

**INDUSTRIAL TRAINING REPORT**  
**B.E. (ECE)- 8th SEMESTER**

**INTERNET OF THINGS**

Undertaken at

**National Institute of Electronics and  
Information Technology**



Submitted by

**PARMEET SINGH (PJ111/12)**  
**PRABHAKAR KUMAR GIRI (PJ111/08)**

Under the Guidance of

**DR. SARWAN SINGH AND MANIK KALSI**

**MAY-JUNE, 2018**

## **ACKNOWLEDGEMENT**

I would like to express my deepest appreciation to all those who provided me the possibility to complete this project. A special gratitude I give to our final year project mentor Dr. Sarwan Singh and Mr. Manik Kalsi whose contribution in stimulating suggestions and encouragement helped me to coordinate my project especially in writing this report. I would also like to thank to my parents for giving encouragement, enthusiasm and invaluable assistance to me. Without all this, I might not be able to complete this subject properly.

Second, I would like to thank to Mr Vishal Sharma, faculty coordinator of ECE Department, UIET for giving me the opportunity to undergo industrial training.

My thanks and appreciations also go to my teammates who have willingly helped me out with their abilities.

I perceive this opportunity as a big milestone in my career development. I will strive to use gained skills and knowledge in the best possible way, and I will continue to work on their improvement, in order to attain desired career objectives.

Prabhakar Kumar Giri

## **ABSTRACT**

This report discusses my experience as industrial trainee in National Institute of Electronics and Information Technology from 8th January, 2018 to 22nd June, 2018. Industrial training helped me to understand the current revolution in Internet, mobile, and machine-to-machine (M2M) technologies that can be seen as the first phase of the IoT.

Internet of things is evergreen in the field of technology. Internet of Things development has become impossible without the use of electronics. Skill development in this domain will certainly help individuals as well as organizations for developing innovative products.

Consistent efforts are being made in the field of IoT in order to maximize the productivity and reliability of urban infrastructure. Problems such as, traffic congestion, limited car parking facilities and road safety are being addressed by IoT. The proposed Smart Parking system and Smart Bin consists of an on-site deployment of an IoT module that is used to monitor and signalize the state of availability of each single parking space. Details of the tools and languages used during the tenure are also explained in the report.

## **Table Of Contents**

<b>S.No.</b>	<b>Topic</b>	<b>Page No.</b>
i	Acknowledgement	i
ii	Abstract	ii
iii	Table Of Content	iii
iv	List Of Figures	iv
v	List Of Acronyms	v
vi	Organization Details	vi
1.	<b>Chapter 1:Introduction to IoT</b>	1
	1.1 What internet of things can do for us?	1
	1.2 Potential benefits of IoT in the business world include:	2
	1.3 Network Devices and the Internet of Things	2
	1.4 Issues Around IoT	2
2.	<b>Chapter 2: Use of IoT in Embedded System</b>	4
3.	<b>Chapter 3: Study of Microcontroller AT89C51</b>	5
	3.1 Introduction to Microcontroller AT89C51 (8051)	5
	3.2 Features	5
	3.3 Criteria For Selection of Microcontroller in Embedded System	5
	3.4 Block diagram of 8051 Microcontroller (AT89C51)	6
	3.5 Microcontroller 8051 Architecture	6
	3.6 Pin diagram	8
	3.7 Description of each pin is discussed here	8
	3.8 Interfacing various devices with 8051	9
4.	<b>Chapter 4: Introduction to Arduino</b>	14
	4.1 Introduction	14
	4.2 The arduino board	14
	4.3 Board description	15
	4.4 Integrated development environment	17
	4.5 Input/Output functions	18
	4.6 Other useful functions	19
	4.7 Program to blink led	21

5.	<b>Chapter 5: Proteus Design Suite</b>	22
	5.1. Water tank prototype	22
	5.2. Half wave rectifier prototype	22
	5.3. RL low pass filter prototype	23
	5.4. RL high pass filter prototype	23
	5.5. Full Adder Prototype	24
	5.6. Keypad on virtual terminal	24
	5.7. LCD Interfacing with arduino	25
4.	<b>Chapter 6: Introduction to ESP8266, Raspberry pi and MQTT</b>	26
	6.1 ESP 8266	26
	6.2 Raspberry pi	30
	6.3 MQTT	33
	<b>Chapter 7: IoT based Smart Parking System</b>	35
	7.1 Introduction to IoT based Smart Parking System	35
	7.2 Objectives of IoT based Smart Parking System	35
	7.3 Project Architecture	35
	7.4 Project Implementation and working	42
	7.5 Codes used in IoT based Smart Parking System	43
	<b>Chapter 8: IoT based Smart Dustbin</b>	54
	8.1 Introduction to Iot based Smart Dustbin	54
	8.2 Objectives of Iot based Smart Dustbin	54
	8.3 System Architecture	54
	8.4 Project Implementation and working	56
	8.5 Code used in the project	57
6.	<b>Conclusion</b>	63
7.	<b>References</b>	64

## List Of Figures

S.No.	Figure Description	Page No.
1.	Figure 1.1. Architecture view of IoT	1
2.	Figure 2.1. Use of IoT in embedded systems	4
3.	Fig 3.1 block diagram of 8051	6
	Fig. 3.2 Architecture of 8051	7
	Figure: 3.3 PIN diagram of 8051	8
	Figure: 3.4 Interfacing of LED with 8051	10
	Figure. 3.5 Interface of LCD	11
	Figure: 3.6 Interface of LCD with 8051	12
	Figure: 3.7 interfacing of Keypad	13
	Figure:3.8 Interfacing of 7 segment display	13
4.	Figure:4.1 Arduino Board	14
	Figure: 4.2 Components of Arduino board	15
	Figure: 4.3 A to B connector cable	15
	Figure: 4.4 Arduino IDE opening window	17
	Figure: 4.5 program to blink led	21
5.	Figure 5.1. Water Tank	22
	Figure 5.2. Half wave rectifier	22
	Figure 5.3. RL low pass filter	23
	Figure 5.4. RL high pass filter	23
	Figure 5.5. Full adder	24
	Figure 5.6. Keypad on virtual terminal	24
	Figure 5.7. LCD Interfacing with arduino	25
6.	Figure 6.1.1: ESP8266 pin diagram	26
	Figure 6.1.2: ESP with Arduino Uno	28
	Figure 6.1.3: Interfacing Humidity Sensor	28
	Figure 6.2.1: Raspberry Pi Board	31
	Figure 6.2.2: Raspberry pi 3 Model B	31
		32
	Figure 6.2.3: Difference between various models of raspberry pi	33

	Figure 6.3.1: Working of MQTT	33
7.	Figure 7.1 Arduino IDE Sketch	36
	Figure 7.2 HTML webpage	36
	Figure 7.3 MQTT lens	37
	Figure 7.4 NodeMCU	38
	Figure 7.5 Ultrasonic Sensor working	38
	Figure 7.6 Jump wires	39
	Figure 7.7 Arduino Uno	39
	Figure 7.8 Transformer	40
	Figure 7.9 Capacitor 2200uf 16v	40
	Figure 7.10 LM7805 Pinout Diagram	41
	Figure 7.11 LD33V Regulator	41
	Figure 7.12 Diode	41
	Figure 7.13 Project webpage	42
8.	Figure 8.1 View of IoT based ThingSpeak	55
	Figure 8.2 Arduino Uno	55
	Figure 8.3 ESP8266	56
	Figure 8.4 Projects Monitoring on Cloud	57

### **List Of Acronyms**

**VSM:** Virtual System Modeling

**EPD:** Electronic Product Design

**IoT:** Internet of Things

## **Organization Details**

**Organization's perspective:** National Institute of Electronics & Information Technology (NIELIT), an Autonomous Scientific Society under the administrative control of Ministry of Electronics & Information Technology, Government of India, was set up to carry out Human Resource Development and related activities in the area of Information, Electronics & Communications Technology (IECT). NIELIT is engaged both in Formal & Non-Formal Education in the area of IECT besides development of industry oriented quality education and training programs in the state-of-the-art areas. NIELIT has endeavored to establish standards to be the country's premier institution for Examination and Certification in the field of IECT. It is also one of the National Examination Body, which accredits institutes/organizations for conducting courses in IT in the non-formal sector.

**Organization Vision:** To be the leader in the development of industry oriented quality education and training and be the country's premier Institution for examination and certification in the field of Information, Electronics and Communications Technology (IECT).

**NIELIT Services:** The basket of activities of NIELIT is further augmented by the wide range of projects that it undertakes. NIELIT has demonstrated its capability and capacity to undertake R&D projects, consultancy services, turnkey projects in office automation, software development, website development etc. NIELIT is also the nodal implementing agency on behalf of Data Digitization of the population of 15 assigned States and 1 Union Territories for the creation of National Population Register (NPR) project of Registrar General of India (RGI)

NIELIT, Chandigarh has proven its capability of handing large projects of National importance. It has carved a niche for itself by executing projects in a time bound manner. Be it Power, Health, Education and Agriculture or any other sector, NIELIT Chandigarh has enormous success stories par excellence. The Centre offers services in the following area:-

- Capacity Building in IECT
- Computer Education in Schools
- Corporate Training
- Turn key Projects in the field of IT
- large scale data capturing & data processing



**DEPARTMENT OF COMPANY:**

NIELIT Chandigarh is providing the quality education in the IT and Electronic industry.

**Department of Electronic and Hardware**

Demand of highly skilled, imaginative product design engineers are continue to rise and career opportunity are varied, challenging and rewarding. Course on Electronic Product Design and Embedded design are beneficial for those candidates who wants to be focus a career in electronics design or manufacturing or in the engineering industry or production of complex, modern electronic system. Networking and Hardware is a great career field where one can easily get an entry to any company.

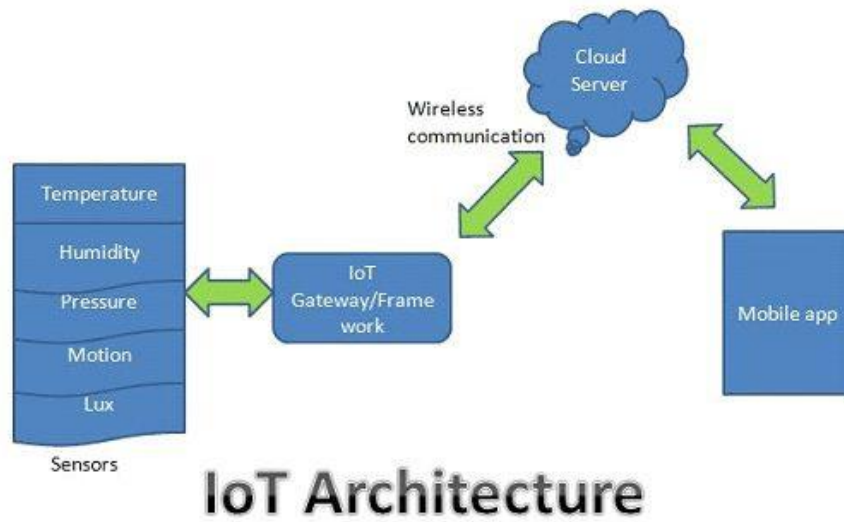
**Courses include:**

1. Electronic Product Design with live designing of a product.
1. Computer Hardware and Networking-Assembly and Troubleshooting.
3. Embedded system design with a project.

**The following topics are covered in electronics**

1. Embedded C language
1. Designing microcontroller based system-8051.
3. VHDL
4. PCB design and layout
5. Fundamentals of design and technology of electronic and product.
6. ARM Cortex -7.

## Chapter 1: Introduction to IoT



**Figure 1.1. Architecture view of IoT**

The term Internet of Things (often abbreviated IoT) was coined by industry researchers but has emerged into mainstream public view only more recently. IoT is a network of physical devices, including things like smart phones, vehicles, home appliances, and more, that connects to and exchange data with computer.

Internet of Things represents a general concept for the ability of network devices to sense and collect data from the world around us, and then share that data across the Internet where it can be processed and utilized for various interesting purposes. Some also use the term industrial Internet interchangeably with IoT. This refers primarily to commercial applications of IoT technology in the world of manufacturing. The Internet of Things is not limited to industrial applications, however.

### **1.1 What internet of things can do for us?**

Some future consumer applications envisioned for IoT sound like science fiction, but some of the more practical and realistic sounding possibilities for the technology include:

1. Receiving warnings on your phone or wearable device when IoT networks detect some physical danger is detected nearby.
2. Self-parking automobiles.

3. Automatic ordering of groceries and other home supplies.
4. Automatic tracking of exercise habits and other day-to-day personal activity including goal tracking and regular progress reports.

### **1.2 Potential benefits of IoT in the business world include:**

- location tracking for individual pieces of manufacturing inventory.
- fuel savings from intelligent environmental modeling of gas-powered engines.
- new and improved safety controls for people working in hazardous environments.

### **1.3 Network Devices and the Internet of Things**

- All kinds of ordinary household gadgets can be modified to work in an IoT system. [Wi-Fi](#) network adapters, motion sensors, cameras, microphones and other instrumentation can be embedded in these devices to enable them for work in the Internet of Things.
- [Home automation systems](#) already implement primitive versions of this concept for things like [smart light bulbs](#), plus other devices like wireless scales and wireless blood pressure monitors that each represent early examples of IoT gadgets. Wearable computing devices like [smart watches](#) and glasses are also envisioned to be key components in future IoT systems.
- The same wireless communication protocols like Wi-Fi and [Bluetooth](#) naturally extend to the Internet of Things also.

### **1.4 Issues Around IoT**

- Internet of Things immediately triggers questions around the privacy of personal data. Whether real-time information about our physical location or updates about our weight and blood pressure that may be accessible by our health care providers, having new kinds and more detailed data about ourselves streaming over wireless networks and potentially around the world is an obvious concern.
- Supplying power to this new proliferation of IoT devices and their network connections can be expensive and logistically difficult. Portable devices require batteries that someday must be replaced. Although many mobile devices are optimized for lower power usage, energy costs to keep potentially billions of them running remains high.

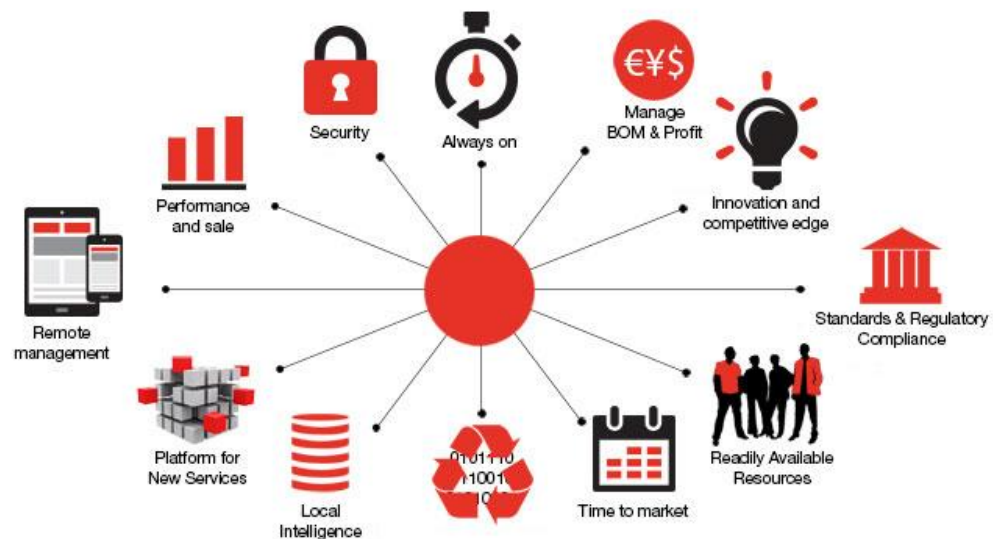
- IoT assumes that the underlying network equipment and related technology can operate semi-intelligently and often automatically. Simply keeping [mobile devices](#) connected to the Internet can be difficult enough much less trying to make them smarter.
- People have diverse needs that require an IoT system to adapt or be configurable for many different situations and preferences. Finally, even with all those challenges overcome, if people become too reliant on this automation and the technology is not highly robust, any technical glitches in the system can cause serious physical and/or financial damage.
- Numerous corporations and start-up ventures have latched onto the Internet of Things concept looking to take advantage of whatever business opportunities are available. While competition in the market helps lower prices of consumer products, in the worst case it also leads to confusing and inflated claims about what the products do.

## **Chapter 2: Use of IoT in Embedded System**

Embedded systems are part and parcel of every modern electronic component. These are low power consumption units that are used to run specific tasks for example remote controls, washing machines, microwave ovens, RFID tags, sensors, actuators and thermostats used in various applications, networking hardware such as switches, routers, modems, mobile phones, PDAs, etc.

