

CHAPTER 1

COMPANY PROFILE

The internship named “Embedded System Development & IoT Applications” was conducted by Vitvara Technologies, Mangalore. This chapter briefs about the company profile and the trainer details. The main domain of the internship was based on embedded development. Embedded software is computer software, written to control machines or devices that are not typically thought of as computers, commonly known as embedded systems. It is typically specialized for the particular hardware that it runs on and has time and memory constraints. An Embedded System employs a combination of hardware & software to perform a specific function; is part of a larger system that may not be a “computer works in a reactive and time-constrained environment.

1.1 About the Company

Vitvara is ISO certified company established in 2011 approved by QCI. Vitvara was nurtured by a group of entrepreneurs with a sole mission of establishing a dedicated Research & Development Cell to fertilize the innovations of budding engineers. It is specialized for Software development, embedded product development and educational services.

Vitvara is specialized in creating company portfolio websites, Windows and Web applications, Android application and embedded products. They help their clients to boost the profits, enhance productivity, effective competition & facilitate growth by creating quality products [1]. The team always ensure that every solution that is developed at Vitvara is efficient in working and bring tangible results to their clients.

1.1.1 Products and Services

The Products and Services in a business plan outlines the product or service, why it is needed by the market, and how it will compete with other business selling the same or similar products and services.

Table 1.1 provides the products and services offered by Vitvara Technologies in various fields.

Table 1.1 Products and Services offered

Fields	Products and Services offered
Software development	Web and computer application Android application Database management application
Hardware development	R&D work on embedded products Manufacturer of sensors, development boards, educational brain boards
Educational Services	Research support for Engineering students Technology and Robotics training for young minds ATL- Atal Tinkering Labs services and support

1.2 About the Trainer

Mr. Manoj Kumar M is a professional embedded system and software application developer with eight years of experience developing a wide range of application projects. He has 4 years of experience in Robert Bosch on 'Embedded Design and Development'.

His core skills include Embedded product design and development, Android application development and Database application development. He also has efficient programming skills on HTML 5, CSS, Java, XML, PHP, Embedded C, Python, PLC, and MySQL database.

CHAPTER 2

TASK PERFORMED

This chapter briefs about the concepts learnt and covered during the internship. The academy was mainly focused on providing theoretical and practical knowledge on below mentioned concepts.

An embedded system is a combination of computer software and hardware which is either fixed in capability or programmable. An embedded system can be either an independent system, or it can be a part of a large system. It is mostly designed for a specific function or functions within a larger system. The detailed Introduction to Arduino IDE where IDE stands for Integrated Development Environment – An official software introduced by Arduino.cc, that is mainly used for writing, compiling and uploading the code in the Arduino Device. Almost all Arduino modules are compatible with this software that is an open source and is readily available to install.

The internet of things, or IoT, is a system of interrelated computing devices, mechanical and digital machines, objects, animals or people that are provided with Unique Identifiers (UIDs) and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction. A thing in the internet of things can be a person with a heart monitor implant, a farm animal with a biochip transponder, an automobile that has built-in sensors to alert the driver when tire pressure is low or any other natural or man-made object that can be assigned an Internet Protocol (IP) address and is able to transfer data over a network.

During this internship, we developed embedded system administration skills and to command over embedded system technical domains. Concepts such as API, Database management and PHP programming are also introduced to us.

2.1 Week 1: Getting Started with Arduino

Arduino is the name of the board powered with AtMega168 or AtMega32 microcontroller. The main advantage of Arduino is its simple programming language. It's designed to make new user to get a glimpse and boost their confidence of programming. Originally, Arduino was designed for school kids to get a taste of

programming and hardware design. It's suitable for making a demo model of any idea. This is done because of the presence of the 0.5KB of boot loader, that allows the program to be dumped into the circuit. The Arduino tool window contains a toolbar with a various button like new, open, verify, upload and serial monitor. And additionally, it comprises of a text editor (employed to write the code), a message space (displays the result) like showing the errors, the text console, that displays the output and a series of menus just like the file, tool menu and edit.

2.1.1 Introduction to Arduino IDE

- Arduino IDE is an open source software that is mainly used for writing and compiling the code into the Arduino Module.
- It is an official Arduino software, making code compilation too easy that even a common person with no prior technical knowledge can get their feet wet with the learning process.
- It is easily available for operating systems like MAC, Windows, Linux and runs on the Java Platform that comes with inbuilt functions and commands that play a vital role for debugging, editing and compiling the code in the environment.
- A range of Arduino modules available including Arduino Uno, Arduino Mega, Arduino Leonardo, Arduino Micro and many more.
- Each of them contains a microcontroller on the board that is actually programmed and accepts the information in the form of code.
- The main code, also known as a sketch, created on the IDE platform will ultimately generate a Hex File which is then transferred and uploaded in the controller on the board.
- The IDE environment mainly contains two basic parts: Editor and Compiler where former is used for writing the required code and later is used for compiling and uploading the code into the given Arduino Module.
- This environment supports both C and C++ languages.

The IDE environment is mainly distributed into three sections

1. Menu Bar
2. Text Editor.
3. Output pane.

The bar appearing on the top is called **Menu Bar** that comes with five different options as follow

File – You can open a new window for writing the code or open an existing one. Following table shows the number of further subdivisions the file option is categorized into.

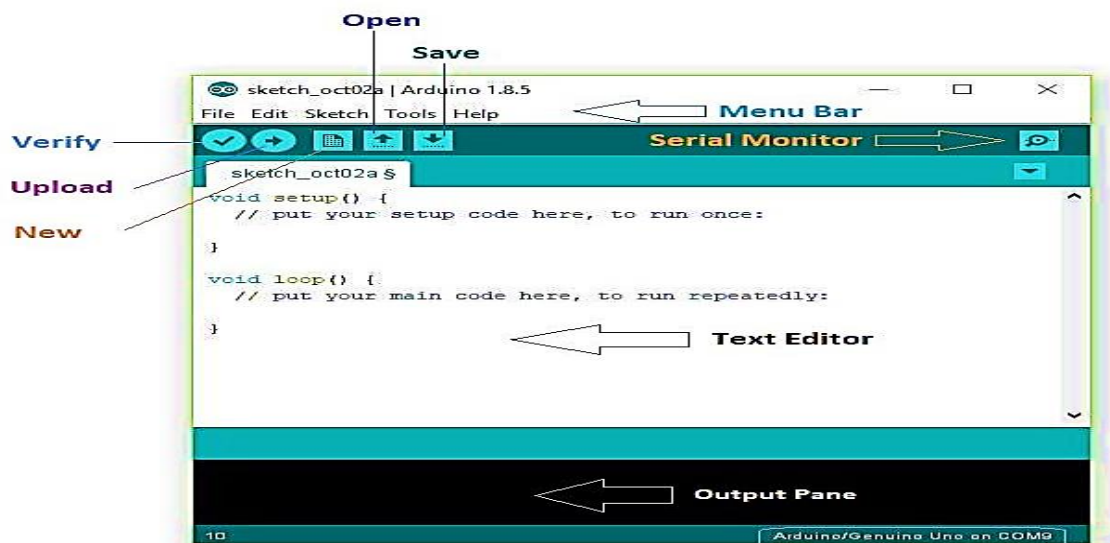


Figure 2.1 Introduction to Arduino IDE

File	
New	This is used to open new text editor window to write your code
Open	Used for opening the existing written code
Open Recent	The option reserved for opening recently closed program
Sketchbook	It stores the list of codes you have written for your project
Examples	Default examples already stored in the IDE software
Close	Used for closing the main screen window of recent tab. If two tabs are open, it will ask you again as you aim to close the second tab
Save	It is used for saving the recent program
Save as	It will allow you to save the recent program in your desired folder
Page setup	Page setup is used for modifying the page with portrait and landscape options. Some default page options are already given from which you can select the page you intend to work on
Print	It is used for printing purpose and will send the command to the printer
Preferences	It is page with number of preferences you aim to setup for your text editor page
Quit	It will quit the whole software all at once

Figure 2.2 Content of File

As you go to the preference section and check the compilation section, the Output Pane will show the code compilation as you click the upload button.

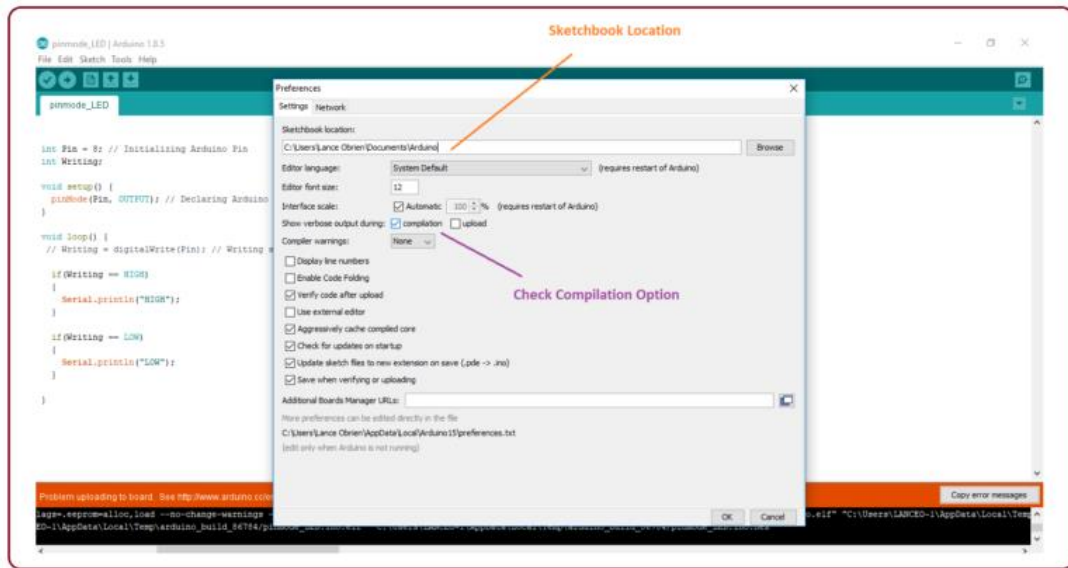


Figure 2.3 The preference section

And at the end of compilation, it will show you the hex file it has generated for the recent sketch that will send to the Arduino Board for the specific task you aim to achieve.



Figure 2.4 The Hex file generated window

- **Edit** – Used for copying and pasting the code with further modification for font
- **Sketch** – For compiling and programming
- **Tools** – Mainly used for testing projects. The Programmer section in this panel is used for burning a bootloader to the new microcontroller.
- **Help** – In case you are feeling skeptical about software, complete help is available from getting started to troubleshooting.

The **Six Buttons** appearing under the Menu tab are connected with the running program as follow.

