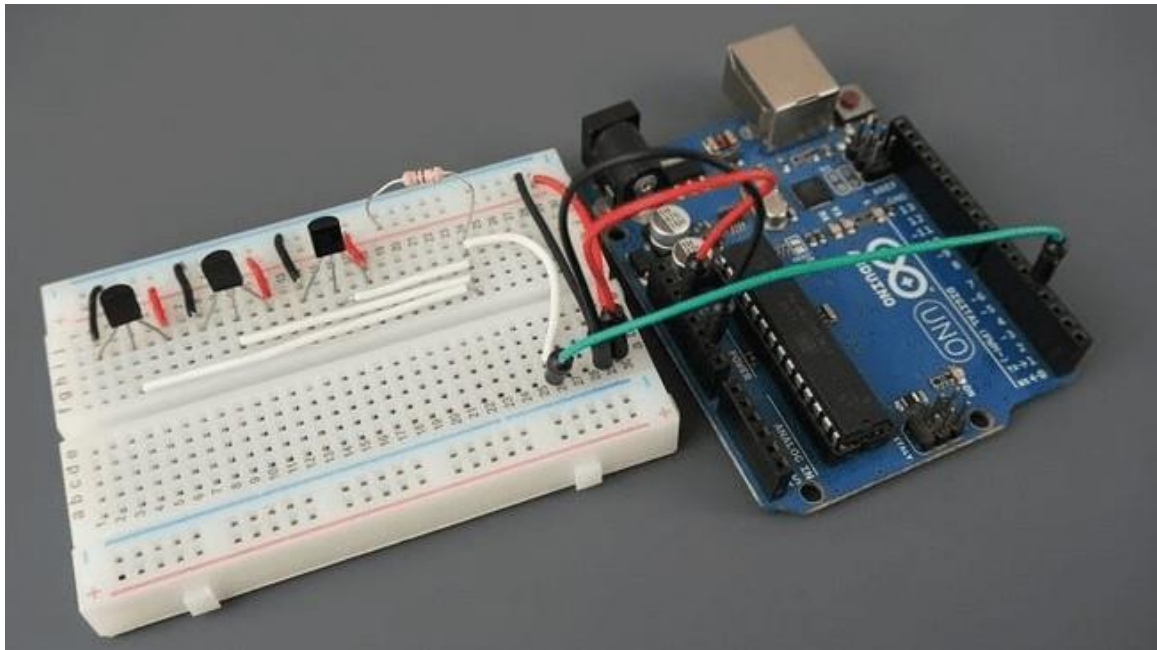


Figure: 3.36 Multiple Ds18b20 Sensors

The DS18B20 temperature sensor communicates using one-wire protocol and each sensor has a unique 64-bit serial code, so you can read the temperature from multiple sensors using just one single Arduino digital Pin.

Schematic

To read the temperature from multiple sensors, you just need to wire all data lines together as shown in the following schematic diagram:

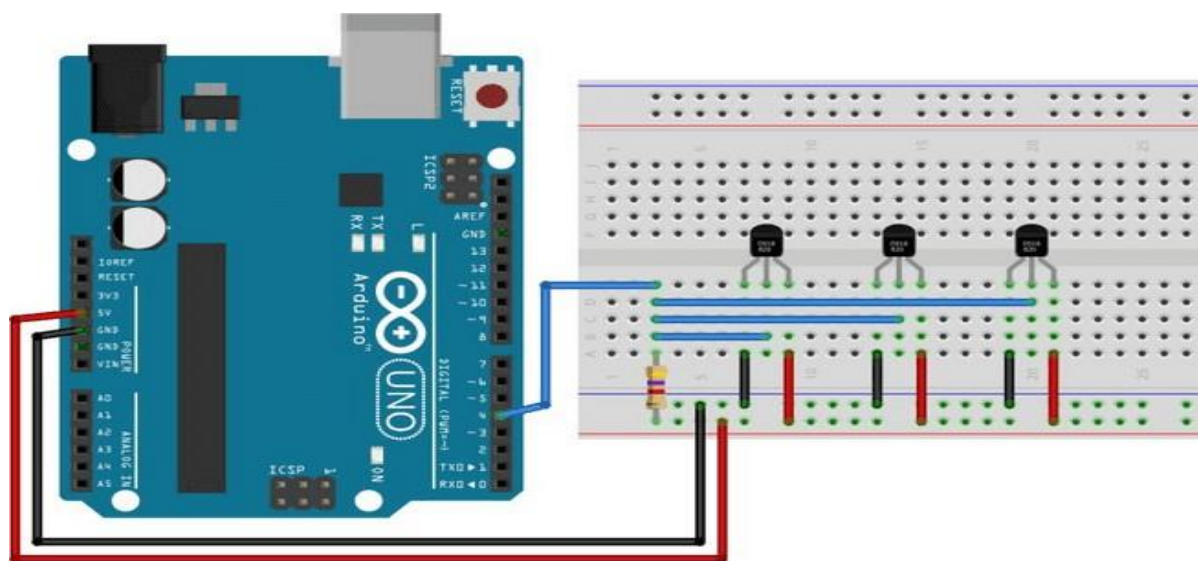


Figure: 3.37 Schematic Diagram

3.1.7 Battery

The **nine-volt battery**, or **9-volt battery**, is a common size of battery that was introduced for early transistor radios. It has a rectangular prism shape with rounded edges and a polarized snap connector at the top. This type is commonly used in smoke detectors, gas detectors, clocks, walkie-talkies, electric guitars and effects units. The nine-volt battery format is commonly available in primary carbon-zinc and alkaline chemistry, in primary lithium iron disulfide, and in rechargeable form in nickel-cadmium, nickel-metal hydride and lithium-ion. Mercury-oxide batteries of this format, once common, have not been manufactured in many years due to their mercury content. Designations for this format include NEDA 1604 and IEC

6F22 (for zinc-carbon) or MN1604 6LR61 (for alkaline). The size, regardless of chemistry, is commonly designated **PP3** a designation originally reserved solely for carbon-zinc, or in some countries, E or E-block. Most nine-volt alkaline batteries are constructed of six individual 1.5 V LR61 cells enclosed in a wrapper. These cells are slightly smaller than LR8D425 AAAA cells and can be used in their place for some devices, even though they are 3.5 mm shorter. Carbon-zinc types are made with six flat cells in a stack, enclosed in a moisture-resistant wrapper to prevent drying. Primary lithium types are made with three cells in series. 9-volt batteries accounted for 4% of alkaline primary battery sales in the United States in 2007, and 2% of primary battery sales and 2% of secondary battery (rechargeable) sales in Switzerland in 2008.

Contents

- History
- Connectors
- Technical specifications
- Testing and charging
- Lithium
- See also
- References History
- PP (Power Pack) battery family from left to right: PP1, PP3, PP4, PP6, PP7, PP8, PP9, PP10, PP11

Historically, the now popular PP3 battery size was a member of the PP (Power Pack) battery family that was originally manufactured by Ever Ready in the United Kingdom and Eveready in the United States. The company claims that it introduced the PP3 battery in 1956, then it was added as an ANSI standard in 1959, currently known as ANSI-1604A.

PP (Power Pack) battery family

Name	Voltage	Capacity	Depth	Width	Height
PP1	6 volts	4 Ah	55.6 mm	65.1 mm	55.6 mm
PP3	9 volts	0.5 Ah	17.5 mm	26.5 mm	48.5 mm
PP4	9 volts	0.9 Ah	Dia. 25.8 mm, Length: 49.8 mm -41.5 mm ex contacts		
PP6	9 volts	1 Ah	34.1 mm	35.7 mm	69.9 mm
PP7	9 volts	2.5 Ah	46 mm	46 mm	63 mm
PP8	6 volts	15 Ah	55 mm	61 mm	200 mm
PP9	9 volts	5 Ah	51.5 mm	65 mm	80 mm
PP10	9 volts	15 Ah	66 mm	65 mm	267 mm

PP (Power Pack) battery family

Name	Voltage	Capacity	Depth	Width	Height
PP1	4.5-volt x ₂	5 Ah	51.5 mm	65 mm	91 mm

Table 3.4 PP (Power Pack) Battery Family

The PP11 consists of two isolated 4.5-volt batteries with four terminals.

Today, only the PP3 (smallest 9V size), PP6, PP7 and PP9 sizes can still be purchased, with the PP3 being extremely common. Modern batteries can have higher capacity and lower internal resistance than the originals.

Before the mid-1950s, in the days of vacuum tube (valve) radios used batteries designed specifically for vacuum tubes, there was a nine-volt grid bias battery or (US) "C" battery, which had taps for various voltages from 1.5 to 9 volts.

Early transistorized radios and other equipment needed a suitable voltage miniature battery. Early transistor radios required a $22\frac{1}{2}$ -volt battery. Although the transistors would theoretically operate from lower voltages, in 1954, the point contact transistors had to be operated very close to their V_{CB0} limit in order to get the required frequency response. However, a suitable miniature battery was already marketed for (vacuum tube) hearing aids. As transistors rapidly improved, particularly when alloy transistors were introduced, radios were able operate from lower voltages and the battery manufacturers introduced suitable batteries as the demand arose.

Technical specifications



Figure: 3.39 9-Volt Battery

Collage of images showing the opening of a 9-volt battery to reveal six LR61 size cells, which are like the LR8D425 AAAA cells often used in medical equipment



Figure: 3.40 Alkaline Battery

Alkaline battery showing rectangular cell construction



Figure: 3.41 Three Different 9-Volt Primary Battery

Three different kinds of 9-volt primary battery internals: rectangular cell zinc- carbon, rectangular cell alkaline, and cylindrical cell alkaline



Figure: 3.42 Rechargeable Battery

Rechargeable (NiMH) 9-volt battery internals

The most common type of nine-volt battery is often called a 9-volt, although there are less common nine-volt batteries of different sizes. Codes for the usual size include PP3 (for size and voltage, any technology), 6LR61 (IEC code for alkaline

batteries), and in Japan 006P. The PP3 size battery is 48.5 mm × 26.5 mm × 17.5 mm or 1.91 in × 1.04 in × 0.69 in. Both terminals are at one end and their centers are 0.5 inches (12.7 mm) apart.

Inside an alkaline or carbon-zinc 9-volt battery there are six cylindrical or flat cells connected in series. Some brands use welded tabs internally to attach to the cells, others press foil strips against the ends of the cells.

Rechargeable nickel–cadmium (NiCad) and nickel–metal hydride (NiMH) batteries of nominal 9V rating have between six and eight 1.2 volt cells. Lithium ion versions typically use two cells (3.7–4.2 V nominal each). There are also lithium polymer and low self-discharge NiMH versions.

Mercury batteries were formerly made in this size. They had higher capacity than the then-standard carbon-zinc types, a nominal voltage of 8.4 volts, and very stable voltage. Once used in photographic and measuring instruments or long-life applications, they are no longer manufactured as mercury is highly toxic and accumulates in the ecosystem, posing a risk to humans and wildlife.

Type		IEC name	ANSI/NEDA name	Typical capacity in mAh	Nominal voltages
Primary (disposable)	Alkaline	6LR61	1604A	550	9
		6LP31 4 6	1604A	550	9

	Zinc–carbon	6F22	1604D	400	9
	Lithium		1604LC	1200	9
Rechargeable	NiCad	6KR61	11604	120	7.2, 8.4
	NiMH	6HR61	7.2H5	175-300	7.2, 8.4, 9.6
	Lithium polymer			520	7.4

Type		IEC name	ANSI/NEDA name	Typical capacity in mAh	Nominal voltages
	Lithium-ion			620	7.4
	Lithium ion phosphate			200-320	9.6

Table 3.5 Testing and Charging

Testing and charging

Most battery voltage testers and chargers that can also test nine-volt need another snap clip to hold the battery, while cylindrical batteries often share a holder that may be adjustable in size. Because of the proximity of the positive and negative terminals at the top of the battery and relatively low current of most common batteries, one informal method of testing voltage is to place the two terminals across a tongue. A strong tingle would indicate a battery with a strong charge, the absence, a discharged battery. While there have been stories circulating of unfortunate outcomes, the process is rarely dangerous under normal circumstances, though it may be unpleasant.

Lithium

Lithium 9-volt batteries are consumer-replaceable, disposable high-energy-density batteries. In the PP3 size they are typically rated at 0.8-1.2Ah about twice the capacity of alkaline batteries. Manufacturers claim "High energy density, up to 5x more than alkaline". Common applications for lithium nine-volt batteries are smoke and carbon monoxide (CO) alarms, and electronic parking meters.

3.1.8 ZIGBEE

ZigBee is a wireless technology developed as an open global standard to address the unique needs of low-cost, low-power, wireless sensor networks. The standard takes full advantage of the IEEE 802.15.4 physical radio specification and operates in unlicensed bands worldwide at the following frequencies: 2.400–2.484 GHz, 902-928 MHz and 868.0–868.6 MHz

- a) The power levels (down from 5v to 3.3v) to power the ZigBee module.
- b) The communication lines (TX, RX, DIN, and DOUT) to the appropriate voltages.

The Zigbee module acts as both transmitter and receiver. The Rx and Tx pins

of ZIGBEE are connected to Tx and Rx of microcontroller respectively. The data from microcontroller is serially transmitted to Zigbee module via UART port. Then Zigbee transmits the data to another Zigbee. The data's from Zigbee transmitted from D out pin. The Zigbee from other side receives the data via Din pin.

3.1.9 Force Sensor

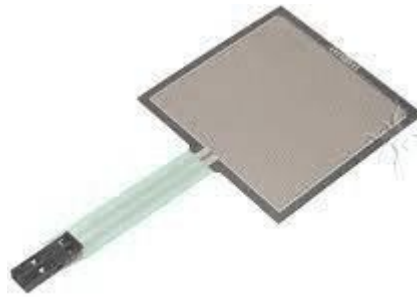


Figure: 3.43 Force Sensor

Force Sensing Resistors (FSR) are a polymer thick film (PTF) device which exhibits a decrease in resistance with an increase in the force applied to the active surface. Its force sensitivity is optimized for use in human touch control of electronic devices. FSRs are not a load cell or strain gauge, though they have similar properties. FSRs are not suitable for precision measurements.

Shape and Size

Most FSR's feature either a circular or rectangular sensing area. The square FSR is good for broad-area sensing, while the smaller circular sensors can provide more precision to the location being sensed.

The rectangular FSR's include a small-is square 1.75 x 1.75" sensor and a long 0.25 x 24" strip. The rest of the sensors feature a circular sensing area.

Sensing Range

Another key characteristic of the FSR is it's rated sensing range, which defines the minimum and maximum amounts of pressure that the sensor can differentiate between the lower the force rating.

But! Any pressure beyond the sensor's maximum limit will be unmeasurable (and

may damage the component). The small 1kg-rated FSR will provide more sensitive readings from 0 to 1kg, but will not be able to tell the difference between a 2kg and 10kg weight.

How does an FSR work?

The resistance of an FSR depends on the pressure that is applied to the sensing area. The more pressure you apply, the lower the resistance. The resistance range is quite large: $> 10\text{ M}\Omega$ (no pressure) to $\sim 200\ \Omega$ (max pressure). Most FSRs can sense force in the range of 100 g to 10 kg.

Selecting a Static Resistor

The tricky part of voltage-dividing an FSR is selecting a static resistor value to pair with it. You do not want to overpower the maximum resistance of the FSR, but you also do not want the FSR's minimum resistance to be completely overshadowed either.

It helps to know what range of force you will be reading. If your project's force-sensing covers the broad range of the FSR (e.g., 0.1-10kg), try to pick a static resistance in the middle-range of the FSR's resistive output -- something in the middle of 200-6k Ω . 3k Ω , or a common resistor like **3.3k Ω** , is a good place to start. The FSR is made of 2 layers separated by a spacer. The more one presses, the more of those Active Element dots touch the semiconductor and that makes the resistance go down. FSRs are basically a resistor that changes its resistive value (in ohms Ω) depending on how much it is pressed. These sensors are low cost, and easy to use but they are rarely accurate. They also vary some from sensor to sensor perhaps 10%. So basically, when you use FSRs you should only expect to get ranges of response. While FSRs can detect weight, they are a bad choice for detecting exactly how many pounds of weight are on them. However, for most touch-sensitive applications like "has this been squeezed or pushed and about how much" they are.

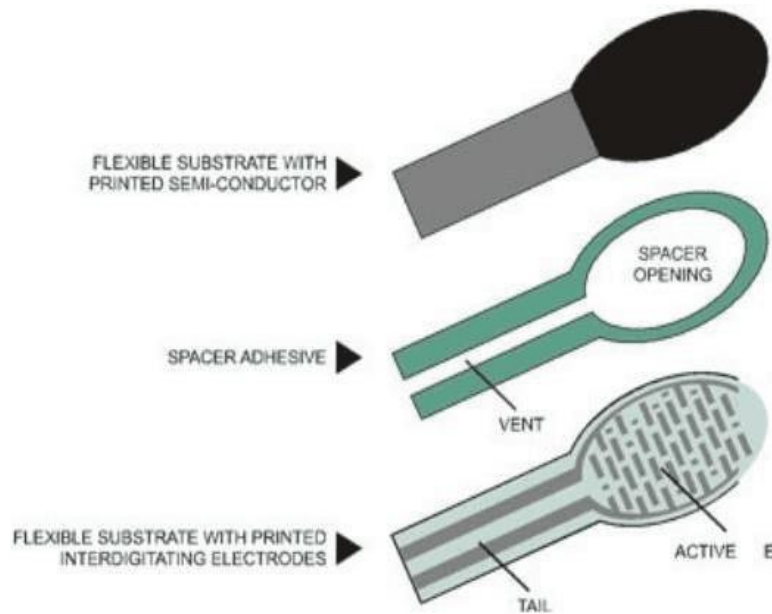


Figure: 3.44 Static Resistor

Force vs Resistance:

The force vs. resistance characteristic shown in figure below. provides an overview of FSR typical response behavior. For interpretational convenience, the force vs. resistance data is plotted on a log/log format. These data are representative of our typical devices, with this force-resistance characteristic being the response of evaluation.

