A

Practical Training Report

on

Voice Controlled Robot with Object Detection using NodeMCU and Raspberry Pi

Submitted in partial fulfillment for the award of degree of

BACHELOR OF TECHNOLOGY

In

Computer Science & Engineering

Submitted To:

Mr. Amit Bohra

Submitted By:

Shivoham Tewari

(17EGJCS150)

Department of Computer Science & Engineering

GLOBAL INSTITUTE OF TECHNOLOGY

JAIPUR (RAJASTHAN)-302022

SESSION: 2019-20

CERTIFICATE

ICT Summer Training Certificate

This is to certify that

Dr. / Mr. / Ms. **Shivoham Tewari** has successfully

completed a 4 week course on

IoT

from ICT - Indian Institute of Technology, Kanpur

Date of Issue: 08 Jul 2019

System Identification No.: 7603-213239

Prof. B. V. Phani

Prof. Amey Karkare

E & ICT Coordinators

`

ACKNOWLEDGEMENT

With great pleasure, respect and deep gratitude, I would like to express my sincere thanks to Mr. Mukesh Kumar Head of ICT-IIT KANPUR for providing me the opportunity to accomplish training.

I am also extremely indebted to Mr. Manoj Gulati Training Officer for his obliging allowance and efforts in organizing the whole training program.

I am especially grateful to my program guide, Mrs. Shalini Kumari for her benevolent help and support throughout the training program, without whom things could not have proceeded as smoothly as they did.

Last but not the least, I would like to extend my thanks to the very cooperative **IOT Staff and Faculty** for their crutial hard work and dedication for Summer Training Program.

I am also indebted to my parents for their valuable and constant encouragement.

ABSTRACT

Today computers and the Internet are almost wholly dependent on human beings for information. Nearly all of the roughly 50 petabytes (1 petabyte=1015 bytes) of data available on the Internet were first captured and created by human beings by typing, pressing a record button, taking a digital picture, or scanning a bar code. Conventional diagrams of the Internet leave out the most numerous and important routers of all - people. The problem is, people have limited time, attention and accuracy all of which means they are not very good at capturing data about things in the real world. And that's a big deal.

Yet today's information technology is so dependent on data originated by people that our computers know more about ideas than things. If we had computers that knew everything there was to know about things using data they gathered without any help from us we would be able to track and count everything, and greatly reduce waste, loss and cost.

We would know when things needed replacing, repairing or recalling, and whether they were fresh or past their best. The Internet of Things has the potential to change the world, just as the Internet did or even more.

`

LIST OF CONTENTS

`

LIST OF TABLE

`

LIST OF FIGURES

`

CHAPTER 1
INTRODUCTION

- ## INTERNET OF THINGS:

The definition of the Internet of Things has evolved due to the convergence of multiple technologies, real-time analytics, machine learning, commodity sensors, and embedded systems.[1] Traditional fields of embedded systems, wireless sensor networks, control systems, automation (including home and building automation), and others all contribute to enabling the Internet of Things. In the consumer market, IoT technology is most synonymous with products pertaining to the concept of the "smart home", covering devices and appliances (such as lighting fixtures, thermostats, home security systems and cameras, and other home appliances) that support one or more common ecosystems, and can be controlled via devices associated with that ecosystem, such as smartphones and smart speakers.

Everyone is eyeing the next big thing after the .com boom which will make riches. World has never being the same after advent of the internet.
Investment gurus and statisticians may have many proposals to make but one thing is for sure, the next big move which will shape the century will depend on internet and embedded technology. That is, in other words **internet of things** definition is what interests major players now. What we do, how we do and when we do is never going to be the same when the physical environment around us gets lively and starts communicating.
The Internet of Things **(**IoT) is here and is becoming an increasing topic of interest among technology giants and business communities.
The hype is not baseless as there are enough evidences to support the success of “Internet of Things” in the coming years. According to a report by Gartner there will be an increase in the number of connected devices in 2016 as compared to 2015 with 6.4 billion IoT devices entering the realm of internet of things. The number is further expected to increase to 26 billion by 2020

Fig.1 Overview of IoT

- ## IMPACT OF IoT:

To put things simply any object that can be connected will be connected by the IoT. This might not make sense for you on the forefront but it is of high value. With interconnected devices you can better arrange your life and be more productive, safer, smarter and informed than ever before.

For instance how easy it will be for you to start your day if your alarm clock is not only able to wake you up but also able to communicate with your brewer to inform it that you are awake at the same time notifies your geezer to start water heating. Or you wearable wrist health band keeps track of your vitals to inform you when you are most productive during the day. These are just few examples but applications of internet of things are numerous.

On large scale transportation, healthcare, defence, environment monitoring, manufacturing and every other field you can imagine of can be benefited from IoT.

It is very to conceive the whole application domain of internet of things at the moment but you can clearly understand why it is such an interesting and hot topic at the moment.

Fig 2 Introduction to IoT

- ## APPLICATIONS OF IoT:

There are several application domains which will be impacted by the emerging Internet of Things. The applications can be classified based on the type of network availability, coverage, scale, heterogeneity, repeatability, user involvement and impact. We categorize the applications into four application domains:
(1) Personal and Home
(2)Enterprise
(3)Utilities
(4)Mobile.

There is a huge crossover in applications and the use of data between domains. For instance, the Personal and Home IoT produces electricity usage data in the house and makes it available to the electricity (utility) company which can in turn optimize the supply and demand in the Utility IoT. The internet enables sharing of data between different service providers in a seamless manner creating multiple business opportunities

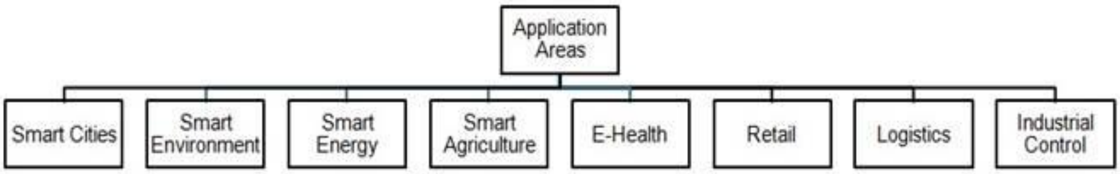


Fig 3 Application Areas

- ## ARCHITECTURE OF INTERNET OF THINGS:

Architecture of internet Of Things contains basically 4 layers:

1. Application Layer
2. Gateway and the network layer
3. Management Service layer
4. Sensor layer

1. APPLICATION LAYER:

- Lowest Abstraction Layer
- With sensors we are creating digital nervous system.
- Incorporated to measure physical quantities
- Interconnects the physical and digital world
- Collects and process the real time information

2. GATEWAY AND THE NETWORK LAYER:

- Robust and High-performance network infrastructure.
- Supports the communication requirements for latency, bandwidth or security
- Allows multiple organizations to share and use the same network independently

3. MANAGEMENT LAYER:

- Capturing of periodic sensory data
- Data Analytics (Extracts relevant information from massive amount data).
- Streaming Analytics (Process real time data)
- Ensures security and privacy of data.

4. SENSOR LAYER:

- Provides a user interface for using IoT.
- Different applications for various sectors like Transportation, Healthcare, Agriculture, Supply chains, Government, Retail etc.

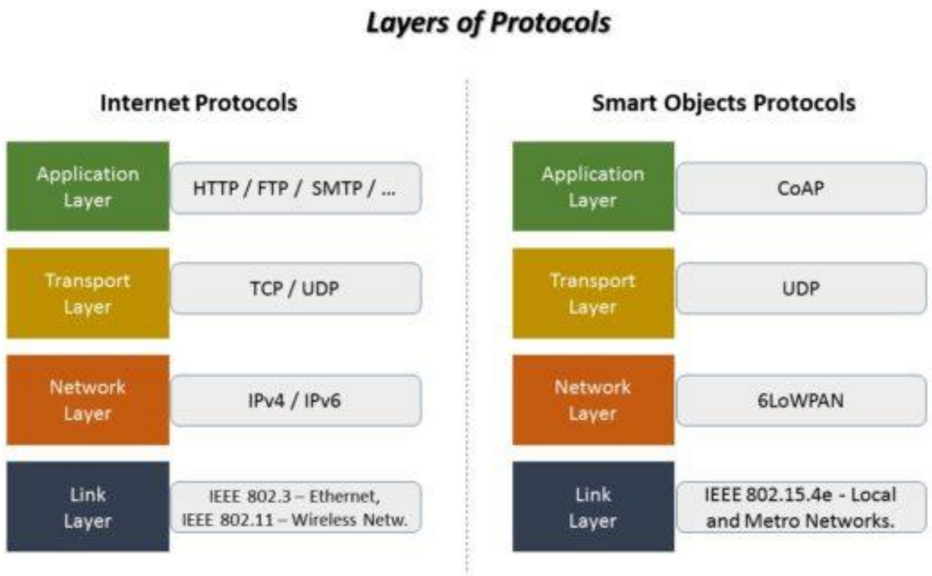


Fig 4 Layers Of Protocols

- ## BENEFITS OF INTERNET OF THINGS

1. Improved citizen's quality of life
2. Healthcare from anywhere

   Better safety, security and productivity
3. New business opportunities
4. IoT can be used in every vertical for improving the efficiency

   Creates new businesses, and new and better jobs
5. Economical growth
6. Billions of dollars in savings and new services
7. Better environment

8. Saves natural resources and trees, helps in creating a smart, greener and sustainable planet
9. Improved competitiveness
10. Competitive in providing cutting edge products/services

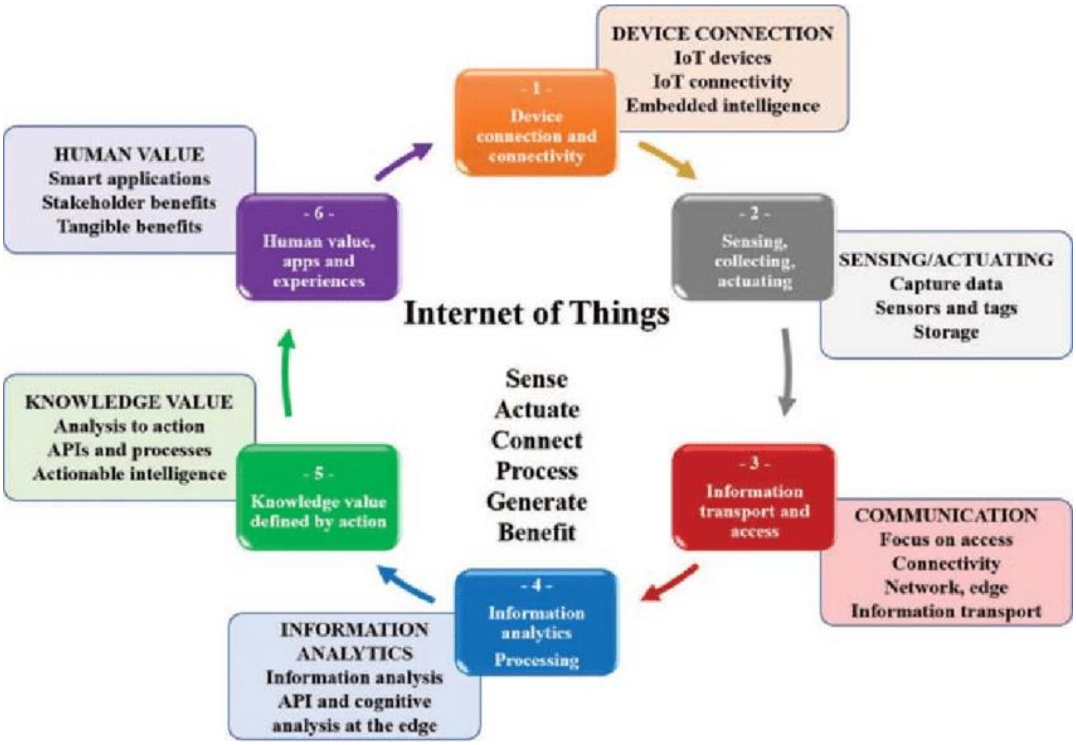


Fig 5: IoT Generated Benefits

CHAPTER 2
PROJECT DETAILS

- ## COMPONENTS USED:

  a. **Hardware components**:

     i. Node MCU Breakout Board(ESP8266)
     ii. Raspberry Pi
     iii. Android Device
     iv. Motor Driver(L298N)
     v. Motor Robot Car Chassis Kit
     vi. Batteries and Charger

