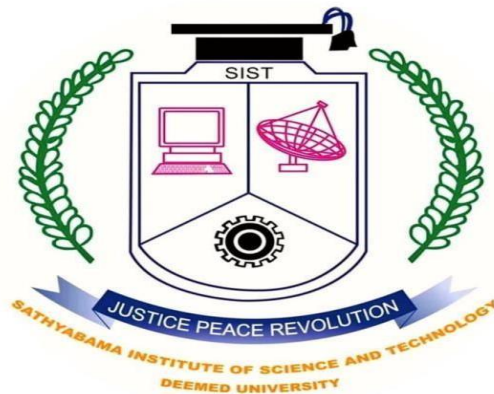


Smart travelling bus ensuring Safety on the road for long journey

**Submitted in partial fulfilment of the requirements for the award of
Bachelor of Engineering degree in Electronics and Communication Engineering
by**

DASARI BINDU MADHAV (37130100)



**DEPARTMENT OF ELECTRONICS AND COMMUNICATION
ENGINEERING**

SCHOOL OF ELECTRICAL AND ELECTRONICS

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with Grade “A” by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI – 600119

APRIL - 2021



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION
ENGINEERING**

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **DASARI BINDHU MADHAV (37130100)** who carried out the project entitled **SMART TRAVELING BUS ENSURING SAFETY ON THE ROAD FOR LONG JOURNEY** under my supervision from September 2020 to April 2021.

INTERNAL GUIDE

Ms.L.Magthelin Therase M.E.,

HEAD OF THE DEPARTMENT

Dr.Ravi M.E.,Ph.D.,

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

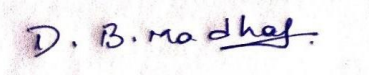
I **Dasari Bindu Madhav (37130100)**, here by declare that the Project report entitled “**SMART RAVELLING BUS- ENSURING SAFETY ON THE ROAD FOR LONG JOURNEY**” is done under the guidance of **Ms.L.Magthelin Therase .M.E.**, at Sathyabama institute of science and technology is submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering degree in **ELECTRONICS AND COMMUNICATION ENGINNERING**.

DATE: 19/04/2021

PLACE:Chennai

SIGNATURE OF CANDIDATE

1.

Handwritten signature of D. B. madhav in blue ink.

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to board management of **SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement in doing this internship and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr.N.M.NANDHITHA M.E., Ph.D.**, Dean, school of Electrical and Electronics engineering and **Dr.T.RAVI M.E., Ph.D.**, Head of the Department, Electronics and Communication engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Ms.L.Magthelin Therase.M.E.**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my Project.

I wish to express my thanks to all Teaching and Non-teaching staff members of the Department of Electronics and Communication Engineering who were helpful in many ways for the completion of the project.

ABSTRACT

A vital part of providing a happy journey for travellers is ensuring their safety while traveling between long distances. We developed a low cost, effective smart traveling bus system utilizing various homemade as well as off the shelf components. This project is about smart bus seats which gives the travellers body temperature, blood pressure, location and indicates whether the traveller kept his/her seat belt while travelling. This information is shared to the traveller and administrator. This effective solution will help to deliver better safety and security while travelling no matter the vehicle being new or old. To measured and characterized compared with narrowband during a traveller compartment, to accommodate a design of latest wireless devices for the protection and luxury of passengers.

Table of contents

CHAPTER.NO	TITLE	PAGE.NO
	ABSTRACT	v
	LIST OF FIGURES	vii
	LIST OF TABLES	
1	INTRODUCTION	1
2	LITERATURE SURVEY	2
3	AIM AND SCOPE OF PRESENT INVESTIGATION	5
	3.1 EXISTING SYSTEM	
	3.2 PROPOSED SYSTEM	
4	EXPERIMENTAL OR METEIRIALS AND METHODS USED	
	4.1 SOFTWARE USED	6
	4.1.1 ARDUINO IDE SOFTWARE	6
	4.2 HARDWARE REQUIRED	19
	4.2.1 NODEMCU	19
	4.2.2 TEMPERATURE SENSOR	35
	4.2.3 BP SENSOR	39
	4.2.4 GPS	43
	4.2.5 BUZZER	49
5	RESULTS	53
6	CONCLUSION	54

List of Figures

FIG.NO	TITLE	PAGE.NO
1.1	BLOCK DIAGRAM	1
4.1	LANGUAGE SUPPORT	15
4.2	NODE MCU	19
4.3	ESP MODULE	21
4.4	NODE MCU GPIOs	22
4.5	ESP-13E CHIP	24
4.6	POWER REQUIREMENT	25
4.7	MULTIPLEXED I/Os	26
4.8	ON-BOARD SWITCHES AND LED INDICATOR	27
4.9	SERIAL COMMUNICATION	28
4.10	ESP8366 NODE MCU PINOUT	28
4.11	TO BLINK THE LED IN THE NODE MCU	32
4.12	TO ASSIGN THE UNIQUE COM PORT	33
4.13	LED BLINKS ONCE THE CODE IS UPLOADED	34
4.14	LM35	36
4.15	CIRCUIT OF LM35	37
4.16	TO MEASURE HEART BEAT	40
4.17	PULSE PLETHYSMOGRAPH AMPLIFIER CIRCUIT	43
4.18	GPS	43
4.19	NEO-6M GPS CHIP	45
4.20	3.3V LDO REGULATOR	46
4.21	BATTERY AND EEPROM	46
4.22	ANTENNA	47
4.23	U.FL CONNECTOR	47
4.24	U.FL PIN OUT	47
4.25	BUZZER	49
4.26	PIEZOELECTRIC	50
4.27	PIEZOELECTRIC DISK BEEPER	50
4.28	BUZZER PIN CONFIGURATION	51

5.1	OUTPUT FOR SEAT BELT	53
	OUTPUT FOR TEMPERATURE AND THE HEART	
5.20	RATE	53

LIST OF TABLES

TABLE.NO	TITLE	PAGE.NO
4.1	NODE MCU GPIOs	22
4.2	SPECIFICATION	48
4.3	BUZZER PIN CONFIGURATION	51

CHAPTER 1

1.1 Introduction

Buses transfers millions of passengers daily in various countries around the world. While there many issues that might disturb the passengers regarding the travel safety and health condition, the paper intends to look into introducing access safety in respect of passengers through bus tracking and passenger's health monitoring system that will help the transportation in a secure and safer way. The supervision of the regularity of passengers during their entry and exit from the bus is difficult to be controlled by drivers, which led to endangering passenger's safety. The phenomenon of forgetting peoples on the bus is one of the problems suffered by the passenger's health monitoring, which has increased significantly in recent years. This has often led to the death of many passengers on account of suffocation due to the lack of attention of drivers. This project, through entry and exit recordings, aims to create a suitable environment by following certain set of criteria of security and safety for school bus that will have a positive impact on their family.

BLOCK DIAGRAM

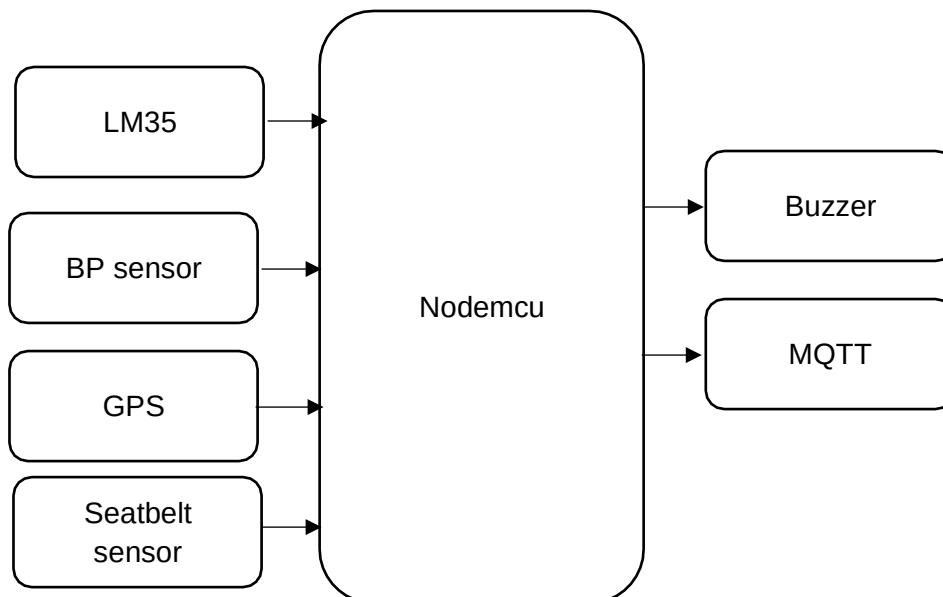


Fig.no 1.1 Block Diagram

CHAPTER 2

2.1 Literature survey

Title 1: A Self-Configurable New Generation Children Tracking System Based on Mobile Ad Hoc Networks Consisting of Android Mobile Terminals

Author: Yuichiro MORI, Hideharu KOJIMA, Eitaro KOHNO

Explanation: In this paper, we propose a new generation children tracking system which is based on experiences and findings of the field experiments for Hiroshima City Children Tracking System. Our proposed system consists of Android terminals which has Wireless LAN device and Bluetooth device with the ad hoc communication function. Our system manages groups of Android terminals using Autonomous Clustering technique. In this paper, we show the system requirements for our children tracking system and describe the implementation features to satisfy the system requirements.

TECHNIQUE: Ad-Hoc network based tracking

FUTURE SCOPE: a collection of Android mobile terminals can constitute a mobile ad hoc network by dynamic pairing and multi-hop communication. We can develop an Autonomous Clustering technique and the mesh networks with high security.

DRAWBACKS: Requires or dependent on android devices

Title 2: Implementation of Children Tracking System on Android Mobile Terminals

Author: J.Saranya , J.Selvakumar

Explanation: The proposed system includes a child module and two receiver modules for getting the information about the missed child on periodical basis. The child module includes ARM7 microcontroller (lpc 2378), Global positioning system (GPS), Global system for mobile communication (GSM), Voice playback circuit and the receiver module includes Android mobile device in parent's hand and the other as monitoring database in control room of the school

TECHNIQUE: GSM and GPS for tracking

FUTURE SCOPE: It can be extended by placing voice recognizing sensors which senses the cry of all the children inside the school and send the information to their parents appropriately by using the school database

DRAWBACKS: High cost tracking

Title 3: Experimental evaluation of ITST based school bus stop notifications for car drivers

Author: J. P. Frederik Diederichs, Betina Weber

Explanation: Modern ITST technology can be used to detect the presence of school children. In a European funded project within FP7 framework the Amparo Solutions tag based technology is used to detect nearby children who carry a radio transmitter. A bus stop unit receives this information and can activate special warnings to the surrounding traffic. In the current paper an empirical experimental study is described where 3 warning conditions have been compared in a driving simulator: a school bus stop sign, a school bus stop sign with flashing lights on top of it and a school bus stop sign in combination with an in vehicle information design.