A stainless- steel actuator with a 0.4" [10.0 mm] diameter hemispherical tip of 60 durometer polyurethane rubber was used to actuate the FSR device. In general, FSR response approximately follows an inverse power-law characteristic (roughly $1/R$). Referring to Figure below, at the low force end of the force-resistance characteristic, a switchlike response is evident. This turn-on threshold, or 'break force,' that swings the resistance from greater than 100 k Ω to about 10 k Ω (the beginning of the dynamic range that follows a power-law) is determined by the substrate and overlay thickness and flexibility, size and shape of the actuator, and spacer-adhesive thickness (the gap between the facing conductive elements). Break force increases with increasing substrate and overlay rigidity, actuator size, and spacer adhesive

thickness. Eliminating the adhesive, or keeping it well away from the area where the force is being applied, such as the center of a large FSR device, will give it a lower rest resistance (e.g., stand-off resistance).

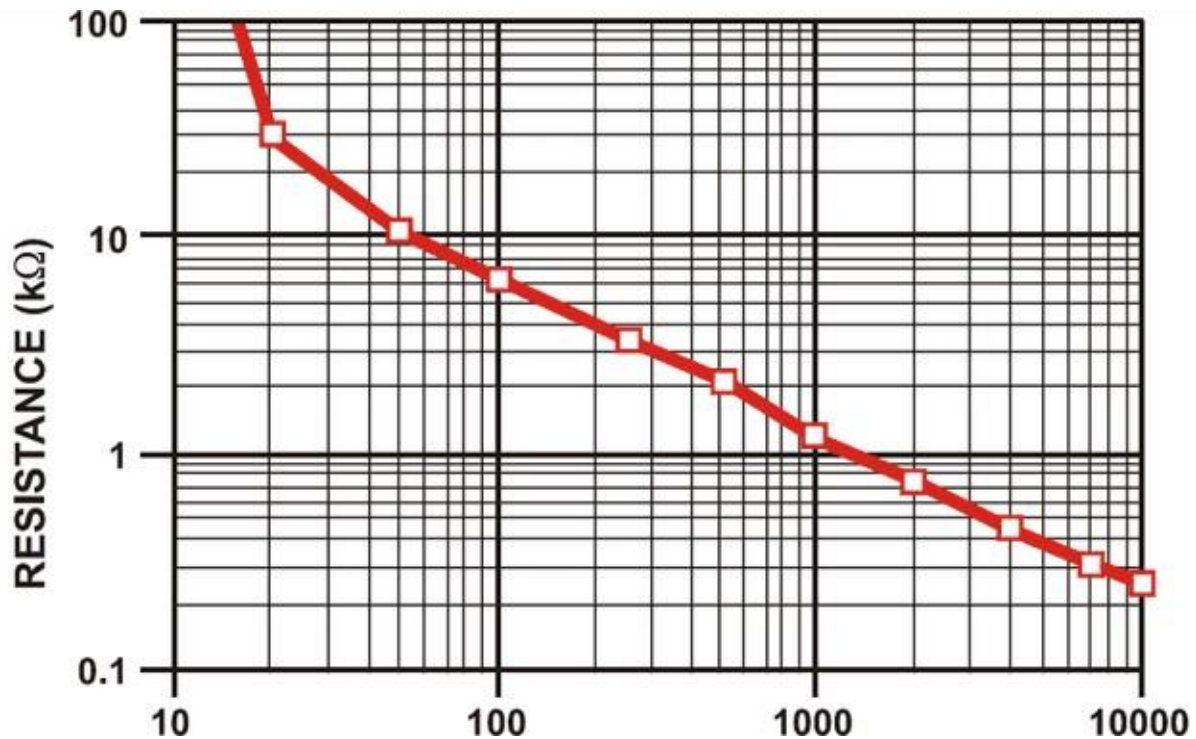


Figure: 3.45 Stand-Off Resistance

end to Power and the other to a pull-down resistor to ground. Then the point between the fixed pulldown resistor and the variable FSR resistor is connected to the analog input of a microcontroller such as an Arduino (shown). For this example, I am showing it with a 5V supply but note that you can use this with a 3.3v supply just as easily. In this configuration the analog voltage reading ranges from 0V (ground) to about 5V. The way this works is that as the resistance of the FSR decreases, the total resistance of the FSR and the pulldown resistor decreases from about 100Kohm to 10Kohm. That means that the current flowing through both resistors increases which in turn causes the voltage across the fixed 10K resistor to increase.

This table indicates the approximate analog voltage based on the sensor force/resistance w/a 5V supply and 10K pulldown resistor. Note that our method takes the somewhat linear resistivity but does not provide linear voltage.

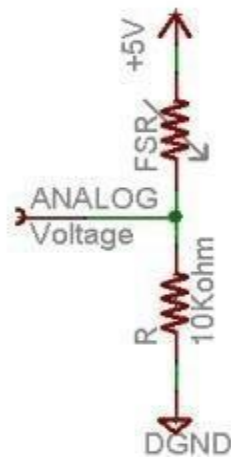
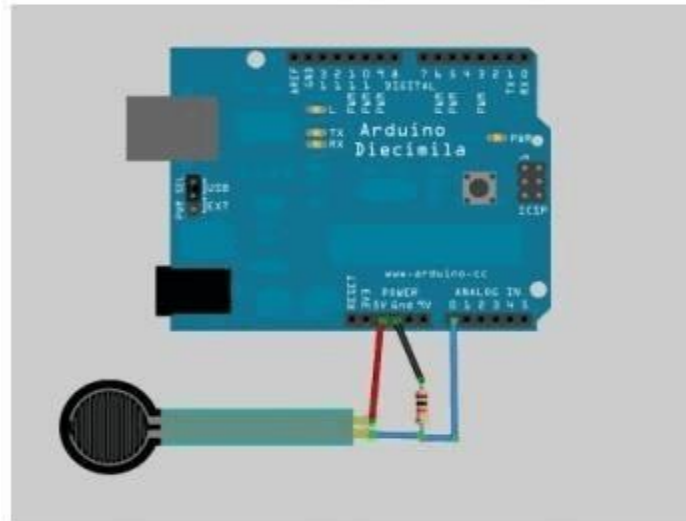


Figure: 3.46 FSR Analog Voltage

Force (lb)	Force (N)	FSR Resistance	(FSR + R) ohm	Current thru FSR+R	Voltage across R
None	None	Infinite	Infinite!	0 mA	0V
0.04 lb	0.2 N	30 Kohm	40 Kohm	0.13 mA	1.3 V
0.22 lb	1 N	6 Kohm	16 Kohm	0.31 mA	3.1 V
2.2 lb	10 N	1 Kohm	11 Kohm	0.45 mA	4.5 V
22 lb	100 N	250 ohm	10.25 Kohm	0.49 mA	4.9 V

Table 3.6 FSR Analog Voltage

That is because the voltage equation is:

$$V_o = V_{cc} (R / (R + FSR))$$

That is, the voltage is proportional to the inverse of the FSR resistance.

Force sensor connecting with Arduino uno

The FSR changes its resistance with force. It ranges from near infinite when not being touched, to under 300ohms when pressed hard. So, we can measure that change using one of the Arduino analog inputs. But to do that we need a fixed resistor that we can use for that comparison (we are using a 10 K resistor). This is called a Voltage divider and divides the 5v between the FSR and the resistor. The analog read on your Arduino is basically a Voltage meter. At 5V (its max) it will read 1023, and at 0v it will read 0. So, we can measure how much voltage is on the FSR using the analog Read and we will have our force reading. So, if the FSR and the resistor have the same resistance, the 5V is split evenly (2.5 V) to each part.

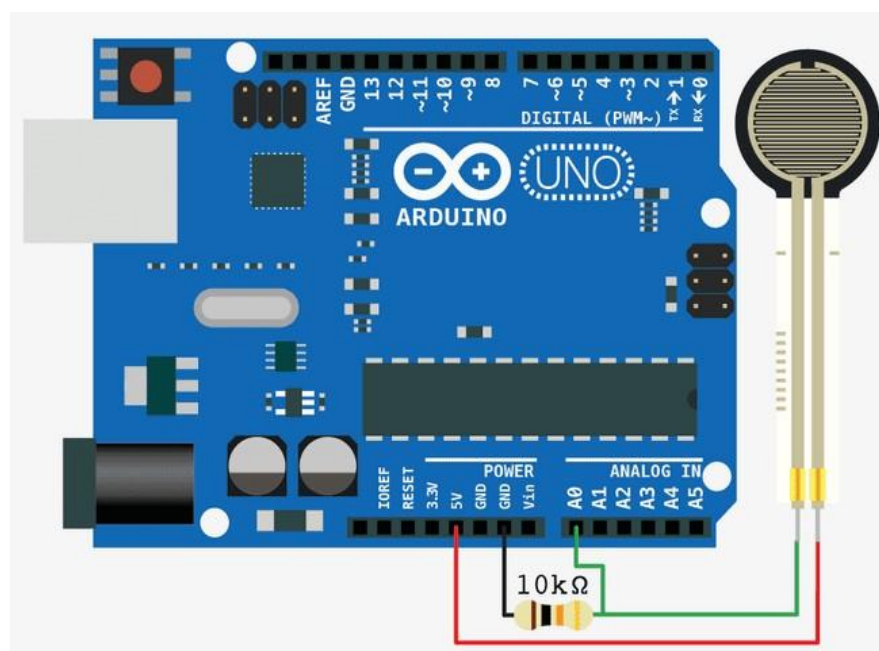


Figure: 3.47 Force Sensor Connecting with Arduino

But if the FSR is pressed on hard, reading only 1k of resistance, the 10K resistor is going to soak up 10 times as much of that 5V. So, the FSR Would only get 45 V. (analog reading of 92).

Features

Feature	Condition	Value*	Notes
Actuation Force		0.1 Newtons	
Force Sensitivity Range		0.1 - 10.0 ² Newtons	
Force Repeatability ³	(Single part)	± 2%	
Force Resolution ³		continuous	
Force Repeatability ³	(Part to Part)	±6%	
Non-Actuated Resistance		10M W	
Size		18.28mm diameter	
Thickness Range		0.2 - 1.25 mm	
Stand-Off Resistance		>10M ohms	Unloaded, unbent
Switch Travel	(Typical)	0.05 mm	Depends on design
Hysteresis ³		+10%	$(R_{F+} - R_{F-})/R_{F+}$
Device Rise Time		<3 microseconds	measured w/steel ball
Long Term Drift		<5% per log ₁₀ (time)	35 days test, 1kg load
Temp Operating Range	(Recommended)	-30 - +70 °C	
Number of Actuations	(Life time)	10 Million tested	Without failure

Table 3.7 Features

Application information

FSRs are two-wire devices with a resistance that depends on applied force. For specific application needs please contact Interlink Electronics support team.

Applications

There are numerous applications for force sensing resistors in various fields such as

- Foot pronation systems.
- Automobiles like car sensors.
- Resistive touch-pads.
- musical instruments
- keypads
- portable electronics

3.1.10 Vibration Motors

Precision Microdrive's currently produces coin vibration motors, also known as shaftless or pancake vibrator motors, generally in Ø8mm - Ø12mm diameters for

An integration guide is also available. For a simple force-to-voltage conversion, the FSR device is tied to a measuring resistor in a voltage divider configuration (see Figure 3). The output is described by the equation

$$V_{OUT} = \frac{R_M V +}{(R_M + R_{FSR})}$$

shown configuration, the output voltage increases with increasing force. If RFSR and RM are swapped, the output swing will decrease with increasing force. The measuring resistor, RM, is chosen to maximize the desired force sensitivity range and to limit current. Depending on the impedance requirements of the measuring circuit, the voltage divider could be followed by an op-amp. A family of force vs. VOUT curves is shown on the graph below for a standard FSR in a voltage divider configuration with various RM resistors. A (V+) of +5V was used for these examples our Pico Vibe range. Pancake motors are compact and convenient to use. They integrate into many designs because they have no external moving parts, and can be affixed in place with a strong permanent self-adhesive mounting system. Enclosures can easily be moulded to accept the coin form of our shaftless vibration motors. Within the coin motor range, we offer both leaded and spring & pad mountable versions. Like all our vibration motors, we are happy to quote for variations to the

base design such as a modification to the lead length and connectors.

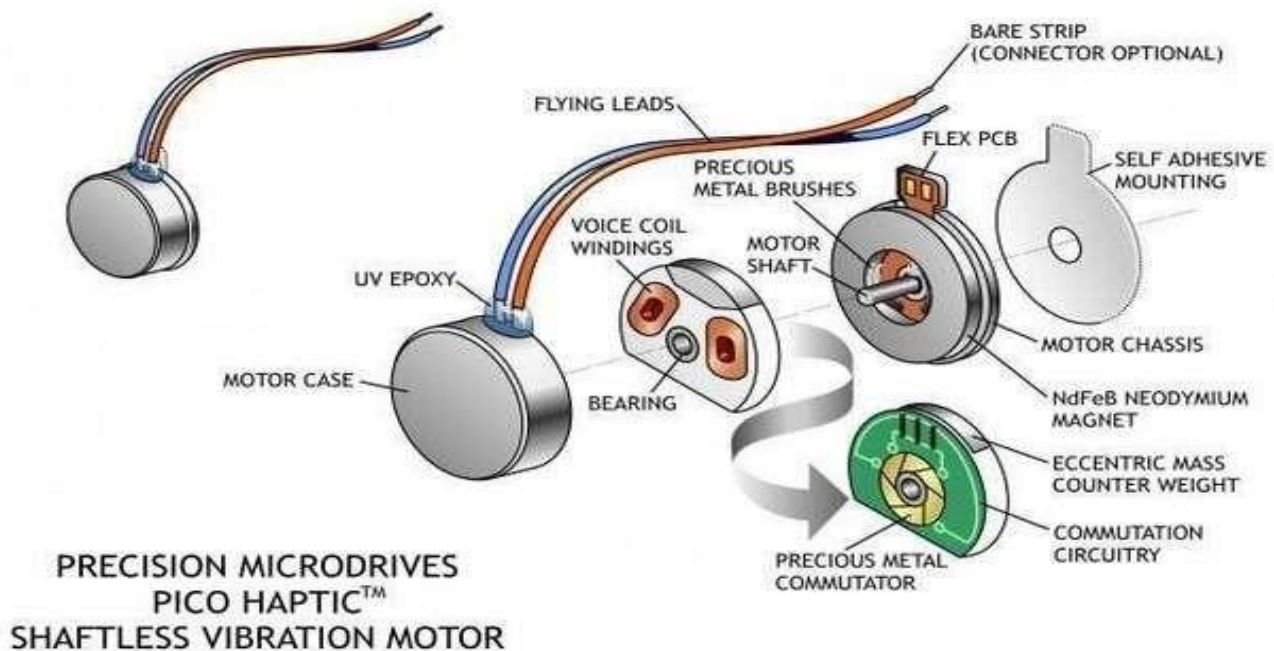


Figure: 3.48 Exploded Coin

Applications

Due to their small size and enclosed vibration mechanism, coin vibrating motors are a popular choice for many different applications. They are great for haptics, particularly in handheld instruments where space can be at a premium:

- Mobile phones
 - RFID scanners
 - Industrial tools or equipment user interfaces
 - Portable instruments
 - Medical applications
- General Layout and Operation

Our coin or pancake vibrating motors are all Eccentric Rotating Mass (ERM) motors. Therefore, they can be driven in the same manner as their pager motor counterparts. They have the same motor drive principles, including H-bridge

circuitry for active braking.

Brushed coin vibration motors are constructed from a flat PCB on which the 3-pole commutation circuit is laid out around an internal shaft in the center. The vibration motor rotor consists of two 'voice coils' and a small mass that is integrated into a flat plastic disc with a bearing in the middle, which sits on a shaft. Two brushes on the underside of the plastic disc make contact to the PCB commutation pads and provide power to the voice coils which generate a magnetic field. This field interacts with the flux generated by a disc magnet that is attached to the motor chassis.

The commutation circuit alternates the direction of the field through the voice coils, and this interacts with the N-S pole pairs that are built into the neodymium magnet. The disc rotates and, due to the built-in off-centered eccentric mass, the motor vibrates! Equivalent Coin Motor Circuitry, Commutation, Terminal Resistance

The commutator is formed by 6 segments connected to two coils. The equivalent circuit is shown on the right. The coils can be magnetized in 6 different ways, effectively making this a 6-pole machine. However, a peculiarity of this commutation design is that during one rotation the resistance through the brushes is not constant. For a third of the revolution the brushes "see" the two coils in series instead of only one; which is why in some orientations the resistance seen by a circuit will be double and therefore the start current half the rated value. The current figures presented in Conformity Limits Specifications sections of datasheets represent the worst-case current draw; i.e., where the brushes see only one coil. The full term 'Maximum Start Voltage' is the lowest voltage that you can apply to the motor and still be sure that it will start. Start Voltages and Drive Signals. Coin vibration motors have a relatively high start voltage (compared to cylinder pager vibration motors) which must be considered in designs. Typically, this is around 2.3v (all coin vibration motors have a nominal voltage of 3v), and failure to respect this could result in motors not starting when the application is lying in certain orientations.

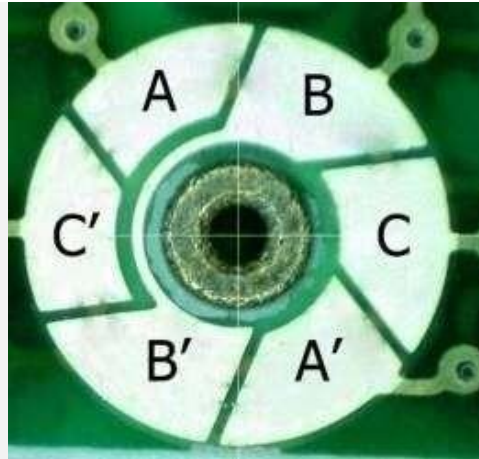


Figure: 3.49 Coin Motor Commutation Circuitry

This problem arises because, in the vertical orientation, the coin vibrating motor must force the eccentric mass over the top of the shaft on the initial cycle.

Due to the start voltage issue, we recommend that coin type vibrating motors are switched hard on and off at a voltage above the guaranteed start voltage unless a well-tested haptics driver is used.