  b. Software Components:

     i. Arduino IDE
     ii. Raspberry Pi

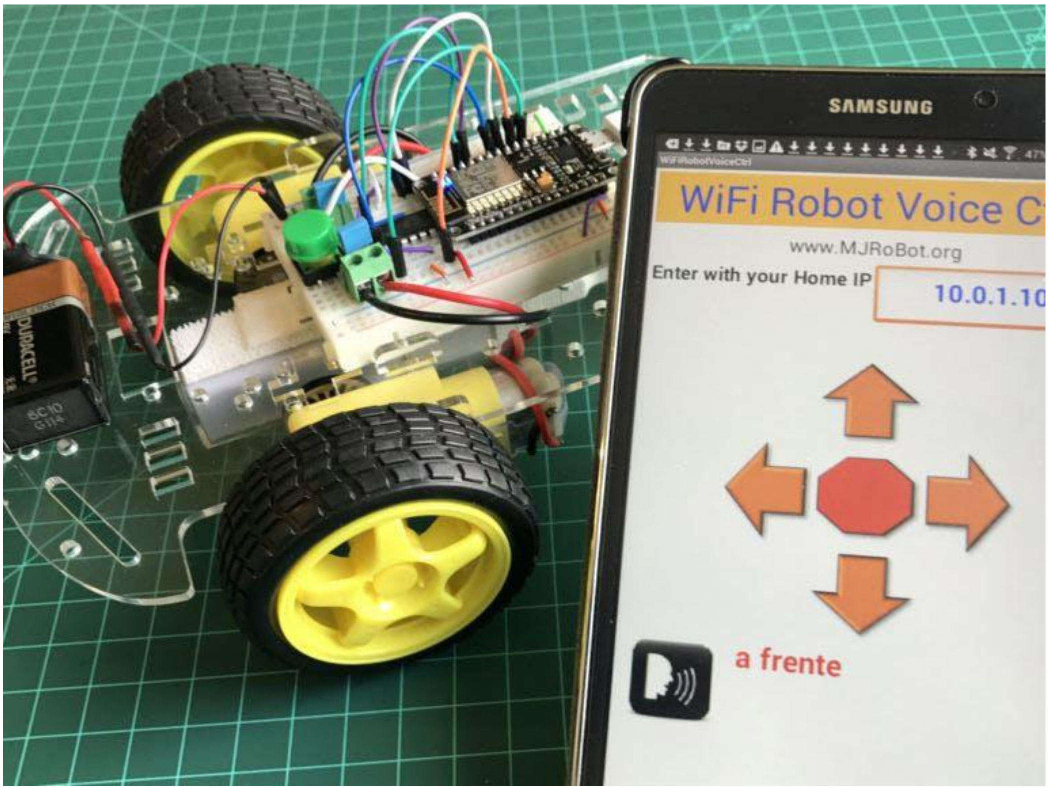


Fig.6 Robot with Controller

- ## **L298N Motor Driver:**

The L298N is a dual H-Bridge motor driver which allows speed and direction control of two DC motors at the same time. The module can drive DC motors that have voltages between 5 and 35V, with a peak current up to 2A.

Let's take a closer look at the pinout of L298N module and explain how it works. The module has two screw terminal blocks for the motor A and B, and another screw terminal block for the Ground pin, the VCC for motor and a 5V pin which can either be an input or output.

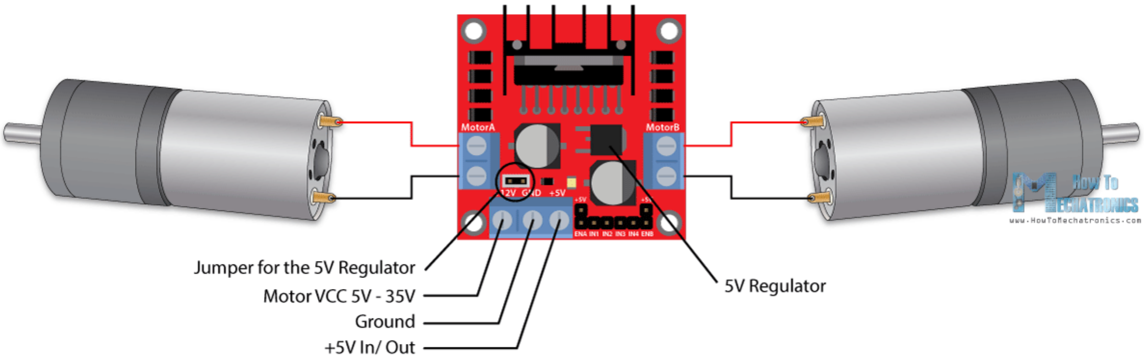


Fig.7 L298N Pinout

Next are the logic control inputs. The Enable A and Enable B pins are used for enabling and controlling the speed of the motor. If a jumper is present on this pin, the motor will be enabled and work at maximum speed, and if we remove the jumper we can connect a PWM input to this pin and in that way control the speed of the motor. If we connect this pin to a Ground the motor will be disabled.

Next, the Input 1 and Input 2 pins are used for controlling the rotation direction of the motor A, and the inputs 3 and 4 for the motor B. Using these pins we actually control the switches of the H-Bridge inside the L298N IC. If input 1 is LOW and input 2 is HIGH the motor will move forward, and vice versa, if input 1 is HIGH and input 2 is LOW the motor will move backward. In case both inputs are same, either LOW or HIGH the motor will stop. The same applies for the inputs 3 and 4 and the motor B.

CHAPTER 3

NODE MCU (ESP8266)

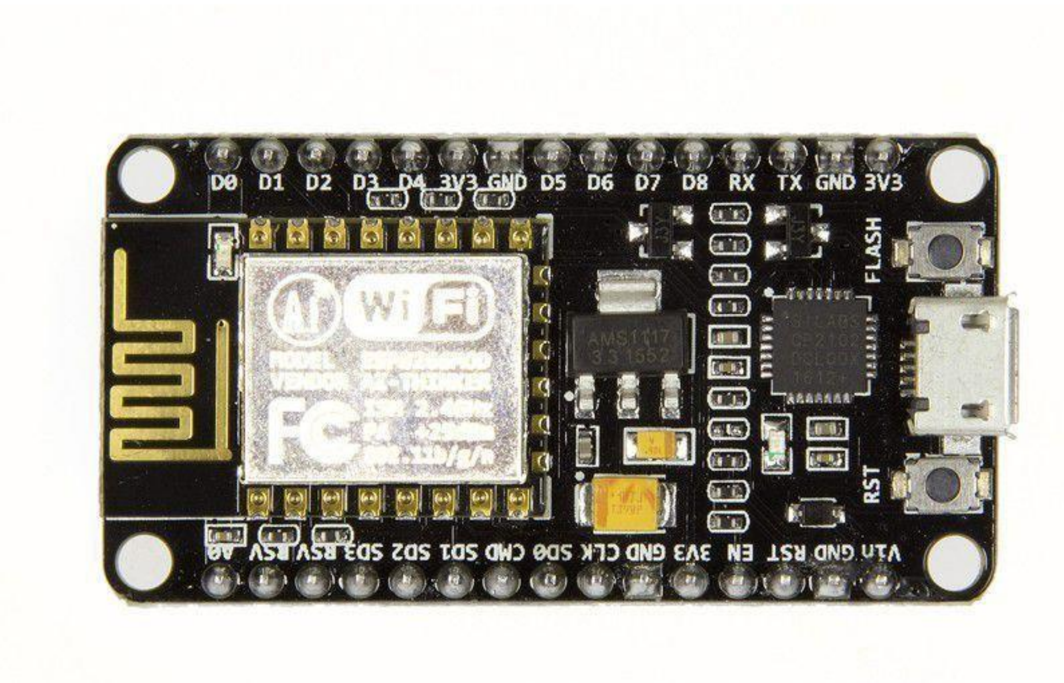


Fig 8 Node MCU

The ESP8266 Wi-Fi Module is a self-contained SOC with integrated TCP/IP protocol stack that can give any microcontroller access to your Wi-Fi network. The ESP8266 is capable of either hosting an application or offloading all Wi-Fi networking functions from another application processor. Each ESP8266 module comes pre-programmed with an AT command set firmware, meaning, you can simply hook this up to your Arduino device and get about as much WiFi-ability as a WiFi Shield offers (and that's just out of the box)! The ESP8266 module is an extremely cost-effective board with a huge, and ever growing, community.

This module has a powerful enough on-board processing and storage capability that allows it to be integrated with the sensors and other application specific devices through its GPIOs with minimal development up-front and minimal loading during runtime. Its high degree of on-chip integration allows for minimal external circuitry, including the front-end module, is designed to occupy minimal PCB area. The ESP8266 supports APSD for VoIP applications and Bluetooth co-existence interfaces, it contains a self-calibrated RF allowing it to work under all operating conditions and requires no external RF parts.

There is an almost limitless fountain of information available for the ESP8266, all of which has been provided by amazing community support. In the *Documents* section below, you will find many resources to aid you in using the ESP8266, even instructions on how to transforming this module into an IoT (Internet of Things) solution!

Espressif's ESP8266EX delivers highly integrated Wi-Fi SoC solution to meet users' continuous demands for efficient power usage, compact design and reliable performance in the Internet of Things industry.

With the complete and self-contained Wi-Fi networking capabilities, ESP8266EX can perform either as a standalone application or as the slave to a host MCU. When ESP8266EX hosts the application, it promptly boots up from the flash. The integrated highspeed cache helps to increase the system performance and optimize the system memory. Also, ESP8266EX can be applied to any microcontroller design as a Wi-Fi adaptor through SPI/SDIO or UART interfaces.

ESP8266EX integrates antenna switches, RF balun, power amplifier, low noise receive amplifier, filters and power management modules. The compact design minimizes the PCB size and requires minimal external circuitries.

Besides the Wi-Fi functionalities, ESP8266EX also integrates an enhanced version of Tensilica's L106 Diamond series 32-bit processor and on-chip SRAM. It can be interfaced with external sensors and other devices through the GPIOs. Software Development Kit (SDK) provides sample codes for various applications.

Espressif Systems' Smart Connectivity Platform (ESCP) enables sophisticated features including:

- Fast switch between sleep and wakeup mode for energy-efficient purpose;
- Adaptive radio biasing for low-power operation
- Advance signal processing
- Spur cancellation and RF co-existence mechanisms for common cellular , Bluetooth, DDR, LVDS, LCD interference mitigation.

