**INTELLIGENT MEDICINE BOX FOR MEDICATION MANAGEMENT USING IOT**

**ABSTRACT**

Now a day’s trend in healthcare is to move routine medical checks and other health care services from hospital to the home environment. A modern healthcare IoT platform with an intelligent medicine box along with sensors for health monitoring and diagnosis will be proposed. An intelligent home based medicine box with wireless connectivity along with an iot application helps patients and doctors to be in more close communication. The proposed model has an intelligent medicine box that gives alerts to patients for their medication at right time. It is connected to internet to make timely updates about medicine to patient’s Smartphone through notices in iot application. The system automatically gives alarm to the patient to take the medicine at right time. Sms alerts are given to predefined guardian if there are any vital signs noticed.

**INTRODUCTION**

Now a day’s healthcare is a burden factor for systems are struggling with aging population, prevalence of chronic diseases, and the accompanying rising costs. Aging populations and the rising incidence of chronic disease consume a disproportionate amount of healthcare resources. In the World, about 75% of healthcare dollars go to chronic disease care and two out of every three Medicare recipients suffer from at least two chronic diseases. More than 40% of world population suffers from chronic conditions, often with no cure in sight, which can get hugely expensive. healthcare has been recognized to be the next generation form of healthcare, and distributed, patient-centric and self-managed care is emphasized as an alternative to the traditional hospitalized, staff-centric and professional-managed care [1]. Pervasive healthcare based on the emerging technologies of the Internet-of-Things (IoT), as so-called Health-IoT, is highlighted as one of the killer applications of the IoT. Many projects and initiatives have been devoted in this promising area. Unfortunately, the concern to prescription medication noncompliance, a basic type of self-managed care, is insufficient in the existing research [2]. A frequently cited fact is: medication noncompliance costs the United States healthcare system up to $100 billion per year, and it is the cause of approximately 11% of US annual hospitalizations [3].

It has been proven that, for the 4 most drug-spending chronic conditions (diabetes, hypertension, hypercholesterolemia, and congestive heart failure), hospitalization rates are significantly lower for patients with higher medication compliance [4]. More startling figures are listed in a report from the National Council on Patient Information and Education (NCPIE) in 2007 [5]: only about 50% of American patients take their medicines as prescribed, resulting in approximately $177 billion direct and indirect costs to the U.S. economy annually. To address the medication noncompliance problem, one solution from traditional packaging industry is the One Dose Packaging [6] which packetizes the tablets or capsules of one dosage into one small box of bag. It just makes medication more convenient for patients, but neither improves the compliance nor prevents from noncompliance. Noncompliance detecting and recording capability is offered by the Smart Medical Refrigerator in [7], the microchip powered tablet package in [8] and the Smart Dose Reminder in [9]. But these are mainly afterward checking measure instead of preventive measure, and the operations of these solutions are so complicated that they are only usable for trained caregivers instead of the elderly, disabled, and patients. At the same time, the increasing demands of daily monitoring prompt the Health-IoT solution to integrate more sensing and data processing capacities especially for on-site diagnosis and prognosis. For example, tri-axis accelerometer, electrocardiogram (ECG), blood pressure, blood oxygen saturation (SpO2), respiration oxygen saturation, blood sugar concentration, body temperature can be monitored on 24/7 basis [10-14].

So a powerful in-home terminal is needed not only to address the medication noncompliance but also to be used as a generic in-home healthcare station (IHHS) in everyone’s home. In this paper, extending our previous works in [16-21], an in-home medication management and healthcare system is proposed based on intelligent and interactive packaging (I2Pack) and intelligent medicine box (iMedBox). Preventive medication management is enabled by the intelligent pharmaceutical packaging which is sealed by Controlled Delamination Material (CDM) and controlled by wireless communication. Various vital parameters can also be collected by wearable biomedical sensors through the wireless link. On-site diagnosis and prognosis of these vital signals are supported by the powerful architecture. Additionally, friendly user interface is emphasized to ease the operations especially for the elderly, disabled and patients. A prototyping system is implemented and verified by field trials.

Once a patient falls prey to a chronic disease, the need for continuous health monitoring becomes more important than in prevention and wellness. A number of device makers and other players are aiming to tackle this challenge through integrating the relevant streams of data needed to accurately monitor the health of a patient with a given condition.The pressure for relief will grow as the population ages with approximately 10,000 new patients estimated to enrol in medical care every day for the next 15 years. Using the concepts of IoT, Android Application, cloud computing in patients or his caretaker’s smart phone the patient can view his medication status [1]. Internet of Things definition (IoT): The vast network of devices connected to the Internet, including smart phones and tablets and almost anything with a sensor on it. These “things” collect and exchange data. The machine-to-machine (M2M) technology behind it – are bringing a kind of “super visibility” to nearly every industry.

Imagine utilities and Telco that can predict and prevent service outages, airlines that can remotely monitor and optimize plane performance, and healthcare organizations that can base treatment on real-time genome analysis. Enter the Internet of Things (IoT), also referred to as the Internet of Medical Things (IoMT) within the healthcare industry. IoT is something that most are well familiar with, but for the sake of clarity, we define it here as the purposeful connection of intelligent sensors, devices, and software to computer networking systems using Bluetooth, Wi-Fi, RFID or M2M wireless technology in order to promote an inter-functionality that serves a greater purpose. In healthcare, that greater purpose is the achievement of less costly and more information-driven and efficient patient care.

A complete solution for in-home health care is still missing. A desirable system should be capable of taking care of the patients from all aspects, covering personalized medication, vital signs monitoring, on-site diagnosis and interaction with remote physicians. In addition, the existing systems rarely integrate new materials or apply new manufacturing approaches, which are always the key elements for bringing new devices or solutions into healthcare fields. By taking the above-mentioned issues into consideration, an intelligent home-based healthcare IoT system, iHome Health-IoT, is proposed. Recent technological advances in sensors, low-power integrated circuits, and wireless communications have enabled the design of low-cost, miniature, lightweight, and intelligent physiological sensor nodes. These nodes, capable of sensing, processing, and communicating one or more vital signs, can be seamlessly integrated into wireless personal or body networks for health monitoring. These network promise to revolutionize health care by allowing inexpensive, non-invasive, continuous, ambulatory health monitoring with almost real time updates of medical records via the Internet. To resolve these issues, developing an intelligent medicine box equipped with medication reminders and IoT module focusing on problems faced by the elderly required to take medication.

The system consists of three parts, health monitoring part, medicine box and IoT module. The sensors attached with the person and monitor the real time data and sends to the medicine box. By comparing these values with normal values, if any abnormalities are present in the values a text message will be sent to the doctor/relatives through the embedded GSM module. The medicine box compares the time with real time clock and opens the each medicine slot according to the prescribed medicine time. The doctor can reset the medicine time through IoT module. When a patient consults a doctor or hospital, his/her details including medicine prescription should be uploaded to the web server and each patient gets a unique user ID. The information about the patient is given to cloud. So at any time the doctor or relative can visit the web page and can follow the current status of the patient. According to the value of sensors the doctor can provide on-line prescription. That is he can ``reschedule the time of each medicine. The system automatically reset the time schedules according with the new prescription. The rest of the report is organized as follows. In second chapter, the related works of this paper are discussed. Chapter three describes the Methodology of the medicine box. The result and analysis of the project is presented in more details in chapter four. Finally, chapter five concludes the work and also talks about the future scope of the work.

**LITERATURE REVIEW**

In order to stabilize the condition of the patient, it is necessary for the patient to take medications regularly as per the prescribed dose and duration. Traditional plaid based medicine boxes were widely used, and many modifications on the traditional medicine box have been proposed previously.

**“Visual Identication of Medicine Boxes Using Features Matching”[ 2012]**

Gomes and Benjamim et.al, proposed a system which used visual features matching in the identification of medicine boxes for visually impaired people. It uses a camera device, available in devices like computers, televisions and cell phones, to identify relevant features on medicine box. After the medicine box detection, related audios are played to inform about dosage, indications and contra indications of the medication. For each medicine separate audios are recorded. This vision system can help many visually impaired people to take the right medicine at the right time prescribed by the doctor. Experiments with 15 blind folded volunteers demonstrated that 93% of them believe that the system was useful to identify the medicine box [10].

**“A Smart Pill Box with Remind and Consumption Confirmation Functions”[2015].**

Huai-Kuei Wu et.al, proposed pill box, where the camera is placed in inner side of the box to detect the matrix barcode and the medicine bag. A hardware module above the box was used to provide pill reminding and alarm functions. After visiting a doctor and returning home, a patient need only scans the matrix barcode using the camera of the pillbox, and all medicine related information will be loaded into the pill box. After the matrix barcode is scanned, the patient places the medicine bags in the pill box without dispensing the medicine in to the cell. This method is suitable for the elderly who do not have access to the internet as well. Furthermore, because private medical information is not transmitted via the internet, the risk of information theft is greatly reduced. Moreover, if an internet connection does exist, then patients can search for their medicine information from the hospital database [11].

**“The Smart Pill Box”[2012]**.

Brianna Abbey et al. proposed the smart pill box in 2012. The purpose of this system was to develop a medication device that increased medication compliance, monitored medication taking behaviour, and communicated with pharmacists. The device consists of 28 chambers that are placed in seven columns made up of four rows. Each column represents each day of a week. The 4 rows represent four distinct dosage times in a day. The LED light located behind each chamber provides the light source for the patients. The light also used as the indication for the ambient light sensor which is used to detect the light when the pills are removed from the chamber [12].

**“The Intelligent Pill Box - Design and Implementation”[2014].**

Shih-Chang Huang et.al, proposed a system where the infrared sensors are fixed at the entering where the patients take the medicine package. The detection of medicine taken will be delivered to the back-end control system and record to SQL server via the wireless serial port. The time to take the medicine package away will be recorded. The motor controls the spring which is used to put the medicine packages shown as figure. A base board is installed under the spring. As the motor rotates, the packages will be pushed forward. A package drops to the entrance when it moves out of the range of the board. There is disadvantage that the number of motors increased with number of medicine type [13]

**"A Pervasive and Preventive Healthcare Solution for Medication Noncompliance and Daily Monitoring",[2009]**

Pervasive healthcare solution for medication noncompliance problem would help to save $177 billion annually in the United States. And the rapidly increasing demanding of daily monitoring with onsite diagnosis and prognosis is driving homecare solutions to integrate more and more sensing and data processing capacities. So a powerful system is needed not only to address the medication noncompliance but also to be used as a pervasive healthcare station in home. In this paper, a pervasive and preventive healthcare solution for medication noncompliance and daily monitoring is proposed using an intelligent package sealed by controlled delamination material (CDM) and controlled by radio frequency identification (RFID). Onsite diagnosis and prognosis capacities for kinds of health parameters are supported due to scalable and intensive computing capacitance of the 2D-Mesh-NoC based multi-core architecture. Additionally, friendly human-machine interface is emphasized to make it usable for the elderly, disabled and patients due to enhanced multimedia performance. Experimental results of an implemented prototype confirmed the necessity of the multi-core architecture and approved the feasibility of the proposed intelligent package.[1].

**"Congestion-aware, loss-resilient biomonitoring sensor networking for mobile health applications"[2013]**

Decision-Support System (DSS) frameworks provide the mechanism with which clinical data from tele health devices is analyzed. This paper will also show the importance of DSS in tele health management systems for all patients with a chronic disease, not just those with diabetes [2].Android is a mobile operating system developed by Google, based on the Linux kernel and designed primarily for touch screen mobile devices such as Smartphone and tablets. Android's user interface is mainly based on direct manipulation, using touch gestures that loosely correspond to real-world actions, such as swiping, tapping and pinching, to manipulate on-screen objects, along with a virtual keyboard for text input [2].

**"An In-home Medication Management Solution Based on Intelligent Packaging and Ubiquitous Sensing".[2013]**

A healthcare solution for medication noncompliance problem would help to save $177 billion annually in the United States. In addition, an in-home healthcare station (IHHS) is needed to meet the rapidly increasing demands for daily monitoring with on-site diagnosis and prognosis. In this paper, an intelligent medication management system is proposed based on intelligent package and ubiquitous sensing technologies. Preventive medication management is enabled by an intelligent package sealed by Controlled Delamination Material (CDM) and controlled by RFID link. Various vital parameters are collected by wearable biomedical sensors through the short range wireless link. Onsite diagnosis and prognosis based on these health parameters are supported by the scalable architecture. Additionally, friendly human-machine interface is emphasized to make it convenient for the elderly or disabled patients. A prototype system including the hardware, embedded software, user interface, database and some intelligent packages is implemented to verify the concepts[3].

**"A cloud computing solution for patient's data collection in health care institutions",[2010]**

Existing processes for patients' vital data collection require a great deal of labor work to collect, input and analyze the information. These processes are usually slow and error-prone, introducing a latency that prevents real-time data accessibility. This scenario restrains the clinical diagnostics and monitoring capabilities. We propose a solution to automate this process by using sensors attached to existing medical equipment’s that are inter-connected to exchange service. The proposal is based on the concepts of utility computing and wireless sensor networks. The information becomes available in the ¿cloud¿ from where it can be processed by expert systems and/or distributed to medical staff. The proof-of-concept design applies commodity computing integrated to legacy medical devices, ensuring cost-effectiveness and simple integration.

Android has always been focused on device communication and networking. It’s got vast and complete libraries for a variety of wired and wireless communication protocols. It supports and has drivers for a large amount of radio devices/chips from the most common manufacturers. Where medical devices have suffered in the past due to the lack of a standard communication protocol outside of HL7, Android based medical devices can take advantage of everything modern software has to offer such as the simple and flexible exchange of data via API’s. Cloud Computing is a commercial extension of computing resources which provides scalable resources and economic benefits to its users over the internet. It acts as software and provides data access and storage services which don’t need the knowledge of the end users physical location and the systems configuration that provides the computing resources [3]. In Cloud Computing, the users use the web browsers as an interface, while the software and data are stored on the remote servers and hence it is device independent [4].

**"Medical application on internet of things",[2011]**

Internet technology provides various services via network. With the diversification of terminals and development of internet technology, internet technology has come into the stage of Next Generation Network(NGN) technology. Comparing with the current Internet technology that provides services in the imaginary space, the technology on Internet of Things (IOT) is based on real word. It links things together via sensors and wireless communication technology to collect a variety of information on the condition of people and their surrounding space in the real world. The combination of Internet technology and Technology on IOT integrates physical world and imaginary space on a shared platform to eliminate the constraints of imaginary space and provide intricate, diverse, and advanced services focusing on people, which have not been achieved. The future direction for the integration of Internet technology and technology on IOT, the technology of body sensor network and information services are suggested. This paper analyze the possibility and related issues of providing advanced services for human health management in the real world and research direction of medical technology on IOT[5].