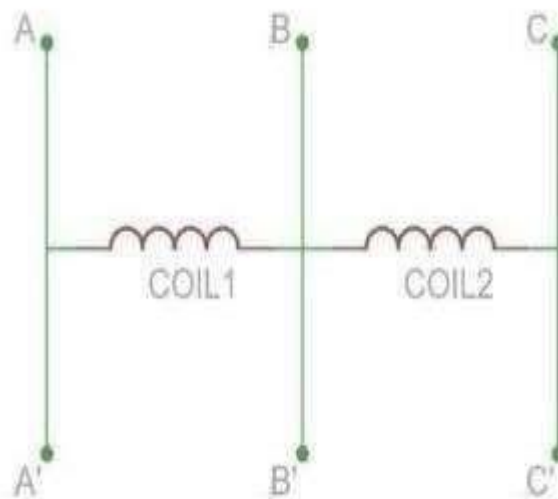


Figure: 3.50 Coin Motor Equivalent Circuit



Figure: 3.51 Self-Adhesive Vibration Motor Mounting

Coin vibration motors are designed to be easy to mount. They come with either spring PCB connectors or a high-strength long life self-adhesive backing sheet that is pre-attached to the underside of the chassis. The adhesive allows for a secure mounting of the vibration motor to a wide range of surfaces such as PCBs or flat internal surfaces of the enclosure and makes manufacturing installation fast and clean.

Three brands of adhesive are typically used on our coin vibrator motors depending on availability (they all have very similar specifications).



Figure: 3.52 8mm Coin (Shaftless) Vibration Motor

These are:

- 3M VHB 9448
- Sony 4000T
- Nitto 5000NS

These 0.16mm thick adhesive tapes typically offer a 180 deg peeling strength of

15N/20mm, and a tensile strength of around 20N/10mm. The acrylic adhesive is resistant to most solvents, UV light, moisture, and temperature extremes.

As with all adhesives, the final bond strength is dependent on the cleanliness of the mating surface. It is recommended that this mating surface is clean, dry, and offers a good unified fit to the motor backing-plate (on which the self-adhesive pad is stuck).

Spring PCB vibrator motors have spring-loaded fingers on the motor which mate with pads on the PCB. This makes assembly easier for applications where it's desired to have the motor mounted to the enclosure. Also, higher frequency harmonics are absorbed and reduced by the rubber 'boots' that enclose these kinds of motors.

If extra security is required, consider the molding securing walls within the enclosure body. This technique is commonly used in mobile phones to ensure that the maximum amount of vibration is transmitted through the case.

Leads & Connectors



Figure: 3.53 10mm Coin Spring-Tab Vibration Motor

Some models are available with sprung gold terminals that sit on top of the motor (shown right). This pancake vibration motor is designed to be fitted to the enclosure. The springs mate with pads on the motherboard PCB which makes the connection, which is a neat way to avoid routing flying leads around the case if there is PCB real-estate available. You can read more on spring & pad vibration motors as part of our PCB mounted vibration motor pages.

3.2 SOFTWARE REQUIREMENTS

3.2.1 Embedded C

Embedded C is most popular programming language in software field for developing electronic gadgets. Each processor used in electronic system is associated with embedded software. Embedded C programming plays a key role in performing specific function by the processor. In day-to-day life we used many electronic devices such as mobile phone, washing machine, digital camera, etc. These all-device working is based on microcontroller that are programmed by embedded C. Let us see the block diagram representation of embedded system programming:

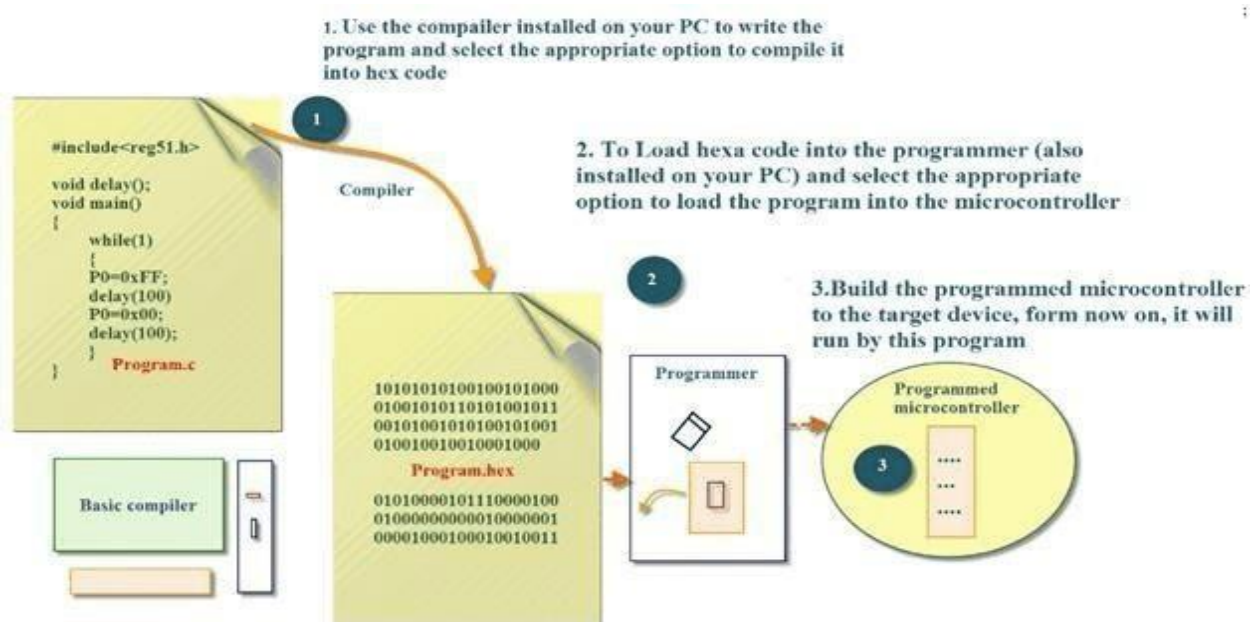


Figure: 3.54 Block Diagram of Embedded C

The Embedded C code written in above block diagram is used for blinking the LED connected with Port0 of microcontroller.

In embedded system programming C code is preferred over other language. Due to the following reasons:

- Easy to understand

- High Reliability
- Portability
- Scalability

Embedded System Programming

Basic Declaration

Let us see the block diagram of Embedded C Programming development:

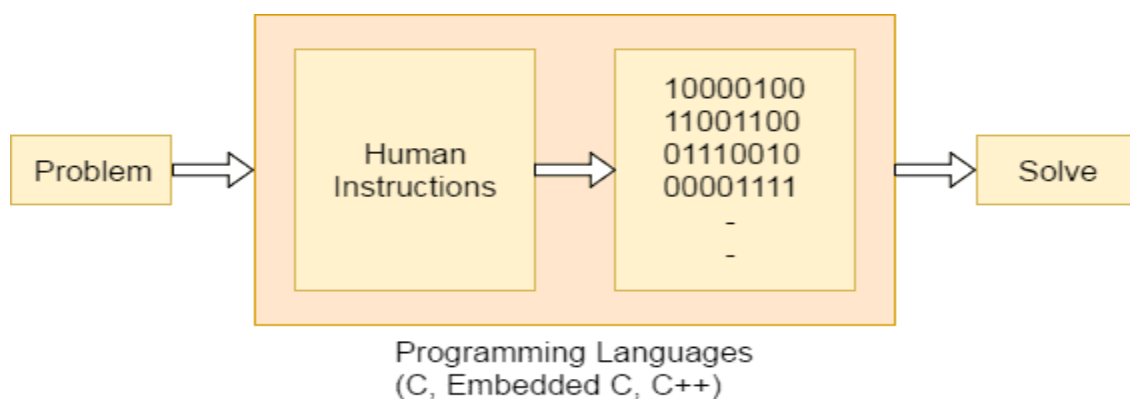


Figure: 3.55 Basic Declaration of Embedded C

Function is a collection of statements that is used for performing a specific task and a collection of one or more functions is called a programming language. Every language is consisting of basic elements and grammatical rules. The C language programming is designed for function with variables, character set, data types, keywords, expression and so on are used for writing a C program.

The extension in C language is known as embedded C programming language. As compared to above the embedded programming in C is also have some additional features like data types, keywords, and header file etc. is represented by

```
#include<microcontroller name's>.
```

Basic Embedded C Programming Steps

Let us see the block diagram representation of Embedded C Programming Steps:

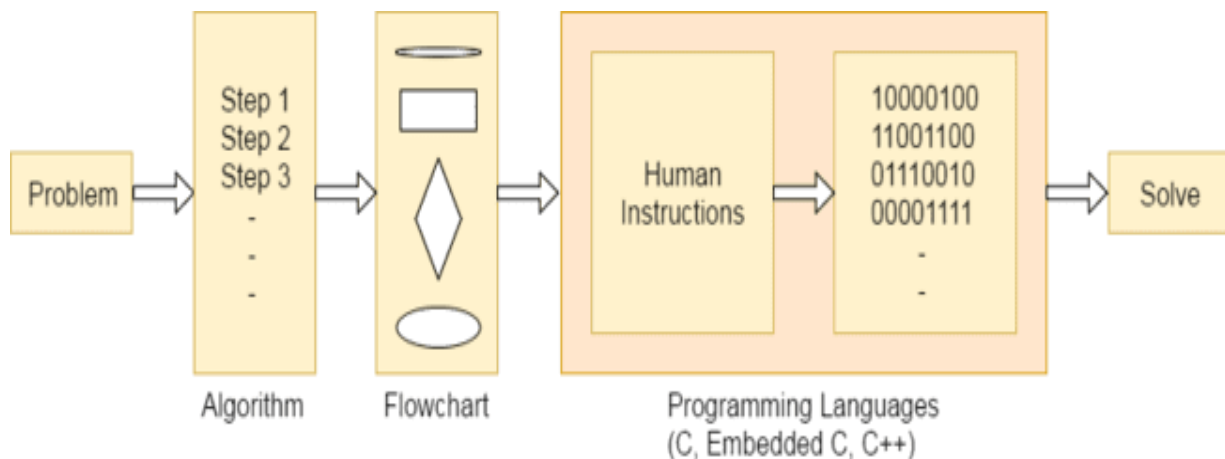


Figure: 3.56 Steps of Embedded C Programming

The microcontroller programming is different for each type of operating system. Even though there are many operating system is existed such as Windows, Linux, RTOS, etc. but RTOS has several advantages for embedded system development.

Embedded Systems

Embedded System is a system composed of hardware, application software and real time operating system. It can be small independent system or large combinational system. Our Embedded System tutorial includes all topics of Embedded System such as characteristics, designing, processors, microcontrollers, tools, addressing modes, assembly language, interrupts, embedded c programming, led blinking, serial communication, lcd programming, keyboard programming, project implementation etc.

System

System is a way of working, organizing, or performing one or many tasks according to a fixed set of rules, program or plan. It is an arrangement in which all the unit combined to perform a work together by following certain set of rules in real time computation. It can also be defined as a way of working, organizing, or doing one or many tasks according to a fixed plan. An Embedded System is a system that has software embedded into computer- hardware, which makes a system dedicated for a variety of application or specific part of an application or product or

part of a larger system. An embedded system can be a small independent system or a large combinational system. It is a microcontroller-based control system used to perform a specific task of operation.

An embedded system is a combination of three major components:

- **Hardware:** Hardware is physically used component that is physically connected with an embedded system. It comprises of microcontroller based integrated circuit, power supply, LCD display etc.
- **Application software:** Application software allows the user to perform varieties of application to be run on an embedded system by changing the code installed in an embedded system.
- **Real Time Operating system (RTOS):** RTOS supervises the way an embedded system work. It acts as an interface between hardware and application software which supervises the application software and provide mechanism to let the processor run on the basis of scheduling for controlling the effect of latencies.

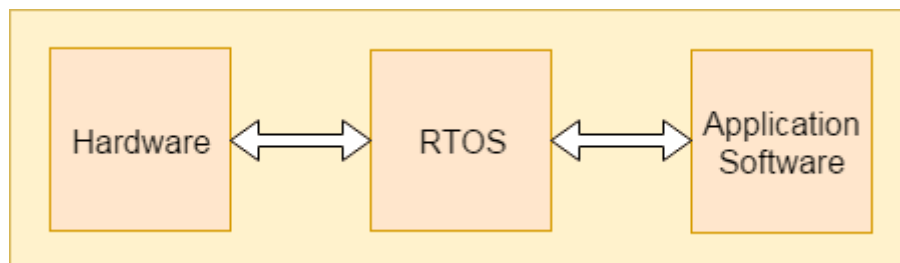


Figure: 3.57 RTOS

Process

Characteristics Of Embedded System

- An embedded system is software embedded into computer hardware that makes a system dedicated to be used for variety of application.
- Embedded system generally used for do specific task that provide real-time output based on various characteristics of an embedded system.
- Embedded system may contain a smaller part within a larger device

that used for serving the more specific application to perform variety of task.

- It provides high reliability and real-time computation ability.

Advantages

- Same hardware can be used in variety of application.
- Lesser power requirement
- Lower operational cost of system
- Provide high performance and efficiency

Disadvantages

- Developing a system required more time. Due to functional complexity.
- Skilled engineers required because one mistake may result in destroying of complete project.

Designing Of an Embedded System

Basic Structure of An Embedded System

Let us see the block diagram shows the basic structure of an embedded system.

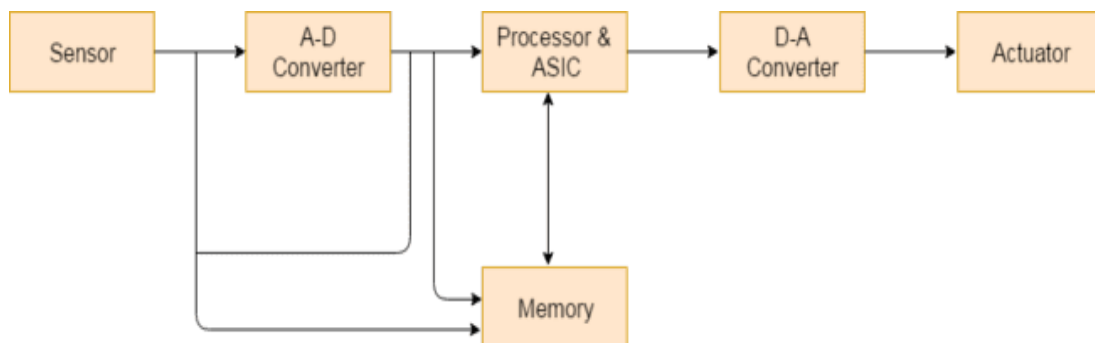


Figure: 3.58 Basic Structure of Embedded System

Sensor: Sensor used for sensing the change in environment condition and it generate the electric signal based on change in environment condition. Therefore, it is also called as transducers for providing electric input signal on the basis of change in environment condition.

A-D Converter: An analog-to-digital converter is a device that converts analog electric input signal into its equivalent digital signal for further processing in an embedded system.

Processor & ASICs: Processor used for processing the signal and data to execute desired set of instructions with high-speed of operation. Application specific integrated circuit (ASIC) is an integrated circuit designed to perform task specific operation inside an embedded system.

D-A Converter: A digital-to-analog converter is a device that converts digital electric input signal into its equivalent analog signal for further processing in an embedded system.

Actuators: Actuators is a comparator used for comparing the analog input signal level to desired output signal level for providing the error free output from the system.

Design Steps Required for The Development Of Embedded System

Designing steps required for embedded system are different from the design process of another electronic system. Let's see a flow chart represent the design steps required in the development of an embedded system:

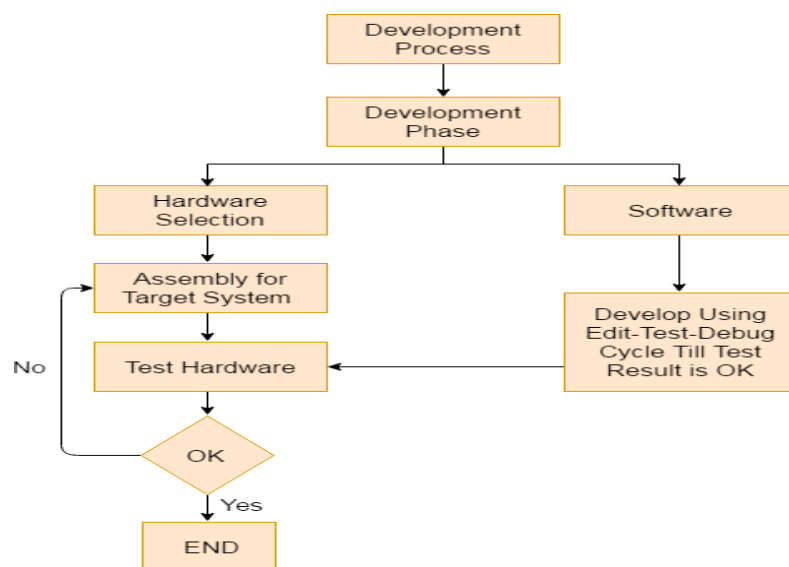


Figure: 3.59 Flow Chart of Developing Embedded System

Embedded System Tools and Peripherals

Compiler

Compiler is used for converting the source code from a high-level programming language to a low-level programming language. It converts the code written in high level programming language into assembly or machine code. The main reason for conversion is to develop an executable program.

Let us see the operations performed by compiler are:

- Code generation
- Code optimization
- Parsing
- Syntax direct translation
- Preprocessing

Cross-Compiler

If a program compiled is run on a computer having different operating system and hardware configuration than the computer system on which a compiler compiled the program, that compiler is known as cross-compiler.

Decompiler

A tool used for translating a program from a low-level language to high-level language is called a decompiler. It is used for conversion of assembly or machine code to high-level programming language.

Assembler

Assembler is embedded system tool used for translating a computer instruction written in assembly language into a pattern of bits which is used by the computer processor for performing its basic operations. Assembler creates an object code by translating assembly language instruction into set of mnemonics for representing each low-level machine operation.

Debugging Tools in An Embedded System

Debugging is a tool used for reducing the number of error or bugs inside a computer program or an assembled electronic hardware. Debugging of a compact subsystem is difficult because a small change in one subsystem can create bugs in another system. The debugging used inside embedded system differs in terms of their development time and debugging features.

Let us see the different debugging tools used in embedded system are Simulators:

Simulator is a tool used for simulation of an embedded system. Code tested for microcontroller unit by simulating code on the host computer. Simulator is used for model the behavior of the complete microcontroller in software.

Functions of simulators

Let us see the functions performed by simulator are:

- It defines the processing or processor device family with various version of target system.
- It monitors the detailed information of a source code and symbolic arguments as the execution goes for each single step of operation.
- It simulates the ports of target system for each single step of execution.
- It provides the working status of RAM.
- It monitors the response of system and determines the throughput.
- It provides the complete meaning of the present command.
- It monitors the detailed information of the simulator commands entered from the keyboard or selected from the menu.
- It facilitates synchronization of internal peripherals and delays.