1. Specification:

- **Wifi:**

  Certification: Wi-Fi Alliance

  Protocols: 802.11 b/g/n (HT20)

  Frequency Range: 2.4G ~ 2.5G (2400M ~ 2483.5M)

  TX Power: 802.11 b: +20 dBm

  802.11 g: +17 dBm

  802.11 n: +14 dBm

  Rx Sensitivity: 802.11 b: –91 dbm (11 Mbps)

  802.11 g: –75 dbm (54 Mbps)

  802.11 n: –72 dbm (MCS7)

  Antenna: PCB Trace, External, IPEX Connector, Ceramic Chip

- **Hardware:**

  CPU: Tensilica L106 32-bit processor

  Peripheral Interface: UART/SDIO/SPI/I2C/I2S/IR Remote Control

  GPIO/ADC/PWM/LED Light & Button

  Operating Voltage: 2.5V ~ 3.6V

  Operating Current : Average value-80 mA

  Operating Temperature Range –40°C ~ 125°C

  Package Size QFN32-pin (5 mm x 5 mm)

  External Interface –

- **Software:**

  Wi-Fi Mode: Station/SoftAP/SoftAP+Station

  Security: WPA/WPA2

  Encryption: WEP/TKIP/AES

  Firmware Upgrade: UART Download / OTA (via network)

  Software Development: Supports Cloud Server Development / Firmware and SDK for fast on-chip programming

  Network Protocols: IPv4, TCP/UDP/HTTP

  User Configuration: AT Instruction Set, Cloud Server, Android/iOS App

`

2. **Pin Definition**:

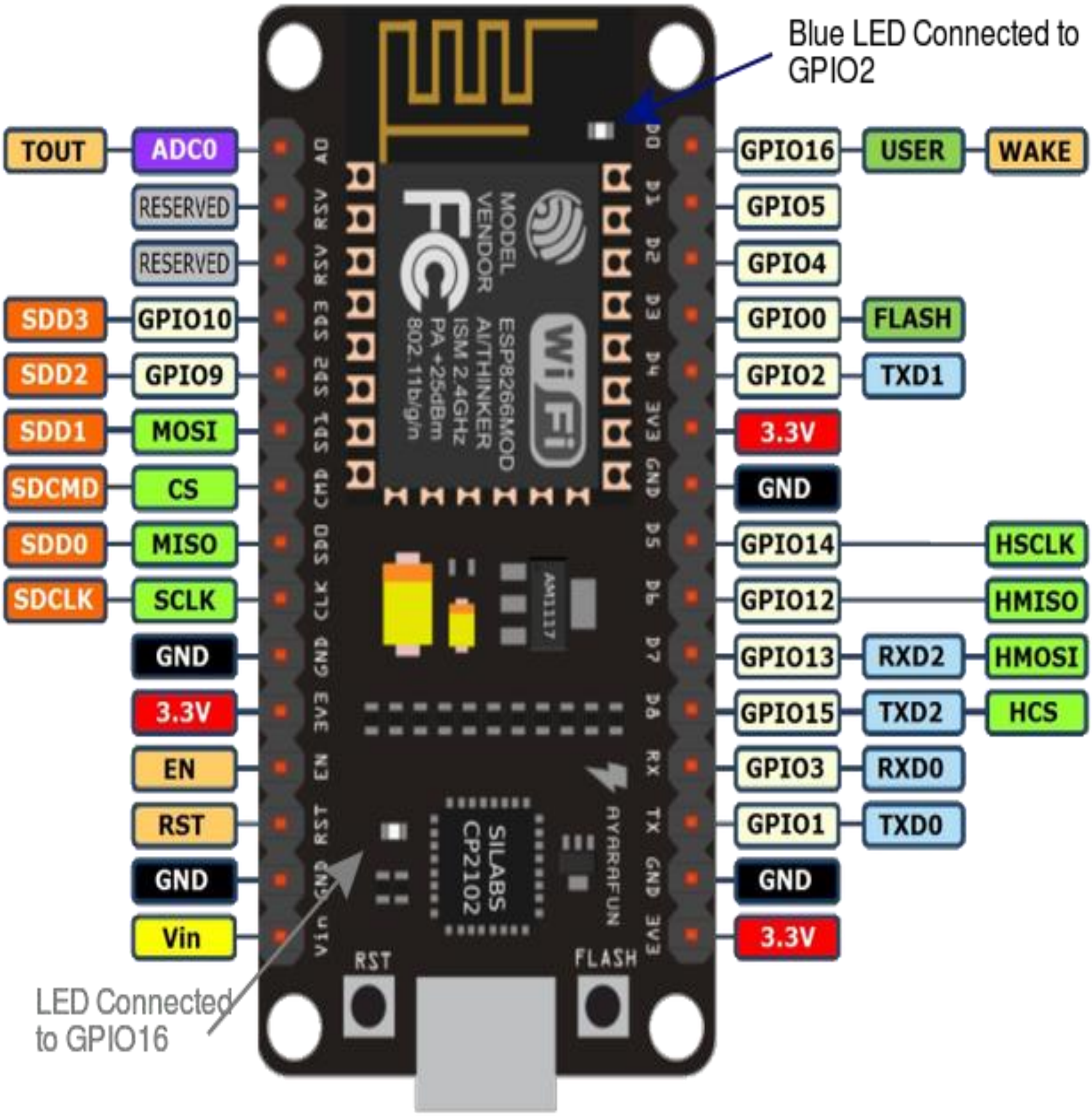


Fig 9 Node MCU Pinout

`

3. Peripheral Interface:

1. General Purpose Input/Output Interface (GPIO):

ESP8266EX has 17 GPIO pins which can be assigned to various functions by programming the appropriate registers.

Each GPIO PAD can be configured with internal pull-up or pull-down (XPD_DCDC can only be configured with internal pull-down, other GPIO PAD can only be configured with internal pull-up), or set to high impedance. When configured as an input, the data are stored in software registers; the input can also be set to edge-trigger or level trigger CPU interrupts. In short, the IO pads are bi-directional, non-inverting and tristate, which includes input and output buffer with tristate control inputs.

These pins, when working as GPIOs, can be multiplexed with other functions such as I2C, I2S, UART, PWM, and IR Remote Control, etc.

2. Serial Peripheral Interface (SPI/HSPI):

ESP8266EX has two SPIs.

• One general Slave/Master SPI

• One general Slave HSPI

Functions of all these pins can be implemented via hardware.

- General SPI:

Pin Name	Pin Num	io	Function NAme
SDIO_CLK	21	IO6	SPICLK
SDIO_DATA0	22	IO7	SPIQ/MISO
SDIO_DATA1	23	IO8	SPID/MOSI
SDIO_DATA2	18	IO9	SPIHD
SDIO_DATA3	19	IO10	SPIWP
U0TXD	26	IO1	SPICS1
GPIO0	15	IO0	SPICS2
SDIO_CMD	20	IO11	SPICS0

Table 1: Pin Definitions of SPIs

Pin Name	Pin Num	IO	Function Name
MTMS	9	IO14	I2C_SCL
GPIO2	14	IO2	I2C_SDA

Table 2: Pin Definitions of I2C

3. **I2C Interface:**

ESP8266EX has one I2C, which is realized via software programming, used to connect with other microcontrollers and other peripheral equipment such as sensors. The pin definition of I2C is as below.

4. **Universal Asynchronous Receiver Transmitter (UART):**

ESP8266EX has two UART interfaces UART0 and UART1. Data transfers to/from UART interfaces can be implemented via hardware. The data transmission speed via UART interfaces reaches 115200 x 40 (4.5 Mbps). UART0 can be used for communication. It supports flow control. Since UART1 features only data transmit signal (TX), it is usually used for printing log.

5. **Pulse-Width Modulation (PWM):**

ESP8266EX has four PWM output interfaces. They can be extended by users themselves. The pin definitions of the PWM interfaces are defined as below.

Pin Name	Pin Number	IO	Function Name
MTD	10	IO12	PWM0
MTDO	13	IO15	PWM1
MTMS	9	IO14	PWM2
GPIO4	16	IO4	PWM3

Table 3: Pin Definitions of PWM

6. **IR Remote Control:**

ESP8266EX currently supports one infrared remote-control interface. For detailed pin definitions, please see Table below.

Pin Name	Pin Num	IO	Function Name
MTMS	9	IO14	IR TX
GPIO5	24	IO5	IR RX

Table 4: Pin Definitions of IR Remote Control

The functionality of Infrared remote control interface can be implemented via software programming. NEC coding, modulation, and demodulation are supported by this interface. The frequency of modulated carrier signal is 38 kHz, while the duty ratio of the square wave is 1/3. The transmission range is around 1m which is determined by two factors: one is the maximum current drive output, the other is internal current-limiting resistance value in the infrared receiver. The larger the resistance value, the lower the current, so is the power, and vice versa.

4. **CPU, Memory, and Flash:**

1**. CPU:**

The ESP8266EX integrates a Tensilica L106 32-bit RISC processor, which achieves extra low power consumption and reaches a maximum clock speed of 160 MHz. The Real-Time Operating System (RTOS) and Wi-Fi stack allow 80% of the processing power to be available for user application programming and development. The CPU includes the interfaces as below:

• Programmable RAM/ROM interfaces (iBus), which can be connected with memory controller, and can also be used to visit flash

. • Data RAM interface (dBus), which can connected with memory controller.

• AHB interface which can be used to visit the register.

2. Memory:

ESP8266EX Wi-Fi SoC integrates memory controller and memory units including SRAM and ROM. MCU can access the memory units through iBus, dBus, and AHB interfaces. All memory units can be accessed 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, SRAM space available to users is assigned as below.

• RAM size < 50 kB, that is, when ESP8266EX is working under the Station mode and connects to the router, the maximum programmable space accessible in Heap + Data section is around 50 kB.

• There is no programmable ROM in the SoC. Therefore, user program must be stored in an external SPI flash.

3. External Flash:

ESP8266EX uses external SPI flash to store user programs, and supports up to 16 MB memory capacity theoretically. The minimum flash memory of ESP8266EX is shown below:

• OTA disabled: 512 kB at least

• OTA enabled: 1 MB at least

CHAPTER 4
RASPBERRY PI

Fig 10 Raspberry Pi 3

The Raspberry Pi is a remarkable device: a fully functional computer in a tiny and low-cost package. Whether you're looking for a device you can use to browse the web or play games, are interested in learning how to write your own programs, or are looking to create your own circuits and physical devices, the Raspberry Pi – and its amazing community – will support you every step of the way. The Raspberry Pi is known as a single-board computer, which means exactly what it sounds like: it's a computer, just like a desktop, laptop, or smartphone, but built on a single printed circuit board. Like most single-board computers, the Raspberry Pi is small – roughly the same footprint as a credit card – but that doesn't mean it's not powerful: a Raspberry Pi can do anything a bigger and more power-hungry computer can do, though not necessarily as quickly. The Raspberry Pi family was born from a desire to encourage more hands-on computer education around the world. Its creators, who joined together to form the non-profit Raspberry Pi Foundation, had little idea that it would prove so popular: the few thousand built in 2012 to test the waters were immediately sold out, and millions have been shipped all over the world in the years since. These boards have found their ways into homes, classrooms, offices, data centres, factories, and even self-piloting boats and spacefaring balloons. Various models of Raspberry Pi have been released since the original Model B, each bringing either improved specifications or features specific to a particular use-case. The Raspberry Pi Zero family, for example, is a tiny version of the full-size Raspberry Pi which drops a few features – in particular the multiple USB ports and wired network port – in favour of a significantly smaller layout and lowered power needs.

All Raspberry Pi models have one thing in common, though: they're compatible, meaning that software written for one model will run on any other model. It's even possible to take the very latest version of the Raspberry Pi's operating system and run it on an original pre-launch Model B prototype. It will run more slowly, it's true, but it will still run. Throughout this book you'll be learning about the Raspberry Pi 3 Model B+, the latest and most popular version of the Raspberry Pi. What you learn, though, can be easily applied to other models in the Raspberry Pi family, so don't worry if you're using a different version.

Unlike a traditional computer, which hides its inner workings in a case, a Raspberry Pi has all its components, ports, and features out on display – although you can buy a case to provide extra protection, if you'd prefer. This makes it a great tool for learning about what the various parts of a computer do, and also makes it easy to learn what goes where when it comes time to plug in the various extras – known as peripherals – you'll need to get started.

While it may look like there's a lot packed into the tiny board, the Raspberry Pi is very simple to understand – starting with its components, the inner workings that make the device tick.

- ## Components of RPi:

Like any computer, the Pi is made up of various different components, each of which has a role to play in making it work. The first, and arguably most important, of these can be found just above the centre point on the top side of the board.

Fig 11. System on Chip

The name system-on-chip is a great indicator of what you would find if you prised the metal cover off: a silicon chip, known as an integrated circuit, which contains the bulk of the Raspberry Pi’s system. This includes the central processing unit (CPU), commonly thought of as the ‘brain’ of a computer, and the graphics processing unit (GPU), which handles the visual side of things. A brain is no good without memory, however, and on the underside of the Raspberry Pi you’ll find exactly that: another chip, which looks like a small, black, plastic square (Figure 11). This is the Pi’s random access memory (RAM). When you’re working on the Pi, it’s the RAM that holds what you’re doing; only when you save your work will it be written to the microSD card. Together, these components form the Pi’s volatile and non-volatile memories: the volatile RAM loses its contents whenever the Pi is powered off, while the non-volatile microSD card keeps its contents.