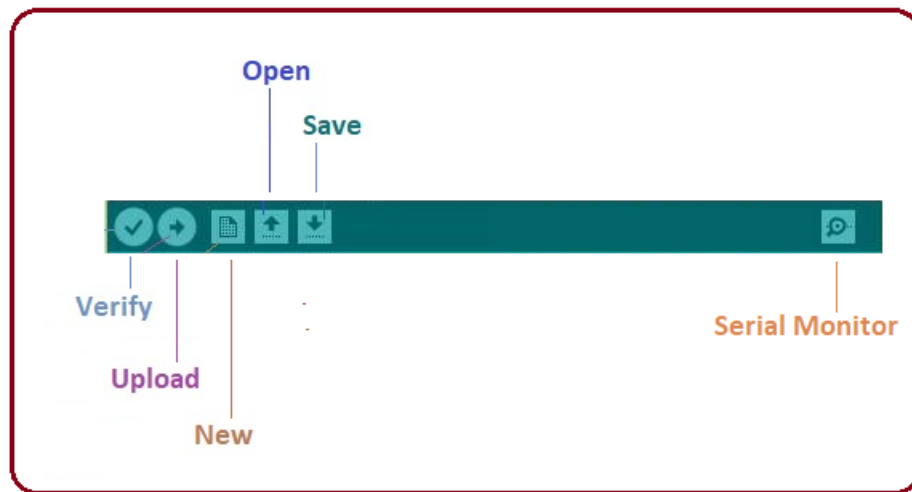


Figure 2.5 The Menu tab

The Six Buttons appearing under the Menu tab are connected with the running program functions as given below.

- The check mark appearing in the circular button is used to verify the code. Click this once you have written your code.
- The arrow key will upload and transfer the required code to the Arduino board.
- The dotted paper is used for creating a new file.
- The upward arrow is reserved for opening an existing Arduino project.
- The downward arrow is used to save the current running code.
- The button appearing on the top right corner is a Serial Monitor – A separate pop-up window that acts as an independent terminal and plays a vital role for sending and receiving the Serial Data. We can also go to the Tools panel and select Serial Monitor, or pressing Ctrl+Shift+M all at once will open it instantly. The Serial Monitor will actually help to debug the written Sketches where you can get a hold of how your program is operating. Arduino Module should be connected to the computer by USB cable in order to activate the Serial Monitor.

In order to upload the sketch, we need to select the relevant board you are using and the ports for that operating system. We need to select the baud rate of the Arduino Board. For Arduino Uno baud rate is 9600. As we write the code and click the serial monitor the output will be shown as in Figure 2.4.

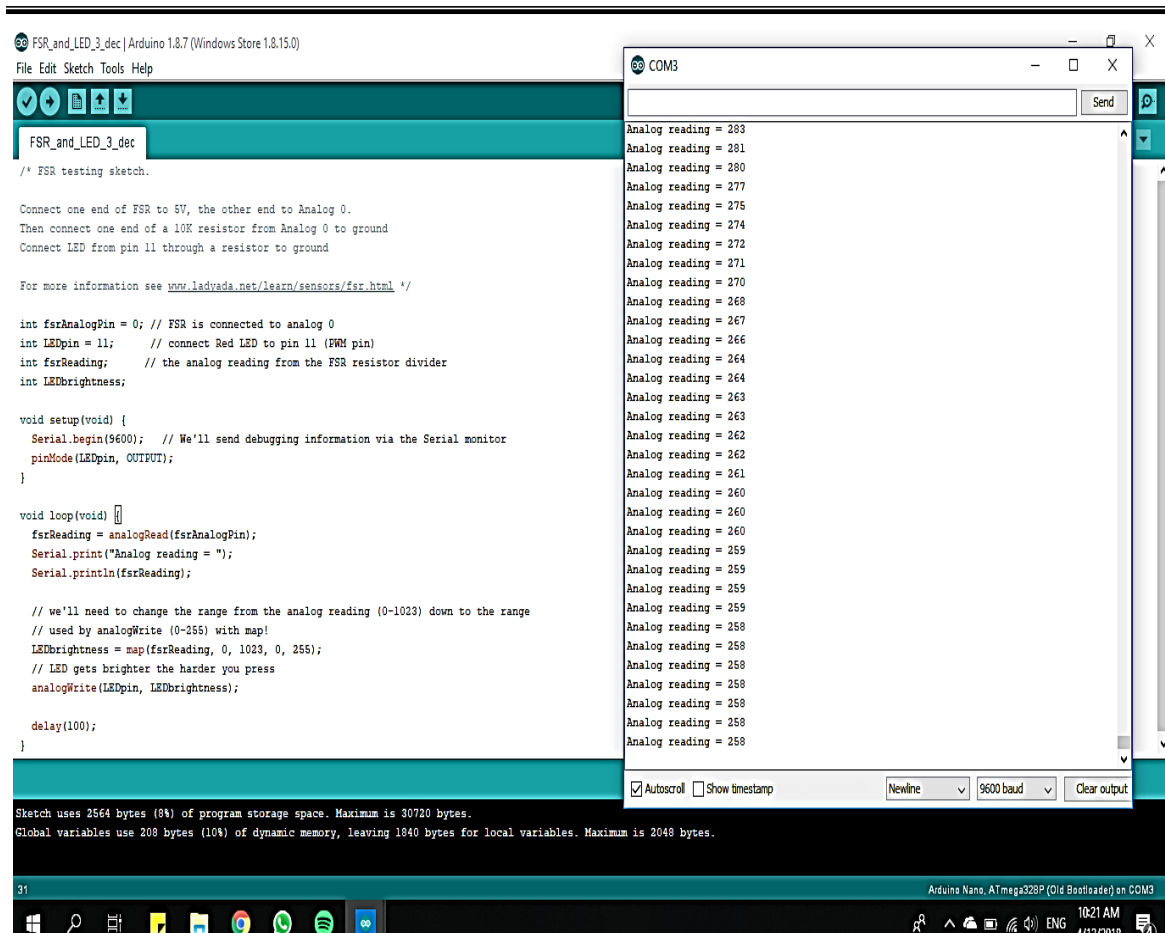


Figure 2.6 Arduino Serial Monitor Window

The main screen below the Menu bard is known as a simple text editor used for writing the required code.



Figure 2.7 The Text Editor

The bottom of the main screen is described as an Output Pane that mainly highlights the compilation status of the running code: the memory used by the code, and errors occurred

in the program. You need to fix those errors before you intend to upload the hex file into your Arduino Module.

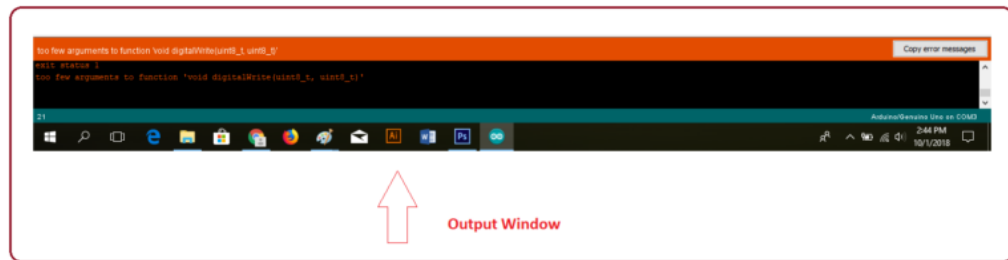


Figure 2.8 The Output window

More or less, Arduino C language works similar to the regular C language used for any embedded system microcontroller, however, there are some dedicated libraries used for calling and executing specific functions on the board.

2.1.2 Libraries

Libraries are very useful for adding the extra functionality into the Arduino Module. There is a list of libraries you can add by clicking the Sketch button in the menu bar and going to Include Library.

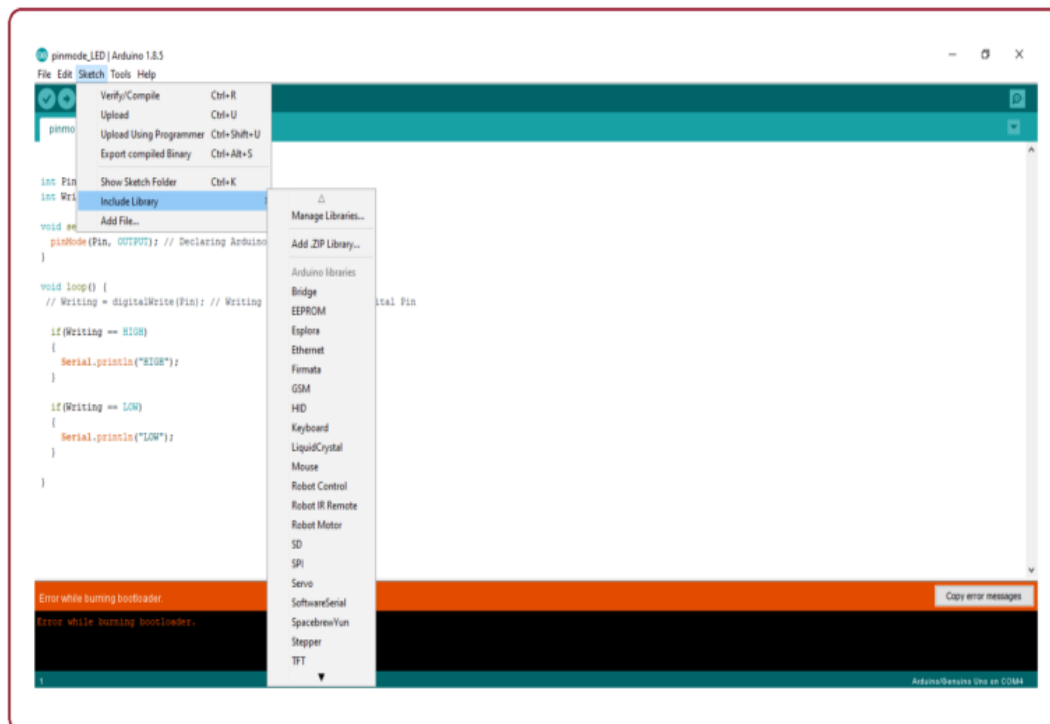


Figure 2.9 The list of libraries

As you click the Include Library and Add the respective library it will on the top of the sketch with a #include sign. Suppose, I Include the EEPROM library, it will appear on the text editor as

```
#include <EEPROM.h>.
```

2.1.3 Making pins input or output

The digitalWrite and digital Write commands are used for addressing and making the Arduino pins as an input and output respectively. These commands are text sensitive i.e. you need to write them down the exact way they are given like digitalWrite starting with small “d” and write with capital “W”. Writing it down with Digitalwrite or digitalwrite won’t be calling or addressing any function.