Embedded systems are at the cornerstone for the deployment of many Internet of Things (IoT) solutions, especially within certain industry verticals and Industrial Internet of Things (IIoT) applications.

Major players in embedded system hardware and software developments are aiming to bring these transformations into their products to take advantage of growing IoT market.



**Figure 2.1. Use of IoT in embedded systems**

## **Chapter 3: Study of Microcontroller AT89C51**

**3.1 Introduction to Microcontroller AT89C51 (8051):** A Microcontroller is a VLSI IC that contains a CPU (Processor) along with some other peripherals like Memory (RAM and ROM), I/O Ports, Timers/Counters, Communication Interface, ADC, etc. Originally, 8051 Microcontrollers were developed using N-MOS Technology but the use of battery powered devices and their low power consumption lead to usage of CMOS Technology (which is famous for its low power consumption). Majority of the modern 8051 Microcontrollers are Silicon IP Cores (Intellectual Property Cores) but discrete 8051 Microcontroller IC's are also available. Because of their low power consumption, smaller size and simple architecture, 8051 IP Cores are used in FPGAs (Field Programmable Gate Array) and SoCs (System on Chip) instead of Advanced ARM Architecture based MCUs.

### **3.2 Features:**

- Microcontroller (MC) may be called computer on chip since it has basic features of micro processor with internal ROM, RAM, Parallel and serial ports within single chip. or we can say microprocessor with memory and ports is called as microcontroller. This is widely used in washing machines, VCD player, microwave oven, robotics or in industries.
- Microcontroller can be classified on the basis of their bits processed like 8 bit MC, 16 bit MC.
- 8 bit microcontroller means it can read, write and process 8 bit data. Ex. 8051 microcontroller. Basically 8 bit specifies the size of data bus. 8 bit microcontroller means 8bit data can travel on the data bus or we can read, write process 8 bit data.

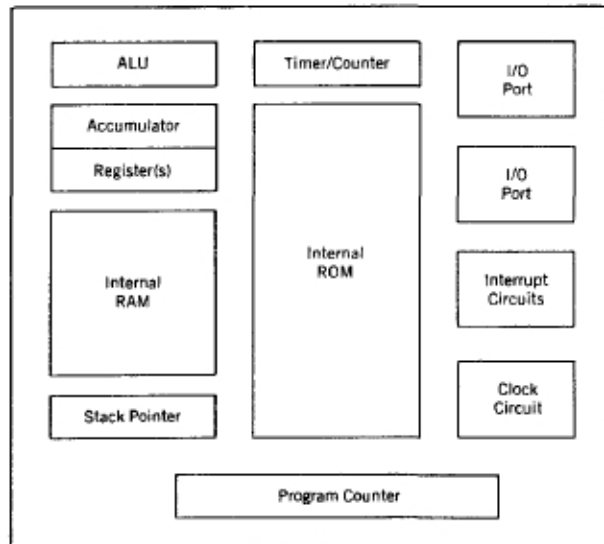
### **3.3 Criteria For Selection of Microcontroller in Embedded System**

3.3.1 Meeting the computing needs of task at hand efficiently and cost effectively:

- Speed of operation
- Packing
- Power consumption
- Amount of RAM and ROM on chip
- No. of I/O pins and timers on chip

- Cost

### 3.4 Block diagram of 8051 Microcontroller (AT89C51):



**Fig 3.1 block diagram of 8051**

### 3.5 Microcontroller 8051 Architecture

It is 8bit microcontroller, means MC 8051 (AT89C51) can Read, Write and Process 8 bit data. This is mostly used microcontroller in the robotics, home appliances like mp3 player, washing machines, electronic iron and industries.

Mostly used blocks in the architecture of 8051 are as follows:

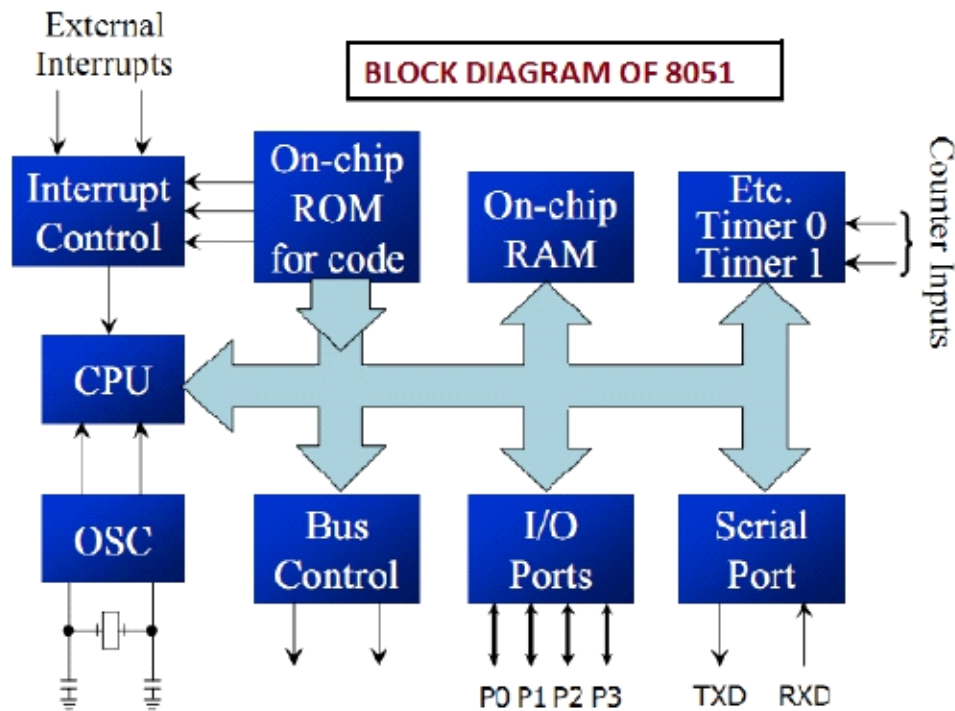


Fig. 3.2 Architecture of 8051

### 3.5.1 118 Byte RAM for Data Storage

MC 8051 has 118 byte Random Access memory for data storage. Random access memory is non-volatile memory. During execution for storing the data the RAM is used. RAM consists of the register banks, stack for temporary data storage. It also consists of some special function register (SFR) which are used for some specific purpose like timer, input output ports etc. Normally microcontroller has 156 byte RAM in which 118 byte is used for user space which is normally Register banks and stack. But other 118 byte RAM which consists of SFRs.

### 4 KB ROM

- In 8051, 4KB read only memory (ROM) is available for program storage. This is used for permanent data storage. Or the data which is not changed during the processing like the program or algorithm for specific applications. Address range of PC is 0000 H to 0FFF H means total 4 KB locations are available from 0000H to 0FFF H. At which we can save the program.



- Address Range of PC: Address range of PC means program counter (which points the next instruction to be executing) can be moved between these locations or we can save the program from this location to this location.

### 3.6 Pin diagram

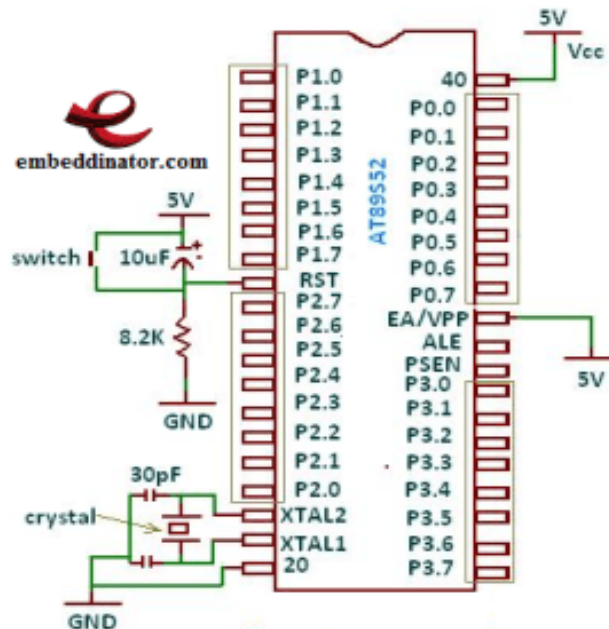


Figure: 3.3 PIN diagram of 8051

### 3.7 Description of each pin is discussed here:

- VCC → 40<sup>th</sup> pin - 5V supply
- VSS → 10<sup>th</sup> pin - GND
- XTAL1/XTAL2 are for oscillator input
- Port 0 – 31 to 39 – AD0/AD7 and P0.0 to P0.7
- Port 1 – 1 to 8 – P1.0 to P1.7
- Port 1 – 11 to 18 – P1.0 to P1.7 and A 8 to A15
- Port 3 – 10 to 17 – P3.0 to P3.7
- RST – for Restarting 8051

- ALE – Address latch enable

1 – Address on AD 0 to AD 7

0 – Data on AD 0 to AD 7

Pin-40 : Named as Vcc is the main power source. Usually its +5V DC.

Pins 31-39: Known as Port 0 (P0.0 to P0.7) –In addition to serving as I/O port, lower order address and data bus signals are multiplexed with this port (to serve the purpose of external memory interfacing).

Pin-31:- ALE aka Address Latch Enable is used to demultiplex the address-data signal of port 0

(for external memory interfacing.) 1 ALE pulses are available for each machine cycle.

Pin- 19:PSEN or Program Store Enable is used to read signal from external program memory.

Pins- 11-18:- Known as Port 1 (P 1.0 to P 1.7) in addition to serving as I/O port, higher order address bus signals are multiplexed with this quasi bidirectional port.

Pins 18 and 19:- Used for interfacing an external crystal to provide system clock.

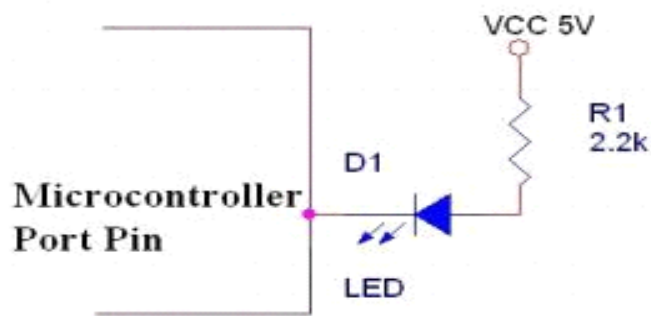
Pins 10 – 17: Known as Port 3. This port also serves some other functions like interrupts, timer input, control signals for external memory interfacing RD and WR , serial communication signals RxD and TxD etc. This is a quasi bi directional port with internal pull up.

Pin 9:- As explained before RESET pin is used to set the 8051 microcontroller to its initial values,while the microcontroller is working or at the initial start of application. The RESET pin must be set high for 1 machine cycles.

Pins 1 – 8:- Known as Port 1. Unlike other ports, this port does not serve any other functions. Port 1 is an internally pulled up, quasi bi directional I/O port.

### **3.8 INTERFACING OF VARIOUS DEVICES WITH 8051**

#### **3.8.1. Interfacing of LEDs with AT89C51**

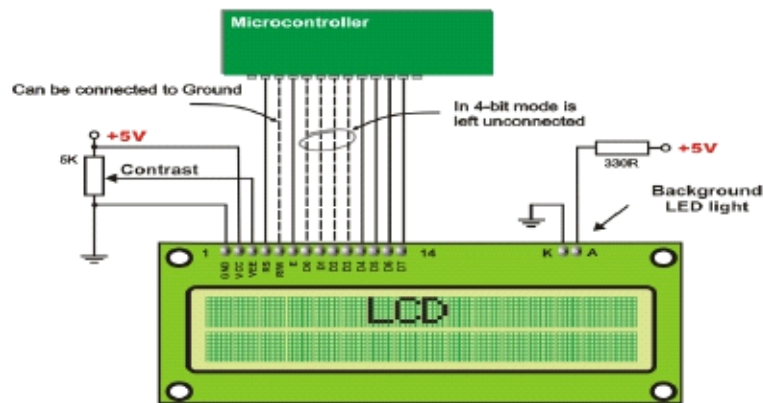


**Figure: 3.4 Interfacing of LED with 8051**

Figure 4.4 shows how to interface the LED to microcontroller. As you can see the Anode is connected through a resistor to Vcc & the Cathode is connected to the Microcontroller pin. So when the Port Pin is HIGH the LED is OFF & when the Port Pin is LOW the LED is turned ON.

### **3.8.2 Interfacing of LCD with AT89C51**

A Liquid Crystal Display is an electronic device that can be used to show numbers or text. There are two main types of LCD display, numeric displays (used in watches, calculators etc) and alphanumeric text displays (often used in devices such as photocopiers and mobile telephones). The display is made up of a number of shaped 'crystals'. In numeric displays these crystals are shaped into 'bars', and in alphanumeric displays the crystals are simply arranged into patterns of 'dots'. Each crystal has an individual electrical connection so that each crystal can be controlled independently. When the crystal is 'off' (i.e. when no current is passed through the crystal) the crystal reflects the same amount of light as the background material, and so the crystal cannot be seen. However when the crystal has an electric current passed through it, it changes shape and so absorbs more light. This makes the crystal appear darker to the human eye - and so the shape of the dot or bar can be seen against the background.



**Figure. 3.5 Interface of LCD**

On most LCD displays there is memory for 40 characters on each line. Each space in the RAM memory can be thought of as a 'box' which is ready to hold a single character.

Each RAM 'box' has a numbered address to describe it. The first line RAM 'boxes' are at addresses 118 to 191, the second line RAM 'boxes' are from 191 to 155.

16x1 displays have a window that is two lines deep. That means that 16 letters can be seen on each line. If a character is to be printed on the second line, it is necessary to move the cursor to the start of line 1. Moving the cursor is very simple; simply send the RAM address (of the 'box' to be moved) as an instruction. Therefore to move the cursor to the start of the second line, simply send the instruction '191' to the LCD module. To move the cursor to the fifth position on the second line send the instruction '197' (=191+5).

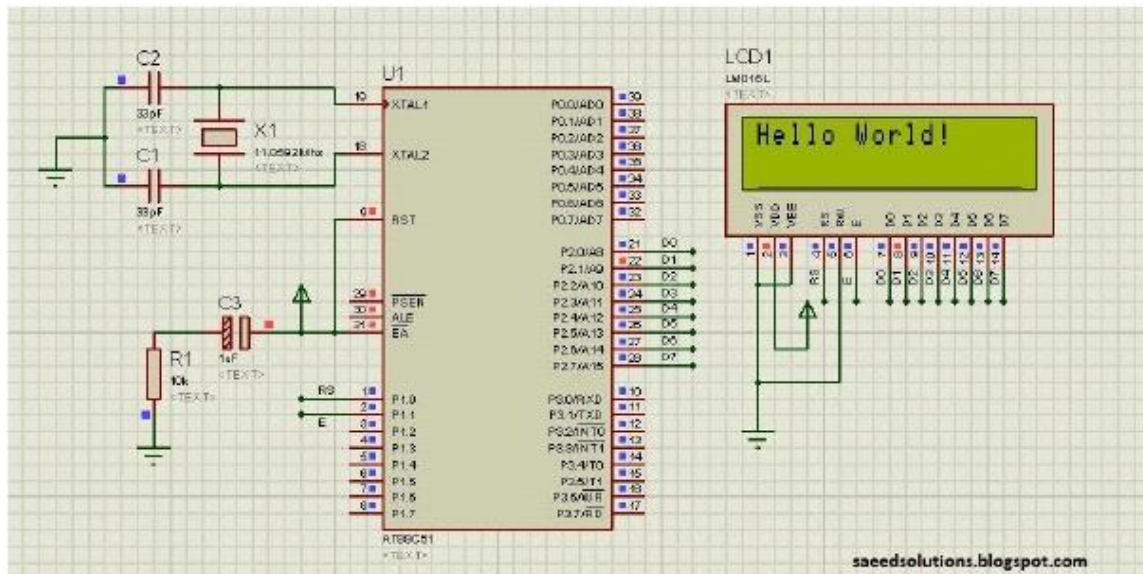


Figure: 3.6 Interface of LCD with 8051

### 3.8.3 Interfacing with keypad

As you can see the switch release is clean without any bouncing. When a switch is pressed the contacts open & close for about 10ms. In the above programs the LED will flicker initially when the switch is pressed because of the SWITCH BOUNCING but since the flickering will be very fast & will not be detected by human eye. Even though 10ms is very short time in human terms for a microcontroller it is a very long time. Without SWITCH DEBOUNCING the controller will think that the switch was pressed many times. As you have seen in the interfacing of switches to microcontroller, normally the port pin is high but when a switch is pressed the controller pin gets a Low signal and we come to know that a switch has been pressed. One end of switch is connected to the port pin whereas the other end is connected to the Ground. In case of matrix Keypad both the ends of switches are connected to the port Pin. Over here we have considered a 4x3 matrix keypad i.e. four rows and three columns. So in all twelve switches have been interfaced using just seven lines. The adjoining figure shows the diagram of a matrix keypad and how it is interfaced with the controller. As you can see no pin is connected to ground, over here the controller pin itself provides the ground. We pull one of the Column Pins low & check the row pins if any of the Pin is low then we come to know which switch is pressed. Suppose we make column 1 pin low and while checking the rows we get Row 3 is low then we come to know switch 7 has been pressed.