Microcontroller Starter Kit

For developing an embedded system-based project a complete microcontroller starter kit is required. The major advantage of this kit over simulator

is that they work in real-time operating condition. Therefore, it allows the easy input/output functional verification. Consider a microcontroller starter kit consists of: -

- Hardware Printed Circuit Board (PCB)
- In-System Programmer (ISP)
- Some embedded system tools like compiler, assembler, linker, etc.
- Sometimes, there is a requirement of an Integrated Development Environment (IDE)

The above component available in microcontroller starter kit is completely enough and the cheapest option available for developing simple microcontroller projects.

Emulators

An emulator is a software program or a hardware kit which emulates the functions of one computer system into another computer system. Emulators have an ability to support closer connection to an authenticity of the digital object. It can also be defined as the ability of a computer program in electronic device to emulate another program or device. It focusing on recreating the original computer environment and helps a user to work on any type of application or operating system.

Peripheral Devices in Embedded Systems

Communication of an embedded system with an outside environment is done by using different peripheral devices as a combination with microcontroller.

Let us see the different peripheral devices in embedded system are: -

- Universal Serial Bus (USB)
- Networks like Ethernet, Local Area Network (LAN) etc.
- Multi Media Cards (SD Cards, Flash memory, etc.

- Serial Communication Interface (SCI) like RS-232, RS-485, RS-422, etc.
- Synchronous Serial Communication Interface like SPI, SSC and ESSI
- Digital to Analog/ Analog to Digital (DAC/ADC)
- General Purpose Input/Output (GPIO)
- Debugging like In System Programming (ISP), In Circuit Serial Programming (ICSP), BDM Port, etc.

Criteria For Choosing Microcontroller

Choosing a microcontroller is essential process in designing of embedded system. While selecting a microcontroller, make sure that it meets the system need and it must be cost effective. We need to decide whether an 8-bit, 16-bit or 32-bit microcontroller is best suitable for the computing needs of a task. In addition to above, the following points need to be kept in mind while selecting a microcontroller: -

- **Speed:** The operational speed of the microcontroller or the highest speed microcontroller can support.
- **Packaging:** Packaging is important for improving the assembling, space and prototyping of an end-product.
- **RAM and ROM:** Based on operation of embedded system and memory need for storage data and programs the type of microcontroller required for designing system is decided.
- **Count of I/O pins:** The number of input and output devices connected with the system plays an essential role in choosing the type of microcontroller.
- **Cost per unit:** It is important in terms of final cost of the product in which the microcontroller is to be used.
- **Power consumption:** Power consumption plays an important role for maintaining the efficiency of an embedded system.

3.2.2 Arduino Software Ide

The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino and Genuine hardware to upload programs and communicate with them.

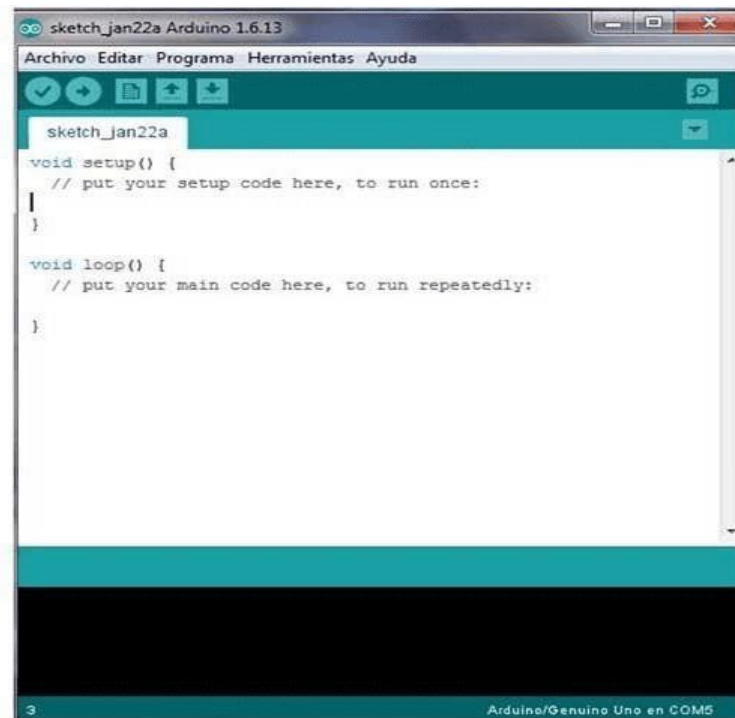


Figure: 3.60 Arduino Software Ide

Writing Sketches

Programs written using Arduino Software (IDE) are called **sketches**. These sketches are written in the text editor and are saved with the file extension. ion. The editor has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and displays errors. The console displays text output by the Arduino Software (IDE), including complete error messages and other information. The bottom right-hand corner of the window displays the configured board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.

NB: Versions of the Arduino Software (IDE) prior to 1.0 saved sketches with the extension. pde. It is possible to open these files with version 1.0, you will be prompted to save the sketch with the. ino extension on save.

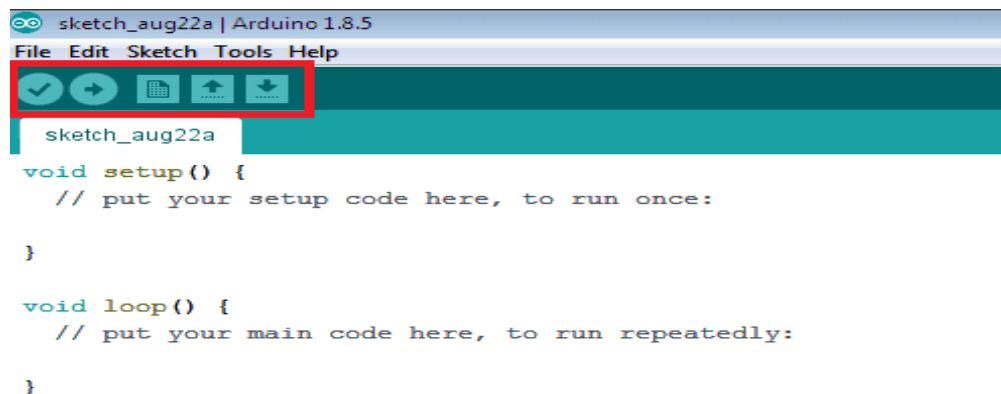





Figure: 3.61 Writing Sketches

	<p>Verify</p> <p>Checks your code for errors compiling it.</p>
	<p>Upload</p> <p>Compiles your code and uploads it to the configured board. See uploading below for details.</p>
	<p>Note: If you are using an external programmer with your board, you can hold down the "shift" key on your computer when using this icon. The text will change to "Upload using Programmer"</p>
	<p>New</p> <p>Creates a new sketch.</p>




	<p>Open</p> <p>Presents a menu of all the sketches in your sketchbook. Clicking one will open it within the current window overwriting its content.</p> <p>Note: due to a bug in Java, this menu doesn't scroll; if you need to open a sketch late in the list, use the File Sketchbook menu instead.</p>
	<p>Save</p> <p>Saves your sketch.</p>
	<p>Serial Monitor</p> <p>Opens the serial monitor.</p>

Table 3.8 Writing Sketches

Additional commands are found within the five menus: **File**, **Edit**, **Sketch**, **Tools**, and help. The menus are context sensitive, which means only those items relevant to the work currently being carried out are available.

File

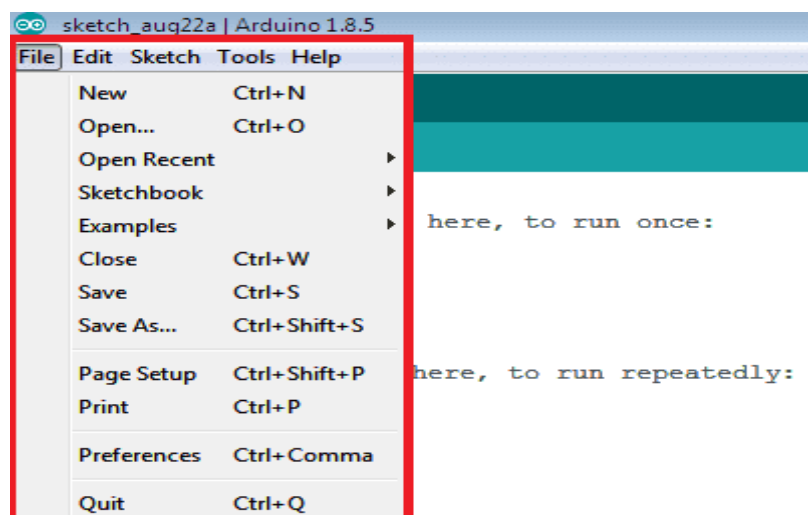


Figure: 3.62 File

➤ **New**

Creates a new instance of the editor, with the bare minimum structure of a sketch already in place.

➤ **Open**

Allows loading a sketch file browsing through the computer drives and folders.

➤ **Open Recent**

Provides a short list of the most recent sketches, ready to be opened.

➤ **Sketchbook**

Shows the current sketches within the sketchbook folder structure; clicking on any name opens the corresponding sketch in a new editor instance.

➤ **Examples**

Any example provided by the Arduino Software (IDE) or library shows up in this menu item. All the examples are structured in a tree that allows easy access by topic or library.

➤ **Close**

Closes the instance of the Arduino Software from which it is clicked.

➤ **Save**

Saves the sketch with the current name. If the file has not been named before, a name will be provided in a "Save as." window.

➤ **Save as**

Allows saving the current sketch with a different name.

➤ **Page Setup**

It shows the Page Setup window for printing.

➤ **Print**

Sends the current sketch to the printer according to the settings defined in Page Setup.

➤ **Preferences**

Opens the Preferences window where some settings of the IDE may be customized, as the language of the IDE interface.

➤ **Quit**

Closes all IDE windows. The same sketches open when quit was chosen will be automatically reopened the next time you start the IDE.

Edit

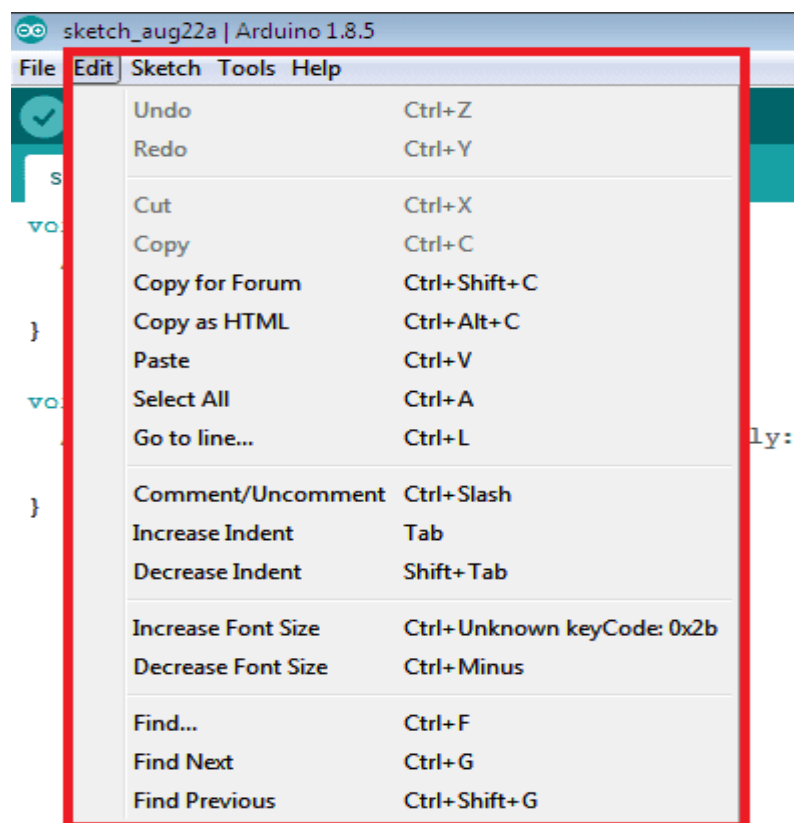


Figure: 3.63 Edit

➤ **Undo/Redo**

Goes back of one or more steps you did while editing; when you go back, you may go forward with Redo.

➤ **Cut**

Removes the selected text from the editor and places it into the clipboard.

➤ **Copy**

Duplicates the selected text in the editor and places it into the clipboard.

➤ **Copy for Forum**

Copies the code of your sketch to the clipboard in a form suitable for posting to the forum, complete with syntax coloring.

➤ **Copy as HTML**

Copies the code of your sketch to the clipboard as HTML, suitable for embedding in web pages.

➤ **Paste**

Puts the contents of the clipboard at the cursor position, in the editor.

➤ **Select All**

Selects and highlights the whole content of the editor.

➤ **Comment/Uncomment**

Puts or removes the // comment marker at the beginning of each selected line.

➤ **Increase/decrease Indent**

Adds or subtracts a space at the beginning of each selected line, moving the text one space on the right or eliminating a space at the beginning.

➤ **Find**

Opens the Find and Replace window where you can specify text to search inside the current sketch according to several options.

➤ **Find Next**

Highlights the next occurrence - if any - of the string specified as the search item in the Find window, relative to the cursor position.

➤ **Find Previous**

Highlights the previous occurrence - if any - of the string specified as the search item in the Find window relative to the cursor position.

Sketch

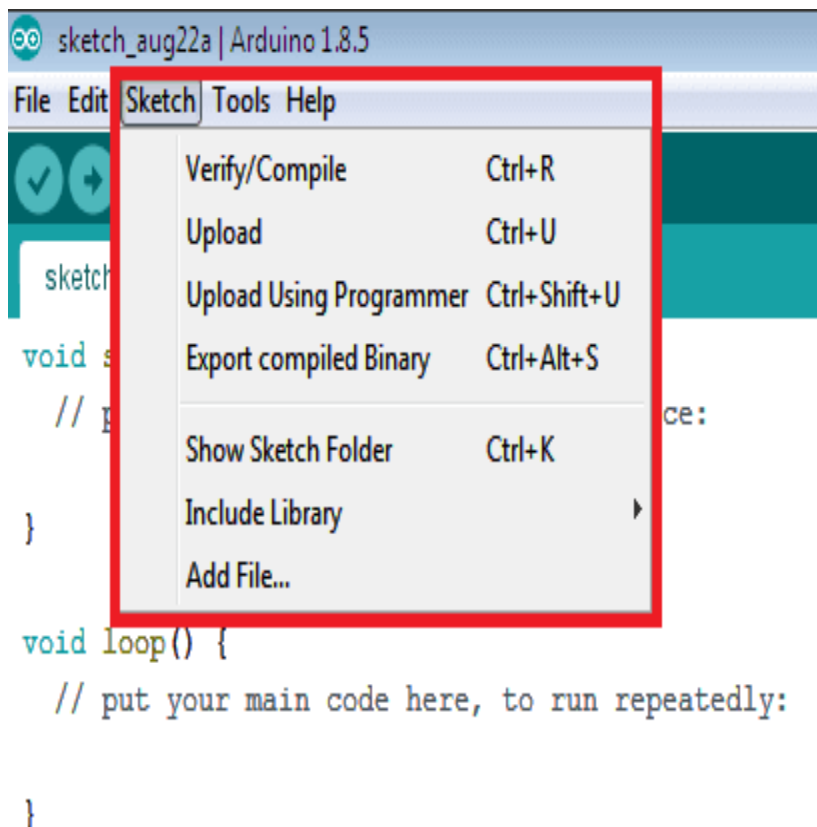


Figure: 3.64 Sketch

➤ **Verify/Compile**

Checks your sketch for errors compiling it; it will report memory usage for code and variables in the console area.

➤ **Upload**

Compiles and loads the binary file onto the configured board through the configured Port.

➤ **Upload Using Programmer**

This will overwrite the boot loader on the board; you will need to use Tools > Burn Boot loader to restore it and be able to Upload to USB serial port

again. However, it allows you to use the full capacity of the Flash memory for your sketch. Please note that this command will NOT burn the fuses. To do so a Tools -> Burn Bootloader command must be executed.

➤ **Export Compiled Binary**

Saves a .hex file that may be kept as archive or sent to the board using other tools.

➤ **Show Sketch Folder**

Opens the current sketch folder.

➤ **Include Library**

Adds a library to your sketch by inserting #include statements at the start of your code. For more details, see libraries below. Additionally, from this menu item you can access the Library Manager and import new libraries from .zip files.

➤ **Add File...**

Adds a source file to the sketch (it will be copied from its current location). The new file appears in a new tab in the sketch window. Files can be removed from the sketch using the tab menu accessible clicking on the small triangle icon below the serial monitor one on the right side of the toolbar.

Tools

➤ **Auto Format**

This formats your code nicely: i.e., indents it so that opening and closing curly braces line up, and that the statements inside curly braces are indented more.

➤ **Archive Sketch**

Archives a copy of the current sketch in .zip format. The archive is placed in the same directory as the sketch.

➤ **Fix Encoding & Reload**

Fixes possible discrepancies between the editor char map encoding.

➤ **Burn Boot loader**

The items in this menu allow you to burn a boot loader onto the microcontroller on an Arduino board. This is not required for normal use of an Arduino or Genuino board but is useful if you purchase a new AT mega microcontroller (which normally comes without a boot loader). Ensure that you have selected the correct board from the **Boards** menu before burning the boot loader on the target board. This command also set the right fuses.

Help

Here you find easy access to several documents that come with the Arduino Software (IDE). You have access to Getting Started, Reference, this guide to the IDE and other documents locally, without an internet connection. The documents are a local copy of the online ones and may link back to our online website.