2.1.4 To select the board

In order to upload the sketch, you need to select the relevant board you are using and the ports for that operating system. As you click the Tools on the Menu, it will open like the figure below.

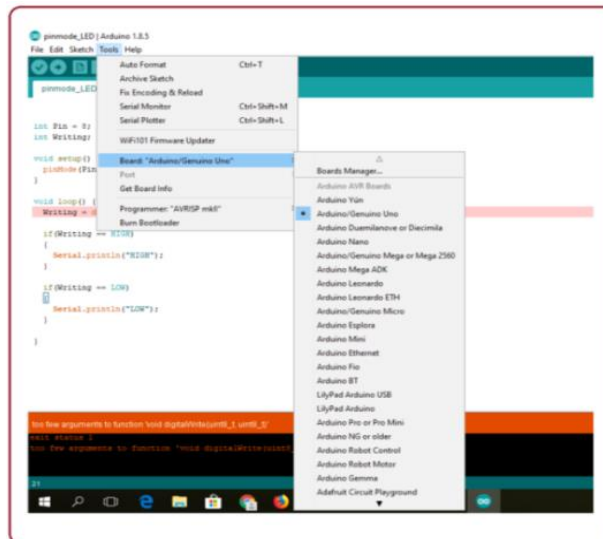


Figure 2.10 The board menu

- Select the board you aim to work on. Similarly, COM1, COM2, COM4, COM5, COM7 or higher are reserved for the serial and USB board. You can look for the USB serial device in the ports section of the Windows Device Manager.

Following Figure 2.11 shows COM4 is connected to Arduino and indicates COM4 port is used for the project.

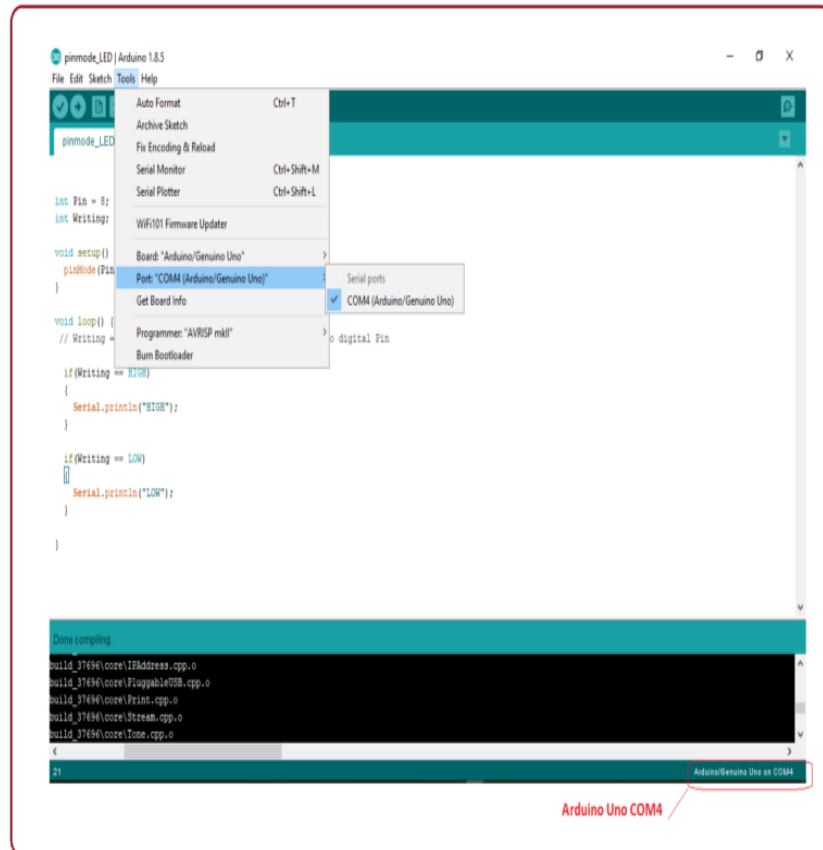


Figure 2.11 The COM4

- After correct selection of both Board and Serial Port, click the verify and then upload button appearing in the upper left corner of the six button section or you can go to the Sketch section and press verify/compile and then upload.
- The sketch is written in the text editor and is then saved with the file extension .ino.

It is important to note that the recent Arduino Modules will reset automatically as you compile and press the upload button the IDE software, however, older version may require the physical reset on the board.

- Once you upload the code, TX and RX LEDs will blink on the board, indicating the desired program is running successfully.

- The amazing thing about this software is that no prior arrangement or bulk of mess is required to install this software, you will be writing your first program within 2 minutes after the installation of the IDE environment.

2.1.5 Bootloader

As you go to the Tools section, you will find a bootloader at the end. It is very helpful to burn the code directly into the controller, setting you free from buying the external burner to burn the required code.

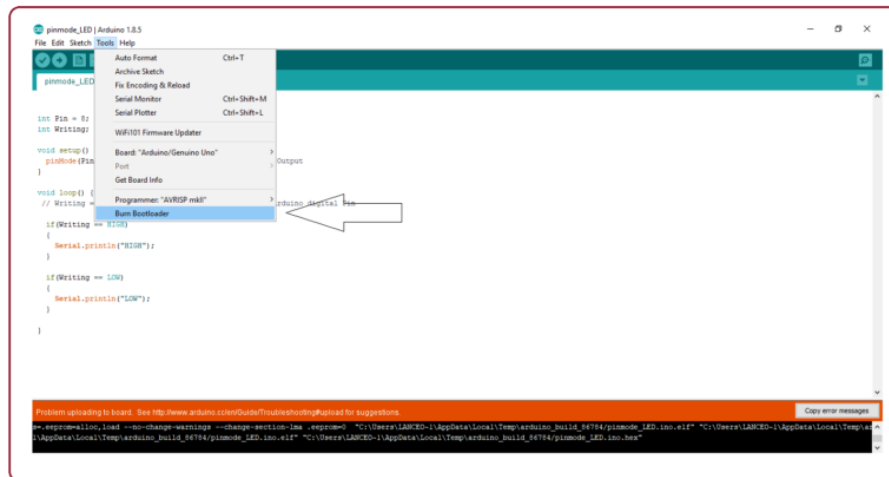


Figure 2.12 The tool bar

When we buy the new Arduino Module, the bootloader is already installed inside the controller. However, if you intend to buy a controller and put in the Arduino module, you need to burn the bootloader again inside the controller by going to the Tools section and selecting the burn bootloader.

2.1.6 Introduction to Arduino Uno

Arduino is a single-board microcontroller meant to make the application more accessible which are interactive objects and its surroundings. The hardware features with an open-source hardware board designed around an 8-bit Atmel AVR microcontroller or a 32-bit Atmel ARM. Current models consists a USB interface, 6 analog input pins and 14 digital I/O pins as in Figure 2.13 that allows the user to attach various extension boards.

The Arduino Uno board is a microcontroller based on the ATmega328. Out of 14 digital input/output pins available, 6 can be used as PWM outputs, a 16 MHz ceramic resonator, an ICSP header, a USB connection, 6 analog inputs, a power jack and a reset button. This

contains all the required support needed for microcontroller. In order to get started, they are simply connected to a computer with a USB cable or with an AC-to-DC adapter or battery.

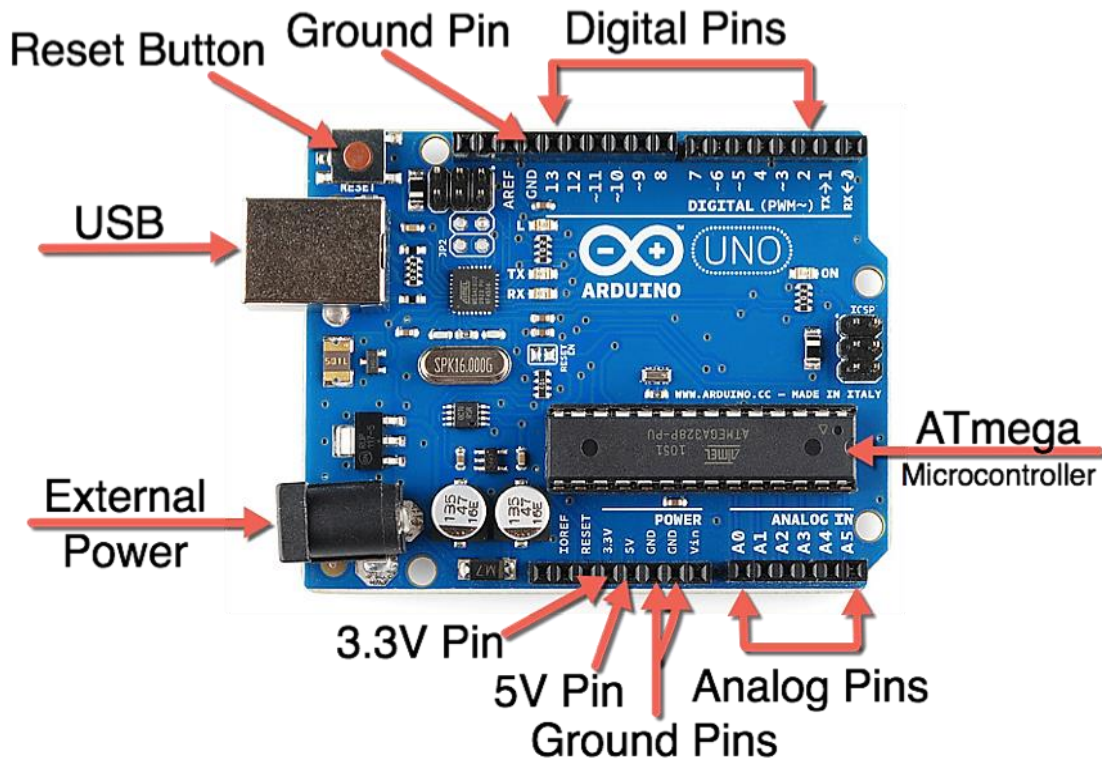


Figure 2.13 Arduino Uno Board

Arduino Uno Board varies from all other boards and they will not use the FTDI USB-to-serial driver chip in them. It is featured by the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter.

There are various types of Arduino boards. The most official versions available are the Arduino Uno R3 and the Arduino Nano V3. Both of these run a 16MHz Atmel ATmega328P 8-bit microcontroller with 32KB of flash RAM, 14 digital I/O and six analogue I/O and the 32KB will not sound like as if running Windows. Arduino projects can be stand-alone or they can communicate with software on running on a computer. For e.g. Flash, Processing, Max/MSP.