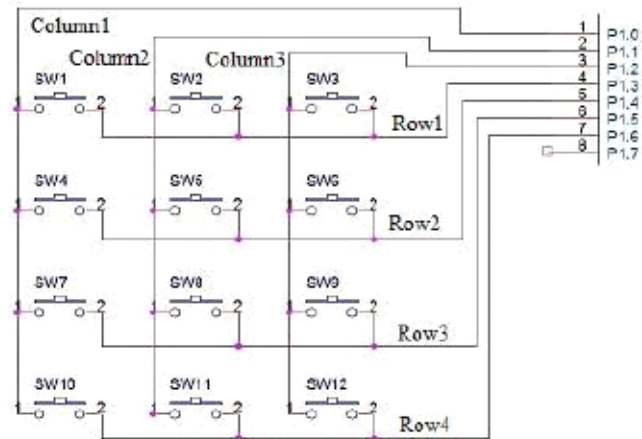


Figure: 3.7 interfacing of Keypad

### 3.8.4. Interfacing of 7-Segment Display

7 Segment displays are basically 7 LED's. It will be much easier to understand if you first read Interfacing LED's to Microcontroller.

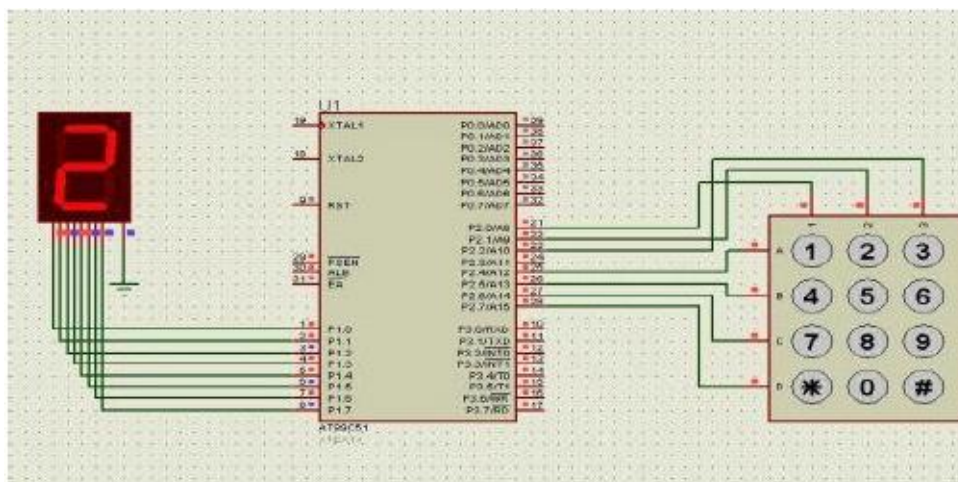
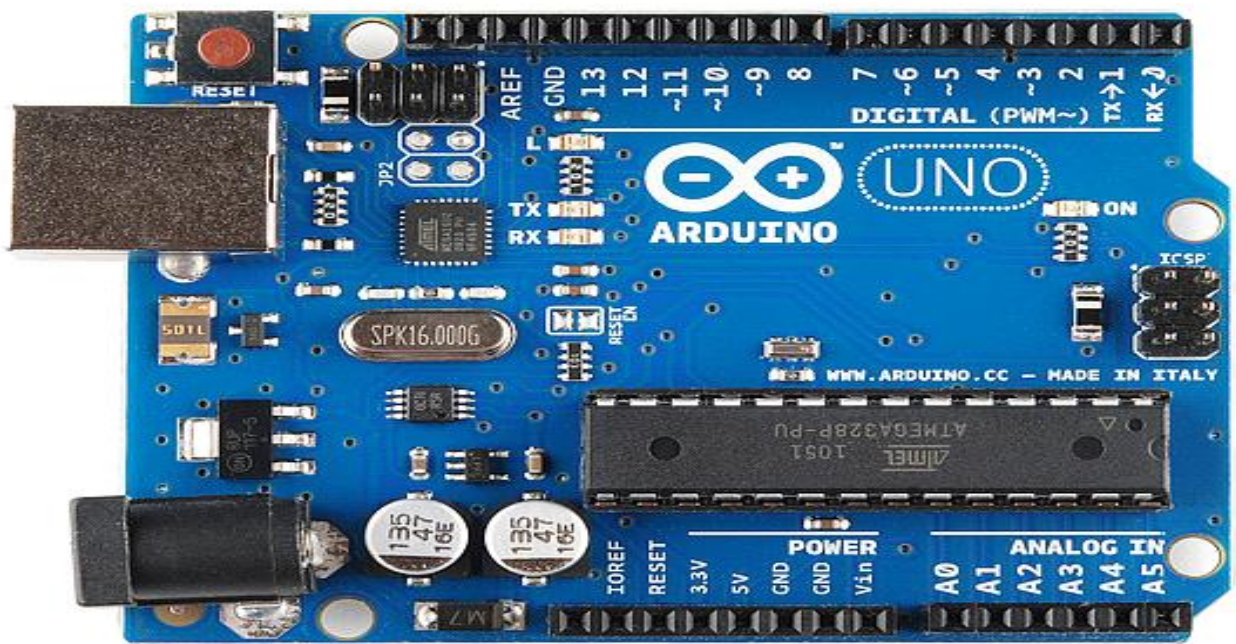


Figure:3.8 Interfacing of 7 segment display



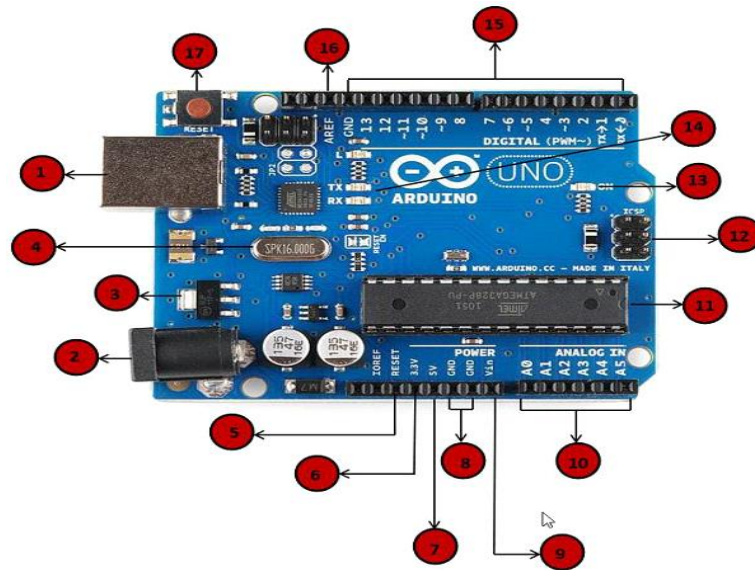
## Chapter 4: Introduction to Arduino

**4.1 Introduction:** A micro-controller is a small computer on a single integrated circuit containing a processor core, memory, and programmable input/output peripherals. The important part for us is that a micro-controller contains the processor (which all computers have) and memory, and some input/output pins that you can control. (often called GPIO -General Purpose Input Output pins). The Arduino platform has become quite popular with people just starting out with electronics, and for good reason. Unlike most previous programmable circuit boards, the Arduino does not need a separate piece of hardware (called a programmer) in order to load new code onto the board – you can simply use a USB cable. Additionally, the Arduino IDE uses a simplified version of C++, making it easier to learn to program. Finally, Arduino provides a standard form factor that breaks out the functions of the micro-controller into a more accessible package.



**Figure:4.1 Arduino Board**

**4.2 The arduino board:** There are many varieties of Arduino boards (explained on the next page) that can be used for different purposes. Some boards look a bit different from the one below, but most Arduinos have the majority of these components in common:



**Figure: 4.2 Components of Arduino board**

### 4.3 Board description

#### Power USB(1)

Arduino board can be powered by using the USB cable from your computer. All you need to do is connect the USB cable to the USB connection (1).

#### USB 1.0 A TO B CABLES

Standard USB 1.0 peripheral cables for connecting printers, scanners, USB hard drives, and other USB devices.



**Figure: 4.3 A to B connector cable**

#### Power (Barrel Jack):



Arduino boards can be powered directly from the AC mains power supply by connecting it to the Barrel Jack (1).

### **Voltage Regulator(3)**

The function of the voltage regulator is to control the voltage given to the Arduino board and stabilize the DC voltages used by the processor and other elements.

### **Crystal Oscillator(4)**

The crystal oscillator helps Arduino in dealing with time issues. How does Arduino calculate time? The answer is, by using the crystal oscillator. The number printed on top of the Arduino crystal is 16.000MHz. It tells us that the frequency is 16,000,000 Hertz or 16 MHz.

### **Arduino Reset(17)**

You can reset your Arduino board, i.e., start your program from the beginning. You can reset the UNO board in two ways. First, by using the reset button (17) on the board. Second, you can connect an external reset button to the Arduino pin labelled RESET (5).

### **Analog pins(10)**

The Arduino UNO board has five analog input pins A0 through A5. These pins can read the signal from an analog sensor like the humidity sensor or temperature sensor and convert it into a digital value that can be read by the microprocessor.

### **Power LED indicator(13)**

This LED should light up when you plug your Arduino into a power source to indicate that your board is powered up correctly. If this light does not turn on, then there is something wrong with the connection.

### **TX and RX LEDs (14)**

On your board, you will find two labels: TX (transmit) and RX (receive). They appear in two places on the Arduino UNO board. First, at the digital pins 0 and 1, to indicate the pins responsible for serial communication. Second, the TX and RX led (13). The TX led flashes with different speed while sending the serial data. The speed of flashing depends on the baud rate used by the board. RX flashes during the receiving process.

## Digital I/O(15)

The Arduino UNO board has 14 digital I/O pins (15) (of which 6 provide PWM (Pulse Width Modulation) output. These pins can be configured to work as input digital pins to read logic values (0 or 1) or as digital output pins to drive different modules like LEDs, relays, etc. The pins labeled “~” can be used to generate PWM.

## AREF(16)

AREF stands for Analog Reference. It is sometimes, used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the analog input pins.

### 4.3 Integrated development environment

A arduino C/C++ sketch, as seen by the arduino IDE programmer, consist of only two functions.

**a) Setup:** This function is called once when a sketch starts after power-up or reset. It is used to initialize variables, input and output pin modes, and other libraries needed in the sketch.

**b) Loop:** After setup has been called, function loop is executed repeatedly in the main program. It controls the board until the board is powered of or is reset.

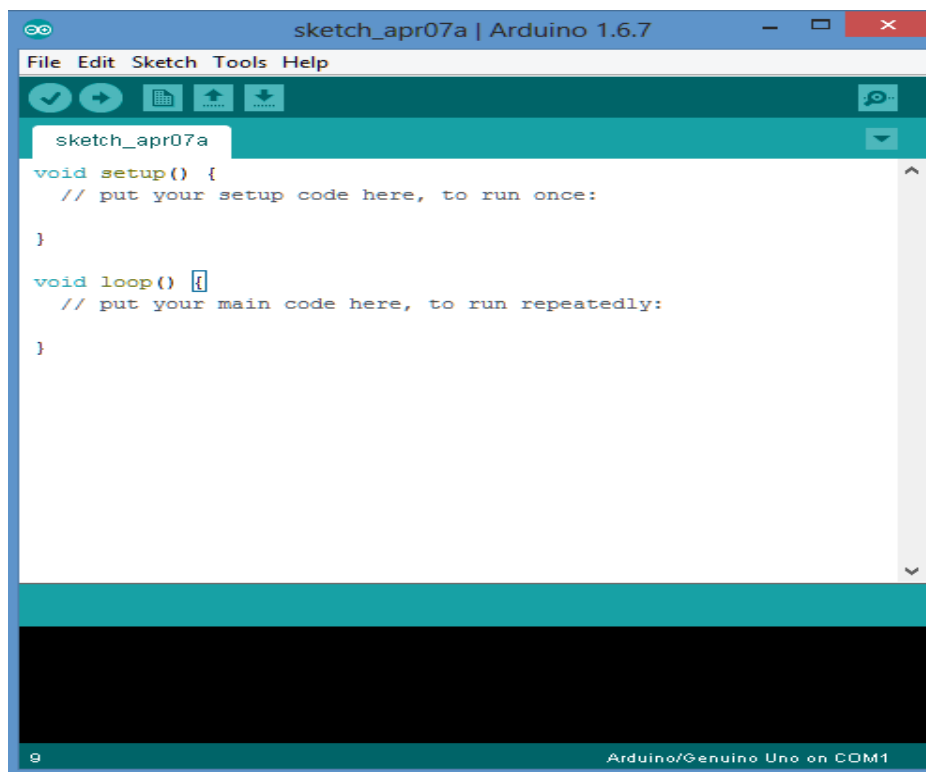


Figure: 4.4 arduino IDE opening window

## 4.4 Input/Output functions

### **pinMode**

This command is used to set pins as input or output in the Setup Loop. Arduino digital pins are set to input by default, hence there is need to specifically declare them as inputs using `pinMode()`.

Pins configured as INPUTS are said to be in a high impedance state.

Syntax: `pinMode(11, INPUT);`

Pins set as OUPUTS are said to be in a low impedance state and can supply 40mA of current to connected circuit or components.

Syntax: `pinMode(11, OUTPUT);`                      or  
`pinMode(11, INPUT);`

### **digitalRead**

This function is used to read the digital value of a pin. The result is either HIGH or LOW.

Syntax: `int value = digitalRead(11);`

### **digitalWrite**

`digitalWrite` sets the pins ON or OFF, by giving them either HIGH or LOW output.

Syntax: `digitalWrite(11,HIGH);`                      or  
`digitalWrite(11,LOW);`

### **analogRead**

This function reads the value from a specified analogue pin and returns a value between 0-1013. It only works on the 6 analogue pins from 0-5.

Syntax: `int value = analogRead(0);`

### **analogWrite**

This function is used to give an analog output. Analog outputs can only be given to pins 3,5,6,9,10 and 11 on the UNO. Values between 0-155 can be given to the analog pins.

Syntax: `analogWrite(3,value);`

#### **4.5 Other useful functions**

##### **delay()**

It pauses the program for a specified time . the arguments takes time duration in milliseconds.

Syntax: `delay(1000);`

##### **Serial.println(value)**

Prints the value passed into the brackets on to the serial Monitor:

Syntax: `int value 100;`

`Serial.println(value);`

##### **Serial.begin()**

This function opens up the USB port for serial communication and also sets the baud rate.

Syntax: `serial.begin(9600);`

##### **min(x,y)**

It returns the smaller of the two values passed into the function .

Syntax: `int x,y;`

`int smaller_num = min(x,y);`

##### **max(x,y)**

It returns the larger of the two values passed into the function.