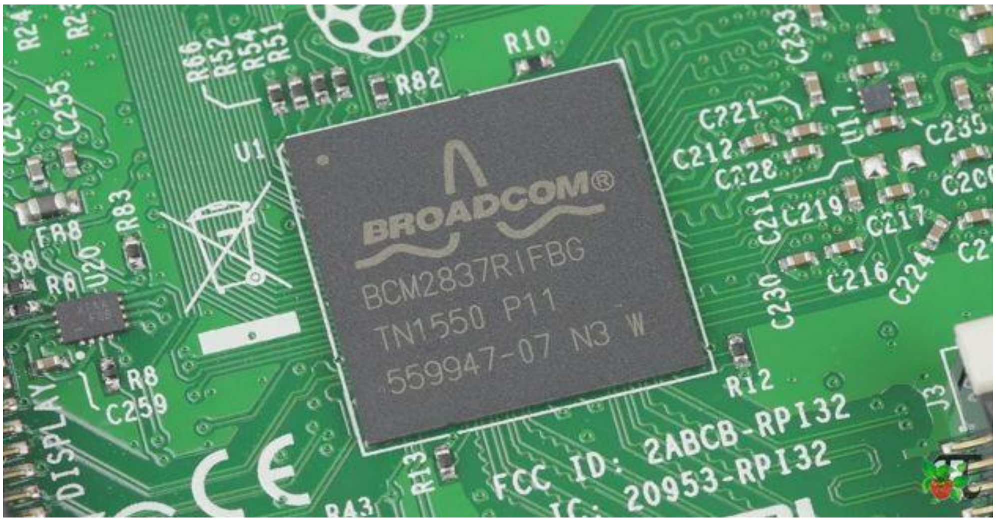


Fig 12. The Raspberry Pi's random-access memory (RAM)

Turning the board over again you'll find another metal lid to the upper-right, this one featuring an etched Raspberry Pi logo (Figure 13, overleaf). This covers the radio, the component which gives the Raspberry Pi the ability to communicate with devices wirelessly. The radio itself acts as two main components, in fact: a WiFi radio, for connecting to computer networks; and a Bluetooth radio, for connecting to peripherals like mice and for sending data to or receiving data from nearby smart devices like sensors or smartphones.

Fig 13. The Raspberry Pi's radio module

Another black, plastic-covered chip can be seen to the bottom edge of the board, just behind the middle set of USB ports. This is the network and USB controller, and is responsible for running the Ethernet port and the four USB ports. A final black chip, much smaller than the rest, can be found a little bit above the micro USB power connector to the upper-left of the board (Figure 14); this is known as a power management integrated circuit (PMIC), and handles turning the power that comes in from the micro USB port into the power the Pi needs to run.

Fig 14. RPi's(PMIC)

- ## Raspberry Pi's ports:

The Raspberry Pi has a range of ports, starting with four Universal Serial Bus (USB) ports (Figure 15) to the middle and right-hand side of the bottom edge. These ports let you connect any USB-compatible peripheral, from keyboards and mice to digital cameras and flash drives, to the Pi. Speaking technically, these are known as USB 2.0 ports, which means they are based on version two of the Universal Serial Bus standard.

`

Fig 15. The Raspberry Pi’s USB ports

To the left of the USB ports is an Ethernet port, also known as a network port (Figure 16). You can use this port to connect the Raspberry Pi to a wired computer network using a cable with what is known as an RJ45 connector on its end. If you look closely at the Ethernet port, you’ll see two light-emitting diodes (LEDs) at the bottom; these are status LEDs, and let you know that the connection is working.

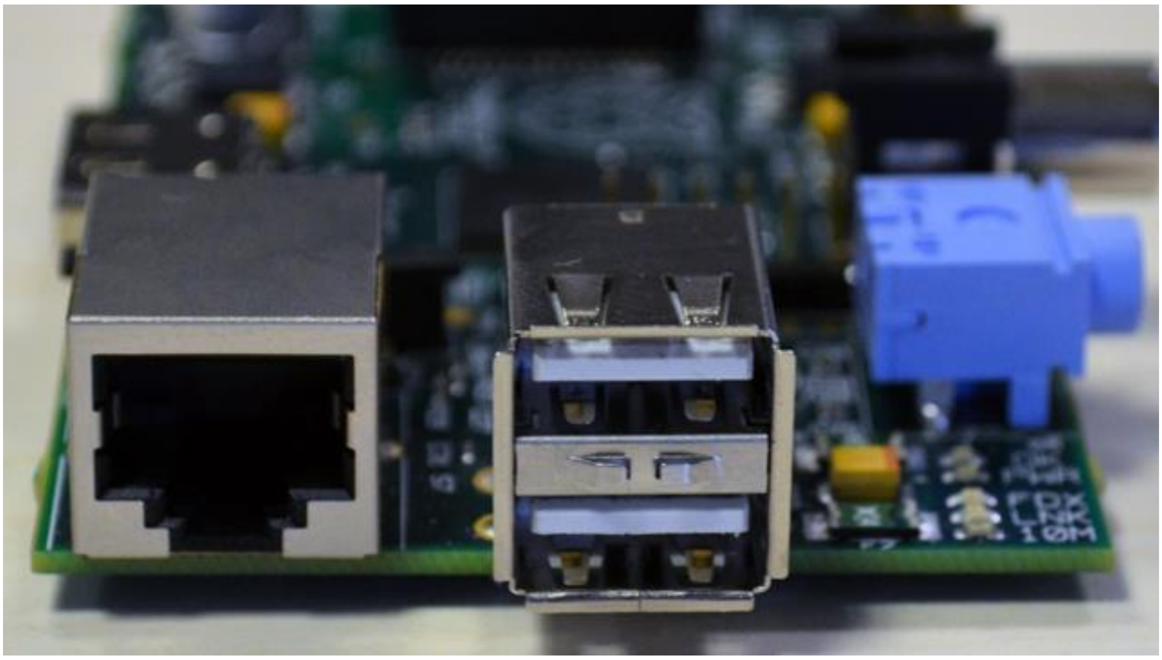


Fig 16. The Raspberry Pi’s Ethernet ports

Just above the Ethernet port, on the left-hand edge of the Raspberry Pi, is a 3.5 mm audio-visual (AV) jack (Figure 17). This is also known as the headphone jack, and it can be used for that exact purpose – though you'll get better sound connecting it to amplified speakers rather than headphones. It has a hidden, extra feature, though: as well as audio, the 3.5 mm AV jack carries a video signal which can be connected to TVs, projectors, and other displays that support a composite video signal using a special cable known as a tip-ring-ring-sleeve (TRRS) adapter.

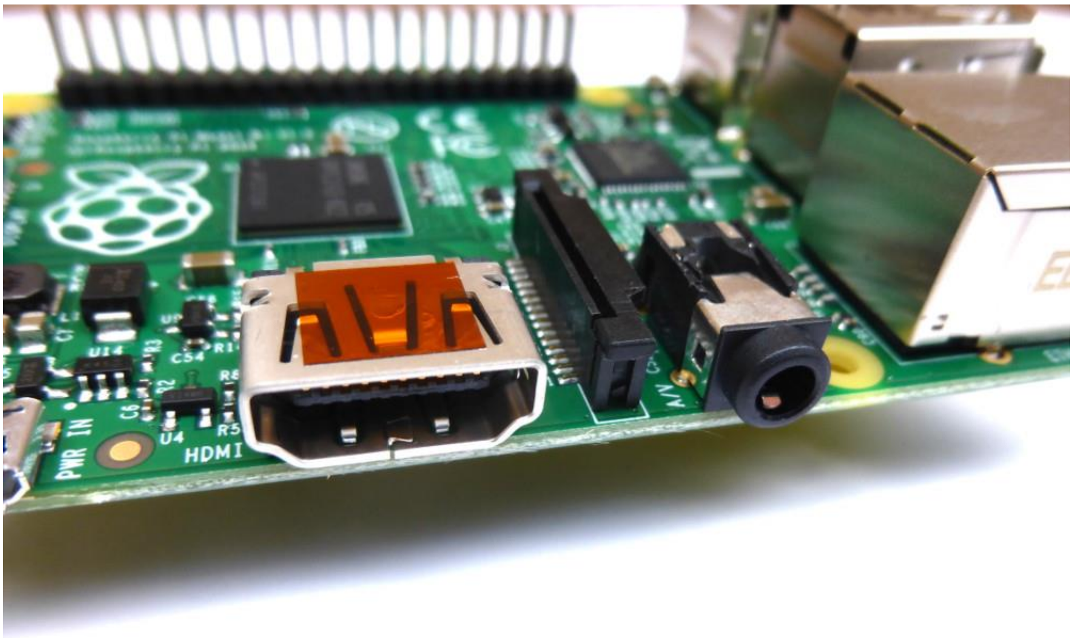


Fig 17. The Raspberry Pi's 3.5 mm AV jack

Directly above the 3.5 mm AV jack is a strange-looking connector with a plastic flap which can be pulled up; this is the camera connector, also known as the Camera Serial Interface (CSI) (Figure 18). This allows you to use the specially designed Raspberry Pi Camera Module.

Fig 18. The Raspberry Pi’s camera connector

Above that, still on the left-hand edge of the board, is the High-Definition Multimedia Interface (HDMI) port (Figure 19), which is the same type of connector you’ll find on a games console, set-top box, and TV. The multimedia part of its name tells you that it carries both audio and video signals, while high-definition tells you that you can expect excellent quality. You’ll use this to connect the Raspberry Pi to your display device, whether that’s a computer monitor, TV, or projector.

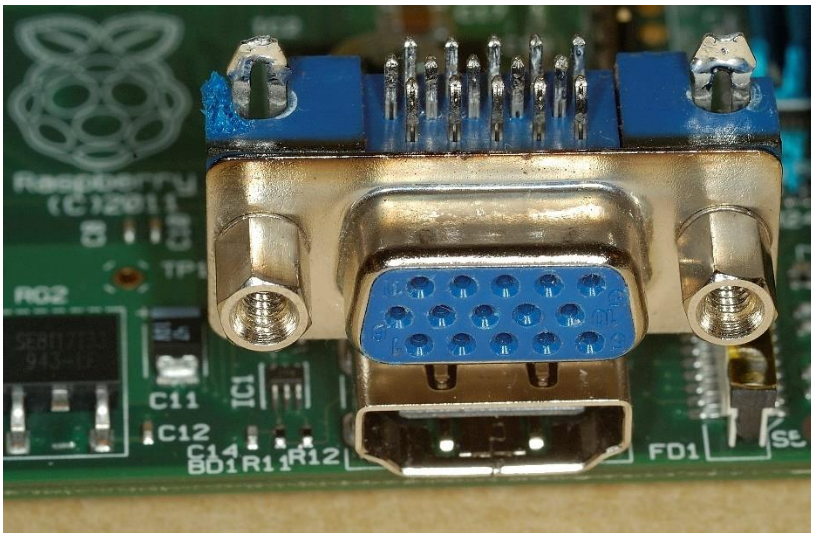


Fig 19.The Raspberry Pi’s HDMI Port

Above the HDMI port is a micro USB power port (Figure 15), which you'll use to connect the Raspberry Pi to a power source. The micro USB port is a common sight on smartphones, tablets, and other portable devices. So you could use a standard mobile charger to power the Pi, but for best results you should use the official Raspberry Pi USB Power Supply.

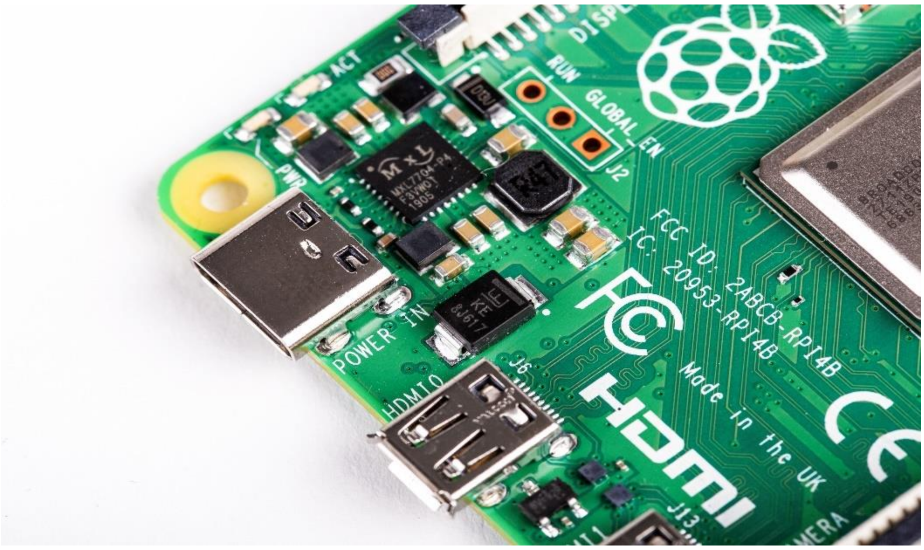


Fig. 20 The Raspberry Pi's micro USB power port

At the top edge of the board is another strange-looking connector (Figure 18), which at first glance appears to be identical to the camera connector. This, though, is the exact opposite: a display connector, or Display Serial Interface (DSI), designed for use with the Raspberry Pi Touch Display.