2.2 Week 2: Introduction to Embedded Systems

An embedded system is one kind of a computer system mainly designed to perform several tasks like to access, process, store and also control the data in various

electronics-based systems. Embedded systems are a combination of hardware and software where software is usually known as firmware that is embedded into the hardware. One of its most important characteristics of these systems is, it gives the o/p within the time limits. Embedded systems support to make the work more perfect and convenient. So, we frequently use embedded systems in simple and complex devices too. The applications of embedded systems mainly involve in our real life for several devices like microwave, calculators, TV remote control, home security and neighbourhood traffic control systems, etc.

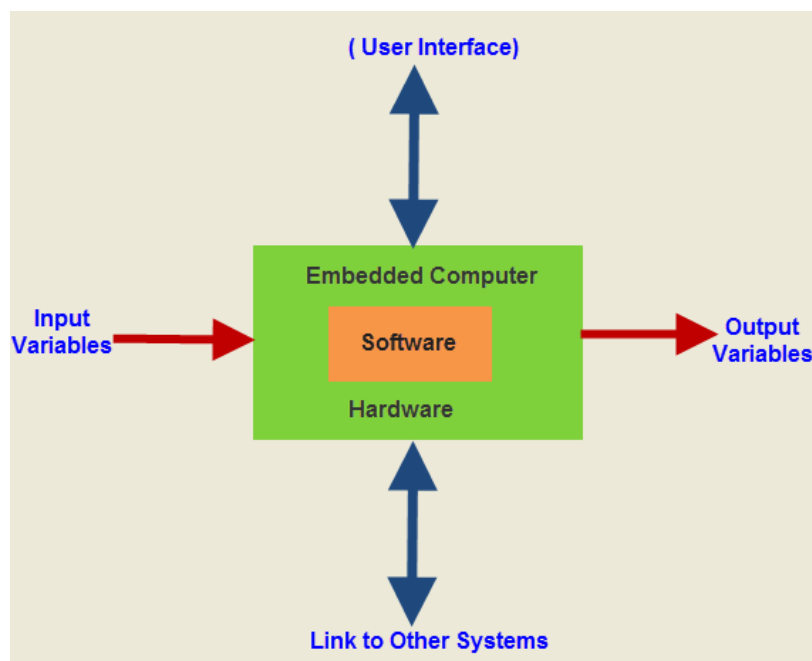


Figure 2.14 Block diagram of Embedded System

Embedded system software is written in a high-level language, and then compiled to achieve a specific function within a non-volatile memory in the hardware. Embedded system software is designed to keep in view of three limits. They are availability of system memory and processor speed. When the system runs endlessly, there is a need to limit the power dissipation for events like run, stop and wake up. The Embedded system hardware includes elements like user interface, Input/output interfaces, display and memory, etc. Generally, an embedded system comprises power supply, processor, memory, timers, serial communication ports and system application specific circuits.

2.2.1 Introduction to Various Types of Microcontrollers

A microcontroller (μC) is a solitary chip microcomputer fabricated from VLSI fabrication. A micro controller is also known as embedded controller. Today various types of microcontrollers are available in market with different word lengths such as 4bit, 8bit, 64bit and 128bit microcontrollers. Microcontroller is a compressed microcomputer manufactured to control the functions of embedded systems in office machines, robots, home appliances, motor vehicles, and a number of other gadgets.

The basic structure of a microcontroller comprises of CPU, memory, I/O ports, serial ports, timers, ADC, DAC, interpret control and a special functioning block. These are divided into categories according to their memory, architecture, bits and instruction sets. Table 2.1 provides comparison of three types of microcontrollers:

Table 2.1 Comparison of Various Microcontrollers

Parameter	Intel 8051	ATmega 328	ARM 7
Bits	8bit	8bit	32bit
RAM	128bytes	2KB	8KB-40KB
Flash memory	NA	32KB	32KB-512KB
EEPROM	NA	1KB	NA
ADC	NA	6	12
DAC	NA	NA	NA
PWM	NA	6	6
Protocol	UART	I2C, UART, SPI	UART, I2C, SPI, USB

There are a thousand different type of microcontrollers, each of them with a unique feature or competitive advantage from form factor, to package size, to the capacity

of the RAM and ROM that makes them fit for certain applications and unfit for certain applications.

MICROPROCESSOR V/S MICROCONTROLLERS

Products using microprocessors generally fall into two categories. The first category uses high-performance microprocessors such as the Pentium in application where system performance is critical. However, the second category of applications in which, performance is secondary; issues of power, space and rapid development are more critical than raw processing power. The microprocessors for this category are often called a microcontroller.

Microprocessors	Microcontrollers
Microprocessors contain no RAM or ROM.	Microcontrollers have an internal RAM and a ROM.
Microprocessors do not have any I/O ports.	Microcontrollers have I/O ports.
Advantage of versatility.	Advantages of less power consumption.
Expensive.	Cheaper in comparison.
With the addition of external RAM and ROM, the system is bulkier	Occupies less space.

- The prime use of microprocessor is to read data, perform extensive calculations on that data and store them in the mass storage device or display it. The prime functions of microcontroller is to read data, perform limited calculations on it, control its environment based on these data.
- Thus the microprocessor is said to be general-purpose digital computers whereas the microcontroller are intend to be special purpose digital controller.
- Microprocessor need many opcodes for moving data from the external memory to the CPU, microcontroller may require just one or two, also microprocessor may have one or two types of bit handling instructions whereas microcontrollers have many.

- Lastly, the microprocessor design accomplishes the goal of flexibility in the hardware configuration by enabling large amounts of memory and I/O that could be connected to the address and data pins on the IC package. The microcontroller design uses much more limited set of single and double byte instructions to move code and data from internal.

2.2.2 Execution of Miniprojects Assigned

Mini Project helps us to explore and strengthen the understanding of fundamentals through the practical application of theoretical concepts. It boosts our skills and widens our horizon of thinking. It acts like a beginner's guide to do larger projects later in the career. Here are some mini projects assigned to us during our internship period.

1. Door open and Door close :

Here, the main objective of this mini-project is to open or close the door according to the user input. The user gives the input using 2 push buttons, one for opening the door and another for closing the door. When the open push button is pressed, the door must be opened and when the close push button is pressed, the door must be closed. Here, DC motor is used as a door.

```
#define close_swt 2
```

```
#define open_swt 3
```

```
boolean door_status=0;
```

```
void setup() {
```

```
  Serial.begin(9600);
```

```
  pinMode(close_swt,INPUT_PULLUP);
```

```
  pinMode(open_swt,INPUT_PULLUP);
```

```
  pinMode(4,OUTPUT);
```

```
  pinMode(5,OUTPUT);
```

```
}

void loop() {

    if(digitalRead(close_swt)==0)

    {

        if(door_status==0)

        {

            Serial.println("Openeing door");

            door_open();

            door_status=1;

        }

    }

    else if(digitalRead(open_swt)==0)

    {

        if(door_status == 1)

        {

            Serial.println("Closeing door");

            door_close();

            door_status=0;

        }

    }

}

void door_open()
```

```
{  
  
    digitalWrite(4,0);  
  
    digitalWrite(5,1);  
  
    delay(500);  
  
    digitalWrite(4,0);  
  
    digitalWrite(5,0);  
  
}  
  
void door_close()  
  
{  
  
    digitalWrite(4,1);  
  
    digitalWrite(5,0);  
  
    delay(500);  
  
    digitalWrite(4,0);  
  
    digitalWrite(5,0);  
  
}
```

2. LED ON/OFF :

This aims at switching ON or OFF the LED based on the user input. The user gives the input from push button. When button is pressed, LED goes ON and when pressed again, the LED goes OFF.

```
void setup()  
  
{  
  
    pinMode(2,INPUT_PULLUP);  
  
    pinMode(3,OUTPUT);
```

```

}

void loop()

{
  if(digitalRead(2)==0)

  {
    digitalWrite(3,1);
  }

  else

  {
    digitalWrite(3,0);
  }
}

```

3. LED toggle :

This miniproject aims at toggling the LED based on the input given by the user. The input is given with the help of push buttons. When the push button is pressed, the LED goes ON and when the same push button is released, the LED goes OFF.

```

boolean led_status=0;

void setup()

{
  pinMode(2,INPUT_PULLUP);
  pinMode(3,OUTPUT);
}

void loop()

{
  if(digitalRead(2)==0)

  {
    if(led_status==0)

```

```

{
    digitalWrite(3,1);
    led_status=1;
}
else
{
    digitalWrite(3,0);
    led_status=0;
}
while(digitalRead(2)==0);
}
}

```

4. Increasing LED brightness :

In this miniproject, the brightness of LED is controlled based on the inputs provided by the user. Push button is used to provide the input. When the push button is pressed, the brightness of LED increases by 20%. Similarly when pressed again, the brightness increases by 50%, then 75% and finally 100%.

```

int led_status=0;

void setup()
{

    pinMode(2,INPUT_PULLUP);
    pinMode(3,OUTPUT);

}

void loop()
{

```

```

if(digitalRead(2)==0)
{
    if(led_status==0)
    {
        analogWrite(3,0);

    }
    else if(led_status==1)
    {
        analogWrite(3,50);

    }
    else if(led_status==2)
    {
        analogWrite(3,100);

    }
    else if(led_status==3)
    {
        analogWrite(3,150);

    }
    else if(led_status==4)
    {
        analogWrite(3,200);

    }
    else if(led_status==5)

```

```
{  
    analogWrite(3,255);  
  
}  
  
while(digitalRead(2)==0);  
  
}
```

2.3 Week 3: Introduction to IoT Applications

Basically, IoT is a network in which all physical objects are connected to the internet through network devices or routers and exchange data. IoT allows objects to be controlled remotely across existing network infrastructure. IoT is a very good and intelligent technique which reduces human effort as well as easy access to physical devices. This technique also has autonomous control feature by which any device can control without any human interaction. Figure 2.15 shows the connectivity of various devices of different fields with Internet and exchange data between them. This represent the connectivity of world through various existing technologies.