**PROPOSED SYSTEM**

Our system includes a featured medicine box which is wirelessly connected to the hospital administration. Hospital administration monitors the routine details through a webpage which is managed at the hospital side. An android application is installed on the patient’s smart phone. Through this application patient could view their prescriptions and get notifications regarding medicine intake. Medicine box is provided with different compartments. An LED on top of each compartment denotes the correct box. Whenever patient opens a wrong compartment, a buzzer will get activated with the help of Microcontroller. A Wi-Fi shield is attached to the microcontroller board and this microcontroller picks up the data and sends it through ESP8266 Wi-Fi module which automatically updates these details to the hospital webpage.

**BLOCK DIAGRAM**

Power supply

Microcontroller

Wi-Fi

Buzzer

Servo motors

Led indicators

Buttons

APP

humidity sensor

Temperature sensor

**Hardware Requirements**

* Microcontroller
* Buzzer
* Buttons/switches
* Led’s
* Heartbeat sensor
* Temperature sensor
* Wi-Fi module
* Power supply 5v/2a
* Magnetics reed switches
* Servo motor

**Software requirements**

* C++
* Complier IDE
* IoT cloud
* App

**METHDOLOGY**

The Intelligent Medicine Box platform is built based on the integration of the medicine package and smart sensors via various wireless links. The focus is to regulate and optimize the accessibility of medical drugs and efficiently provide home-based health care services. The developed IoT system connects the individual home environment with hospital, emergency center and other medical facilities and provides remote and medication non-compliance service.

**IoT**

The **Internet of Things** (**IoT**) is a system of interrelated computing devices, mechanical and digital machines, objects, animals or people that are provided with unique [identifiers](https://en.wikipedia.org/wiki/Identifiers) (UIDs) and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction.

The definition of the Internet of Things has evolved due to the convergence of multiple technologies, real-time [analytics](https://en.wikipedia.org/wiki/Analytics), [machine learning](https://en.wikipedia.org/wiki/Machine_learning), [commodity](https://en.wikipedia.org/wiki/Commodity) [sensors](https://en.wikipedia.org/wiki/Sensors), and [embedded systems](https://en.wikipedia.org/wiki/Embedded_system). Traditional fields of embedded systems, [wireless sensor networks](https://en.wikipedia.org/wiki/Wireless_sensor_network), [control systems](https://en.wikipedia.org/wiki/Control_system), [automation](https://en.wikipedia.org/wiki/Automation) (including [home](https://en.wikipedia.org/wiki/Home_automation) and [building automation](https://en.wikipedia.org/wiki/Building_automation)), and others all contribute to enabling the Internet of Things. In the consumer market, IoT technology is most synonymous with products pertaining to the concept of the "smart home", covering devices and [appliances](https://en.wikipedia.org/wiki/Home_appliance) (such as lighting fixtures, [thermostats](https://en.wikipedia.org/wiki/Thermostats), home [security systems](https://en.wikipedia.org/wiki/Security_systems) and cameras, and other home appliances) that support one or more common ecosystems, and can be controlled via devices associated with that ecosystem, such as [smartphones](https://en.wikipedia.org/wiki/Smartphone) and [smart speakers](https://en.wikipedia.org/wiki/Smart_speaker).

The IoT's major significant trend in recent years is the explosive growth of devices connected and controlled by the Internet. The wide range of applications for IoT technology mean that the specifics can be very different from one device to the next but there are basic characteristics shared by most.

The IoT creates opportunities for more direct integration of the physical world into computer-based systems, resulting in efficiency improvements, economic benefits, and reduced human exertions.

The number of IoT devices increased 31% year-over-year to 8.4 billion in the year 2017and it is estimated that there will be 30 billion devices by 2020. The global market value of IoT is projected to reach $7.1 trillion by 2020.

**Medical and healthcare**

The **Internet of Medical Things** (**IoMT**), (also called the **Internet of health things**), is an application of the IoT for medical and health related purposes, data collection and analysis for research, and monitoring.The IoMT has been referenced as "Smart Healthcare",as the technology for creating a digitised healthcare system, connecting available medical resources and healthcare services.

IoT devices can be used to enable [remote health monitoring](https://en.wikipedia.org/wiki/Remote_patient_monitoring) and [emergency notification systems](https://en.wikipedia.org/wiki/Emergency_notification_system). These health monitoring devices can range from blood pressure and heart rate monitors to advanced devices capable of monitoring specialised implants, such as pacemakers, Fitbit electronic wristbands, or advanced hearing aids.Some hospitals have begun implementing "smart beds" that can detect when they are occupied and when a patient is attempting to get up. It can also adjust itself to ensure appropriate pressure and support is applied to the patient without the manual interaction of nurses. A 2015 Goldman Sachs report indicated that healthcare IoT devices "can save the United States more than $300 billion in annual healthcare expenditures by increasing revenue and decreasing cost." Moreover, the use of mobile devices to support medical follow-up led to the creation of 'm-health', used "to analyse, capture, transmit and store health statistics from multiple resources, including sensors and other biomedical acquisition systems".

Specialized sensors can also be equipped within living spaces to monitor the health and general well-being of senior citizens, while also ensuring that proper treatment is being administered and assisting people regain lost mobility via therapy as well. These sensors create a network of intelligent sensors that are able to collect, process, transfer, and analyze valuable information in different environments, such as connecting in-home monitoring devices to hospital-based systems. Other consumer devices to encourage healthy living, such as connected scales or [wearable heart monitors](https://en.wikipedia.org/wiki/Wearable_technology), are also a possibility with the IoT. End-to-end health monitoring IoT platforms are also available for antenatal and chronic patients, helping one manage health vitals and recurring medication requirements.

Advances in plastic and fabric electronics fabrication methods have enabled ultra-low cost, use-and-throw IoMT sensors. These sensors, along with the required RFID electronics, can be fabricated on [paper](https://en.wikipedia.org/wiki/Paper) or [e-textiles](https://en.wikipedia.org/wiki/E-textiles) for wirelessly powered disposable sensing devices. Applications have been established for [point-of-care medical diagnostics](https://en.wikipedia.org/wiki/Point-of-care_testing), where portability and low system-complexity is essential.

As of 2018 IoMT was not only being applied in the [clinical laboratory](https://en.wikipedia.org/wiki/Clinical_laboratory) industry,but also in the healthcare and health insurance industries. IoMT in the healthcare industry is now permitting doctors, patients, and others, such as guardians of patients, nurses, families, and similar,to be part of a system, where patient records are saved in a database, allowing doctors and the rest of the medical staff to have access to patient information.[  Moreover, IoT-based systems are patient-centered, which involves being flexible to the patient's medical conditions. IoMT in the insurance industry provides access to better and new types of dynamic information. This includes sensor-based solutions such as biosensors, wearables, connected health devices, and mobile apps to track customer behaviour. This can lead to more accurate underwriting and new pricing models.

The application of the IOT in healthcare plays a fundamental role in managing chronic diseases and in disease prevention and control. Remote monitoring is made possible through the connection of powerful wireless solutions. The connectivity enables health practitioners to capture patient's data and applying complex algorithms in health data analysis.



**PROBLEM STATEMENT**

In the present world, problem of medication non-adherence is critical because of

1) Forgetfulness in the case of elders

2) Inability to read and understand the prescription and the names on the medicines.

Limitations of existing system:

• Cannot identify which medicine to be taken.

**ADVANTAGES**

* Cost efficient: Our product cost is affordable compare to other product available in market.
* User friendly: User can set time table of medicine by himself.
* Highly reliable: Good in quality and performance; able to be trusted for patients & old age people.
* Provide comfort and health: Comfortable for old age people and provide healthy life for patients who are regularly take medicines.
* Long-Lasting: The product can be used for long time

**DESIGN:**

The word system is possibly the most overused and abused term in the technical lexicon. System can be defined as the “a set of fact, principles, rules etc., classified and arranged in an orderly form so as to show a logical plan linking the various parts” here the system design defines the computer based information system. The primary objective is to identify user requirements and to build a system that satisfies these requirements.

Design is much more creative process than analysis. Design is the first step in the development of any system or product. Design can be defined as “the process of applying various techniques and principles for the purpose of defining a device, a process or a system in sufficient detail to permit its physical realization”.

It involves four major steps they are

1. Understanding how the system is working now;

2. Finding out what the system does now;

3. Understanding what the new system will do; and

4. Understanding how the new system will work.

So as to avoid these difficulties, a new system was designed to keep these requirements in mind. Therefore the manual process operation has been changed into GUI based environment, such that the user can retrieve the records in a user-friendly manner and it is very easy to navigate to the corresponding information.

INPUT DESIGN

Input design is the bridge between users and information system. It specifies the manner in which data enters the system for processing it can ensure the reliability of the system and produce reports from accurate data or it may results in output of error information.

OUTPUT DESIGN

### Outputs from the computer system are rewired primary to communicate the results of processing to the uses. They also used to provide a permanent copy of these results for later consultation / verification. The main points on designing an output are deciding the media, designing layout and report to be printed. The outputs are designed from the system, are simple to read and interpre

DATA FLOW DIAGRAM

A DFD is a logical model of the system. The model does not depend on the hardware, software and data structures of file organization. It tends to be easy for even non-technical users to understand and thus serves as an excellent communication tool.

DFD can be used to suggest automatic boundaries for proposed system at pa very high level; the entire system is shown as a single logical process clearly identifying the sources and destination of data. This is often referred to as zero level DFD.

Then the processing is exploded into major processes and the same is depicted as level one DFD.

**Methodology diagram**

Application

Server

Middleware

Gateway

Sensor

Sensing data transmission

Sensing data transmission

Data storage compare with threshold

Data request

Requested data transmission

Requested data transmission

Alert about the state and buzzer on

**USE CASE DIAGRAM**

Microcontroller user/admin

Robot

**Flow chart**

If Wi-Fi connected

CONNECT TO WIFI (ACCESS POINT)