Syntax: `int x,y;`

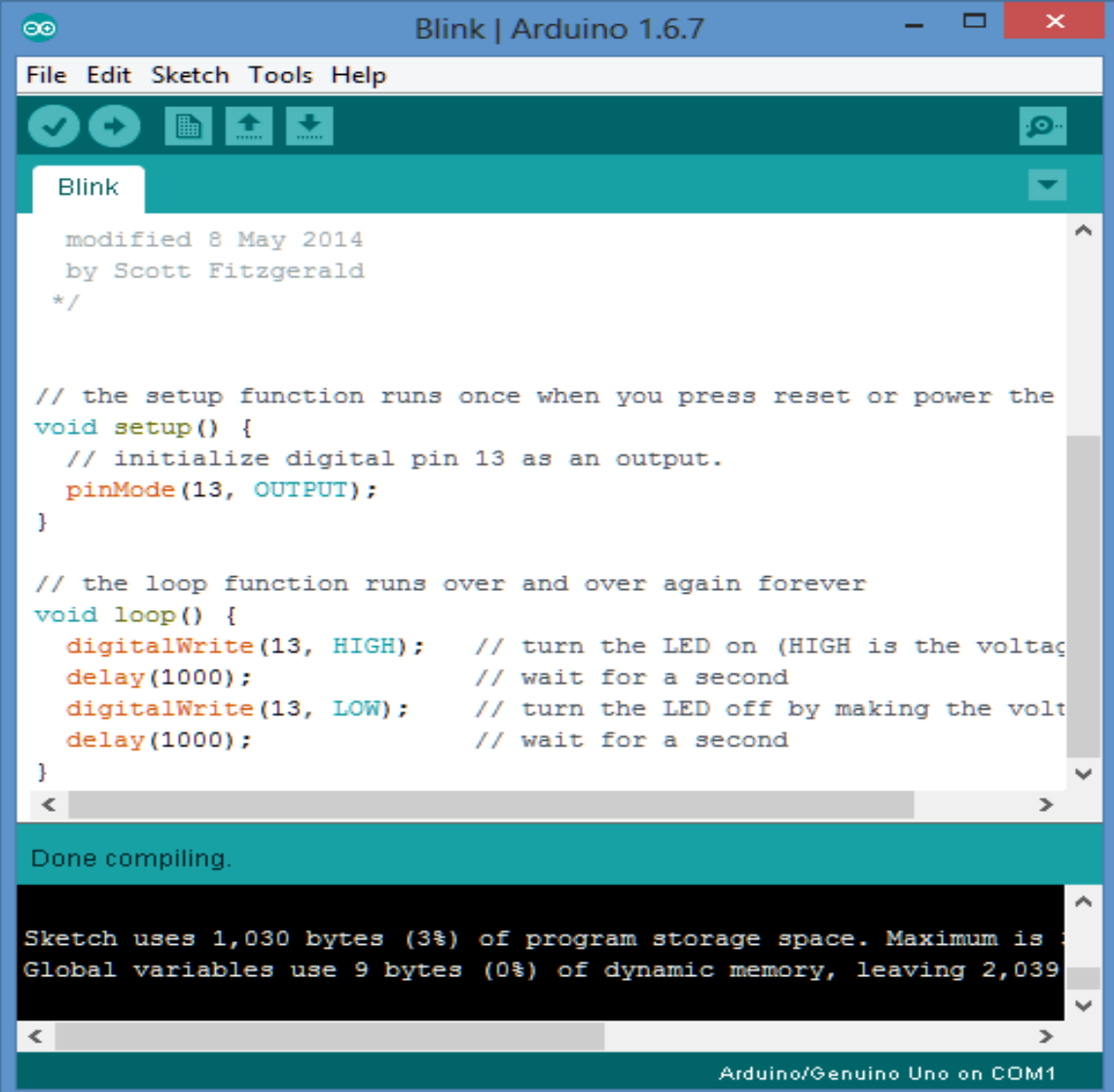
`int larger_num = max(x,y);`

We use the Arduino IDE on our computer (picture following) to create, open, and change sketches (Arduino calls programs sketches. We will use the two words interchangeably in this book.). Sketches define what the board will do. You can either use the buttons along the top of the IDE or the menu items.

Tools of the IDE: (from left to right, top to bottom) :

- Compile –Before your program code can be sent to the board, it needs to be converted into instructions that the board understands. This process is called compiling.
- Stop- This stops the compilation process.
- Create new Sketch- This opens a new window to create a new sketch.
- Open Existing Sketch - This loads a sketch from a file on your computer.
- Save Sketch-This saves the changes to the sketch you are working on.
- Upload to Board - This compiles and then transmits over the USB cable to your board.
- Serial Monitor – This tool is used for check the serially transmitting data.
- Sketch Editor - This is where you write or edit sketches.
- Text Console - This shows you what the IDE is currently doing and is also where error messages display if you make a mistake in typing your program(often called a syntax error).
- Line Number - This shows you what line number your cursor is on. It is useful since the compiler gives error messages with a line number.

#### 4.6 Program to blink led



The screenshot shows the Arduino IDE interface with the 'Blink' sketch loaded. The menu bar includes File, Edit, Sketch, Tools, and Help. The toolbar contains icons for checking, running, and saving. The sketch editor displays the following code:

```
modified 8 May 2014
by Scott Fitzgerald
*/

// the setup function runs once when you press reset or power the
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH); // turn the LED on (HIGH is the voltage)
  delay(1000);            // wait for a second
  digitalWrite(13, LOW);  // turn the LED off by making the voltage LOW
  delay(1000);            // wait for a second
}
```

Below the code editor, a status bar indicates 'Done compiling.' The bottom panel shows memory usage: 'Sketch uses 1,030 bytes (3%) of program storage space. Maximum is : Global variables use 9 bytes (0%) of dynamic memory, leaving 2,039'.

At the bottom right of the IDE, the text 'Arduino/Genuino Uno on COM1' is visible.

Figure: 4.5 program to blink led

## Chapter 5: Proteus Design suite

### 5.1. Water tank prototype:

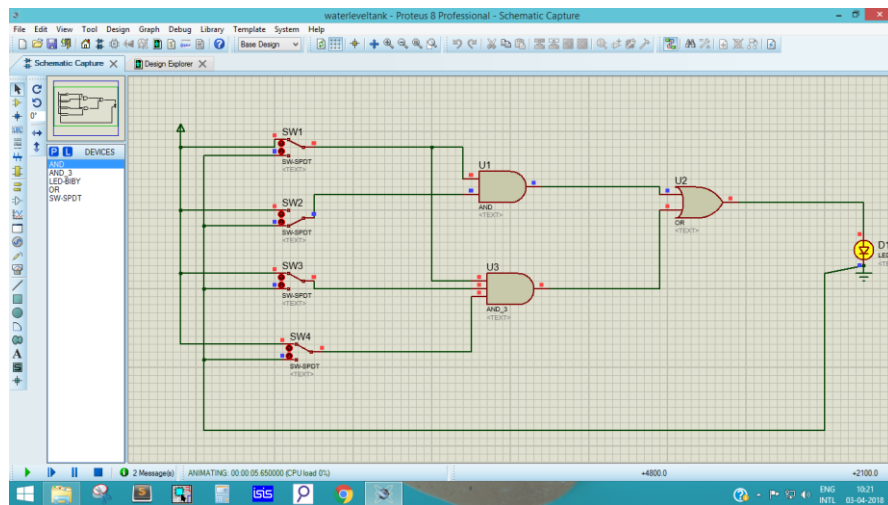


Figure 5.1. Water Tank

The control system from which water level of both tanks are observed with simultaneous water pump control is based on existing water level technology using the principle of ultrasound for level sensing. A prototype of the proposed microcontroller based water pump controller was tested.

### 5.2. Half wave rectifier prototype:

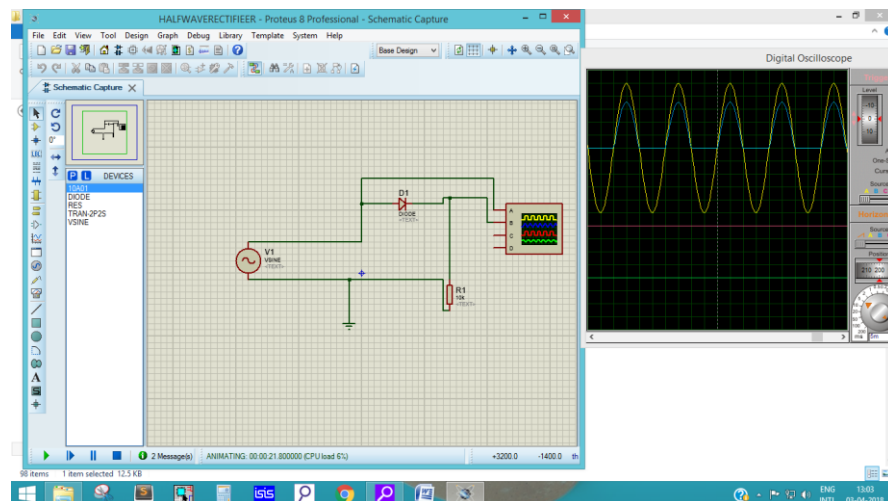


Figure 5.2. Half wave rectifier

The Half wave rectifier is a circuit, which converts an ac to dc voltage. A prototype was tested.

### 5.3. RL low pass filter prototype:

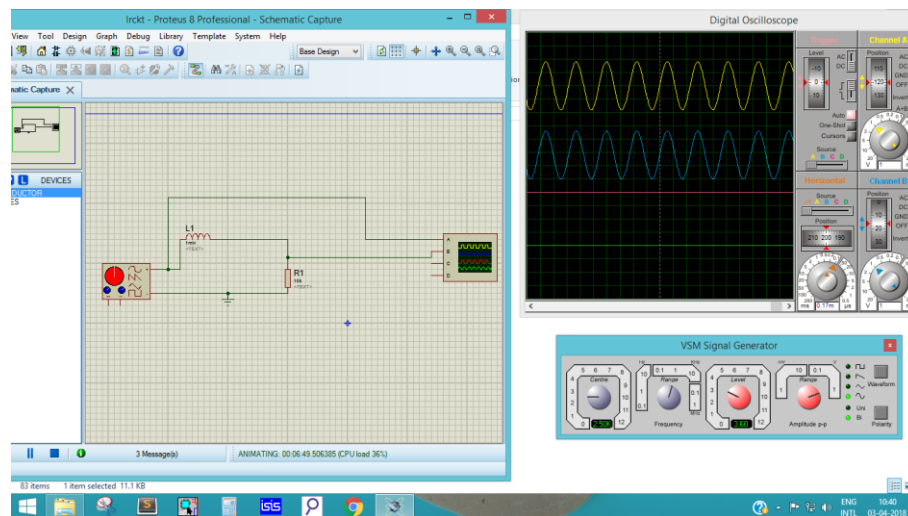


Figure 5.3. RL low pass filter

This is a low-pass filter implemented using a resistor and an inductor. The inductor passes lower frequencies, causing the output voltage to fluctuate more. Higher frequencies are blocked, and there is reduced current across the resistor, keeping the output voltage closer to ground.

### 5.4. RL high pass filter prototype:

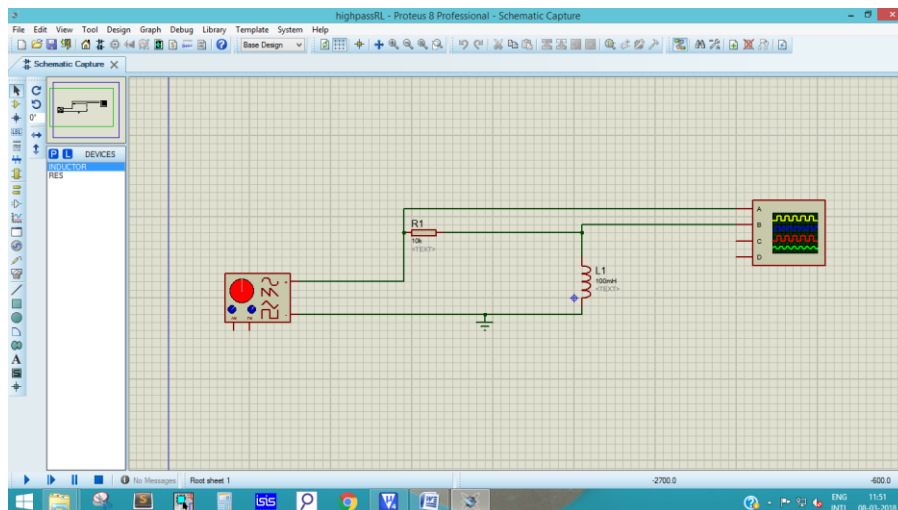


Figure 5.4. RL high pass filter

This is a high-pass filter implemented using a resistor and an inductor. The inductor passes lower frequencies, causing the voltage across it to be reduced and keeping the output voltage closer to ground. The inductor blocks higher frequencies, causing reduced current across the resistor and keeping the output voltage closer to the input voltage.



## 5.5. Full Adder Prototype:

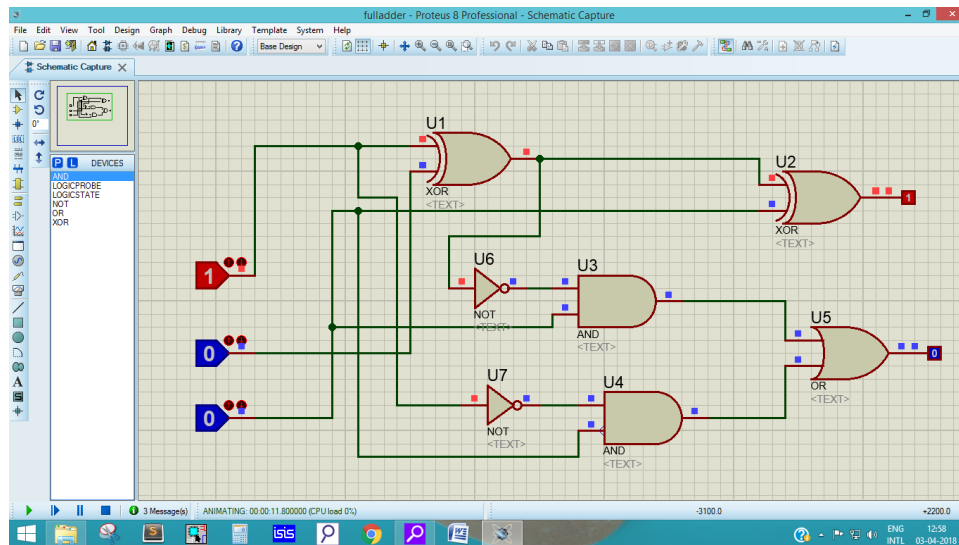


Figure 5.5. Full adder

Full Adder is the adder which adds three inputs and produces two outputs. The first two inputs are A and B and the third input is an input carry as C-IN. The output carry is designated as C-OUT and the normal output is designated as S which is SUM.

## 5.6. Keypad on virtual terminal:

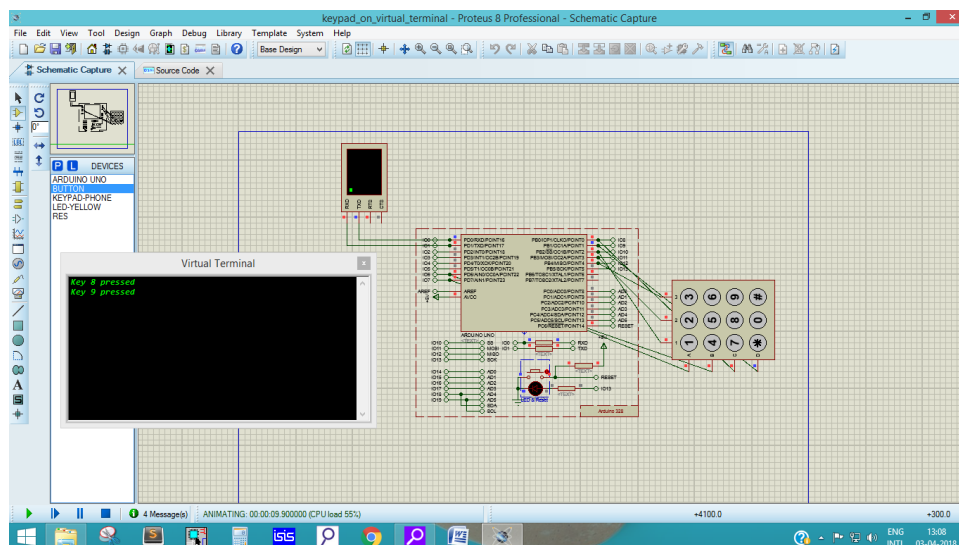


Figure 5.6. Keypad on virtual terminal

Processing payment through the mail, over the phone, or via fax is called MOTO credit card processing. MOTO processing allows you to complete payments without the customer or card being there in-person by manually entering the customer's credit card information with the keypad or virtual terminal.

## 5.7. LCD Interfacing with arduino:

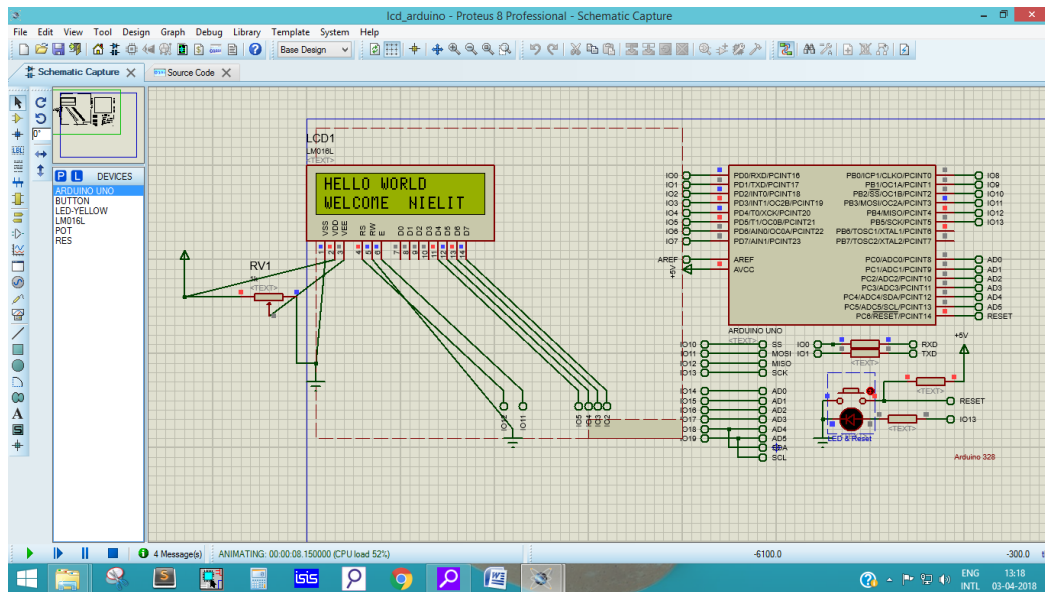


Figure 5.7. LCD Interfacing with arduino

LCD interfacing with different microcontrollers was tested and practical approach to various microcontrollers interfacing was held.