At the right-hand edge of the board you'll find 40 metal pins, split into two rows of 20 pins. This is the GPIO (general-purpose input/output) header, a feature of the Raspberry Pi used to talk to additional hardware from LEDs and buttons all the way to temperature sensors, joysticks, and pulse-rate monitors. You'll learn more about the GPIO header in Chapter 6, Physical computing with Scratch and Python. Just below and to the left of this header is another, smaller header with four pins: this is used to connect the Power over Ethernet (PoE) HAT, an optional add-on which lets the Raspberry Pi receive power from a network connection rather than the micro USB socket.

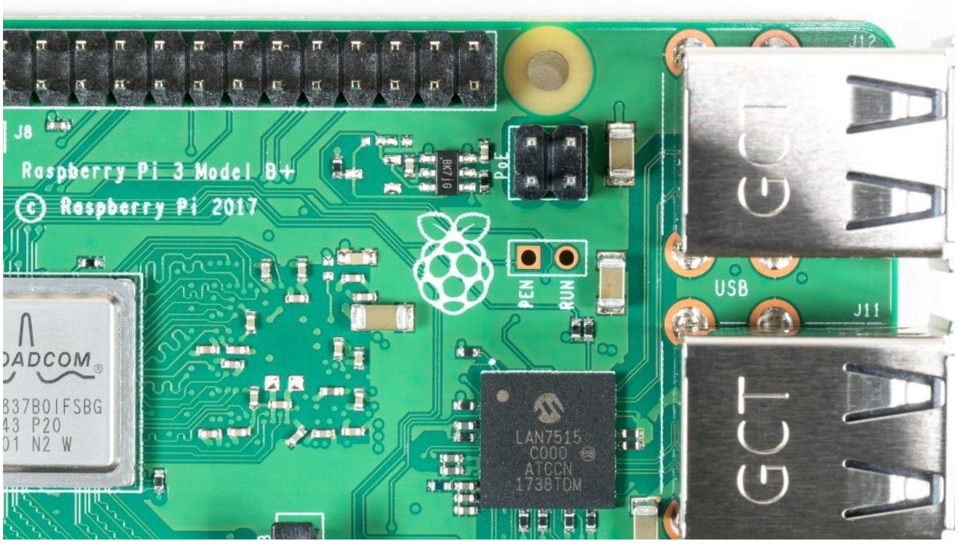


Fig. 21 The Raspberry Pi's pins

There's one final port on the Raspberry Pi, but you won't see it on the top. Turn the board over and you'll find a microSD card connector on the opposite side of the board to the display connector. This is the Raspberry Pi's storage: the microSD card inserted in here contains all the files you save, all the software you install, and the operating system that makes the Raspberry Pi run.

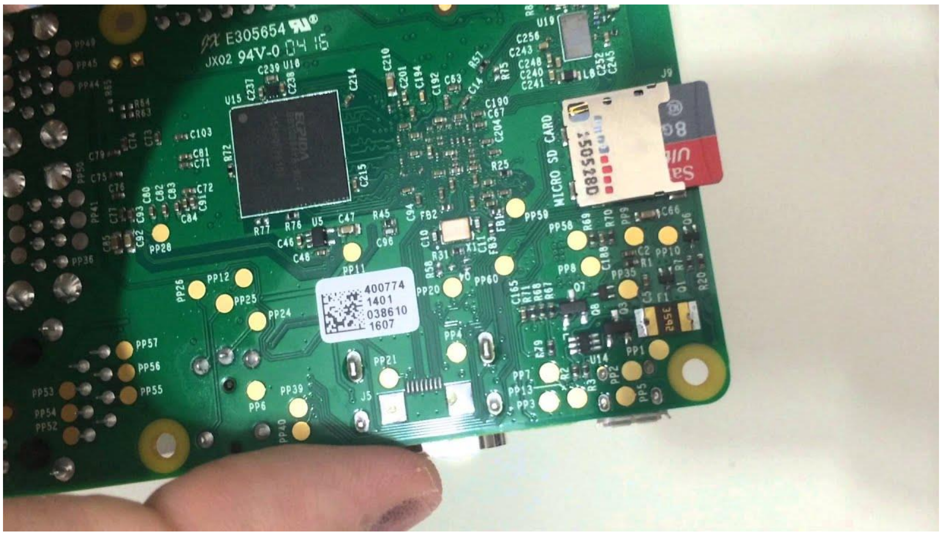


Fig.22 The Raspberry Pi's micro SD card Connector

- **Raspberry Pi’s peripherals:**

A Raspberry Pi by itself can’t do very much, just the same as a desktop computer on its own is little more than a door-stop. To work, the Raspberry Pi needs peripherals: at the minimum, you’ll need a microSD card for storage; a monitor or TV so you can see what you’re doing; a keyboard and mouse to tell the Pi what to do; and a 5 volt (5 V) micro USB power supply rated at 2.5 amps (2.5 A) or better. With those, you’ve got yourself a fully functional computer. You’ll learn how to connect all these peripherals to your Raspberry Pi Getting started with your Raspberry Pi. Those aren’t the end of the peripherals you can use with your Pi, though. Official accessories produced by the Raspberry Pi Foundation include: the Raspberry Pi Case, which helps protect the Pi while you’re using it without blocking your access to its various ports, The Raspberry Pi Camera Module; the Raspberry Pi Touch Display, which connects to the display port and provides both a video display and a tablet-style touchscreen interface; and the Sense HAT (Figure 23), A wide assortment of third-party accessories are also available, ranging from kits to turn a Raspberry Pi into a laptop or tablet, to add-ons which give it the ability to understand your speech and even talk back to you. While it’s tempting to run out and fill a shopping trolley, though, remember that you’ll need to learn to walk with your Raspberry Pi before taking on the hardware equivalent of a marathon!

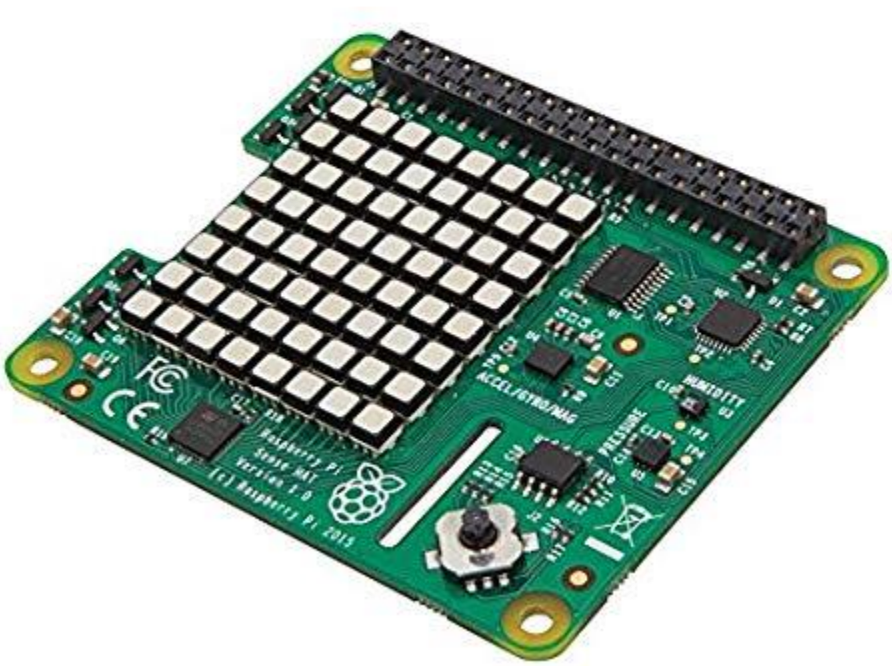


Fig.23 The Raspberry Pi’s Sense Hat

CHAPTER 5
PROJECT DESIGN AND DEVELOPMENT

- **Steps followed for moving robot:**

Step 1: The H-Bridge L298N

In order to drive the motors, we will use the L298N a Quadruple Half-H Drivers H-Bridge. According its Datasheet, The L298 and L298N devices are quadruple high-current half-H drivers. The L298 is designed to provide bidirectional drive currents of up to 1 A at voltages from 4.5 V to 36 V. The L298N is designed to provide bidirectional drive currents of up to 600-mA at voltages from 4.5 V to 36 V. Both devices are solenoids, DC and bipolar stepping motors, as well as supply applications.

We will use the IC directly with the NodeMCU, not using a shield, that is also common.

Our Robot has two wheels, that are driven by 2 DC motors:

- LEFT Motor
- RIGHT Motor

We will connect the motors to our H-Bridge as shown at above diagram. The NodeMCU will have 6 GPIOs that will command those motors.

```
int rightMotor2 = 13;   // D7 - right Motor -
int rightMotor1 = 15;   // D8 - right Motor +
int leftMotor2 = 0;     // D3 - left Motor -
int leftMotor1 = 2;     // D4 - left Motor +
int eneLeftMotor = 12;  // D6 - enable Mortor Left
int eneRightMotor = 14; // D5 - enable Mortor Right
```

For example, to drive LEFT motor FORWARD you must put:

- HIGH at pin D4 (left Motor +) and
- LOW at pin D3 (left Motor -)

For the RIGHT Motor you must do the opposite:

- HIGH at pin D8 (right Motor +) and
- LOW at pin D7 (left Motor -)

Due the way that my motors are assembly and the robot moves, the above combination will drive both motors in the same direction pushing the Robot forward.

`

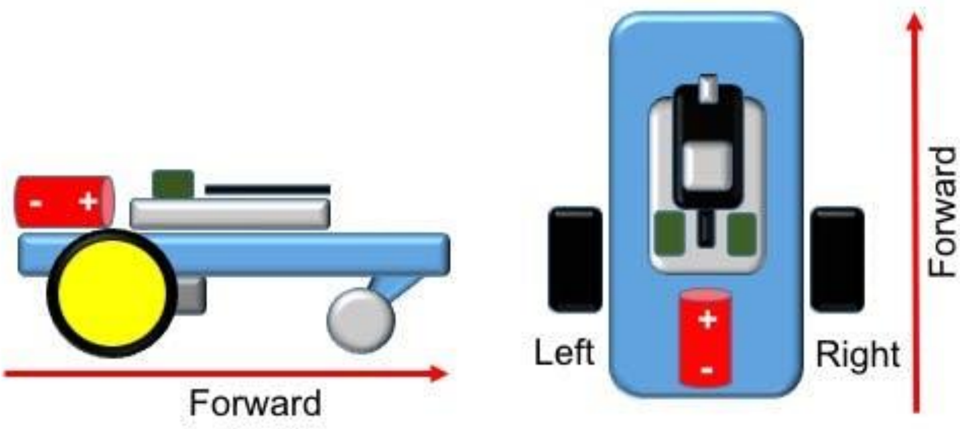


Fig.24 Robot Movements

Look the above diagram that define how the robot will move. This is important for proper definition of your variables.

In the previous step we learned how we can move the robot foreword, let's think now how to move it around. Will drive robot forward will only work if the enable pins (eneLeftMotor and eneRightMotor) are both in HIGH. You can connect those L293D pins (1 and 9) directly to +VCC and forget them. I do not like it, because for a full stop, those pins are better to be in LOW. Also, associating the enable pins to a NodeMCU PWM digital outputs will allow you to control motor speed (using digital values as HIGH and LOW, will set the speed to MAX and ZERO, respectively).

So, for creating a function to drive our robot forward, we should pu together a code like this

```
void forwardMotor(void)
{
  digitalWrite(eneLeftMotor,HIGH);
  digitalWrite(eneRightMotor,HIGH);

  digitalWrite(leftMotor1,HIGH);
  digitalWrite(leftMotor2,LOW);
  digitalWrite(rightMotor1,HIGH);
  digitalWrite(rightMotor2,LOW);
}
```


`

Step 2: Directions

In the previous step we learned how we can move the robot forward, let's think now how to move it around.