NO

yes

Read the value from all the sensors

Weight & vibrationThershold

NO

Alert and buzzer on

YES

Alert and buzzer on

Application

Compare

Send values

Server

Sensor

0.0

**DATA FLOW DIAGRAM LEVEL 0**

Compare

Server

Sensor



Send values

Connect

Connect with server

Send path

Travel

Connect

Alert and Buzzer ON

**DATAFLOW DIAGRAM LEVEL 1**

Initialize Microcontroller

Initialize sensor

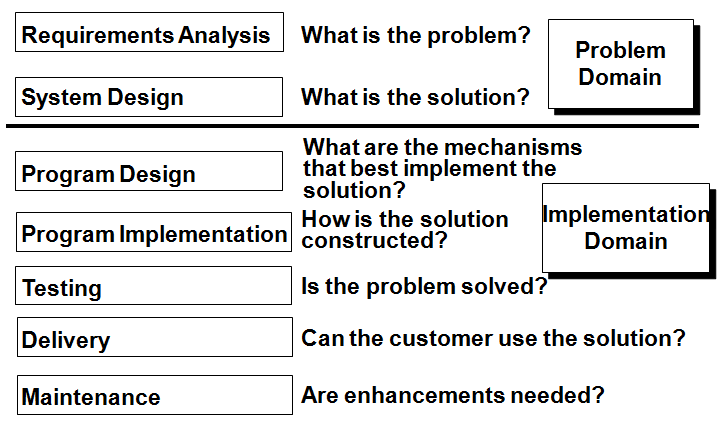
Detect the VALUE

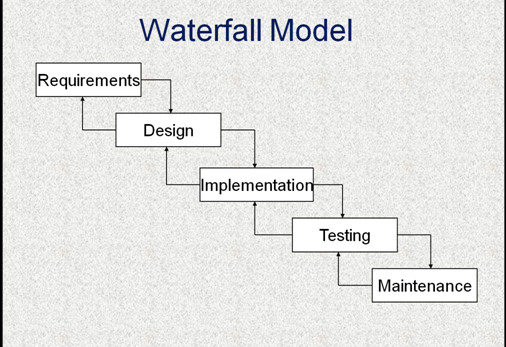
Alert message

Activate Buzzer / SEND NOTIFICATION

**Activity diagram**

**POSSIBLE IDENTIFICATION OF SOFTWARE DEVELOPMENT ACTIVITIES**

****

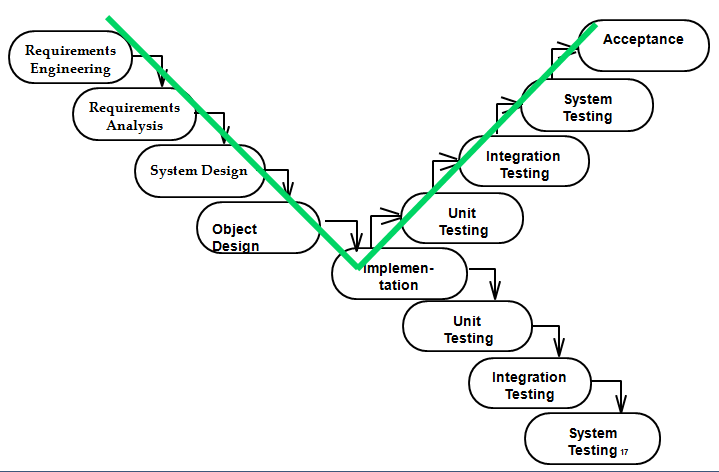
****

**SOFTWARE DEVELOPMENT AS APPLICATION DOMAIN: A USE CASE MODEL**

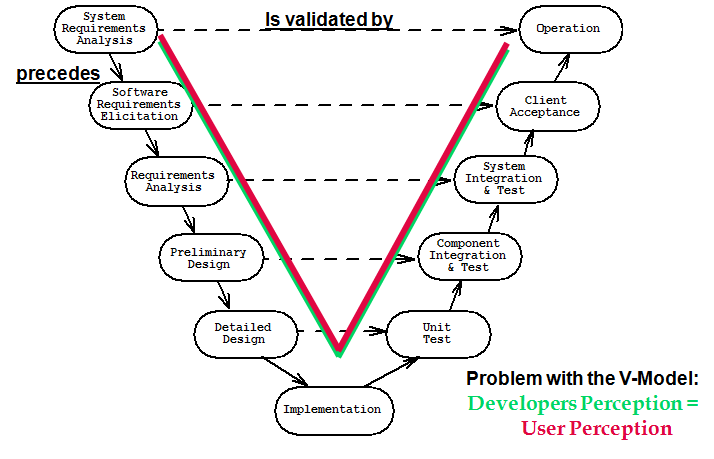
****

**THE WATERFALL MODEL OF THE SOFTWARE LIFE CYCLE**

**WATERFALL MODEL TO THE V MODEL**

****

**ACTIVITY DIAGRAM OF A V MODEL**

****

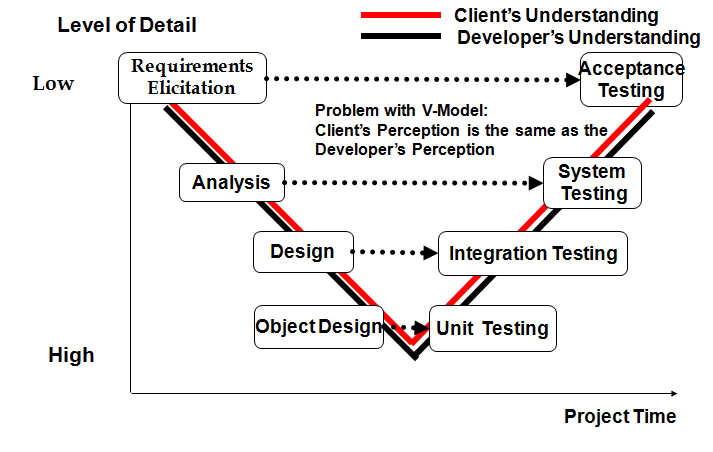
**A Generic Requirements Process**

****

**A Generic Software Design Process**

****

***V Model: Distinguishes between Development and Verification Activities***

****

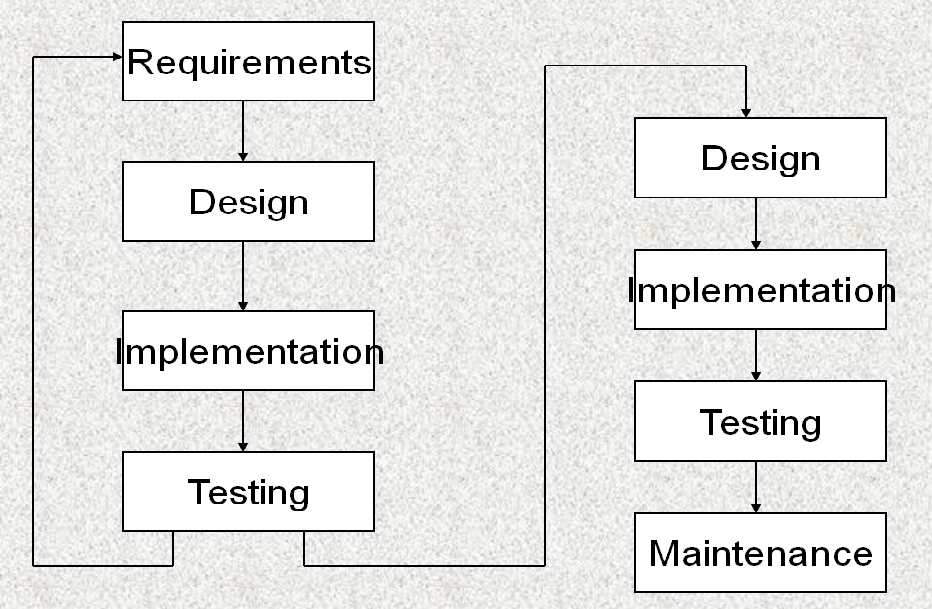
**A Generic Testing Process**

****

**V-Model of Test Planning**

****

**PROTOTYPING**

****

**INCREMENTAL DEVELOPMENT**

****

**SPIRAL MODEL PROCESS OF DEVELOPMENT IN IOT PROJECTS**

****

**FEASIBILITY STUDY**

## Introduction

A feasibility analysis involves a detailed assessment of the need, value and practicality of a p systems development... Feasibility analysis n forms the transparent decisions at crucial points during the developmental process as we determine whether it is operationally, economically and technically realistic to proceed with a particular course of action.

Feasibility analysis can be used in each of the steps to assess the financial, technical and operational capacity to proceed with particular activities.

### Types of feasibility

A feasibility analysis usually involves a thorough assessment of the financial (value), technical (practicality), and operational (need) aspects of a proposal. In systems development projects, business managers are primarily responsible for assessing the operational feasibility of the system, and information technology (IT) analysts are responsible for assessing technical feasibility. Both then work together to prepare a cost–benefit analysis of the proposed system to determine its economic feasibility.

#### Operational feasibility

A systems development project is likely to be operationally feasible if it meets the 'needs' and expectations of the organisation. User acceptance is an important determinant of operational feasibility. It requires careful consideration of:

corporate culture;

staff resistance or receptivity to change;

management support for the new system;

the nature and level of user involvement in the development and implementation of the system; direct and indirect impacts of the new system on work practices;

anticipated performance and outcomes of the new system compared with the existing system;

training requirements and other change management strategies; and

‘pay back’ periods (ie trade-off between long-term organisational benefits and short-term inefficiencies during system development and implementation).

#### Technical feasibility

A systems development project may be regarded as technically feasible or practical if the organization has the necessary expertise and infrastructure to develop, install, operate and maintain the proposed system. Organizations will need to make this assessment based on:

Knowledge of current and emerging technological solutions;

Availability of technically qualified staff in-house for the duration of the project and subsequent maintenance phase;

Availability of infrastructure in-house to support the development and maintenance of the proposed system;

Where necessary, the financial and/or technical capacity to procure appropriate infrastructure and expertise from outside;

Capacity of the proposed system to accommodate increasing levels of use over the medium term;

The capacity of the proposed system to meet initial performance expectations and accommodate new functionality over the medium term.

#### Economic feasibility

A systems development project may be regarded as economically feasible or good value to the organization if its anticipated benefits outweigh its estimated costs. Many development costs are easier to identify. These costs may include the time, budget and staff resources invested during the design and implementation phase, as well as infrastructure, support, training and maintenance costs incurred after implementation. Nonetheless, it can also be difficult to accurately quantify project costs when new technologies and complex systems are involved. In these high-risk situations it may be appropriate to use sophisticated cost-benefit analysis tools to make appropriate assessments of financial feasibility.

**SOFTWARE ENGINEERING PARADIGM APPLIED:**

As per the Software development lifecycle,

1. The requirement specification of the project is collected properly.

2. Feasibility studies are taken upon the requirement collected to test the feasibility of the project.

3. According to the requirement the project gets modularized.

4. Architectural and detailed designs are drawn for each and every module.

5. During implementation phase, comments are used to describe the actions that are taking place, which is important for documentation and future reference.

6. Testing is extensively done to check whether the project is doing the task for what it has been designed.

* 1. **Security Feasibility**

It is that a third person watching the communication between the sender and the receiver will not be able to find out whether the sender has been active, and when, in the sense that he really embedded a message in the cover-text. In other words, stegotexts should be indistinguishable from covertexts.

* 1. **System Feasibility**

The proposed system can be developed using the present hardware and software technologies. The project requires following requirements.

Hardware and software architecture with minimum requirements, which supports an operating system on which Java toolkit and Media player applications can be developed and deployed. The estimated time given to different phases in the project such as Analysis, Design, implementation and testing all sum up to make a total time required to complete the project as approximately equal to 2 months (Excluding future Enhancement).

* 1. **Special Features**

For video, a combination of sound and image techniques can be used. This is due to the fact that video generally has separate inner files for the video (consisting of many images) and the sound. So techniques can be applied in both areas to hide data. Due to the size of video files, the scope for adding lots of data is much greater and therefore the chances of hidden data being detected is quite low.

Bottom of Form

Bottom of Form

**Gantt chart**

45

25

20

50

Days-180

40

Analysis

Design

Implementation

Testing

Maintenance

Completed

Total Number of days to complete: **180 Days**

**Implementation**

Admin Module

Create/ Delete User

**Testing**

20

25

**Design**

All Type of **Maintenance**

Unit Testing /

I/O Testing

50

45

Transaction Details and

All operation

Login/Logout

40

**Analysis**

**LOC – BASED ESTIMATION**

**AREA OF CODE OPSTEMISTIC MOST LIKELY PESSIMISTIC LOC-EV**

Client Programming 2200 3250 3700 3200

Network Programming 1250 1500 1450 1350

4550

The three points or expected value is computed as

EV = (S opt + 4 Sm + S Press ) / 6

Thus the estimated lines of code(LOC) are 4550

Assuming the organization average productivity for system of this type is 1620 LOC/pm

Based on the burden labor rate of Rs. 7000 per month, the cost per line of code is approximately Rs. 4.00

Based on the LOC estimate,

The estimation effort e is computed as

E = estimated LOC/(LOC / pm)

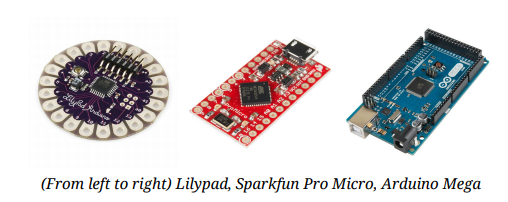
= 4550 / 1620 = 2.8086

app – 4 persons in a month or 1 person in 4 month

**SOFTWARE REQUIREMENTS**

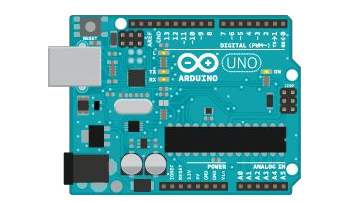
**ARDUINO IDE**

Arduino first and foremost is an open-source computer hardware and software company. The Arduino Community refers to the project and user community that designs and utilizes microcontroller-based development boards. These development boards are known as Arduino Modules, which are open-source prototyping platforms. The simplified microcontroller board comes in a variety of development board packages.



The most common programming approach is to use the Arduino IDE, which utilizes the C programming language. This gives you access to an enormous Arduino Library that is constantly growing thanks to open-source community.

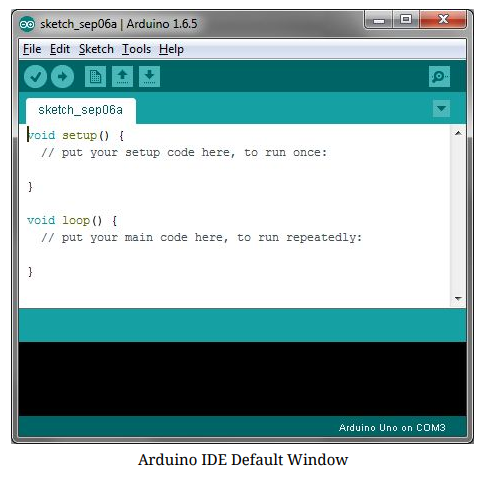
Arduino IDE is not: AVR Studio (Yes, we know you loved EE 346, but unfortunately you won’t be utilizing Assembly Language)



Arduino Uno dev. board (Fritzing part graphic)

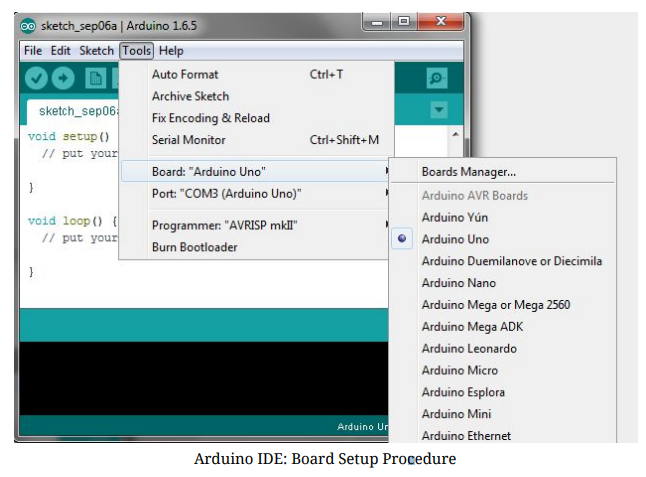
Find out all the information you ever wanted to know about Arduino here: https://learn.sparkfun.com/tutorials/what-is-an-arduino Also, checkout sparkfun’s Arduino Buying Guide for further information on available dev. boards: <https://www.sparkfun.com/arduino_guide>

Download Arduino Integrated Design Environment (IDE) here (Most recent version: 1.6.5): https://www.arduino.cc/en/Main/Software This is the Arduino IDE once it’s been opened. It opens into a blank sketch where you can start programming immediately. First, we should configure the board and port settings to allow us to upload code. Connect your Arduino board to the PC via the USB cable.



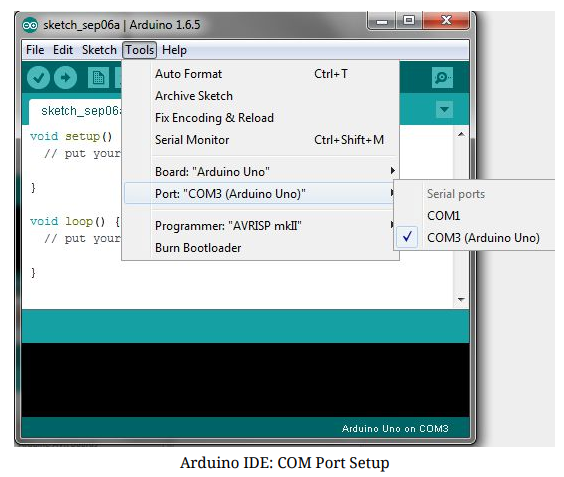
Board Setup

You have to tell the Arduino IDE what board you are uploading to. Select the Toolspulldown menu and go to Board.This list is populated by default with the currently available Arduino Boards that are developed by Arduino. If you are using an Uno or an Uno-Compatible Clone (ex. Funduino, SainSmart, IEIK, etc.), select Arduino Uno. If you are using another board/clone, select that board.



IDE: COM Port Setup

If you downloaded the Arduino IDE before plugging in your Arduino board, when you plugged in the board, the USB drivers should have installed automatically. The most recent Arduino IDE should recognize connected boards and label them with which COM port they are using. Select the Tools pulldown menu and then Port.Here it should list all open COM ports, and if there is a recognized Arduino Board, it will also give it’s name. Select the Arduino board that you have connected to the PC. If the setup was successful, in the bottom right of the Arduino IDE, you should see the board type and COM number of the board you plan to program. Note: the Arduino Uno occupies the next available COM port; it will not always be COM3.



At this point, our board should be set up for programming, and you can begin writing and uploading code.

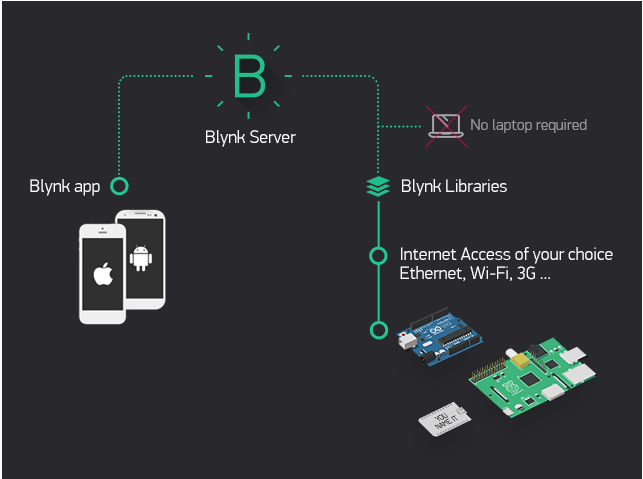
BLYNK

Blynkwas designed for the Internet of Things. It can control hardware remotely, it can display sensor data, it can store data, vizualize it and do many other cool things.