## Chapter 6: Introduction to ESP8266, Raspberry pi and MQTT

### 6.1 ESP 8266

#### 6.1.1 Introduction

- The ESP8266 WiFi Module is a self contained SOC with integrated TCP/IP protocol stack that can give any microcontroller access to your WiFi 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.

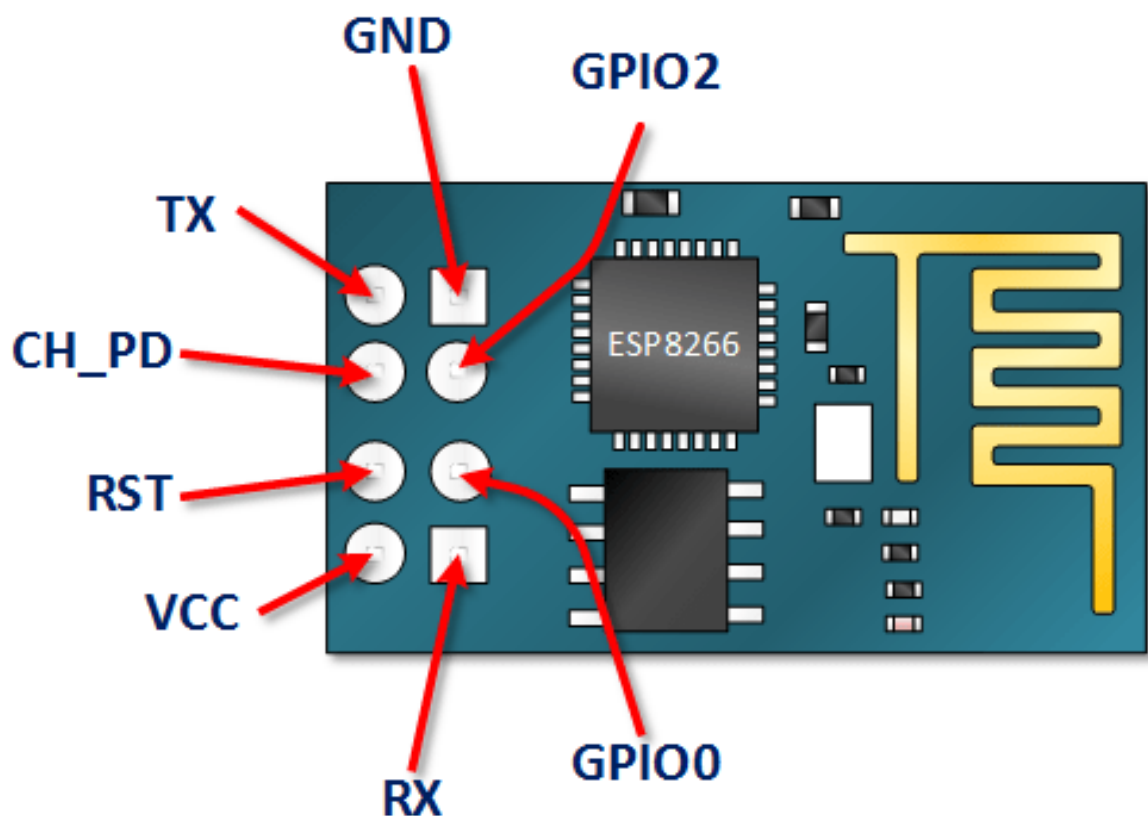


Figure 6.1.1: ESP8266 pin diagram

#### 6.1.2 Power Management In ESP8266

ESP8266EX is designed with advanced power management technologies and intended for mobile devices, wearable electronics and the Internet of Things applications. The low-power architecture operates in three modes: active mode, sleep mode and Deep sleep mode. ESP8266EX consumes about 20  $\mu$ A of power in Deep-sleep mode (with RTCClock still running) and less than 1.0 mA (DTIM=3) or less than 0.6 mA (DTIM=10) to stay connected to the access point.

#### **6.1.3 The power saving architecture operates in 3 modes:**

- active mode,
- sleep mode and
- deep sleep mode

#### **6.1.4 Important Specifications of ESP8266 are:**

- Certification : Wi-Fi Alliance
- Frequency Range : 2.4G ~ 2.5G (2400M ~ 2483.5M)
- CPU ( 32 bit, 26MHz-52MHz, 64KB instruction RAM, 64KB boot ROM, 96KB data )
- Operating Voltage : 2.5V ~ 3.6V
- Operating Current Average value: 80 mA
- Operating Temperature Range :  $-40^{\circ}\text{C}$  ~  $125^{\circ}\text{C}$
- Storage Temperature Range:  $-40^{\circ}\text{C}$  ~  $125^{\circ}\text{C}$
- Network Protocols : IPv4, TCP/UDP/HTTP/FTP
- User Configuration : AT Instruction Set, Cloud Server, Android/iOS App

#### **6.1.5 ESP 8266 Applications**

- Smart power plugs
- Home automation
- Mesh network
- Industrial wireless control
- Baby monitors
- IP Cameras
- Sensor networks
- Wearable electronics
- Wi-Fi location-aware devices
- Security ID tags

- Wi-Fi position system beacons

### 6.1.6 ESP With Arduino Uno R3

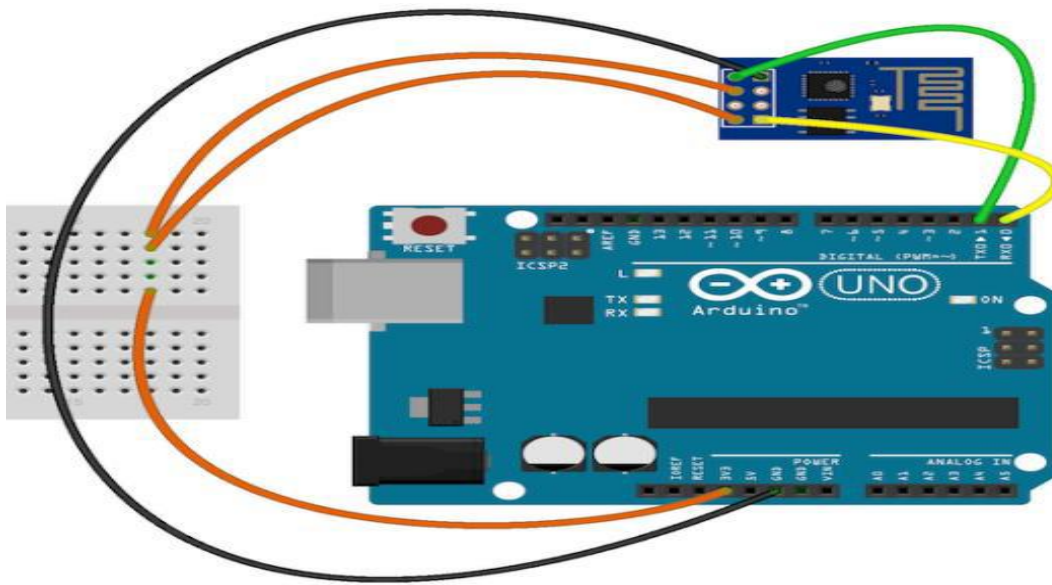


Figure 6.1.2: ESP with Arduino Uno

### 6.1.7 Interfacing Humidity Sensor:

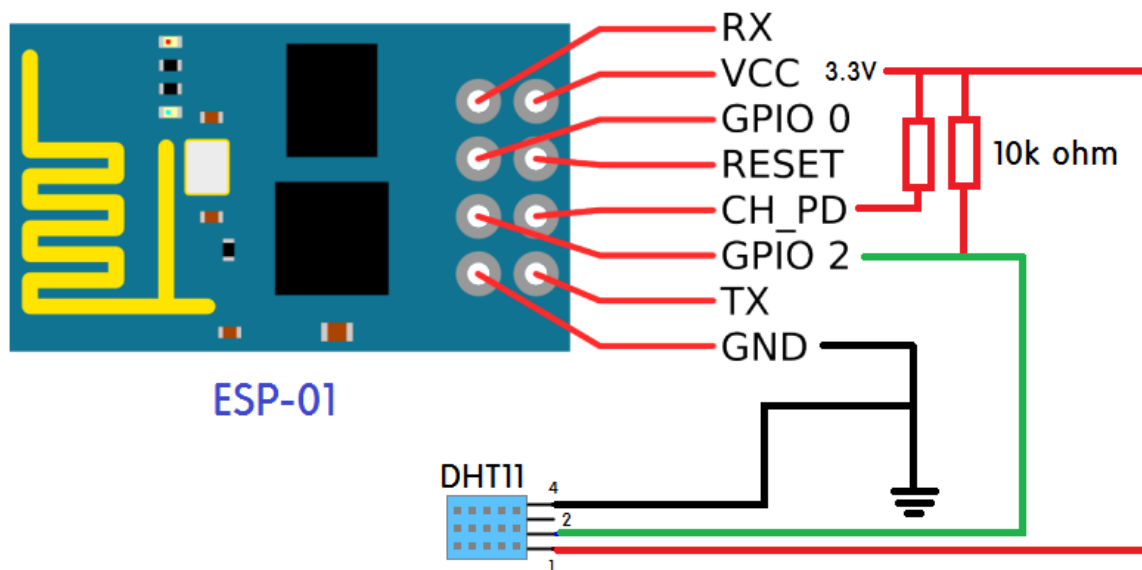


Figure 6.1.3: Interfacing Humidity Sensor

### 6.1.8 Code to test the working of ESP:

```
#include <SoftwareSerial.h>

SoftwareSerial WiFi_Serial(3,4); //3-Rx, 4-Tx

String str=""; boolean wf = false;

void setup()
```

```

{ Serial.begin(115200); // Begin serial monitor to receive 115200 bits per second (BAUD
RATE)
  WiFi_Serial.begin(9600); // Begin serial monitor to receive 9600 bits per second (BAUD
RATE)
  WiFi_Serial.println("AT+UART_DEF=9600,8,1,0,0"); delay(1000); // set WiFi Send/Receive
from 115200 bits per second (BAUD RATE) to 9600 bits per second
  WiFi_Serial.println("ATE0"); delay(200);
  WiFi_Serial.println("AT+CWQAP");delay(500); // Disconnect from previous network
connections
}
void WiFi_check(){

  WiFi_Serial.println("AT"); delay(500);// send a Attention command
  if (WiFi_Serial.available())
  { if (WiFi_Serial.find("OK")) // check with expected output
    { Serial.println("WIFI PLUGGED ON TO THE BOARD..!");
      WiFi_Serial.println("AT+CWMODE=1"); //set mode to client mode
      delay(500); wf = true;
    }
  }else{
    Serial.println("WIFI NOT PLUGGED..! ");
    Serial.println("PLUG IN YOUR WIFI CHIP"); wf =false;
  }
}
void connectWifi() // connect to wifi network
{ WiFi_Serial.println("AT+CWJAP?"); delay(2000);//check if WIFI connected to any WiFi
network
  if (WiFi_Serial.available())
  {
    if (WiFi_Serial.find("No AP")) //we receive reponse "No AP" when not connected to any
network
    {
      //Serial.println("NOT CONNECTED TO WIFI NETWORK");
      Serial.println("Trying to Connect to WiFi Network");
    }
  }
}

```

```

str="AT+CWJAP=\"asdf\", \"1234567890\"";
WiFi_Serial.println(str); // CWJAP followed by username and password
delay(2000);

if (WiFi_Serial.available())
{
  String RES_input = "";
  while (WiFi_Serial.available()) // read the data into a variable as long as the
  { RES_input += (char)WiFi_Serial.read(); }
  Serial.println("--->" + RES_input);
  if (WiFi_Serial.find("WIFI CONNECTED")){
    Serial.println("CONNECTED TO WIFI NETWORK..!");
  }else {
    Serial.println("ESP not CONNECTED TO WIFI NETWORK..!");
  }
}
}
}
}
}
void loop()
{
  Serial.println("Welcome to ESP8266 interfacing with cloud");

  WiFi_check() ;
  connectWifi();
  delay(1000);
}

```

## 6.2 Raspberry pi

A Raspberry Pi is a credit card-sized computer originally designed for education, inspired by 1981 BBC Micro. Creator Eben Upton's goal was to create a low-cost device that would improve programming skills and hardware understanding at pre- university level.

Computer and can provide all the expected abilities that implies, at a low power consumption level.



Figure 6.2.1: Raspberry Pi Board

#### 6.2.1.1 Different Types of Raspberry Pi Models

The different types of raspberry pi models are following:

- Raspberry Pi 1 model B
- Raspberry Pi 1 model A
- Raspberry Pi 1 model A+
- Raspberry Pi Zero
- Raspberry Pi 2

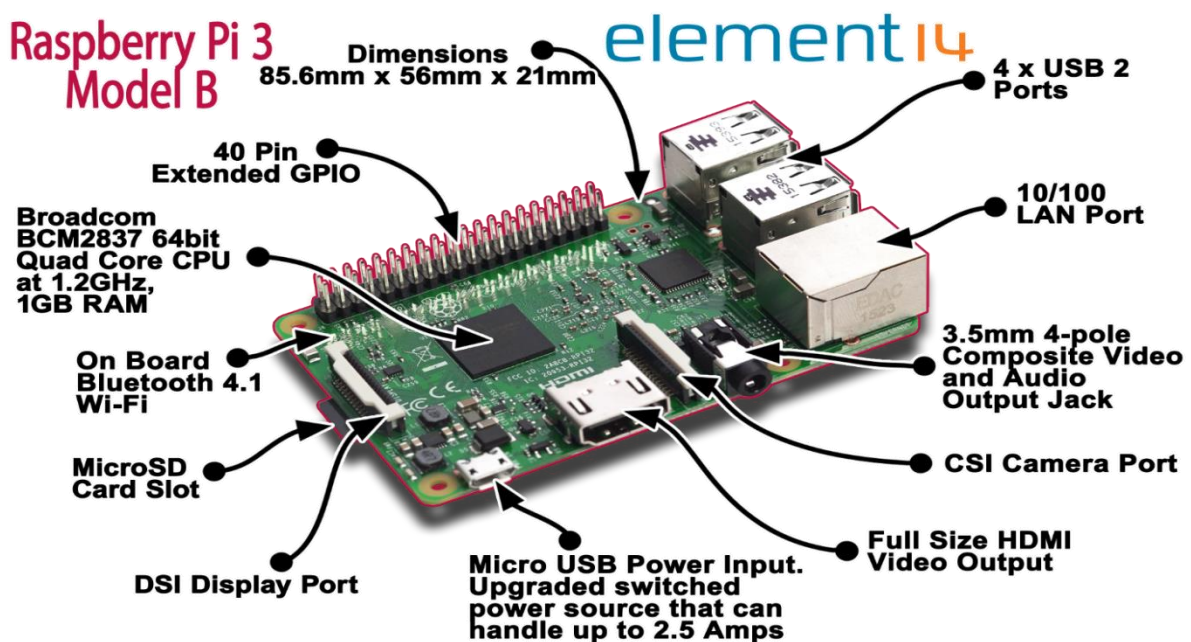


Figure 6.2.2: Raspberry pi 3 Model B

#### 6.2.2 Features:

- quad-core Cortex-A53 CPU at 1.2GHz with a VideoCore IV GPU clocked at 300-400MHz (3D clocked at 300MHz, video at 400MHz)



- 1GB of RAM,
- 802.11n wireless
- power consumption of 4W

	Raspberry Pi 3 Model B	Raspberry Pi 2 Model B	Raspberry Pi Model B+	Raspberry Pi Model A+
<b>Processor Chipset</b>	Broadcom BCM2837 64Bit Quad Core Processor powered Single Board Computer running at 1.2GHz	Broadcom BCM2836 32Bit Quad Core Processor powered Single Board Computer running at 900MHz	Broadcom BCM2835 32Bit SoC full HD multimedia applications processor	Broadcom BCM2835 32Bit SoC full HD multimedia applications processor
<b>GPU</b>	Videocore IV	Videocore IV	Videocore IV	Videocore IV
<b>Processor Speed</b>	QUAD Core @1.2 GHz	QUAD Core @900 MHz	Single Core @700 MHz	Single Core @700 MHz
<b>RAM</b>	1GB SDRAM @ 400 MHz	1GB SDRAM @ 400 MHz	512 MB SDRAM @ 400 MHz	256 MB SDRAM @ 400 MHz
<b>Storage</b>	MicroSD	MicroSD	MicroSD	MicroSD
<b>USB 2.0</b>	4x USB Ports	4x USB Ports	4x USB Ports	1x USB Port
<b>Max Power Draw/voltage</b>	2.5A @ 5V	1.8A @ 5V	1.8A @ 5V	1.8A @ 5V
<b>GPIO</b>	40 pin	40 pin	40 pin	40 pin
<b>Ethernet Port</b>	Yes	Yes	Yes	No
<b>WiFi</b>	Built in	No	No	No
<b>Bluetooth LE</b>	Built in	No	No	No

**Figure 6.2.3: Difference between various models of raspberry pi**

### 6.2.3 Preparing SD card for pi

- Download NOOBS(New Out Of Box Software) from the raspberrypi.org downloads
- Insert a (4 GB+) SD Card into your computer
- Format the disk (Windows)
- Extract the file you downloaded in Step 1
- Copy the files you just extracted to your SD Card

Irrespective of its size, Raspberry Pi is a powerhouse of a computer. It can drive HDMI displays, mouse, keyboard, camera – above all it runs full featured Linux distribution.