We will define 5 possible commands:

- STOP
- Turn to LEFT
- Turn to RIGHT
- REVERSE
- FORWARD

The first command "STOP" is simple to accomplish. All H-Bridge inputs must be LOW, this way the motors will not be moving:

```
void stopMotor(void)
{
  digitalWrite(eneLeftMotor,LOW);
  digitalWrite(eneRightMotor,LOW);
  digitalWrite(leftMotor1,LOW);
  digitalWrite(leftMotor2,LOW);
  digitalWrite(rightMotor1,LOW);
  digitalWrite(rightMotor2,LOW);
}
```

The same way let's think about turn the robot to one direction, let's say LEFT. Consider that the robot is going Forward, if we want to turn to left, we can have two situations:

- Stop LEFT motor and keep RIGHT going FW
- Reverse LEFT Motor and keep RIGHT going FW

At first situation, the robot will do an arc trajectory to the left. On the second, the robot will turn in its way to the left. We will implement the second one.

```
Void leftMotor(void)
{
  digitalWrite(eneLeftMotor,HIGH);
  digitalWrite(eneRightMotor,HIGH);

  digitalWrite(leftMotor1,LOW);
  digitalWrite(leftMotor2,HIGH);
  digitalWrite(rightMotor1,HIGH);
  digitalWrite(rightMotor2,LOW);
}
```

Step 3: Completing the Hardware

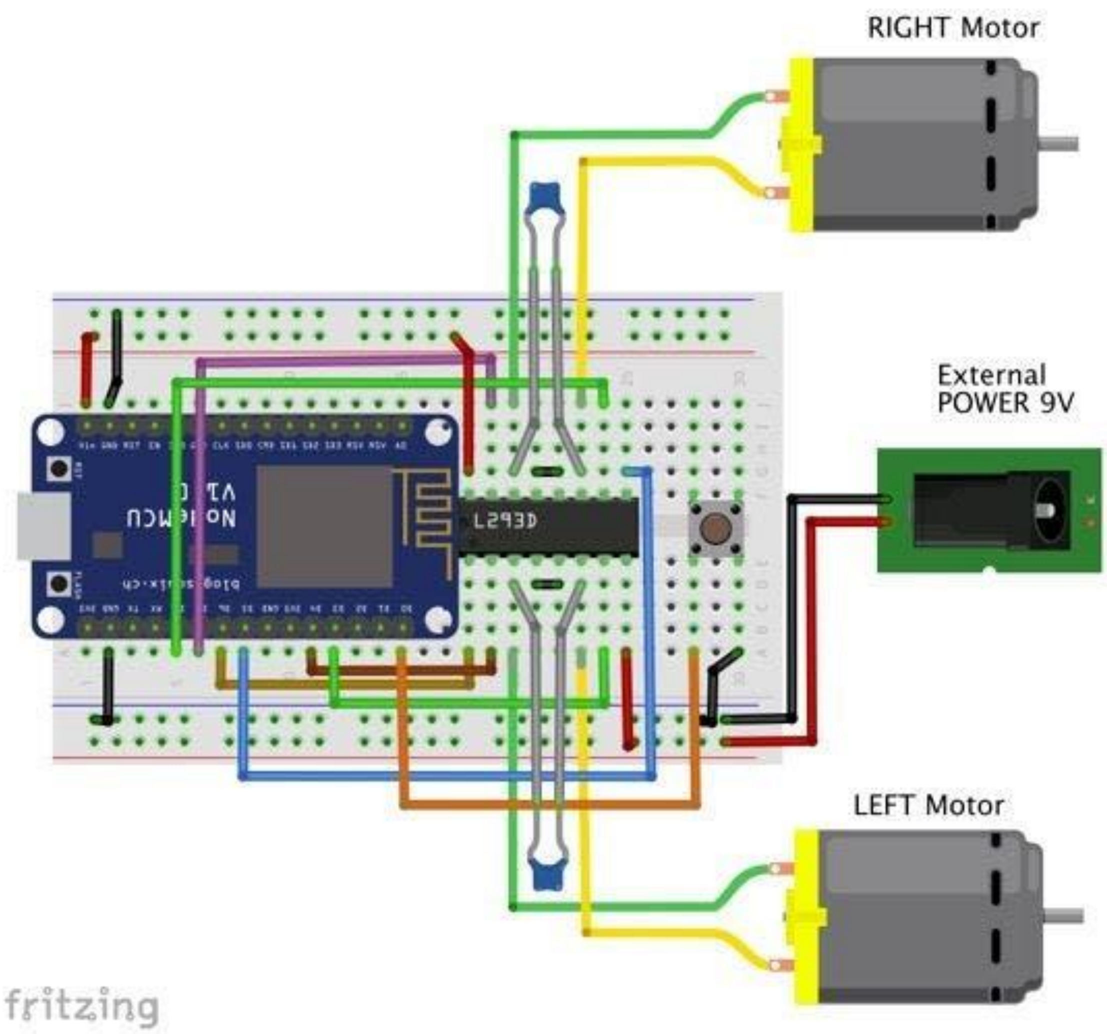


Fig.25 Connections of Robot

Follow the above diagram and complete the Robot HW. You can make simple tests to see you your code are OK. It's good for simplify tests, to introduce a "Button" for starting your robot. We will install it at NodeMCU port D0 as shown at diagram.

When you press the button, the robot will start doing the movements defined on loop(). It will continuous until you press "Reset" on NodeMCU. Pressing the button, the cycle repeats

Step 4: Connecting the NodeMCU to Local Wi-Fi

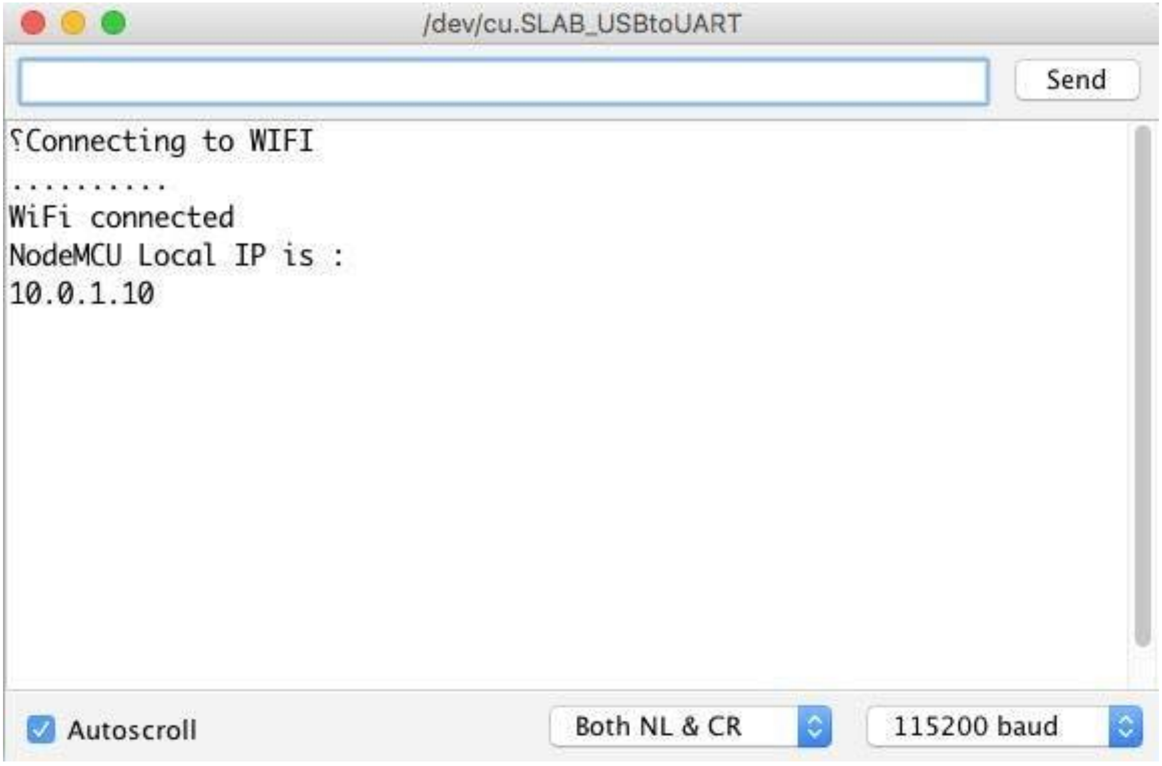


Fig.26 Wi-Fi and NodeMCU Connection

Let's connect the NodeMCU to our local WiFi and check its IP address. For that, let's use the below small program:

```
#include <ESP8266WiFi.h>
WiFiClient client;
WiFiServer server(80);
const char* ssid = "shiv";
const char* password = "ssssssss";
void setup()
{
  Serial.begin(115200);
  connectWiFi();
```

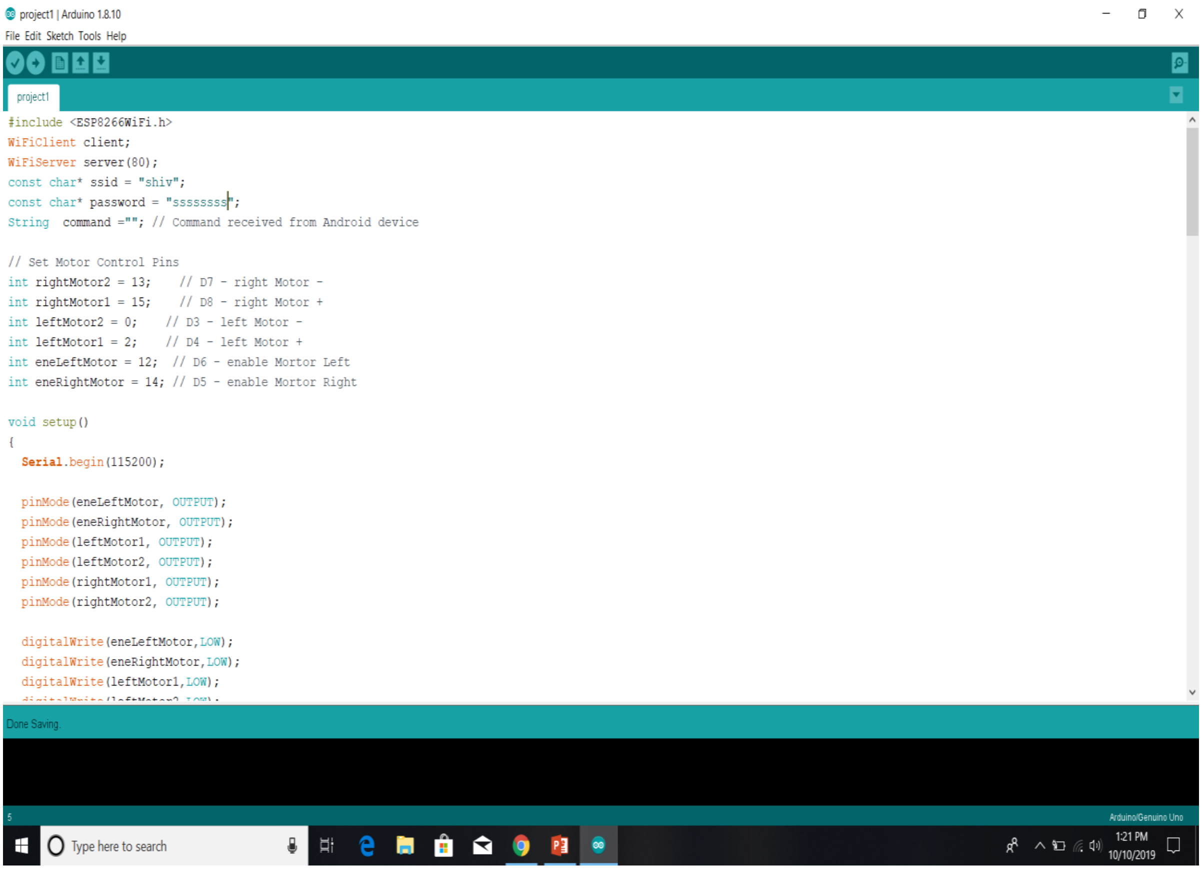
```
  server.begin();
}
void loop()
{
}
/* connecting Wi-Fi */
void connectWiFi()
{
  Serial.println("Connecting to WIFI");
  WiFi.begin(ssid, password);
  while ((!(WiFi.status() == WL_CONNECTED)))
  {
    delay(300);
    Serial.print("..");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("NodeMCU Local IP is : ");
  Serial.print((WiFi.localIP()));
}
```