There are three major components in the platform:

* **Blynk App** - allows to you create amazing interfaces for your projects using various widgets we provide.
* **Blynk Server** - responsible for all the communications between the smartphone and hardware. You can use our Blynk Cloud or run your [private Blynk server](https://docs.blynk.cc/#blynk-server) locally. It’s open-source, could easily handle thousands of devices and can even be launched on a Raspberry Pi.
* **Blynk Libraries** - for all the popular hardware platforms - enable communication with the server and process all the incoming and outcoming commands.

Now imagine: every time you press a Button in the Blynk app, the message travels to  the Blynk Cloud, where it magically finds its way to your hardware. It works the same in the opposite direction and everything happens in a blynk of an eye.



Features

* Similar API & UI for all supported hardware & devices
* Connection to the cloud using:
  + WiFi
  + Bluetooth and BLE
  + Ethernet
  + USB (Serial)
  + GSM
* Set of easy-to-use Widgets
* Direct pin manipulation with no code writing
* Easy to integrate and add new functionality using virtual pins
* History data monitoring via SuperChart widget
* Device-to-Device communication using Bridge Widget
* Sending emails, tweets, push notifications, etc.
* new features are constantly added!

You can find [example sketches](https://github.com/blynkkk/blynk-library/tree/master/examples) covering basic Blynk Features. They are included in the library. All the sketches are designed to be easily combined with each other.

**What do I need to Blynk?**

At this point you might be thinking: **“Ok, I want it. What do I need to get started?”** – Just a couple of things, really:

#### **1. Hardware**.

An Arduino, Raspberry Pi, or a similar development kit.

**Blynk works over the Internet.** This means that the hardware you choose should be able to connect to the internet. Some of the boards, like Arduino Uno will need an Ethernet or Wi-Fi Shield to communicate, others are already Internet-enabled: like the ESP8266, Raspberri Pi with WiFi dongle, Particle Photon or SparkFunBlynk Board. But even if you don’t have a shield, you can connect it over USB to your laptop or desktop (it’s a bit more complicated for newbies, but we got you covered). What’s cool, is that the [list of hardware](https://docs.blynk.cc/#supported-hardware) that works with Blynk is huge and will keep on growing.

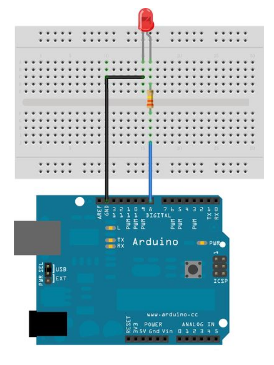
#### **2. A Smartphone**.

The Blynk App is a well-designed interface builder. It works on both iOS and Android.

Getting Started

Let’s get you started in 5 minutes (reading doesn’t count!). We will switch on an LED connected to your Arduino using the Blynk App on your smartphone.

Connect an LED as shown here:

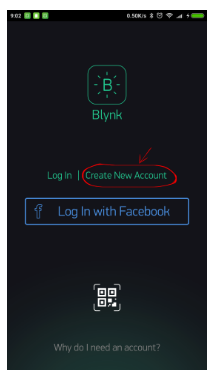


Getting Started With TheBlynk App

1. Create a Blynk Account

After you download the Blynk App, you’ll need to create a New Blynk account. This account is separate from the accounts used for the Blynk Forums, in case you already have one.

We recommend using a **real** email address because it will simplify things later.



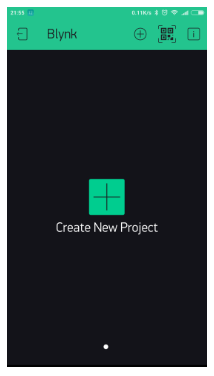
#### Why do I need to create an account?

An account is needed to save your projects and have access to them from multiple devices from anywhere. It’s also a security measure.

You can always set up your own [Private Blynk Server](https://docs.blynk.cc/#blynk-server) and have full control

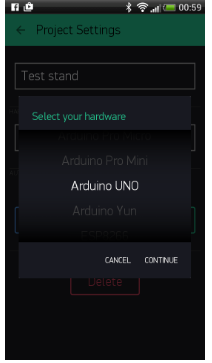
Create a New Project

After you’ve successfully logged into your account, start by creating a new project.



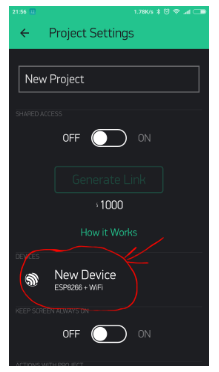
Choose Your Hardware

Select the hardware model you will use. Check out the [list of supported hardware](https://docs.blynk.cc/#supported-hardware)!



Auth Token

**Auth Token** is a unique identifier which is needed to connect your hardware to your smartphone. Every new project you create will have its own Auth Token. You’ll get Auth Token automatically on your email after project creation. You can also copy it manually. Click on devices section and selected required device :



**NOTE:** Don’t share your Auth Token with anyone, unless you want someone to have access to your hardware.

It’s very convenient to send it over e-mail. Press the e-mail button and the token will be sent to the e-mail address you used for registration. You can also tap on the Token line and it will be copied to the clipboard.

Now press the **“Create”** button.

Add a Widget

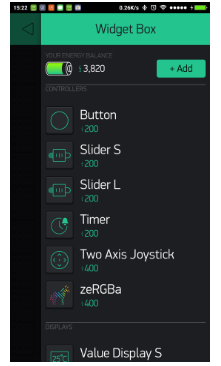
Your project canvas is empty, let’s add a button to control our LED.

Tap anywhere on the canvas to open the widget box. All the available widgets are located here. Now pick a button.

**Widget Box**

**Drag-n-Drop** - Tap and hold the Widget to drag it to the new position.

**Widget Settings** - Each Widget has it’s own settings. Tap on the widget to get to them.



Run The Project

When you are done with the Settings - press the **PLAY** button. This will switch you from EDIT mode to PLAY mode where you can interact with the hardware. While in PLAY mode, you won’t be able to drag or set up new widgets, press **STOP** and get back to EDIT mode.

You will get a message saying “Arduino UNO is offline”. We’ll deal with that in the next section.

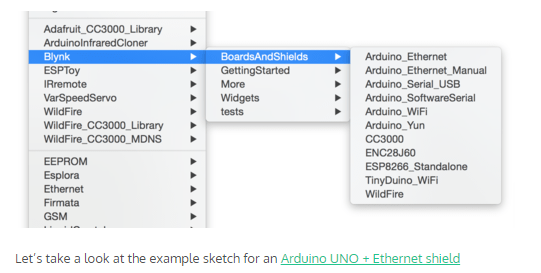
Getting Started With Hardware

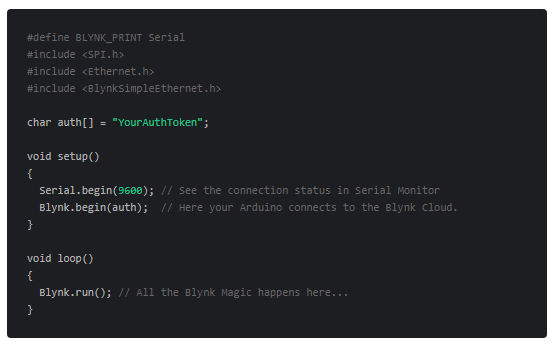
How To Use an Example Sketch

You should by now have the Blynk Library installed on your computer. If not - [click here](https://docs.blynk.cc/#downloads-blynk-library).

Example sketches will help you get your hardware online quickly and major Blynk features.

Open the example sketch according to the hardware model or shield you are using.





Auth Token

In this example sketch, find this line:

charauth[] = "YourAuthToken";

This is the [Auth Token](https://docs.blynk.cc/#getting-started-getting-started-with-application-4-auth-token) that you emailed yourself. Please check your email and copy it, then paste it inside the quotation marks.

It should look similar to this:

charauth[] = "f45626c103a94983b469637978b0c78a";

Upload the sketch to the board and open Serial Terminal. Wait until you see something like this:

Blynkv.X.X.X

Your IP is 192.168.0.11

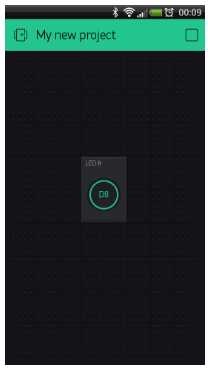
Connecting...

Blynk connected!

**Congrats! You are all set! Now your hardware is connected to the Blynk Cloud!**

Blynking

Go back to the Blynk App, push the button and turn the LED on and off! It should be Blynking



Check out [other example sketches](https://github.com/blynkkk/blynk-library/tree/master/examples).

Feel free to experiment and combine different examples together to create your own amazing projects.

For example, to attach an LED to a [PWM](http://www.arduino.cc/en/Tutorial/Fading)-enabled Pin on your Arduino, set the slider widget to control the brightness of an LED. Just use the same steps described above.

**Web requirements**

**Languages**

**HTML**

HTML is a hypertext markup language which is in reality a backbone of any website. Every website can’t be structured without the knowledge of html. If we make our web page only with the help of html, than we can’t add many of the effective features in a web page, for making a web page more effective we use various platforms such as CSS. So here we are using this language to make our web pages more effective as well as efficient. And to make

**CSS**

CSS stands for "Cascading Style Sheet" Cascading style sheets are used to format the layout of Web pages. They can be used to define text styles, table sizes, and other aspects of Web pages that previously could only be defined in a page's HTML. The basic purpose of CSS is to separate the content of a web document (written in any markup language) from its presentation (that is written using Cascading Style Sheets). There are lots of benefits that one can extract through CSS like improved content accessibility, better flexibility and moreover, CSS gives a level of control over various presentation characteristics of the document. It also helps in reducing the complexity and helps in saving overall presentation time. CSS gives the option of selecting various style schemes and rules according to the requirements and it also allows the same HTML document to be presented in more than one varying style.

**JAVASCRIPT**

JavaScript is considered to be one of the most famous scripting languages of all time. JavaScript, by definition, is a Scripting Language of the World Wide Web. The main usage of JavaScript is to add various Web functionalities, Web form validations, browser detections, creation of cookies and so on. JavaScript is one of the most popular scripting languages and that is why it is supported by almost all web browsers available today like Firefox, We used the browser Opera or Internet Explorer. JavaScript is considered to be one of the most powerful scripting languages in use today. It is often used for the development of client-side web development. JavaScript is used to make web pages more interactive and dynamic. JavaScript is a light weight programming language and it is embedded directly into the HTML code. JavaScript, as the name suggests, was influenced by many languages, especially Java.

**PHP**

Precisely, PHP is a very powerful server-side scripting language for developing dynamic web applications. Using PHP, one can build interactive and dynamic websites with ease. PHP script can be embedded straight into the heart of html code. PHP is compatible with various web servers like Apache and the Microsoft’s IIS as well. All the PHP scripts are executed on the server and it supports various databases like MySQL, Oracle, Solid, Generic ODBC etc.; however, it is mostly used with MySQL.

**MySQL**

SQL stands for Structured Query Language. MySQL lets us access and manipulate databases. MySQL is an ANSI (American National Standards Institute) standard. MySQL can execute queries against a database ,retrieve data from a database, insert records in a database, update records in a database, delete records from a database, create new databases , create new tables in a database , create stored procedures in a database, create views in a database, set permissions on tables, procedures, and views.

* **Software Configuration:**

**Web based application**

* + Operating system : Windows XP/7/10
  + IDE : Sublime text
  + Front-end : Bootstrap framework
  + Server side programming : PHP
  + Client side scripting : Ajax, Javascript
  + Database : MySQLi
  + Server : Apache server
  + Tool Kit : Google Chrome developer tools

Other libraries : JQuery

* **System Design**

The System design is the phase where the system functionality is outlined using the specifications provided in the previous chapters. This system design elaborates the proposed system with the flow diagrams and the block diagrams of the technique used. By using the system design the quality of the software can be predicted. Thus it is one among the critical phases to be carried during the implementation of any project.

The modules and their specifications are also discussed in this chapter. This phase builds the communication between the requirement and the implementation phase.

**DESIGN REQUIREMENTS**

The Requirements are determined in light of the craved objectives, the outline objectives are utilized to distinguish the nature of the framework created or executed. The majority of the objectives that are really planned falls under non-functional prerequisites and even the application area take after the same arrangement of criteria. The System outline is the most basic component influencing the nature of the product and has real effect on the later stage, particularly on the testing and upkeep. This is used to identify the modules of the system, specifications and detailed development of these modules that specifies the interaction with each other to produce the desired output. Once the system has been designed all major types, modules and expected output, the specifications for implementing are decided.

This deals with data flow diagram, detailed flow graph, requirement analysis, and the design process of the front end and back end design of the student information management system.

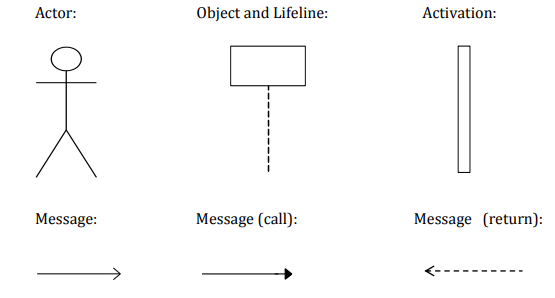
Database Design

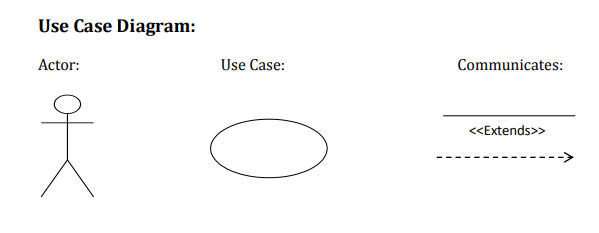
It is fair to say that database play a critical role in almost all areas where computers are used, including Business, electronic commerce, engineering, medicine, law, education, and library science. A database is collection of a related data. A database has the following implicit properties:

**Database Management System (DBMS)** is a collection of programs that enables users to create and maintain a database. DBMS is a general purpose software system that facilitates the process of defining, constructing, manipulating, and sharing database among various users and applications. Defining a database involves the specifying the data types, structures, and constraints of the data to be stored in the database. The database definition or descriptive information is also stored in the database in the form of dictionary; it is called Meta data constructing the database is the process of storing the data on the storage medium that is controlled by the DBMS.