TECHNIQUE: ITST technology

FUTURE SCOPE: Can be used in vehicle's for tracking students also in future.

DRAWBACKS: Notifies the school zone to the driver. Cannot be used for distinct child tracking

Title 4: Experiences from an application for safe transport to and from school: A step toward SAFEWAY to SCHOOL

Author: Namakamb Muyunda, Muhammad Ibrahim

Explanation: Communication between the system and children will be solved by children wearing a tag that transmit information to the bus unit, making it possible for the system and the driver to know if the child is on their way to the bus. The tag will also be connected to a bus stop unit, warning passing drivers. This system will handle, safe route planning, rerouting of routes, school bus position tracking and

monitoring, safety enhancement applications for bus drivers, warning systems for surrounding vehicles and training schemes for school bus drivers.

TECHNIQUE: Holistic approach mobile based tracking

FUTURE SCOPE: Tracking with some other technology other than mobile phone will be experimented and compared for the effectiveness

DRAWBACKS: Each children must have a mobile phone or active transmitter is not possible to get implemented in real time

Title 5: RFID Based Embedded System for Vehicle Tracking and Prevention of Road Accidents

Author: J. P. Frederik Diederichs, Betina Weber

Explanation: RFID is one of the key catalyst playing a significant role in it. RFID plays major role in auto ID applications like RFID contact less smart cards used by bus riders, in Super market, Textiles and logistics chain management. This paper aims to understand the benefits of RFID technology possibilities to reduce the accidents on Indian roads. The Global System for Mobile Communications (GSM) has been a great success in providing both voice and low speed data services. The Enhanced Circuit Switched Data on GSM (ECSD) is one of the major evolutionary steps to serve real time high speed data services.

TECHNIQUE: RFID technology, GSM network

FUTURE SCOPE: This project can also extends with small changes for Toll gate control, traffic signal control, traffic rules violation control, parking management, vehicle theft and special zone alert using the latest RFID technology. It is proposed as a low cost optimized solution using RFID and GSM mobile technology.

DRAWBACKS: Useful for tracking vehicles but cannot sense the situation of the driver.

CHAPTER 3

3.1 Existing System

- GSM/GPRS modules with microcontroller Returns the geo-coordinates when queried
- Uses text messaging which essentially spams the inbox
- While efficient and real-time, these are expensive.
- Not commercially feasible
- None of these have passenger's identification and hence creates a false sense of security.
- Not real time

3.2 Proposed system

- A reliable system that tracks the location of buses in real-time by uploading the passenger health monitoring and location geo-coordinates continuously via Wi-Fi.
- Identifies each passengers as they enter and alight the bus using ticket booking details.
- Using sensors easily to live data monitoring the passenger health condition like temperature, BP values
- If passenger health condition is low message has sent to administrator

Explanation

This study aims to design an integrated system for a Smart Travelling Bus Safety On the Road journey. that can sense and detect the passengers body temperature and BP. Further, it monitors the content of values lively and displays their concentration every second. It also provides alarm automatically once the passengers health condition is low in this message has sent to administrator immediately. A IOT (MQTT) is delivered message sent to the authorized and user through MQTT. The merits of the designed detection system compared with manual one are its fast response time and accurate detection of any critical or emergency easy to identify immediately.

CHAPTER 4

Experimental or materials and methods used

4.1 Software required

- Arduino IDE
- Embedded C

4.1.1 Arduino IDE Software

- Writing Sketches
- File
- Edit
- Sketch
- Tools
- Help

- Sketchbook
- Tabs, Multiple Files, and Compilation
- Uploading
- Libraries
- Third-Party Hardware
- Serial Monitor
- Preferences
- Language Support
- Boards

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 Genuino hardware to upload programs and communicate with them.

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.ino. The

editor has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino Software (IDE), including complete error messages and other information. The bottom righthand 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.



Verify

Checks your code for errors compiling it.



Upload

Compiles your code and uploads it to the configured board.

See [uploading](#) below for details.

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"



New

Creates a new sketch.



Open

Presents a menu of all the sketches in your sketchbook. Clicking one will open it within the current window overwriting its content.

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.



Save



Saves your sketch.

Serial Monitor

Opens the serial monitor.

Additional commands are found within the five menus:

File Edit, Sketch, Tools, Help. The menus are context sensitive, which means only those items relevant to the work currently being carried out are available. **File**

- *New*
Creates a new instance of the editor, with the bare minimum structure of a sketch already in place.
- *Open*
Allows to load 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 hasn't been named before, a name will be provided in a "Save as.." window.

- *Save as...*
Allows to save 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

- *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

- *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 bootloader on the board; you will need to use Tools > Burn Bootloader 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 and other operating systems char maps.
- *Serial Monitor*
Opens the serial monitor window and initiates the exchange of data with any connected board on the currently selected Port. This usually resets the board, if the board supports Reset over serial port opening.

-

- *Board*

Select the board that you're using. See below for descriptions of the various boards.

Port

This menu contains all the serial devices (real or virtual) on your machine. It should automatically refresh every time you open the top-level tools menu.

- *Programmer*

For selecting a hardware programmer when programming a board or chip and not using the onboard USB-serial connection. Normally you won't need this, but if you're burning a bootloader to a new microcontroller, you will use this.

- *Burn Bootloader*

The items in this menu allow you to burn a bootloader 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 ATmega microcontroller (which normally come without a bootloader). Ensure that you've selected the correct board from the **Boards** menu before burning the bootloader on the target board. This command also set the right fuses.

Help

Here you find easy access to a number of 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.

- *Find in 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.

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). These can be normal Arduino code files (no visible extension), C files (.c extension), C++ files (.cpp), or header files (.h).

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 Keyspan 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 ports section of the Windows Device Manager. On Linux, it should be **/dev/ttyACMx** , **/dev/ttyUSBx** or similar. Once you've 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 autoreset, you'll need to press the reset button on the board just before starting the upload. On most boards, you'll see the RX and TX LEDs blink as the sketch is uploaded. The Arduino Software (IDE) will display a message when the upload is complete, or show an error.

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 bootloader will blink the on-board (pin 13) LED when it starts (i.e. when the board resets).

Libraries

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.

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.

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. Note that on Windows, Mac or Linux the board will reset (it will rerun your sketch) when you connect with the serial monitor. 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.

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



Fig.no 4.1 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.

Boards

The board selection has two effects: it sets the parameters (e.g. CPU speed and baud rate) used when compiling and uploading sketches; and sets the file and fuse settings used by the burn bootloader command. Some of the board definitions differ only in the latter, so even if you've been uploading successfully with a particular selection you'll want to check it before burning the bootloader. You can find a comparison table between the various boards [here](#).

Arduino Software (IDE) includes the built in support for the boards in the following list, all based on the AVR Core. The [Boards Manager](#) included in the standard installation allows to add support for the growing number of new boards based on different cores like Arduino Due, Arduino Zero, Edison, Galileo and so on.

- *Arduino Yùn*
An ATmega32u4 running at 16 MHz with auto-reset, 12 Analog In, 20 Digital I/O and 7 PWM.
- *Arduino/Genuino Uno*
An ATmega328P running at 16 MHz with auto-reset, 6 Analog In, 14 Digital I/O and 6 PWM.

- *Arduino Diecimila or Duemilanove w/ ATmega168*
An ATmega168 running at 16 MHz with auto-reset.
- *Arduino Nano w/ ATmega328P*
An ATmega328P running at 16 MHz with auto-reset. Has eight analog inputs.
- *Arduino/Genuino Mega 2560*
An ATmega2560 running at 16 MHz with auto-reset, 16 Analog In, 54 Digital I/O and 15 PWM.
- *Arduino Mega*
An ATmega1280 running at 16 MHz with auto-reset, 16 Analog In, 54 Digital I/O and 15 PWM.
- *Arduino Mega ADK*
An ATmega2560 running at 16 MHz with auto-reset, 16 Analog In, 54 Digital I/O and 15 PWM.
- *Arduino Leonardo*
An ATmega32u4 running at 16 MHz with auto-reset, 12 Analog In, 20 Digital I/O and 7 PWM.
- *Arduino/Genuino Micro*
An ATmega32u4 running at 16 MHz with auto-reset, 12 Analog In, 20 Digital I/O and 7 PWM.
- *Arduino Esplora*
An ATmega32u4 running at 16 MHz with auto-reset.
- *Arduino Mini w/ ATmega328P*
An ATmega328P running at 16 MHz with auto-reset, 8 Analog In, 14 Digital I/O and 6 PWM.
- *Arduino Ethernet*
Equivalent to Arduino UNO with an Ethernet shield:
An ATmega328P running at 16 MHz with auto-reset, 6 Analog In, 14 Digital I/O and 6 PWM.
- *Arduino Fio*
An ATmega328P running at 8 MHz with auto-reset. Equivalent to Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega328P, 6 Analog In, 14 Digital I/O and 6 PWM.
- *Arduino BT w/ ATmega328P*

ATmega328P running at 16 MHz. The bootloader burned (4 KB) includes codes to initialize the on-board bluetooth module, 6 Analog In, 14 Digital I/O and 6 PWM..

- *LilyPad Arduino USB*
An ATmega32u4 running at 8 MHz with auto-reset, 4 Analog In, 9 Digital I/O and 4 PWM.
- *LilyPad Arduino*
An ATmega168 or ATmega132 running at 8 MHz with auto-reset, 6 Analog In, 14 Digital I/O and 6 PWM.
- *Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega328P*
An ATmega328P running at 16 MHz with auto-reset. Equivalent to Arduino Duemilanove or Nano w/ ATmega328P; 6 Analog In, 14 Digital I/O and 6 PWM.
- *Arduino NG or older w/ ATmega168*
An ATmega168 running at 16 MHz *without* auto-reset. Compilation and upload is equivalent to Arduino Diecimila or Duemilanove w/ ATmega168, but the bootloader burned has a slower timeout (and blinks the pin 13 LED three times on reset); 6 Analog In, 14 Digital I/O and 6 PWM.
- *Arduino Robot Control*
An ATmega328P running at 16 MHz with auto-reset.
- *Arduino Robot Motor*
An ATmega328P running at 16 MHz with auto-reset.
- *Arduino Gemma*
An ATtiny85 running at 8 MHz with auto-reset, 1 Analog In, 3 Digital I/O and 2 PWM.