`

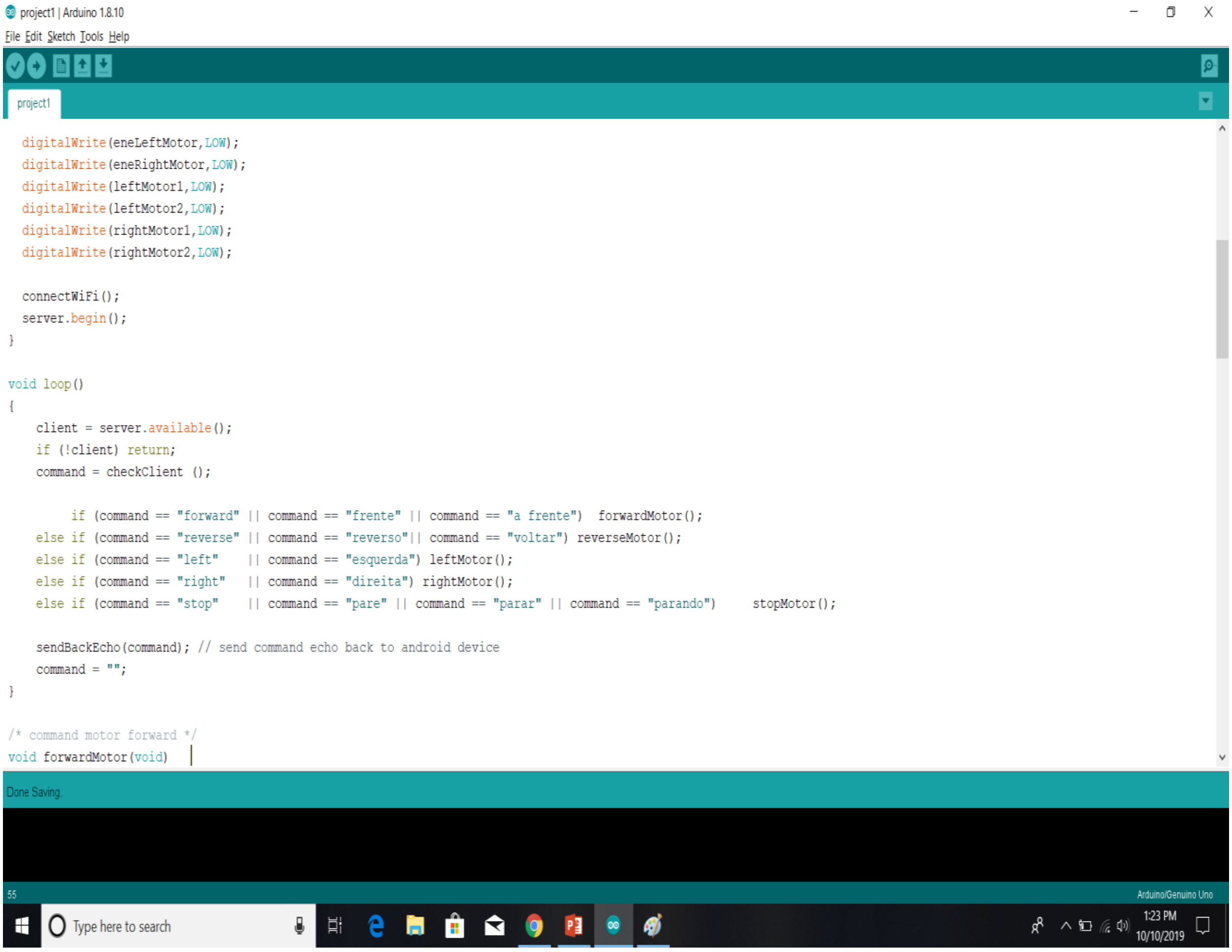
CHAPTER 6

SOURCE CODE

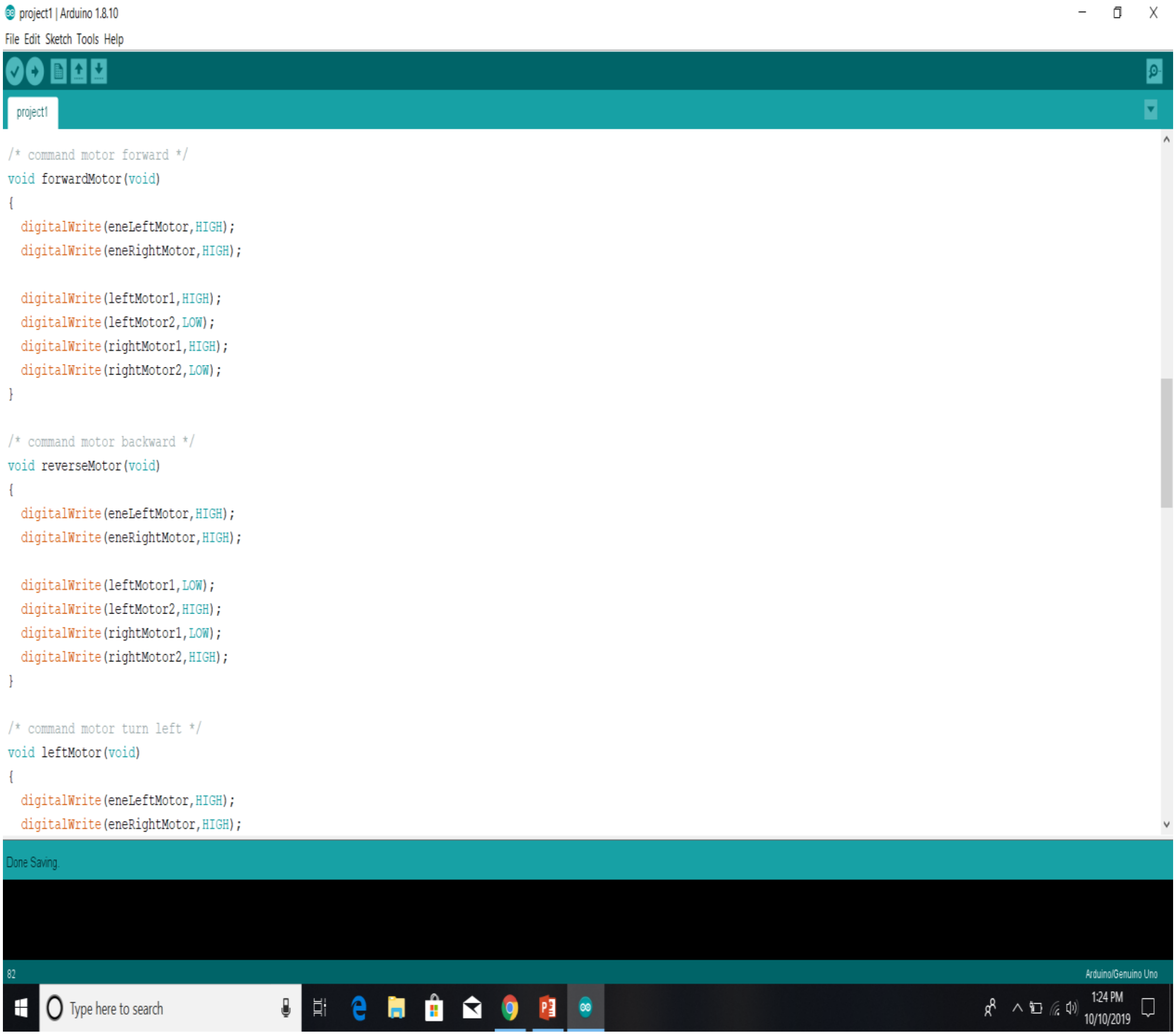
- ## Source Code in Arduino IDE:




```
#include <ESP8266WiFi.h>
WiFiClient client;
WiFiServer server(80);
const char* ssid = "shiv";
const char* password = "sssssss";
String  command =""; // Command received from Android device

// Set Motor Control Pins
int rightMotor2 = 13;    // D7 - right Motor -
int rightMotor1 = 15;    // D8 - right Motor +
int leftMotor2 = 0;    // D3 - left Motor -
int leftMotor1 = 2;    // D4 - left Motor +
int eneLeftMotor = 12;  // D6 - enable Mortor Left
int eneRightMotor = 14; // D5 - enable Mortor Right

void setup()
{
  Serial.begin(115200);

  pinMode(eneLeftMotor, OUTPUT);
  pinMode(eneRightMotor, OUTPUT);
  pinMode(leftMotor1, OUTPUT);
  pinMode(leftMotor2, OUTPUT);
  pinMode(rightMotor1, OUTPUT);
  pinMode(rightMotor2, OUTPUT);

  digitalWrite(eneLeftMotor,LOW);
  digitalWrite(eneRightMotor,LOW);
  digitalWrite(leftMotor1,LOW);
```