* Manipulating a database includes functions such as querying the database to retrieve specific data, updating the database to reflect in the mini-world, and generating reports from the data. Sharing a database allows a multiple users and programs to access the database simultaneously.
* Application program accesses the database by sending queries or request for data to the DBMS [5]. A query typically causes some data to be retrieved; a transaction may cause some data to be read and some data to be written into the database.
* A database represents some aspect of the real world, sometimes called the mini-world or the Universe OfDiscourse (UOD) changes to the mini world are reflected in the database.
* A database is a logically coherent collection of data with some inherent meaning. A random assortment ofdata cannot correctly be referred to as a database.
* A database is designed, built, and populated with data for a specific purpose. It is an intended group ofusers and some preconceived application which these users are interested.
* **Design Notations**

Sequence Diagram:





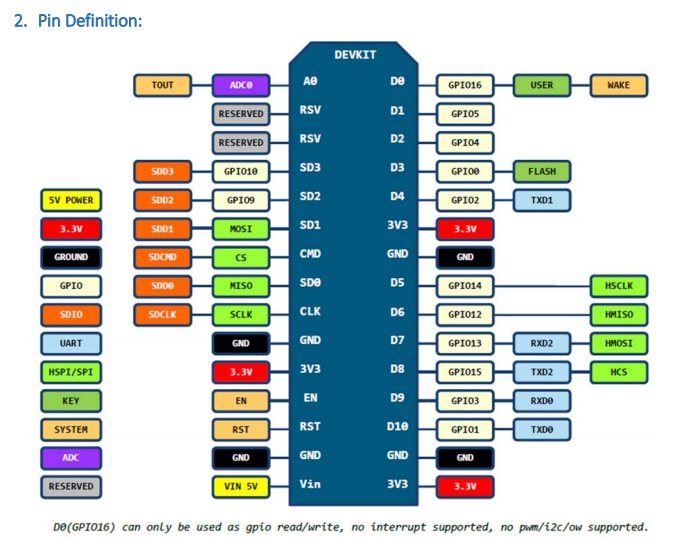
**NODEMCU**



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

Specification:

1. • Voltage:3.3V.
2. • Wi-Fi Direct (P2P), soft-AP.
3. • Current consumption: 10uA~170mA.
4. • Flash memory attachable: 16MB max (512K normal).
5. • Integrated TCP/IP protocol stack.
6. • Processor: Tensilica L106 32-bit.
7. • Processor speed: 80~160MHz.
8. • RAM: 32K + 80K.
9. • GPIOs: 17 (multiplexed with other functions).
10. • Analog to Digital: 1 input with 1024 step resolution.
11. • +19.5dBm output power in 802.11b mode
12. • 802.11 support: b/g/n.
13. • Maximum concurrent TCP connections: 5.

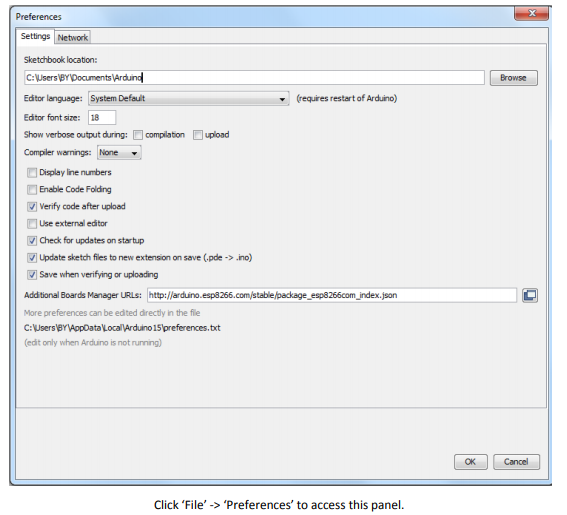


Using Arduino IDE

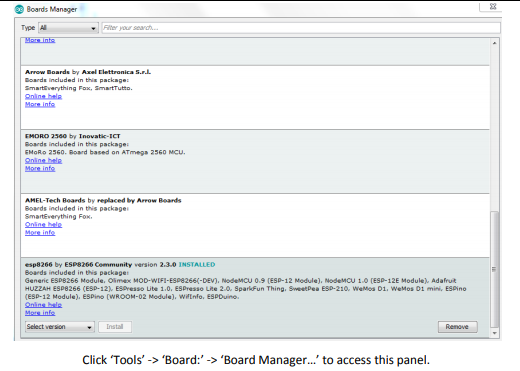
The most basic way to use the ESP8266 module is to use serial commands, as the chip is basically a WiFi/Serial transceiver. However, this is not convenient. What we recommend is using the very cool Arduino ESP8266 project, which is a modified version of the Arduino IDE that you need to install on your computer. This makes it very convenient to use the ESP8266 chip as we will be using the well-known Arduino IDE. Following the below step to install ESP8266 library to work in Arduino IDE environment.

Install the Arduino IDE 1.6.4 or greater Download Arduino IDE from Arduino.cc (1.6.4 or greater) - don't use 1.6.2 or lower version! You can use your existing IDE if you have already installed it. You can also try downloading the ready-to-go package from the ESP8266-Arduino project, if the proxy is giving you problems.

Install the ESP8266 Board Package Enter http://arduino.esp8266.com/stable/package\_esp8266com\_index.json into Additional Board Manager URLs field in the Arduino v1.6.4+ preferences.

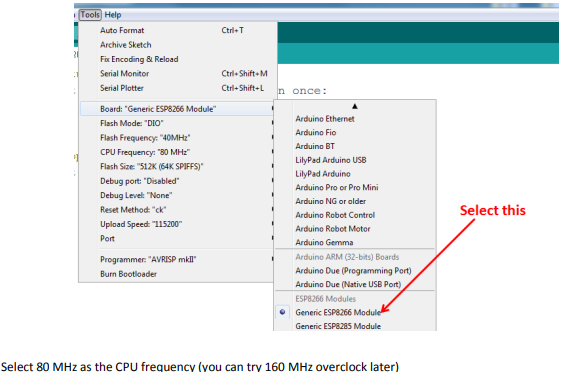


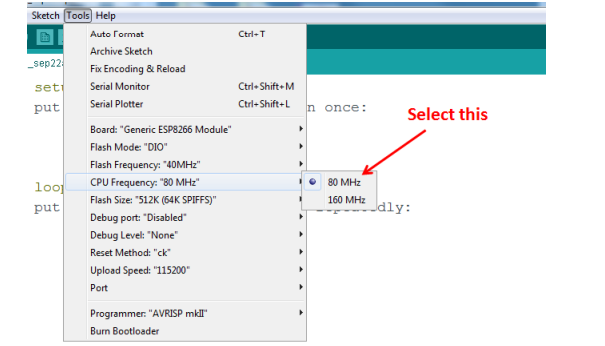
Next, use the Board manager to install the ESP8266 package.



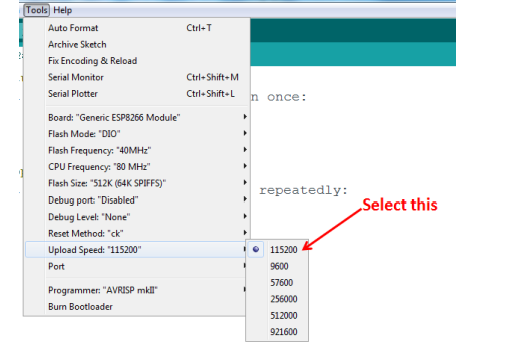
Scroll down to ‘esp8266 by ESP8266 Community ’ and click “Install” button to install the ESP8266 library package. Once installation completed, close and re-open Arduino IDE for ESP8266 library to take effect.

Setup ESP8266 Support When you've restarted Arduino IDE, select ‘Generic ESP8266 Module’ from the ‘Tools’ -> ‘Board:’ dropdown menu.

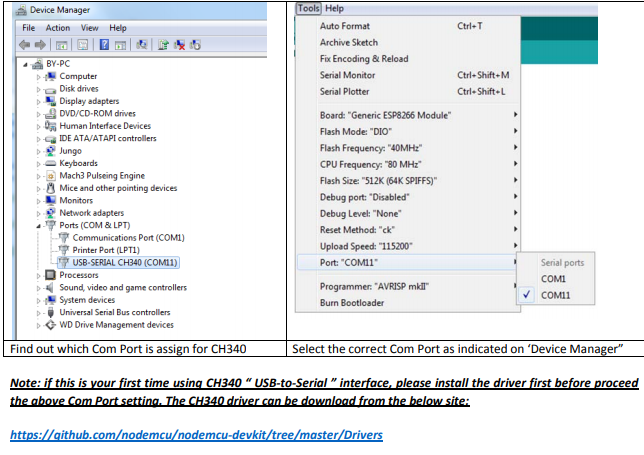




Select ‘115200’ baud upload speed is a good place to start - later on you can try higher speeds but 115200 is a good safe place to start.



Go to your Windows ‘Device Manager’ to find out which Com Port ‘USB-Serial CH340’ is assigned to. Select the matching COM/serial port for your CH340 USB-Serial interface.



Blink Test

We'll begin with the simple blink test.

Enter this into the sketch window (and save since you'll have to).

Connect a LED as shown in

void setup()

{

pinMode(5, OUTPUT); // GPIO05, Digital Pin D1

}

void loop()

{

digitalWrite(5, HIGH);

delay(900);

digitalWrite(5, LOW);

delay(500);

}

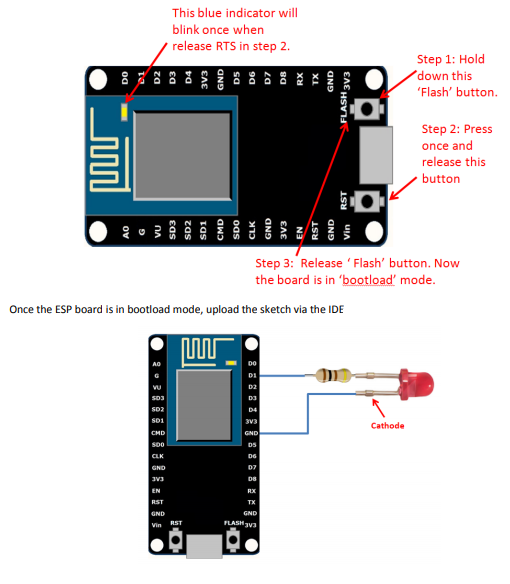
Now you'll need to put the board into bootload mode. You'll have to do this before each upload. There is no timeout for bootload mode, so you don't have to rush!

• Hold down the ‘Flash’ button.

• While holding down ‘ Flash’, press the ‘RST’ button.

• Release ‘RST’, then release ‘Flash

• When you release the ‘RST’ button, the blue indication will blink once, this means its ready to bootload.



**GRAPHIC LCD DISPLAY WITH I2C**

**ALPHA-NUMERIC LCD DISPLAY**

A liquid crystal display is a thin, flat electronic visual display that uses the light modulating properties of liquid crystals (LCs). LCs does not emit light directly.

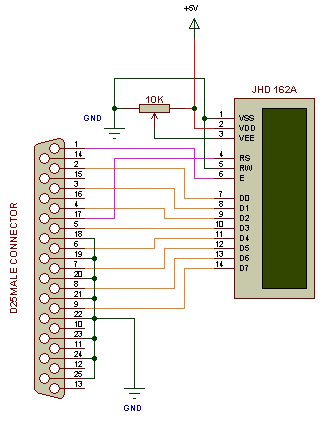
In liquid crystal displays (LCDs) of liquid crystal technology is the most common applications. An advanced VGA computer screen from the pervasive wrist watch and pocket calculator, this type of display has evolved into an important and ambidextrous interface.

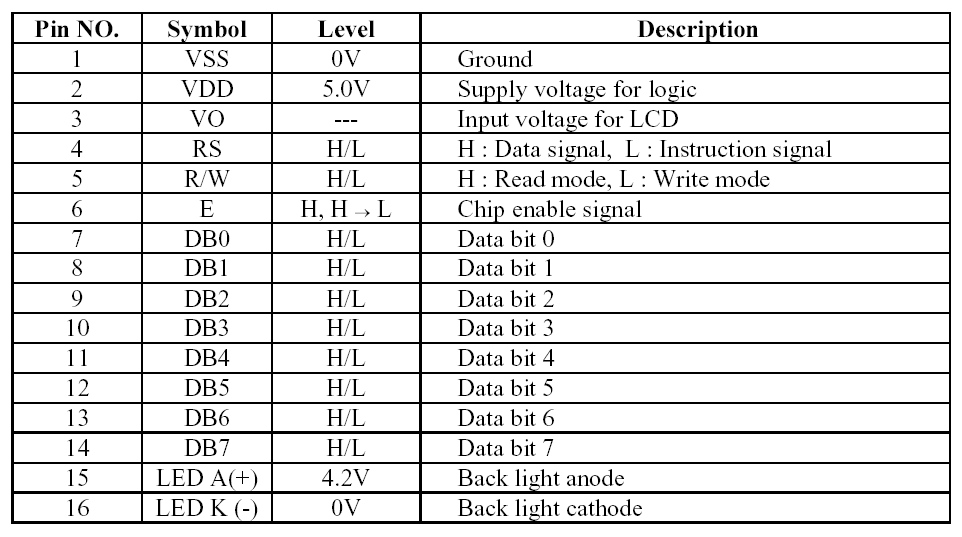
Consist of a liquid crystal display, an array of tiny segments (called pixels) and to present the information that can be manipulated. This basic common idea is to all displays, alienate from simple calculators to a full color LCD television.

The primary factor was size, an LCD consisting of primarily with some liquid crystal material between them of two glass plates. There is no bulk amount picture tube. This gives LCDs practical for applications where size (as well as weight) is necessary.

In general, LCDs uses very low power than the cathode-ray tube (CRT) counterparts. Many LCDs are ruminative, means that they use only atmosphere light to illuminate the display. Even displays that do consume much less power than CRT devices require an external light source (i.e. computer displays).





****



Make sure that 5V and GND lines are properly connected otherwise you may end up in damaging parallel port.

If you want backlight than connect pin 15 of LCD to 5V and pin 16 of LCD to GND. By adjusting 10k resistor make pin 3 of LCD at 0V. If connections are proper you will see this after power on.

The Graphic LCD display used in this project is 128x64 pixels, where it has 128 columns and 64 rows. Supply voltage is 5V matching the voltage for most microcontrollers. The LCD controller is Samsung KS0108B. JHD12864J module uses 8-bit data bus (DB0 – DB7). Nevertheless, it is a straight forward module comparing to other LCD series like T6963C. JHD12864J is split logically in half with controller #1 (CS1) driving the left half of the display, and controller #2 (CS2) driving the right half. These two portions map directly to the physical display area.