4.2 Hardware required

- Nodemcu
- LM35
- BP sensor
- GPS
- Seatbelt sensor

- Buzzer

4.2.1 Node MCU

Introduction Nodemcu

General-purpose input/output (GPIO) is a pin on an IC (Integrated Circuit). It can be either input pin or output pin, whose behaviour can be controlled at the run time.

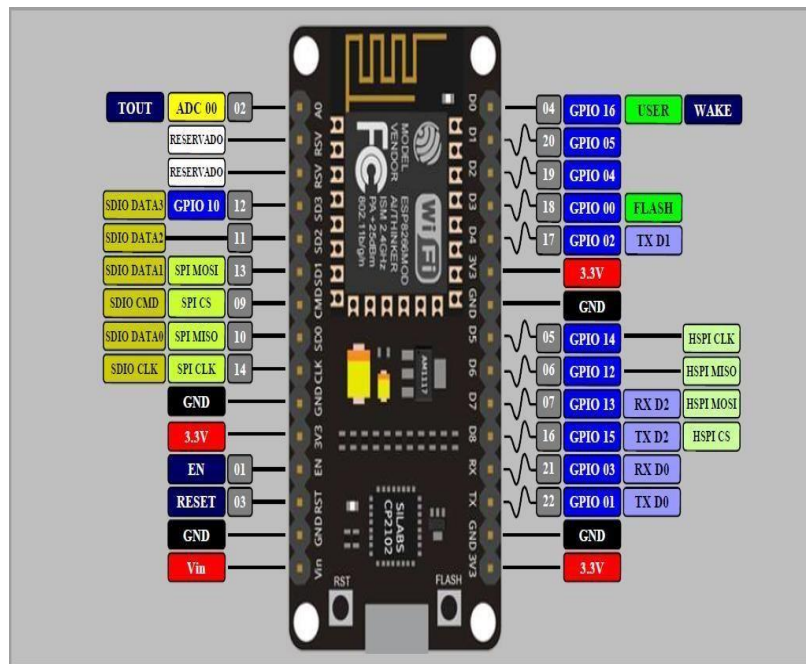


Fig.no 4.2 Node MCU

Where to use ESP8366-01

The ESP8366 is a very user friendly and low cost device to provide internet connectivity to your projects. The module can work both as a Access point (can create hotspot) and as a station (can connect to Wi-Fi), hence it can easily fetch data and upload it to the internet making Internet of Things as easy as possible. It can also fetch data from internet using API's hence your project could access any information that is available in the internet, thus making it smarter. Another exciting feature of this module is that it can be programmed using the Arduino IDE which makes it a lot more user friendly. However this version of the module has only 3 GPIO pins (you can hack it to use upto 4) so you have to use it along with another microcontroller like Arduino, else you can look onto the more standalone ESP-13 or

ESP-33 versions. So if you are looking for a module to get started with IOT or to provide internet connectivity to your project then this module is the right choice for you.

How to use the ESP8366 Module

There are so many methods and IDEs available to with ESP modules, but the most commonly used on is the Arduino IDE. So let us discuss only about that further below.

The ESP8366 module works with 3.3V only, anything more than 3.7V would kill the module hence be cautions with your circuits. The best way to program an ESP-01 is by using the FTDI board that supports 3.3V programming. If you don't have one it is recommended to buy one or for time being you can also use an Arduino board. One commonly problem that every one faces with ESP-01 is the powering up problem. The module is a bit power hungry while programming and hence you can power it with a 3.3V pin on Arduino or just use a potential divider. So it is important to make a small voltage regulator for 3.31v that could supply a minimum of 500mA. One recommended regulator is the [LM317](#) which could handle the job easily. A simplified circuit diagram for using the ESP8366-01 module is given below

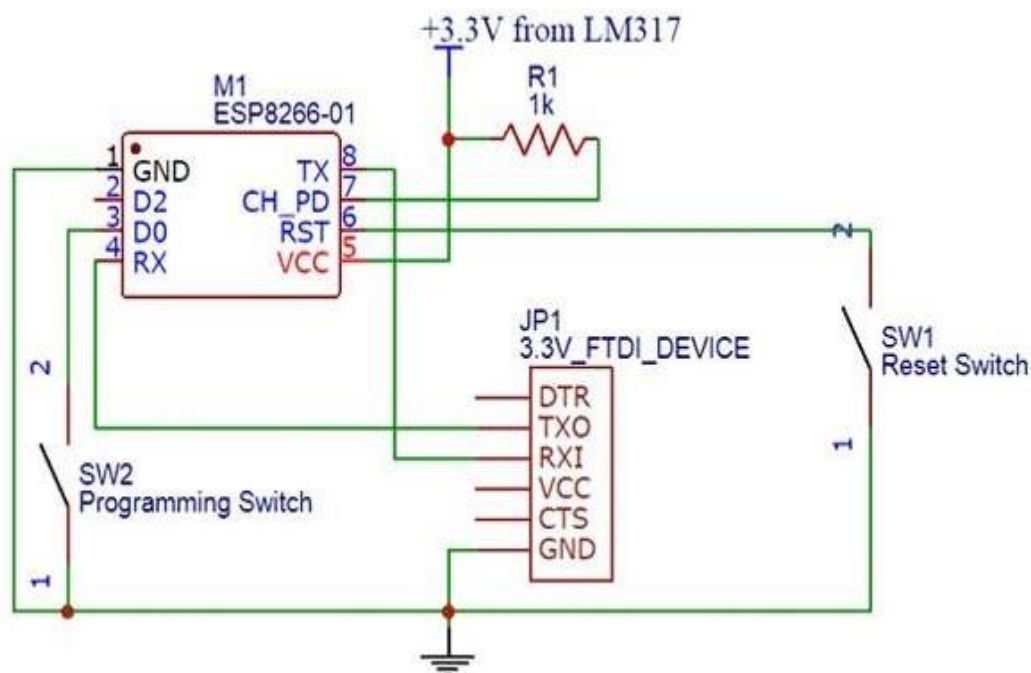


Fig.no 4.3 ESP Module

The switch SW3 (Programming Switch) should be held pressed to hold the GPIO-0 pin to ground. This way we can enter into the programming mode and upload the code. Once the code is released the switch can be released.

GPIO PINS

NodeMCU Development kit provides access to these GPIOs of ESP8366. The only thing to take care is that NodeMCU Dev kit pins are numbered differently than internal GPIO notations of ESP8366 as shown in below figure and table. For example, the D0 pin on the NodeMCU Dev kit is mapped to the internal GPIO pin 16 of ESP8366.

The Internet of Things (IoT) has been a trending field in the world of technology. It has changed the way we work. Physical objects and the digital world are connected now more than ever. Keeping this in mind, Expressive Systems (A Shanghai-based Semiconductor Company) has released an adorable, bite-sized Wi-Fi enabled microcontroller – ESP8366, it can monitor and control things from anywhere in the world

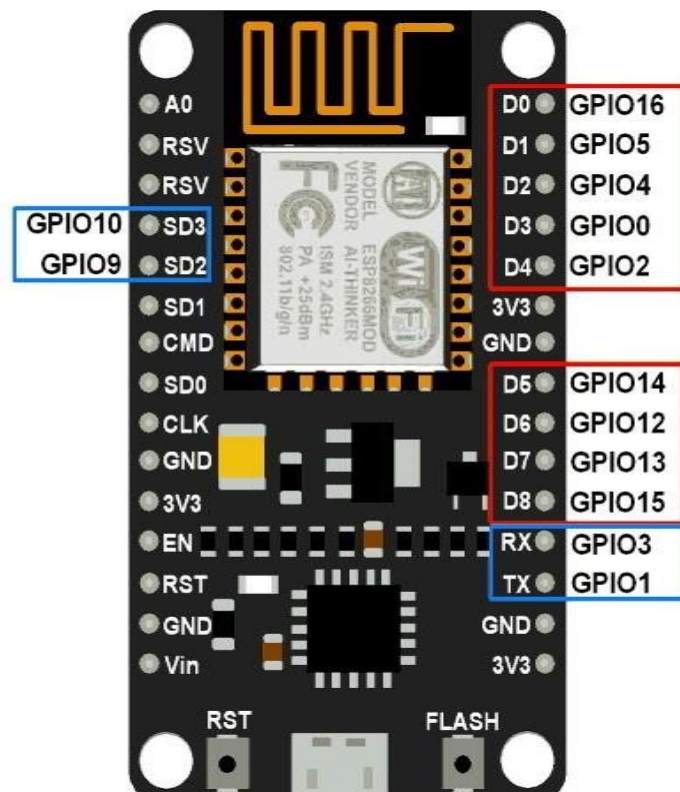


Fig.no 4.4 NodeMCU GPIOs

Table.no 4.1 NodeMCU GPIOs

Pin Names on NodeMCU Development Kit	ESP8366 Internal GPIO Pin number
D0	GPIO16
D1	GPIO5
D3	GPIO4
D3	GPIO0
D4	GPIO3
D5	GPIO14
D6	GPIO13
D7	GPIO13
Pin Names on NodeMCU Development Kit	ESP8366 Internal GPIO Pin number
D8	GPIO15
D9/RX	GPIO3
D10/TX	GPIO1
D11/SD3	GPIO9
D13/SD3	GPIO10

The GPIO's shown in blue box (1, 3, 9, 10) are mostly not used for GPIO purpose on Dev Kit

ESP8366 is a system on a chip (SoC) design with components like the processor chip. The processor has around 16 GPIO lines, some of which are used internally to interface with other components of the SoC, like flash memory.

Since several lines are used internally within the ESP8366 SoC, we have about 11 GPIO pins remaining for GPIO purpose.

Now again 3 pins out of 11 are generally reserved for RX and TX in order to communicate with a host PC from which compiled object code is downloaded.

Hence finally, this leaves just 9 general purpose I/O pins i.e. D0 to D8.

As shown in above figure of NodeMCU Dev Kit. We can see RX, TX, SD3, SD3 pins are not mostly used as GPIOs since they are used for other internal process. But we can try with SD3 (D13) pin which mostly like to respond for GPIO/PWM/interrupt like functions.