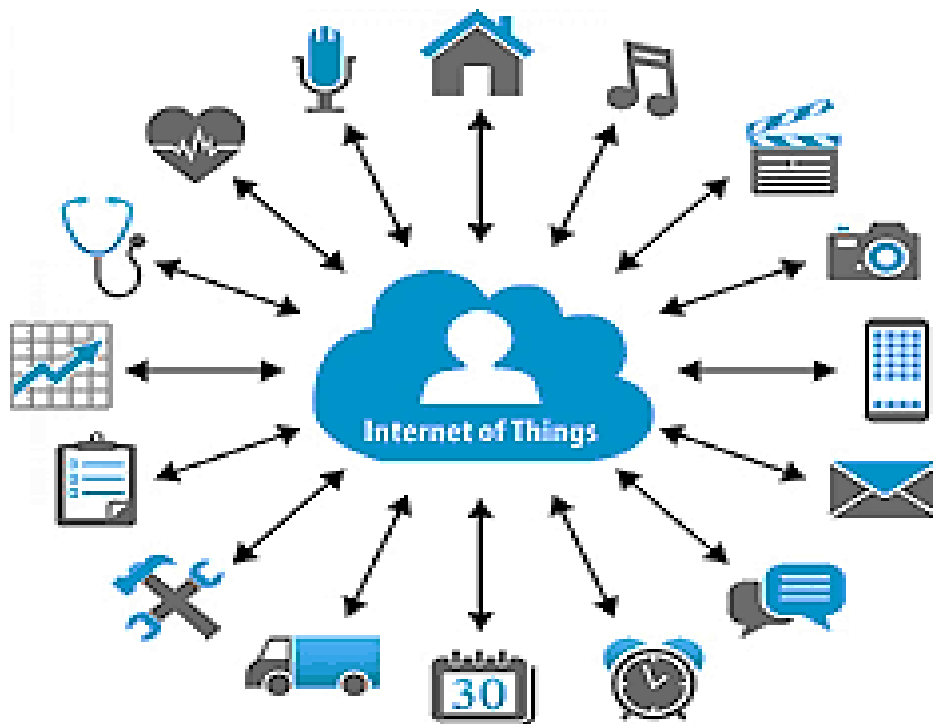


Figure 2.15 IoT Applications

Things in the IoT sense, is the mixture of hardware, software, data, and services. Things can refer to a wide variety of devices such as DNA analysis devices for environmental monitoring, electric clamps in coastal waters, Arduino chips in home automation and many other. These devices gather useful data with the help of various existing technologies and share that data between other devices. Examples include Home Automation System which uses Wi-Fi or Bluetooth for exchange data between various devices of home. When we look at today's state of technologies, we get a clear indication of how IoT will be implemented on a global level in near future. Use of the internet is increasing day-by-day. Commute and connectivity became easier in the present scenario. In near future, the number of internets connected devices would increase exponentially.

2.3.1 API and Database in IoT

API is an abbreviation for Application Programming Interface which is a collection of communication protocols and subroutines used by various programs to communicate between them. A programmer can make use of various API tools to make its program easier and simpler. Also, an API facilitates the programmers with an efficient way to develop their software programs. Thus, in simpler terms, an API helps two programs or applications to communicate with each other by providing them with necessary tools and functions. It takes the request from the user and sends it to the service provider and then again sends the result generated from the service provider to the desired user.

A database is an organized collection of data, generally stored and accessed electronically from a computer system. Where databases are more complex, they are often developed using formal design and modelling techniques. Databases are structured to facilitate the storage, retrieval, modification, and deletion of data in conjunction with various data-processing operations. IoT data management enables businesses to discover usage patterns. It also challenges assumptions made during design and development phases, identifying weaknesses in connected devices. In other words, it helps to create the best-connected products possible.

APIs are tightly linked with IoT because they allow you to securely expose connected devices to customers, go-to-market channels, and other applications in your IT infrastructure. Because APIs connect important things like cars, medical devices, energy grids, and thermostats to your ecosystem, it's critical to deploy API management that is

flexible, scalable and secure. APIs allow developers to build context-based applications that can interact with the physical world instead of purely through UI. To truly achieve IoT we need a REST API for every device. REST allows data to flow over internet protocols and to delegate and manage authorization.

NodeMCU is based on the ESP8266-12E Wi-Fi System-On-Chip, loaded with an open-source, Lua-based firmware. It is perfect for IoT applications, and other situations where wireless connectivity is required. This chip has a great deal in common with the Arduino, they are both microcontrollers equipped prototyping boards which can be programmed using the Arduino IDE. Figure 2.12 shows the NodeMCU module.

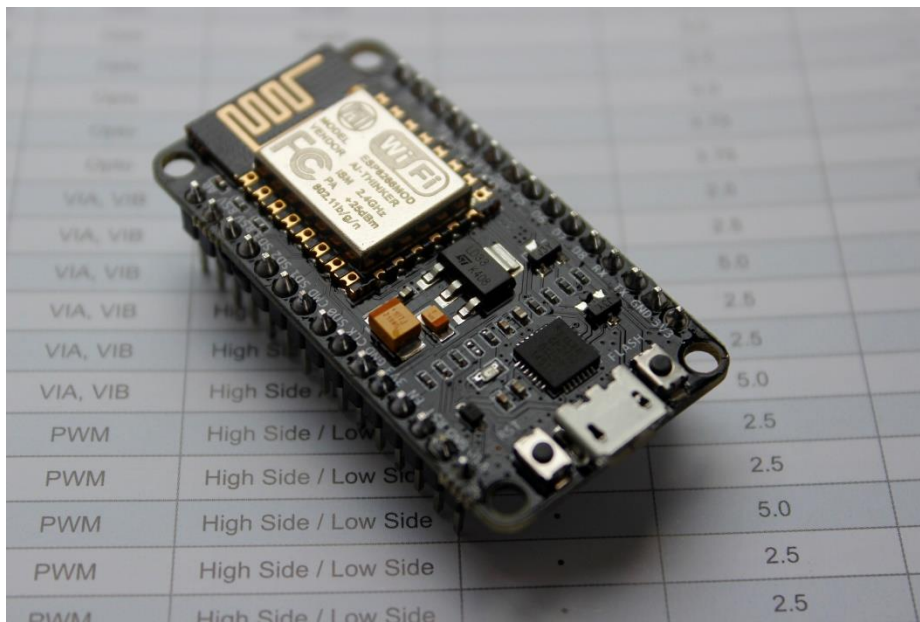


Figure 2.16 NodeMCU

Both the firmware and prototyping board designs are open source.

The firmware uses the Lua scripting language. The firmware is based on the eLua project, and built on the Espressif Non-OS SDK for ESP8266. Due to resource constraints, users need to select the modules relevant for their project and build a firmware tailored to their needs. Support for the 32-bit ESP32 has also been implemented.

The prototyping hardware typically used is a circuit board functioning as a dual in-line package (DIP) which integrates a USB controller with a smaller surface-mounted board containing the MCU and antenna. The choice of the DIP format allows for easy prototyping on breadboards. The design was initially was based on the ESP-12 module of

the ESP8266, which is a Wi-Fi SoC integrated with a Tensilica Xtensa LX106 core, widely used in IoT applications.

The steps to setup Arduino IDE for NodeMCU ESP8266 are mentioned here. Open the Arduino window, open the file and click on the preferences as shown in Figure 2.17.

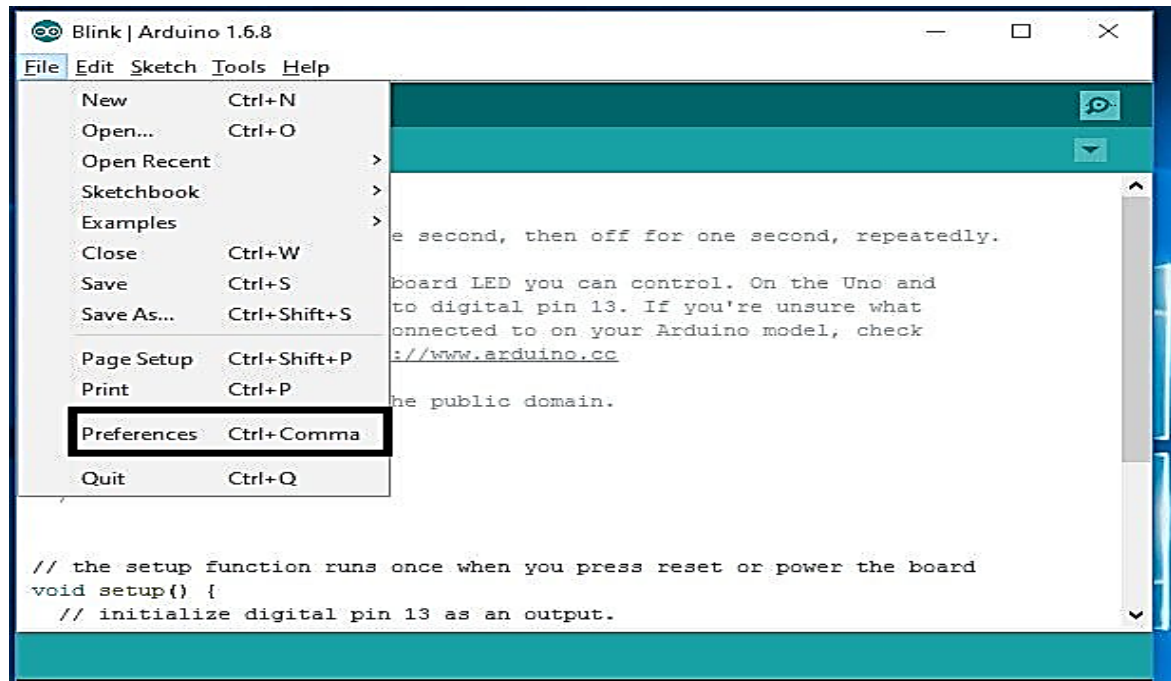


Figure 2.17 Preferences

In the Additional Boards Manager enter the URL as in Figure 2.10 and click on OK.

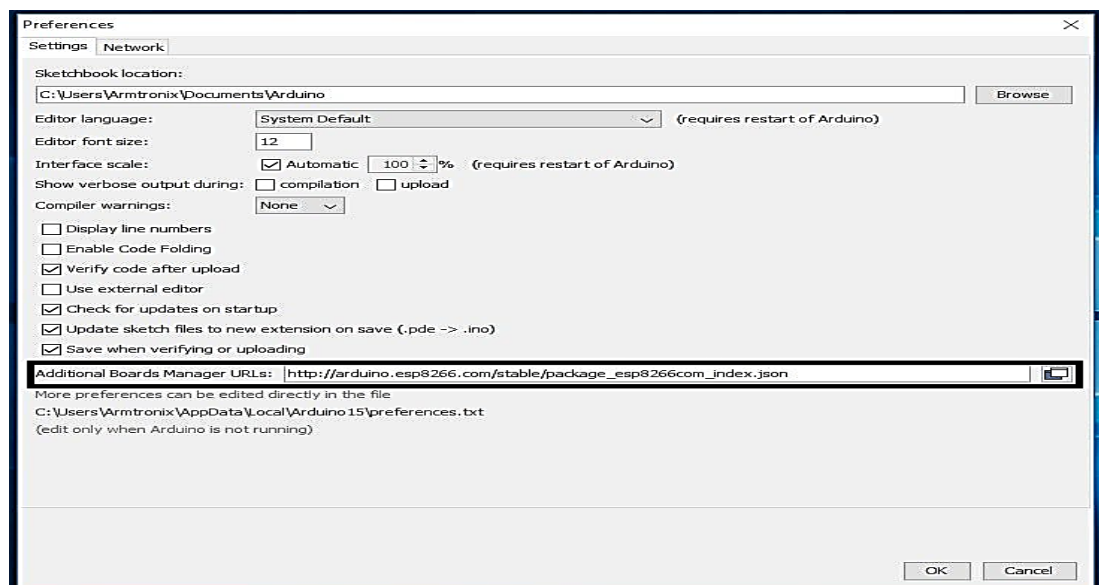


Figure 2.18 Adding ESP8266 Board Manager

Now open the tools in that select Board: “Arduino/Genuino Uno” and click on the Boards Manager. The Boards Manager window opens, scroll the window page to bottom till module with the name ESP8266 is seen. Once we get it, select that module and select version and click on the Install button. When it is installed, it shows Installed in the module as shown in the Figure 2.15 and then close the window.