# 

RS is equivalent to PIN D/I as stated on JHD12864J data sheet. It controls data or

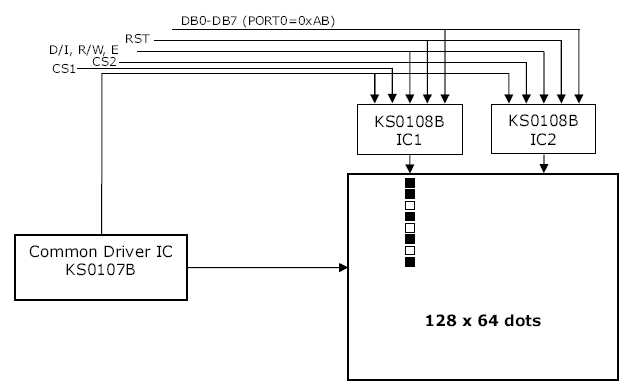
command action (D/I=LOW \_ command; D/I=HIGH \_ data). Horizontal pixel addressed by Y address counter (0-63). The nomenclature is not the same as our convention of Cartesian coordinate system (x-y) learned in secondary school. The Y address indicates the column position in the horizontal direction. Why only 64 pixels but not 128 pixels? Because the LCD is spitted logically in half with controller #1 (CS1) driving the left half of the display, and

controller #2 (CS2) driving the right half. We need to handle each half individually.

The term Page refers to 8-pixel vertically. There are 8 pages ranging from 0 to 7,

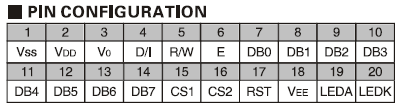
thus matching a vertical matrix size of 64 pixels. Refer to Figure 8.1.2 for

illustration.



R/W controls data READ/WRITE (R/W=LOW \_ write; R/W=HIGH \_ read). The

reason of writing bytes to the LCD is obvious: we need to display something on the LCD. However, being capable of reading from the module is also important because it is only possible to write to a whole Page in 8-bit format. As an example, we want to display a single pixel at the 10th column on 3rd pixel down the top Page where there is an existing byte 0xAB. If we simply output 0x40 (0b0000 0100), the byte pattern 0xAB would be erased. One possible way is to perform a data read first, store the byte in background to a temporary variable, and do a bitwise OR operation with 0x40. This makes a new byte value of 0xAF.



Besides D/I, and R/W pins, there are other control pins to take care including CS1,

CS2, E, and RST pins. Direct low-level access and signal timing requirement will be taken care by hardware dependent functions. The application interface function (API) is hardware-independent. This idea is to allow easy porting to other microcontrollers since we only have to re-write the hardware interface for other microcontrollers or compliers.

History

When connecting multiple devices to a microcontroller, the address and data lines of each devices were conventionally connected individually. This would take up precious pins on the microcontroller, result in a lot of traces on the PCB, and require more components to connect everything together. This made these systems expensive to produce and susceptible to interference and noise.

To solve this problem, Philips developed Inter-IC bus, or I2C, in the 1980s. I2C is a low-bandwidth, short distance protocol for on board communications. All devices are connected through two wires: serial data (SDA) and serial clock (SCL).

|  |
| --- |
| Sample I2C Implementation |
| *Figure 1: Sample I2C Implementation.  Regardless of how many slave units are attached to the I2C bus, there are only two signals connected to all of them. Consequently, there is additional overhead because an addressing mechanism is required for the master device to communicate with a specific slave device.* |

Because all communication takes place on only two wires, all devices must have a unique address to identify it on the bus. Slave devices have a predefinied address, but the lower bits of the address can be assigned to allow for multiples of the same devices on the bus.

[Top](http://www.totalphase.com/docs/articles/article01/#top#top) [Top](http://www.totalphase.com/docs/articles/article01/#top#top)

Theory of Operation

I2C has a master/slave protocol. The master initiates the communication. The sequence of events are:

1. The Master device issues a start condition. This condition informs all the slave devices to listen on the serial data line for instructions.
2. The Master device sends the address of the target slave device and a read/write flag.
3. The Slave device with the matching address responds with an acknowledgement signal.
4. Communcation proceeds between the Master and the Slave on the data bus. Both the master and slave can receive or transmit data depending on whether the communcation is a read or write. The transmitter sends 8-bits of data to the receiver which replies with a 1-bit acknowledgement.
5. When the communication is complete, the master issues a stop condition indicating that everything is done.

|  |
| --- |
| I2C Communication Protocol |
| *Figure 2: I2CCommunication Protocol  Since there are only two wires, this protocol includes the extra overhead of an addressing mechanism and an acknowledgement mechanism* |

[Top](http://www.totalphase.com/docs/articles/article01/#top#top) [Top](http://www.totalphase.com/docs/articles/article01/#top#top)

Features

I2C has many features other important features worth mentioning. It supports multiple data speeds: standard (100 kbps), fast (400 kbps) and high speed (3.4 Mbps) communications.

Other features include:

* Built in collision detection,
* 10-bit Addressing,
* Mutli-master support,
* Data broadcast (general call).

For more information about other features, take a look at the references at the end of this article.

[Top](http://www.totalphase.com/docs/articles/article01/#top#top) [Top](http://www.totalphase.com/docs/articles/article01/#top#top)

Benefits and Drawbacks

Since only two wires are required, I2C is well suited for boards with many devices connected on the bus. This helps reduce the cost and complexity of the circuit as additional devices are added to the system.

Due to the presence of only two wires, there is additional complexity in handling the overhead of addressing and acknowledgments. This can be inefficient in simple configurations and a direct-link interface such as SPI might be preferred.

## I2C Glossary

[Print Version](http://www.totalphase.com/docs/articles/article02/?print) [Print Version](http://www.totalphase.com/docs/articles/article02/?print) | [Top](http://www.totalphase.com/docs/articles/) [Back](http://www.totalphase.com/docs/articles/)

**Master**

The device which initiates a transfer, generates clock signals and terminates a transfer.

**Slave**

The device addressed by a master.

**Transmitter**

The device which sends the data to the bus.

**Receiver**

The device which receives data from the bus.

**Standard Mode**

Maximum bit rate of 100 kbps.

**Fast Mode**

Maximum bit rate of 400 kbps.

**High Speed Mode**

Maximum bit rate of 3.4 Mbps.

**SDA**

Serial Data line. The signal used to transfer data between the transmitter and the receiver.

**SCL**

Serial Clock line. The signal used to synchronize communication between the master and the slave.

**Multi-Master**

More than one master can attempt to control the bus at the same time without corrupting the message.

**Arbitration**

Procedure to ensure that, if mroe than one master simultaneously tries to control the bus, only one is allowed to do so and the winning message is not corrupted.

**Synchronization**

Procedure to synchronize the clock signals of two or more devices.

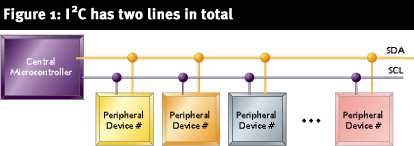
**Slave Address**

Each slave has a unique address to identify it on the bus. These addresses are pre-defined, but the least significant bits can be set by the user to allow for multiples of the same device. In standard I2C, the slave address is 7-bits, but the protocol has been extended to also support 10-bit addresses.

**Inside the box**

I2C is appropriate for interfacing to devices on a single board, and can be stretched across multiple boards inside a closed system, but not much further. An example is a host CPU on a main embedded board using I2C to communicate with user interface devices located on a separate front panel board. A second example is SDRAM DIMMs, which can feature an I2C EEPROM containing parameters needed to correctly configure a memory controller for that module.

I2C is a two-wire serial bus, as shown in Figure 1. There's no need for chip select or arbitration logic, making it cheap and simple to implement in hardware.



The two I2C signals are serial data (SDA) and serial clock (SCL). Together, these signals make it possible to support serial transmission of 8-bit bytes of data-7-bit device addresses plus control bits-over the two-wire serial bus. The device that initiates a transaction on the I2C bus is termed the master. The master normally controls the clock signal. A device being addressed by the master is called a slave.

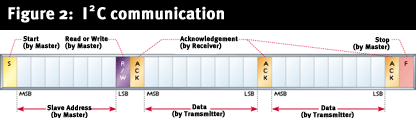
In a bind, an I2C slave can hold off the master in the middle of a transaction using what's called clock stretching (the slave keeps SCL pulled low until it's ready to continue). Most I2C slave devices don't use this feature, but every master should support it.

The I2C protocol supports multiple masters, but most system designs include only one. There may be one or more slaves on the bus. Both masters and slaves can receive and transmit data bytes.

Each I2C-compatible hardware slave device comes with a predefined device address, the lower bits of which may be configurable at the board level. The master transmits the device address of the intended slave at the beginning of every transaction. Each slave is responsible for monitoring the bus and responding only to its own address. This addressing scheme limits the number of identical slave devices that can exist on an I2C bus without contention, with the limit set by the number of user-configurable address bits (typically two bits, allowing up to four identical devices).

**Communication**

As you can see in Figure 2, the master begins the communication by issuing the start condition (S). The master continues by sending a unique 7-bit slave device address, with the most significant bit (MSB) first. The eighth bit after the start, read/not-write (), specifies whether the slave is now to receive (0) or to transmit (1). This is followed by an ACK bit issued by the receiver, acknowledging receipt of the previous byte. Then the transmitter (slave or master, as indicated by the bit) transmits a byte of data starting with the MSB. At the end of the byte, the receiver (whether master or slave) issues a new ACK bit. This 9-bit pattern is repeated if more bytes need to be transmitted.



In a write transaction (slave receiving), when the master is done transmitting all of the data bytes it wants to send, it monitors the last ACK and then issues the stop condition (P). In a read transaction (slave transmitting), the master does not acknowledge the final byte it receives. This tells the slave that its transmission is done. The master then issues the stop condition.

**A simple bus**

As we've seen, the I2C signaling protocol provides device addressing, a read/write flag, and a simple acknowledgement mechanism. There are a few more elements to the I2C protocol, such as general call (broadcast) and 10-bit extended addressing. Beyond that, each device defines its own command interface or address-indexing scheme.

Standard I2C devices operate up to 100Kbps, while fast-mode devices operate at up to 400Kbps. A 1998 revision of the I2C specification (v. 2.0) added a high-speed mode running at up to 3.4Mbps. Most of the I2C devices available today support 400Kbps operation. Higher-speed operation may allow I2C to keep up with the rising demand for bandwidth in multimedia and other applications.

Most often, the I2C master is the CPU or microcontroller in the system. Some microcontrollers even feature hardware to implement the I2C protocol. You can also build an all-software implementation using a pair of general-purpose I/O pins (single master implementations only).

Since the I2C master controls transaction timing, the bus protocol doesn't impose any real-time constraints on the CPU beyond those of the application. (This is in contrast with other serial buses that are timeslot-based and, therefore, take their service overhead even when no real communication is taking place.)

**The elegance of I2C**

I2C offers good support for communication with on-board devices that are accessed on an occasional basis. I2C's competitive advantage over other low-speed short-distance communication schemes is that its cost and complexity don't scale up with the number of devices on the bus. On the other hand, the complexity of the supporting I2C software components can be significantly higher than that of several competing schemes (SPI and MicroWire, to name two) in a very simple configuration. With its built-in addressing scheme and straightforward means to transfer strings of bytes, I2C is an elegant, minimalist solution

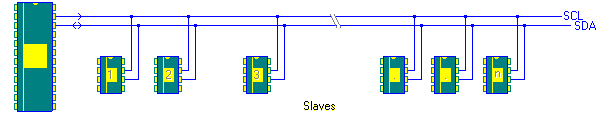
#### The I2C Bus Protocol

The I2C bus physically consists of 2 active wires and a ground connection. The active wires, called SDA and SCL, are both bi-directional. SDA is the Serial DAta line, and SCL is the Serial CLock line.

Every device hooked up to the bus has its own unique address, no matter whether it is an MCU, LCD driver, memory, or ASIC. Each of these chips can act as a receiver and/or transmitter, depending on the functionality. Obviously, an LCD driver is only a receiver, while a memory or I/O chip can be both transmitter and receiver.

The I2C bus is a multi-master bus. This means that more than one IC capable of initiating a data transfer can be connected to it. The I2C protocol specification states that the IC that initiates a data transfer on the bus is considered the *Bus Master*. Consequently, at that time, all the other ICs are regarded to be *Bus Slaves*.

As bus masters are generally microcontrollers, let's take a look at a general 'inter-IC chat' on the bus. Lets consider the following setup and assume the MCU wants to send data to one of its slaves (also see [here](http://www.esacademy.com/faq/i2c/busevents/i2csendb.htm) for more information; click [here](http://www.esacademy.com/faq/i2c/busevents/i2crecei.htm) for information on how to receive data from a slave).



First, the MCU will issue a [*START*](http://www.esacademy.com/faq/i2c/busevents/i2cstast.htm) condition. This acts as an 'Attention' signal to all of the connected devices. All ICs on the bus will listen to the bus for incoming data.

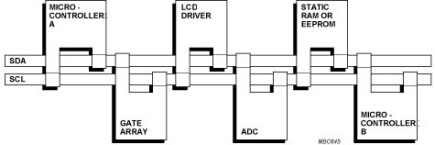
Then the MCU sends the [*ADDRESS*](http://www.esacademy.com/faq/i2c/busevents/i2csendb.htm) of the device it wants to access, along with an indication whether the access is a Read or Write operation (Write in our example). Having received the address, all IC's will compare it with their own address. If it doesn't match, they simply wait until the bus is released by the stop condition (see below). If the address matches, however, the chip will produce a response called the [*ACKNOWLEDGE*](http://www.esacademy.com/faq/i2c/busevents/i2cgetak.htm) signal.

Once the MCU receives the acknowledge, it can start transmitting or receiving *DATA*. In our case, the MCU will transmit data. When all is done, the MCU will issue the [*STOP*](http://www.esacademy.com/faq/i2c/busevents/i2cstast.htm) condition. This is a signal that the bus has been released and that the connected ICs may expect another transmission to start any moment.