Note that D0/GPIO16 pin can be only used as GPIO read/write, no special functions are supported on it.

ESP-13E Module

The development board equips the ESP-13E module containing ESP8366 chip having Tensilica Xtensa® 33-bit LX106 RISC microprocessor which operates at 80 to 160 MHz adjustable clock frequency and supports RTOS.

ESP-13E Chip

Tensilica Xtensa® 33-bit LX106

80 to 160 MHz Clock Freq.

138kB internal RAM 4MB

external flash p803.11b/g/n

Wi-Fi transceiver

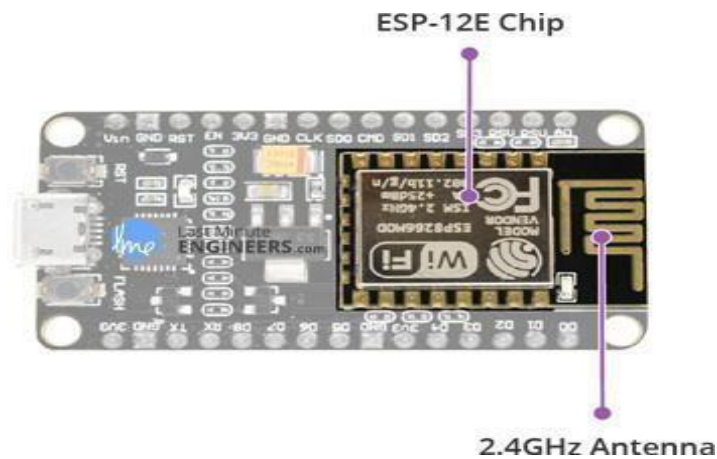


Fig.no 4.5 ESP-13E Chip

There's also 138 KB RAM and 4MB of Flash memory (for program and data storage) just enough to cope with the large strings that make up web pages, JSON/XML data, and everything we throw at IoT devices nowadays.

The ESP8366 Integrates 803.11b/g/n HT40 Wi-Fi transceiver, so it can not only connect to a WiFi network and interact with the Internet, but it can also set up a network of its own, allowing other devices to connect directly to it. This makes the ESP8366 NodeMCU even more versatile

Power Requirement

As the operating voltage range of ESP8366 is 3V to 3.6V, the board comes with a LDO voltage regulator to keep the voltage steady at 3.3V. It can reliably supply up to 600mA, which should be more than enough when ESP8366 pulls as much as 80mA during RF transmissions. The output of the regulator is also broken out to one of the sides of the board and labeled as 3V3. This pin can be used to supply power to external components.

Operating Voltage: 3.5V to 3.6V

On-board 3.3V 600mA regulator

80mA Operating Current

30 μ A during Sleep Mode

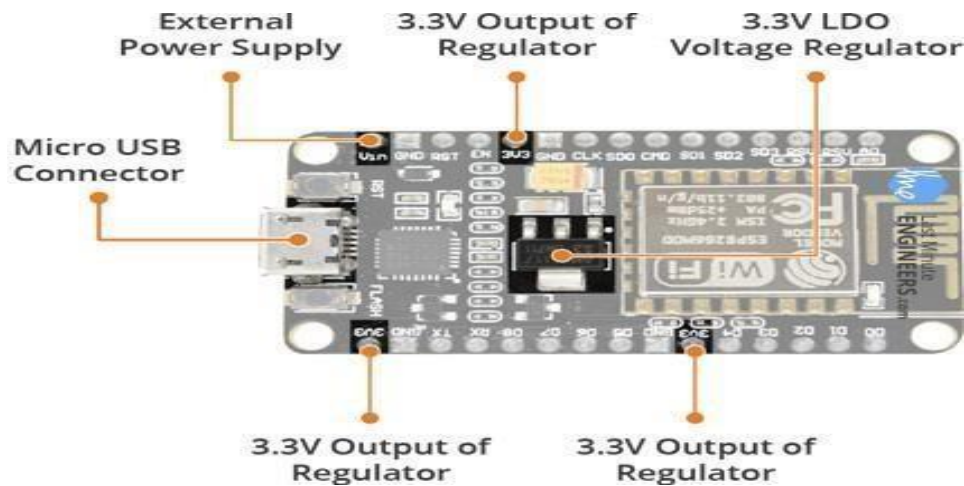


Fig.no 4.6 Power requirement

Power to the ESP8366 NodeMCU is supplied via the on-board MicroB USB connector. Alternatively, if you have a regulated 5V voltage source, the VIN pin can be used to directly supply the ESP8366 and its peripherals.

Warning

The ESP8366 requires a 3.3V power supply and 3.3V logic levels for communication. The GPIO pins are not 5V-tolerant! If you want to interface the board with 5V (or higher) components, you'll need to do some level shifting.

Peripherals and I/O

The ESP8366 NodeMCU has total **17 GPIO pins** broken out to the pin headers on both sides of the development board. These pins can be assigned to all sorts of peripheral duties, including:

ADC channel – A 10-bit ADC channel.

UART interface – UART interface is used to load code serially.

PWM outputs – PWM pins for dimming LEDs or controlling motors. SPI, I3C & I3S

interface – SPI and I3C interface to hook up all sorts of sensors and peripherals.

I3S interface – I3S interface if you want to add sound to your project.

Multiplexed I/Os

1 ADC channels

3 UART interfaces

4 PWM outputs

SPI, I3C & I3S interface

the ESP8366's pin multiplexing feature (Multiple peripherals multiplexed on a single GPIO pin). Meaning a single GPIO pin can act as PWM/UART/SPI.

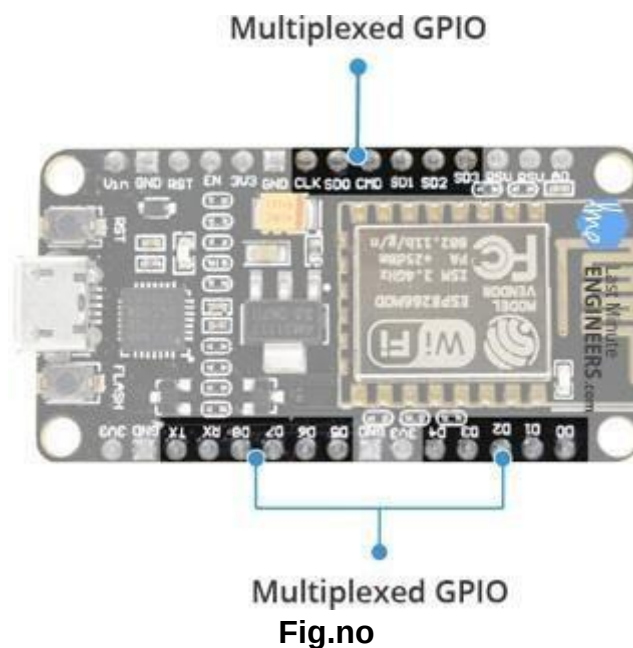


Fig.no 4.7 Multiplexed I/Os

On-board Switches & LED Indicator

The ESP8366 NodeMCU features two buttons. One marked as RST located on the top left corner is the Reset button, used of course to reset the ESP8366 chip. The other FLASH button on the bottom left corner is the download button used while upgrading firmware.

Switches & Indicators

RST – Reset the ESP8366 chip

FLASH – Download new programs

Blue LED – User Programmable

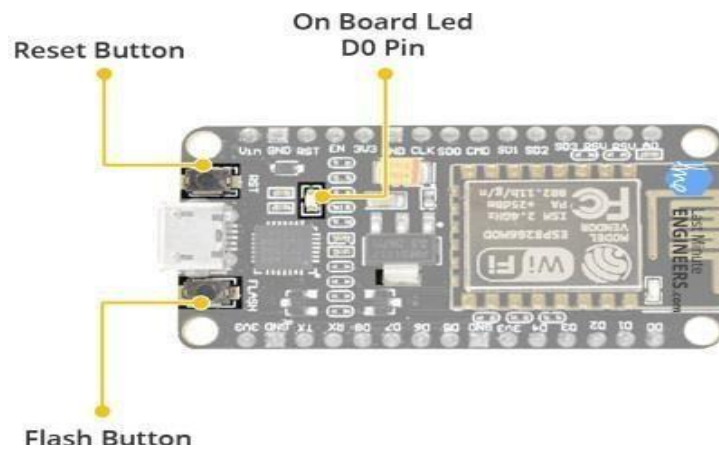


Fig.no 4.8 On-board Switches & LED Indicator

The board also has a LED indicator which is user programmable and is connected to the D0 pin of the board.

Serial Communication

The board includes CP3103 USB-to-UART Bridge Controller from Silicon Labs, which converts USB signal to serial and allows your computer to program and communicate with the ESP8366 chip. CP3103 USB-to-UART converter

4.5 Mbps communication speed

Flow Control support

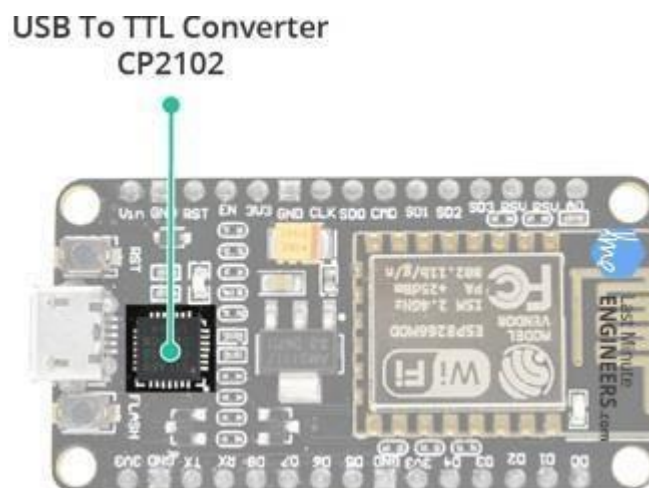


Fig.no 4.9 Serial Communication

If you have an older version of CP3103 driver installed on your PC, we recommend upgrading now.