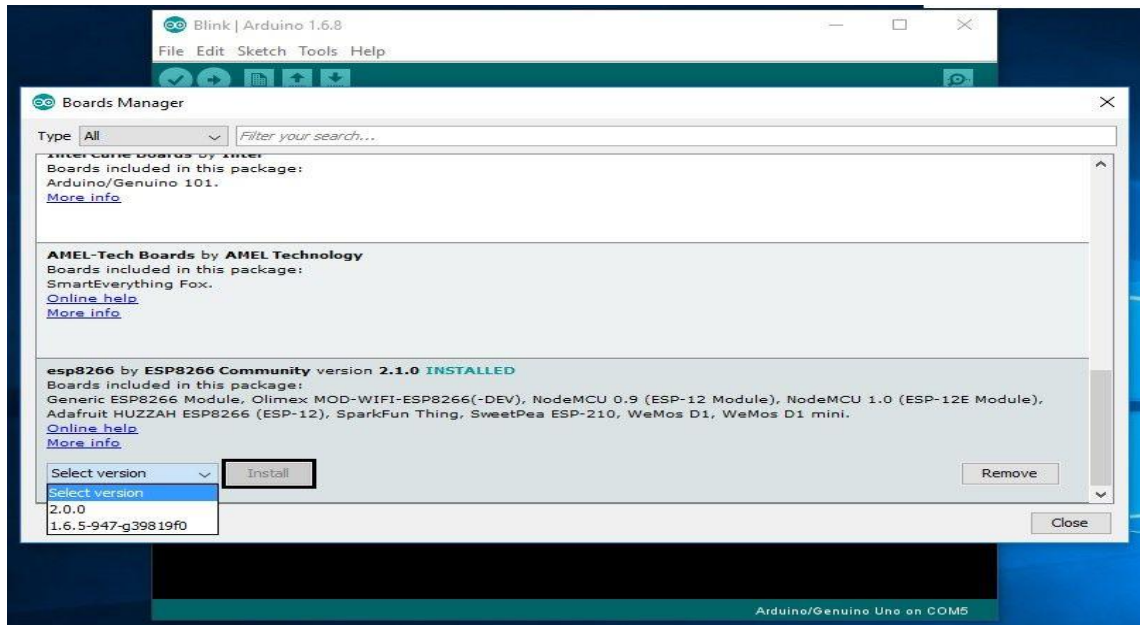


Figure 2.19 ESP8266 Board Package

To run the ESP8266 with Arduino we have to select the Board: “Arduino/Genuino Uno” and then change it to NodeMCU 1.0 (ESP-12E Module). Connect ESP8266 module to the computer through USB cable. When module is connected to the USB, COM port has to be detected.

2.4: Week 4: Working on the Assigned Project

The Project was assigned in the last week of the internship. The project was on IoT based RFID attendance system. Attendance monitoring system using RFID technology and with the application of Internet of Things (IoT) and cloud technology the attendance of the students can be monitored in real time, so that all parties who need information such as lecturers and the administration can immediately find out if there are students who skip classroom from the data stored in the database and it can immediately be prevented.

COMPONENTS REQUIRED

- Arduino uno board
- Node MCU
- RFID cards
- RFID card reader
- Connecting wires

2.4.1 RFID card reader

Radio-frequency identification (RFID) uses electromagnetic fields to automatically identify and track tags attached to objects. An RFID tag consists of a tiny radio transponder; a radio receiver and transmitter. When triggered by an electromagnetic interrogation pulse from a nearby RFID reader device, the tag transmits digital data, usually an identifying inventory number, back to the reader. This number can be used to inventory goods. RFID tags are made out of three pieces: a microchip (an integrated circuit which stores and processes information and modulates and demodulates radio-frequency (RF) signals), an antenna for receiving and transmitting the signal and a substrate. The tag information is stored in a non-volatile memory. The RFID tag includes either fixed or programmable logic for processing the transmission and sensor data, respectively.

RFID tags can be either passive, active or battery-assisted passive. An active tag has an on-board battery and periodically transmits its ID signal. A battery-assisted passive has a small battery on board and is activated when in the presence of an RFID reader. A passive tag is cheaper and smaller because it has no battery; instead, the tag uses the radio energy transmitted by the reader. However, to operate a passive tag, it must be illuminated with a power level roughly a thousand times stronger than an active tag for signal transmission.



Figure 2.20 RDM6300 RFID Card Reader

Specifications is as follows:

- Supports external antenna
- Maximum distance: 50 mm
- Decoding time: <100 ms
- UART interface
- Bi-color LED

2.4.2 RFID card

RFID is useful for sensing and identifying tagged people and objects for access control, automation, and a whole range of different applications. This basic RFID tag works in the 125kHz RF range and comes with a unique 32-bit ID. It is not re-programmable. Dimensions: 2.13 x 3.35 x 0.03" (54 x 85.5 x 0.8mm)



Figure 2.21 RFID cards

Features of RFID cards are:

- EM4001 ISO based RFID IC
- 125kHz Carrier
- 2kbps ASK
- Manchester encoding
- 32-bit unique ID
- 64-bit data stream [Header+ID+Data+Parity]

In most of the RFID system, tags are attached to all items that are to be tracked. These tags are made from a tiny tag-chip that is connected to an antenna. The tag chip contains memory which stores the product's electronic product code (EPC) and other variable information so that it can be read and tracked by RFID readers anywhere. An RFID reader is a network connected device (fixed or mobile) with an antenna that sends power as well as data and commands to the tags. The RFID reader acts as an access point for RFID tagged items so that the tags' data can be made available to business applications.

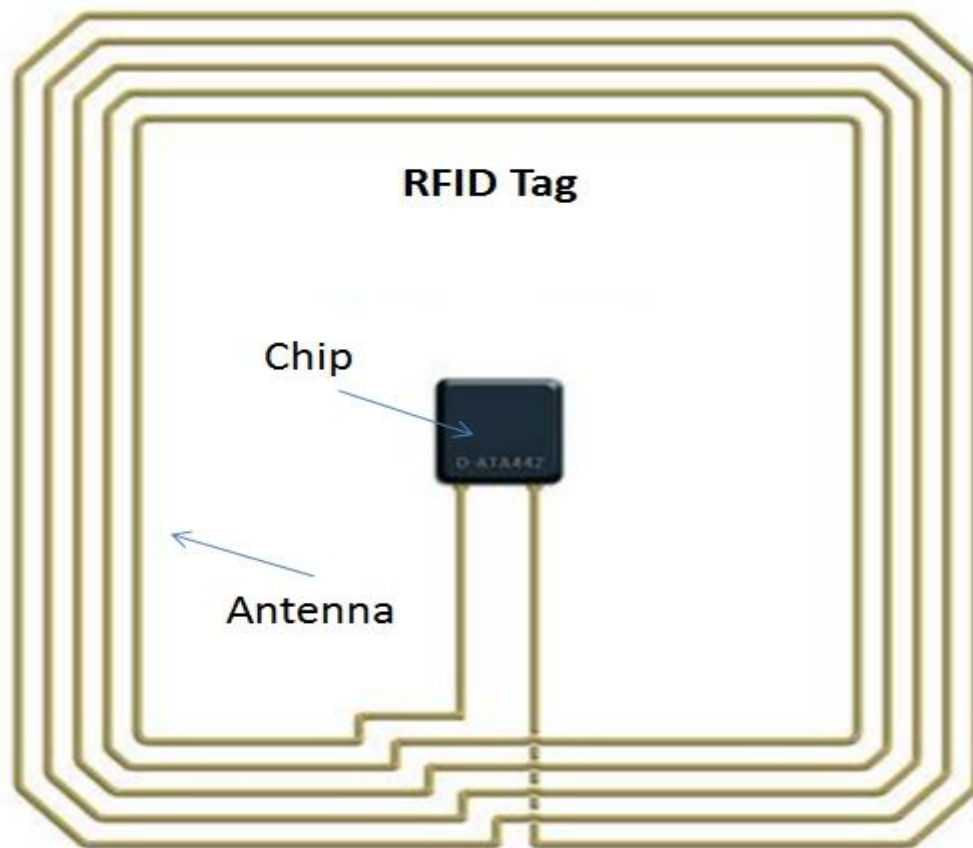


Figure 2.22 Inner look of RFID card

The RFID antenna propagates the wave in both vertical and horizontal dimensions. The field coverage of the wave and also its signal strength is partially controlled by the number of degrees that the wave expands as it leaves the antenna. While the higher number of degrees means a bigger wave coverage pattern it also means lower strength of the signal. Passive RFID tags utilize an induced antenna coil voltage for operation. This induced AC voltage is rectified to provide a voltage source for the device. As the DC

voltage reaches a certain level, the device starts operating. By providing an energizing RF signal, a reader can communicate with a remotely located device that has no external power source such as a battery.

Tag antennas collect energy and channel it to the chip to turn it on. Generally, the larger the tag antenna's area, the more energy it will be able to collect and channel toward the tag chip, and the further read range the tag will have. Tag antennas can be made from a variety of materials; they can be printed, etched, or stamped with conductive ink, or even vapor deposited onto labels. The tag antenna not only transmits the wave carrying the information stored in the tag, but also needs to catch the wave from the reader to supply energy for the tag operation. Tag antenna should be small in size, low-cost and easy to fabricate for mass production. In most cases, the tag antenna should have omnidirectional radiation or hemispherical coverage. Generally, the impedance of the tag chip is not 50 ohm, and the antenna should realize the conjugate match with the tag chip directly, in order to supply the maximum power to the tag chip. Tag antenna may be a signal turn or multiple turns as shown here.