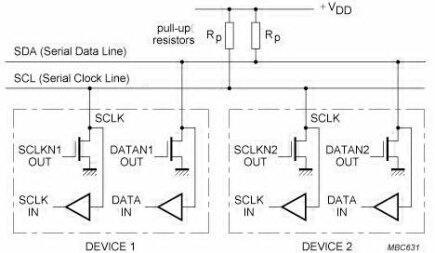
We have had several states on the bus in our example: [*START*](http://www.esacademy.com/faq/i2c/busevents/i2cstast.htm), [*ADDRESS*](http://www.esacademy.com/faq/i2c/busevents/i2csendb.htm), [*ACKNOWLEDGE*](http://www.esacademy.com/faq/i2c/busevents/i2cgetak.htm), *DATA ,* [*STOP*](http://www.esacademy.com/faq/i2c/busevents/i2cstast.htm). These are all unique conditions on the bus. Before we take a closer look at these bus conditions we need to understand a bit about the physical structure and hardware of the bus.

|  |  |  |
| --- | --- | --- |
| The I2C Bus Hardware Structure As explained earlier, the bus physically consists of 2 active wires called **SDA** (data) and **SCL** (clock), and a ground connection.  Both SDA and SCL are initially bi-directional. This means that in a particular device, these lines can be driven by the IC itself or from an external device. In order to achieve this functionality, these signals use open collector or open drain outputs (depending on the technology). | | |
|  | http://www.esacademy.com/faq/i2c/general/i2chardw_files/g01hardw.gif | The bus interface is built around an input buffer and an open drain or open collector transistor. When the bus is IDLE, the bus lines are in the logic HIGH state (note that external pull-up resistors are necessary for this which is easily forgotten). To put a signal on the bus, the chip drives its output transistor, thus pulling the bus to a LOW level. The "pull-up resistor" in the devices as seen in the figure is actually a small current source or even non-existent. | |
| The nice thing about this concept is that it has a "built-in" bus mastering technique. If the bus is "occupied" by a chip that is sending a 0, then all other chips lose their right to access the bus. More will be explained about this in the section about [bus arbitration](http://www.esacademy.com/faq/i2c/general/i2carbit.htm).  However, the open-collector technique has a drawback, too. If you have a long bus, this will have a serious effect on the speed you can obtain. Long lines present a capacitive load for the output drivers. Since the pull-up is passive, you are facing an RC constant which will reflect on the shapes of the signals. The higher this RC constant, the slower you can go. This is due to the effect that it influences the slew rate of the edges on the I2C bus. At a certain point, the ICs will not be able to distinguish clearly between a logic 1 and 0.  http://www.esacademy.com/faq/i2c/general/i2chardw_files/g02hardw.gif  What's more is that you can get reflections at high speed. This can be so bad that "ghost signals" disturb your transmission and corrupt the data you transmit. Not even Schmitt triggers at the IC's inputs will be able to eliminate this effect.  http://www.esacademy.com/faq/i2c/general/i2chardw_files/g03hardw.gif  Therefore some strict [electrical specifications](http://www.esacademy.com/faq/i2c/general/i2cspecver.htm) have been put together.  To overcome this problem, Philips has developed an active I2C terminator. This device consists of a twin charge pump and you can look at it as a dynamic resistor. The moment the state changes, it provides a large current (low dynamic resistance) to the bus. In doing so it can charge the parasitic capacitor very quickly. Once the voltage has risen above a certain level, the high current mode cuts out and the output current drops sharply.  Take a look at the following figure. As long as the bus is kept low (transistor C is on), the charge pump is disabled because the gate of transistor B is kept low by transistor A. | | | |
| http://www.esacademy.com/faq/i2c/general/i2chardw_files/g04hardw.gif  As soon as the chip releases the bus (A and C turn off), the capacitor will start charging, drawing current trough all four of the resistors (1 - 4). The voltage drop over resistor 2 will cause the transistor B to turn on, shorting out resistor 3. Since resistor 3 is a relatively low value, the current will rise. At a certain point in time, the drop between transistor B's gate and source will not be big enough to keep it switched on. It will then switch off and the charge injection will stop. At that time, only the external pull-up resistor remains to overcome the charge leakage on the bus. | | | |
|  | Please note that this is a simple explanation. The actual device implements more circuitry, e.g. to prevent "overcharging" if another chip is still pulling the bus low.  This device can come in handy if you need to overcome several meters of I2C bus length. | |  |

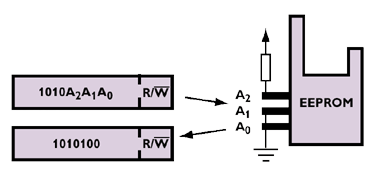
### How the I²C-Bus works



Any I²C device can be attached to an I²C-bus and every device can talk with any master, passing information back and forth. There needs to be at least one master (e.g., microcontroller or DSP) on the bus but there can be more than one, with all masters having equal priority. Devices can be easily added to and removed from the I²C-bus.

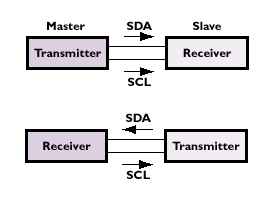


Total bus capacitance needs to be less than 400 pF (e.g., about 20-30 devices or 10 m of trace) to respect rise and fall time requirements. Each device must be able to drive up to 3 mA for a logic low level of 0.4 mA on an open drain bus with pull-ups in the range of 2 K to 10 K ohms. Bi-directional I²C bus buffers are available that isolate the capacitance on different legs of the bus, allowing larger (e.g., 2000 pF) and longer (e.g., 2000 m) buses.

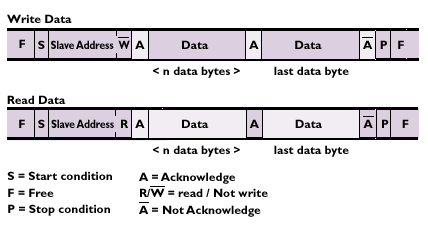


Each device has a unique 7-bit I²C address so that the master knows specifically whom they are communicating with. Typically the four most significant bits are fixed and assigned to specific categories of devices (e.g. 1010 is assigned to serial EEPROMs). The three less significant bits (e.g., A2, A1 and A0) are programmable through hardware address pins allowing up to eight different I²C address combinations and therefore allowing up to eight of that type of device to operate on the same I²C-bus. These pins are held high to VCC (1) or held low to GND (0). 7-bit addressing allows up to 128 devices on the same bus but some of these addresses are reserved for special commands so the practical limit is around 120.

### I²C-Bus Terminology



**Transmitter** - the device that sends data to the bus. A transmitter can either be a device that puts data on the bus on its own accord (a 'master-transmitter'), or in response to a request from the master (a 'slave-transmitter').  
**Receiver** - the device that receives data from the bus. A receiver can either be a device that receives data on its own request (a 'master-receiver), or in response to a request from the master (a 'slave-receiver).  
**Master** - the component that initializes a transfer (Start command), generates the clock (SCL) signal and terminates the transfer (Stop command). A master can be either a transmitter or a receiver.  
**Slave** - the device addressed by the master. A slave can be either receiver or transmitter.  
**Multi-master** - the ability for more than one master to co-exist on the bus at the same time without collision or data loss. Typically "bit-banged" software implemented masters are not multi-master capable. Parallel to I²C bus controllers provide an easy way to add a multi-master hardware I²C port to DSPs and ASICs.  
**Arbitration** - the prearranged procedure that authorizes only one master at a time to take control of the bus.  
**Synchronization** - the prearranged procedure that synchronizes the clock signals provided by two or more masters.  
**SDA** - data signal line (**S**erial **DA**ta)  
**SCL** - clock signal line (**S**erial **CL**ock)

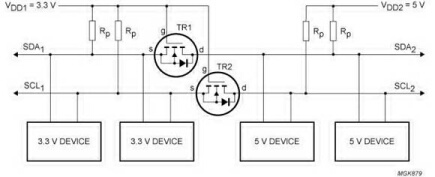


I²C address of the targeted device is sent in the 1st byte and the lest significant bit of this initial byte indicates if the master is going to send (write) or receive (read) data from the receiver, called the slave device. Each transmission sequence must begin with the Start condition and end with the Stop or ReStart condition. If there are two masters on the same I²C-bus, there is an arbitration procedure if both try to take control of the bus at the same time by generating the Start command at the same time. Once a master (e.g., microcontroller) has control of the bus, no other master can take control until the first master sends a Stop condition and places the bus in an idle state.

### Terminology for Bus Transfer

**F (FREE)** - the bus is free or idle; the data line SDA and the SCL clock are both in the high state.  
**S (START) or R (RESTART)** - data transfer begins with a Start condition. The level of the SDA data line changes from high to low, while the SCL clock line remains high. When this occurs, the bus becomes 'busy'.  
**C (CHANGE)** - while the SCL clock line is low, the data bit to be transferred can be applied to the SDA data line by a transmitter. During this time, SDA may change its state, as long as the SCL line remains low.  
**D (DATA)** - a high or low bit of information on the SDA data line is valid during the high level of the SCL clock line. This level must be kept stable during the entire time that the clock remains high to avoid misinterpretation as a Start or Stop condition.  
**P (STOP)** - data transfer is terminated by a Stop condition. This occurs when the level on the SDA data line passes from the low state to the high state, while the SCL clock line remains high. When the data transfer has been terminated, the bus is free once again.

### Level-shifting I²C



As new I²C devices are operating at various voltage levels, Philips Semiconductors developed a fully bi-directional data transfer circuit for I²C-devices operating from different supply voltages for almost no additional design-in effort or cost. The simple addition of two low-cost transistors, placed between the different voltage level sections of the I²C-bus, separates and transfers the logic voltage levels of the bus lines on either side of the level-shifter. This set-up also allows the level-shifter to be used to isolate a section of powered-down devices from the bus, allowing powered I²C devices to operate normally. These level-shifting functions can also be done with low cost level shifting bus buffer or GTL translator devices. The I²C-bus specification has been extended for devices operating below 2.7 V to ensure the I²C-bus will remain the serial bus of choice for future systems into the next century.

### High-speed mode I²C-bus

Developments in high-speed serial RAMs and mixed technology telecom systems have created a demand for buses capable of operating at high speeds and with a variety of supply voltages. The I²C-bus High-speed mode (3.4 MHz) meets these needs without sacrificing its compatibility with existing Standard and Fast-mode devices or its low-cost simplicity - no special logic levels, timing or drive capability are needed. A purpose-designed bridge in a High-speed mode master allows for bi-directional communications between Fast- and Standard-mode devices within a single I²C-bus system and, if required, this master can also perform bi-directional level shifting, supporting a variety of different voltage devices. The high-speed mode is normally found only on LCD display, high bit count ADC or high capacity EEPROM devices due to the need to transfer large amount of data. Most maintenance and control applications, the typical reason to use the I²C-bus, continue to operate at 100 kHz.

### 10-bit I²C addressing

10-bit addressing allows the use of up to 1024 additional addresses to prevent problems with the allocation of slave addresses as the number of I²C devices rapidly expands. It does not change the format for addresses defined in the I²C-bus specification, using addresses reserved in the existing specification. 10-bit addressing does not affect the existing 7-bit addressing; allowing devices with 7-bit or 10-bit addresses to be connected to the same I²C-bus, and both types of devices can be used in Standard-, Fast- or High-speed mode systems. Most all I²C devices continue to use 7-bit addressing however as designers work around addressing concerns by using multiple I²C bus ports or multiplexers/switches to allow smaller device populations on the active bus.

### I²C vs SMBus

Developed by Intel in the mid-1990s, the System Management Bus, also known as SMBus, is a popular derivative of the I²C-bus that is, in most cases, compatible with I²C. Both buses use a two-wire communication scheme and have addressable slaves. The SMBus is limited to a maximum data transfer rate of only 100 kbps, requiring special handling in systems that also use the higher transfer rates available with I²C. Other differences include timeout and minimum clock speed, voltage levels, pull-up resistor values, and current levels. New I²C devices are starting to incorporate SMBus features such as timeout that can be turned on and off depending on the application requirements.

#### History of the I2C Bus

The I2C bus was developed in the early 1980's by Philips Semiconductors. Its original purpose was to provide an easy way to connect a CPU to peripheral chips in a TV-set.

Peripheral devices in embedded systems are often connected to the MCU as memory-mapped I/O devices, using the microcontroller's parallel address and data bus. This results in lots of wiring on the PCB's to route the address and data lines, not to mention a number of address decoders and glue logic to connect everything. In mass production items such as TV-sets, VCR's and audio equipment, this is not acceptable. In these appliances, every component that can be saved means increased profitability for the manufacturer and more affordable products for the end customer. Furthermore, lots of control lines implies that the systems is more susceptible to disturbances by Electromagnetic Interference (EMI) and Electrostatic Discharge (ESD).

The research done by Philips Labs in Eindhoven (The Netherlands) to overcome these problems resulted in a 2-wire communication bus called the I2C bus. I2C is an acronym for Inter-IC bus. Its name literally explains its purpose: to provide a communication link between Integrated Circuits.

Today, the I2C bus is used in many other application fields than just audio and video equipment. The bus is generally accepted in the industry as a de-facto standard. The I2C bus has been adopted by several leading chip manufacturers like Xicor, ST Microelectronics, Infineon Technologies, Intel, Texas Instruments, Maxim, Atmel, Analog Devices and others.

# POWER SUPPLY

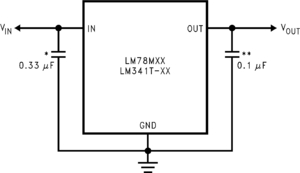
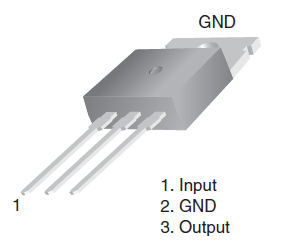
An AC to DC adaptor as been used to get DC input for the mother board. In mother board, we have developed a 5V regulator circuit, which is needed for microcontroller as supply voltage. IR transmitters are also connected to 5V supply, so that they always transmit high signal. LM7805 is used for 5V regulated supply.



**LM7805 Positive Voltage Regulator**

**General Description**

The LM78XX series of three terminal positive regulators are available in the TO-220 package and with several fixed output voltages, making them useful in a wide range of applications. Each type employs internal current limiting, thermal shut down and safe operating area protection, making it essentially indestructible. If adequate heat sinking is provided, they can deliver over 1A output current. Although designed primarily as fixed voltage regulators, these devices can be used with external components to obtain adjustable voltages and currents.



**Features**

* Output current in excess of 0.5A
* No external components
* Internal thermal overload protection
* Internal short circuit current-limiting
* Output transistor safe-area compensation
* Available in TO-220, TO-39, and TO-252 D-PAK packages
* Output voltages of 5V, 12V, and 15V

**TESTING**

The testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

* **Functional testing**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

* **Structural testing**

Structural testing ensures that the entire integrated software structure meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. Structural testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

* **Levels of testing**

**Performance Testing**

The Performance test ensures that the output be produced within the time limits, and the time taken by the system for compiling, giving response to the users and request being send to the system for to retrieve the results.

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

**System Testing**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

* **Testing the project**

Testing was done flowing each testing method mentioned and at last Unit testing was done.

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produces valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results

## IMPLEMENTATION

Once the system has been designed, the next step is to convert the designed one in to actual code, so as to satisfy the user requirements as excepted. If the system is approved to be error free it can be implemented.

When the initial design was done for the system, the department was consulted for acceptance of the design so that further proceedings of the system development can be carried on. After the development of the system a demonstration was given to them about working of the system. The aim of the system illustration was to identify any malfunctioning of the system.