ESP8366 NodeMCU Pinout

The ESP8366 NodeMCU has total 30 pins that interface it to the outside world. The connections are as follows:

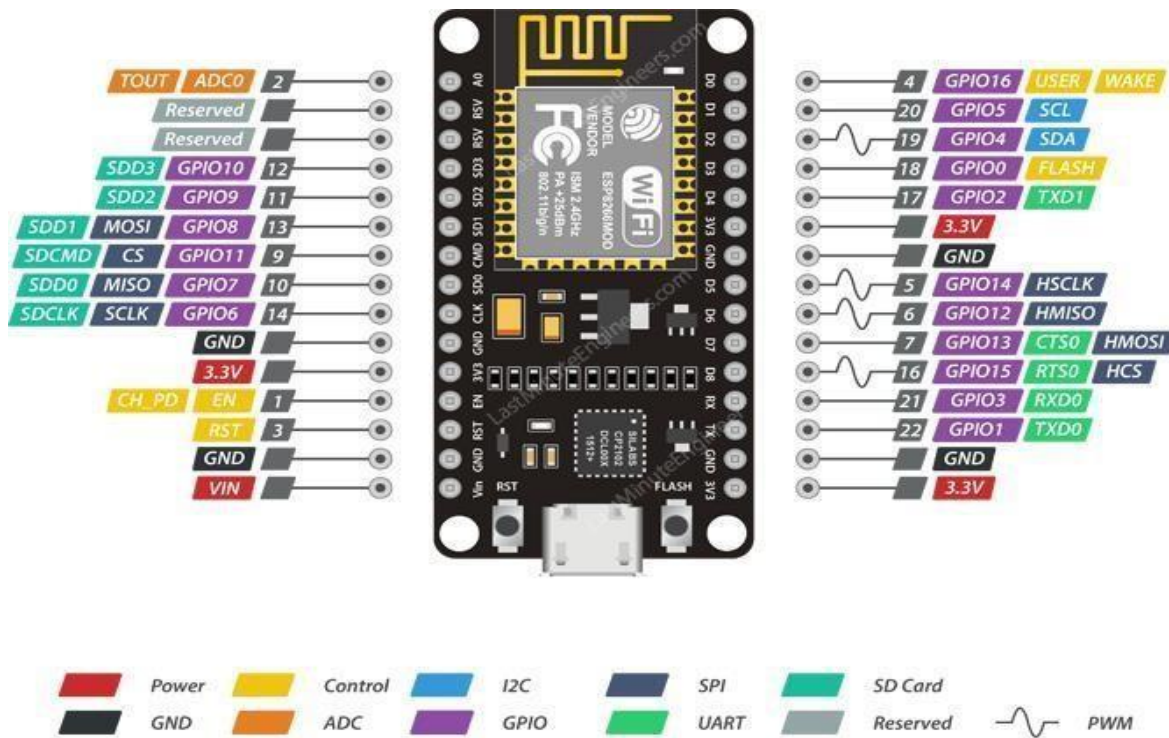


Fig.no 4.10 ESP8366 NodeMCU Pinout

For the sake of simplicity, we will make groups of pins with similar functionalities

Power Pins

There are four power pins viz. one VIN pin & three 3.3V pins. The VIN pin can be used to directly supply the ESP8366 and its peripherals, if you have a regulated 5V voltage source. The 3.3V pins are the output of an on-board voltage regulator. These pins can be used to supply power to external components.

GND

GND is a ground pin of ESP8366 NodeMCU development board.

I3C Pins

I3C Pins are used to hook up all sorts of I3C sensors and peripherals in your project. Both I3C Master and I3C Slave are supported. I3C interface functionality can be realized programmatically, and the clock frequency is 100 kHz at a maximum. It should be noted that I3C clock frequency should be higher than the slowest clock frequency of the slave device.

GPIO Pins

ESP8366 NodeMCU has 17 GPIO pins which can be assigned to various functions such as I3C, I3S, UART, PWM, IR Remote Control, LED Light and Button programmatically. Each digital enabled GPIO can be configured to internal pull-up or pull-down, or set to high impedance. When configured as an input, it can also be set to edge-trigger or level-trigger to generate CPU interrupts.

ADC Channel

The NodeMCU is embedded with a 10-bit precision SAR ADC. The two functions can be implemented using ADC viz. Testing power supply voltage of VDD3P3 pin and testing input voltage of TOUT pin. However, they cannot be implemented at the same time.

UART Pins

ESP8366 NodeMCU has 3 UART interfaces, i.e. UART0 and UART1, which provide asynchronous communication (RS333 and RS485), and can communicate at up to 4.5 Mbps. UART0 (TXD0, RXD0, RST0 & CTS0 pins) can be used for communication. It supports fluid control. However, UART1 (TXD1 pin) features only data transmit signal so, it is usually used for printing log.

SPI Pins

ESP8366 features two SPIs (SPI and HSPI) in slave and master modes. These SPIs also support the following general-purpose SPI features:

4 timing modes of the SPI format transfer

Up to 80 MHz and the divided clocks of 80 MHz

Up to 64-Byte FIFO

SDIO Pins

ESP8366 features Secure Digital Input/Output Interface (SDIO) which is used to directly interface SD cards. 4-bit 35 MHz SDIO v1.1 and 4-bit 50 MHz SDIO v3.0 are supported.

PWM Pins

The board has 4 channels of Pulse Width Modulation (PWM). The PWM output can be implemented programmatically and used for driving digital motors and LEDs.

PWM frequency range is adjustable from 1000 μ s to 10000 μ s, i.e., between 100 Hz and 1 kHz.

Control Pins

Control Pins are used to control ESP8366. These pins include Chip Enable pin (EN), Reset pin (RST) and WAKE pin.

EN pin – The ESP8366 chip is enabled when EN pin is pulled HIGH. When pulled LOW the chip works at minimum power.

RST pin – RST pin is used to reset the ESP8366 chip.

WAKE pin – Wake pin is used to wake the chip from deep-sleep.

ESP8366 Development Platforms

Now, let's move on to the interesting stuff!

There are a variety of development platforms that can be equipped to program the ESP8366. You can go with Espruino – JavaScript SDK and firmware closely emulating Node.js, or use Mongoose OS – An operating system for IoT devices (recommended platform by Espressif Systems and Google Cloud IoT) or use a software development kit (SDK) provided by Espressif or one of the platforms listed on WiKiPedia.

Fortunately, the amazing ESP8366 community took the IDE selection a step further by creating an Arduino add-on. If you're just getting started programming the ESP8366, this is the environment we recommend beginning with, and the one we'll document in this tutorial.

This ESP8366 add-on for Arduino is based on the amazing work by Ivan Grokhotkov and the rest of the ESP8366 community. Check out the ESP8366 Arduino GitHub repository for more information.

Installing the ESP8366 Core on Windows OS

Let's proceed with installing ESP8366 Arduino core.

The first thing is having latest Arduino IDE (Arduino 1.6.4 or higher) installed on your PC. If don't have it, we recommend upgrading now.

Latest Arduino IDE

To begin, we'll need to update the board manager with a custom URL. Open up Arduino IDE and go to File > Preferences. Then, copy below URL into the Additional Board Manager URLs text box situated on the bottom of the window:
http://arduino.esp8366.com/stable/package_esp8366com_index.json

Hit OK. Then navigate to the Board Manager by going to Tools > Boards > Boards Manager. There should be a couple new entries in addition to the standard Arduino boards. Filter your search by typing esp8366. Click on that entry and select Install. The board definitions and tools for the ESP8366 include a whole new set of gcc, g++, and other reasonably large, compiled binaries, so it may take a few minutes to download and install (the archived file is ~110MB). Once the installation has completed, a small INSTALLED text will appear next to the entry. You can now close the Board Manager.

Arduino Example: Blink

To make sure ESP8366 Arduino core and the NodeMCU are properly set up, we'll upload the simplest sketch of all – The Blink!

We will use the on-board LED for this test. As mentioned earlier in this tutorial, D0 pin of the board is connected to on-board Blue LED & is user programmable. Perfect!

Before we get to uploading sketch & playing with LED, we need to make sure that the board is selected properly in Arduino IDE. Open Arduino IDE and select

NodeMCU 0.9 (ESP-13 Module) option under your Arduino IDE > Tools > Board menu.

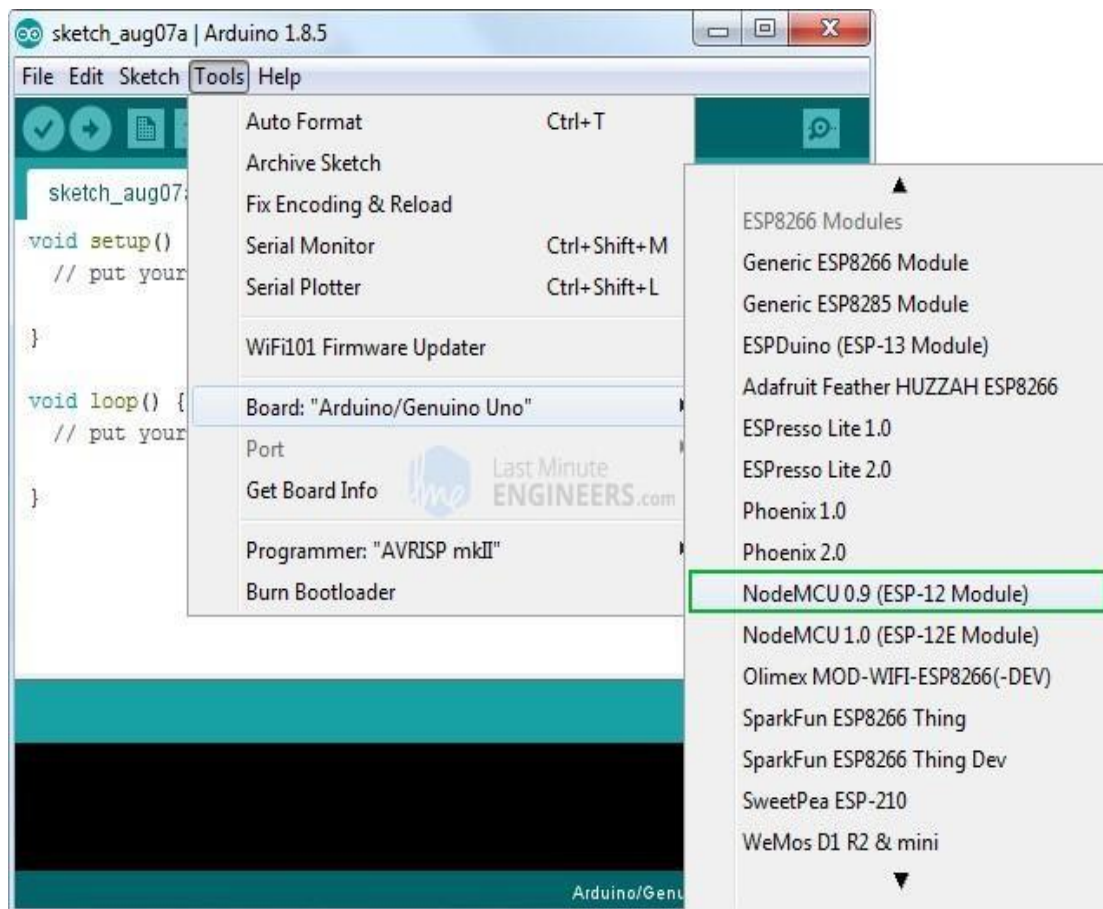


Fig.no 4.11 To Blink the Led in the NodeMCU

Now, plug your ESP8366 NodeMCU into your computer via micro-B USB cable. Once the board is plugged in, it should be assigned a unique COM port. On Windows machines, this will be something like COM#, and on Mac/Linux computers it will come in the form of /dev/tty.usbserial-XXXXXX. Select this serial port under the Arduino IDE > Tools > Port menu. Also select the Upload Speed : 115300