- Not only computer it is hardware prototyping tool.

- The Pi has bi-directional I/O pins, which can be used to drive LEDs, spin motors, or read button presses.

### 6.3 MQTT

MQTT is a binary client-server publish/subscribe messaging transport protocol. It is lightweight, open, simple, and easy to implement. Designed with a minimal protocol overhead, this protocol is a good choice for a variety of (M2M) and IoT applications. MQTT utilizes many characteristics of the TCP transport. So the minimum requirement for using MQTT is a working TCP stack, which is now available for even the smallest microcontrollers. The most recent version of MQTT is 3.1.1, which has many improvements over the first public MQTT release.

#### 6.3.1 History of MQTT

- MQTT was developed by Andy Stanford-Clark (IBM) and Arlen Nipper (Eurotech; now Cirrus Link) in 1999 for the monitoring of an oil pipeline through the desert.
- The goals were to have a protocol, which is bandwidth-efficient and uses little battery power, because the devices were connected via satellite link and this was extremely expensive at that time.

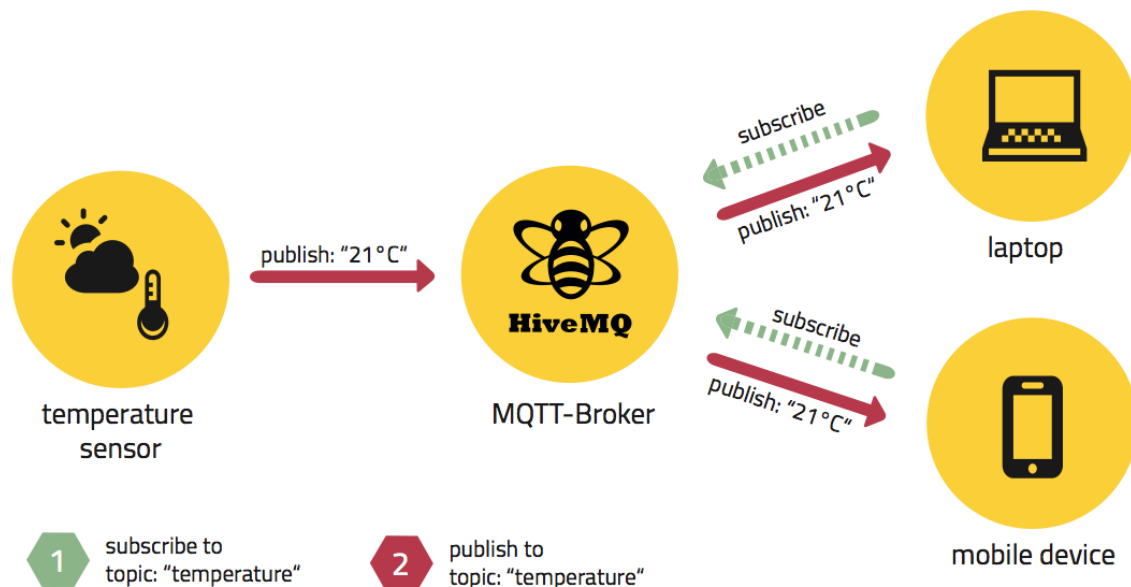


Figure 6.3.1: Working of MQTT

#### 6.3.2 Advantages of MQTT

- MQTT is a binary client-server publish/subscribe messaging transport protocol

- It is lightweight, open, simple, and easy to implement.
- Designed with a minimal protocol overhead,
- This protocol is a good choice for a variety of (M2M) and IoT applications
- MQTT utilizes many characteristics of the TCP transport.
- So the minimum requirement for using MQTT is a working TCP stack, which is now available for even the smallest microcontrollers.
- The most recent version of MQTT is 3.1.1, which has many improvements over the first public MQTT release, MQTT 3.1.

## **Chapter 7: IoT based Smart Parking System**

### **7.1 Introduction to IoT based Smart Parking System**

In recent times the concept of smart cities has gained great popularity. Thanks to the evolution of Internet of things the idea of smart city now seems to be achievable. Consistent efforts are being made in the field of IoT in order to maximize the productivity and reliability of urban infrastructure. Problems such as, traffic congestion, limited car parking facilities and road safety are being addressed by IoT. India is facing a lot of problems nowadays – lack of sufficient parking space. With families getting smaller and the total number of motor vehicles exceeding the total number of heads per family, the parking scenario is woefully falling short of the current requirements in the country. The situation is such that on any given working day approximately 40% of the roads in urban India are taken up for just parking the cars. The proposed Smart Parking system consists of an on-site deployment of an IoT module that is used to monitor and signalize the state of availability of each single parking space.

The device will get the data of the parking slot status (whether it is occupied or not) from the ultrasonic sensors and cameras that are present over there. These sensors send the data to the microcontroller and in turn the data will be processed and the status of parking slots will be displayed to the user on either MQTT dashboard. The data will also be sent to the cloud which can be integrated onto an Android App, so that the user can see the slots available directly from their mobile phones.

### **7.2 Objectives of IoT based Smart Parking System:**

- Enabling cities to develop fully integrated multimodal intelligent transportation systems with great security and efficiency.
- The system benefits of IoT based smart parking to avoid time wastage.
- Developing smart parking solutions within a city to solve pollution problems.

### **7.3 Project Architecture**

The IoT based parking system using IoT that you develop can be implemented in covered parks, open parks and also street side parking. The smart parking system will have a cloud service provider that provides cloud storage to store information about the parking status in the slots. There will be a centralized server which stores the information about the number of parking slots, availability status and also the parking time.

The functionalities of the components of the IoT based smart parking system are as follows:

#### **7.3.1 Software used in IoT based Smart Parking System**

### 7.3.1.1 Arduino IDE

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

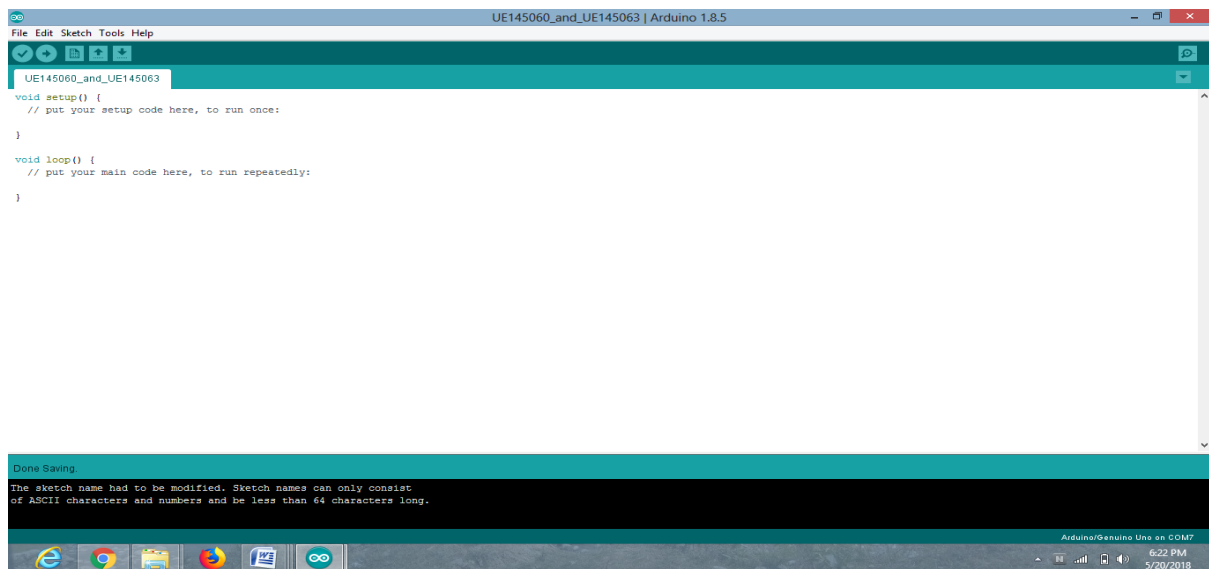


Figure 7.1 Arduino IDE Sketch

### 7.3.1.2. HTML

Hypertext Markup Language (HTML) is the standard markup language for creating web pages and web applications with Cascading Style Sheets (CSS) and JavaScript.

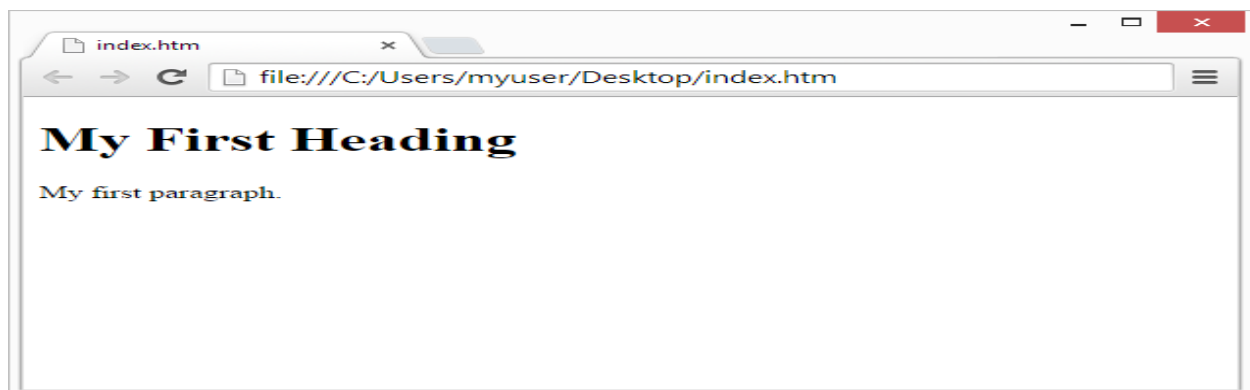


Figure 7.2 HTML webpage

### 7.3.1.3 MQTT lens

MQTT Lens is build on top of MQTT.js. All UI components are created with HTML and JavaScript, based on Polymer and Web Components. With one click – MQTT Lens can be installed via the Google Chrome Web Store, making it easy and convenient to get started.

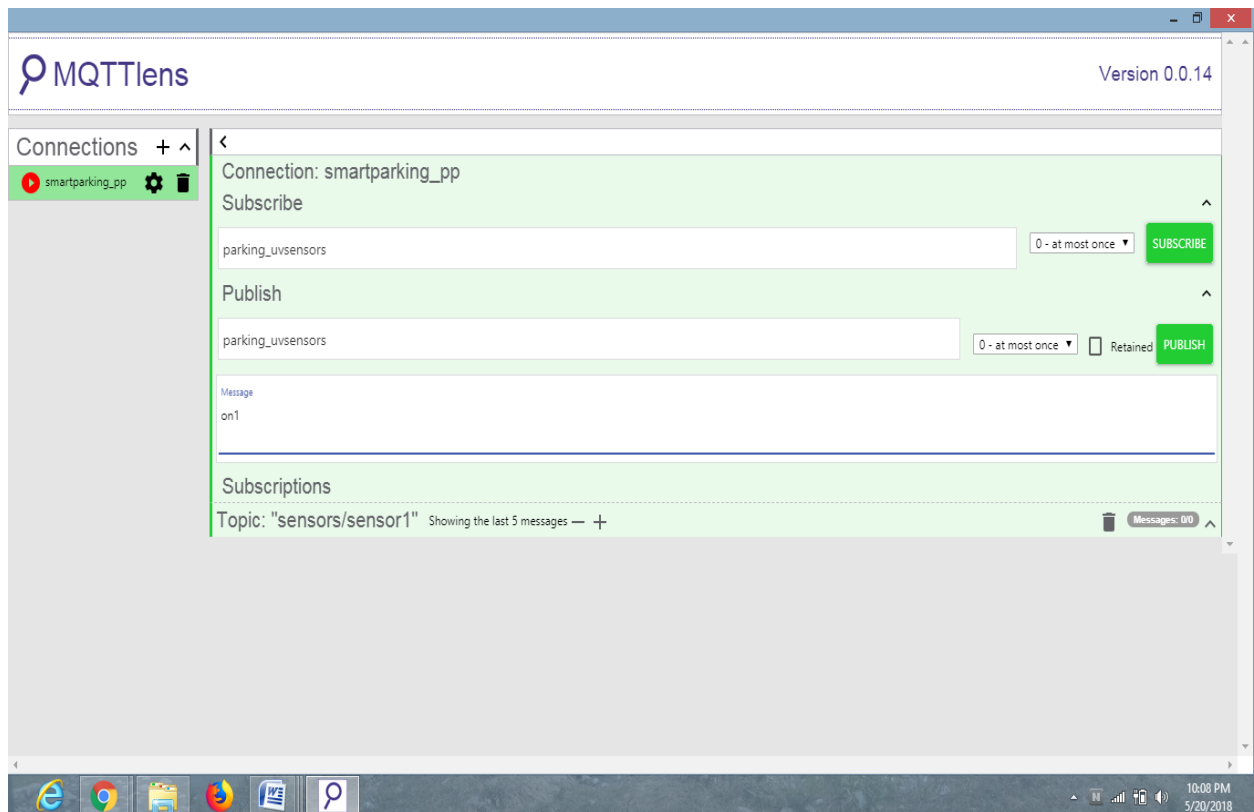
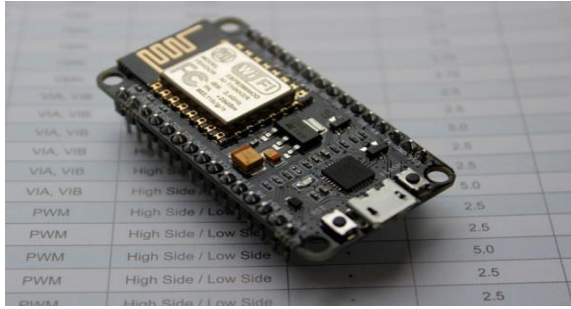


Figure 7.3 MQTT lens

## 7.3.2 Hardware used in IoT based Smart Parking System

### 7.3.2.1 NodeMCU

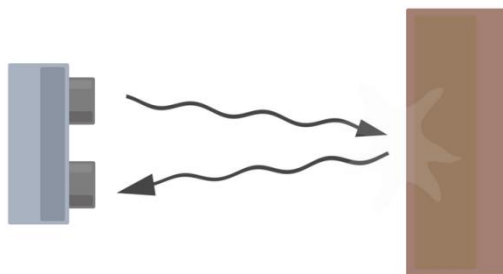
NodeMCU is an open source IoT platform. It includes firmware which runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which is based on the ESP-12 module. The term "NodeMCU" by default refers to the firmware rather than the development kits. It contains all crucial elements of the modern computer: CPU, RAM, networking (wifi), and even a modern operating system and SDK.



**Figure 7.4 NodeMCU**

### **7.3.2.2 Ultrasonic sensor**

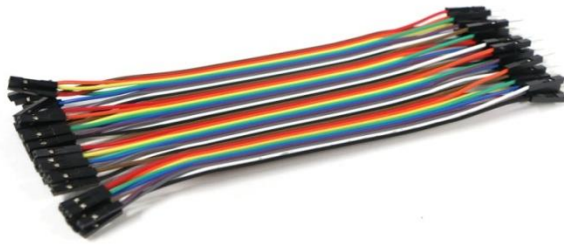
An Ultrasonic sensor is a device that can measure the distance to an object by using sound waves. It measures distance by sending out a sound wave at a specific frequency and listening for that sound wave to bounce back. By recording the elapsed time between the sound wave being generated and the sound wave bouncing back, it is possible to calculate the distance.