Implementation includes proper training to end-users. The implemented software should be maintained for prolonged running of the software.

Initially the system was run parallel with manual system. The system has been tested with data and has proved to be error-free and user-friendly. Training was given to end -user about the software and its features.

# MAINTENANCE

The term “software maintenance” is used to describe the software engineering activities that occur following delivery of a software product to the customer. The maintenance phase of the software life cycle is the time period in which a software product performs useful work.Maintenance activities involve making enhancement to software products, adapting products to new environments and correcting problems. Software product enhancement may involve providing new functional capabilities, improving user display and modes of interaction, and upgrading external documents. Adaptation of software to a new environment may involve moving the software to a different machine. Problem correction involves modification and revalidation of software to correct errors. The enhancement of this project can be accomplished easily. That is, any new functional capabilities can be added to the project by simply including the new module in the homepage and giving a hyperlink to that module. Adaptation of this project to a new environment is also performed easily.

**Corrective Maintenance**

Even with the best quality assurance activities, it is likely that they customer will uncover defects in the software. Corrective maintenance changes the software to correct defects.

**Adaptive Maintenance**

An activity that modifies the software to properly interface with a changing environment. The system has been modified so that various change include to the new system.

In case of Fund Transfer, adoptive maintenance has been performed, that is in earlier system (character based UNIX system) changes are fixed and if any new changes are to be included, was a difficult task. Now provisions are given so that the user can define various changes. Such as it designed to accommodate the new change in further.

**Enhancement Maintenance**

As software is used, the customer/user will recognize additional functions that will provide benefit. Perceptive maintenance extends the software beyond its original functional requirements.

In the case of visual cryptography ,system can be added new functions such that the user can able to retrieve the information in a user friendly and it will be very helpful for future development.

**system testing AND IMPLEMENTATION**

***TESTING METHODOLOGIES:***

1. **Black box testing**

This is done by testing the system without any knowledge of internal design or code. This typically will be carried by a functional expert than a technical expert. Tests are based on requirements and functionality.

1. **White box testing**

This testing is based on knowledge of the internal logic of an application's code. Tests are based on coverage of code statements, branches, paths, and conditions.

1. **Unit Testing**

Unit Testing is the first level of dynamic testing and is first the responsibility of the developers and then of the testers. Unit testing is performed after the expected test results are met or differences are explainable /acceptable.

1. **Parallel/Audit Testing**

Testing where the user reconciles the output of the new system to the output of the current system to verify the new system performs the operations correctly.

1. **Functional Testing**

Black-box type of testing geared to functional requirements of an application. Testers should perform this type of testing.

1. **Usability Testing**

Testing for 'user-friendliness'. Clearly this is subjective and will depend on the targeted end user or customer. User interviews, surveys, video recording of user sessions, and other techniques can be used. Programmers and testers are usually not appropriate as usability testers.

1. **Incremental Integration Testing**

Continuous testing of an application as new functionality is recommended. This may require various aspects of an application's functionality be independent enough to work separately before all parts of the program are completed, or that test drivers are developed as needed. This type of testing may be performed by programmers or by testers.

1. **Integration Testing**

Upon completion of unit testing, integration testing, which is, black box testing, will begin. The purpose is to ensure distinct components of the application still work in accordance to customer requirements. Test sets will be developed with the express purpose of exercising the interfaces between the components. This activity is to be carried out by the Test Team. Integration test will be termed complete when actual results and expected results are either in line or differences are explainable/acceptable based on user/management input.

1. **System Testing**

Upon completion of integration testing, the Test Team will begin system testing. During system testing, which is a black box test, the complete system is configured in a controlled environment to validate its accuracy and completeness in performing the functions as designed. The system test will simulate production in that it will occur in the "production-like" test environment and test all of the functions of the system that will be required in production. The Test Team will complete the system test.

Prior to the system test, the unit and integration test results will be reviewed by Software Quality Assurance (SQA) to ensure all problems have been resolved. It is important for higher level testing efforts to understand unresolved problems from the lower testing levels. System testing is deemed complete when actual results and expected results are either in line or differences are explainable/acceptable based on client input.

1. **End-to-End Testing**

Similar to system testing, the 'macro' end of the test scale involves testing of a complete application environment in a situation that mimics real-world use, such as interacting with a database, using network communications, or interacting with other hardware, applications, or systems if appropriate.

1. **Regression Testing**

The objective of regression testing is to ensure software remains intact. A baseline set of data and scripts will be maintained and executed to verify changes introduced during the release have not "undone" any previous code. Expected results from the baseline are compared to results of the software being regression tested. All discrepancies will be highlighted and accounted for, before testing proceeds to the next level.

1. **Sanity Testing**

Sanity testing will be performed whenever cursory testing is sufficient to prove the application is functioning according to specifications. This level of testing is a subset of regression testing. It will normally include a set of core tests of basic GUI functionality to demonstrate connectivity to the database, application servers, printers, etc.

1. **Performance Testing**

Although performance testing is described as a part of system testing, it can be regarded as a distinct level of testing. Performance testing will verify the load, volume, and response times as defined by requirements.

1. **Load Testing**

Testing an application under heavy loads, such as the testing of a web site under a range of loads to determine at what point the systems response time degrades or fails.

1. **Installation Testing**

Testing full, partial, or upgrade install/uninstall processes. The installation test for a release will be conducted with the objective of demonstrating production readiness. This test is conducted after the application has been migrated to the client's site. It will encompass the inventory of configuration items (performed by the application's System Administration) and evaluation of data readiness, as well as dynamic tests focused on basic system functionality. When necessary, a sanity test will be performed following the installation testing.

1. **40Security/Penetration Testing**

Testing how well the system protects against unauthorized internal or external access, willful damage, etc. This type of testing may require sophisticated testing techniques.

1. **Recovery/Error Testing**

Testing how well a system recovers from crashes, hardware failures, or other catastrophic problems.

1. **Compatibility Testing**

Testing how well software performs in a particular hardware / software / operating system / network etc. environment.

1. **Comparison Testing**

Testing that compares software weaknesses and strengths to competing products.

1. **Acceptance Testing**

Acceptance testing, which black box is testing, will give the client the opportunity to verify the system functionality and usability prior to the system being moved to production. The acceptance test will be the responsibility of the client; however, it will be conducted with full support from the project team. The Test Team will work with the client to develop the acceptance criteria.

1. **Alpha Testing**

Testing of an application, when development is nearing completion Minor design changes may still be made as a result of such testing. Alpha Testing is typically performed by end users or others, not by programmers or testers.

1. **Beta Testing**

Testing when development and testing are essentially completed and final bugs and problems need to be found before the final release. Beta Testing is typically done by end users or others, not by programmers or testers.

***testing fundamentals***

Software testing is an important element of S/W quality assurance and represents the ultimate review of specification, design and coding. The increasing visibility of S/W as a system element and the costs associated with a S/W failure are motivating forces for well planned, through testing.

Though the test phase is often thought of as separate and distinct from the development effort--first develop, and then test--testing is a concurrent process that provides valuable information for the development team.

There are at least three options for integrating Project

Builder into the test phase:

 Testers do not install Project Builder, use Project Builder functionality to compile and source-control the modules to be tested and hand them off to the testers, whose process remains unchanged.

 the testers import the same project or projects that the developers use.

 Create a project based on the development project but customized for the testers (for example, it does not include support documents, specs, or source), who import it.

A combination of the second and third options works best. Associating the application with a project can be useful during the testing phase, as well. We can create actions to automatically run test scripts or add script types and make them dependent on the modules to test.

**testing objectives**

There are several rules that can serve as testing objectives.

They are

* Testing is a process of executing a program with the intent of finding an error.
* A good test case is one that has a high probability of finding an undiscovered error.
* A successful test is one that uncovers an undiscovered error.

If testing is conducted successfully according to the objectives stated above, it will uncover errors in the software. Also, testing demonstrates that software functions appear to the working according to specification, that performance requirements appear to have been met.

**OBJECT ORIENTED TESTING**

***UNIT TESTING***

Unit testing focuses the verification effort on the smallest unit of S/W design i.e., the module. The unit testing is always white-box oriented and the step can be conducted in parallel for modules.

During unit test, testers can use the same project or projects as the developers, if functional units organize the project, or separate projects have been created for functional units. The project or projects can also be exported, so unit test can take place in a variety of environments and on a variety of platforms.

***Unit test considerations***

The tests that occur as part of unit testing. The module ‘interface’ is tested to ensure that information properly flows into and out of the program unit under test. The ‘local data structures’ are examined to ensure that data stored temporarily maintains its integrity during all steps in an algorithms execution.

‘Boundary Conditions’ are tested to ensure that the module operates properly at boundaries established to limit or restrict processing. All ‘independent paths’ through the control structures are exercised to ensure that all statements in a module have been executed at least once. Finally, all ‘error-handling paths’ are tested.

***Unit test procedures***

Unit testing is considered an equivalent to the coding step. After the source level code has been developed, reviewed and verified for correct syntax, unit test case design begins since a module is not a stand alone program, ‘driver’ and/or ‘stub’ S/W must be developed for each unit test. In most applications, a driver is nothing more than a main program that accepts test case data, passes such data to the module to be tested, and prints the

relevant results. The stubs serve to replace modules that are subordinates called by the modules to be tested. A stub or a dummy stub or a dummy subprogram uses the subordinate modules interface, may do minimal data manipulation, prints verification of entry, and returns. The drivers and scrubs represent overhead i.e., both are S/W that must be written but that is not delivered with the final S/W product. If the drivers and the stub are kept simple, then the overhead is low.

The Unit Test is carried out in this project, and is found successful. The data is flowing correctly to all part of the project.

***INTEGRATION TESTING***

Integration testing is a systematic technique for constructing the program structure while at the same time conducting test to uncover errors associated with interfacing. The objective is to take unit-tested modules and build a program structure that has been dictated by design.

***Top-down integration***

##### This method is an incremental approach to the construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main program module.

##### Importing the test project and setting up the testing environment

The process of importing a test project and setting up a testing environment is the same as the process for importing a project and setting up the environment for development.

##### Adding test scripts and test data to the project

You may need to add some items, such as test scripts, to the project. In addition, you may need to add connection strings to database accounts containing test data.

Remember that you can automate the running of test scripts just as you can automate actions associated with the modules in your application.

***Modifying actions and macros to facilitate testing***

If actions specifying "run with debugging" have not already been provided, you can either modify existing actions to include a debug flag, or create new actions.

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

**6.1 Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**6.1.1 Functional test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

**6.1.2 System Test**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**6.1.3 Performance Test**

The Performance test ensures that the output be produced within the time limits,

and the time taken by the system for compiling, giving response to the users and request being send to the system for to retrieve the results.

# 6.2 Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

**Integration testing for Server Synchronization:**

* Testing the IP Address for to communicate with the other Nodes
* Check the Route status in the Cache Table after the status information is received by the Node
* The Messages are displayed throughout the end of the application

***6.3 Acceptance Testing***

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Acceptance testing for Data Synchronization:**

* The Acknowledgements will be received by the Sender Node after the Packets are received by the Destination Node
* The Route add operation is done only when there is a Route request in need
* The Status of Nodes information is done automatically in the Cache Updation process

**IMPLEMENTATION SOURCE CODE**

**SNAPSHOTS**

CONCLUSION

Health IoT thus helps the hospital authorities to monitor the patients on an ongoing basis as it reminds the patient that the medicines are intimate. So this allows the doctor to have a direct view of his patients. Thus the medication procedures can be shifted from hospital-centric to home-centric. Earlier the medicine box proposed with Ultrasonic sensors, IR sensors and Weight-based sensors.Due to ambient light and unsuitable calibration problems, these sensors may have less reliability. The proposed Magnetic Reed switches are therefore rugged in operation, operated by stepper motors. The stepper motors are controlled by the microcontroller.

**REFERENCES**

[1] Zhibo Pang, Qiang Chen, Lirong Zheng, "A Pervasive and Preventive Healthcare Solution for Medication Noncompliance and Daily Monitoring", 2nd Inte. Symp. on Applied Sciences in Biomedical and Communication Technologies (ISABEL 2009), pp1-6, Nov. 2009.

[2] Fei Hu, Yang Xiao, Qi Hao, "Congestion-aware, loss-resilient biomonitoring sensor networking for mobile health applications", IEEE Journal on Selected Areas in Communications, Vol27, Iss4, May 2009.

[3] Zhibo Pang, Qiang Chen; Lirong Zheng, Elena Dubrova. "An In-home Medication Management Solution Based on Intelligent Packaging and Ubiquitous Sensing". International Conference on Advanced Communications Technology (ICACT). Jan 2013.

[4] C.Rolim, F. Koch, C. Westphall, J. Werner, A. Fracalossi, G. Salvador, "A cloud computing solution for patient's data collection in health care institutions", Second Int. Conf. on eHealth Telemedicine and Social Medicine ETELEMED '10., pp. 95-99, Feb 2010.

[5] W. Zhao, C. Wang, Y. Nakahira, "Medical application on internet of things", IET Int. Conf. on Com. Tech. and Application (ICCTA 2011), pp. 660-665, Oct 2011.

[6] Fei Hu, Yang Xiao, Qi Hao, “Congestion-aware, loss-resilient biomonitoring sensor networking for mobile health applications”, IEEE Journal on Selected Areas in Communications,Vol27, Iss4, May 2009.

[7] Mr.Khyamling A. Parane, Mr.Naveenkumar C.Patil, Mr.Shivananda R. Poojara, Mr.Tejaskumar S.Kamble [Cloud based Intelligent Healthcare Monitoring System, 2014]

[8] Ee-May Fong and Wan-Young Chung [Mobile Cloud-ComputingBased Healthcare Service, 2 December 2013]

[9] Home Based Health Monitoring System Using Android Smart phone, International Journal of Electical,Electronics and Data Communication, Vol-2, Issue-2, Feb-2014, Sushama Pawar, P.W.Kulkarni.

[10] Xiankleber C. Benjamim, Rafael B. Gomes, Aquiles F. Burlamaqui, Luiz MarcosG, “Visual Identication of Medicine Boxes Using Features Matching”, IEEE, 2012

[11] Huai-Kuei Wu, Chi-Ming Wong,et.al.,“A Smart Pill Box with Remind and Consumption Confirmation Functions”,IEEE 4th Global Conference on Consumer Electronics (GCCE), 2015.

[12] B.Abbey, A.Alipour, C.Camp, C.Hofer, “The Smart Pill Box”, in proceedings of the Rehabilitation Engineering and Assistive Technology Society of North America,2012.

[13] Shih-Chang Huang, Hong-Yi Chang, Yu-Chen Jhu, Guan-You Chen, “The Intelligent Pill Box - Design and Implementation”,ICCE-Taiwan2014.