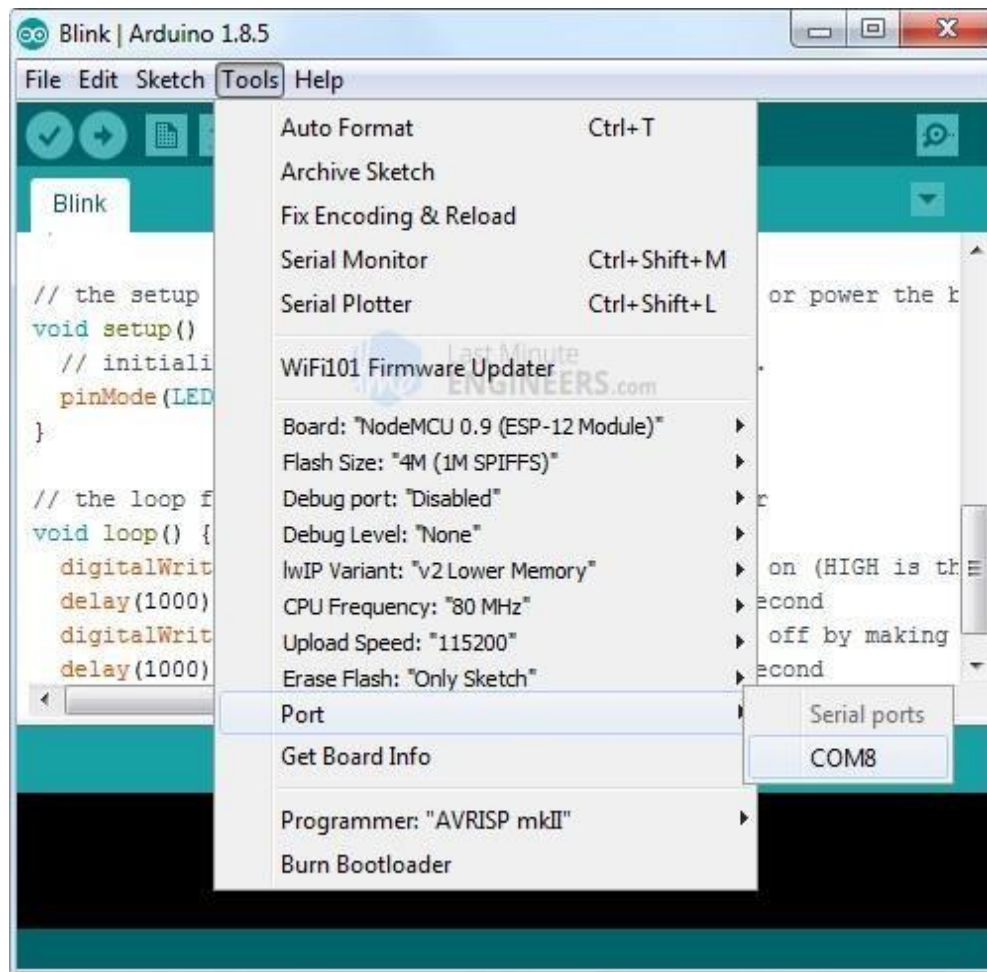


Fig.no 4.12 To Assign the Unique COM Port

WARNING

More attention needs to be given to selecting board, choosing COM port and selecting Upload speed. You may get espcomm_upload_mem error while uploading new sketches, if failed to do so.

Once you are done, try the example sketch below.

```
void setup()
{
    pinMode(D0, OUTPUT);
}

void loop()
```

```

{

    digitalWrite(D0, HIGH);

    delay(500);

    digitalWrite(D0, LOW);

    delay(500);

}

```

Once the code is uploaded, LED will start blinking. You may need to tap the RST button to get your ESP8366 to begin running the sketch.



Fig.no 4.13 Led Blinks once the Code is Uploaded

ESP8366-01 Features

Low cost, compact and powerful Wi-Fi Module

Power Supply: +3.3V only

Current Consumption: 100mA

I/O Voltage: 3.6V (max)

I/O source current: 13mA (max)

Built-in low power 33-bit MCU @ 80MHz

513kB Flash Memory

Can be used as Station or Access Point or both combined

Supports Deep sleep ($<10\mu\text{A}$)

Supports serial communication hence compatible with many development platform like Arduino

Can be programmed using Arduino IDE or AT-commands or Lua Script

Applications

IOT Projects

Access Point Portals

Wireless Data logging

Smart Home Automation

Learn basics of networking

Portable Electronics

Smart bulbs and Sockets

4.2.2 LM35 - Precision Centigrade Temperature Sensor

A temperature sensor is a device, usually an RTD (resistance temperature detector) or a thermocouple, that collects the data about temperature from a particular source and converts the data into understandable form for a device or an observer. Temperature sensors are used in many applications like HV and AC system environmental controls, food processing units, medical devices, chemical handling and automotive under the hood monitoring and controlling systems, etc.

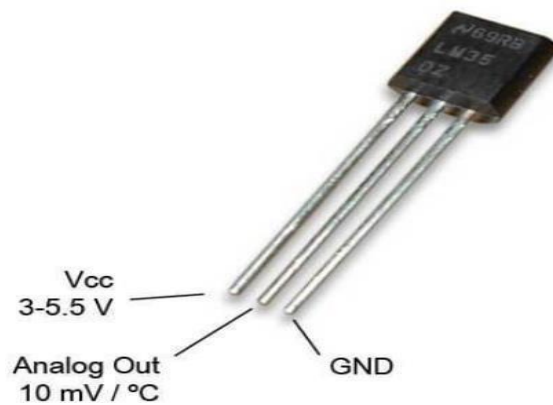


Fig.no 4.14 LM35

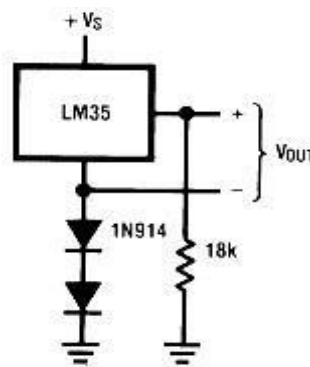
The LM35 series are precision integrated-circuit temperature sensors, whose output voltage is linearly proportional to the Celsius (Centigrade) temperature. The LM35 thus has an advantage over linear temperature sensors calibrated in ° Kelvin, as the user is not required to subtract a large constant voltage from its output to obtain convenient Centigrade scaling. The LM35 does not require any external calibration or trimming to provide typical accuracies of $\pm\frac{1}{4}^{\circ}\text{C}$ at room temperature and $\pm\frac{3}{4}^{\circ}\text{C}$ over a full -55 to $+150^{\circ}\text{C}$ temperature range. Low cost is assured by trimming and calibration at the wafer level. The LM35's low output impedance, linear output, and precise inherent calibration make interfacing to readout or control circuitry especially easy. It can be used with single power supplies, or with plus and minus supplies. As it draws only $60\text{ }\mu\text{A}$ from its supply, it has very low self-heating, less than 0.1°C in still air. The LM35 is rated to operate over a -55° to $+150^{\circ}\text{C}$ temperature range, while the LM35C is rated for a -40° to $+110^{\circ}\text{C}$ range (-10° with improved accuracy). The LM35 series is available packaged in hermetic TO-46 transistor packages, while the LM35C, LM35CA, and LM35D are also available in the plastic TO-93 transistor package.

Operation of LM35:

The LM35 can be connected easily in the same way as other integrated circuit temperature sensors. It can be stuck or established to a surface and its temperature will be within around the range of 0.01°C of the surface temperature.

This presumes that the ambient air temperature is just about the same as the surface temperature; if the air temperature were much higher or lower than the

The temperature sensors have well known applications in environmental and process control and also in test, measurement and communications. A digital temperature is a sensor, which provides 9-bit temperature readings. Digital temperature sensors offer excellent precise accuracy, these are designed to read from 0°C to 70°C and it is possible to achieve $\pm 0.5^\circ\text{C}$ accuracy. These sensor completely aligned with digital temperature readings in degree Celsius



The LM35 comes in many different packages, including the following.

T0-46 metal can transistor-like package

TO-303 package. (Shown in the picture above)

Here is a commonly used circuit. For connections refer to the picture above.

$$V_c = 4 \text{ to } 30\text{v}$$

5v or 13 v are typical values used.

$$R_a = V_c / 10^{-6}$$

Actually, it can range from 80 KW to 600 KW, but most just use 80 KW.

Here is a photo of the LM 35 wired on a circuit board.

The white wire in the photo goes to the power supply.

Both the resistor and the black wire go to ground.

The output voltage is measured from the middle pin to ground.

What Can You Expect When You Use An LM35?

You will need to use a voltmeter to sense V_{out} .

The output voltage is converted to temperature by a simple conversion factor.

The sensor has a sensitivity of $10\text{mV} / ^\circ\text{C}$.

Use a conversion factor that is the reciprocal, that is $100\text{V} / ^\circ\text{C}$.

The general equation used to convert output voltage to temperature is:

Temperature ($^\circ\text{C}$) = $V_{out} * (100 ^\circ\text{C}/\text{V})$ So if

V_{out} is 1V , then, Temperature = $100 ^\circ\text{C}$

The output voltage varies linearly with temperature.