**Figure 7.5 Ultrasonic Sensor working**

### **7.3.2.3 Jump wire**

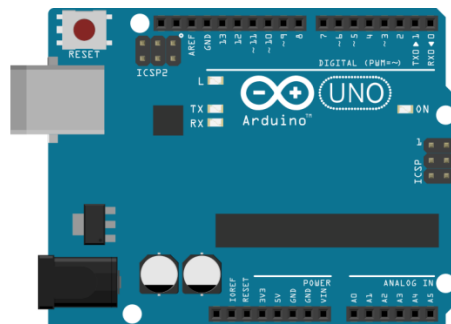
A jump wire (also known as jumper, jumper wire, jumper cable, DuPont wire, or DuPont cable – named for one manufacturer of them) is an electrical wire, or group of them in a cable, with a connector or pin at each end (or sometimes without them – simply "tinned"), which is normally used to interconnect the components of a breadboard or other prototype or test circuit, internally or with other equipment or components, without soldering.



**Figure 7.6 Jump wires**

#### **7.3.2.4 Arduino Microcontroller**

**Arduino Uno** is a microcontroller board based on the ATmega328P (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with an AC-to-DC adapter.

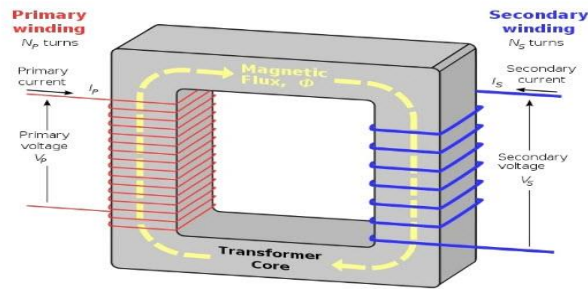


**Figure 7.7 Arduino Uno**

#### **7.3.2.5 Transformer**

A transformer is a static electrical device that transfers electrical energy between two or more circuits through electromagnetic induction. A varying current in one coil of the transformer produces a varying magnetic field, which in turn induces a varying electromotive force (emf) or "voltage" in a second coil. Power can be transferred between the two coils, without a metallic connection between the two circuits. Faraday's law of induction discovered in 1831 described this effect. Transformers are used to increase or decrease the alternating voltages in electric power applications.

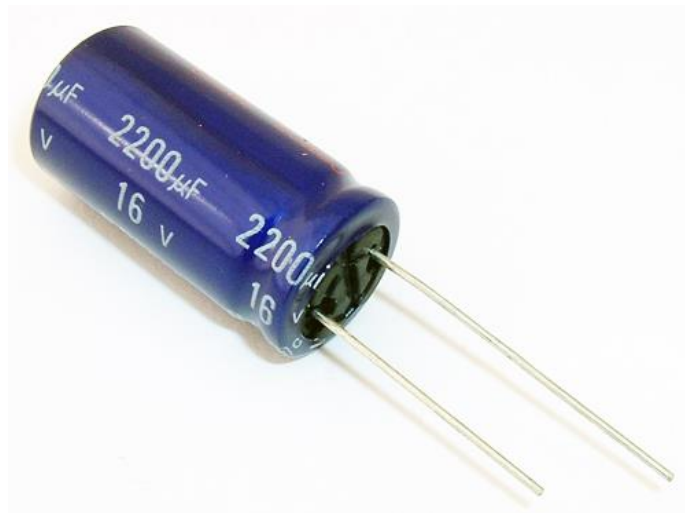




**Figure 7.8 Transformer**

### **7.3.2.6 Capacitor**

Capacitor is a device used to store an electric charge, consisting of one or more pairs of conductors separated by an insulator.



**Figure 7.9 Capacitor 2200uf 16v**

### **7.3.2.7 Voltage regulator 7805**

A regulated power supply is very much essential for several electronic devices due to the semiconductor material employed in them have a fixed rate of current as well as voltage. The device may get damaged if there is any deviation from the fixed rate. The AC power supply gets converted into constant DC by this circuit. By the help of a voltage regulator DC, unregulated output will be fixed to a constant voltage. The circuit is made up of linear voltage regulator 7805 along with capacitors and resistors with bridge rectifier made up from diodes.

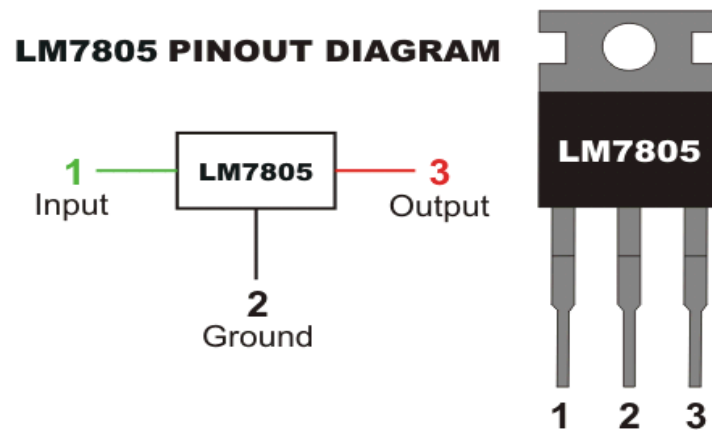


Figure 7.10 LM7805 Pinout Diagram

### 7.3.2.8 LD33V

It is used to convert 5V to 3.3V regulated voltage with and without decoupling capacitors. LD33V is used to supply power to NodeMCU which takes up 3.3V in order to function well. In this way both ultrasonic sensor and NodeMCU which is used as a microcontroller works and functions well with the insight of fulfilling the aim to achieve Smart Parking System.

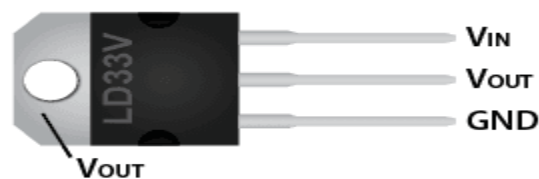


Figure 7.11 LD33V Regulator

### 7.3.2.9 Diode

A diode is a two-terminal electronic component that conducts current primarily in one direction; it has low resistance in one direction, and high resistance in the other.

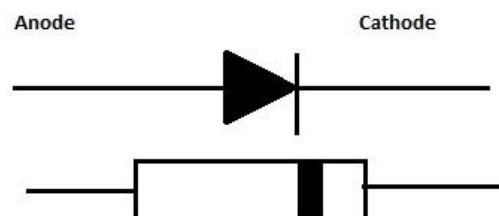


Figure 7.12 Diode

## 7.4 Project Implementation and working

In country like India, cities are not planned due to which every next individual face problem of parking their vehicles whenever they go out. Here, smart parking technology enables the customer to find spaces quickly and easily. Whether the requirement is on-street or off-street, smart parking can improve the service. In order to execute this I have come out with a solution where a device called ultrasonic sensor will be used in the walls of parking area so that the issues which arise due to parking problems like noise pollution and air pollution is solved.

The reason for using ultrasonic sensor and not any other sensor is that it is easy to implement on the walls like other electrical appliances without the fear of any damage being caused to it due to rain, storm or any other natural calamity. Ultrasonic sensor mounted on the walls detect whether the slot in the requested area is vacant or not and the person can make the estimations about their arrival and departure time. Ultrasonic sensor is connected to nodeMCU which is used to publish data on the internet. NodeMCU is connected to local wifi hotspot and when sensor senses any vehicle, slot is occupied and the respective data is published on web using a protocol called MQTT which is programmed by using HTML and JavaScript language.

The link for the webpage is <http://arshbir.com/sp/>



Figure 7.13 Project webpage

In my system prototype I have used 3 parking slots.

Symbols used:

- If the car is parked then a car is used to demonstrate the scenario of an occupied slot.
- If the area is vacant then it specifies vacant slot.

## 7.5 Codes used in IoT based Smart Parking System

### 5.5.1 HTML Code

```
<!DOCTYPE html>

<html>

<head>

<title>Smart Parking</title>

<head>

<title>Bootstrap Example</title>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<linkrel="stylesheet"href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.c
ss"><script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>

</head> </head>

<body>

<style>

h1 {

    text-align: center;

    color:blue ;

}

</style>



<h1><p><font size="9" color="blue">Welcome to Smart Parking System!</font></p></h1>

<body background="black.png"> <!-- import jquery library -->

<script src="https://code.jquery.com/jquery-3.1.0.min.js"></script>

<!-- import paho MQTT library -->
```

```

<scriptsrc="https://cdnjs.cloudflare.com/ajax/libs/paho-mqtt/1.0.1/mqttws31.min.js"
type="text/javascript"></script>

<!-- our javascript --> <script src="app.js"></script>

<div class="row" align="middle">

<div class="col-sm-4">

<p><font color="red"><center>Parking 1</center></font></p>

<img id="myDIV1" src=car.png style="background-color:lightgray;" height=250 width=450
align=center>

<img id="myDIV1v" src=vacantp.png height=250 width=450 align=center
style="display:none;">

</div>

<div class="col-sm-4">

<p><font color="red"><center>Parking 2</center></font></p>

<img id="myDIV2" src=car.png style="background-color:lightgray;" height=250 width=450
align=center>

<img id="myDIV2v" src=vacantp.png height=250 width=450 align=center
style="display:none;">

</div>

<div class="col-sm-4">

<p><font color="red"><center>Parking 3</center></font></p>

<img id="myDIV3" src=car.png style="background-color:lightgray;" height=250 width=450
align=center>

<img id="myDIV3v" src=vacantp.png height=250 width=450 align=center
style="display:none;">

</div>

</div>

<style>

h2 {

text-align: center;

color:purple ;

}

</style>

```

```

<h2>Number of Parking slots = 3</h2>

<h4><center><font color="blue"><font color="red">Instructions:</font> 1) Parked Car: Slot is
occupied 2) Vacant: Slot is unoccupied</font></center></h4>

<p><font color="red"><center>Thanks, please visit again!</center></font></p>

<meta name="viewport" content="width=device-width, initial-scale=1">

<style>

.footer {
position: fixed;
left: 0;

    bottom: 0;
    width: 100%;
    color: yellow;
    text-align: center;
}

</style>

</head>

<body>

<div class="footer">

    <p>Website belongs to Parmeet Singh(UE145060) and Prabhakar Kumar Giri(UE145063)</p>

</div>

```

### 5.5.2 app.js Code

```

// Generate a new random MQTT client id on each page load

var MQTT_CLIENT_ID = "client@sp_pp"+Math.floor((1 + Math.random()) *
0x1000000000000).toString(16);

// Create a MQTT client instance

var MQTT_CLIENT = new Paho.MQTT.Client("iot.eclipse.org", 80, "/ws",
MQTT_CLIENT_ID);

// Tell the client instance to connect to the MQTT broker

```

```

MQTT_CLIENT.connect({ onSuccess: myClientConnected });

// Tell MQTT_CLIENT to call myMessageArrived(message) each time a new message arrives
MQTT_CLIENT.onMessageArrived = myMessageArrived;

// This is the function which handles subscribing to topics after a connection is made
function myClientConnected() {
    MQTT_CLIENT.subscribe("parking_uvssensors");
}

// This is the function which handles received messages
function myMessageArrived(message) {
    // Get the payload
    var messageBody = message.payloadString;
    var x1 = document.getElementById("myDIV1");
    var x1v = document.getElementById("myDIV1v");
    var x2 = document.getElementById("myDIV2");
    var x2v = document.getElementById("myDIV2v");
    var x3 = document.getElementById("myDIV3");
    var x3v = document.getElementById("myDIV3v");

    if (messageBody == 'on1'){
        x1.style.display = "block";
        x1v.style.display = "none";
    }
    if (messageBody == 'off1'){
        x1.style.display = "none";
        x1v.style.display = "block";
    }
}

```

```

if (messageBody == 'on2'){
    x2.style.display = "block";
    x2v.style.display = "none";
}
if (messageBody == 'off2'){
    x2.style.display = "none";
    x2v.style.display = "block";
}
if (messageBody == 'on3'){
    x3.style.display = "block";
    x3v.style.display = "none";
}
if (messageBody == 'off3'){
    x3.style.display = "none";
    x3v.style.display = "block";
}

```

// Create a new HTML element wrapping the message payload

```
var messageHTML = $("<p> - "+messageBody+"</p>");
```

// Insert it inside the ``id=updateMe`` element above everything else that is there

```
$("#updateMe").prepend(messageHTML);
```

```
};
```

// This is the function which handles button clicks

```
function myButtonWasClicked() {
```

// create a new MQTT message with a specific payload

```
var mqttMessage = new Paho.MQTT.Message("Hello from website");
```



```

// Set the topic it should be published to
mqttMessage.destinationName = "sensors/sensor1";

// Publish the message
MQTT_CLIENT.send(mqttMessage);

// Set the topic it should be published to
mqttMessage.destinationName = "sensors/sensor2";

// Publish the message
MQTT_CLIENT1.send(mqttMessage);

}

```

### 5.5.3 NodeMCU Code

```

#include <ESP8266WiFi.h>
#include <PubSubClient.h>

void setup_wifi();

void callback(char* topic, byte* payload, unsigned int length);

void reconnect();

int duration1, distance1, duration2, distance2, duration3, distance3;

const char* ssid = "demo";

const char* password = "smartparking";

const char* mqtt_server = "iot.eclipse.org";

int const trigPin1 = 5;

int const echoPin1 = 4;

int const trigPin2 = 0;

int const echoPin2 = 2;

int const trigPin3 = 14;

int const echoPin3 = 12;

```

```

WiFiClient espClient;

PubSubClient client(espClient);

void setup() {

  pinMode(trigPin1, OUTPUT); // trig pin will have pulses output
  pinMode(echoPin1, INPUT); // echo pin should be input to get pulse width
  pinMode(trigPin2, OUTPUT); // trig pin will have pulses output
  pinMode(echoPin2, INPUT); // echo pin should be input to get pulse width
  pinMode(trigPin3, OUTPUT); // trig pin will have pulses output
  pinMode(echoPin3, INPUT); // echo pin should be input to get pulse width
  Serial.begin(115200);
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
  reconnect();
}

void setup_wifi(){

  delay(10);
  // We start by connecting to a WiFi network
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);

```

```

    Serial.print(".");
}

Serial.println("");
Serial.println("WiFi connected");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());
}

void callback(char* topic, byte* payload, unsigned int length) {
    Serial.print("Message arrived [");
    Serial.print(topic);
    Serial.print("] ");
    for (int i = 0; i < length; i++) {
        Serial.print((char)payload[i]);
    }
    Serial.println();
}

void reconnect() {
    // Loop until we're reconnected
    while (!client.connected()) {
        Serial.print("Attempting MQTT connection...");
        // Attempt to connect
        if (client.connect("client@sp_pp")) {
            Serial.println("connected");
            // Once connected, publish an announcement...

            digitalWrite(trigPin1, HIGH);

```

```

delay(1);
digitalWrite(trigPin1, LOW);
// Measure the pulse input in echo pin
duration1 = pulseIn(echoPin1, HIGH);
// Distance is half the duration divided by 29.1 (from datasheet)
distance1 = (duration1/2) / 29.1;
Serial.print("distance1 : ");Serial.println(distance1);
int pub_status;
if (distance1 >= 2 && distance1 <= 25) {
    // object detected
    pub_status = client.publish("parking_uvssensors", "on1");
    Serial.print (" publish sensor1 on1 status " );
    if (pub_status==1) Serial.println("successful");
    else Serial.println("Not successful");
} else {
    pub_status =client.publish("parking_uvssensors", "off1");
    Serial.print(" publish sensor1 off1 status: " );
    if (pub_status==1) Serial.println("successful");
    else Serial.println("Not successful");

}

////////// for second sensor
digitalWrite(trigPin2, HIGH);
delay(1);
digitalWrite(trigPin2, LOW);
// Measure the pulse input in echo pin
duration2 = pulseIn(echoPin2, HIGH);
// Distance is half the duration divided by 29.1 (from datasheet)
distance2 = (duration2/2) / 29.1;

```

```

Serial.print("distance2 : ");Serial.println(distance2);
if (distance2 >= 2 && distance2 <= 25) {
    // object detected

    pub_status = client.publish("parking_uvssensors", "on2");
    Serial.print (" publish sensor1 on2 status " );
    if (pub_status==1) Serial.println("successful");
    else Serial.println("Not successful");
} else {
    pub_status =client.publish("parking_uvssensors", "off2");
    Serial.print(" publish sensor1 off2 status: " );
    if (pub_status==1) Serial.println("successful");
    else Serial.println("Not successful");

}

////////// for third sensor

    digitalWrite(trigPin3, HIGH);
    delay(1);
    digitalWrite(trigPin3, LOW);
    // Measure the pulse input in echo pin
    duration3 = pulseIn(echoPin3, HIGH);
    // Distance is half the duration divided by 29.1 (from datasheet)
    distance3 = (duration3/2) / 29.1;
    Serial.print("distance3 : ");Serial.println(distance3);
    if (distance3 >= 2 && distance3 <= 25) {
        // object detected

        pub_status = client.publish("parking_uvssensors", "on3");
        Serial.print (" publish sensor1 on3 status " );
        if (pub_status==1) Serial.println("successful");
        else Serial.println("Not successful");
    }

```

```

    } else {
        pub_status =client.publish("parking_uvssensors", "off3");
        Serial.print(" publish sensor1 off3 status: " );
        if (pub_status==1) Serial.println("successful");
        else Serial.println("Not successful");

    }

    client.disconnect();
    delay(5000);
    // ... and resubscribe
    //client.subscribe("sensors/sensor1");
} else {
    Serial.print("failed, rc=");
    Serial.print(client.state());
    Serial.println(" try again in 5 seconds");
    // Wait 5 seconds before retrying
    delay(5000);
}
}
}

void loop() {

    if (!client.connected()) {
        reconnect();
    }
    client.loop();
}

```

## **Chapter 8: IoT based Smart Dustbin**

### **8.1 Introduction to Iot based Smart Dustbin**

The ultimate need of a developing nation like India is “Smart City”. Smart cities include managing air pollution, water pollution, noise pollution and other biological factors which also includes waste management. But as we all know, these wastes are not managed the way they should be due to which various health problems arise. In virtue of this I have come out with a solution to manage wastes by using the evolving technology Internet of things where the wastes can be handled with much ease in no time. This design designates a technique in which the garbage level could be checked at regular intervals which would prevent the undesirable overflow of the bin.