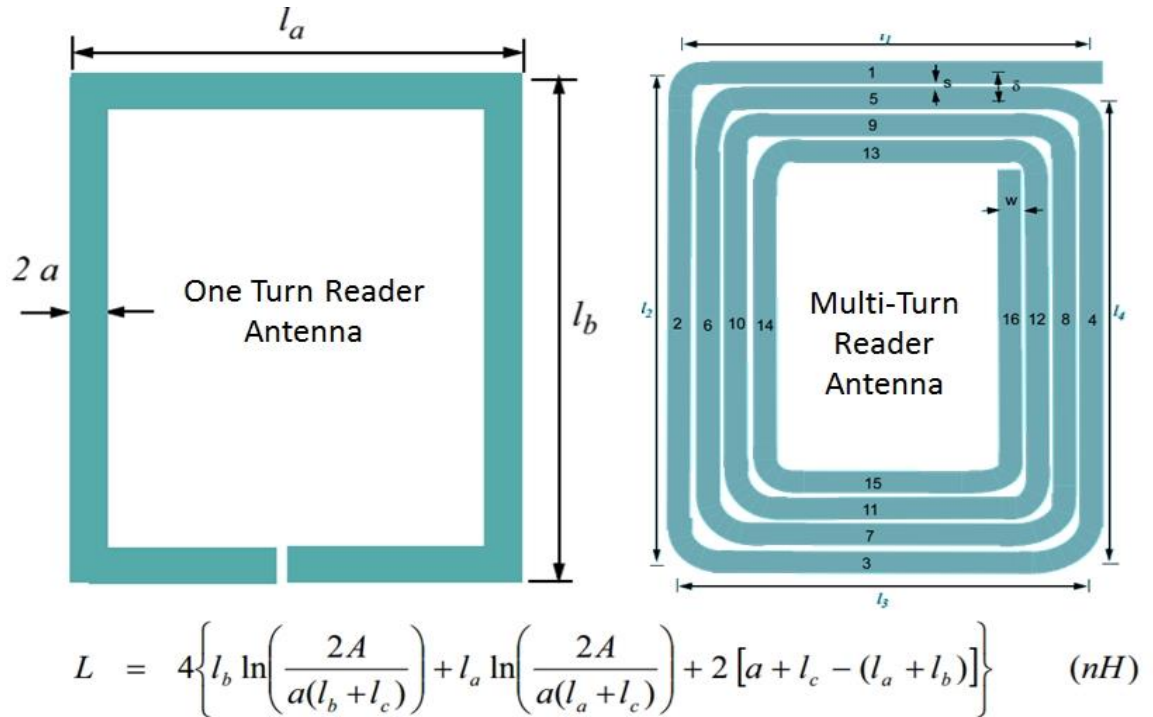


Figure 2.23 Tag antenna with single and multiple turns

Reader antennas convert electrical current into electromagnetic waves that are then radiated into space where they can be received by a tag antenna and converted back to electrical current.

2.4.3 ESP32 NodeMCU

NodeMCU is an open source Lua based firmware for the ESP32 and ESP8266 WiFi SOC from Espressif and uses an on-module flash-based SPIFFS file system. NodeMCU is implemented in C and is layered on the Espressif ESP-IDF.

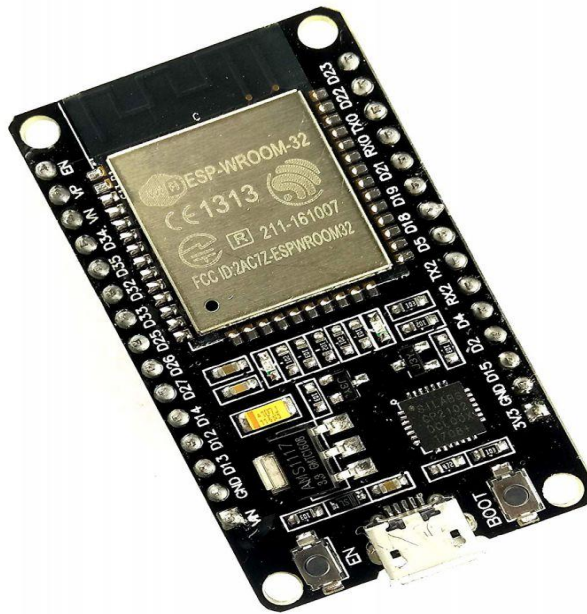


Figure 2.24 ESP32 NodeMCU

NodeMCU is an open-source IoT platform. It includes firmware that runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which is based on the ESP-12 module. It is based on the eLua project and built on the Espressif Non-OS SDK for ESP8266. It uses many open source projects, such as lua-cjson and SPIFFS. The prototyping hardware typically used is a circuit board functioning as a dual in-line package (DIP) which integrates a USB controller with a smaller surface-mounted board containing the MCU and antenna. The choice of the DIP format allows for easy prototyping on breadboards.

NodeMCU is based on the ESP8266-12E Wi-Fi System-On-Chip, loaded with an open-source, Lua-based firmware. It is perfect for IoT applications, and other situations where wireless connectivity is required. This chip has a great deal in common with the Arduino, they are both microcontrollers equipped prototyping boards which can be programmed

using the Arduino IDE. Figure 2.20 shows the pins on the NodeMCU ESP32 development board.

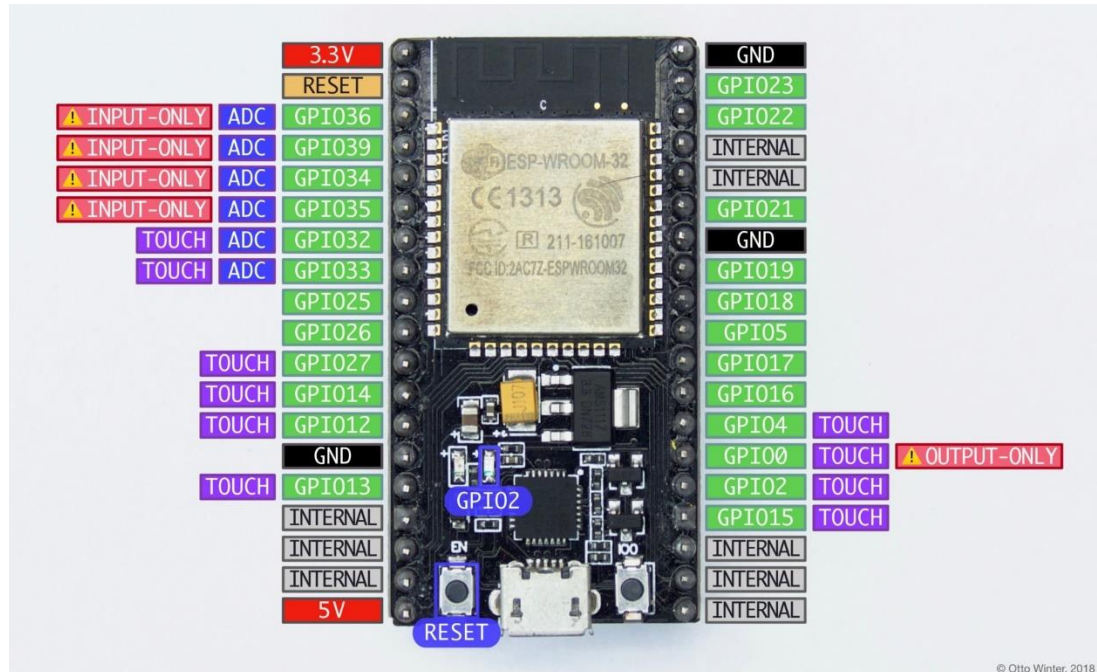


Figure 2.25 Pins on the NodeMCU ESP32 development board.

- GPIO0 is used to determine the boot mode on startup. It should therefore not be pulled LOW on startup to avoid booting into flash mode. You can, however, still use this as an output pin.
- GPIO34-GPIO39 cannot be used as outputs (even though GPIO stands for “general purpose input output”).
- GPIO32-GPIO39: These pins can be used with the Analog To Digital Sensor to measure voltages.
- GPIO2: This pin is connected to the blue LED on the board as seen in the picture above. It also supports the touch pad binary sensor as do the other pins marked touch in the above image.
- 5V is connected to the 5V rail from the USB bus and can be used to power the board. Note that the UART chip is directly connected to this rail and you therefore cannot supply other voltages into this pin.

WORKING:

Here, an IoT based RFID attendance system with RDM6300 RFID Reader, ESP32 NodeMCU, RFID tags and Arduino. When you swipe an RFID tag next to the RFID reader, it saves the user UID and time in a MySQL database with the help of ESP32 NodeMCU. The block diagram of this project is shown in figure 2.21.