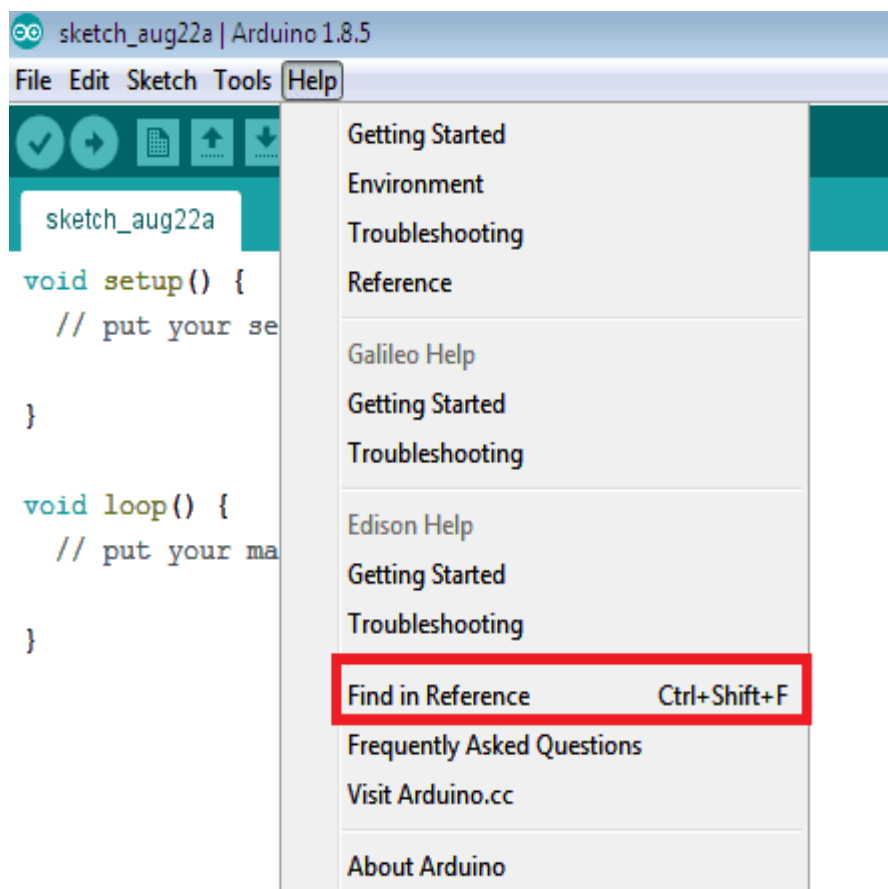


Figure: 3.66 Help

➤ **Finding Reference**

This is the only interactive function of the Help menu: it directly selects the relevant page in the local copy of the Reference for the function or command under the cursor.

Sketchbook

The Arduino Software (IDE) uses the concept of a sketchbook: a standard place to store your programs (or sketches). The sketches in your sketchbook can be opened from the **File > Sketchbook** menu or from the **Open** button on the toolbar. The first time you run the Arduino software, it will automatically create a directory for your sketchbook. You can view or change the location of the sketchbook location from with the **Preferences** dialog.

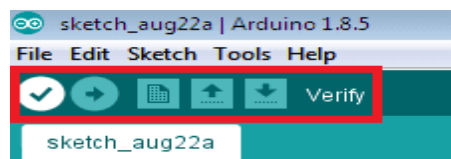


Figure: 3.67 Sketchbook

Beginning with version 1.0, files are saved with a .ino file extension. Previous versions use the. pde extension. You may still open. pde named files in version 1.0 and later, the software will automatically rename the extension to. ino.

Tabs, Multiple Files, And Compilation

Allows you to manage sketches with more than one file (each of which appears in its own tab). The sketches in your sketchbook can be opened from the **File > Sketchbook** menu or from the **Open** button on the toolbar. These can be normal Arduino code files (no visible extension), C files (.c extension), C++ files (.cpp), or header files (.h).

Uploading

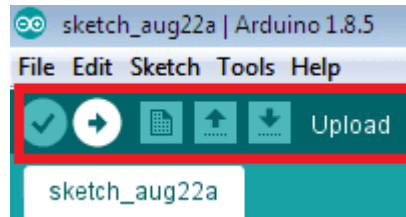


Figure: 3.68 Uploading

Before uploading your sketch, you need to select the correct items from the **Tools > Board** and **Tools > Port** menus. The boards are described below. On the Mac, the serial port is probably something like **/dev/tty.usbmodem241** (for an Uno or Mega2560 or Leonardo) or **/dev/tty.usbserial-1B1** (for a Duemilanove or earlier USB board), or **/dev/tty. USA19QW1b1P1.1** (for a serial board connected with a Key span USB-to-Serial adapter). On Windows, it's probably COM1 or COM2 (for a serial board) or COM4, COM5, COM7, or higher (for a USB board) -to find out, you look for USB serial device in the sports section of the Windows Device Manager. On Linux, it should be **/dev/ttyACMx**, **/dev/ttyUSBx** or similar. Once you have selected the correct serial port and board, press the upload button in the toolbar or select the **Upload** item from the **Sketch** menu. Current Arduino boards will reset automatically and begin the upload. With older boards (pre-Diecimila) that lack auto-reset, you will need to press the reset button on the board just before starting the upload.

When you upload a sketch, you're using the Arduino **bootloader**, a small program that has been loaded on to the microcontroller on your board. It allows you to upload code without using any additional hardware. The bootloader is active for a few seconds when the board resets; then it starts whichever sketch was most recently uploaded to the microcontroller. The boot loader will blink the on-board (pin 13) LED when it starts (i.e., when the board resets).

Libraries provide extra functionality for use in sketches, e.g., working with

hardware or manipulating data. To use a library in a sketch, select it from the **Sketch > Import Library** menu. This will insert one or more **#include** statements at the top of the sketch and compile the library with your sketch. Because libraries are uploaded to the board with your sketch, they increase the amount of space it takes up. If a sketch no longer needs a library, simply delete its **#include** statements from the top of your code.

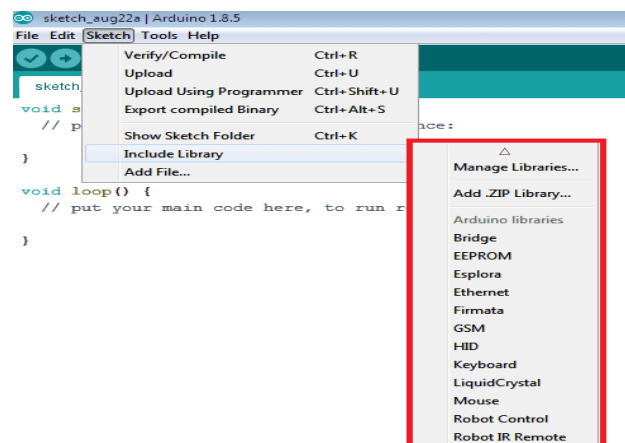


Figure: 3.69 Libraries

There is a list of libraries in the reference. Some libraries are included with the Arduino software. Others can be downloaded from a variety of sources or through the Library Manager. Starting with version 1.0.5 of the IDE, you do can import a library from a zip file and use it in an open sketch. See these instructions for installing a third-party library.

Third-Party Hardware

Support for third-party hardware can be added to the **hardware** directory of your sketchbook directory. Platforms installed there may include board definitions (which appear in the board menu), core libraries, bootloaders, and programmer definitions. To install, create the **hardware** directory, then unzip the third-party platform into its own sub-directory. (Don't use "Arduino" as the sub-directory name or you'll override the built-in Arduino platform.) To uninstall, simply delete its directory. For details on creating packages for third-party

hardware, see the Arduino IDE 1.5 3rd party Hardware specification.

Serial Monitor

This displays serial sent from the Arduino or Genuino board over USB or serial connector. To send data to the board, enter text and click on the "send" button or press enter. Choose the baud rate from the drop-down menu that matches the rate passed to **Serial. Begin** in your sketch. Please note that the Serial Monitor does not process control characters; if your sketch needs a complete management of the serial communication with control characters, you can use an external terminal program and connect it to the COM port assigned to your Arduino board. You can also talk to the board from Processing, Flash, MaxMSP, etc. (see the interfacing page for details).

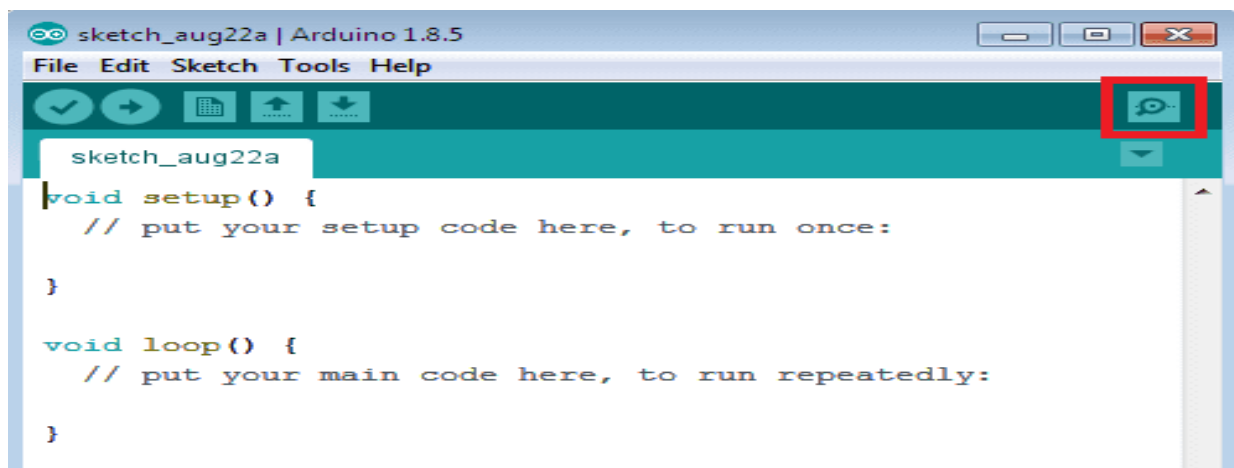


Figure: 3.70 Arduino 1.8.5 Board

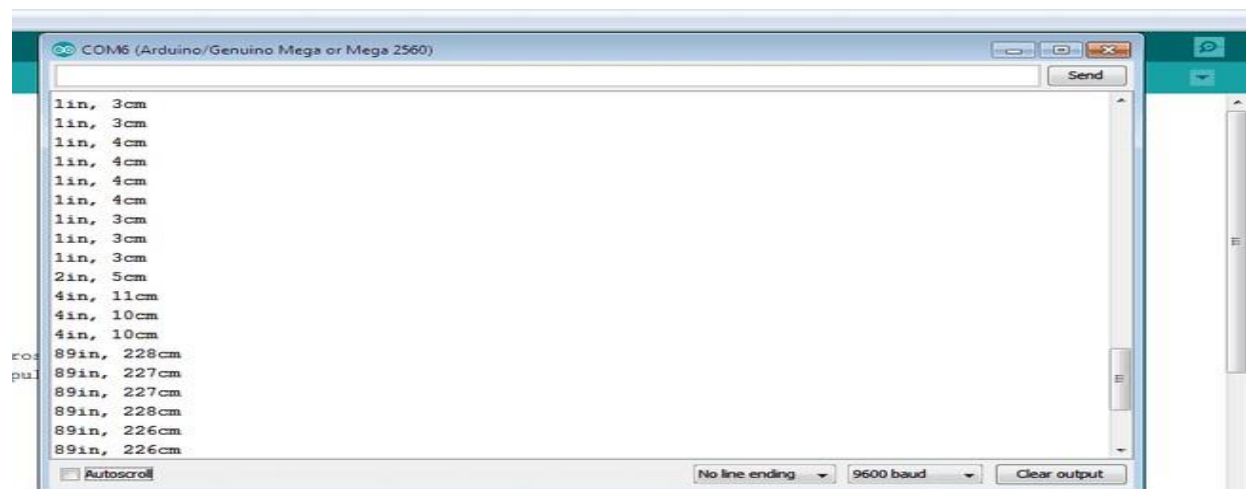


Figure: 3.71 Com5 Port

Preferences

Some preferences can be set in the preferences dialog (found under the **Arduino** menu on the Mac, or **File** on Windows and Linux). The rest can be found in the preferences file, whose location is shown in the preference dialog.

Language Support



Figure: 3.72 Language Support

Since version 1.0.1, the Arduino Software (IDE) has been translated into 30+ different languages. By default, the IDE loads in the language selected by your operating system. (Note: on Windows and possibly Linux, this is determined by the locale setting which controls currency and date formats, not by the language the operating system is displayed in.) If you would like to change the language manually, start the Arduino Software (IDE) and open the **Preferences** window. Next to the **Editor Language** there is a dropdown menu of currently supported languages.

Select your preferred language from the menu, and restart the software to use the selected language. If your operating system language is not supported, the Arduino Software (IDE) will default to English. You can return the software to its default setting of selecting its language based on your operating system by selecting

System Default from the **Editor Language** drop-down. This setting will take effect when you restart the Arduino Software (IDE). Similarly, after changing your operating system's settings, you must restart the Arduino Software (IDE) to update it to the new default language.



```

// Example
#include <Wire.h>

const int MPU = 0x6B;

int16_t AccX, AccY, AccZ, Temp, GyroX, GyroY, GyroZ;

const int MPU1 = 0x6B;

int16_t AccX, AccY, AccZ, Temp, GyroX, GyroY, GyroZ;

int minVal = 255;
int maxVal = 0;
double a;
double g;
double t;

void setup()
{
  Serial.begin(9600);
  Serial.println("STARTING");
  Serial.println("I2C: ,X,Y,Z");
  Serial.println("XXXXXXXXXX");
  Wire.begin();
  Wire.beginTransmission(MPU);
  Wire.write(0x6B);
  Wire.endTransmission();
  Wire.requestFrom(MPU, 1);
  delay(1000);
  Wire.begin();
  Wire.beginTransmission(MPU);
  Wire.write(0x6B);
  Wire.endTransmission();
  Wire.requestFrom(MPU, 1);
  delay(1000);
}

```

Figure 3.73 C language Source Code

CHAPTER 4

SYSTEM DESIGN

CHAPTER 4

SYSTEM DESIGN

4.1 BLOCK DIAGRAM

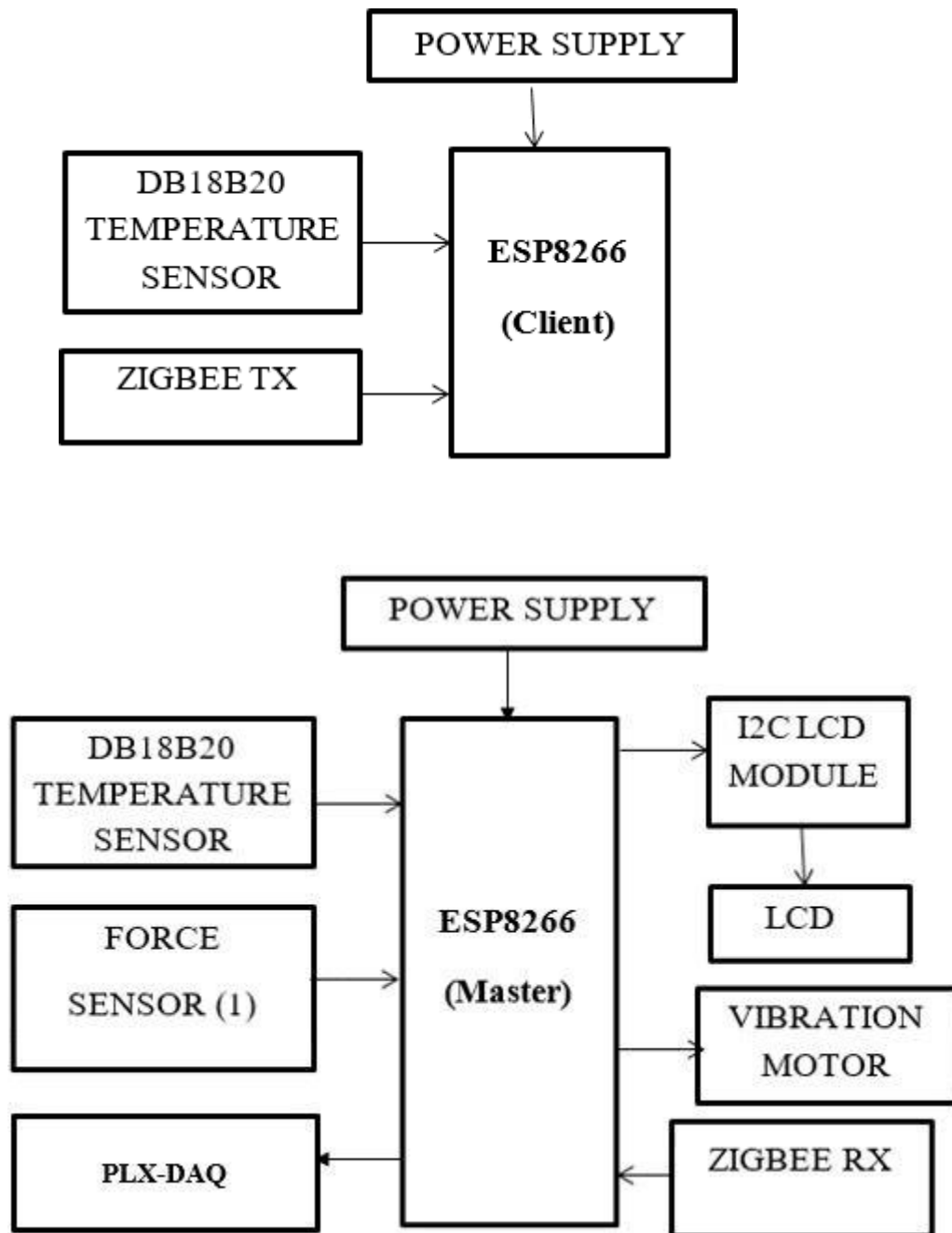


Figure 4.1 Block Diagram for Varicose Veins Disease Detection using BAN

4.2 ENTITY RELATIONSHIP DIAGRAM

An entity relationship model, also called an entity-relationship (ER) diagram, is a graphical representation of entities and their relationships to each other, typically used in computing in regard to the organization of data within databases or information systems.

In an ER diagram, symbols are commonly used to represent the types of information. Most diagrams will use the following symbols:

- Boxes represent entities.
- Diamonds represent relationships
- Circles (ovals) represent attributes.

ENTITY

An entity is a piece of data-an object or concept about which data is stored. An entity is a thing that exists either physically or logically.

Relationships Between Entities:

A relationship is how the data is shared between entities.

There are three types of relationships between entities:

1. One-to-One

One instance of an entity is associated with one other instance of another entity.

2. One-to-Many

One instance of an entity is associated with zero, one or many instances of another entity, but for one instance of the other entity B there is only one instance of former entity.

3. Many-to-Many

One instance of an entity is associated with one, zero or many instances of another

entity, and one instance of the other entity is also associated with one, zero or many instances of former entity.