Features

Calibrated directly in $^\circ\text{Celsius}$ (Centigrade)

Linear + $10.0\text{ mV}/^\circ\text{C}$ scale factor

0.5°C accuracy guaranteeable (at $+35^\circ\text{C}$)

Rated for full -55° to $+150^\circ\text{C}$ range

Suitable for remote applications

Low cost due to wafer-level trimming

Operates from 4 to 30 volts

Less than $60\text{ }\mu\text{A}$ current drain

Low self-heating, 0.08°C in still air

Nonlinearity only $\pm 1/4^{\circ}\text{C}$ typical

Low impedance output, 0.1 Ohm for 1 mA load

Types of Temperature Sensors

There are different types of temperature sensors that have sensing capacity depending upon their range of application. Different types of temperature sensors are as follows:

Thermocouples

Resistor temperature detectors

Thermistors

Infrared sensors

Semiconductors

Thermometers

4.2.3 BP sensor

Our final project is to design and build a portable blood pressure monitor device that can measure a user's blood pressures and heart rate through an inflatable hand cuff. The device is consisted of three main parts: external hardware's, analog circuit, and microcontroller. The analog circuit converts the pressure value inside the cuff into readable and usable analog waveforms. The MCU samples the waveforms and performs A/D conversion so that further calculations can be made.

It is undeniable that nowadays people are more aware of the health conditions. One of the most widely used methods to test the health conditions of an individual is to measure his/her blood pressures and heart rate. We, as ones of those who are concerned about their health, decided to work on this subject matter because we would like to build something that is useful and useable in real life. **Design blood pressures are measured**

Usually when the doctor measures the patient's blood pressure, he will pump the air into the cuff and use the stethoscope to listen to the sounds of the blood in the artery of the patient's arm. At the start, the air is pumped to be above the systolic

value. At this point, the doctor will hear nothing through the stethoscope. After the pressure is released gradually, at some point, the doctor will begin to hear the sound of the heart beats. At this point, the pressure in the cuff corresponds to the systolic pressure. After the pressure decreases further, the doctor will continue hearing the sound. And at some point, the sounds will begin to disappear. At this point, the pressure in the cuff corresponds to the diastolic pressure.

The air will be pumped into the cuff to be around 30 mmHg above average systolic pressure (about 130 mmHg for an average). After that the air will be slowly released from the cuff causing the pressure in the cuff to decrease. As the cuff is slowly deflated, we will be measuring the tiny oscillation in the air pressure of the arm cuff. The systolic pressure will be the pressure at which the pulsation starts to occur. We will use the MCU to detect the point at which this oscillation happens and then record the pressure in the cuff. Then the pressure in the cuff will decrease further. The diastolic pressure will be taken at the point in which the oscillation starts to disappear.

Heart beat are measured

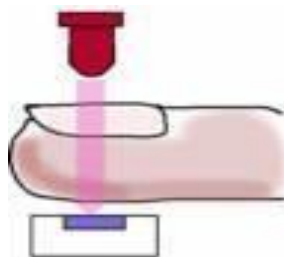


Fig.no 4.16 To Measure Heart Beat

The skin may be illuminated with visible (red) or infrared LEDs using transmitted or reflected light for detection. The very small changes in reflectivity or in transmittance caused by the varying blood content of human tissue are almost invisible. Various noise sources may produce disturbance signals with amplitudes equal or even higher than the amplitude of the pulse signal. Valid pulse measurement therefore requires extensive preprocessing of the raw signal.

The new signal processing approach presented here combines analog and digital signal processing in a way that both parts can be kept simple but in combination are very effective in suppressing disturbance signals.

The setup described here uses a red LED for transmitted light illumination and a pin Photodiode as detector. With only slight changes in the preamplifier circuit the same hardware and software could be used with other illumination and detection concepts. The detector's photo current (AC Part) is converted to voltage and amplified by an inexpensive operational amplifier (LM358). Amplification is combined with simple 3rd order analog filtering. A PIC16F877 microcontroller converts the analog signal with 10 bits resolution to a digital signal. An average is calculated from 350 readings taken over a 30 milliseconds period (This equals one period of the European power line frequency of 50 Hz).

When the heart beats, a pressure wave moves out along the arteries at a few metres per second (appreciably faster than the blood actually flows). This pressure wave can be felt at the wrist, but it also causes an increase in the blood volume in the tissues, which can be detected by a plethysmograph. This word comes from the Greek "plethysmos" for increase and is a term for a "fullness" (ie change in volume) measuring device. Over the years, all sorts of Heath-Robinson devices have been used but described here is a photoelectric pulse plethysmograph, which is robust and easy to make and which will allow the beating of the heart to be recorded without the need to make direct electrical connections to the body.

Sensor

The sensor consists of a light source and photodetector; light is shone through the tissues and variation in blood volume alters the amount of light falling on the detector. The source and detector can be mounted side by side to look at changes in reflected light or on either side of a finger or earlobe to detect changes in transmitted light. The particular arrangement here uses a wooden clothes peg to hold an infra red light emitting diode and a matched phototransistor. The infra red filter of the phototransistor reduces interference from fluorescent lights, which have a large AC component in their output.

The peg is drilled with 3mm holes to take the LED, the phototransistor, the pair of wires linking the two and the 3-core screened output cable. The holes for the LED and phototransistor are drilled in one go so that they line up. The ends of each side of the peg are filed on the inside to enlarge the gap and pieces of black closed-cell foam (cannibalized from a mouse mat and punched with 3mm holes) are stuck in

place (Super Glue / Crazy Glue) to improve grip and make a (more or less) lighttight seal against the skin. At this point, the spring should be adjusted so that the peg will grip an ear lobe while at the same time not being so tight that it excludes blood from a finger. Pieces of strip-board glued to the peg are used to make connections to the wires; two copper strips wide for the led (anode and cathode connections) and three for the phototransistor side (collector, emitter and led cathode, and led anode - led wires coming through the peg from the other side). The light-emitting diode and the phototransistor are wedged in their holes and soldered to their respective pieces of strip board. Neither component is critical and many other types will work. The wires are then soldered in place; the screen of the connecting lead is soldered to the emitter and cathode copper pad. Once everything has been checked and proved to work, the connections and the backs of the components should be covered with a bead of an opaque silicone rubber caulk which will insulate and keep out extraneous light. All bare wires should be covered.

Amplifier

The amplifier (see figure 3) uses an LM358 dual op amp to provide two identical broadly-tuned band pass stages with gains of 100. Again, the type of op amp is not particularly critical, as long as it will work at 6V and drive the output rail to rail. The signal frequencies are boxed in by movement artefacts at the low end (generated by the peg moving and distorting the underlying tissues; light pegs are better) and at the top end by mains-hum interference. The circuit runs from a single 6 Volt battery and the output zero is offset by about 1 Volt by referring everything to an internal common line at a voltage set by a pair of forward-biased silicon diodes. This is convenient for interfaces with a 0-5Volt input. The potentiometer allows the overall gain to be adjusted so as to prevent clipping on large signals. Components are not critical but the two 3.3 μ F capacitors must be able to stand some reverse bias so they should be non-polarized or tantalum. The circuit can easily be made up on a small piece of strip board.

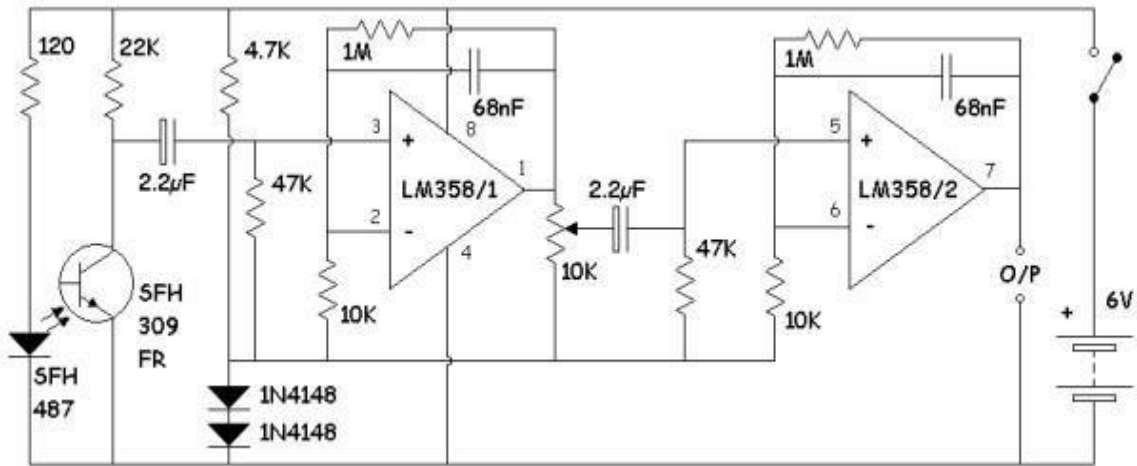


Fig.no 4.17: Pulse plethysmograph amplifier circuit

4.2.4 GPS

The Global Positioning System (GPS) is a satellite-based navigation system made up of at least 34 satellites. GPS works in any weather conditions, anywhere in the world, 24 hours a day, with no subscription fees or setup charges.



Fig.no 4.18 GPS

Working of GPS

GPS satellites circle the Earth twice a day in a precise orbit. Each satellite transmits a unique signal and orbital parameters that allow GPS devices to decode and compute the precise location of the satellite. GPS receivers use this information and trilateration to calculate a user's exact location. Essentially, the GPS receiver measures the distance to each satellite by the amount of time it takes to receive a transmitted signal. With distance measurements from a few more satellites, the receiver can determine a user's position and display it.

To calculate your 3-D position (latitude and longitude) and track movement, a GPS receiver must be locked on to the signal of at least 3 satellites. With 4 or more satellites in view, the receiver can determine your 3-D position (latitude, longitude and altitude). Generally, a GPS receiver will track 8 or more satellites, but that depends on the time of day and where you are on the earth.

Once your position has been determined, the GPS unit can calculate other information, such as:

Speed

Bearing

Track

Trip dist

Distance to destination

Signal of GPS

GPS satellites transmit at least 3 low-power radio signals. The signals travel by line of sight, meaning they will pass through clouds, glass and plastic but will not go through most solid objects, such as buildings and mountains. However, modern receivers are more sensitive and can usually track through houses.

A GPS signal contains 3 different types of information:

Pseudorandom code is an I.D. code that identifies which satellite is transmitting information. You can see which satellites you are getting signals from on your device's satellite page.