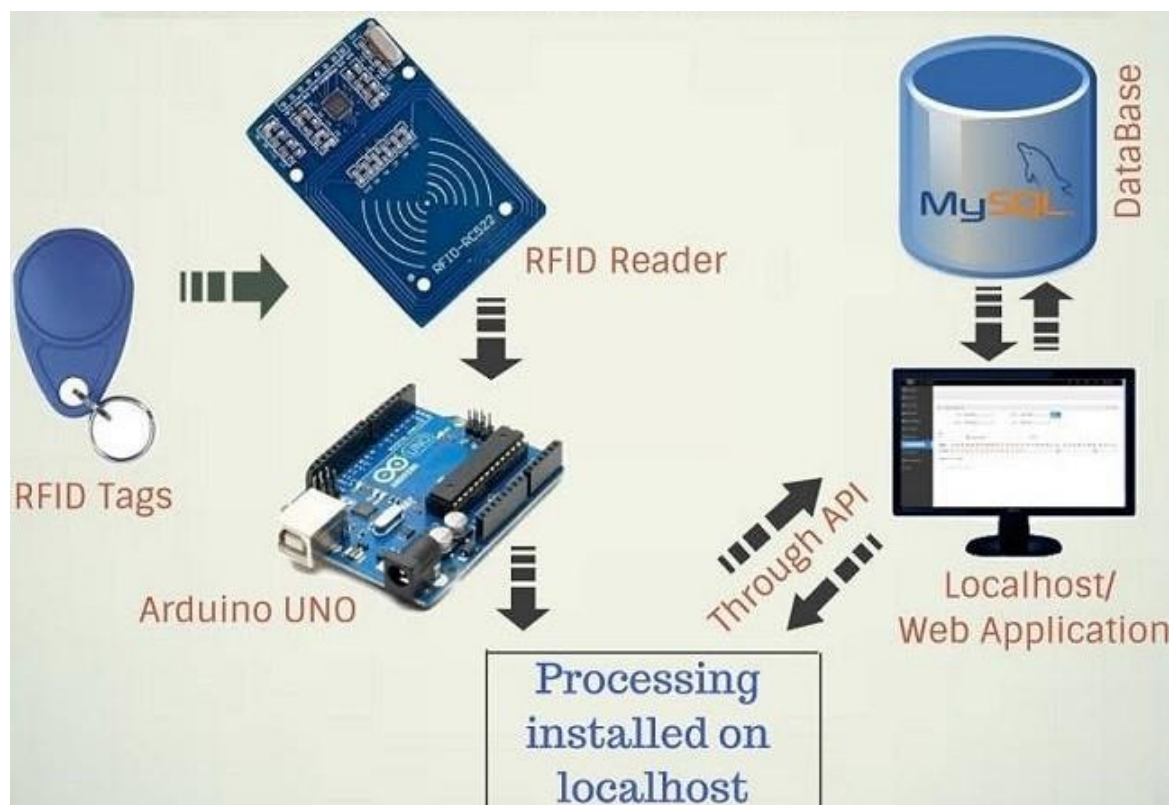


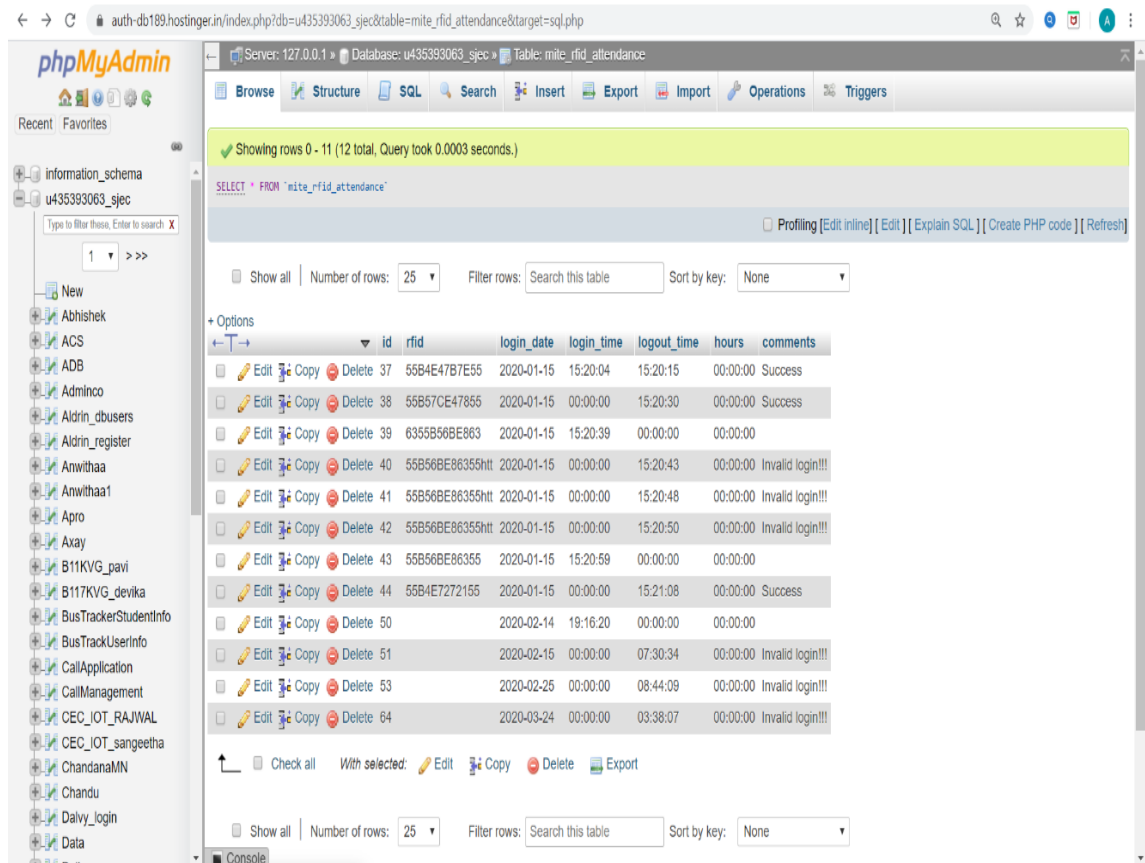
Figure 2.26 Block diagram of IoT based RFID attendance system

Here from laptop power supply is given to both Arduino and Node MCU, then Arduino receiver pin is connected to transmitter pin of Node MCU. The RX pin of Arduino is connected to TX pin of RFID card reader and the VCC and GND pins of card reader is connected to 5V and GND of Arduino.

In this project, 2 push buttons are used as login and logout and these push buttons are connected to the pins of Arduino. Then RFID tag is swiped on a RFID card reader. The RFID card reader sends the unique 32-bit ID to the Arduino and this ID is processed. Then, this ID is sent to MySQL database using ESP32 NodeMCU.

In MySQL database, a PHP code is written such that the ID number, RFID unique ID, login date, login time, logout time and the comments is updated in the form of a table. So, when the RFID card is swiped and the login button is pressed, the login date and time is

updated in the database and similarly when that card is swiped again and logout button is pressed, the logout time is updated and the comment is updated as success. But when one card is used to login and different card is used to logout, comment is updated as invalid login. Same RFID card that is used for logging in must be used for logging out. Any number of cards can be swiped for logging in by pressing login button and that date and time will be updated in the database. But if same card is swiped by pressing logout without being logged in, the comment is updated as invalid login. When login button or logout button is pressed and the RFID card is not swiped, the comments updated as invalid login. If different card is used for logging in and logging out, the comment is updated as invalid login. The screenshot of the database which was used in the project for updating the status is shown in figure 2.27.



Showing rows 0 - 11 (12 total, Query took 0.0003 seconds.)

SELECT * FROM 'mile_rfid_attendance'

Options: Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

		id	rfid	login_date	login_time	logout_time	hours	comments
<input type="checkbox"/>	Edit Copy Delete	37	55B4E47B7E55	2020-01-15	15:20:04	15:20:15	00:00:00	Success
<input type="checkbox"/>	Edit Copy Delete	38	55B57CE47855	2020-01-15	00:00:00	15:20:30	00:00:00	Success
<input type="checkbox"/>	Edit Copy Delete	39	6355B56BE863	2020-01-15	15:20:39	00:00:00	00:00:00	
<input type="checkbox"/>	Edit Copy Delete	40	55B56BE86355ht	2020-01-15	00:00:00	15:20:43	00:00:00	Invalid login!!!
<input type="checkbox"/>	Edit Copy Delete	41	55B56BE86355ht	2020-01-15	00:00:00	15:20:48	00:00:00	Invalid login!!!
<input type="checkbox"/>	Edit Copy Delete	42	55B56BE86355ht	2020-01-15	00:00:00	15:20:50	00:00:00	Invalid login!!!
<input type="checkbox"/>	Edit Copy Delete	43	55B56BE86355	2020-01-15	15:20:59	00:00:00	00:00:00	
<input type="checkbox"/>	Edit Copy Delete	44	55B4E7272155	2020-01-15	00:00:00	15:21:08	00:00:00	Success
<input type="checkbox"/>	Edit Copy Delete	50		2020-02-14	19:16:20	00:00:00	00:00:00	
<input type="checkbox"/>	Edit Copy Delete	51		2020-02-15	00:00:00	07:30:34	00:00:00	Invalid login!!!
<input type="checkbox"/>	Edit Copy Delete	53		2020-02-25	00:00:00	08:44:09	00:00:00	Invalid login!!!
<input type="checkbox"/>	Edit Copy Delete	64		2020-03-24	00:00:00	03:38:07	00:00:00	Invalid login!!!

Options: Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

Figure 2.27 Screenshot of database

For further implementation, the hours taken by the RFID card being swiped to login and logout can also be updated in the database. This can be done by taking the login time and logout time and then calculating the difference between them.

ARDUINO CODE:

```
#include <SoftwareSerial.h>

SoftwareSerial nodeMCUSerial(6,7);

#define logout  3

#define login  2

#define pressed  0

boolean logout_btn = false;

bool switch_status=0;

char data_in;

char rfid[12];

void setup() {

  Serial.begin(9600);

  nodeMCUSerial.begin(9600);

  pinMode(login,INPUT_PULLUP);

  pinMode(logout,INPUT_PULLUP);

}

void loop()

{

  if(digitalRead(login)==pressed)

  {

    logout_btn = false;

    data();

  }

}
```

```
    while(digitalRead(login)==pressed);

}

if(digitalRead(logout)==pressed)

{

    logout_btn = true;

    data();

    while(digitalRead(login)==pressed);

}}

void data()

{

    boolean flag = 1;

    while(flag)

    {

        if(digitalRead(login)==pressed)

        {

            logout_btn = false;

        }

        if(digitalRead(logout)==pressed)

        {

            logout_btn = true;

        }

        if(Serial.available())
```

```

{

data_in=Serial.read();

if(data_in>'0' & data_in<'F'){

    rfid[0]=data_in;

    for(int i=1;i<12;)

    {

        if(Serial.available())

        {

            data_in=Serial.read();

            if(data_in>'0' & data_in<'F')

            {

                rfid[i]=data_in;

                i++;

            }

        }

    }

    //      Serial.print("http://thantrajna.com/mite/update_data.php?rfid=" );

    if(login_btn)

    {Serial.print("http://thantrajna.com/mite/update_data.php?rfid=");

        nodeMCUSerial.print("http://thantrajna.com/mite/update_data.php?rfid=");

    }

    else

    {

        Serial.print("http://thantrajna.com/mite/insert_data.php?rfid=");
    }
}

```

```
nodeMCUSerial.print("http://thantrajna.com/mite/insert_data.php?rfid=");}

for(int i=0; i<12; i++) {

Serial.print(rfid[i]);

nodeMCUSerial.print(rfid[i]);

flag = 0;}

//      Serial.print("&");

//      nodeMCUSerial.print("&");

Serial.println();

for(int i=0; i<100; i++)

{

if(Serial.available())

{

data_in = Serial.read();

}}

Serial.end();

Serial.flush();

delay(100);

Serial.begin(9600);

}}}}
```

Chapter 3

CONCLUSION

This internship was very useful and was really worth attending it. I learnt the basic Arduino programming and came across different applications of Arduino. The knowledge regarding embedded system and IOT have been increased after attending this internship.

I completed internship on Embedded System and IOT topic at Vitvara Technologies, Mangalore. Overall this internship gave me a worthy and precious experiences of learning.

REFERENCES

- [1] www.vitvara.com
- [2] www.tuttokits.com
- [3] URL: <https://www.arduino.cc/en/Guide/ArduinoUno>
- [4] URL: <https://store.arduino.cc/usa/arduino-uno-rev3>
- [5] URL: <https://www.guru99.com/iot-tutorial.html>
- [6] URL: <https://icircuit.net/interfacing-arduino-uno-wtih-nodemcu/1474>
- [7] www.projectsguide.com
- [8] URL: <https://en.wikipedia.org/wiki/NodeMCU>
- [9] URL: <https://circuitdigest.com/article/servo-motor-basics>
- [10] URL: <https://www.theengineeringprojects.com/2018/10/introduction-to-arduino-ide.html>