4.2.1 ER Diagram for Varicose Veins Disease Detection Using BAN

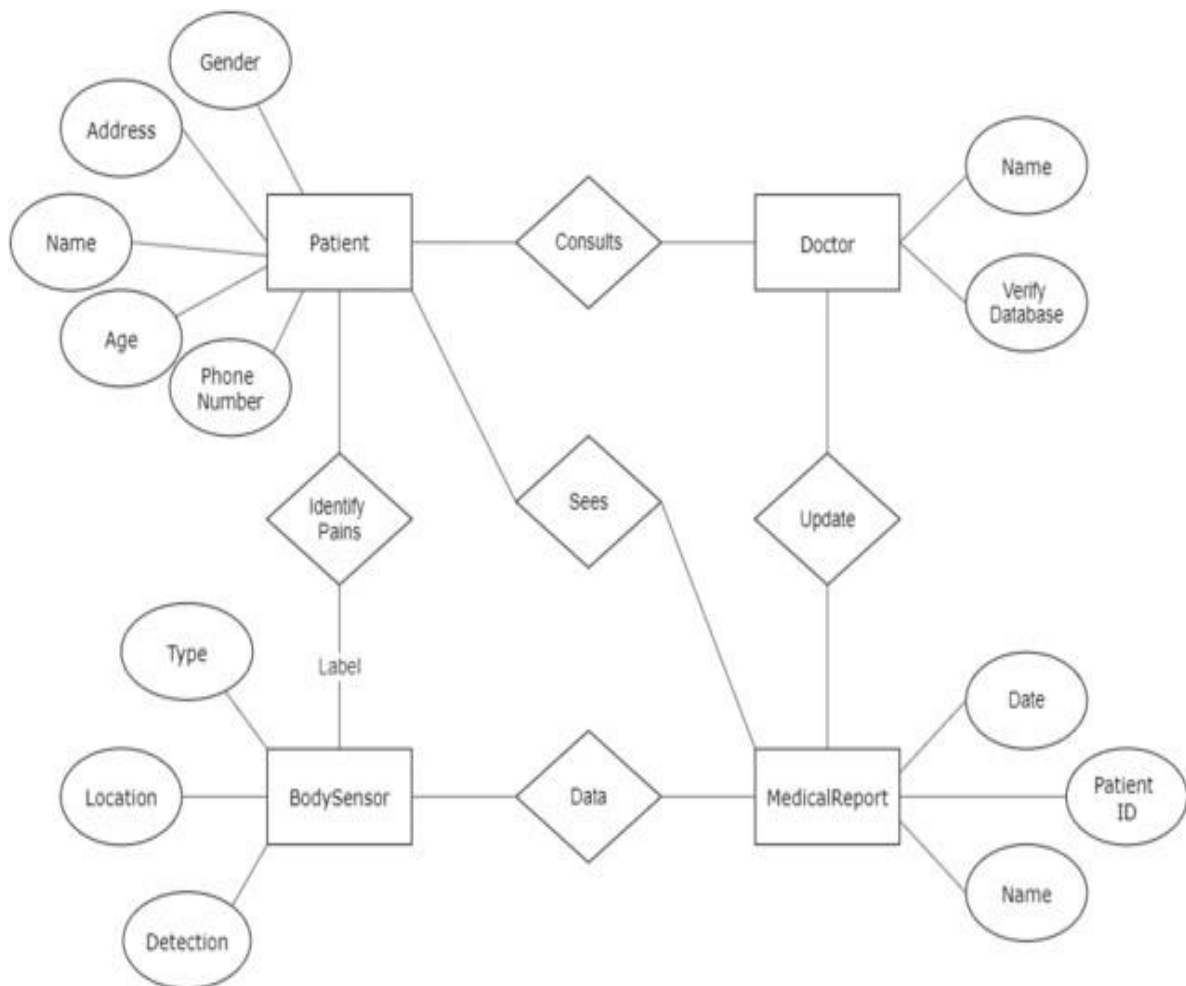


Figure 4.2 ER Diagram for Varicose Veins Disease Detection

4.3 DATA FLOW DIAGRAM


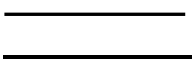
A Data Flow Diagram (DFD) is traditional visual representation of the information flows within a system. A neat and clear DFD can depict a good amount of the system requirements graphically. It can be manual, automated, or combination of both. It shows how information enters and leaves the system, what changes the information and where information is stored. The purpose of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communications tool between a systems analyst and any person who plays a part

in the system that acts as the starting point for redesigning a system. It is usually beginning with a context diagram as the level 0 of DFD diagram, a simple representation of the whole system. To elaborate further from that, we drill down to a level 1 diagram with lower-level functions decomposed from the major functions of the system. This could continue to evolve to become a level 2 diagram when further analysis is required.

Components of DFD

DFDs are constructed using four major components:

- External entities are represented by squares and specify the source of data as input to the system. They are also the destination of system data. External entities can be called Data stores outside the system.
- Data stores specify storage of data within the system, for example, computer files or databases. An open-ended box represents a data, which implies store data at rest or a temporary repository of data.
- Processes represent activities in which data is manipulated by being stored or retrieved or transferred in some way. In other words, the process of transformation of input data into output data. Circles stand for a process that converts data into information.
- Data flow represents the movement of data from one component to the other. An arrow (\Rightarrow) identifies data flow ie. data in motion. It is a pipeline through which information flows. Data flows are generally shown as one-way only. Data flows between external entities are shown as dotted lines.

SYMBOLS	NAME	DESCRIPTION
	External Entity	They are outside the system and they either supply input data into the system or use system output.
	Data Store	Huge collection of data and used for storage purpose.

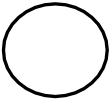

	Process	Shows transformation manipulation of Data Flows within the system.
	Data Flow	Shows flow of information from input to output.

Table 4.1 Different Symbols used in Data Flow Diagram with their functions

Data Flow Diagrams are either Logical or Physical.

Physical DFD – It is an implementation-dependent view of the current system, showing what tasks are carried out and how they are performed. Physical characteristics can include: Names of people, form and document names or numbers, names of departments, master and transaction files, equipment and devices used, locations, names of procedures.

Physical DFD offers the following advantages:

- It describes each process in detail than Logical DFDs.
- It can identify temporary data stores.
- It specifies the actual name of the files in use.
- It describes whether the process is manual or automated.
- It contains additional controls to ensure the processes are done optimally.

Logical DFD – It is an implementation-independent view of the system, focusing on the flow of data between processes without regard for the specific devices, storage locations or people in the system.

Logical DFDs offers the following advantages:

- Communication with users is better than in physical DFDs.
- Stable System, Good flexibility and maintenance
- Better understanding of business by analysts.
- Elimination of redundancies and better creation of physical model.

4.3.1 Level – 0 DFD:

The outermost level (Level 0) is concerned with how the system interacts with the outside world. All the components within the system boundary are included within a single process box in the DFD. External entities lie outside the system boundary; internal entities will become locations for processes. The level – 0 DFD of the system sparingly describes the entities Patient, Doctor and Medical Report are related to the system, where the entities log into the system and collaborate data.

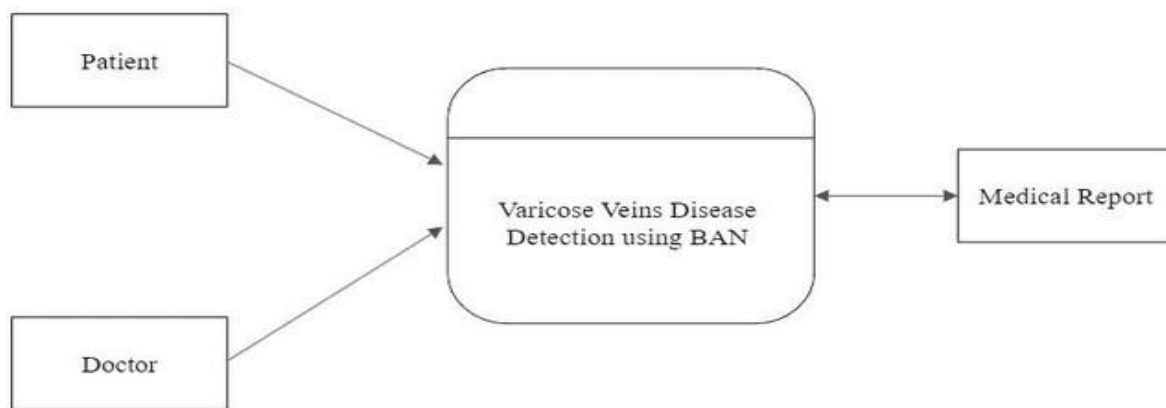


Figure 4.3 Level – 0 DFD

4.3.2 Level – 1 DFD:

The patient wears a temperature sensor as part of the BAN system. The temperature sensor measures the patient's body temperature and converts it into a digital signal. The digital signal is transmitted wirelessly to a central hub or gateway device. The central hub receives the temperature data from the sensor and forwards it to the Varicose Veins Disease Detection system. The Varicose Veins Disease Detection system uses the temperature data along with other physiological data to analyses for patterns or anomalies that indicate the presence of varicose veins disease. If the system detects signs of varicose veins disease, it alerts the patient and/or their healthcare provider.

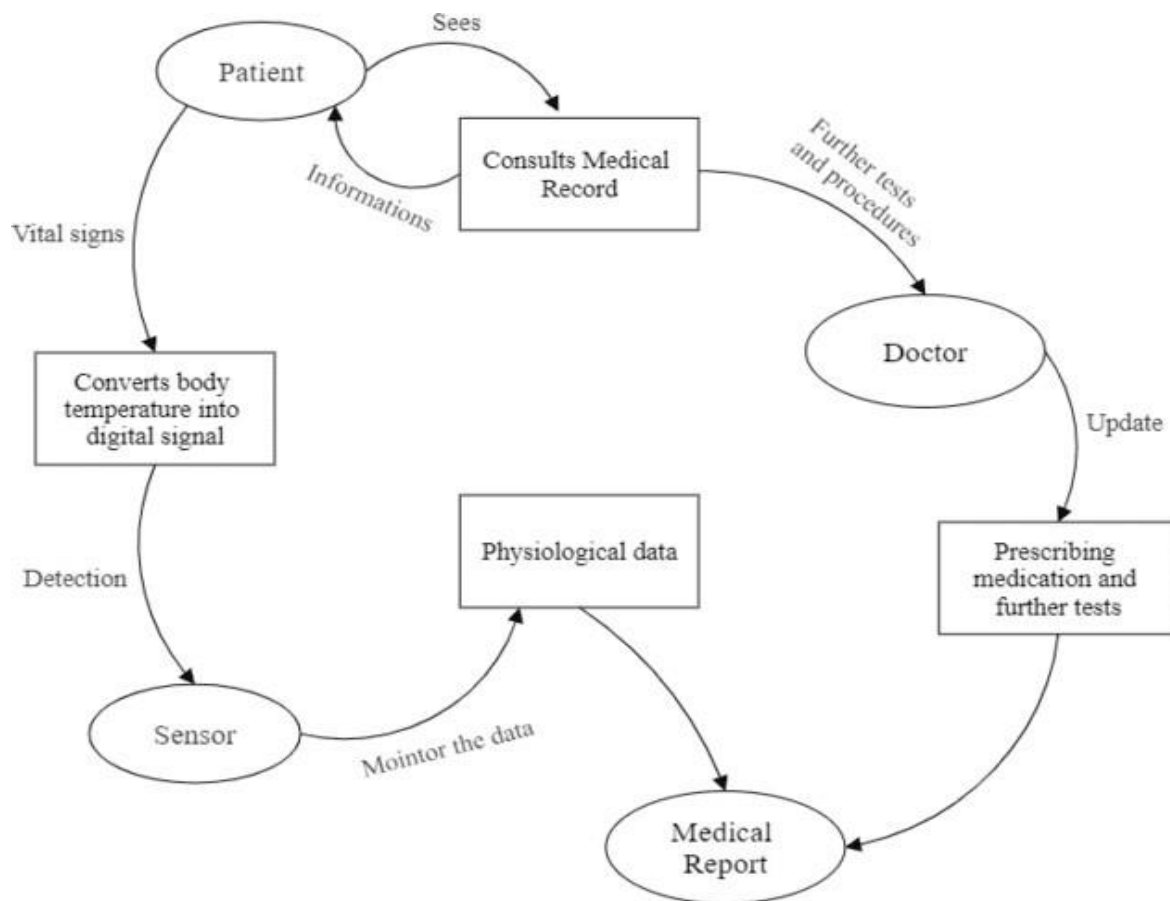


Figure 4.4 Level – 1 DFD

4.4 UML DIAGRAMS

UML stands for Unified Modeling Language. It's a rich language to model software solutions, application structures, system behavior and business processes. Unified Modeling Language is a standard visual modeling language intended to be used for

- Modeling Business and similar processes
- Analysis, Design, and Implementation of Software-based Systems

UML is a common language for Business Analysts, Software Architects and Developers used to describe, specify, design, and document existing or new business processes, structure and behavior of artifacts of software systems. Once Specifications are explained that process:

- Provides guidance as to the order of a team's activities,

- Specifies what artifacts should be developed,
- Directs the tasks of individual developers and the team as a whole, and
- Offers criteria for monitoring and measuring a project's products and activities.

UML is intentionally process independent and could be applied in the context of different processes. Still, it is most suitable for use case driven, iterative and incremental development processes. An example of such process is Rational Unified Process (RUP). UML is not complete, and it is not completely visual. Given some UML diagram, we cannot be sure to understand depicted part or behavior of the system from the diagram alone. Some information could be intentionally omitted from the diagram, some information represented on the diagram could have different interpretations, and some concepts of UML have no graphical notation at all, so there is no way to depict those on diagrams. For example, semantics of multiplicity of actors and multiplicity of use cases on use case diagrams is not defined precisely in the UML specification and could mean either concurrent or successive usage of use cases.

Name of an abstract classifier is shown in italics while final classifier has no specific graphical notation, so there is no way to determine whether classifier is final or not from the diagram. There are two main categories, structured diagrams and behavioral diagrams.

Structure Diagrams:

Structure diagrams show the things in the modeled system. In a more technical term, they show different objects in a system. Structure diagrams include Class diagram, Component diagram etc.

Behavioral Diagram:

Behavioral diagrams show what should happen in a system. They describe how the objects interact with each other to create a functioning system.

Behavioral diagrams include Use case diagram, Activity diagram, Sequence diagram etc.

4.4.1 USE CASE DIAGRAM

A use case diagram is a graphic depiction of the interactions among the elements of a system. A use case is a methodology used in system analysis to identify, clarify, and organize system requirements. The use cases, which are the specific roles played by the actors within and around the system

. In general use case diagrams are used for:

- Analyzing the requirements of a system.
- High-level visual software designing.
- Capturing the functionalities of a system.
- Modelling the basic idea behind the system.
- Forward and reverse engineering of a system using various test cases.

Use Case Diagram Notations:

Actor

Actors are usually individuals involved with the system defined according to their roles.

Use Case

A use case describes how actors uses a system to accomplish a particular goal. ...

Relationship

The relationships between and among the actors and the use cases.

System Boundary

A system boundary is a rectangle that you can draw in a use-case diagram to separate the use cases that are internal to a system from the actors that are

external to the system. It does not add semantic value to the model.

4.4.1.1 Use Case Diagram for Varicose Veins Detection Using BAN

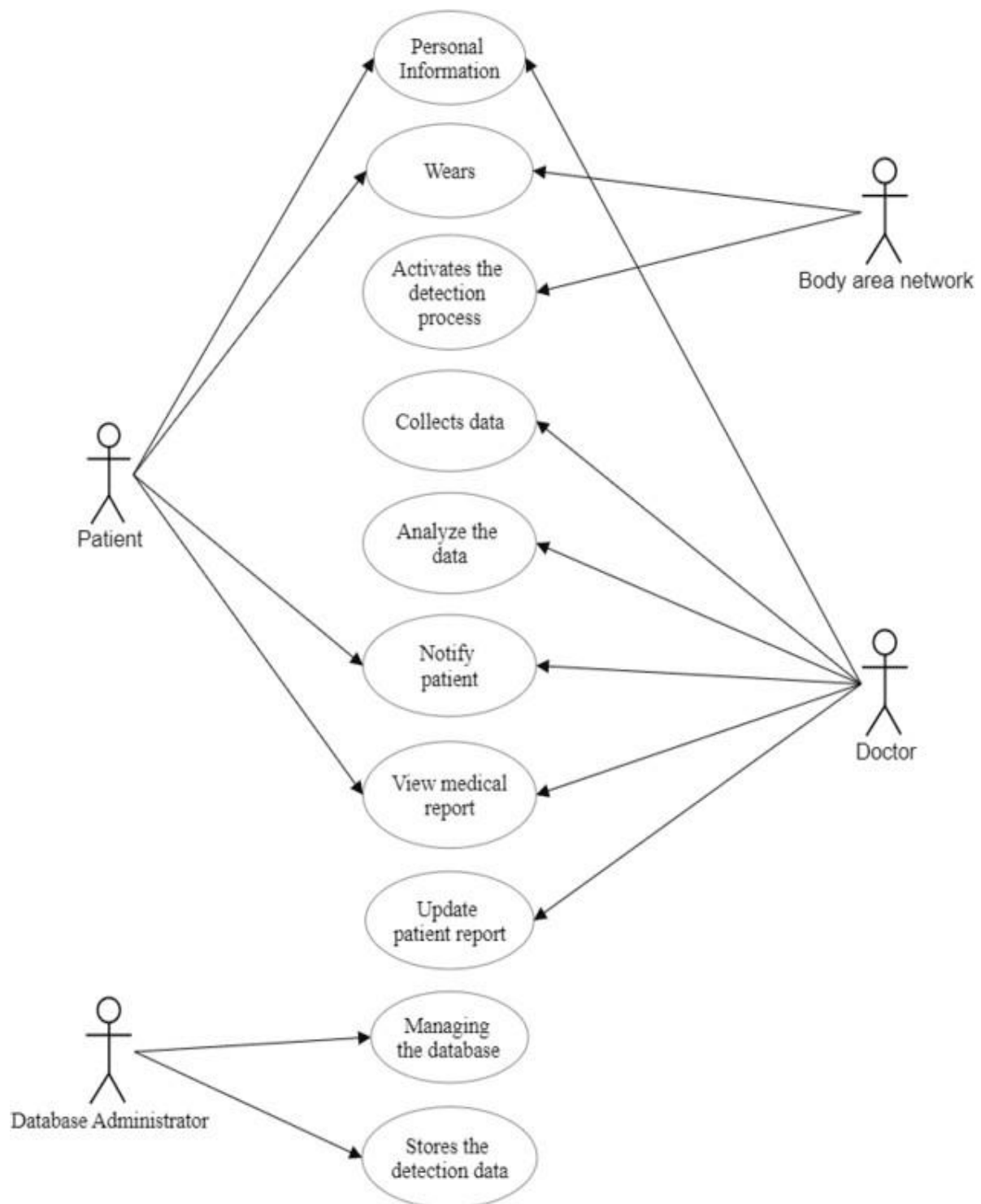


Figure 4.5 Use Case Diagram for Varicose Veins Detection Using BAN

4.4.2 CLASS DIAGRAM

Class diagrams are the main building block of any object-oriented solution. It shows the classes in a system, attributes, and operations of each class and the relationship between each class. In most modeling tools, a class has three parts. Name at the top, attributes in the middle and operations or methods at the bottom.