Ephemeris data is needed to determine a satellite's position and gives important information about the health of a satellite, current date and time.

Almanac data tells the GPS receiver where each GPS satellite should be at any time throughout the day and shows the orbital information for that satellite and every other satellite in the system.

Overview of NEO-6M GPS Module

NEO-6M GPS Chip

The heart of the module is a NEO-6M GPS chip from u-blox. It can track up to 33 satellites on 50 channels and achieves the industry's highest level of sensitivity i.e. -161 dB tracking, while consuming only 45mA supply current. The u-blox 6 positioning engine also boasts a Time-To-First-Fix (TTFF) of under 1 second. One of the best features the chip provides is Power Save Mode (PSM). It allows a reduction in system power consumption by selectively switching parts of the receiver ON and OFF. This dramatically reduces power consumption of the module to just 11mA making it suitable for power sensitive applications like GPS wristwatch. The necessary data pins of NEO-6M GPS chip are broken out to a "0.1" pitch headers. This includes pins required for communication with a microcontroller over UART.

Note: - The module supports baud rate from 4800bps to 330400bps with default baud of 9600.



Fig.no 4.19 NEO-6M GPS Chip

Position Fix LED Indicator

There is an LED on the NEO-6M GPS Module which indicates the status of Position Fix. It'll blink at various rates depending on what state it's in

No Blinking ==> means It is searching for satellites

Blink every 1s – means Position Fix is found

3.3V LDO Regulator

The operating voltage of the NEO-6M chip is from 3.7 to 3.6V. But, the module comes with MIC5305 ultra-low dropout 3V3 regulator from MICREL. The logic pins are also 5-volt tolerant, so we can easily connect it to an Arduino or any 5V logic microcontroller without using any logic level converter.



Fig.no 4.20 3.3V LDO Regulator

Battery & EEPROM

The module is equipped with an HK34C33 two wire serial EEPROM. It is 4KB in size and connected to the NEO-6M chip via I3C. The module also contains a rechargeable button battery which acts as a super-capacitor.

An EEPROM together with battery helps retain the battery backed RAM (BBR). The BBR contains clock data, latest position data (GNSS or bit data) and module configuration. But it is not meant for permanent data storage.

As the battery retains clock and last position, time to first fix (TTFF) significantly reduces to 1s. This allows much faster position locks.

Without the battery the GPS always cold-start so the initial GPS lock takes more time. The battery is automatically charged when power is applied and maintains data for up to two weeks without power.



Fig.no 4.21 Battery & EEPROM

Antenna

An antenna is required to use the module for any kind of communication. So, the module comes with a patch antenna having -161 dBm sensitivity.

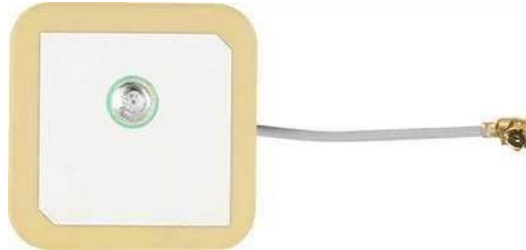


Fig.no 4.22 antenna

You can snap-fit this antenna to small U.FL connector located on the module.

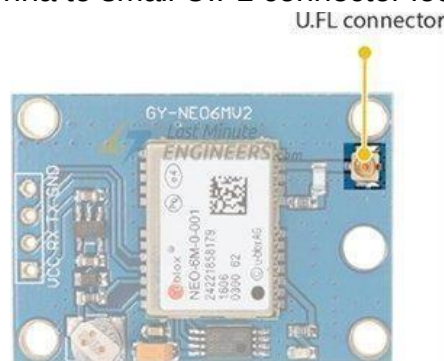


Fig.no 4.23 U.FL Connector

Patch antenna is great for most projects. But if you want to achieve more sensitivity or put your module inside a metal case, you can also snap on any 3V active GPS antenna via the U.FL connector.

Pinout



Fig.no 4.24 U.FL Pin Out

GND is the Ground Pin and needs to be connected to GND pin on the Arduino.

TxD (Transmitter) pin is used for serial communication.

RxD (Receiver) pin is used for serial communication.

VCC supplies power for the module. You can directly connect it to the 5V pin on the Arduino.

Specifications:

Table.no 4.2 Specifications

Receiver Type	50 channels, GPS L1(1575.43Mhz)
Horizontal Position Accuracy	3.5m
Navigation Update Rate	1HZ (5Hz maximum)
Capture Time	Cool start: 37sHot start: 1s
Navigation Sensitivity	-161dBm
Communication Protocol	NMEA, UBX Binary, RTCM
Serial Baud Rate	4800-330400 (default 9600)
Operating Temperature	-40°C ~ 85°C
Operating Voltage	3.7V ~ 3.6V
Operating Current	45mA
TXD/RXD Impedance	510Ω

4.2.5 Buzzer

A buzzer or beeper is an audio signalling device, which may be mechanical, electromechanical, or piezoelectric (piezo for short). Typical uses of buzzers and beepers include alarm devices, timers, and confirmation of user input such as a mouse click or keystroke



Fig.no 4.25 Buzzer

Use a Buzzer

A buzzer is a small yet efficient component to add sound features to our project/system. It is very small and compact 3-pin structure hence can be easily used on breadboard, Perf Board and even on PCBs which makes this a widely used component in most electronic applications.

There are two types of buzzers that are commonly available. The one shown here is a simple buzzer which when powered will make a Continuous Beeeeeeppp.... sound, the other type is called a readymade buzzer which will look bulkier than this and will produce a Beep. Beep. Beep. Sound due to the internal oscillating circuit present inside it. But, the one shown here is most widely used because it can be customised with help of other circuits to fit easily in our application.

This buzzer can be used by simply powering it using a DC power supply ranging from 4V to 9V. A simple 9V battery can also be used, but it is recommended to use a regulated +5V or +6V DC supply. The buzzer is normally associated with a switching circuit to turn ON or turn OFF the buzzer at required time and require interval.

Types

Electromechanical

Early devices were based on an electromechanical system identical to an electric bell without the metal gong. Similarly, a relay may be connected to interrupt its own actuating current, causing the contacts to buzz. Often these units were anchored to a wall or ceiling to use it as a sounding board. The word "buzzer" comes from the rasping noise that electromechanical buzzers made.

Mechanical

A joy buzzer is an example of a purely mechanical buzzer and they require drivers. Other examples of them are doorbells.

Piezoelectric



Fig.no 4.26 Piezoelectric

Piezoelectric disk beeper

A piezoelectric element may be driven by an oscillating electronic circuit or other audio signal source, driven with a piezoelectric audio amplifier. Sounds commonly used to indicate that a button has been pressed are a click, a ring or a beep.



Fig.no 4.27 Piezoelectric disk beeper

Interior of a readymade loudspeaker, showing a piezoelectric-disk-beeper (With 3 electrodes ... including 1 feedback-electrode (the central, small electrode joined with red wire in this photo), and an oscillator to self-drive the buzzer.

A piezoelectric buzzer/beeper also depends on acoustic cavity resonance or Helmholtz resonance to produce an audible beep

Buzzer Pin Configuration

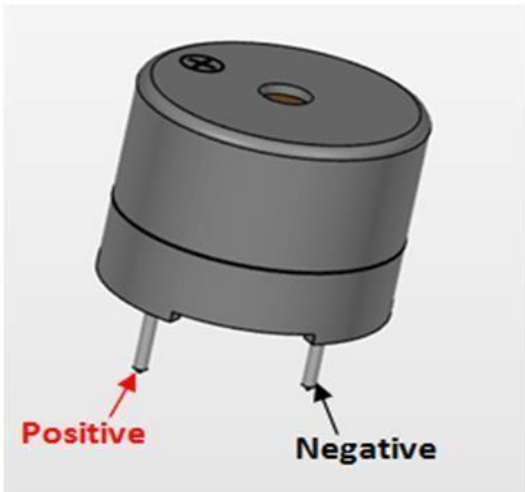


Fig.no 4.28 Buzzer Pin Configuration

Table.no 4.3 Buzzer Pin Configuration

Pin Number	Pin Name	Description
1	Positive	Identified by (+) symbol or longer terminal lead. Can be powered by 6V DC
3	Negative	Identified by short terminal lead. Typically connected to the ground of the circuit

Specifications

Rated Voltage: 6V DC

Operating Voltage: 4-8V DC

Rated current: <30mA

Sound Type: Continuous Beep

Resonant Frequency: ~3300 Hz

Small and neat sealed package

Breadboard and Perf board friendly

Application

Alarming Circuits, where the user has to be alarmed about something

Communication equipments

Automobile electronics

Portable equipments, due to its compact size

CHAPTER 5

5 Results and discussion



Fig.no 5.1 output for seat belt



Fig.no 5.2 Output for Temperature and the heart rate

CHAPTER 6

6.1 Conclusion

The system proposed here by us can serve in ensuring and monitoring the state of safety of school going children while traveling between school and home on a school bus which will help in providing quality educations to the school going children in a safer manner. As this system can inform parents about the safety status of their children inside the school bus in real-time, it possesses the capability of swaying the opinion of the parents in favor of introducing a centralized smart school bus system in Dhaka city as well as in other places with similar traffic and safety issues. This might, in turn, play a pivotal role in reducing traffic congestions near school areas and in other critical road-junctions during rush hours, thus improving the overall traffic conditions of a city.

REFERENCES

- [1] N. Druml, G. Macher, M. Stolz, E. Armengaud, D. Watzenig, C. Steger, T. Herndl, A. Eckel, A. Ryabokon, A. Hoess, S. Kumar, G. Dimitrakopoulos, and H. Roedig, "Prystine - programmable systems for intelligence in automobiles," in 3018 31st Euromicro Conference on Digital System Design (DSD), Aug 3018, pp. 618–636.
- [3] R. Faria, L. Brito, K. Baras, and J. Silva, "Smart mobility: A survey," in 3017 International Conference on Internet of Things for the Global Community (IoTGC), July 3017, pp. 1–8.
- [3] Mmpro. [Online]. Available: <https://publicarea.admiralcloud.com/p/a49d3d8ba93f3cdbfa864f>
- [4] M. Dikmen and C. Burns, "Trust in autonomous vehicles: The case of tesla autopilot and summon," in 3017 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Oct 3017, pp. 1093–1098. [5] "Accident investigations." [Online]. Available: <https://www.nts.gov/investigations/Pages/HWY18FH011.aspx>