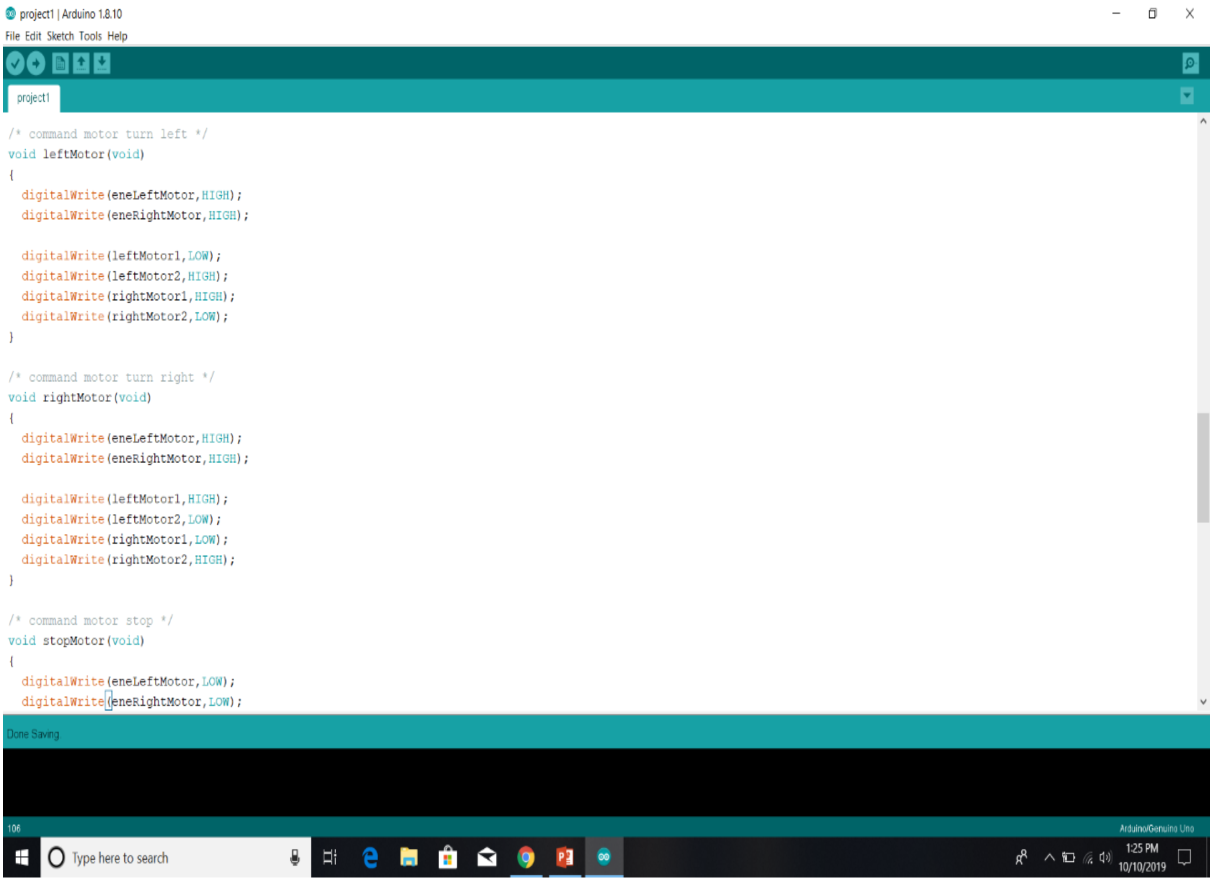


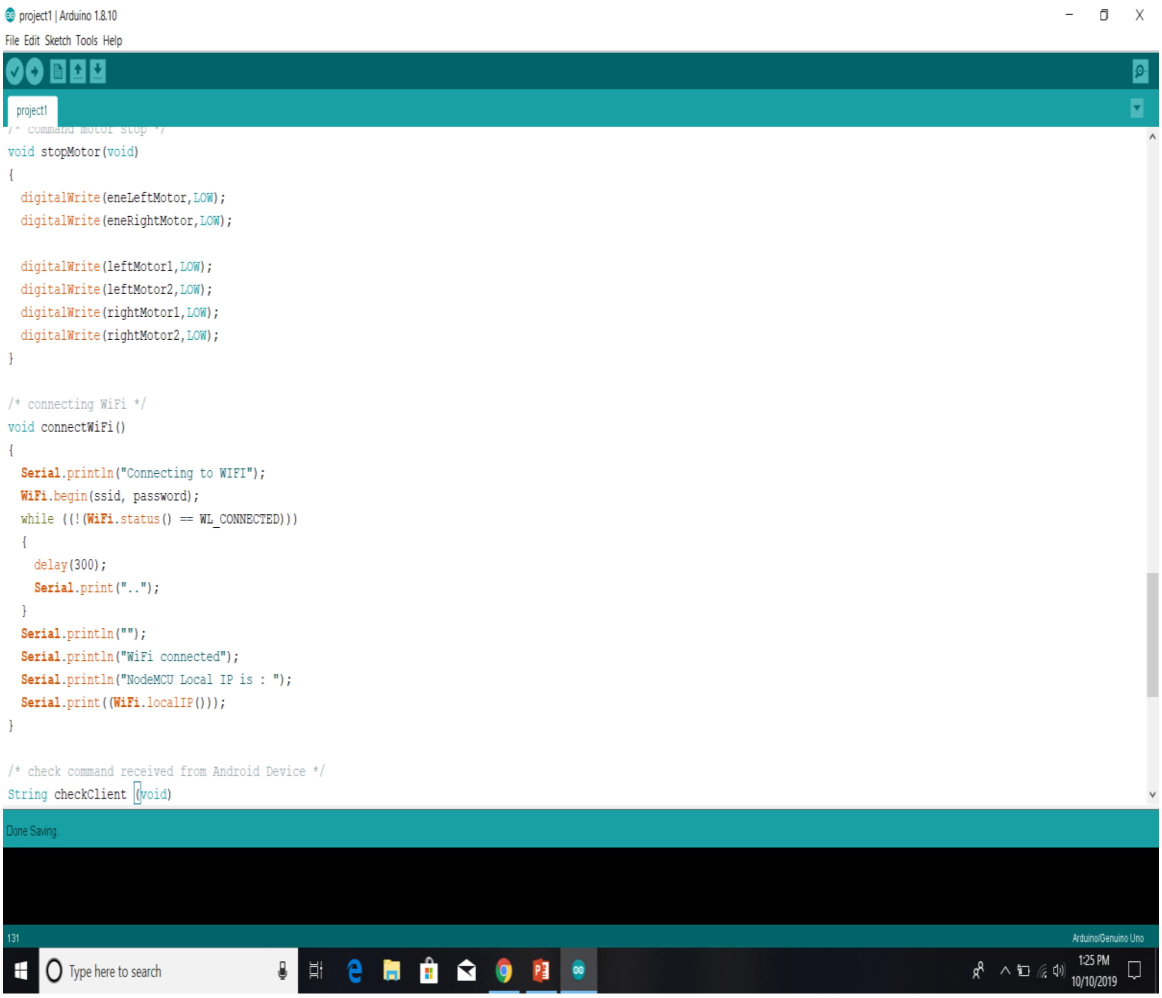
```
  digitalWrite(eneLeftMotor,LOW);
  digitalWrite(eneRightMotor,LOW);
  digitalWrite(leftMotor1,LOW);
  digitalWrite(leftMotor2,LOW);
  digitalWrite(rightMotor1,LOW);
  digitalWrite(rightMotor2,LOW);

  connectWiFi();
  server.begin();
}

void loop()
{
    client = server.available();
    if (!client) return;
    command = checkClient ();

         if (command == "forward" || command == "frente" || command == "a frente")  forwardMotor();
    else if (command == "reverse" || command == "reverso"|| command == "voltar") reverseMotor();
    else if (command == "left"    || command == "esquerda") leftMotor();
    else if (command == "right"   || command == "direita") rightMotor();
    else if (command == "stop"    || command == "pare" || command == "parar" || command == "parando")     stopMotor();

    sendBackEcho(command); // send command echo back to android device
    command = "";
}

/* command motor forward */
void forwardMotor(void)
```



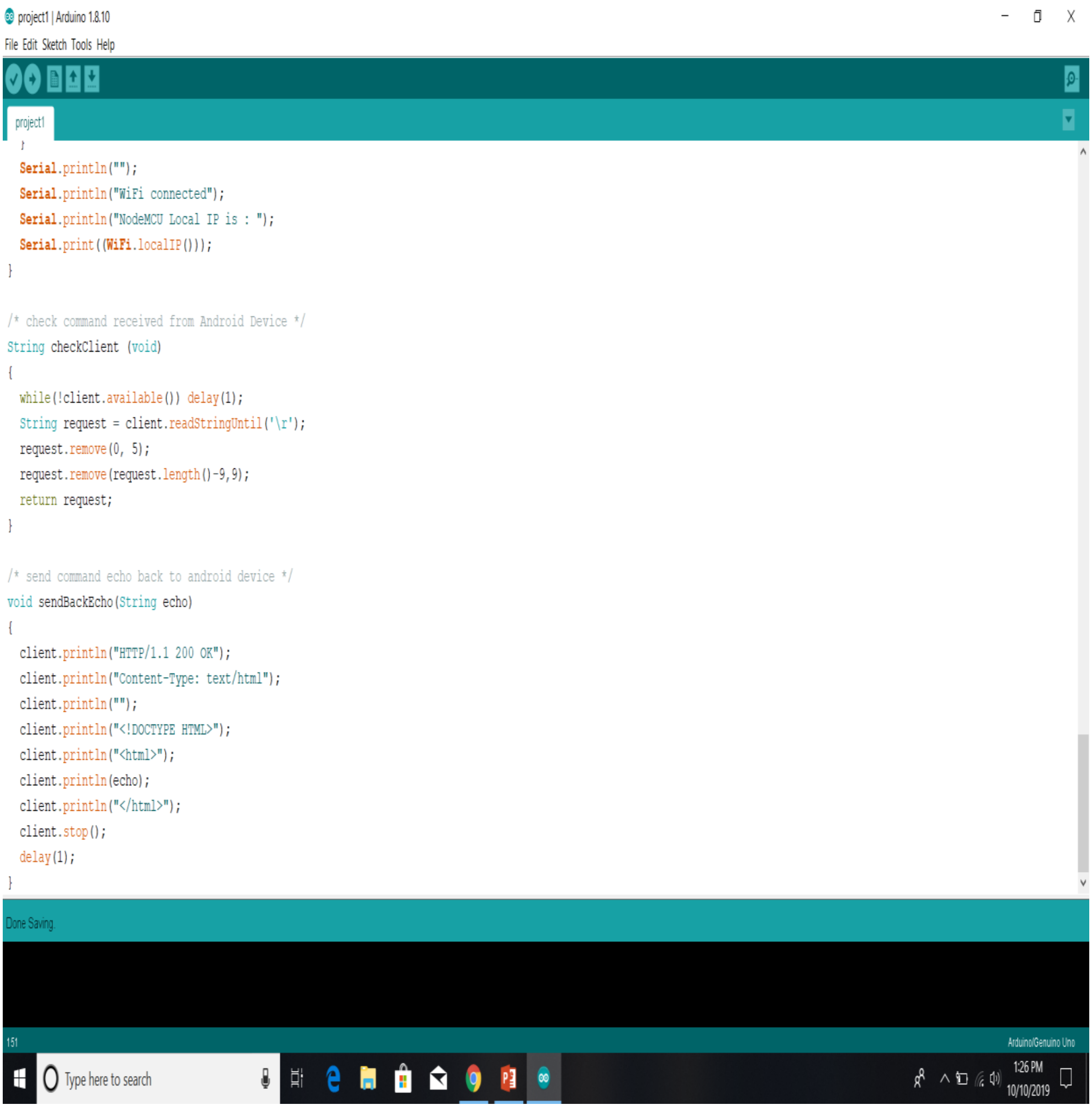

```
/* command motor forward */
void forwardMotor(void)
{
  digitalWrite(eneLeftMotor,HIGH);
  digitalWrite(eneRightMotor,HIGH);

  digitalWrite(leftMotor1,HIGH);
  digitalWrite(leftMotor2,LOW);
  digitalWrite(rightMotor1,HIGH);
  digitalWrite(rightMotor2,LOW);
}

/* command motor backward */
void reverseMotor(void)
{
  digitalWrite(eneLeftMotor,HIGH);
  digitalWrite(eneRightMotor,HIGH);

  digitalWrite(leftMotor1,LOW);
  digitalWrite(leftMotor2,HIGH);
  digitalWrite(rightMotor1,LOW);
  digitalWrite(rightMotor2,HIGH);
}

/* command motor turn left */
void leftMotor(void)
{
  digitalWrite(eneLeftMotor,HIGH);
  digitalWrite(eneRightMotor,HIGH);
```


`


project1 | Arduino 1.8.10
File Edit Sketch Tools Help
project1
/* command motor turn left */
void leftMotor(void)
{
digitalWrite(eneLeftMotor,HIGH);
digitalWrite(eneRightMotor,HIGH);
digitalWrite(leftMotor1,LOW);
digitalWrite(leftMotor2,HIGH);
digitalWrite(rightMotor1,HIGH);
digitalWrite(rightMotor2,LOW);
}
/* command motor turn right */
void rightMotor(void)
{
digitalWrite(eneLeftMotor,HIGH);
digitalWrite(eneRightMotor,HIGH);
digitalWrite(leftMotor1,HIGH);
digitalWrite(leftMotor2,LOW);
digitalWrite(rightMotor1,LOW);
digitalWrite(rightMotor2,HIGH);
}
/* command motor stop */
void stopMotor(void)
{
digitalWrite(eneLeftMotor,LOW);
digitalWrite(eneRightMotor,LOW);
Done Saving.
106
Arduino/Genuino Uno
Type here to search
1:25 PM
10/10/2019




```
/* command motor stop */
void stopMotor(void)
{
  digitalWrite(eneLeftMotor,LOW);
  digitalWrite(eneRightMotor,LOW);

  digitalWrite(leftMotor1,LOW);
  digitalWrite(leftMotor2,LOW);
  digitalWrite(rightMotor1,LOW);
  digitalWrite(rightMotor2,LOW);
}

/* connecting WiFi */
void connectWiFi()
{
  Serial.println("Connecting to WIFI");
  WiFi.begin(ssid, password);
  while ((!(WiFi.status() == WL_CONNECTED)))
  {
    delay(300);
    Serial.print("..");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("NodeMCU Local IP is : ");
  Serial.print((WiFi.localIP()));
}

/* check command received from Android Device */
String checkClient (void)
```


project1 | Arduino 1.8.10
File Edit Sketch Tools Help
project1
}
Serial.println("");
Serial.println("WiFi connected");
Serial.println("NodeMCU Local IP is : ");
Serial.print((WiFi.localIP()));
}
/* check command received from Android Device */
String checkClient (void)
{
while(!client.available()) delay(1);
String request = client.readStringUntil('\r');
request.remove(0, 5);
request.remove(request.length()-9,9);
return request;
}
/* send command echo back to android device */
void sendBackEcho(String echo)
{
client.println("HTTP/1.1 200 OK");
client.println("Content-Type: text/html");
client.println("");
client.println("<!DOCTYPE HTML>");
client.println("<html>");
client.println(echo);
client.println("</html>");
client.stop();
delay(1);
}
Done Saving.
151
Arduino/Genuino Uno
Type here to search
1:26 PM
10/10/2019

CHAPTER 7
CONCLUSION

As always, I hope this project can help others find their way in the exciting world of electronics, robotics and IoT!

It helped me in making my knowledge extremely useful in world of IoT

And Developed various skills in me.

`

CHAPTER 8

REFERENCES

1) Beginner’s Guide in Raspberry Pi – Rpi.org

2) Node MCU Datasheet by Handsontech

3) Embedded Systems by Manoj Gulati

4) Computer Networks - Frozen

`