A class is a blueprint for an object. In the diagram, classes are represented with boxes that contain three compartments:

- The top compartment contains the name of the class. It is printed in bold and cantered, and the first letter is capitalized.
- The middle compartment contains the attributes of the class. They are left-aligned and the first letter is lowercase.
- The bottom compartment contains the operations the class can execute. They are also left-aligned and the first letter is lowercase.

In a large system with many related classes, classes are grouped together to create

class diagrams. Different relationships between classes are shown by different types of arrows.

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.

4.4.2.1 Class Diagram for Varicose Veins Detection Using BAN

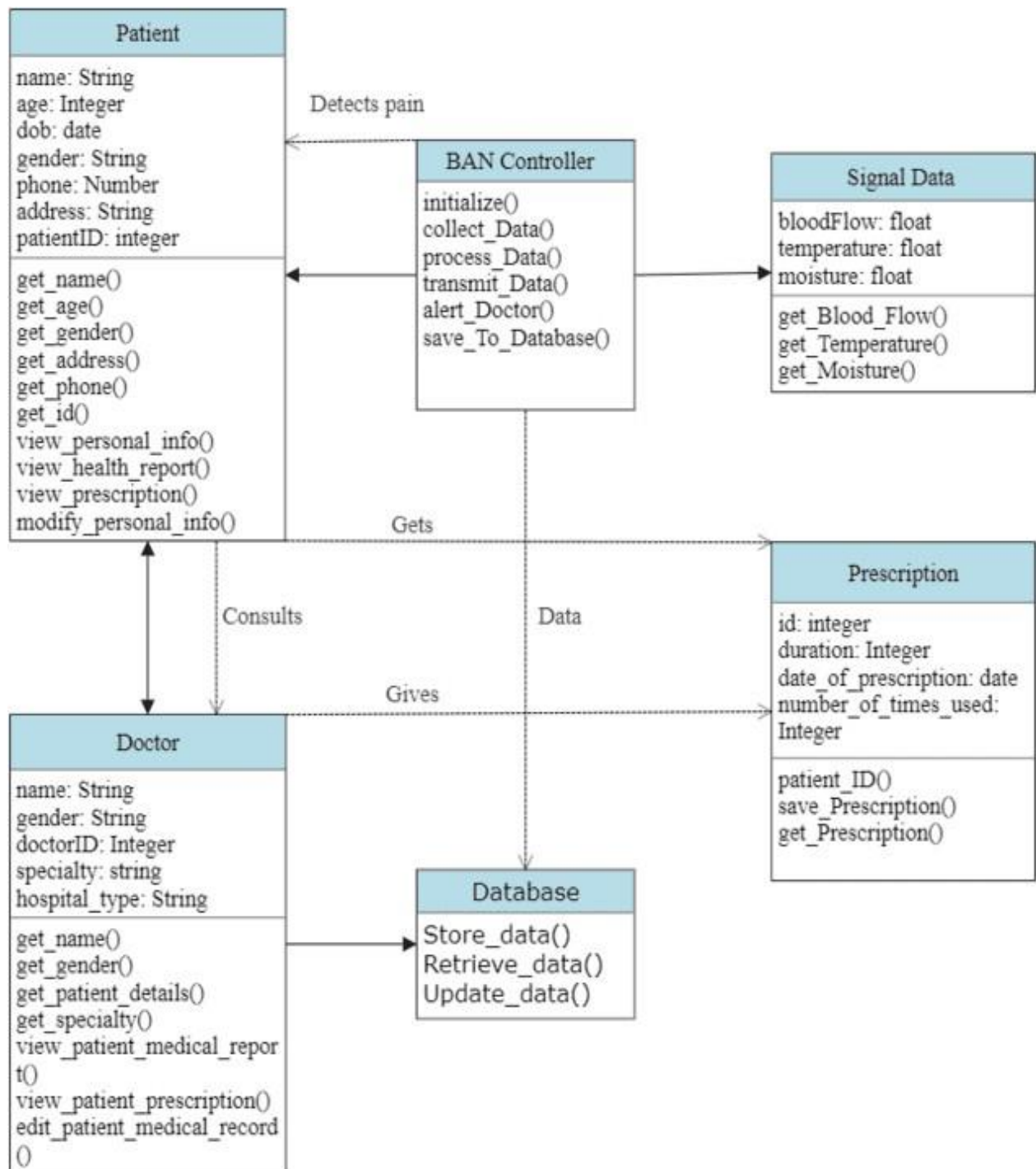


Figure 4.6 Class Diagram for Varicose Veins Detection Using BAN

4.4.3 ACTIVITY DIAGRAM

Activity diagrams represent workflows in a graphical way. They can be used to describe the business workflow or the operational workflow of any component in a system. Sometimes activity diagrams are used as an alternative to State machine diagrams.

In the Unified Modeling Language, activity diagrams are intended to model both computational and organizational processes (i.e., workflows), as well as the data flows intersecting with the related activities. Although activity diagrams primarily show the overall flow of control, they can also include elements showing the flow of data between activities through one or more data stores. Activity diagrams can be regarded as a form of a structured flowchart combined with traditional data flow diagram. Typical flowchart techniques lack constructs for expressing concurrency.

Activity diagrams are constructed from a limited number of shapes, connected with arrows. Arrows run from the start towards the end and represent the order in which activities happen.

The most important shape types are:

- ellipses represent actions.
- diamonds represent decisions.
- bars represent the start (split) or end (join) of concurrent activities.
- a black circle represents the start (initial node) of the workflow.
- an encircled black circle represents the end (final node).

4.4.3.1 Activity Diagram for Varicose Veins Detection Using BAN

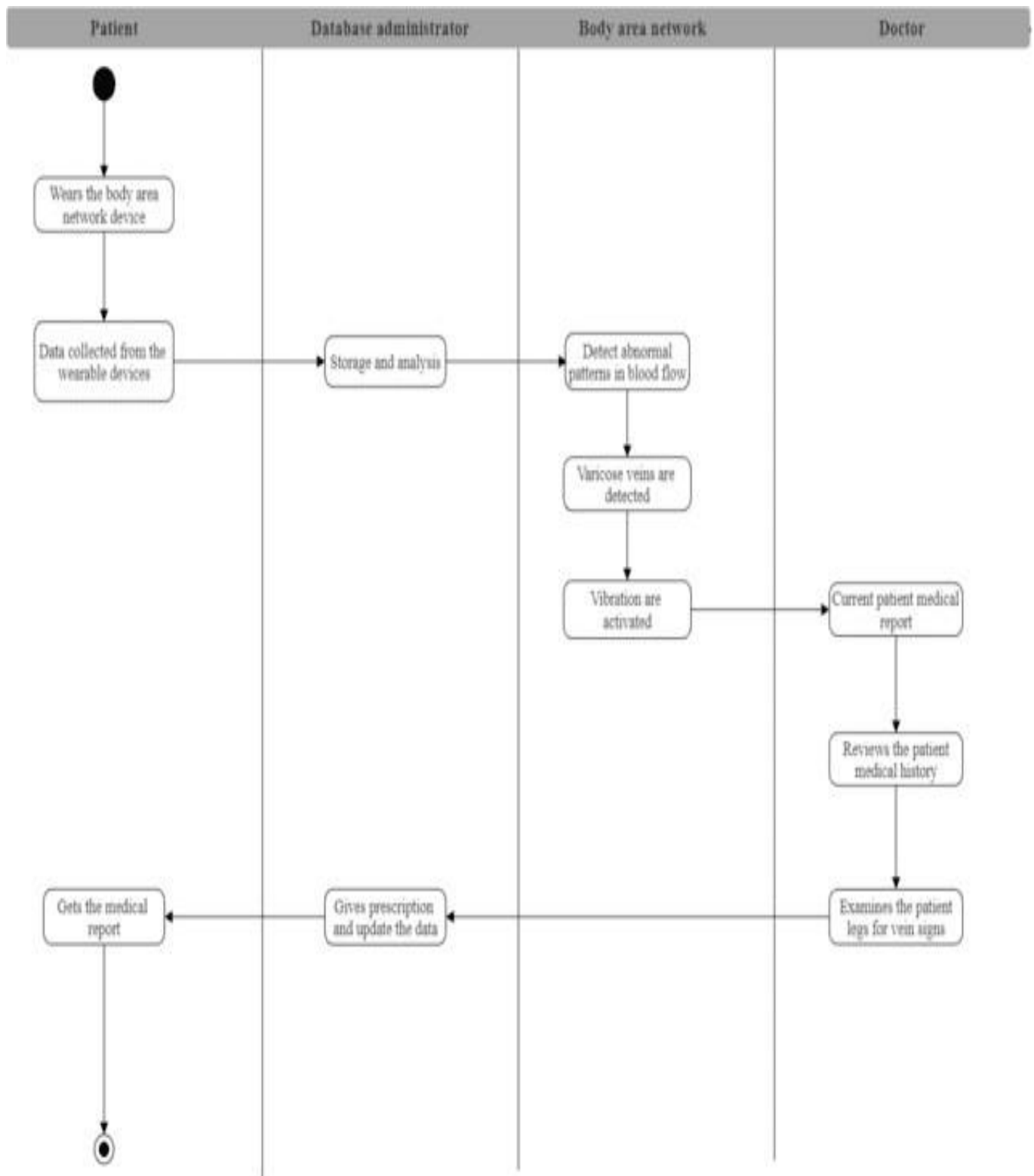


Figure 4.7 Activity Diagram for Varicose Veins Detection Using BAN

CHAPTER 5

SYSTEM MODULE

CHAPTER 5

SYSTEM MODULE

5.1 Module Description

The delay in adequate treatment or hospitalization, due to the long waiting times in an emergency reception room, is the leading cause of patient mortality in hospitals. This situation can be improved by identifying any deterioration of patient's health conditions at an early stage. Continuous real- time monitoring of patients inside the emergency waiting room is crucial for a rapid intervention in order to minimize severe medical complications. This can be achieved by sending timely data from the health monitoring sensors, using wireless body area networks (WBANs) to a central computer “This work was supported by the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No. 764461 (VisIoN). It is also based upon work from COST Action CA19111 NEWFOCUS, supported by COST (European Cooperation in Science and Technology)”.

5.2 Module Names

- Temperature Monitor
- Rectify Varicose Veins

5.2.1 Temperature Monitor

In this project our main motive is to relief the varicose veins for the time. The thermistor used for to identify the varicose veins, that time temperature will be increase. So, the thermistor will be identified. That time the ZigBee will be send the ZigBee.

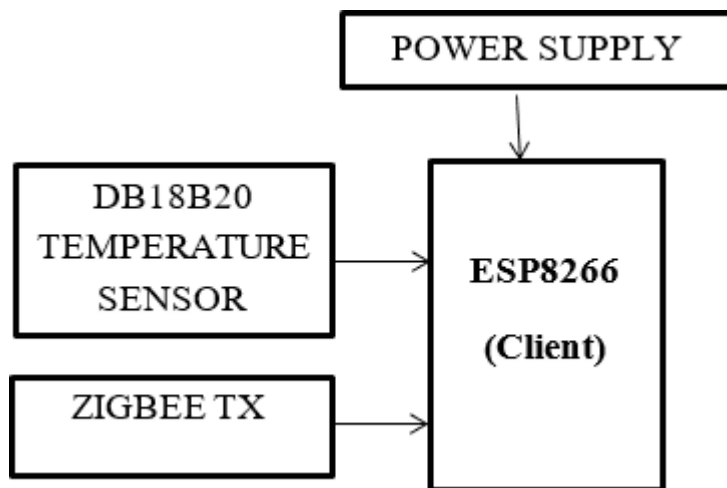


Figure: 5.1 Temperature Monitor

5.2.2 Rectify Varicose Veins

The ZigBee receiver to receive the signal from thermistor and the vibration motor will be vibrate and rectify the varicose veins.

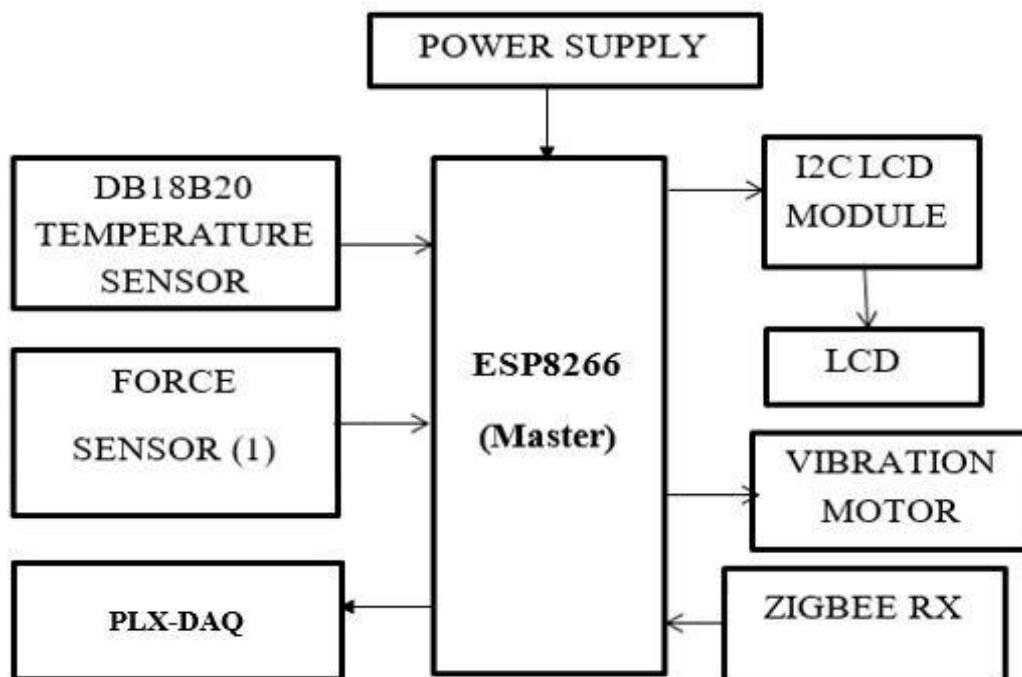


Figure: 5.2 Rectify Varicose Veins

CHAPTER 6

SYSTEM TESTING

CHAPTER 6

SYSTEM TESTING

The testing approach document is designed for Information and Technology Services' upgrades to PeopleSoft. The document contains an overview of the testing activities to be performed when an upgrade or enhancement is made, or a module is added to an existing application. The emphasis is on testing critical business processes, while minimizing the time necessary for testing while also mitigating risks. It's important to note that reducing the amount of testing done in an upgrade increases the potential for problems after go-live. Management will need to determine how much risk is acceptable on an upgrade-by-upgrade basis. System testing is simply testing the system as a whole; it gets all the integrated modules of the various components from the integration testing phase and combines all the different parts into a system which is then tested. Testing is then done on the system as all the parts are now integrated into one system the testing phase will now have to be done on the system to check and remove any errors or bugs. In the system testing process the system will be checked not only for errors but also to see if the system does what was intended, the system functionality and if it is what the end user expected.

There are various tests that need to be conducted again in the system testing which include:

- Test Plan
- Test Case
- Test Data

If the integration stage was done accurately then most of the test plan and test cases would already have been done and simple testing would only have to be done in order to ensure there are no bugs because this will be the final product. As in the integration stage, the above steps would need to be re-done as now we have integrated all modules into one system.

6.1.1 Unit Testing

In computer programming, unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures are tested to determine if they are fit for use. In object-oriented programming, a unit is often an entire interface, such as a class, but could be an individual method. Unit tests are short code fragments created by programmers or occasionally by white box testers during the development process. Ideally, each test case is independent from the others. Substitutes such as method stubs, mock objects, fakes, and test harnesses can be used to assist testing a module in isolation. Unit tests are typically written and run by software developers to ensure that code meets its design and behaves as intended.

6.1.2 Integration Testing

It is defined to be set of interaction, all defined interaction among components needs to be tested. The architecture and design can give the detail of interaction within the system and explain how they interact with each other. The types of integration testing are Top-Down testing, Bottom-Up testing, Bi-Directional testing and System integration.

6.2 TEST CASES AND REPORTS

6.2.1 Upper Body Module

Test case 1: Temperature Sensor 1:

Test Case ID	Test Case Description	Input Data	Expected Output	Actual Output	Pass/Fail	Remarks
T1	To check the upper body temperature	Valid temperature	To show valid upper body temperature	To show valid upperbody temperature	Pass	Efficient
T2	To check the upper body temperature	Invalid temperature	Shows Invalid temperature	Shows Invalid temperature	Pass	Efficient

Table 6.1 Temperature Sensor 1

Test case 2: NodeMCU 1:

Test Case ID	Test Case Description	Input Data	Expected Output	Actual Output	Pass/Fail	Remarks
T1	To check the transmission data	Temperature data from temperature sensor	Transmit data to the receiver in module 2	Transmit data to the receiver in module 2	Pass	Efficient
T2	To check the transmission data	Invalid temperature data from temperature sensor	Transmit data to the receiver in module 2	Transmit data to the receiver in module 2	Pass	Efficient

Table 6.2: NodeMCU 1

6.2.2 Lower Body Module

Test case 3: Temperature Sensor 2:

Test Case ID	Test Case Description	Input Data	Expected Output	Actual Output	Pass/Fail	Remarks
T1	To check the lower body temperature	Valid temperature	To show valid lower body temperature	To show valid lower body temperature	Pass	Efficient
T2	To check the lower body temperature	Invalid temperature	Shows Invalid temperature	Shows Invalid temperature	Pass	Efficient

Table 6.3 Temperature Sensor 2

Test case 4: NodeMCU 2:

Test Case ID	Test Case Description	Input Data	Expected Output	Actual Output	Pass/Fail	Remarks
T1	To check the received data from the transmission	Temperature data from temperature sensor	Received data from transmitter in module 1	Received data from transmitter in module 1	Pass	Efficient
T2	To check the received data from the transmission	Invalid temperature data from temperature sensor	Received data from transmitter in module 1	Received data from transmitter in module 1	Pass	Efficient

Table 6.4 NodeMCU 2

Test case 5: Pressure Sensor:

Test Case ID	Test Case Description	Input Data	Expected Output	Actual Output	Pass/Fail	Remarks
T1	To check the pressure from varicose veins	Low pressure from the expansion of the vein	Pressure measures accurately	Pressure measures accurately	Pass	Efficient
T2	To check the pressure from varicose veins	Medium pressure from the expansion of the vein	Pressure measures accurately	Pressure measures accurately	Pass	Efficient
T3	To check the pressure from varicose veins	High pressure from the expansion of the vein	Pressure measures accurately	Pressure measures accurately	Pass	Efficient

Table 6.5 Pressure Sensor**6.2.3 Database Module****Test case 6: PLX-DAQ:**

Test Case ID	Test Case Description	Input Data	Expected Output	Actual Output	Pass/Fail	Remarks
T1	To check the software is ON and store the data	Software is ON module works properly	Received data accurately	Received data accurately	Pass	Efficient
T1	To check the software is ON and store the data	Software is OFF module works properly	Received data accurately	Received data accurately	Pass	Efficient

Table 6.6 PLX-DAQ

Test Results:

All the test cases mentioned above passed successfully. No defects encountered during the testing process.