### **8.2 Objectives of Iot based Smart Dustbin**

- In order to handle waste management with greater ease.
- To avoid wasting petrol and diesel for making regular checks of the bins.
- To enhance healthy and hygienic environment.
- Time saving operation and automatic process.
- To make proper use of dustbin

### **8.3 System Architecture**

#### **8.3.1 Software used in the project are as follows**

##### **8.3.1.1 Arduino IDE**

Arduino IDE (Integrated Development Environment) software is used to write and upload codes from the software to physical board. The language used in Arduino IDE is C++. Most of the libraries are written in C/C++. It runs on Windows, Mac and Linux. The software can be used with any arduino board.

##### **8.3.1.2 ThingSpeak**

ThingSpeak™ is an IoT analytics platform service that allows aggregating, visualizing and analyzing live data streams in the cloud.

ThingSpeak provides instant visualization of data posted by the devices to ThingSpeak. ThingSpeak is often used for prototyping and proof of concept IoT systems that require analytics.

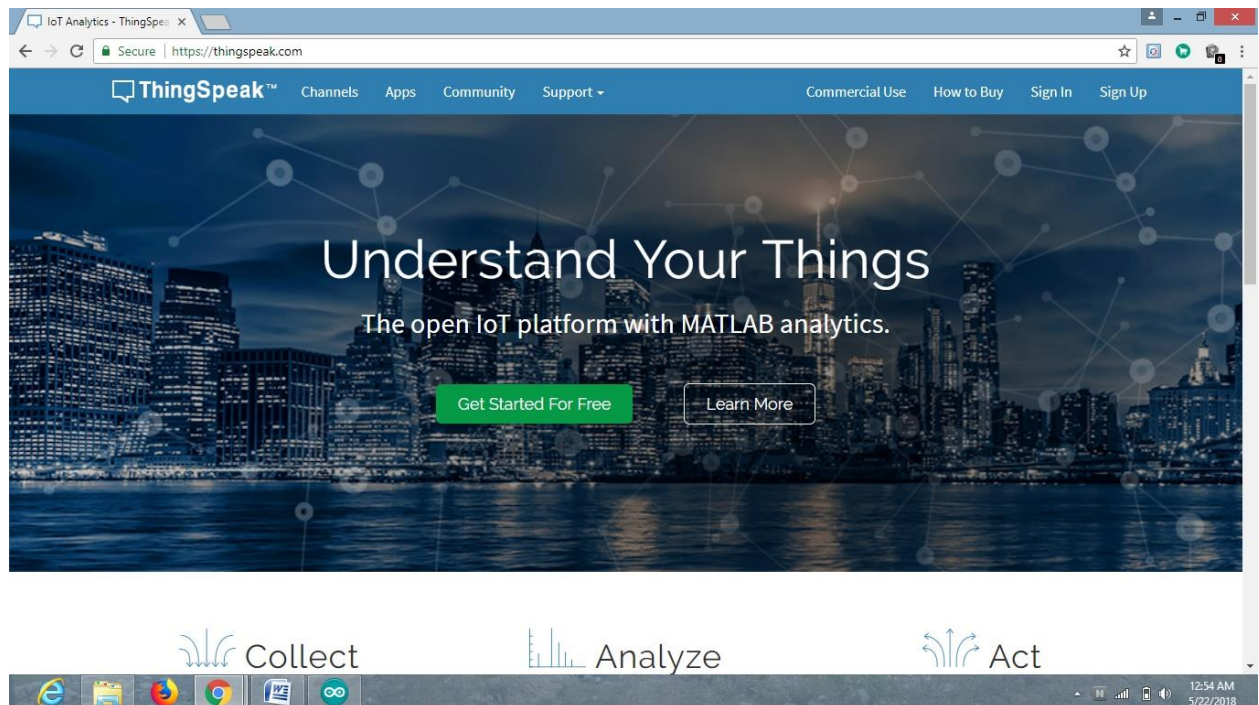


Figure 8.1 View of IoT based ThingSpeak

## 8.3.2 Hardware used in Project

### 8.3.2.1 Arduino Microcontroller

The Arduino Uno board is a microcontroller based on the ATmega328. It has 14 digital input/output pins in which 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.

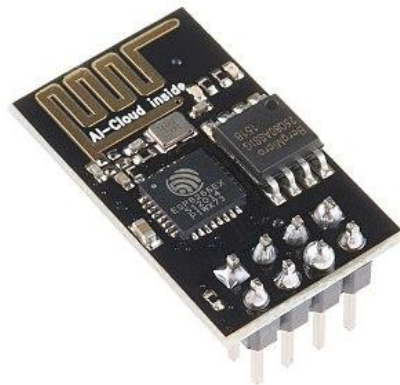


Figure 8.2 Arduino Uno

### 8.3.2.2 ESP8266



The ESP8266 is a low-cost Wi-Fi microchip with full TCP/IP stack and microcontroller capability produced by Shanghai-based Chinese manufacturer, Espressif Systems.



**Figure 8.3 ESP8266**

#### **8.3.2.3 Ultrasonic sensor**

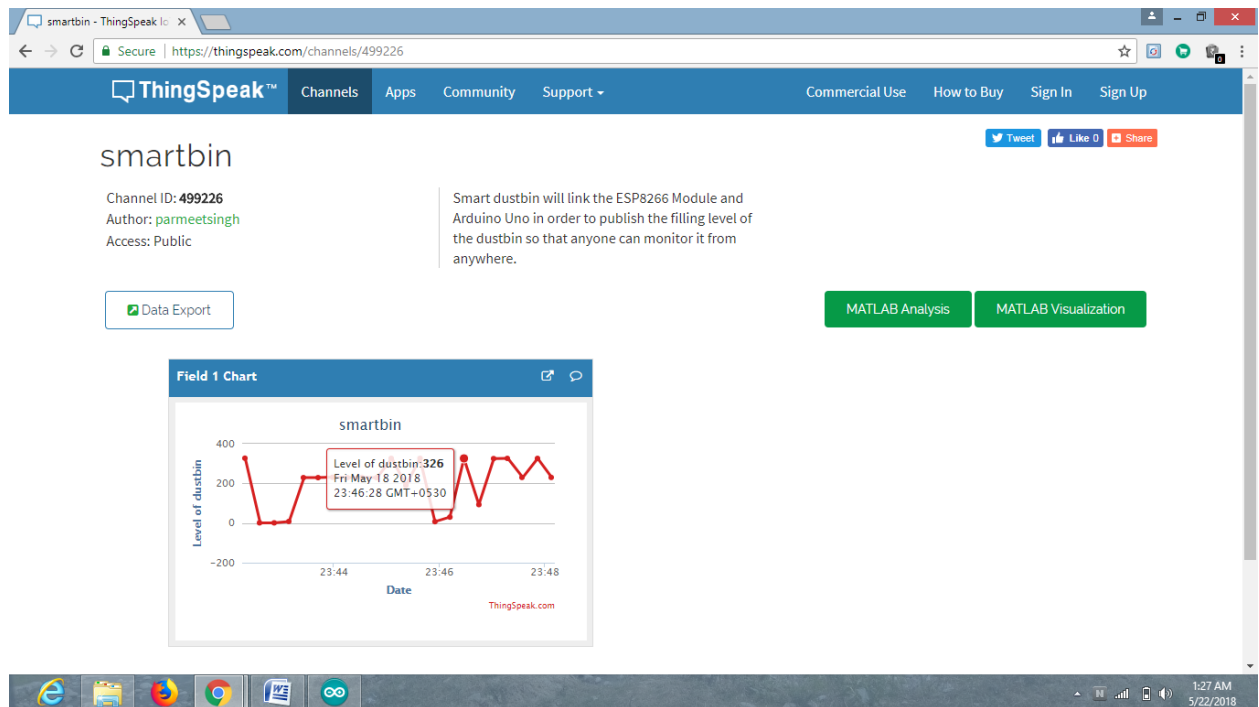
*Ultrasonic sensor measures distance using ultrasonic waves.* The sensor head emits the ultrasonic wave and receives the wave reflected from the target. Ultrasonic sensor measures the distance from the target by means of time elapse between emission and reception.

#### **8.3.2.4 Jump wires**

*Jumper wires* are used for making connections between items on your breadboard and your Arduino's header pins.

### **8.4 Project Implementation and working**

Waste management is one the key factors of burning concept “Smart City” and in order to achieve this aim I have focused on one of the technologies – Internet of Things. This is implemented by mounting each bin with an ultrasonic sensor which in turn sends the ultrasonic wave and finds the filled length of the bin and helps to monitor the garbage level from anywhere in no time. Ultrasonic sensor is made part of the Arduino Uno microcontroller which is programmed with the help of wifi module ESP8266 to achieve internet facilities. Garbage level is monitored every 2 minutes and the details are sent to the cloud ThinksSpeak.com where the garbage level is labeled along with the date. In this way waste managers can save their time and petrol and collect garbage without any time delay and take the city one step ahead in being “The Smart City”.



**Figure 8.4 Projects Monitoring on Cloud**

## 8.5 Code used in the project

```
#include<SoftwareSerial.h>

#define SSIDE "demo"

#define PASS "smartbin";

# define IP "184.106.153.149"; //IP of THINGSPEAK.COM

String GET="GET /update?api_key=4GGME5I81USOZVCK&field1="; //PATH AGFTER
GET. GET THE PATH FROM THINGSPEAK.COM API KEY REQUESTS. THIS ONE IS TO
UPDATE CHANNEL.

SoftwareSerial espmonitor(3,4); //RX TX

const int trigPin = 9;

const int echoPin = 10;

const int MaxHeight= 17;

float binStatus=0 ;

long duration;

int distance, oldDistance;

boolean firstTime=HIGH ;
```

```

void setup()
{
  espmonitor.begin(9600);
  Serial.begin(9600);
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input
  espmonitor.println("AT");
  Serial.println("AT");
  delay(5000);
  if (espmonitor.find("OK"))
  {
    Serial.println(" RECEIVED OK ");
  }
  else
  {
    Serial.println("AT SYSTEM NOT WORKING PROPERLY");
  }
  connectWifi();
}

void connectWifi()
{
  espmonitor.println("AT+CWMODE=1"); //1-StationMode(CLIENT) 2-HOST(APMODE) 3-
DUAL MODE
  Serial.println("AT+CWMODE=1");
  delay(2000);
  String cmd="AT+CWJAP=\"";
  cmd+=SSID;
  cmd+="\", \"";
  cmd+=PASS;

```

```

cmd+="\"";
espmonitor.println(cmd);
Serial.println(cmd);
delay(5000);
if(espmonitor.find("OK"))
{
    Serial.println("WIFI CONNECTED");
}
else
{
    Serial.println("ERROR : WIFI NOT CONNECTED");
}
}

void updateTemp(String distance)
{
    String cmd="AT+CIPSTART=\"TCP\",\"";
    cmd+=IP;
    cmd+="\",80";
    espmonitor.println(cmd);
    Serial.println(cmd);
    delay(2000);
    if(espmonitor.find("OK"))
    {

        Serial.println("TCP CONNECTION ESTABLISHED");
    }
    if(espmonitor.find("Error"))
    {

```

```

Serial.println("TCP CONNECTION NOT ESTABLISHED");
}

cmd=GET;

cmd+=distance;

cmd+="\r\n";

espmonitor.print("AT+CIPSEND=");

espmonitor.println(cmd.length());


delay(5000);    //

if(espmonitor.find(">"))
{
    Serial.println(">");
    espmonitor.print(cmd);
    Serial.println(cmd);

}

else

{
    espmonitor.println("AT+CIPCLOSE");
    Serial.println("AT+CIPCLOSE");
}

if(espmonitor.find("OK"))
{
    Serial.print("RECEIVED OK");
}

else

{
    Serial.print ("ERROR RECEIVED");
}

```

```

    }
}

void loop() {
    char buffer[10];
    String temper=String(distance);
    Serial.println(temper);
    updateTemp(temper);
    delay(5000);
    // Clears the trigPin
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    // Sets the trigPin on HIGH state for 10 micro seconds
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);
    duration = pulseIn(echoPin, HIGH); // Reads the echoPin, returns the sound wave travel time in microseconds
    distance= duration*0.034/2; // Calculating the distance
    // Prints the distance on the Serial Monitor
    if (distance < (MaxHeight-(MaxHeight*.3)) ){
        distance -=3; // adjustment
    }
    if (distance<=4){
        Serial.println("Bin is Full"); distance =1;
    }
    if ( firstTime == HIGH ){
        Serial.print("Distance: "); Serial.println(distance);
        Serial.print( " Bin ");
        binStatus = (distance*100/MaxHeight) ;
    }
}

```

```

    if (binStatus >100) binStatus = 100;

    Serial.print(binStatus );

    Serial.println("% Empty");

    oldDistance = distance ;

    firstTime = LOW;

}

if ( ( distance < MaxHeight) && (distance != oldDistance) ) {

    Serial.print("Distance: ");   Serial.println(distance);

    Serial.print( " Bin ");

    binStatus = (distance*100/MaxHeight) ;

    Serial.print(binStatus );

    Serial.println("% Empty");

    oldDistance = distance ;

}

Serial.print("Distance: ");   Serial.println(distance);

Serial.print("oldDistance: ");   Serial.println(oldDistance);

delay (2000);

}

```

## **CONCLUSION**

As an electronics engineer trainee at National Institute of Electronics and Information Technology, I learned various methods to design and develop systems based on Electronics. I also learned product design based on small microcontrollers to fairly complex controllers, other supporting factors like reliability, form factors, team work etc. which adds to the functionality of electronic product in IoT.

I was able to complete all of the assigned tasks. Besides the knowledge that was gained from completing my project, I gained a lot from my mentor and fellow classmates. I gained valuable experience and exposure to latest technologies and tools working with team of other engineers from other colleges. Weekly team discussions helped me identify and solve numerous problem issues which enhanced my skills to understand the field of electronics. I would like to convey my thanks for providing me an opportunity to gain idea of competitive environment in professional field.



## **REFERENCES**

- <http://arshbir.com/course/view.php?id=32>
- [https://www.researchgate.net/publication/303842610\\_IoT\\_based\\_Smart\\_Parking\\_System](https://www.researchgate.net/publication/303842610_IoT_based_Smart_Parking_System)
- <https://ieeexplore.ieee.org/document/7772297/>
- [.http://www.instructables.com/id/Smart-Parking-System/](http://www.instructables.com/id/Smart-Parking-System/)<https://www.circuitstoday.com/>
- <https://www.ibm.com/developerworks/library/iot-nodemcu-open-why-use/index.html>
- <http://esp8266.net/>
- <https://opensource.com/resources/raspberry-pi>
- <https://nodered.org/docs/getting-started/installation>
- <https://thingspeak.com/channels/public>
- [http://nodemcu.com/index\\_en.html](http://nodemcu.com/index_en.html)
- <https://internetofthingsagenda.techtarget.com/definition/MQTT-MQ-Telemetry-Transport>