CHAPTER 7

CONCLUSION

CHAPTER 7

6. CONCLUSION AND FUTURE ENHANCEMENT

Conclusion

In conclusion, the use of Body Area Networks (BAN) in the detection and treatment of varicose veins is a promising approach. The proposed system that uses a thermistor sensor and a pressure sensor to detect abnormalities and provide temporary relief to patients suffering from varicose veins is a non-invasive, easy-to-use, and effective approach. The system collects real time data that can be used for further analysis and diagnosis, enabling healthcare providers to provide personalized and effective treatment options. However, further research is needed to explore the full potential of this system and to develop more advanced and automated treatment options.

Future enhancement

In the future, the proposed system can be enhanced by incorporating additional sensors and features to provide more personalized treatment options. For example, the system can be equipped with a heart rate monitor to detect changes in heart rate, which can be used to monitor the patient's stress levels and provide stress relieving treatments. The system can also be integrated with a mobile app that enables patients to monitor their symptoms and receive personalized treatment recommendations. Furthermore, the system can be enhanced by using machine learning algorithms to analyze the collected data and provide more accurate diagnosis and treatment recommendations. With further advancements in technology, BAN-based systems have the potential to revolutionize the detection and treatment of varicose veins and other medical conditions.

CHAPTER 8

SCREENSHOTS

CHAPTER 8

SCREENSHOTS

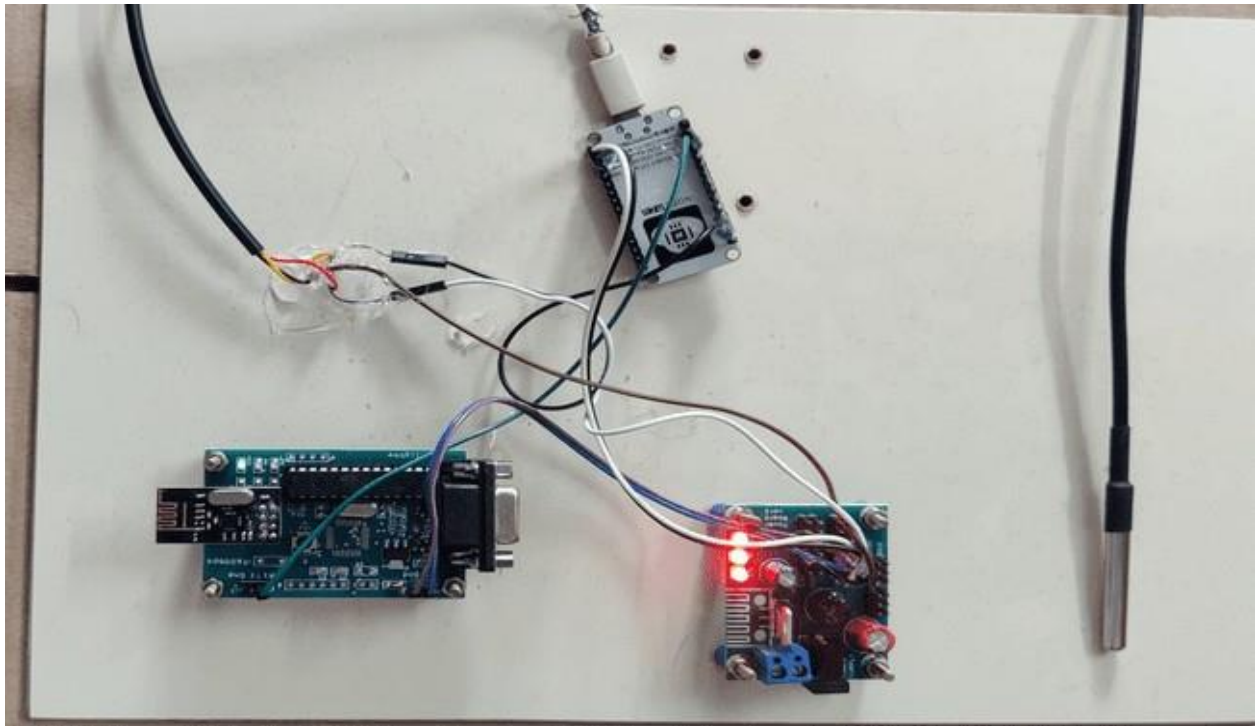


Figure 8.1 Upper Body Module

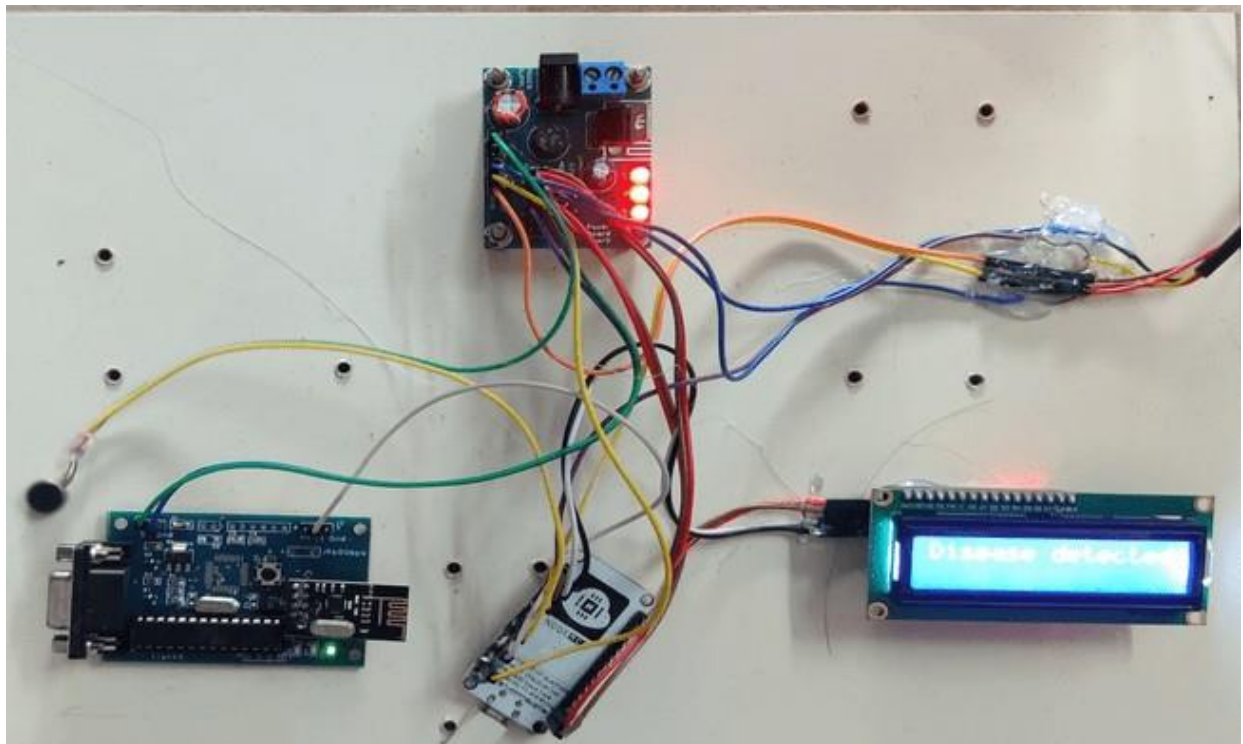


Figure 8.2 Lower Body Module



Figure 8.3 Varicose vein disease detected

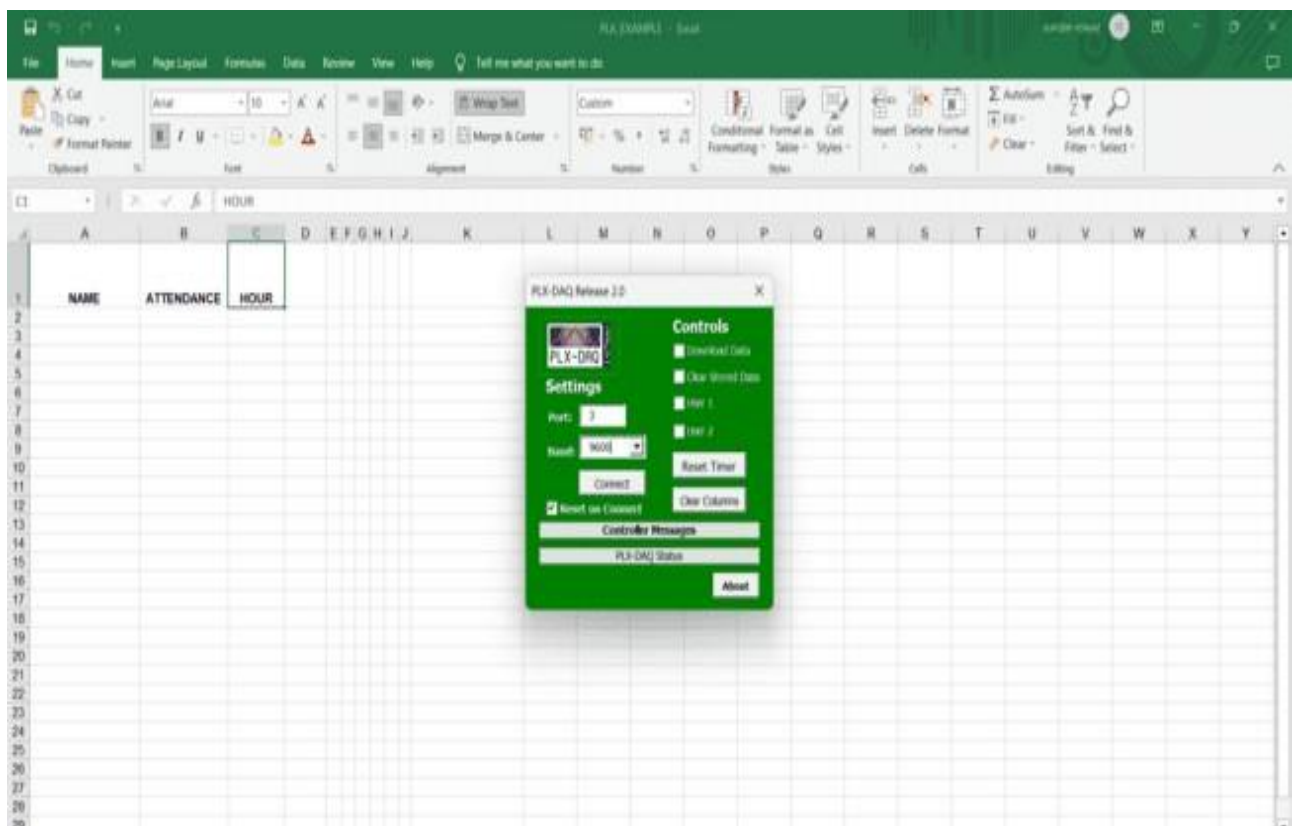


Figure 8.4 Database Module

CHAPTER 9

APPENDIX

CHAPTER 9

APPENDIX

SAMPLE CODE:

```
#include<Wire.h>

const int MPU = 0x68;
int16_t AcX, AcY, AcZ, Tmp, GyX, GyY, GyZ;
const int MPU1 = 0x69;
int16_t BcX, BcY, BcZ, Tmp1, BGyX, BGyY, BGyZ;
int minVal = 265;
int maxVal = 402;
double x;
double y;
double z;
void setup ()
{
  Serial.begin(9600);
  Serial.println("CLEARDATA");
  Serial.println("LABEL, X, Y, Z");
  Serial.println("RESETTIMER");
  Wire.begin();
  Wire.beginTransmission(MPU);
  Wire.write(0x6B);
  Wire.write(0);
  Wire.endTransmission(true);
  delay (1000);
  Wire.begin();
  Wire.beginTransmission(MPU1);
  Wire.write(0x6B);
  Wire.write(0);
```

```

Wire.endTransmission(true);
delay (1000);
}
void loop ()
{
Wire.beginTransmission(MPU);
Wire.write(0x3B);
Wire.endTransmission(false);
Wire.requestFrom(MPU, 12, true);
AcX = Wire.read() << 8 | Wire.read();
AcY = Wire.read() << 8 | Wire.read();
AcZ = Wire.read() << 8 | Wire.read();
int xAng = map (AcX, minVal, maxVal, -90, 90);
int yAng = map (AcY, minVal, maxVal, -90, 90);
int zAng = map (AcZ, minVal, maxVal, -90, 90);
x = RAD_TO_DEG * (atan2(-yAng, -zAng) + PI);
y = RAD_TO_DEG * (atan2(-xAng, -zAng) + PI);
z = RAD_TO_DEG * (atan2(-yAng, -xAng) + PI);
//Serial.println(x);
//Serial.println(y);
// Serial.println(z);
Serial.println((String)"DATA," + "," + x + "," + y + "," + z + ",");
//Serial.print(x);
// Serial.print(",");
//Serial.print(y);
//Serial.print(",");
//Serial.print(z);
// Serial.print(",");
delay (1000);}

```

BIBLIOGRAPHY

BIBLIOGRAPHY

- [1] Md Jahid Hasan, Mohammad Ali Khalighi, Volker Jungnickel, Luis Nero Alves, Bastien Be'chadergue, "An EnergyEfficient Optical Wireless OFDMA Scheme for Medical Body-Area Networks", IEEE Transactions on Green Communications and Networking.
- [2] O. Shumkov, M. Smagin, V. Nimaev, A. Sadovskii, "Radiofrequency ablation of varicose veins in obese patients", International Multiconference Bioinformatics of Genome Regulation and Structure\Systems Biology, 2018.
- [3] O. Haddad, M. A. Khaleghi, S. Zvanovec, and M. Adel, "Channel characterization and modeling for optical wireless body-area networks," IEEE Open Journal of the Communications Society, vol. 1, pp. 760–776, June 2020.
- [4] Ruizong Zhu , Huiping Niu , Ningning Yin , Tianjiao Wu , And Yapei Zhao, "Analysis of Varicose Veins of Lower Extremities Based on Vascular Endothelial Cell Inflammation Images and Multi-Scale Deep Learning", Dec. 2019.
- [5] S. Movassaghi, M. Abolhasan, J. Lipman, D. Smith, and A. Jamalipour, "Wireless body area networks: A survey," IEEE Communications Surveys & Tutorials, vol. 16, no. 3, pp. 1658– 1686, Mar. 2014.
- [6] A. S. Borde, G. V. Savrasov "Mathematical modeling of varicose veins ultrasound heating", IEEE International Conference on Microwaves, Antennas, Communications and Electronic Systems, (COMCAS), 2019.
- [7] B. Latre', B. Braem, I. Moerman, C. Blondia, and P. Demeester, "A survey on wireless body area networks," Wireless Networks, vol. 17, no. 1, pp. 1–18, Jan. 2011.

- [8] Gennady Victorovich Savrasov, Alexander Vasilyevich Gavrilenko, Anna Sergeevna Borde, Nikita Vladimirovich Belikov, Irina Vitalyevna Khaydukova, Irina Alexandrovna Seliverstova, "Comparison of Mechanical Parameters of the Great Saphenous Vein under Various Test Conditions", Ural Symposium on Biomedical Engineering, Radioelectronics and Information Technology (USBREIT), 2019.
- [9] O. Haddad, M. A. Khalighi, and S. Zvanovec, "Channel characterization for optical extra-WBAN links considering local and global user mobility," in *Broadband Access Communication Technologies XIV*, B. B. Dingel, K. Tsukamoto, and S. Mikroulis, Eds., vol. 11307, International Society for Optics and Photonics. SPIE, 2020, pp. 89 – 97.
- [10] Jing Liu, Bryan Yan, Shih-Chi Chen, Yuan-Ting Zhang, Fellow, Charles Sodini, Fellow, and Ni Zhao, "Non-Invasive Capillary Blood Pressure Measurement Enabling Early Detection and Classification of Venous Congestion", 2021.
- [11] M. J. Hasan, M. A. Khalighi, J. GarcaMrquez, and B. Bchadergue, "Performance analysis of optical-CDMA for uplink transmission in medical extra-WBANs," *IEEE Access*, vol. 8, pp. 171 672–171 685, Sept. 2020.
- [12] Z. Ghassemlooy, L. N. Alves, S. Zvanovec, and M. A. Khalighi, Eds., *Visible Light Communications: Theory and Applications*. CRC Press, 2017.
- [13] M. Giordani, M. Polese, M. Mezzavilla, S. Rangan, and M. Zorzi, "Toward 6G networks: Use cases and technologies," *IEEE Communications Magazine*, vol. 58, no. 3, pp. 55–61, Mar. 2020.
- [14] O. Haddad and M. A. Khalighi, "Enabling communication technologies for medical wireless body-area networks," in *Global LiFi Congress (GLC)*, June 2019, pp. 1–5, Paris, France.

[15] T. B. Hoang, S. Sahugue`de, and A. Julien-Vergonjanne, “Optical wire- less network design for off-body-sensor based monitoring,” *Wireless Communications and Mobile Computing*, vol. 2019, p. 13, Sep. 2019

[16] P. K. Plunkett, D. G. Byrne, T. Breslin, K. Bennett, and B. Silke, “Increasing wait times predict increasing mortality for emergency medical admissions,” *European Journal of Emergency Medicine*, vol. 18, pp. 192– 196, Aug. 2011